

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

DÉPARTEMENT DU GÉNIE ÉLECTRIQUE ET INFORMATIQUE

Projet de Fin d'Études

Spécialité : Automatique Électronique

Filière : Embedded Smart Power Electronics

Études et mise en œuvre d'évolutions pour la version 2.0 d'un dispositif d'alerte et de localisation connecté

DOCUMENT CONFIDENTIEL

Auteur :

DEVAUD - Emma

Entreprise :

Ido-data

Référent INSA :

DRAGOMIRESCU - Daniela

Responsable du stage :

CREVEAUX - Thomas

Année 2020-2021

Résumé

Ce rapport présente le travail réalisé dans le cadre de mon projet de fin d'études en tant qu'ingénierie électronique/systèmes embarqués au sein de la start-up Ido-data. Il s'articule autour d'un objet connecté : le bracelet Dial, s'inscrivant de ce fait dans les domaines de l'électronique et de l'Internet of Things ou Internet des Objets.

Après avoir exposé le contexte de base concernant le cadre de mon stage ainsi que mon environnement de travail, les objectifs qui m'ont été confiés et les problématiques qui y sont associées, seront détaillés. Ce rapport développe ensuite d'une part, mes réalisations fonctionnelles, Hardware et Software, et d'autre part, les résultats, mon analyse et les limites rencontrées. Une conclusion sur mes accomplissements et une réflexion autour des perspectives futures terminent ce rapport.

Remerciements

Tout d'abord je tiens à remercier chaleureusement **Yannick Tocquet**, **Thomas Creveaux** et **Antonin Carlesso** les trois co-fondateurs de la start-up Ido-data pour m'avoir accueillie en tant que stagiaire, pour leur confiance et pour la bienveillance qu'ils ont eue à mon égard tout au long de ce stage. Cela m'a permis d'évoluer sereinement et de tirer profit de cette expérience professionnelle. Je remercie plus particulièrement **Mr Creveaux**, mon tuteur, pour son soutien et son aide technique tout au long de ces six derniers mois.

Je souhaite remercier également toute l'équipe d'Ido-data : **Rufine Bouteille**, **Florencia Alipaz**, **Cornellia Allagnon**, **Paola Mbia Messi** et **Joanna Charosse** pour leurs conseils, leur bonne humeur et leur gentillesse qui ont largement contribué à créer un environnement de travail agréable et bienveillant.

Par ailleurs je remercie ma tutrice INSA **Daniela Dragomirescu** pour sa disponibilité et son soutien.

Un remerciement particulier à **Thierry Rocacher** pour m'avoir apporté son aide et prodigué de précieux conseils qui m'ont permis de mener à bien mes missions.

Enfin, un vif merci à mes camarades de la filière ESPE **Xavier Bourlot**, **Cameron Bray**, **Titouan Guezeneuc** et **Romain Vitrat** toujours présents pour échanger au sujet des différentes problématiques que j'ai pu rencontrer, me permettant ainsi de prendre du recul et de mieux les aborder par la suite.

Table des matières

Acronymes	i
Glossaire	ii
1 Introduction	1
2 Cadre et objectifs du stage	3
2.1 Présentation générale de Dial, "le bracelet GPS qui sauve la vie"	3
2.2 Objectifs du stage	4
2.2.1 La formalisation et la méthodologie	4
2.2.2 Des objectifs variés	5
2.2.2.1 Étudier et mettre en place des solutions techniques	5
2.2.2.2 Se challenger sur le plan personnel	6
2.3 Positionnement du stage	7
2.3.1 L'électronique et la programmation embarquée faible énergie pour l'IoT	7
2.3.2 Le contexte métier : la sécurité et la prévention des risques	7
2.3.3 Caractéristiques techniques et les technologies	8
3 Architecture du système Dial et analyse de la solution existante	10
3.1 Fonctionnement haut niveau de la balise Dial	10
3.2 Les fonctionnalités détaillées de la solution	11
3.2.1 Tracking de la position	11
3.2.2 Détection de l'immobilité prolongée	11
3.2.3 Zone de contrôle	11
3.3 La conception de la carte et du logiciel embarqué	12
3.3.1 Les principaux composants de la carte électronique	12
3.3.2 L'analyse du fonctionnement du logiciel actuel	13
4 Analyse et réalisations au niveau matériel de la carte Dial	15
4.1 Achat d'équipement	15
4.2 Changement de composants	16

4.2.1	Du GSM aux récents réseaux LTE-M et Nb-IoT ?	16
4.2.2	Le choix d'un nouveau microcontrôleur	18
4.3	Mise en place d'un nouveau banc de programmation	20
5	Évolution et développement de la version 2.0 du software	22
5.1	Restructuration et fiabilisation	22
5.1.1	Le fonctionnement Foreground/Background	22
5.1.2	Restructuration et nettoyage du code	23
5.2	Repenser le contrôle de la balise avec le bouton	24
5.2.1	Le fonctionnement existant et ses limites	25
5.2.2	Le cahier des charges, les contraintes et la nouvelle machine à états	26
5.2.3	L'implémentation de nouvelles fonctions et tests	27
5.3	L'optimisation de la consommation	27
5.3.1	Mise en place du mode Low Power sur le MSP430	28
5.3.2	Initialisation des pins	30
5.3.3	Calcul théorique et mesures réelles	30
5.3.4	Analyse, limites rencontrées et pistes de réflexion	31
5.4	Bootloader pour une mise à jour "Over The Air"	32
5.4.1	Le fonctionnement de la solution existante	33
5.4.2	"Dual image" et re mapping mémoire	33
5.5	Envoi périodique de la position GPS	35
5.5.1	Nouvelle gestion du temps avec le périphérique Real Time Clock . .	35
5.5.2	Récupération de la position GPS et prochaines actions à mener . .	37
6	Conclusions et perspectives	38
Annexes		41
Annexe : Présentation de la société		41
Annexe : Caractéristiques techniques de la balise Dial		42
Annexe : Matériel électronique acheté pour la start-up		43
Annexe : Algorigramme et interruptions du software actuel		44
Annexe : Code source du squelette de la machine à états		46

Acronymes

DIAL Dispositif Individuel d'Alerte et de Localisation. 1, 3

GNSS Global Navigation Satellite System. 13

GPRS General Packet Radio Service. 8

GPS Global Positioning System. 8

GSM Global System for Mobile Communications. 5

IoT ou IdO Internet of Things ou Internet des Objets. 2

RISC Reduced Instruction Set Computer. 12

SMS Short Message Service. 8

SPI Serial Peripheral Interface. 12

UART Universal Asynchronous Receiver-Transmitter. 12

Glossaire

Conditions de course Correspond à une situation de course, moments où les résultats du programme dépendent du comportement temporel de ce dernier. Elles surviennent lorsqu'il y a une interaction asynchrone entre la partie matérielle (hardware) et la partie logicielle (software). [3]. 14, 22

Firmware Partie du logiciel (software) qui est stockée dans un périphérique matériel et qui permet de le faire fonctionner correctement.. 13, 14

GPRS Protocole réseau dérivée du GSM. Augmente le débit et permet la connexion à internet (autorise le transfert de données en mode paquet et connectivité IP). 12, 16

GPS Système de positionnement par satellites appartenant au gouvernement fédéral des États-Unis. 9–12

GSM Standard de téléphonie mobile adopté en Europe, en Asie et en Australie. Bandes de fréquences utilisées : 900 MHz et 1800MHz. 8–12, 16

Hardware Correspond à tout ce qui est relatif à la partie matérielle (carte électronique, composants, sonde et banc de programmation... etc). 2, 5, 7, 8, 10, 13

IoT Désigne un ensemble d'objets connectés contenant des capteurs, des logiciels dont l'objectif est d'échanger des données via internet. 1

SMS Protocole de communication se basant sur le GSM, utilisé pour envoyer au max 160 caractères. 10–12

Software Correspond à tout ce qui est relatif au logiciel, désigne plus précisément le programme ou les données avec lesquels l'utilisateur va interagir la plupart du temps.. 2, 7, 10

SPI Bus de données série synchrone, périphérique inclut dans le microcontrôleur fonctionnement sur le schéma maître-esclave. 13

UART Périphérique inclut dans le microcontrôleur correspondant à émetteur-récepteur asynchrone universel, permettant l'échange de données entre deux entités. 12, 16, 30, 35

Chapitre 1

Introduction

Dans le cadre de mon projet de fin d'études, j'ai effectué du 8 Février au 6 Août 2021 un stage en tant qu'ingénierie électronique/systèmes embarqués dans la start-up Ido-data. Cette dernière évolue dans deux secteurs d'activité, celui de l'informatique et de l'électronique. Plus précisément, ces services et ces produits s'inscrivent dans le domaine de l'Internet des Objets (IdO) ou aussi appelé Internet of Things (IoT)¹. Elle a été créée en 2017 et elle se situe en région lyonnaise. Pour une présentation complète de l'entreprise vous pouvez vous référer à l'annexe 6. L'équipe d'Ido-data développe des solutions IoT, applicatives et web pour répondre aux problématiques de la prévention des risques et de la sécurité. C'est en accord avec ses valeurs qu'un partenariat avec la SNSM² et le designer Starck est né dans l'objectif de concevoir un objet permettant de faciliter le secours des personnes en danger isolées en mer. Ce partenariat a porté ses fruits et le bracelet qui sauve des vies a été imaginé puis industrialisé en 2019.



FIGURE 1.1 – Logo Dial

Le bracelet DIAL³ est le produit conçu et développé par la start-up. Il s'agit d'une balise étanche et connectée qui permet de géolocaliser le porteur. Elle lui permet de passer une alerte si il se trouve dans une situation dangereuse. Les détails concernant la balise et son fonctionnement seront décrits dans la partie 2.1.

-
1. Désigne un ensemble d'objets connectés contenant des capteurs, des logiciels dont l'objectif est d'échanger des données via internet
 2. Société Nationale de Sauvetage en Mer
 3. Dispositif Individuel d'Alerte et de Localisation

L'entreprise s'adresse aux professionnels de la sécurité, du sauvetage et des premiers secours. Dans le cadre du produit Dial, le client principal était la SNSM. Cependant Ido-data s'occupe aussi de la commercialisation du bracelet et du service après-vente, de ce fait elle s'adresse aussi à des revendeurs de matériel nautique ainsi qu'à des particuliers.

Pour ce projet de fin d'études je souhaitais intégrer une petite structure pour pouvoir découvrir le fonctionnement interne et pour mieux comprendre les différents enjeux liés à l'entrepreneuriat. Pour concrétiser ma formation INSA et plus particulièrement ma spécialité ESPE (Embedded Smart Power Electronics) j'avais aussi l'envie de me diriger vers des problématiques liées à la gestion de l'énergie dans des systèmes de faible puissance. Le choix d'Ido-data pour réaliser mon stage m'a paru tout indiqué d'autant plus que la problématique posée autour du bracelet Dial était intéressante et ambitieuse.

J'ai donc intégré l'entité "Lab" en charge des évolutions et des innovations au sein de l'entreprise. Mon objectif global était d'étudier les axes d'amélioration du produit DIAL, d'implémenter puis de tester les solutions. Le but étant d'optimiser et d'améliorer le produit ainsi que son fonctionnement dans l'objectif d'un déploiement d'une nouvelle version à l'étranger.

Ce stage se situe à cheval entre plusieurs domaines de compétences tous étant intrinsèquement liés au monde de l'Internet des Objets. Certains étaient directement liés à ma formation et à ma spécialité : l'électronique, avec à la fois la partie Hardware⁴ et Software⁵. Ce stage abordait également des notions liées aux réseaux et à la télécommunication me permettant ainsi d'approfondir mes connaissances sur ces sujets.

Dans un premier temps nous allons poser précisément le cadre ainsi que les différents objectifs du stage, en présentant en détails le produit Dial, les problématiques auxquelles je devais répondre ainsi que les outils utilisés. Ensuite, nous analyserons la solution existante, son fonctionnement et ses limites. Nous continuerons par le développement de mes réalisations techniques tant au niveau Hardware que Software, en tâchant d'analyser les résultats obtenus ainsi que les différents problèmes rencontrés. Cette partie expliquera également le travail effectué en lien avec le projet de commercialisation du produit à l'étranger. Enfin, nous conclurons sur cette expérience, ses apports ainsi que son impact sur mon projet professionnel.

4. Correspond à tout ce qui est relatif à la partie matériel (carte électronique, composants, sonde et banc de programmation... etc

5. Correspond à tout ce qui est relatif au logiciel, désigne plus précisément le programme ou les données avec lesquels l'utilisateur va interagir la plupart du temps.

Chapitre 2

Cadre et objectifs du stage

Après avoir introduit brièvement le sujet ainsi que le contexte général de mon projet de fin d'études, nous allons dans ce chapitre, détailler l'environnement et le cadre précis dans lequel s'inscrit mon stage au sein de la société, ainsi que les différents objectifs qui m'ont été confiés.

2.1 Présentation générale de Dial, "le bracelet GPS qui sauve la vie"

Comme évoqué précédemment j'ai intégré l'entité "Lab" qui a pour but de développer des solutions innovantes et de les améliorer. Mon stage est lié au projet d'innovation le plus abouti de l'entreprise : DIAL. Il s'agit du premier produit industrialisé par la start-up. Étant engagée en tant que stagiaire ingénierie en électronique/systèmes embarqués mon stage se concentre plus précisément sur la carte électronique et le logiciel embarqué contenu dans la balise étanche.

Comme évoqué précédemment, le produit DIAL a été fait en partenariat avec la SNSM, il se compose d'une balise connectée étanche, d'un bracelet conçu par le designer Starck, ainsi que d'une application mobile. Lancé en 2018 au salon nautique à Paris, ce produit a été pensé pour être utilisé en mer. La balise étanche (IPX8¹) se glisse dans un bracelet que l'on porte au poignet. Grâce à un bouton poussoir l'utilisateur peut interagir avec la balise, il peut notamment , partager sa localisation, et si il rencontre une situation de danger, envoyer une alerte. Un moteur vibrant permet de signifier à l'utilisateur dans quel état se trouve la balise. En plus du bracelet et de la balise, Ido-Data a développé une application mobile permettant à l'utilisateur de renseigner un référent (contact à qui sera envoyé la notification d'alerte et qui pourra contacter les secours), de modifier certains paramètres, de voir l'état de batterie de la balise... etc. Deux types d'alerte peuvent être déclenchés :

1. Étanche à 20mètres pendant 4 heures

1. Alerte déclenchée par le porteur en appuyant sur le bouton.
2. Alerte si la balise détecte une immobilité prolongée de plus de 3 minutes.

Le référent a accès à la localisation du porteur et si l'une des deux alertes est passées, il est notifié et a la possibilité d'appeler les secours et de transmettre les coordonnées de la balise. Cette position est une information cruciale pour les sauveteurs, elle permet de gagner de précieuses minutes en diminuant significativement le temps d'intervention, augmentant ainsi les chances de survie de la victime.

À ce jour le produit (bracelet+balise+chargeur+abonnement+application) est vendu à 149€ TTC, il est disponible sur plusieurs "marketplaces" (Amazon, FlySurf..etc).



FIGURE 2.1 – Bracelet contenant la balise et page d'alerte de l'application Dial

2.2 Objectifs du stage

Le souhait de l'entreprise est d'avoir au sein de l'équipe des compétences en électronique et en systèmes embarqués afin de pouvoir par la suite internaliser les développements et les évolutions du produit Dial. C'est dans ce contexte que s'inscrivent les différents objectifs qui m'ont été confiés. De façon très globale j'étais en charge d'étudier, d'implémenter et de tester les évolutions de la carte électronique et du logiciel embarqué de Dial.

2.2.1 La formalisation et la méthodologie

Au tout début de mon stage lors de la première réunion avec mes supérieurs nous avons établi, à partir du sujet de stage, une mind map. Cette dernière m'a permis tout au long de mon stage de pouvoir avoir une vue d'ensemble sur les différents objectifs ainsi que leurs priorités. Cet outil m'a permis par la suite d'établir au fur à mesure mon plan d'actions à réaliser. Pour formaliser les actions à faire et pour suivre mon avancée la start-up

utilise l'application Click-Up. Je renseignais dans cette dernière l'état d'avancement des tâches sur lesquelles je travaillais. Sur cette même plate-forme je répertoriais également la documentation que je produisais.

Les objectifs de mon stage ont évolué en fonction de mes avancées et des choix réalisés. Les priorités ont parfois du être révisées mais cette méthodologie m'a permis de toujours garder une idée claire des objectifs à remplir et me permettait d'être organisée et efficace.

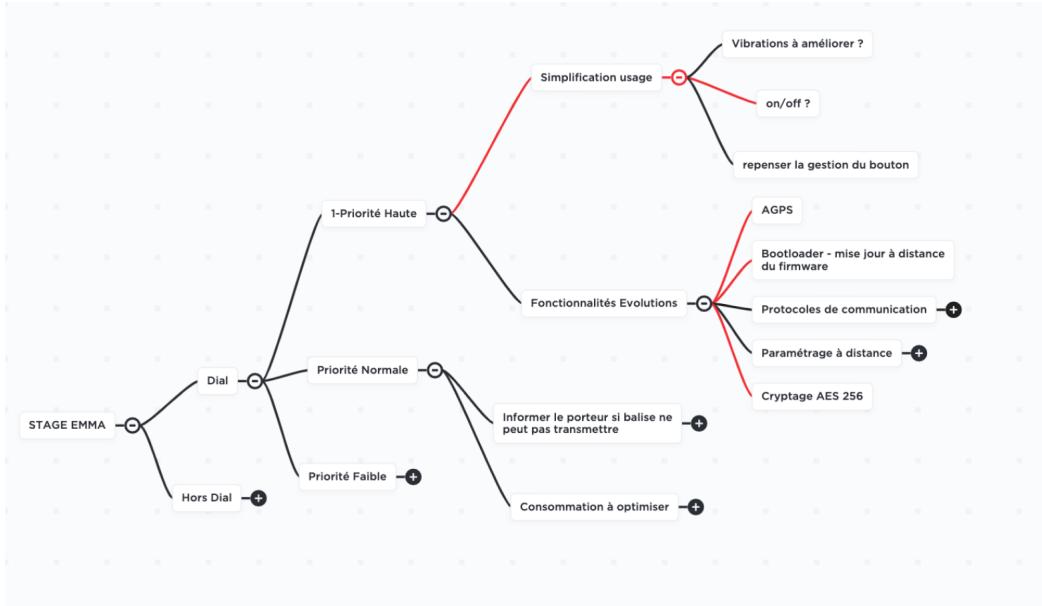


FIGURE 2.2 – Mindmap des objectifs de mon stage

2.2.2 Des objectifs variés

2.2.2.1 Étudier et mettre en place des solutions techniques

Chaque objectif qui m'a été confié répondait à une problématique observée sur la carte ou sur le logiciel embarqué de Dial. Nous allons donc expliciter ces problématiques ainsi les objectifs associés.

Tout d'abord concernant la partie Hardware, l'entreprise n'étant pas suffisamment équipée, j'ai eu pour mission d'acheter du matériel électronique (fer à souder, accessoires de soudure, oscilloscope etc...) ainsi que de concevoir des prototypes fonctionnels à partir des cartes électroniques existantes afin de pouvoir travailler correctement et mener à bien mon stage. Une fois cela fait, j'ai eu la charge d'étudier le changement de deux composants principaux :

1. Le GSM² afin de passer à une technologie plus récente que la 2G.

2. Standard de téléphonie mobile adopté en Europe, en Asie et en Australie. Bandes de fréquences

2. Le microcontrôleur car le logiciel embarqué étant conséquent il atteint les limites de la mémoire.

Ensuite, après avoir présenté et appliqué ou non des changements je me suis concentrée sur les objectifs liés au software embarqué de Dial, après une phase de compréhension et d'analyse de la solution déjà existante.

1. La plupart des utilisateurs ont fait remonter que le contrôle de la balise via le bouton et que le système des vibrations étaient trop complexes à utiliser. Le but a été de simplifier la gestion de ce bouton et de ces vibrations.
2. La balise étant étanche et scellée il est impossible d'accéder à la carte électronique. Pour la mise à jour du software j'ai du adapter et m'assurer du bon fonctionnement d'un bootloader pour pouvoir effectuer des mises à jour à distance et sans fil (Over the Air).
3. Des dysfonctionnements étant observés par les utilisateurs trois de mes objectifs étaient donc de restructurer, fiabiliser et optimiser le code afin de résoudre ces problèmes.

2.2.2.2 Se challenger sur le plan personnel

Pendant toute la durée de mon stage j'ai été la seule personne de l'équipe à avoir suivi une formation électronique/systèmes embarqués j'ai majoritairement travaillé seule sur ces sujets. Je pouvais évidemment compter sur l'aide de mon tuteur en cas de besoin, cependant même si il a une très bonne connaissance du produit ainsi que du code ce n'est pas expert dans ces domaines. Pour pouvoir travailler de façon efficace j'ai du m'imposer une méthode et une rigueur. Cette autonomie et ces responsabilités m'ont permis de m'autoformer sur de nombreux sujets, j'ai pu de ce fait acquérir de nouvelles connaissances de façon durable. Cependant travailler seul présente des limites. À certains moments j'aurais eu besoin d'un binôme ou d'une personne plus expérimentée pour me permettre de prendre du recul sur mon travail. Confronter des idées et mélanger les expériences est selon moi un atout essentiel pour réaliser un projet.

Par ailleurs, j'avais en parallèle de ces objectifs le rôle de "soutien technique" de l'équipe marketing et commerciale. Lorsqu'ils en ressentaient le besoin, ils pouvaient faire appel à moi, pour analyser ou apporter des explications sur les sujets techniques. Cette facette de mon rôle a été extrêmement enrichissante. D'une part, j'ai été amenée à vulgariser certains sujets techniques pour qu'ils soient compris par tout le monde, d'autre part, j'ai participé à des moments d'échanges avec les autres membres de l'équipe où chacun apportait une vision des choses différentes. J'ai aussi l'occasion d'apprendre quelques notions de droit

utilisées : 900 MHz et 1800MHz. Désigne aussi par métonymie le composant électronique qui communique sur ce réseau.

et de marketing.

Quels étaient les objectifs personnels que je voulais atteindre ? Personnellement j'attendais de ce stage qu'il me sorte de ma zone de confort, qu'il me permette de faire des rencontres et qu'il m'aide à prendre de la confiance et de l'assurance vis à vis de mes décisions et de mes réalisations. Nous reviendrons dans la partie 6 sur ces challenges pour faire le bilan de cette expérience.

2.3 Positionnement du stage

2.3.1 L'électronique et la programmation embarquée faible énergie pour l'IoT

Tout d'abord la balise Dial étant un objet connecté, mon stage s'inscrit tout d'abord au sein du domaine l'IoT. Pour être plus précise, même si la partie réseau, protocole de communication et stockage des données sur le cloud a été rapidement abordée, mon stage a été plus orienté vers le Software embarqué. Par ailleurs de bonnes bases en électronique et en conception Hardware étaient nécessaires. De l'analyse de schématiques électroniques, au choix des composants, en passant par la lecture de datasheets, les connaissances en électronique acquises lors des bureaux d'études en électronique pendant ma formation ont été mises en pratiques. En ce qui concerne le côté Software j'ai pu approfondir mes compétences en microcontrôleur et en programmation bas niveau en langage C. En conclusion stage s'inscrit dans la continuation des enseignements suivis à l'INSA notamment ceux de l'option ESPE, orientés vers la gestion de l'énergie dans les systèmes embarqués.

2.3.2 Le contexte métier : la sécurité et la prévention des risques

En France 52 personnes décèdent par jour dans des accidents du quotidien, ce chiffre issu de l'enquête courante sur les accidents de la vie est mis en avant dans le Manifeste de l'entreprise Ido-Data. La philosophie de l'entreprise est donc de développer des solutions appartenant au monde des objets connectés dans le but d'assurer au maximum la sécurité des utilisateurs.



FIGURE 2.3 – Logo Ido-data

Si dans un premier temps la balise Dial a été conçue pour les particuliers, dans le cadre d'une application dans le domaine de la sécurité lors d'activités nautiques (localisation partagée avec les secours maritimes en cas de problèmes), son utilisation peut s'élargir au marché des sports et loisirs de plein air (randonnée, VTT, kayak...etc). Certains utilisateurs se servent de la balise Dial pour des problématiques différentes de celle d'origine. Pour suivre un personne atteinte d'alzheimer, pour mettre à bord de bateaux de location ou encore pour suivre les participants d'une compétition sportive par exemple.

Pour aller encore plus loin, le contexte métier dans lequel s'inscrit cet objet, est la prévention des risques et de la sécurité au sens large. Cet objet intéresse notamment certains organismes professionnels pour équiper les travailleurs qui n'ont pas la possibilité d'avoir un téléphone portable avec eux. Les applications métiers sont très diverses et des discussions avec différents acteurs sont actuellement en cours pour adapter la solution existante et pour l'intégrer à d'autres systèmes afin de répondre à des besoins spécifiques.

2.3.3 Caractéristiques techniques et les technologies

Dans cette sous partie nous allons aborder le contexte technique de mon stage, en commençant par brièvement décrire comment a été conçu la balise connectée, les technologies choisies, ainsi que le matériel et les logiciels utilisés. Tous les détails techniques concernant la balise peuvent être retrouvés dans l'annexe 6

Nous allons commencer par poser le cadre général technique au niveau électronique. Les spécificités Hardware de la carte et des principaux composants seront détaillées dans la partie 3.3.1. Tout d'abord pour calculer sa position, la balise utilise un module GPS³ afin de recevoir les coordonnées permettant de localiser le porteur du bracelet. Ensuite, la balise communique avec un serveur sur le réseau GSM via les fréquences 900 et 1800 MHz. L'envoi des informations se fait à la fois par internet via le réseau GSM en utilisant la norme GPRS⁴ (2G) et par le protocole SMS quand la connexion internet ne s'établit pas. Pour accéder au réseau mobile Dial embarque une carte SIM multi-opérateurs offrant une plus grande couverture pour la connexion au réseau. En ce qui concerne la batterie, elle est rechargeable par induction et l'entreprise communique une autonomie de 6 heures lorsque la balise est en mode alerte et que la position est actualisée toutes les 15 secondes. Côté instrumentation et matériel électronique j'avais à ma disposition :

- Un oscilloscope contenant un analyseur logique
- Un multimètre
- Une station et un fer à souder

3. Système de positionnement par satellites appartenant au gouvernement fédéral des États-Unis

4. Protocole réseau dérivée du GSM. Augmente le débit et permet la connexion à internet (autorise le transfert de données en mode paquet et connectivité IP)

- Divers accessoires pour souder et concevoir des prototypes (connecteurs, fils, étain, plaques d'essais ...etc)

En choisissant les technologies GPS et GSM la balise ne peut donc pas être utilisée à l'intérieur ainsi qu'à un endroit non couvert par le réseau 2G.

Concernant le contexte technique lié à la programmation embarquée en langage C, le microcontrôleur fait parti de la gamme MSP430 de chez Texas Instrument idéal pour les applications "low power" où la gestion de l'énergie est cruciale. L'IDE utilisé Code Composer Studio, est lui aussi issu de chez TI et il est basé sur le framework Eclipse. Pour finir, en ce qui concerne la gestion du code tout a été fait via GitHub. Pour les autres fichiers relatifs au projet ou à la documentation ils sont stockées sur Google Drive ou sur Click-Up.

Chapitre 3

Architecture du système Dial et analyse de la solution existante

Le but de ce chapitre est de donner une vue d'ensemble du projet sur lequel j'ai travaillé au cours de ces six mois de stage. Ce travail d'analyse fait parti des mes réalisations car avant de commencer à étudier de potentielles évolutions j'ai du d'abord comprendre les différents cas d'usage ainsi que les détails de la conception Hardware et Software. Les prochaines sections présenteront ce travail de compréhension au niveau fonctionnel et technique.

3.1 Fonctionnement haut niveau de la balise Dial

La balise peut se trouver dans trois états différents : Allumée, en mode alerte et en veille.

- La balise est "allumée" (ACTIVE) : Une notification est envoyée au référent via l'application et via SMS lorsque la balise est allumée. Il peut alors suivre les mouvements du porteur. Le GPS et l'accéléromètre sont allumés périodiquement pour récupérer les données. Idem pour le GSM qui transmet les informations au serveur.
- La balise est en "alerte" (ALERTE) : Se déclenche lors d'un appui prolongé sur le bouton poussoir. Une notification (notification push, SMS et mail) est envoyé au référent. Sur l'application il peut appeler les secours (112 ou le CROSS 196) selon si le porteur est en mer ou non. Le GPS et le GSM relèvent et envoient respectivement la position toutes les 15 secondes.
- La balise est "éteinte" (SLEEP) : La balise entre dans ce mode lorsque que l'utilisateur "l'éteint". Le GPS n'est plus alimenté, le GSM est en mode sleep, plus rien ne se passe.

Le bouton poussoir et le moteur vibrant permettent à l'utilisateur de contrôler l'état de la balise, ce fonctionnement spécifique fait l'objet de la partie 5.2.3.

Selon ces états le système offre différentes fonctionnalités que nous allons détailler dans le prochain paragraphe.

3.2 Les fonctionnalités détaillées de la solution

La balise Dial avec l'application associée offrent à l'utilisateur plusieurs fonctionnalités pour assurer au maximum sa sécurité. Dans un premier temps l'utilisateur est invité via l'application mobile, à remplir ses informations ainsi que celle concernant son référent (personne qui sera notifiée dès que la balise est en marche). La balise communique via le réseau GSM, les informations sont envoyées par internet et par SMS. La seule interface entre l'utilisateur et la balise est un bouton poussoir qui permet d'allumer/éteindre la balise et de déclencher/d'annuler une alerte, ainsi qu'un moteur vibrant.

3.2.1 Tracking de la position

La balise via le composant GPS récupère puis envoie les données contenant les informations de la position. La périodicité de la remontée des données est différente selon si la balise est en mode alerte ou non. En mode alerte, la position est envoyée toutes les minutes. En fonctionnement normal, cette durée est de 5 minutes par défaut mais l'utilisateur peut la modifier via l'application. La position se compose des coordonnées GPS (latitude et longitude), si un appel aux secours est passé cette position est mise à leur disposition afin qu'ils puissent intervenir au plus vite.

3.2.2 Détection de l'immobilité prolongée

Grâce à un capteur spécifique : un accéléromètre, l'accélération de la balise est calculée. En cas d'une immobilité de plus de trois minutes une alerte est déclenchée. Le référent est prévenu via l'application et via SMS. L'utilisateur peut désactiver cette fonctionnalité. Il est important de noter que si le porteur du bracelet est en mer et donc en mouvement, à cause du courant, aucune immobilité ne peut être détectée.

3.2.3 Zone de contrôle

Enfin l'utilisateur peut décider d'envoyer une alerte si sa position se trouve en dehors d'une zone de contrôle préalablement établie via l'application. Cette zone correspond à un cercle de diamètre réglable autour de la balise ou du téléphone du porteur. Cette fonctionnalité peut être elle aussi désactivée via l'application.

3.3 La conception de la carte et du logiciel embarqué

La solution et l'objet ayant déjà été industrialisés nous allons dans un premier temps présenter l'analyse de la carte électronique existante.

3.3.1 Les principaux composants de la carte électronique

J'ai tout d'abord analysé la dernière version de la carte électronique de Dial, mon but était tout d'abord d'identifier les principales fonctions et les principaux composants ainsi que leur rôle respectif.

- Le microcontrôleur (μ C) : le "cerveau" du système. C'est cet élément qui joue le rôle de chef d'orchestre. Il reçoit les données des différents capteurs, du GPS et du GSM, il les traite et prend ensuite les actions nécessaires à son bon fonctionnement. Il s'agit d'un modèle de chez Texas Instrument le MSP430FR5969 conçu avec une architecture RISC¹ 16bits. Un quartz de 32 kHz est utilisé en tant qu'oscillateur basse fréquence (LFXT). Les principaux périphériques utilisés dans notre application sont les Timers, l'ADC, Power Management System, le RTC et le Watchdog. En terme de bus de communication les protocoles UART² et SPI³ sont sollicités. [4]
- Le composant GSM : Ce composant essentiel permet, via le réseau, d'établir la communication entre un serveur et le microcontrôleur. C'est grâce à lui que la balise se connecte à internet et que l'utilisateur peut interagir avec cette dernière. Pour rentrer plus dans les détails, il s'agit du modèle SIM800C de chez SimCom conçu dans un packaging SMT 42 pins. Il fonctionne à partir des quatre bandes de fréquences (850,900 1800 et 1900 MHz), mais dans notre cas l'antenne du GSM (Pulse W3070 [10]) n'est sensible qu'à deux plages : EGSM900MHz et DCS1800MHz. Il communique avec le microcontrôleur via le bus UART. Ce module inclut également le protocole réseau GPRS qui assure la connexion internet via le protocole de transmission HTTPS. De plus, avec la carte SIM multi-opérateurs des messages SMS sont aussi envoyés si la connexion internet ne s'effectue pas. [13]
- Le composant GPS : Permet de récupérer l'heure précise, la date et les coordonnées. Le composant provient aussi de chez Simcom il s'agit du SIM28M. Lorsque le module est alimenté il a besoin d'un certain temps selon l'endroit pour effectuer son premier fix. Selon la documentation cette durée (Time to First Fix) peut aller jusqu'à 30secondes si la puissance du signal est de -130dB. En pratique cela peut

1. Type d'architecture de processeur qui se caractérise par des instructions de base aisées à décoder, uniquement composé d'instructions simples
2. Périphérique inclus dans le microcontrôleur correspondant à émetteur-récepteur asynchrone universel, permettant l'échange de données entre deux entités
3. Bus de données série synchrone, périphérique inclus dans le microcontrôleur fonctionnement sur le schéma maître-esclave

prendre plusieurs minutes. Une antenne montée sur le PCB (146235-0001 de chez Molex [9]) capte les signaux des satellites dont la fréquence est comprise en 1,556 et 1,607 MHz. On utilise le terme "GPS" mais il serait plus précis de parler de GNSS⁴ pour parler de cette technologie. Le GPS ne correspondant qu'à la constellation de satellite des américains. Aujourd'hui, la plupart du temps, les informations issues de plusieurs constellations sont utilisées pour une plus grande précision. [12]

- Accéléromètre : Le LIS3DH de chez STMicroelectronics est un capteur mesurant l'accélération selon 3axes. C'est grâce à lui qu'à terre il est possible de détecter si l'utilisateur est en immobilité prolongée. Il communique ses données au μ C via le bus de données SPI. [8]
- Batterie : Il s'agit d'une batterie Li-Po rechargeable de 3.7V de 420 mAH. [2]
- Autres composants : Évidemment sur cette carte électronique on retrouve de nombreux composants passifs comme des capacités de découplage, des résistances (pour l'adaptation de tension), des inductances (pour le filtrage) ou des transistors MOS (pour les commandes)...etc. On retrouve aussi un régulateur de tension qui permet d'alimenter le μ C en 3.3V, un amplificateur RF ainsi qu'un composant qui gère la réception de la puissance et le chargement de la batterie via le chargeur par induction (technologie QI).

3.3.2 L'analyse du fonctionnement du logiciel actuel

En parallèle de la compréhension du Hardware et avant de commencer à développer, je me suis d'abord familiarisée avec le Firmware déjà existant. Cette phase m'a permise à la fois de comprendre le fonctionnement actuel et de noter au fur à mesure les pistes d'amélioration.

Tout d'abord le code a été développé principalement par un bureau d'études, ainsi que par mon tuteur. Mis à part les commentaires je n'avais accès qu'à très peu de documentation fonctionnelle. J'ai donc du à partir du code, comprendre le fonctionnement de la balise. Le programme étant conséquent (24 fichiers sources dans trois dossiers), j'ai tout d'abord commencé par réaliser de la documentation fonctionnelle avec un algorithme (cf 6) et des diagrammes de classe pour avoir une vue d'ensemble.

Dans les grandes lignes, le programme se base sur un principe de déroulement quasi parallèle et événementiel. Il repose sur des interruptions ainsi qu'un "Scheduler" qui gère,

4. Détermination de la position et de la vitesse d'un point à la surface ou au voisinage de la Terre, par traitement des signaux radioélectriques en provenance de plusieurs satellites artificiels, reçus en ce point. Le sigle GNSS désigne aussi un système de localisation et de navigation, associant plusieurs systèmes à couverture mondiale, notamment le système GPS (américain), le système Glonass (russe) et le système Galileo (européen), pour répondre aux besoins des utilisateurs des services terrestres, maritimes et aéronautiques.

en prenant en compte un ordre de priorité, les différents évènements qui ont été mis dans une file. En tout il y a 15 évènements, 10 interruptions et de plus le firmware comprend aussi le code d'une interface avec un PC qui avait été développée pour faciliter les tests menés par les fondateurs. Par ailleurs, malgré que les fichiers sources soient répartis dans plusieurs dossiers le logiciel ne semble pas avoir été conçu en couches distinctes. Dans le main.c par exemple on retrouve des appels à des fonctions ainsi que du code bas niveau mettant à jour les valeurs des registres de certains périphériques. Tout ces éléments contribuaient à un manque de clarté du code et à une compréhension plus ardue. Après plusieurs jours d'analyse j'ai aussi conclu qu'en l'état, le fonctionnement de la balise étant assez complexe, on atteint selon moi les limites du modèle. En effet le recours aux interruptions, et donc à des variables globales peut mener à des interdépendances et à des Conditions de course. Le résultat du système dépend du comportement temporel et de ce fait il est quasiment impossible de prévoir précisément tous les comportements possibles. Cela peut mener à des comportements erratiques voir même un crash du système. Cependant pour éviter cela un watchdog est implémenté pour permettre le reset du Firmware en cas de blocage.

Mes premières observations et les premières pistes d'amélioration étaient les suivantes :

- Optimisation possible du temps passé en Low power mode.
- Une gestion très complexe du bouton poussoir, avec notamment un timer qui tourne en continu.
- Le périphérique RTC (Real Time Clock) n'est pas utilisé pour mettre à jour le calendrier.
- La bibliothèque de chez TI pour programmer les périphériques n'est pas utilisée.

C'est grâce à cette première analyse que j'ai pu ensuite orienter et commencer les développements logiciels décrits dans le Chapitre 5.

Chapitre 4

Analyse et réalisations au niveau matériel de la carte Dial

Après m'être familiarisée avec le fonctionnement global du système et du logiciel, j'ai été en charge d'étudier la faisabilité des changements hardware de la carte électronique de Dial.

4.1 Achat d'équipement

Tout d'abord dès mon arrivée, on m'a confié la mission d'équiper la start up avec du matériel électronique. D'une part pour que je puisse avoir ce matériel à disposition pour mener à bien mon stage, d'autre part pour que l'entreprise dispose des équipements de base, pour qu'ils puissent par la suite internaliser les développements. Pour effectuer ces achats j'avais une contrainte budgétaire à respecter, l'entreprise ne pouvant pas se permettre de trop grosses dépenses, le budget était donc d'environ 1000€.

J'ai commencé par lister le matériel qui est essentiel pour travailler :

- Multimètre
- Station et fer à souder
- Accessoire pour souder : étain, fils, tresse à dessouder et loupe
- Oscilloscope

Ensuite, dans l'optique de trouver du matériel de qualité j'ai concentré mes recherches sur les sites des distributeurs d'électronique : Farnell, Mouser et Digikey...etc. Après différents comparatifs, j'ai décidé de passer les commandes via le site de Farnell (farnell.com) car c'est celui qui proposait les meilleures offres et délais de livraison, en plus d'une remise de 15%.

Le choix du matériel n'a pas été aisé, j'ai effectué de nombreuses recherches pour regrouper des avis dans le but de trouver le meilleur compromis entre la qualité et le prix.

Concernant le fer à souder je me suis dirigée vers la marque **Weller** que j'avais déjà utilisée à l'INSA et qui est un des leaders dans ce domaine. Il était aussi primordial que ce fer fonctionne en basse tension pour éviter toutes fuites de courant qui seraient susceptibles d'endommager les composants. Le fer WP80 fonctionnant à 24V, 80W et 350° respecte le cahier des charges. Après quelques mois d'utilisation occasionnelle, le fer et son alimentation ont parfaitement remplis leurs fonctions.

L'oscilloscope était l'élément le plus difficile à choisir car il existe de très grands écarts de qualité et de prix selon les modèles. La bande passante, la fréquence d'échantillonage et la taille mémoire font partie des critères à prendre en compte. De plus, allant être amenée à télétravailler pour respecter les restrictions liées au contexte sanitaire, un oscilloscope facilement transportable faciliterait mon travail. De ce fait, je me suis tournée vers la gamme de chez Picoscope. L'interface utilisateur disponible via un logiciel (Picoscope 6) sur ordinateur, les picoscopes sont très compacts et restent des appareils performants. Ils sont également équipés d'un analyseur logique ce qui est aussi un gros avantage car il permettra de décoder les trames UART circulant entre le GSM ou le GPS et le microcontrôleur. Ce qui s'avèrera très utile durant les développements au niveau software.

Pour la liste complète du matériel choisi et les références se référer à l'annexe 6.

4.2 Changement de composants

Une fois la commande du matériel passée j'ai eu la charge d'étudier les évolutions au niveau du hardware de la carte électronique de Dial. Ces propositions étaient le fruit des réflexions des fondateurs après avoir industrialisé une version et dans l'optique de faire évoluer le produit.

4.2.1 Du GSM aux récents réseaux LTE-M et Nb-IoT ?

La balise est connectée à internet et communique via la 2G. La 2G est la dénomination que l'on donne au protocole GPRS (qui fonctionne via les fréquences du GSM et qui permet la connexion internet). C'est aussi la 2ème génération de réseau téléphonique qui a été créée en 1991. Le problème qui se pose est que dans certains pays le réseau 2G est en train d'être (ou est déjà) démantelé. Ido-data ayant dans l'optique de se développer à l'international l'arrêt des communications 2G qui pourrait s'avérer critique. Ma mission a été d'étudier le remplacement du module GSM Sim800C par un nouveau module communiquant sur un réseau plus récent conçu pour les objets connectés. Les deux réseaux candidats étaient les suivants : LTE_M et Nb-IoT. J'ai du analyser leurs caractéristiques pour voir si ils pouvaient correspondre aux besoins.

LTE-M/LTE CAT-M1 vs NB-IOT/LTE-CAT-NB1

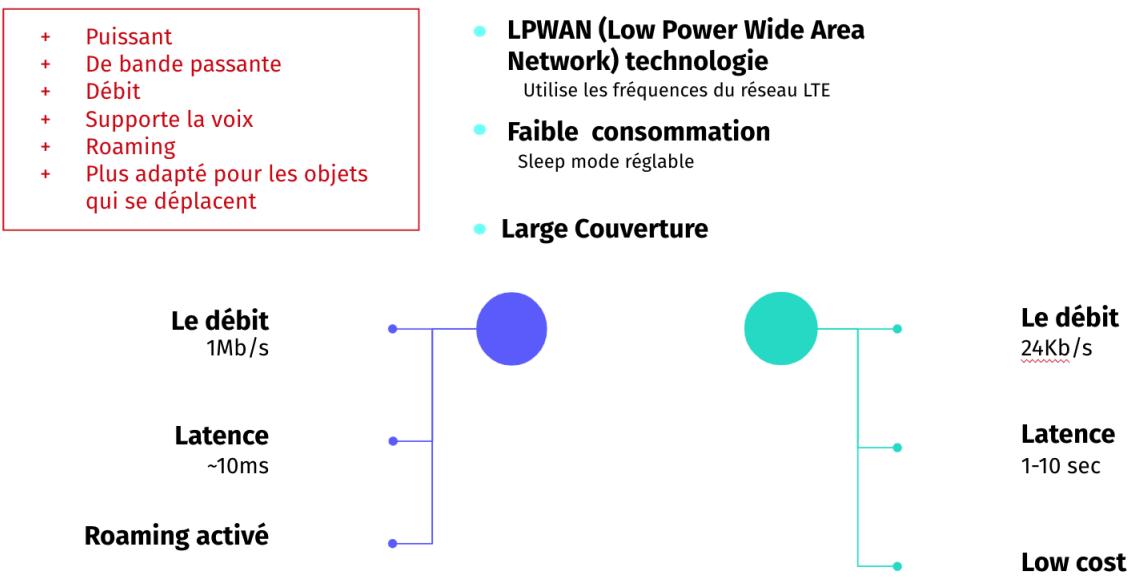


FIGURE 4.1 – Principales caractéristiques des réseaux LTE-M et Nb-IoT

Après m’être renseignée sur les différents cas d’usage et les caractéristiques de ces deux réseaux, pour l’application Dial le réseau LTE-M semble être un meilleur candidat. S’est posé par la suite la question du nouveau composant. Aucun composant communiquant via LTE-M n’était compatible pin à pin avec l’ancien.

Par ailleurs j’ai pu constater que le choix du réseau dépend très fortement de la région du monde où l’on se trouve ainsi que du besoin en terme de qualité de services. Il s’avère qu’au final communiquer sur le réseau 2G se révèle être un choix qui reste pertinent. L’objectif au niveau du développement international est de lancer le produit Dial en Europe. Dans la plupart des pays de l’Union Européenne la 2G est nécessaire car beaucoup de communications essentielles M2M (Machine to Machine) fonctionnent grâce à ce réseau, ce qui implique que les dates du démantèlement ne sont pas fixées ou au pire l’arrêt n’est pas prévu avant 2025. J’ai donc assisté l’équipe marketing pour les aider à choisir les pays à cibler selon ce critère. De plus, choisir de garder le module Sim800C permettait de ne pas modifier le design de la carte électronique. Changer de module aurait impliqué de refaire faire appel au bureau d’études (qui possède tous les fichiers de CAO¹ électronique) pour qu’il fasse un nouveau design avec une nouvelle BOM², un nouveau routage...etc. Il aurait aussi fallu remplacer l’antenne, sa matching structure (les composants qui l’entourent), potentiellement ajouter des nouveaux composants et aussi faire de lourdes modifications du côté du logiciel embarqué.

1. Conception assistée par ordinateur.
2. Bill of Materials, liste des composants électroniques utilisés

J'ai conclu que faire ces changements et réaliser de nouveaux prototypes auraient engendrés des coûts trop importants pour la start up. Après avoir présenté toutes ces réflexions aux fondateurs la décision de ne pas changer le module GSM a été prise.

4.2.2 Le choix d'un nouveau microcontrôleur

La problématique que j'ai ensuite étudiée est celle de la gestion de la place mémoire. En effet le problème rencontré sur le système Dial est le suivant : le firmware est conséquent et le microcontrôleur ne dispose quasiment plus d'espace mémoire disponible. La mémoire du MSP430FR5969 se compose de 64ko de FRAM (mémoire non volatile) et de 2ko de RAM (mémoire volatile). Le mapping mémoire de base a même été modifié pour transformer une partie de la FRAM en RAM.

Nom	Longueur (octets)	Inutilisée (octets)
RAM	0x800	0x608
MORE_RAM	0x2000	0xE10
FRAM	0x1F62	0x98
FRAM 2	0x4000	0x0

TABLE 4.1 – Configuration mémoire du MSP430FR5969 avec le code du firmware et du bootloader

On peut remarquer dans le tableau ci-dessus que la mémoire FRAM est quasiment saturée. Le remapping spécifique de la mémoire optimisant déjà un maximum l'espace disponible, sa saturation ne permet pas de laisser de la marge pour les prochains développements. Cette limite est très contraignante car cela empêche d'implémenter de nouvelles fonctionnalités logicielles. C'est aussi un très gros frein pour l'implémentation du bootloader dual image, cette fonctionnalité étant une priorité pour pouvoir faire des mises à jour du logiciel OTA (Over The Air) et à distance. De ce fait ma mission a été de choisir quel sera le nouveau microcontrôleur.

Tout d'abord j'ai listé les critères que ce nouveau μ C devait avoir :

- Même boîtier (VQFN 48 pins) pour ne pas que l'empreinte soit modifiée. Cela permettait de limiter fortement les coûts, il suffit simplement de remplacer le composant sur des cartes déjà existantes.
- Plus de mémoire RAM et FRAM.
- Compatibilité avec le software existant.
- Compatibilité les périphériques pins à pins. C'est à dire qu'il fallait s'assurer que les périphériques utilisés par le software (ADC, Timers, UART ...etc) étaient bien présents et que leurs entrées/sorties se trouvaient sur les mêmes pins.

— Le prix, l'objectif fixé était de ne pas augmenter le prix de plus de 1 euro.

En prenant en compte ces éléments, il a de suite été évident qu'il fallait choisir un microcontrôleur de la même gamme (MSP430FR). Grâce à l'outil de comparaison de chez TI j'ai pu obtenir un tableau de comparaison entre plusieurs modèles qui pourraient convenir.

Modèle actuel						
	MSP430FR5962	MSP430FR5969	MSP430FR59941	MSP430FR5994	MSP430FR5964	MSP430FR5992
Frequency (MHz)	16	16	16	16	16	16
Features	AES DMA Real-time clock	AES DMA Real-time clock	DMA Low-energy accelerator (LEA) Real-time clock	DMA Low-energy accelerator (LEA) Real-time clock	AES DMA Real-time clock	DMA Low-energy accelerator (LEA) Real-time clock
ADC	12-bit SAR	12-bit SAR	12-bit SAR	12-bit SAR	12-bit SAR	12-bit SAR
Non-volatile memory (kB)	128	64	256	256	256	128
USB	No	No	No	No	No	No
GPIO pins (#)	68	40	68	68	68	68
RAM (kB)	8	2	8	8	8	8
CPU	MSP430	MSP430	MSP430	MSP430	MSP430	MSP430
UART	4	2	4	4	4	4
I2C	4	1	4	4	4	4

Modèle actuel						
	MSP430FR5962	MSP430FR5969	MSP430FR59941	MSP430FR5994	MSP430FR5964	MSP430FR5992
Comparator channels (#)	16	16	16	16	16	16
Package Group	LQFP 64 LQFP 80 NFBGA 87 VQFN 48	VQFN 48	LQFP 64 LQFP 80 NFBGA 87 VQFN 48	LQFP 64 LQFP 80 NFBGA 87 VQFN 48	LQFP 64 LQFP 80 NFBGA 87 VQFN 48	LQFP 64 LQFP 80 NFBGA 87 VQFN 48
Pin count	48 64 80 87	48	48 64 80 87	48 64 80 87	48 64 80 87	48 64 80 87
Package size: mm2:W x L (PKG)	64LQFP: 100 mm2: 10 x 10 (LQFP 64) 80LQFP: 196 mm2: 14 x 14 (LQFP 80) 87NFBGA: 36 mm2: 6 x 6 (NFBGA 87) 48VQFN: 49 mm2: 7 x 7 (VQFN 48)	48VQFN: 49 mm2: 7 x 7 (VQFN 48)	64LQFP: 100 mm2: 10 x 10 (LQFP 64) 80LQFP: 196 mm2: 14 x 14 (LQFP 80) 87NFBGA: 36 mm2: 6 x 6 (NFBGA 87) 48VQFN: 49 mm2: 7 x 7 (VQFN 48)	64LQFP: 100 mm2: 10 x 10 (LQFP 64) 80LQFP: 196 mm2: 14 x 14 (LQFP 80) 87NFBGA: 36 mm2: 6 x 6 (NFBGA 87) 48VQFN: 49 mm2: 7 x 7 (VQFN 48)	64LQFP: 100 mm2: 10 x 10 (LQFP 64) 80LQFP: 196 mm2: 14 x 14 (LQFP 80) 87NFBGA: 36 mm2: 6 x 6 (NFBGA 87) 48VQFN: 49 mm2: 7 x 7 (VQFN 48)	64LQFP: 100 mm2: 10 x 10 (LQFP 64) 80LQFP: 196 mm2: 14 x 14 (LQFP 80) 87NFBGA: 36 mm2: 6 x 6 (NFBGA 87) 48VQFN: 49 mm2: 7 x 7 (VQFN 48)
Prix à l'unité (1)	4,35€	4,14€	6,62€	6,62 €	4,78€	4,91€

(1) Prix affiché sur mouser.fr (/!\ € la gamme [JRGZT](#) vendue en bobine de 250 > gamme [JRGZR](#) vendue en bobine de 2500)

FIGURE 4.2 – Comparatif des microcontrôleurs MPS430 de chez TI

Après avoir vérifié les disponibilités de ces composants sur les sites des revendeurs, j'ai présenté les deux options à mes supérieurs nous avons convenu ensemble de choisir le modèle MSP430FR5964. C'est le μ C avec le plus de mémoire (8ko de RAM et 256ko de FRAM), c'est aussi le plus cher (+64 centimes par rapport au MSP430FR5964) ce prix

pouvant être diminué si il est commandé en grande quantité (4.40 euro pour plus de 100 composants).

Une fois le choix arrêté, j'ai passé une commande de 4 μ C chez Farnell. L'opération de remplacement étant trop délicate et n'étant pas équipée du matériel nécessaire, j'ai envoyé les cartes et les nouveaux MCU à l'industriel afin qu'il effectue le changement.

4.3 Mise en place d'un nouveau banc de programmation

Une fois que les cartes électroniques avec le nouveau μ C furent prêtes, il fallait maintenant les programmer.

La carte a été conçue de façon à être programmée via l'interface JTAG. Des points de test ont été créés afin d'accéder facilement au pins JTAG du μ C. La start up possédait déjà un vieux banc de programmation, très peu pratique car il était la source de nombreux faux contacts. Cela avait entraîné auparavant de nombreuses erreurs pendant la phase de debug, ralentissant ainsi considérablement les développements. L'autre solution était simplement des fils soudés directement entre les points de tests et le connecteur JTAG, efficace mais très fragile il arrivait aussi très fréquemment que des fils se détachent.

J'ai donc décidé de concevoir un banc de programmation fiable et plus simple à transporter afin que le développement software soit plus efficace. Avec Antonin, le responsable du design, j'ai établi un cahier des charges pour qu'il conçoive une pièce avec l'imprimante 3D. Cette pièce devait servir de support légèrement surélevé tout en me permettant de fixer la carte ainsi que d'accéder aux deux faces de cette dernière. Pour l'interface JTAG, j'ai simplement collé au bord de la carte un morceau de plaque d'essais avec des connecteurs. Des connecteurs femelles servent à faire la liaison avec la plaque où se trouve le connecteur JTAG. Il ne reste plus qu'à brancher le MSP-Fet (sonde de debug) au connecteur JTAG permettant ainsi de flasher le programme.

Le banc est fonctionnel, fiable et transportable aisément dans un petit carton en cas de télétravail. Il permet aussi de choisir facilement si l'on souhaite alimenter la carte via le JTAG ou via une alimentation externe. J'ai conçu un petit module avec le moteur vibrant que l'on peut ajouter selon les besoins. J'ai documenté précisément les branchements réalisés afin que le montage puisse être facilement reproduit sur de nouvelles cartes dans le futur.

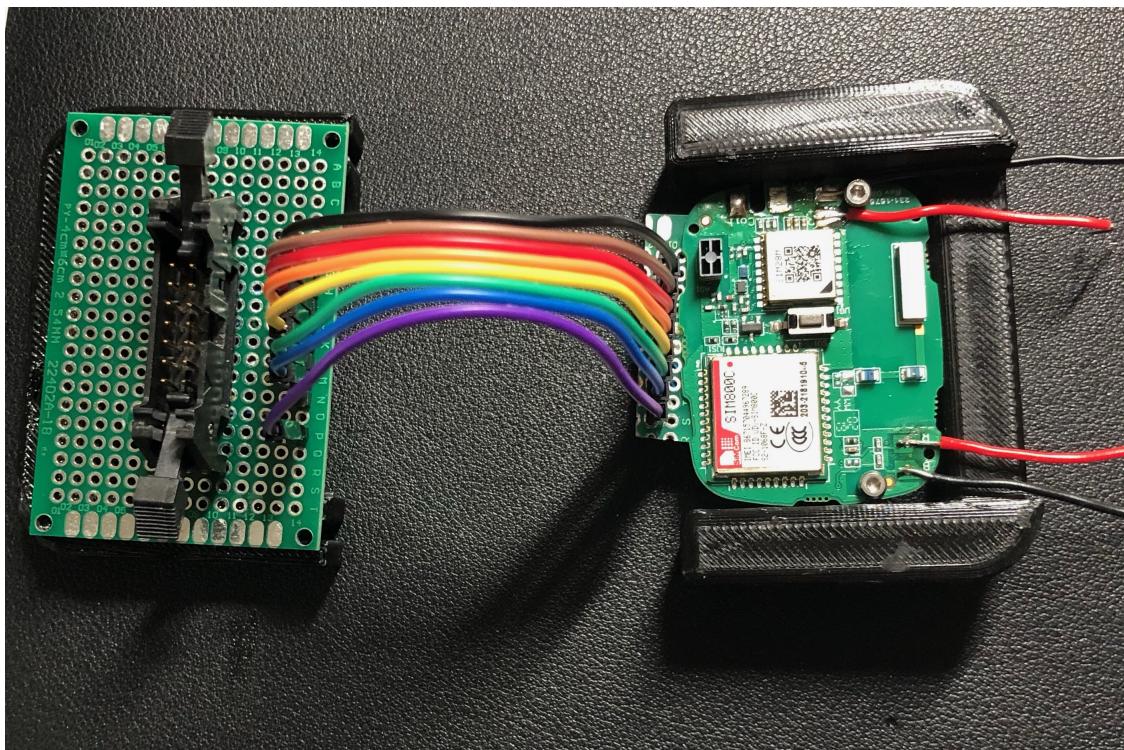


FIGURE 4.3 – Banc de programmation vu du dessus avec connecteur JTAG, carte Dial et support

Chapitre 5

Évolution et développement de la version 2.0 du software

Dans un premier temps, en attendant de recevoir les nouvelles cartes prototypes, j'ai commencé à utiliser l'IDE Code Composer Studio ainsi que les caractéristiques du microcontrôleur grâce au manuel utilisateur, à un kit de développement (<https://www.ti.com/tool/MSP-EXP430FR5969>) et à la datasheet. J'ai pu tester quelques programmes d'exemples afin de prendre en main le matériel et le code.

5.1 Restructuration et fiabilisation

Après la première analyse du code et après avoir échangé avec mon tuteur, nous sommes rapidement venus à la conclusion qu'il était préférable que je crée un nouveau projet avec une nouvelle structure afin de rajouter petit à petit les blocs de fonctions.

5.1.1 Le fonctionnement Foreground/Background

J'ai décidé de conserver l'architecture de fond du firmware existant. Un programme principal qui interagit avec des interruptions. Le passage à un mini OS temps réel a été évoqué, mais cela aurait impliqué une refonte totale du code. Étant seule sur le projet, compte tenu du temps et étant beaucoup plus à l'aise avec la gestion des interruptions, nous avons décidé de garder la base déjà existante. Cependant pour limiter les Conditions de course et les comportements problématiques il fallait respecter quelques règles. Tout d'abord le temps passé dans une routine d'interruption et leur nombre doivent être minimisés. Il faut prêter attention à la modification des variables globales.

5.1.2 Restructuration et nettoyage du code

En créant le nouveau projet j'ai aussi réorganisé les différentes couches logicielles afin de gagner en clarté et en efficacité [14]. Pour ce faire j'ai notamment ajouter la bibliothèque `driverlib` correspondante au MSP430FR5964 qui fournit des fonctions pour initialiser et utiliser les périphériques. L'avantage est que l'on gagne en lisibilité mais qu'en revanche cela utilise plus de place en mémoire, donc par moment il est plus judicieux de programmer directement via les registres. J'ai ensuite créé une couche `Dial_HAL` (Hardware Abstraction Layer) qui permet d'initialiser et d'utiliser que les périphériques dont on a besoin de façon spécifique dans le contexte de l'utilisation de la balise. La couche `Sources` regroupe les fonctions liées à l'utilisation des modules de plus haut niveau (GSM, GPS, accéléromètre...etc). Enfin le dossier `Application` regroupe les actions haut niveau (démarrer/éteindre la balise, envoi d'un message via https ou SMS ...). Le fichier `main.c` contient quant à lui une partie où l'on effectue l'initialisation, deux fonctions (initialisation du système d'horloge et passage en mode "sleep"), ainsi qu'une boucle infinie. C'est dans cette boucle infinie que sera gérer le fonctionnement de la balise.

Le but de structurer le code en différentes couches est de gagner en lisibilité (pas de code bas niveau dans le `main.c` par exemple), cela permet également de créer des blocs pouvant être modifiés et utilisés indépendamment. Le code est de ce fait plus simple à modifier et à faire évoluer.

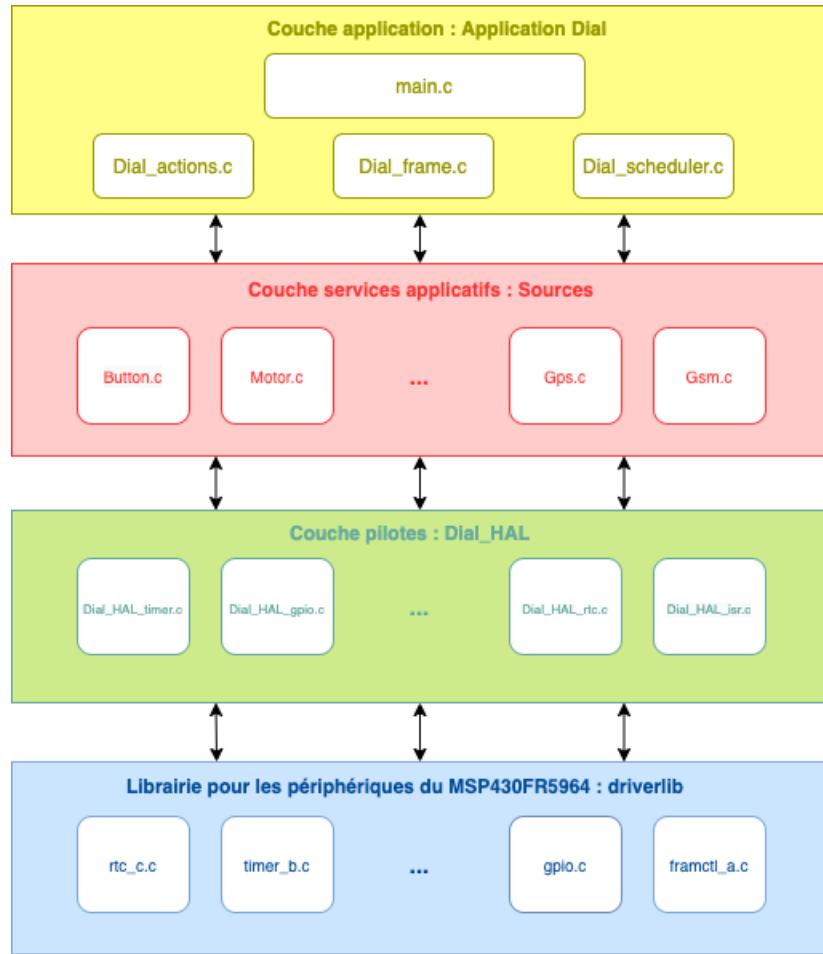


FIGURE 5.1 – Structure en couche du logiciel embarqué

J'ai donc recréé une nouvelle structure pour ensuite effectuer une bascule des différents fichiers existants. J'en ai aussi profité pour nettoyer le code, en effet de nombreuses fonctions étaient inutiles. De plus, une interface avec un PC Windows avait été codée pour permettre une lecture de l'état du système plus facile si l'on n'est pas initié aux outils de debug. N'ayant pas l'utilité de cette interface j'ai décidé de la supprimer, ce qui permettait de gagner à la fois en lisibilité et en espace mémoire.

5.2 Repenser le contrôle de la balise avec le bouton

Une fois le squelette du nouveau firmware mis en place j'ai pu commencer les développements. Le premier point que j'ai étudié est celui du contrôle de la balise, le but étant de rendre l'interface homme-machine plus intuitive et "user-friendly". En effet, avec l'alternante commerciale nous avons étudié les retours clients, l'avis qui revenait le plus fréquemment exprimait que le contrôle de la balise était trop complexe. La plupart précisait qu'ils étaient obligés de se référer à nouveau au guide utilisateur après une longue période sans avoir utilisé la balise.

5.2.1 Le fonctionnement existant et ses limites

Tout d'abord, l'interface entre la balise et l'utilisateur se compose de deux éléments :

1. Un bouton poussoir : permettant à l'utilisateur d'allumer/d'éteindre la balise et de déclencher ou de sortir du mode Alerte.
2. Un moteur vibrant : En fonction des vibrations permet d'indiquer à l'utilisateur dans quel état se situe la balise.

J'ai tout d'abord réalisé un diagramme d'état de la solution actuelle :

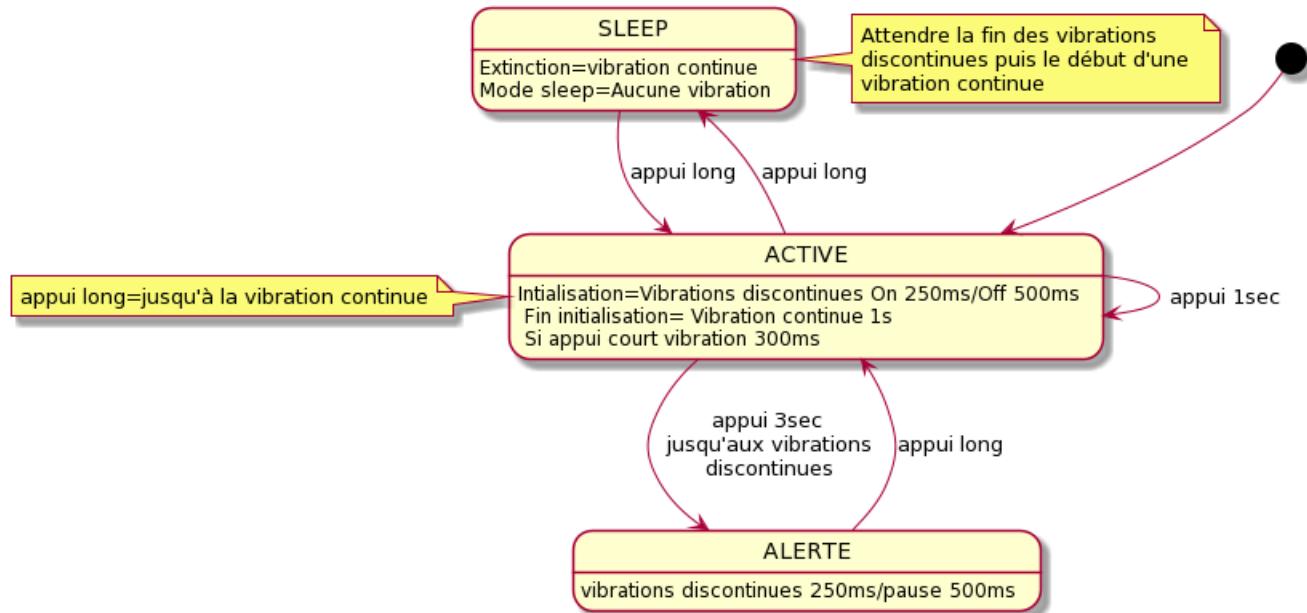


FIGURE 5.2 – Machine à états de la version Dial actuelle

Voici également un lien du site internet où se trouve deux vidéos permettant de comprendre le fonctionnement de la balise : <https://dial.snsm.org/dial-comment-ca-marche/> En effet ce fonctionnement n'est pas très intuitif car une même action (appui jusqu'à ressentir une vibration continue) permet de passer à trois états différents en fonction de l'état actuel de la balise. De plus, pour éteindre la balise l'utilisateur doit effectuer un appui long, ne pas relâcher lors des vibrations discontinues (sinon il passe en mode alerte), attendre la vibration continue pour arrêter d'appuyer. L'objectif de ma mission était de ce fait de proposer une nouvelle solution respectant à la fois le cahier des charges et les contraintes imposées par le hardware.

Pour rentrer brièvement dans les détails du code existant, le parti pris a été celui de scruter l'état du bouton toutes les 500 ms grâce à un timer même lorsque la balise est éteinte. Selon mon analyse ce fonctionnement présente deux limites principales. La première étant que la consommation du μ C n'est pas optimisée et la deuxième cela entraîne des comportements erratiques (l'appui n'est pas détecté systématiquement).

5.2.2 Le cahier des charges, les contraintes et la nouvelle machine à états

La première étape dans l'optique de fournir une solution adéquate a été de définir un nouveau cahier des charge. Cela a été fait suite à une réunion avec mon tuteur, nous avons pensé ce dernier à partir des scénarios d'utilisation. Les éléments du cahier des charges :

- Un type d'appui pour allumer et éteindre la balise : Trois appuis consécutifs avec une durée minimale entre chaque appui.
- Un type d'appui pour passer et sortir du mode alerte : Appui long avec des vibrations discontinues qui indiquent qu'il faut relâcher le bouton.
- Vibrations discontinues avec fréquence ajustable (utilisées en mode alerte et pendant la phase d'initialisation).
- Si la balise est allumée et hors mode alerte, un appui court déclenche une vibration.
- Vibration continue pendant la phase d'extinction.
- L'utilisateur ne peut pas éteindre la balise si elle est en mode alerte.

Cependant il a aussi fallu que je prenne en compte certaines contraintes imposées par le routage ou les périphériques. La plus contraignante étant que la sortie vers le moteur vibrant ne peut pas être une PWM¹ générée par un Timer, la fréquence des vibrations devaient donc être gérée "à la main" au niveau software. Il a fallu également que je prête attention à quels timers j'allais utiliser pour ne pas me retrouver bloquée lors de l'ajout de nouvelles fonctionnalités.

Voici la nouvelle machine à états que j'ai par la suite implémentée :

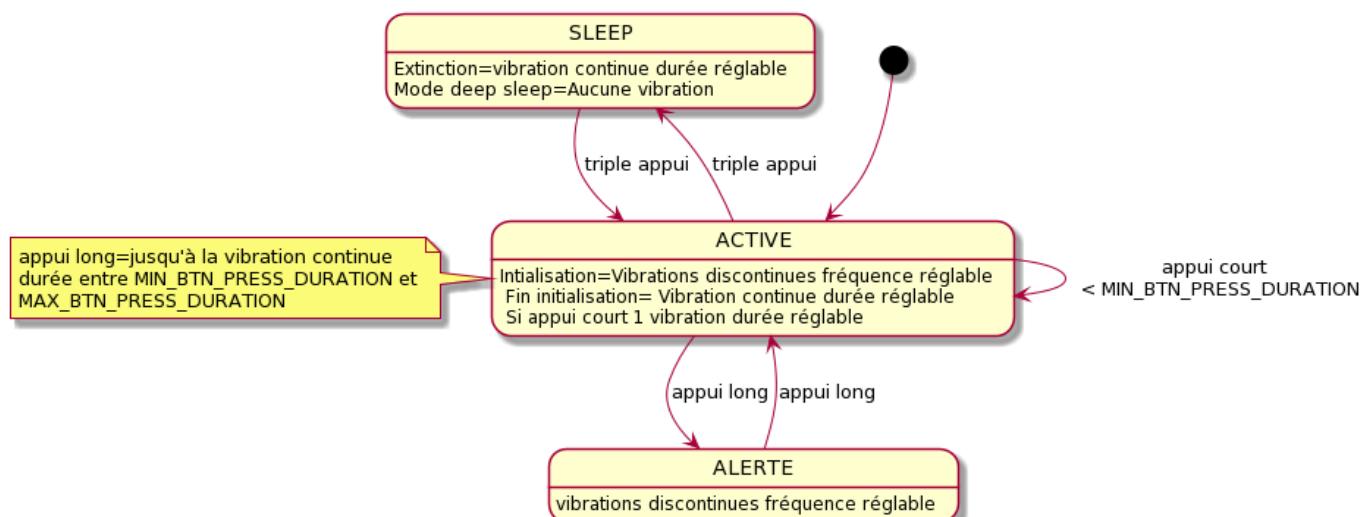


FIGURE 5.3 – Machine à états pour la version 2.0 de la balise

1. Pulse Width Modulation, c'est un signal dont le rapport cyclique varie

5.2.3 L'implémentation de nouvelles fonctions et tests

Comme évoqué précédemment j'ai pris le parti d'utiliser les interruptions. La différence avec la solution existante et qu'ici c'est l'appui sur le bouton qui déclenche une interruption mettant en route un timer permettant de détecter un appui entre 2 et 5 secondes (durées par défaut qui peuvent être modifiées), ou un triple appui.

J'ai crée des fonctions au niveau de la couche `Dial_HAL` pour initialiser les timers, ainsi qu'au niveau de la couche `Sources` pour la gestion du moteur et du bouton. J'ai implémenté la machine à états dans la boucle infinie grâce à deux "switch case" consécutifs, le premier gérant les transitions et le deuxième les actions à effectuer. Je me suis également efforcée d'implémenter des fonctions indépendantes, explicites, pouvant être facilement réutilisables et paramétrables en fonction des différents besoins.

J'ai ensuite réalisé une série de tests unitaires afin de valider les nouvelles fonctions. En somme, le code de la machine à états simplifie la gestion du bouton et est plus lisible. Les transitions entre chaque état peuvent être facilement changées si le cas d'usage évolue. La bibliothèque "Motor.c" fournit plusieurs fonctions déclenchant des vibrations permettant de communiquer avec l'utilisateur mais peut aussi servir d'outil de debug pour le développeur. En revanche, la gestion du moteur se faisant au niveau logiciel (et non pas en utilisant les sorties PWM des timers) complexifie légèrement le code car des variables globales (des "flags") modifiées par les routines d'interruption sont utilisées dans les fonctions des fichiers `Button.c` et `Motor.c`.

5.3 L'optimisation de la consommation

L'un des principaux enjeux des objets connectés réside dans la gestion de l'énergie afin de gagner en autonomie. L'un des problèmes rencontrés par les clients de Dial est que la balise se décharge très rapidement (un ou deux jours) même lorsqu'elle n'est pas utilisée par son propriétaire. Ma mission était d'analyser la consommation de la carte et de trouver des solutions pour augmenter l'autonomie de cette dernière.

La balise se compose d'une batterie Lithium Polymère rechargeable par induction d'une capacité de 420 mAh. L'entreprise communique les durées d'autonomie suivantes :

- 4 jours max avec une remontée de données GPS toutes les 20 minutes
- 2 jours max avec une remontée de données GPS toutes les 5 minutes
- 6 heures max (en mode alerte) avec une remontée de données GPS toutes les 15 secondes

L'enjeu de ma mission est donc d'estimer plus précisément cette consommation et dans la mesure du possible d'optimiser au maximum l'utilisation des différents périphériques.

5.3.1 Mise en place du mode Low Power sur le MSP430

Dans un premier temps, je me suis concentrée sur la minimisation de la consommation du microcontrôleur MSP430FR5964, qui a été conçu spécialement pour des applications faible énergie. En effet, il permet de choisir entre différents "Low Power Modes", ce qui a pour conséquence de plus ou moins réduire le nombre de périphériques disponibles et donc la valeur du courant d'alimentation.

Pour mieux comprendre comment j'ai choisi ces modes il est nécessaire d'expliquer le système d'horloge du microcontrôleur. Il se compose de trois signaux d'horloge, chacun de ces signaux provenant d'une source. Voici les signaux et leur sources utilisés dans le firmware :

- MCKL Master clock : 1MHz source -> DCO Internal Digitally Controlled Oscillator réglé à 8Mhz avec un prescaler qui divise par 8.
- SMCKL Sub-main clock : 8MHz source -> DCO Internal Digitally Controlled Oscillator réglé à 8Mhz
- ACLK Auxiliary clock : 32,768kHz source -> LFX Oscillateur basse fréquence

En fonction de ces low power certaines horloges ne sont pas disponibles, cela en fait donc un critère primordial à prendre en compte.

Table 1-3. Requested vs Actual LPM

Requested LPM (SR Bits according to Table 1-2)	Actual LPM...		
	If No Clock Requested	If Only ACLK Requested	If SMCLK Requested
LPM0	LPM0	LPM0	LPM0
LPM1	LPM1	LPM1	LPM1
LPM2	LPM2	LPM2	LPM0
LPM3	LPM3	LPM3	LPM1
LPM4	LPM4	LPM3	LPM1

FIGURE 5.4 – Extrait du manuel utilisateur du MSP430 présentant les horloges actives en fonction des Low Power Modes

Il a donc fallu à partir de ce tableau choisir le low power mode en adéquation avec les exigences d'horloge.

J'ai tout d'abord dû choisir le mode lorsque la valise est dans l'état « Sleep ». Le but étant de choisir celui qui consomme le moins pour diminuer au maximum la consommation du μ C. Ayant précédemment modifié le fonctionnement du bouton poussoir et des vibrations, c'est désormais l'appui sur le bouton qui génère une interruption et « réveille le système ». Aucun périphérique n'est en fonctionnement quand la balise est éteinte donc j'ai pu implémenter le mode de deepsleep le LPM4.5. Une information à prendre en

compte est que dans ce mode il n'y a pas de rétention de mémoire (la RAM n'est pas alimentée), il faut donc prêter attention à stocker les variables utiles dans la mémoire non volatile (FRAM). J'ai aussi choisi un mode low power dans lequel le système entre quand la balise est à l'état active ou alerte. En effet entre deux actions (relevé de la position GPS toutes les X minutes, relevé de l'accélération, envoi d'un message ...etc) le système pourrait entrer dans un mode sleep. J'ai dans ce cas d'usage choisi le mode LPM4 car j'ai besoin de conserver certaines informations en mémoire RAM.

En ce qui concerne les valeurs typiques du courant d'alimentation de ces deux modes les voici (conditions : à 25° et 3V et SVS² allumé) :

- $I_{LPM4.5,SVS} = 0.25\mu A$. On notera que cette valeur présentée dans un des tableaux de la datasheet est légèrement plus faible que celle présente dans le graphique ci-dessous aussi issue de la même documentation.
- $I_{LPM4,SVS} = 0.6\mu A$

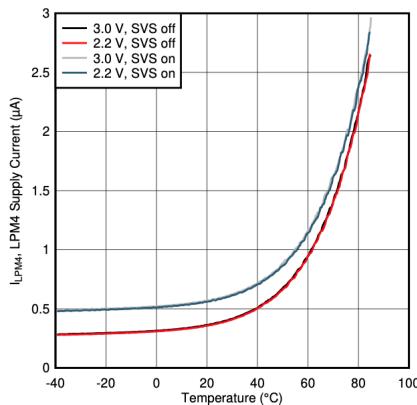


Figure 8-3. LPM4 Supply Current vs Temperature

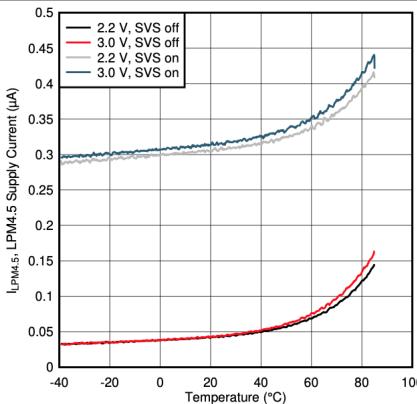


Figure 8-5. LPM4.5 Supply Current vs Temperature

FIGURE 5.5 – Extrait de la datasheet du MSP430FR5964, graphiques présentant le courant d'alimentation en LPM4 et LPM4.5 en fonction de la tension et de la température

Une fois ce choix arrêté, j'ai rajouté au code de la machine à états (cf Annexe 6) les

2. Supply Voltage Supervisor, permet de monitorer la tension d'alimentation et de sécuriser le microcontrôleur en cas de chute de tension ou de suralimentation.

instructions en C et j'ai effectué des tests pour valider le bon fonctionnement.

5.3.2 Initialisation des pins

Toujours dans la même optique de réduire au maximum la consommation du microcontrôleur et des autres périphériques je me suis attelée à initialiser les 48 pins correctement. Le but étant que lorsque la balise est en état SLEEP aucune sortie ne soit à l'état haut inutilement. Pour ce faire, j'ai utilisé les drivers fournis par TI ce qui permet de gagner en lisibilité. Toutes les pins non connectées ont été mises à zéro, tout comme les pins de réception et de transmission UART. En effectuant cette initialisation je me suis rendue compte que deux sorties devaient être tout de même réglées à l'état haut : celle qui contrôle l'alimentation du GPS et celle qui active le chargement par induction. Ces contraintes viennent de la conception hardware de la carte et je n'ai donc pas de marge de manœuvre du côté du software. J'ai par ailleurs, produit un tableau documentant le rôle de chaque pin et son initialisation. Cela m'a permis de mieux maîtriser la schématique de la carte ainsi que le fonctionnement des composants principaux.

5.3.3 Calcul théorique et mesures réelles

Après avoir implémenté les modes "Low Power" et initialisé les pins pour qu'aucun périphérique ne soit actif lorsque la balise est en "veille", j'ai décidé de quantifier la consommation pour avoir une idée plus précise de l'autonomie du système. J'ai donc tout d'abord effectuer une estimation théorique en m'appuyant sur les données fournies dans les datasheets des composants :

μ C MSP430FR5964 (à 80°C)
LPM4=1,6 μ A | LPM4,5=0,3 μ A

GSM SIM800C (pour VBAT=4V sachant qu'en réalité nous avons 3,7V)
Sleep mode = 1,5mA | Power Off = 150 μ A | Idle mode = 12,5mA | Peak Current 2A

GPS SIM28M (VCC=3,3V)
Acquisition=17mA | Continuous tracking= 16mA | Sleep=150 μ A

ACCELEROMETRE LIS3DHTR
Normal=11 μ A | Low Power = 6 μ A | PowerDown=0,5 μ A

MOTEUR
Courant nominal = 80mA | Courant au démarrage = 105mA

Capacité de la batterie = 420mAh

Le courant moyen théorique lorsque tous les périphériques sont éteints (balise à l'état sleep) :

$$I_{MoyThéorique} = 1.6\mu A + 150\mu A + 150\mu A + 0.5\mu A = 0.3mA$$

Estimation de la durée de vie de la batterie [11] :

$$Durée(h) = \frac{Capacité(mAh)}{Imoy(mA)} = \frac{420}{0.3} = 1390h = 57jours$$

Ce résultat ne reste cependant qu'une estimation car les valeurs provenant de la documentation ont été prises dans des conditions spécifiques qui ne sont pas les mêmes que celles du système Dial.

Ensuite, je suis passée à l'étape suivante, le relevé du courant consommé par la carte Dial. Pour ce faire j'avais en ma possession le MSP-Fet emulation development tool, qui est l'outil permettant de flasher le programme et d'accéder au Debug. Il permet aussi entre autre, grâce à la technologie EnergyTrace++ (développée par TI et disponible dans Code composer Studio) d'analyser la consommation, l'utilisation des périphériques ...etc. J'ai donc utilisé ces deux outils pour mesurer le courant moyen, voici les résultats :

	Courant moyen	Courant max	Courant min	Durée	Puissance moy
SLEEP (LPM4.5)	1,0256 mA	1,6906 mA	1,0113mA	18j	3,35mW
ACTIVE (LPM4)	1,0335mA	1,0500mA	1,0166mA	18j	3,37mW
ALERTE (LPM4+Motor)	3,3231mA	74,0911mA	0,8987mA	6j	10,87mW

TABLE 5.1 – Résultats des mesures de courant faite avec le MSP-Fet et EnergyTrace++

Concernant la précision des mesures et l'incertitude, la technologie EnergyTrace permet d'obtenir pour des courants inférieurs à 25mA (et VBUS = 5V constant) une précision de $\pm 2\%$ ou de $\pm 500nA$.

5.3.4 Analyse, limites rencontrées et pistes de réflexion

Après avoir effectué les mesures sur la carte, j'ai pu observer qu'il y avait une différence non négligeable (0.7 mA d'écart) entre la valeur théorique et les valeurs mesurées. Tenter d'expliquer cet écart est assez difficile car je n'avais pas les ressources nécessaires pour calculer et vérifié la consommation de chacun des périphériques. Cependant, l'accéléromètre et le GPS n'étant pas alimentés seul le GSM et le microcontrôleur pourraient être à l'origine de cette surconsommation. J'ai relevé la présence d'une tension alternative (et

donc de la présence d'un potentiel courant) circulant entre certaines pins reliant le GSM au μ C. Cependant l'origine de ce phénomène m'échappe encore n'ayant pas de visibilité sur la conception interne du GSM. Pour pallier à ce phénomène j'ai exploré la piste du changement de la batterie pour augmenter sa capacité. Le problème que cela pose est qu'en augmentant la capacité, la taille de la batterie augmenterait également et ne logerait donc plus dans la coque en plastique qui garantit l'étanchéité.

Les seules recommandations à suivre sont de développer un programme qui minimise l'utilisation du GSM, du GPS et du moteur qui représentent les plus gros postes de consommation de la carte. De plus, TI fournit un outil, l'ULP Advisor, qui permet de faire de nombreuses vérifications au niveau du microcontrôleur permettant de minimiser les ressources utilisées. Pour la suite de mes développements j'ai donc suivi ces préconisations.

Par manque de moyens, étant limité par la conception hardware et car je devais commencer d'autres développements plus prioritaires, je n'ai pas pu trouver une solution permettant de diminuer significativement la consommation de la carte. Je pense que pour avoir un véritable impact il aurait fallu repenser la conception hardware et étudier la possibilité de choisir un nouveau GSM car le modèle actuel est réputé pour être très gourmand en énergie. Après une discussion avec mon tuteur il semblerait que ces considérations de consommation n'aient pas été prises en compte dès le début de la conception de l'objet par les bureaux d'études, de ce fait sans concevoir un nouveau prototype, la marge de manœuvre, seulement limitée au logiciel, est assez mince.

5.4 Bootloader pour une mise à jour "Over The Air"

La balise Dial est étanche, le système qui permet cela est composé de deux coques en plastique qui sont soudées entre elles. Cela garantit l'intégrité du système même si il est immergé, cependant une autre contrainte apparaît. En effet, étant donné qu'aucun contact physique avec la carte n'est possible, la mise à jour du logiciel embarqué ne peut pas être faite via une liaison physique. Il y a donc un véritable enjeu à ce niveau là. Avoir la possibilité de mettre à jour son logiciel est un avantage considérable que cela soit pour renforcer la sécurité, ajouter des fonctionnalités ou pour corriger des bugs. Une solution compatible avec le MPS430FR5969 existait déjà, cependant étant donné que pour avoir plus de place en mémoire ce dernier a été remplacé, mon objectif a donc été de faire les modifications nécessaires afin que la fonctionnalité de mise à jour soit fonctionnelle sur le nouveau μ C choisi.

5.4.1 Le fonctionnement de la solution existante

Pour pouvoir effectuer (ou non) la mise à jour du logiciel, un bootloader dual image sur mesure a été codé. Il s'agit du premier code que le μ C va exécuter et qui va donc déterminer si on lance la mise à jour ou directement l'application. Tout d'abord, les grandes étapes du code du bootloader sont les suivantes : Après le dépôt de la nouvelle image du firmware cryptée sur un serveur (via l'exécution d'un script python). Si le programme du bootloader détecte qu'il faut faire une mise à jour :

- Connexion du GSM au serveur via le protocole https.
- Récupération des fichiers, envoi des informations du GSM vers le μ C via l'uart.
- Décryptage et copie de la nouvelle image dans une zone tampon.
- Vérification que toutes les données ont bien été transmises par contrôle de redondance cyclique (CRC).
- Si tout est validé recopie de l'image à la place de l'ancienne .
- Lancement de la nouvelle application.

La solution déjà développée par un bureau d'études au début de la conception du produit, n'a pas pu être mise en place sur les cartes Dial industrialisées en 2019 à cause de la mémoire trop limitée et d'un manque de temps. Par la suite, un stagiaire en interne avait repris le code et l'avait modifié dans le but l'améliorer et d'optimiser la détection et le lancement de la mise à jour. Je ne vais pas détailler cette partie car elle n'est pas au cœur de ma mission, j'ai du la comprendre pour pouvoir l'utiliser mais je n'ai pas apporté de modifications à ce niveau là.

La spécificité de ce bootloader est qu'il a été fait sur mesure notamment au niveau du mapping mémoire. La mémoire RAM de base étant sous dimensionnée un nouveau mapping a été effectué pour permettre de transformer de la mémoire non volatile (FRAM) en RAM. Ce sous dimensionnement cause deux problèmes :

- Le code du bootloader ne doit pas être trop lourd.
- Une partie de la RAM est écrasée lors de la mise à jour pour servir de zone tampon.

Une autre particularité est que pour la gamme des MSP430FR59XX la table des vecteurs d'interruptions qui redirige le programme vers les routines est unique. Cette table se doit d'être la même pour l'application et pour le bootloader, de ce fait il a fallu s'assurer de ne pas modifier les routines d'interruption utilisées par l'application.

5.4.2 "Dual image" et re mapping mémoire

Le problème du manque de place en mémoire étant résolu, en choisissant un nouveau μ C le MSP430FR5964, je devais effectuer un nouveau mapping mémoire pour le bootloader et pour l'application. C'est le fichier nommé "linker" avec l'extension .cmd qui gère

la compartimentation de la mémoire. Après la compilation on peut retrouver le mapping mémoire détaillé dans le fichier .map. Il contient notamment les fonctions, les variables, les constantes ainsi que leurs adresses de stockage.

Le but était de ne pas avoir à effacer quoique ce soit lors de la copie des données en zone tampon, la place étant suffisante. J'ai commencé par effectuer des recherches dans les ressources de Texas Instrument pour trouver une base de projet. J'ai finalement trouvé une structure d'un bootloader dual image d'un côté, ainsi qu'un script python permettant de générer les fichiers linker de l'application et du bootloader à partir du linker par défaut de l'autre. Ce script m'a donc permis de générer les deux fichiers linker en fonction de la taille du bootloader souhaitée (0x1000 le code étant conséquent) et du mode (Dual image). Une fois cela fait, j'ai du effectuer quelques modifications à la main et j'ai du rajouter les emplacements mémoires des signatures (JTAG notamment) qui sont utilisées pour sécuriser le firmware. Une fois que les fichiers linker furent intégrés à la base du projet bootloader et application j'ai adapté le code du bootloader pour qu'il fonctionne avec le nouveau mapping mémoire.

Dénomination	Variables TI_MSPBoot_MI.h	Variables du fichiers mapping.h	Adresses	
APP_START_FRAM1	APP_START_ADDR	ADR_MIN_FRAM1	0x4000	Size FRAM1 = 0x9F80 >Actual Size FRAM1_CODE = 0x2390
APP_END_CODE_FRAM1		ADR_MAX_FRAM1	0xDF7F	
			0xDF80	
			0xDF84	
			0xDF88	
			0xDF90	
APP_RESET_VECT	APP_RESET_VECTOR_ADDR		0xFFFE	
APP_END_FRAM1	APP_END_ADDR		0xFFFF	
BOOT_START	BOOT_START_ADDR		0xE000	Size BOOT_CODE = 0x1F70 >Actual Size BOOT_CODE = 0x1E1E
BOOT_END_CODE			0xFF6F	
BOOT_SHARED_CALLBACKS			0xFF70	
BOOT/APP_JTAG_SIGNATURE		ADR_MIN_INT_VECT	0xFF80	
BOOT/APP_BSL_SIGNATURE			0xFF84	
BOOT/APP_IPE_SIGNATURE			0xFF88	
BOOT/APP_INT_VECT	BOOT_VECTOR_TABLE		0xFF90	Size INT_VECT = 0x80
BOOT_RESET_VECT		ADR_MAX_INT_VECT	0xFFFFE	
BOOT_RESET_VECT_END			0xFFFF	
APP_START_FRAM2	FLEX_START_ADDR	ADR_MIN_FRAM2	0x10000	Size FRAM2= 0x14FFC >Actual Size FRAM2_CODE = 0x4000
APP_END_CODE_FRAM2	FLEX_END_ADDR	ADR_MAX_FRAM2	0x24FFB	
DOWN_START_FRAM1		ADR_MIN_TAMPON_FRAM1	0x24FFC	Size FRAM1 = 0x9F80
DOWN_END_CODE_FRAM1		ADR_MAX_TAMPON_FRAM1	0x2EFTB	
DOWN_START_INT_VECT		ADR_MIN_TAMPON_INT_VECT	0x2EF7C	Size INT_VECT = 0x80
DOWN_END_INT_VECT		ADR_MAX_TAMPON_INT_VECT	0x2EFFB	
DOWN_START_FRAM2		ADR_MIN_TAMPON_FRAM2	0x2EFFC	Size FRAM2= 0x14FFC
DOWN_END_FRAM2		ADR_MAX_TAMPON_FRAM2	0x43FF7	

TABLE 5.2 – Mapping de la mémoire du MSP430FR5964 incluant un bootloader dual image

Une fois ce mapping réalisé, les fichiers basculés dans le nouveau projet et les modifications du code réalisées, je suis passée à la phase de test. Cette phase a été plutôt rapide car grâce à l'environnement de Debug de l'IDE qui offre la possibilité d'explorer la mémoire j'ai pu rapidement vérifier que les informations étaient copiées au bon endroit. J'ai donc pu fournir à la start up une nouvelle version d'un bootloader Dual Image fonctionnelle. Cette mission a été particulièrement ardue, n'ayant que très peu de connaissances au début. Cette mission constitue une grosse partie de mon stage, cela m'a permis de maîtriser

une bonne partie de ce qui concerne la mémoire du microcontrôleur ainsi que la notion de bootloader.

5.5 Envoi périodique de la position GPS

À ce stade, au niveau software, j'avais pour le moment : créé une nouvelle organisation en couches du firmware ; repensé et implémenté le contrôle de la balise via le bouton poussoir ; développé une librairie de fonctions pour la gestion du moteur vibrant ; remappé la mémoire du microcontrôleur afin d'adapter la solution du bootloader permettant la mise à jour à distance du firmware. Arrivant à la fin de mon stage j'ai donc entrepris le développement d'une nouvelle fonctionnalité. Je me suis donc penchée sur la récupération et l'envoi de la position GPS de la balise. La gestion de cet envoi (qui est assez conséquente) se situe dans l'interruption liée à la réception d'un caractère via l'UART. Chaque caractère reçu déclenche l'interruption, met dans la file l'évènement de gestion de la réception des données GPS qui est donc appelé à chaque fois qu'un caractère est reçu. Ce fonctionnement est complexe et le comportement est difficile à prévoir, le simplifier permettrait de gagner en clarté et en fiabilité.

5.5.1 Nouvelle gestion du temps avec le périphérique Real Time Clock

Pour envoyer les alertes ou pour remonter la position périodiquement certaines données sont indispensables, notamment la position mais aussi la date et l'heure précise. Elles sont nécessaires pour que les secours puissent effectuer la levée de doute et pour qu'ils puissent intervenir dans les meilleurs délais et conditions si besoin. Dans l'état actuel, l'heure est initialisée via le GPS puis toutes les deux secondes un flag est levé dans l'interruption d'un timer et met dans la file d'évènement un ensemble d'actions à réaliser toutes les deux secondes dont la mise à jour du temps. Cette mise à jour est donc fait "à la main" au niveau logiciel. Après avoir analysé ce fonctionnement j'en ai déduit qu'il pourrait être grandement simplifié. En effet en utilisant le périphérique Real Time Clock (RTC) du MSP430 cette gestion du temps serait faite au niveau hardware et on gagnerait également en précision. Le RTC possède un mode calendrier, une fois initialisé, grâce à l'oscillateur à 32.768kHz il met à jour les registres correspondants à la date et à l'heure même lorsque le μ C est dans un low power mode. Il est également possible de déclencher différents types d'interruptions : toutes les secondes, toutes les minutes, à une heure précise ...etc. Dans notre cas il a tout d'abord fallu commencer par initialiser les registres. Grâce à une commande AT (langage de commande utilisé dans la plupart des modems GSM), il est possible de récupérer la date et l'heure. On récupère alors un tableau de caractère qu'il faut ensuite convertir en code hexa afin de pouvoir mettre les valeurs dans les registres du

RTC. La conversion entre les types de données a été la partie la plus complexe. Une fois cela fait il suffit juste de lancer l'horloge du RTC et d'activer les interruptions souhaitées. Ci-dessous la fonction en langage C qui initialise le calendrier du RTC :

```

1
2     void DialHAL_rtc_SetCalendar( void ){
3
4
5         // Get Data and Time From the GSM
6         GSM_LocTime();
7
8         RTCCTL0_H = RTCKEY_H;
9         RTCCTL13 = RTCBCD | RTCHOLD | RTCMODE;
10
11        // Conversion du temps récupéré via le GSM du format ascii au format
12        // hexadécimal pour initialiser le RTC
13        u8 i=0;
14        u8 timeInHex[6]={0};
15        for (i=0; i<6;++i){
16
17            timeInHex[i]=byte_ascii_toHex(gsm_save_data.time[i]);
18
19
20        // Conversion de la date récupérée via le GSM du format ascii au format
21        // hexadécimal pour initialiser le RTC
22        u8 j=0;
23        u8 dateInHex[8]={0};
24        for (j=0; j<8;++j){
25
26            dateInHex[j]=byte_ascii_toHex(gsm_save_data.date[j]);
27
28
29
30        RTCYEAR =(dateInHex[4]<<12) | (dateInHex[5]<<8) |( dateInHex[6]<<4) |
31        dateInHex[7] ; // Year
32        RTCMON = (dateInHex[2]<<4) | dateInHex[3]; // Month
33        RTCDAY = (dateInHex[0]<<4) | dateInHex[1]; // Day
34        RTCDOW = 0x00; // Day of week
35        RTCHOUR =(timeInHex[0]<<4) | timeInHex[1]; // Hour
36        RTCMIN = (timeInHex[2]<<4) | timeInHex[3]; // Minute
37        RTCSEC = (timeInHex[4]<<4) | timeInHex[5]; // Seconds
38

```

```

39     RTC_C_clearInterrupt(RTC_C_BASE,RTC_C_CLOCK_READ_READY_INTERRUPT +
40     RTC_C_TIME_EVENT_INTERRUPT +RTC_C_CLOCK_ALARM_INTERRUPT) ;
41
42     RTC_C_enableInterrupt(RTC_C_BASE, RTC_C_TIME_EVENT_INTERRUPT) ;
43
44
45
46
47 // Start RTC Clock
48 RTC_C_startClock(RTC_C_BASE) ;
49 }
```

Après plusieurs tests le fonctionnement de cette fonction a été validé.

5.5.2 Récupération de la position GPS et prochaines actions à mener

L'utilisation du RTC permet donc de gérer le calendrier ainsi que de déclencher périodiquement une interruption (toutes les minutes). L'objectif final étant que la position GPS soit relevée et envoyée toutes les X minutes. Pour cela il faut tout d'abord récupérer les coordonnées GPS. Ces données sont contenues dans la trame GPPGA (correspondant au standard NMEA³), il faut donc dans un premier temps récupérer cette trame. Cette trame est complète uniquement lorsque le GPS a réalisé son premier fix, il faut donc vérifier que le nombre de caractère reçu est supérieur à 50. C'est ce que la fonction que j'ai créé `u8 gps_get_GPGGA_frame(void)` effectue. C'est à ce niveau que j'ai arrêté les développements logiciel car mon stage touchait à sa fin. La fonction d'envoi des informations via https ou SMS étant déjà implémentée la prochaine brique à modifier est la fonction qui parse la trame GPPGA pour ajouter les données dans la structure `t_gps_NMEA_data`. Il restera ensuite à assembler les différentes fonctions, ajouter un compteur software pour régler la périodicité des données et d'intégrer cette fonctionnalité à la machine à états.

³. La norme NMEA 0183 est une spécification pour la communication entre équipements marins, dont les équipements GPS. Elle est définie et contrôlée par la National Marine Electronics Association.

Chapitre 6

Conclusions et perspectives

Pour conclure, ce stage au sein de la start-up Ido-data constitue l'aboutissement de mes cinq années de formation à l'INSA. Lors de ces six derniers mois j'ai eu la possibilité de réaliser diverses missions. De l'analyse du produit et de sa conception, au changement de composants, à la réalisation d'un banc de programmation, en passant par la mise en place d'évolutions pour le logiciel embarqué, j'ai eu la chance de pouvoir explorer plusieurs facettes et problématiques liées à la conception d'un système embarqué faible énergie du domaine de l'IoT.

Tout en me basant sur les enseignements que j'ai reçus, j'ai essayé d'apporter des analyses pertinentes, des outils pour l'analyse fonctionnelle ainsi qu'une documentation complète sur mes réalisations pour que mon travail puisse être réutilisé. Ido-data dispose suite à mon stage : de cartes avec un nouveau microcontrôleur ayant plus de mémoire ; d'un bootloader fonctionnel qui a été adapté à ce dernier ; d'un banc de programmation ; d'un firmware restructuré avec des fonctionnalités repensées et optimisées en terme d'énergie. Je me suis également efforcée d'appliquer une méthode de travail pour être efficace et pour toujours garder une image claire de mes objectifs à atteindre.

Je suis globalement satisfaite de mes réalisations, cependant j'ai été confrontée à certains freins. Étant la seule personne à travailler sur le projet, j'ai par moment fait face à un sentiment de solitude. La situation sanitaire et le télétravail n'ont pas favorisé, au début, la création de liens avec mes collègues. J'ai subi quelque fois une perte de motivation. Cependant, j'ai su me remobiliser et j'ai au final réussi à accomplir mes missions. J'ai pu retirer de très nombreuses leçons de cette expérience. Lors de ce stage j'ai eu l'entièvre confiance de mes supérieurs qui m'ont laissé travailler en autonomie. Cela m'a permis d'apprendre de façon durable et d'acquérir de nouvelles connaissances. J'ai pu développer une compétence de débrouillardise, et cela m'a également permis de prendre confiance en mes choix et en mes décisions. J'ai pu relever cependant que lorsqu'on travaille seul il est difficile à certains moments de prendre du recul sur le contenu que l'on produit ou sur

les problèmes auxquels l'on est confrontés. J'ai souvent eu besoin de mettre en pause un travail, pour essayer de l'aborder différemment la fois d'après. J'ai aussi suggéré à mes supérieurs d'ouvrir un deuxième sujet de stage autour de ces problématiques afin qu'au moins deux personnes soient impliquées à temps plein, ce qui à mon sens, permettrait de créer plus de synergie, de confronter les idées, et de débloquer des situations plus rapidement.

En ce qui concerne l'impact de ce stage sur mon avenir professionnel et mes choix futurs, tout d'abord cela m'aura permis de me rendre compte de mon désir de détacher un peu de la programmation pour me rapprocher de la conception hardware et des tests matériels. J'ai aussi l'envie de continuer de me former et d'évoluer au sein d'une équipe expérimentée qui développe des projets concrets qui auront du sens pour moi. Cette expérience dans une start-up m'a aussi permis de me rendre compte de l'envers du décors de l'entrepreneuriat, des avantages et aussi des difficultés que cela pouvait engendrer.

Je remercie de nouveau toute l'équipe d'Ido-data de m'avoir accueillie et de m'avoir permis de vivre une expérience enrichissante concluant ainsi ma formation d'ingénierie INSA.

Annexes

Annexe : Présentation de la société

- Société : Ido-data
- Siège social : Le Mix – 4 et 6 avenue Joannes Hubert 69160 Tassin-la-Demi-Lune
- SAS capital 11200€
- 7 salariés
- Mail : hello@ido-data.fr
- Téléphone : +33 (0)4 28 29 61 45
- Code NAF : 6201Z Programmation Informatique
- Site internet : <https://ido-data.fr/>

Pendant la période de mon stage les bureaux de la start-up Ido-Data été situés au sein de Village by CA (Crédit Agricole) à Champagne-Au-Monts-D'or (69410).

Ido-Data est une startup fondée par Yannick Tocquet, Thomas Creveaux et Antonin Carlesso en 2017 dont le but est de mettre à disposition un savoir faire dans les domaines de l'IoT et du digital afin de proposer des solutions innovantes visant à renforcer la sécurité et la prévention des risques. De la définition des cas d'usage au maintien en condition opérationnelle en passant par le développement et la phase de tests, Ido-data accompagne des professionnels en charge de la prévention, de la sécurité, du sauvetage et des premiers secours. L'équipe est composée de 7 personnes :

- Antonin Carlesso : Co-fondateur | Design global
- Yannick Tocquet : Co-fondateur | Partenariat et innovation IoT
- Thomas Creveaux : Co-fondateur | Développeur full stack
- Rufine Bouteille : Alternante | Commerciale
- Paola Mbia Messi : Alternante | Développeuse Web et Mobiles
- Cornelia Allagnon : Alternante | Office Manager
- Florencia Alipaz : Stagiare | Marketing internationale

Annexe : Caractéristiques techniques de la balise Dial



Caractéristiques techniques

INTITULÉ	CARACTÉRISTIQUES
DIMENSIONS DE LA BALISE	Longueur 53mm x Largeur 43,6mm x Hauteur 21,6mm
BRACELET	Taille : 270 mm Tour de poignet ⁽¹⁾ : 135 mm à 210 mm Système d'attache renforcé avec deux boutons de col et un passant
POIDS	Balise : 42 grammes Balise + Bracelet : 80 grammes
TYPE DE BATTERIE	Batterie Lithium Polymère non-replaçable (certifiée UN 38.3 et IEC 62133) tension : 3,7 V - capacité : 420 mAh. Rechargeable par induction (chargeur fourni).
AUTONOMIE	-4 jours max avec une remontée de données GPS toutes les 20 minutes -2 jours max avec une remontée de données GPS toutes les 5 minutes -6 heures max (en mode alerte) avec une remontée de données GPS toutes les 15 secondes
ÉTANCHÉITÉ	IPX6 (protégé contre les forts jets d'eau de toutes directions à la lance (buse de 12,5 mm, distance 2,5 m à 3 m, débit 100 l / min ±5 %)) et IPX8 (étanche à 20 m pendant 4 heures)
TYPE DE SIGNAUX	GPS en réception pour déterminer la localisation GPRS et GSM pour transmettre les informations sur les réseaux mobiles via internet et SMS.
RÉSEAU MOBILE	Carte SIM multi-opérateurs (inclusa dans la balise), fonctionne dans 32 pays Européens & DOM ⁽²⁾ .
BANDE DE PUISSANCE ET FRÉQUENCE	880-915 MHz
DÉBIT D'ABSORPTION SPÉCIFIQUE (DAS)	Sur la bande de fréquence : GPRS 900 : 0,171 W/kg (sur canal 124) GPRS 1 800 : 0,266 W/kg (sur canal 885)

⁽¹⁾ Il est de la responsabilité des parents d'expliquer l'utilisation du produit aux enfants. Le produit n'est pas conseillé pour les enfants dont le tour de poignet est inférieur à 135 mm. Si vous souhaitez utiliser DIAL pour un mineur ne disposant pas de smartphone pour créer un compte personnel, veuillez contacter le support à l'adresse support@dial.snm.org.

⁽²⁾ Europe : Allemagne, Autriche, Belgique, Bulgarie, Chypre, Croatie, Danemark, Espagne (y compris les Canaries), Estonie, Finlande, France, Grèce, Hongrie, Irlande, Islande, Italie, Lettonie, Liechtenstein, Lituanie, Luxembourg, Malte, Norvège, Pays-Bas, Pologne, Portugal, République Tchèque, Roumanie, Royaume-Uni, Slovaquie, Slovénie, Suède, Suisse. DOM : Réunion, Mayotte, Guyane française, Guadeloupe, Martinique, Saint-Barthélemy, Saint-Martin.

FIGURE 6.1 – Caractéristiques techniques de la balise Dial

Annexe : Matériel électronique acheté pour la start-up

Fil de soudure : MULTICORE (SOLDER) 309 99C 5C 1.2MM H 2M, Sans plomb, Diamètre 1.2mm, 227°C

Prix : 0,5€

Tresse à dessouder : CHEMTRONICS 80-1-5 Tresse à dessouder, cuivre, 5 pieds x 0.9 mm

Prix : 2,14€

Sonde point de test : POMONA 5519A Cordon de test, Sonde pointe de test à Fiche banane Angle Droit 4mm 1 kV, 10 A, 1.22 m, 2 Broches

Prix : 16,32€

Multimètre : TENMA 72-13510 DMM, HANDHELD, TRUE RMS, 10A, 600V

Prix : 72,16€

Loupe : LIGHTCRAFT LC8085USB Loupe LED, USB, de table, Grossissement 1.75x

Prix : 31,44€

Fer à souder : WELLER WP80 + WDH10 Fer à souder, Avec support, 24 V, 80 W, 350 °C

Prix : 152,15€

Alimentation fer : PU 81 EU 450°C, 80W, 240 V

Prix : 123€

Picoscope (Oscillo) : PICO TECHNOLOGY PICOSCOPE 2208B Oscilloscope PC USB, Déclenchement numérique, Série PicoScope 2000, 2 canaux, 100 MHz, 1 GSPS

Prix : 579€

Annexe : Algorigramme et interruptions du software Dial actuel

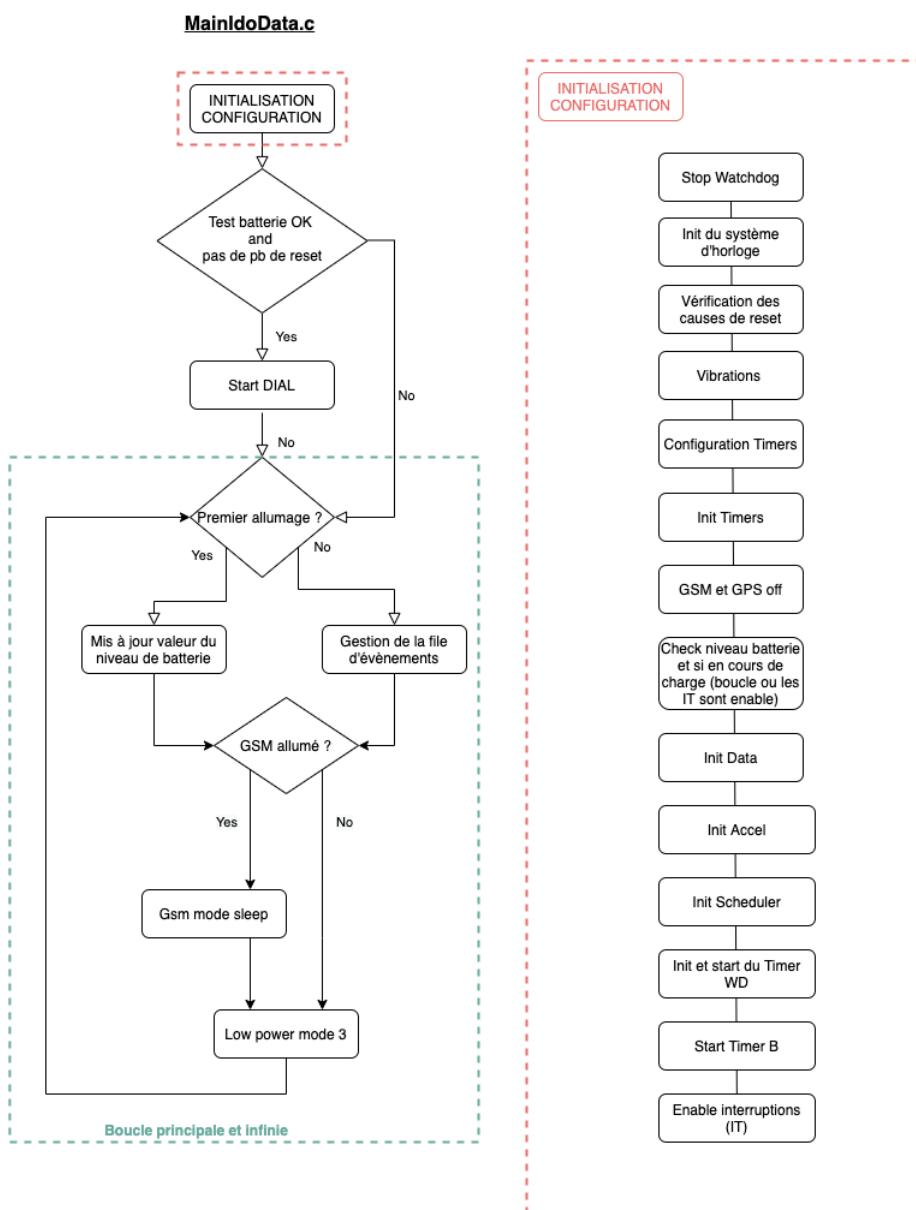


FIGURE 6.2 – Algorigramme du software Dial

Les interruptions

A la suite d'un évènement, saut du programme principal vers une routine spécifique (handler)

2 sources pouvant déclencher une interruption :

- internes, provenant directement en interne des périphériques du µcontrôleur (dépassement timer, ADC ...)

- externes, provenant d'un périphérique ouvert sur l'extérieur (changement état GPIO, réception données...)

HANDLERS



FIGURE 6.3 – Interruptions présentes dans le software Dial

Annexe : Code source du squelette de la machine à états

```
1 //_____ Begin State Machine _____
2
3 //Process pour les transitions qui permettent la mise à jour de l'état
4
5     switch (currentState)
6     {
7         case ACTIVE :
8             if(Button_Detect_Long_Press())
9                 currentState=ALERT;
10            else if (Button_Detect_Triple_Press())
11            {
12                currentState=SLEEP;
13                lastState=ACTIVE;
14                Motor_Multiple_Vibrations(1,100);
15            }
16            break;
17
18        case ALERT :
19            if(Button_Detect_Long_Press())
20            {
21                Motor_Stop_Discontinuous_Vibrations();
22                currentState=ACTIVE;
23            }
24            break;
25
26        case SLEEP :
27            if(Button_Detect_Triple_Press())
28            {
29                currentState=ACTIVE;
30                Motor_Multiple_Vibrations(1,100);
31            }
32
33            break;
34    }
```

```

35
36
37
38 // Process pour la réalisation d'actions selon l'état
39
40     switch (currentState)
41     {
42         case ACTIVE :
43             Button_Check_Active_State() ;
44             break;
45
46         case ALERT :
47             Motor_Start_Discontinuous_Vibrations(
48                 DEFAULT_DISCONTINUOUS_VIBRATION_ON_PERIOD,
49                 DEFAULT_DISCONTINUOUS_VIBRATION_OFF_PERIOD) ;
50             break;
51
52         case SLEEP :
53             System_Go_To_Deep_Sleep() ;
54
55     }
56
57 //_____ End State Machine _____
58
59
60
61     __low_power_mode_4() ;
62     __no_operation() ;
63
64
65
66
67 } // end infinite loop

```

Table des figures

1.1	Logo Dial	1
2.1	Bracelet contenant la balise et page d'alerte de l'application Dial	4
2.2	Mindmap des objectifs de mon stage	5
2.3	Logo Ido-data	7
4.1	Principales caractéristiques des réseaux LTE-M et Nb-IoT	17
4.2	Comparatif des microcontrôleurs MPS430 de chez TI	19
4.3	Banc de programmation vu du dessus avec connecteur JTAG, carte Dial et support	21
5.1	Structure en couche du logiciel embarqué	24
5.2	Machine à états de la version Dial actuelle	25
5.3	Machine à états pour la version 2.0 de la balise	26
5.4	Extrait du manuel utilisateur du MSP430 présentant les horloges actives en fonction des Low Power Modes	28
5.5	Extrait de la datasheet du MSP430FR5964, graphiques présentant le courant d'alimentation en LPM4 et LPM4.5 en fonction de la tension et de la température	29
6.1	Caractéristiques techniques de la balise Dial	42
6.2	Algorigramme du software Dial	44
6.3	Interruptions présentes dans le software Dial	45

Liste des tableaux

4.1	Configuration mémoire du MSP430FR5969 avec le code du firmware et du bootloader	18
5.1	Résultats des mesures de courant faite avec le MSP-Fet et EnergyTrace++	31
5.2	Mapping de la mémoire du MSP430FR5964 incluant un bootloader dual image	34

Bibliographie

- [1] Jorf n°25 du 30 janvier 2005 page 1625 - texte n°43 - vocabulaire des sciences et techniques spatiales. <ur\protect\unhbox\voidb@x\protect\penalty\@M\https://www.legifrance.gouv.fr>, 2005. [Online ; accessed 26-July-2021].
- [2] Akyga. Datasheet batterie lp652730. <https://www.elektronik.ropla.eu/fr/magazyn/magazyn/?ic=AKY0101>, 2013. [Online ; accessed April-2021].
- [3] Elham FIROUZI. Les micro-contrôleurs dans les systèmes embarqués. <https://embarque.developpez.com/cours-tutoriels/programmation/systeme-embarque/introduction-microcontroleur/#LIV-C-6>, 2011. [Online ; accessed May-2021].
- [4] Texas Instrument. Datasheet msp430fr5969. <https://www.ti.com/document-viewer/MSP430FR5969/datasheet/device-overview-slas7047447#SLAS7047447>, 2012. [Online ; accessed March-2021].
- [5] Texas Instrument. User's guide msp430fr59xx. https://www.ti.com/lit/ug/slau367p/slau367p.pdf?ts=1630060743275&ref_url=https%253A%252F%252Fwww.google.com%252F, 2012. [Online ; accessed March-2021].
- [6] Texas Instrument. Datasheet msp430fr5964. <https://www.ti.com/document-viewer/MSP430FR5969/datasheet/device-overview-slas7047447#SLAS7047447>, 2016. [Online ; accessed April-2021].
- [7] Texas Instrument. User's guide debug probe. <https://www.ti.com/lit/ug/sprui94/sprui94.pdf>, 2017. [Online ; accessed May-2021].
- [8] ST Microelectronics. Datasheet accéléromètre lis3dh. <https://www.mouser.fr/datasheet/2/389/cd00274221-1797088.pdf>, 2016. [Online ; accessed March-2021].
- [9] Molex. Datasheet antenne gps 146235-0001 molex. https://www.mouser.fr/datasheet/2/276/1/1462350001_ANTEENAS-958768.pdf, 2019. [Online ; accessed February-2021].
- [10] Pulse. Datasheet antenne gsm w3070. <https://www.mouser.fr/datasheet/2/336/-552802.pdf>, 2009. [Online ; accessed March-2021].
- [11] Ramzy RAMMOUZ. Optimisation de la gestion d'énergie dans les systèmes embarqués. <http://theses.insa-lyon.fr/publication/2017LYSEI122/these.pdf>, 2017. [Online ; accessed 15-May-2021].

- [12] SIMCOM. Datasheet gps sim28m. <https://www.simcom.com/product/SIM28M.html>, 2014. [Online; accessed May-2021].
- [13] SIMCOM. Datasheet gsm sim800c. <https://www.simcom.com/product/SIM800C.html>, 2017. [Online; accessed June-2021].
- [14] Pierre-Emmanuel Hladik Vincent MAHOUT. Développez en C pour l'embarqué. <https://openclassrooms.com/fr/courses/4117396-developpez-en-c-pour-lembarque/4630296-terminez-sur-des-bonnes-pratiques>, 2021. [Online; accessed March-2021].