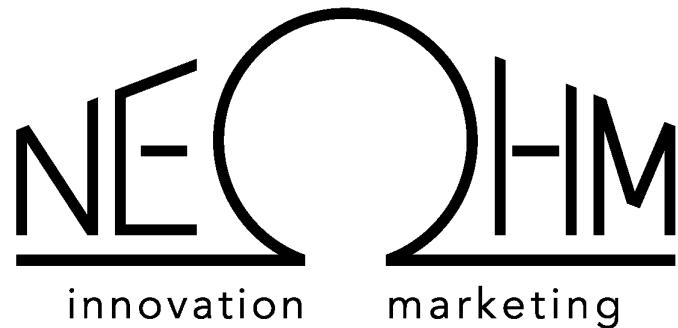




Rapport de stage de césure



Pierre GINTRAND
Promo 2019

Tuteur : David COLLARDEAU

Août 2017 — Décembre 2017

Sommaire

1	Introduction	3
2	Objectifs et contexte	5
3	Démarches et travaux réalisés	7
3.1	Looping / EasyFeed	7
3.2	Site web	10
3.3	Seveur	11
3.4	Worklamp	12
4	Analyse des résultats obtenus	18
5	Apports	19

Remerciements

Je tiens à remercier David COLLARDEAU, Jean WEBER et Guillaume BARADEL pour m'avoir accueilli dans leur service et accompagné avec bienveillance tout au long de mon stage.

Je remercie également tous les collaborateurs qui m'ont entouré : Sy et Green pour leur bonne humeur quotidienne et l'ambiance de travail agréable qu'ils ont instaurée au sein de leur équipe, ainsi que pour leur professionnalisme qu'ils m'ont fait partager.

Chapitre 1

Introduction

Pour mon 1er stage long à réaliser entre ma 2e et 3e année d'école d'ingénieur, mon objectif était de travailler dans un domaine technique à l'étranger. Je tenais particulièrement à occuper un poste dans lequel je pourrai appliquer les connaissances acquises en sein de mes 2 premières années du cursus Supélec. En effet, l'apprentissage en école d'ingénieur est extrêmement théorique et effectuer un tel stage est un complément indispensable à la formation. Effectuer une expérience me permet de sortir de ma zone de confort, de découvrir une nouvelle culture et d'évoluer au sein d'un environnement multiculturel. J'ai répondu à l'offre de stage que Neohm à déposer sur l'Environnement Numérique de Travail de Supélec : un stage de 5 mois et demi à Shenzhen en Chine en tant qu'ingénieur software embarqué. J'avais acquis les prérequis pour postuler grâce à mon TIEP réaliser en classes préparatoires : connaître le langage C et d'avoir une expérience en électronique embarquée. Shenzhen est la référence mondiale dans l'électronique embarquée : la plupart des grandes marques chinoises d'électronique ont leur siège social situé à Shenzhen (Huawei, DJI, Oppo) et quasiment tous les produits électroniques dans le monde sont fabriqués dans la région de Guangdong. Ce stage m'a donc permis d'être plongé dans ce domaine. Neohm est une start-up fondée par 3 français en 2016. Les associées sont Jean WEBER, directeur de l'industrialisation, Guillaume BARADEL, directeur marketing, et David COLLARDEAU, directeur de la recherche et développement. Ce stage m'a permis d'appréhender le milieu professionnel. Il m'a fait découvrir comment se met en place le travail en équipe et l'importance que représente la hiérarchie dans une grande entreprise. Ce rapport constitue un retour d'expérience décrivant le fonctionnement de l'entreprise et les différentes missions que j'ai réalisées au cours de mon stage. J'expliquerai également en quoi il m'a éclairé sur le métier d'ingénieur au sein d'une petite structure et comment il a alimenté mes réflexions autour de mon projet professionnel.



FIGURE 1.1 – Bureau à la frontière Chine - Hong-Kong

Chapitre 2

Objectifs et contexte

Mon stage s'est déroulé dans le pôle recherche et développement de Neohm, sous la direction de David COLLARDEAU. Au début, nous étions 4 à travailler dans ces bureaux : David, Sy et Green et moi. Jean venez dans les bureaux environs une fois toutes les 2 semaines pour faire un point sur l'avancement et préparer au mieux l'industrialisation du produit. Plus tard, 2 nouveaux collaborateurs se sont ajoutés à l'équipe. Le sujet initial du stage dont David m'avait parlé par téléphone en juillet était le projet « Looping », il s'agissait d'un chauffe biberon sur batterie utilisant le phénomène d'induction. Un élément chauffant en métal est placé dans un biberon rempli, le tout est placé sur ou dans le système de chauffe (selon le prototype).

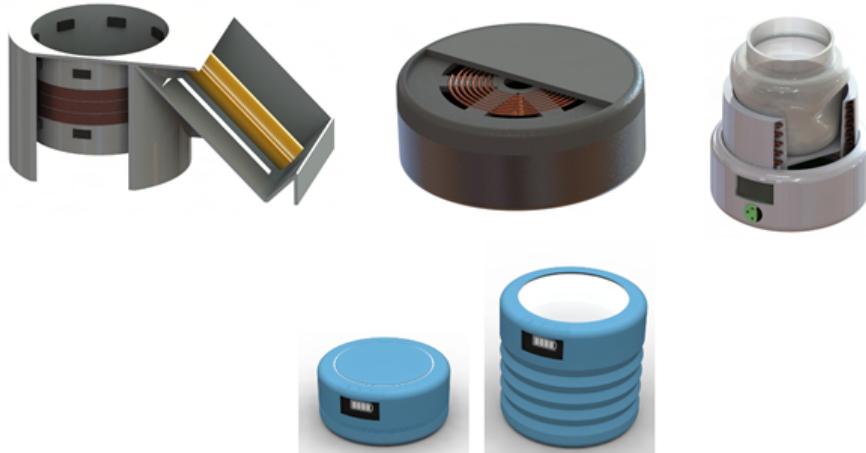


FIGURE 2.1 – Dessins 3D de prototypes de Looping

Guillaume, le directeur marketing basé en France est à l'origine du projet. Il a travaillé plusieurs années dans l'industrie du bébé avant de fonder Neohm, et selon lui il y aurait un fort potentiel pour une tel produit. Cependant, après plusieurs semaines de travail sur ce projet durant l'été 2017, David s'est aperçu que la chauffe à induction comporte de nombreux défis techniques comme la modulation du courant envoyé dans les bobines ou l'isolation du système pour son champ électromagnétique n'interfère pas avec d'autres appareils électroniques. Le rendement de l'induction n'est pas non plus particulièrement bon, l'intérêt de cette technologie est la vitesse de chauffe car seul l'élément chauffant reçoit directement l'énergie. L'équipe a donc décidé de mettre Looping et la chauffe par induction de côté pour se concentrer sur un autre projet : *EasyFeed*. Ce projet est un système portatif qui permet à lui seul de confectionner un

biberon. L'utilisateur y insère le lait en poudre et l'eau et le système remplit un biberon chaud avec les doses adaptées. L'eau sera ici chauffée par une simple résistance. Mon travail sera de programmé le microprocesseur présent dans le produit qui contrôle les différentes fonctionnalité (affichage, contrôle de moteurs, charge des batteries, protections en température du produit...).

Chapitre 3

Démarches et travaux réalisés

3.1 Looping / EasyFeed

Au début du stage, David m'annonce que je commencerai à travailler sur la gestion des batteries dans *EasyFeed*. Dans un premier temps, il me demande de m'intéresser sur l'USB type C. Il veut équiper le produit d'un tel connecteur. Cette nouvelle norme permet d'unifier toutes les connecteurs USB en un seul.

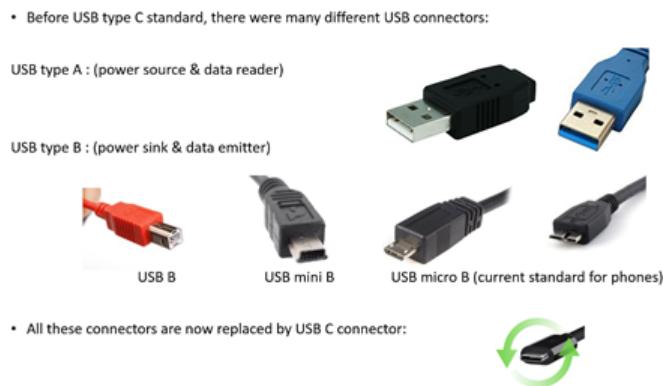


FIGURE 3.1 – Différents types de connecteurs USB

De plus, l'USB C est compatible avec la spécificité Power Delivery qui permet de fournir jusqu'à 100W de puissance électrique. Inclure l'USB C au produit apporterait un confort à l'utilisateur qui pourra par exemple utiliser le chargeur du MacBook d'Apple est un connecteur USB-C et qui fournit 60W. L'USB C permet également un échange de donnée et de d'énergie dans les 2 sens du cable alors qu'auparavant, les données ne pouvait qu'aller du connecteur type B au connecteur type A et l'énergie du connecteur type B au connecteur type B (i.e. il n'était ni possible de chager un pc avec un portable et ni de lire des données d'un pc avec un portable). Je me suis donc renseigné sur la norme USB Power Delivery qui peut en effet fournir jusqu'à 20V / 5A. Cependant, pour fournir une telle puissance, la source doit négocier la tension et le courant avec le débiteur de courant (figure 3.2).

Cette négociation nécessite un contrôleur spécifique. Les recherches que j'ai effectuées m'ont mené à des contrôleurs assez onéreux, environ 5\$/unité. De plus un adaptateur secteur / USB-C supportant 80W coûtent relativement cher car il s'agit d'une technologie nouvelle (notamment utiliser dans les derniers ordinateurs portables et téléphones) Après avoir présenté mes recherches à David, il m'a présenté le PCB qu'il avait conçu pour la gestion de l'alimentation du

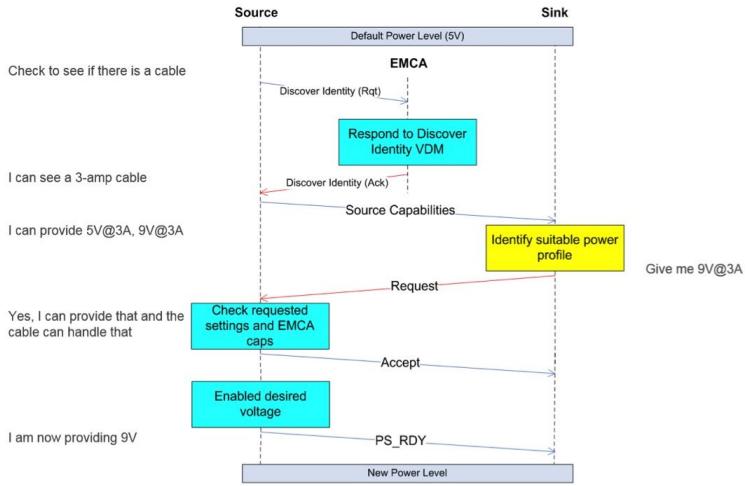


FIGURE 3.2 – Exemple d'une négociation USB PD (simplifiée)

projet Looping car le pack batterie d'*EasyFeed* allait être identique et les besoins d'alimentation semblables.

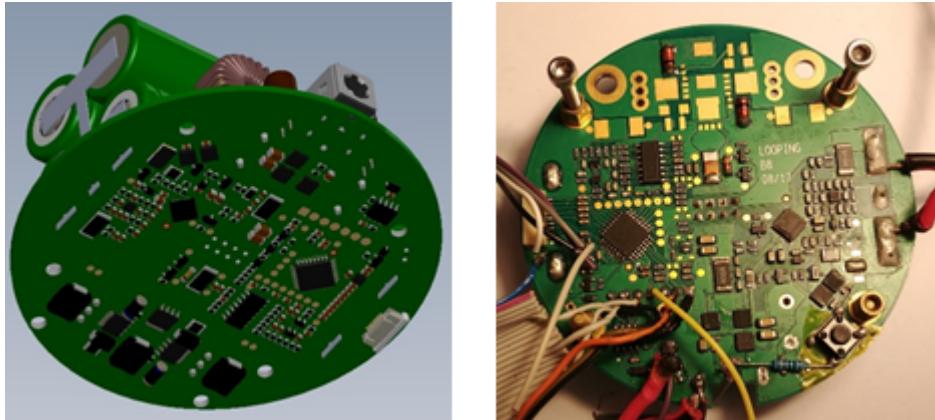


FIGURE 3.3 – PCB du projet Looping

Le circuit était conçu autour d'un contrôleur Texas Instrument BQ40Z60 gérant les phases de charges et de décharges du pack batterie et d'un microprocesseur Silabs 8 bits EFM8LB1 (pas présent sur le schéma) pour communiquer avec le contrôleur et de piloter globalement l'appareil (on/off, affichage du niveau de la batterie, boucle de contrôle de la température, ...).

Mon travail fut donc de programmer le microprocesseur pour communiquer avec le contrôleur de batterie. J'ai dans un premier temps consulté attentivement les documentations et les manuels de ces 2 composants afin de comprendre leur fonctionnement. J'ai dû également m'intéressé au SMBus qui permet aux 2 puces de communiquer via les fils SMBD et SMBC. Le composant BQ40Z60 gère à la fois la charge de la batterie, la décharge, l'équilibre des cellules, les protections en température et en courant, l'estimation du niveau de batterie. Le SMBus permet de configurer la puce et surveiller son état pendant son fonctionnement. SMBus utilise donc 2 lignes pour communiquer :

- SMBD est une ligne de données bidirectionnelle,
- SMBC est une ligne d'horloge de synchronisation bidirectionnelle,

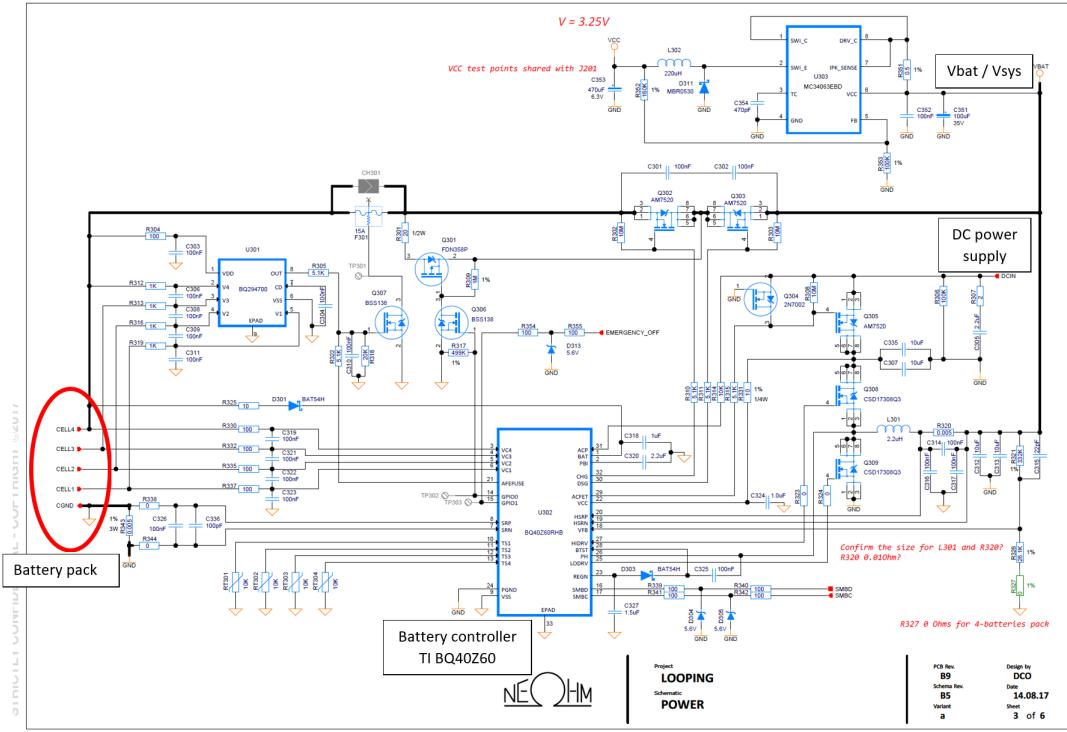


FIGURE 3.4 – Schéma électrique de la gestion de batterie de Looping

Pour visualiser le protocole SMBus, j'avais à disposition un oscilloscope pour observer un représentation des signaux électriques traversant les 2 lignes (figure 3.5) ainsi qu'un analyseur logique (figure 3.6).

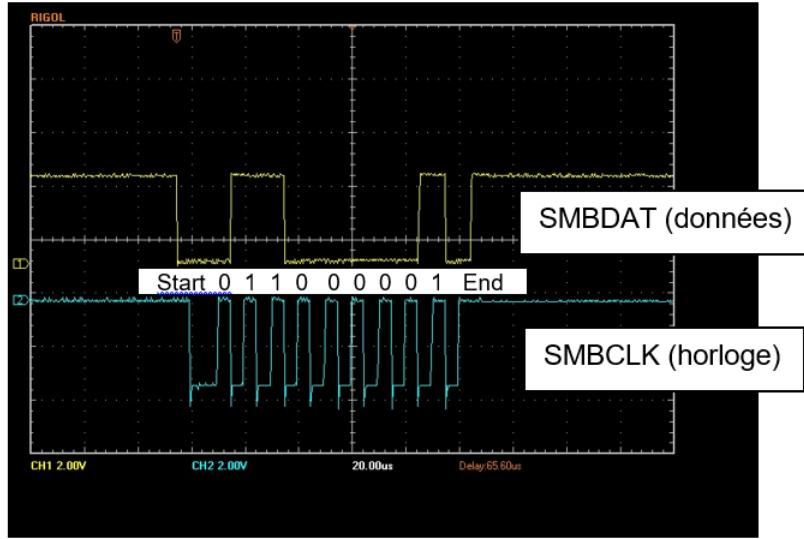


FIGURE 3.5 – Byte SMBus visualisé avec un oscilloscope

L'analyseur logique permet d'être plus rapide car il convertit les données directement en



FIGURE 3.6 – 3 bytes SMBus visualisés avec un analyseur logique

hexadécimal tandis que l'oscilloscope donne un représentation plus réaliste des signaux pour repérer des problèmes analogiques (signaux bruités, capacitifs) par exemple. Le principal problème que j'ai rencontré est que le BQ40Z60 ne semblait pas piloter les 2 transistors FET qui gèrent la charge et la décharge de la batterie (les 2 transistors en série sur le schéma électrique). J'ai donc essayé de contrôler ces transistors manuellement avec des commandes expliquées dans le manuel du composant.

FUNCTION	MANUFACTURER ACCESS COMMAND	SBS COMMAND	ACCESS	FORMAT	DATA READ ON 0x44 OR 0x23	AVAILABLE IN SEALED MODE
ChargeFET	0x001F		W	—	—	—
DischargeFET	0x0020		W	—	—	—
Gauging	0x0021		W	—	—	—
FETControl	0x0022		W	—	—	—

FIGURE 3.7 – Extrait du *BQ40Z60 Technical Reference Manual*

Le contrôleur ne semblait pas apprécier ces commandes car après avoir reçu celles-ci, il ne fonctionnait pas, il renvoyait un NACK au lieu d'un ACK puis plus aucune fonction ne répondait (comme la lecture du numéro de série du composant par exemple). J'ai fait part de mes difficultés à contrôler le composant à David, il a décidé de commander un debugger spécifique pour les contrôleurs batterie de Texas Instruments, le EV2300. Cette appareil propose une interface permettant de visualiser tous les registres du BQ40Z60 et le configurer plus aisément. Ce projet a été mis en pause en attendant de recevoir ce debugger.

3.2 Site web

En parallèle de ce projet, Guillaume, le directeur marketing de la start-up, m'a demandé d'adapter le site internet neohm.net fraîchement créé par David pour qu'il soit correctement visible sur mobile. Le site internet avait été conçu à partir d'un template du framework Bootstrap. Il était donc à 90% adapté pour mobile mais il fallait corriger quelques défauts. J'ai donc modifier dans le feuille de style la taille de certains éléments en utilisant une technologie *responsive* qui s'adapte à la taille de l'écran, modifier la position de conteneurs d'absolue à relatif et ajuster certaines marges qui faisait apparaître une barre de défilement horizontale sous mobile. J'ai également modifier le code JavaScript d'une fonction qu'avait rajouté David, qui permet d'afficher périodiquement des mots sur une image de fond pour que cela fonctionne également sur smartphone. Après ces modifications, le site était parfaitement lisible depuis tout appareil (figure ?? et ...).

FIGURE 3.8 – Registres du BQ40Z60 dans le logiciel BQ studio

3.3 Seveur

David a décidé de mettre en place un serveur au bureau afin de centraliser les projets, les ressources, l'impression. Sy, l'ingénieur chinois, s'est chargé de monter le serveur, d'y installer Linux Ubuntu et de mettre en place le logiciel Samba pour partager des dossiers et l'imprimante en réseau local. Ma mission fût de créer un accès distant au serveur via le protocole FTP. J'ai installé le paquet VsFTPd afin de créer un serveur sécurisé. Le problème est que l'adresse IP de la connexion internet attribuer par notre fournisseur d'accès n'est pas fixe, et demander un IP fixe coûte cher en Chine. Pour parer ce problème, David m'a demandé d'installer le logiciel TeamViewer qui permet de prendre le contrôle d'un PC à distance via une connexion internet. Ainsi, pour se connecter au serveur FTP à distance, il faut dans un premier temps utiliser TeamViewer afin de déterminer l'adresse IP actuel du serveur, puis utiliser un client FTP pour se connecter aux serveurs. David m'a également chargé de mettre en place un gestionnaire centralisé de gestion version. J'ai donc installé Apache Subversion sur le serveur. Travailler avec un tel gestionnaire permet d'être plus efficace en équipe et assure une traçabilité des modifications effectuées sur les projets. A la demande de l'équipe, je me suis occupé de trouver un moyen pour contourner le pare-feu chinois qui nous empêche l'accès à de nombreux sites utiles dans notre travail : Google, Gmail, WhatsApp, YouTube, etc. 2 solutions s'offraient à moi : mettre en place un VPN sur un router wifi ou utiliser le réseau 4G hongkongais que l'on capte dans les locaux (proche de la frontière avec Hong-Kong) pour fournir l'accès internet. Je suis allé à Hong-Kong afin de me renseigner et d'acheter un router wifi compatible. Sur les conseils d'un vendeur, j'ai choisi un router à la fois compatible avec les VPN et avec une clé USB 4G.

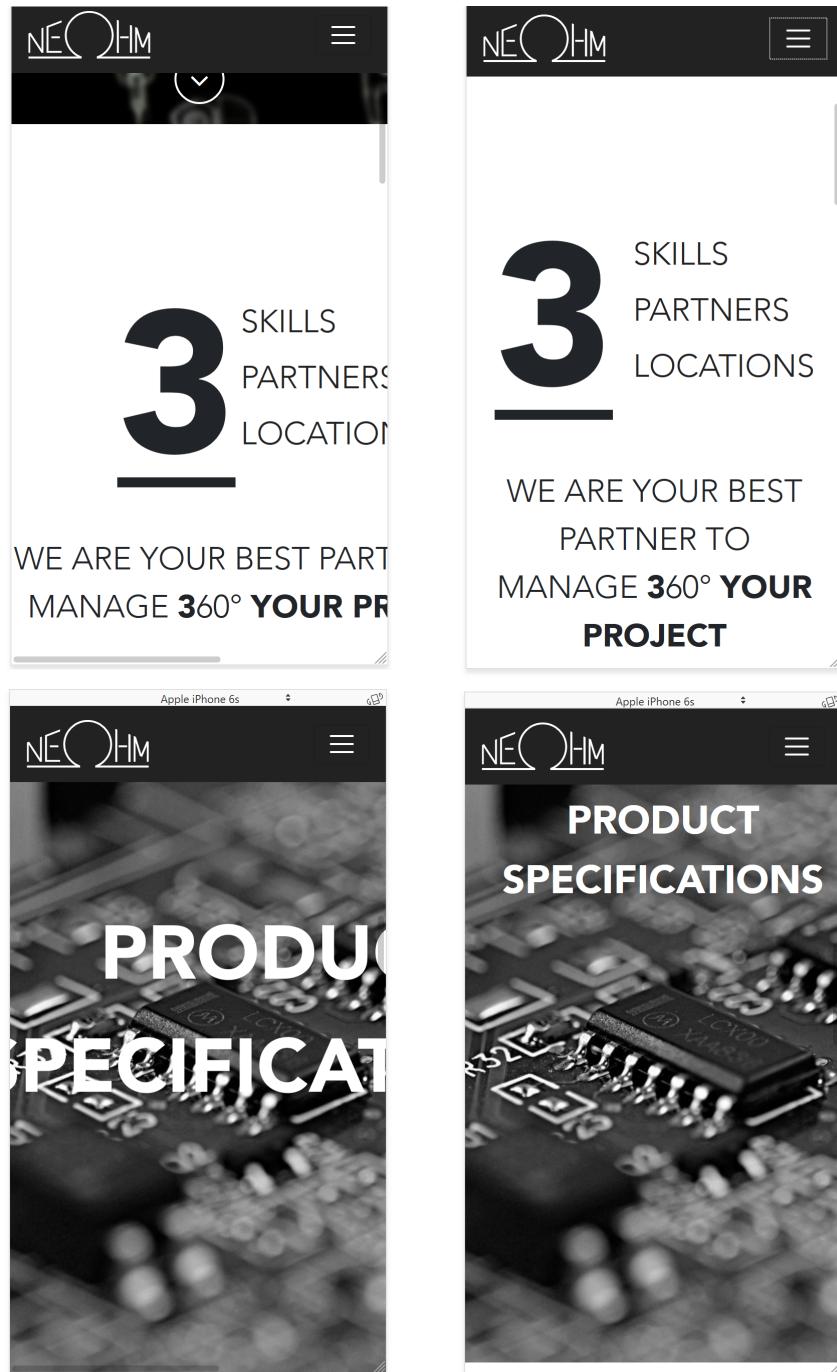


FIGURE 3.9 – Avant/après du site vu d'un smartphone

3.4 Worklamp

David m'a présenté un autre projet en cours sur lequel j'allai travailler : une lampe à LED portable pour peintre. Cette lampe existe sous 2 versions : une version « AC » 80W alimentée par le secteur et une version « DC » 50W alimentée par batterie. Pour simplifier l'industrialisation du produit, le boitier est identique pour les 2 versions, tous les ports ne sont utilisés dans les



FIGURE 3.10 – Router wifi compatible VPN avec une clé USB LTE

2 versions. Le PCB et le panneau sont quand à eux spécifique à chaque version du produit. Il y a plus de LEDs dans la versions 80W et les PCBs n'ont pas le même rôle (redressement et lissage de la tension pour la version AC, gestion de la batterie pour la version DC). Dans les 2 version, un potentiomètre permet de faire varier lumineuse. Les panneaux de LEDs sont composés de 2 types de LEDs (3700K et 4200K) afin de disposer de 3 modes d'éclairage différents. Un bouton permet de changer la température de la couleur émise : 3700K, 5700K et 4200K (les 2 types de LED sont allumés à mi-puissance).

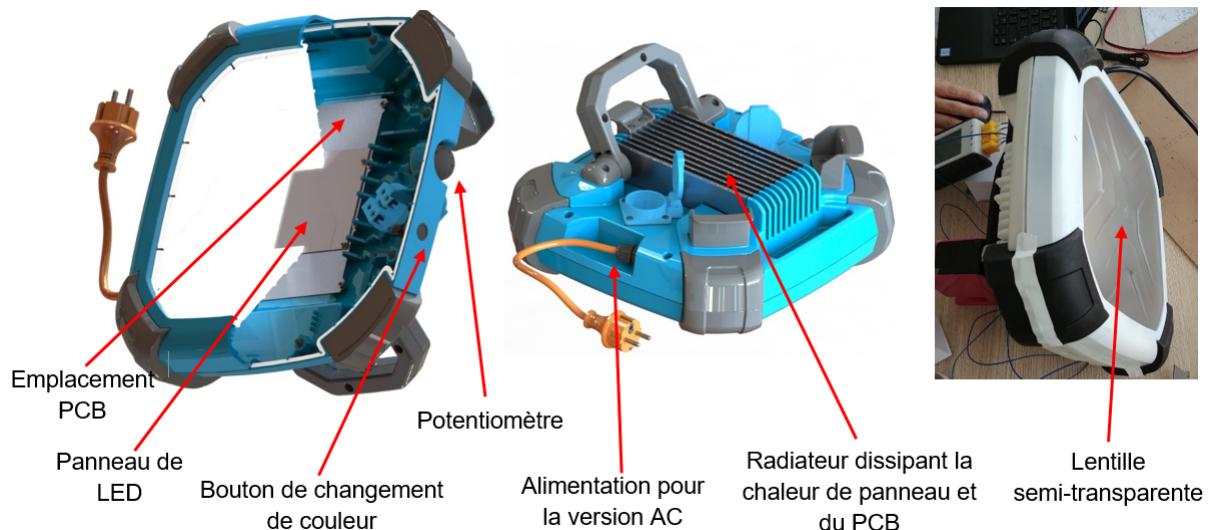


FIGURE 3.11 – Dessin 3D et photo de la worklamp

Ma mission a porté sur la version DC, j'ai été chargé de programmer le microprocesseur présent dans le circuit, de débugger et de tester les prototypes. La version AC est purement hardware, elle est composée de filtres analogiques et de bascules pour gérer le changement de mode de couleur. A contrario, la version DC comporte un processeur pour pouvoir gérer le

Version AC



Version DC



FIGURE 3.12 – PCBs des versions AC et DC de la worklamp

pack batterie (charge, état du niveau de batterie, protection en courant et température) et nécessite donc un développement logiciel. Le processeur est un ... Il se programme en langage C. David m'a donc transmis le schéma électrique (figure 3.13) qu'il avait conçu qui correspond au PCB du modèle DC pour que je me l'approprie pour ensuite implémenter un software dans le microprocesseur.

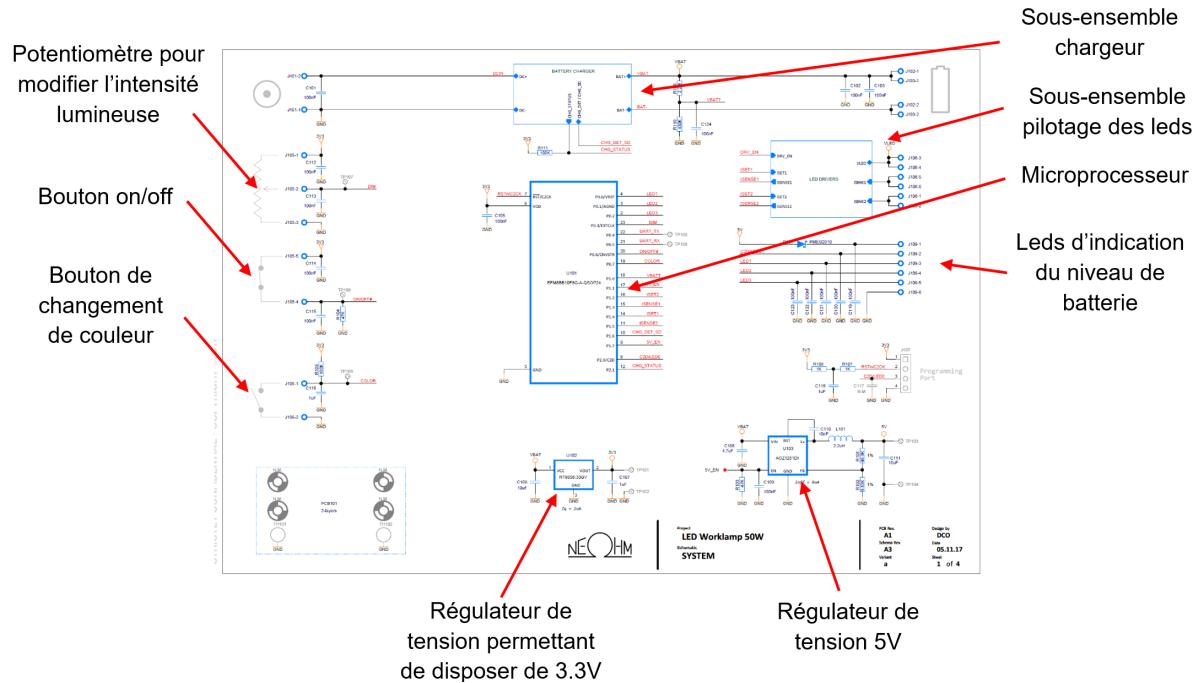


FIGURE 3.13 – Schéma électrique de la worklamp

La première étape fut d'écrire une spécification pour définir les tâches à remplir par le processeur pour ensuite aboutir à un ensemble d'algorithme accomplissant ces missions. Les

tâches que doit effectuer le processeur sont :

- Se mettre en veille (mode basse consommation) sur commande du bouton ON/OFF,
- Moduler l'intensité lumineuse en fonction de l'état du potentiomètre,
- Changer la couleur sur un appui sur le bouton COLOR,
- Indiquer l'état de charge des batteries quand la lampe est en fonctionnement ou en charge à l'aide de 4 leds,
- Fournir un port USB de charge 5V lorsque la lampe n'est pas en veille,
- Gérer la charge du pack batterie,
- Protéger le produit en cas de surchauffe.

Il a ensuite fallu de définir la configuration des pins du microprocesseur en accord avec le schéma grâce la datasheet du EFM8BB1 (figure 3.14). Il s'agit de définir des fonctions et noms variables aux différents pins (input, output, PWM, ADC, UART, SMB, ...).

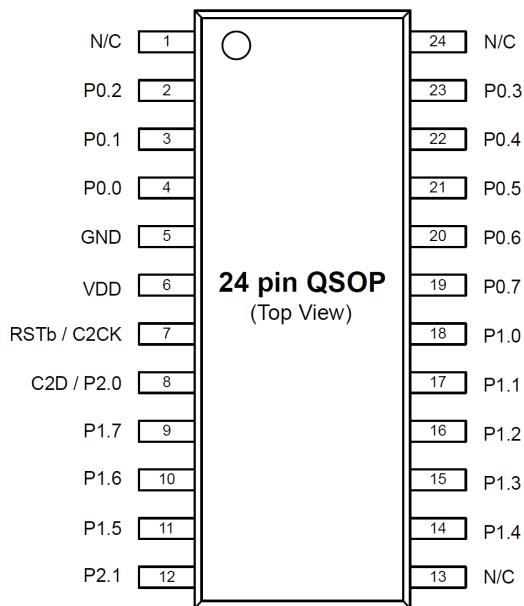


FIGURE 3.14 – Définition de pins sur le processeur EFM8BB1x-QSOP24

Cette étape est cruciale pour la suite du développement et demande quelques notions d'électronique. Par exemple il faut définir si le pin est défini en « open drain », c'est-à-dire que l'état haut correspond à une forte impédance et l'état bas à une faible impédance, ou en « push-pull », si l'on veut que l'état haut soit le 3,3 V du CPU, et état bas du CPU (figure 3.15).

Le pack batterie est composé de 2 batteries lithium-ion 4S (tension nominal $4 \times 3,7 = 14,8$ V) en parallèle pour une capacité totale de 10,4 A.h (153,92 W.h). En fonction de l'avancement du projet, on a changé de contrôleur gérant la charge de la batterie, on est passé du BQ2000 au BQ25703A qui est plus performant et qui nous permet de directement lire la tension de la batterie pour estimer son état de charge et de surveiller le courant de charge/décharge via la protocole I2C. Un contrôleur pour charger la batterie lithium-ion permet de gérer les 3 phases de la charge. La dernière phase est la *trickle charge*, elle est optionnelle et permet d'optimiser la durée de vie de la batterie si celle-ci a un très bas niveau, tous les contrôles n'intègre pas cette phase. Durant la *trickle charge*, un faible courant est transmis à la batterie afin d'atteindre une certaine tension. La 2nd phase est le « constant current » : un courant élevé est envoyé dans le pack batterie jusqu'à atteindre la tension de charge. Dans mon cas, le courant est de $2 \times 2,6 = 5,2$ A (2 packs en parallèle) et la tension de charge de $4 \times 4,2 = 16,8$ V (chaque pack à 4 cellules

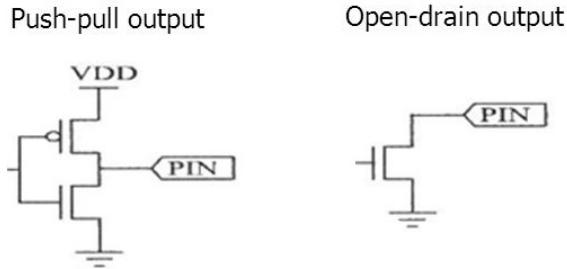


FIGURE 3.15 – Illustration sortie « Open drain » et « Push-pull »

en série). La dernière phase est le « constant voltage » : on fixe une certaine tension (16,8V dans mon cas) aux bornes du pack batterie jusqu'à atteindre un certain courant, puis on garde cette tension pendant 30-60min pour finaliser la charge.

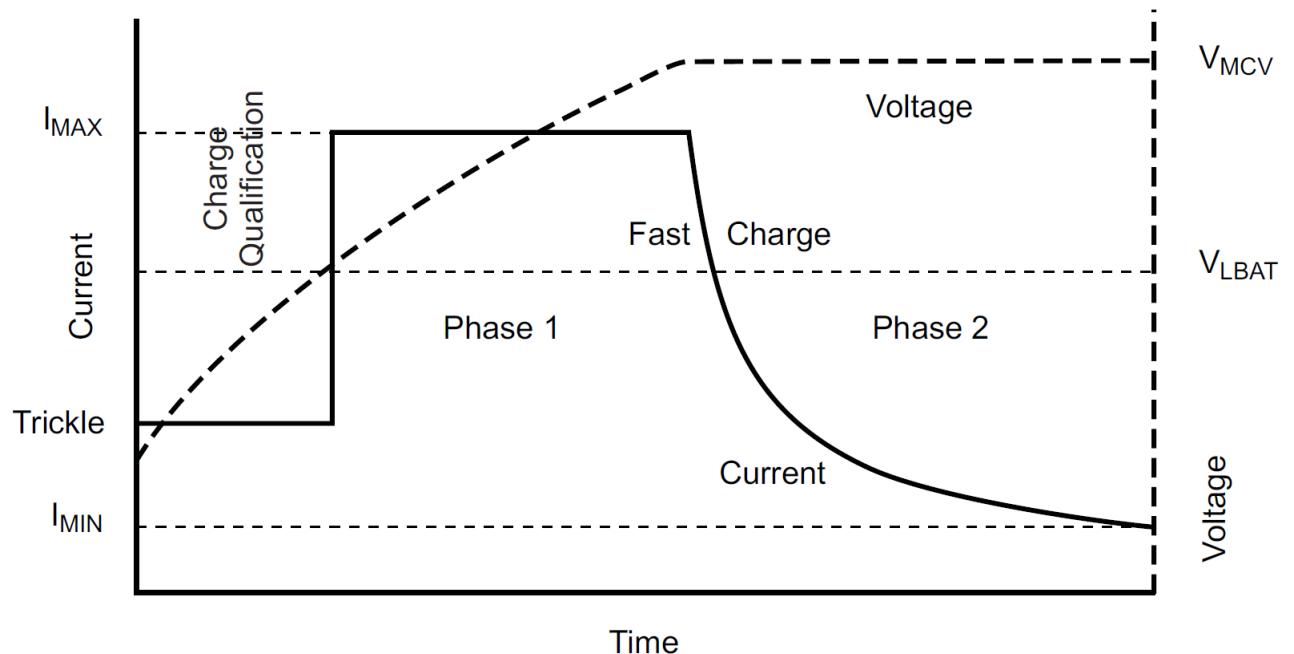


FIGURE 3.16 – Algorithme de charge d'une batterie lithium-ion

J'ai conçu des algoritgrammes pour chaque fonctionnalité que le Software doit gérer (figure 3.17). Cette étape m'a permis d'avancer dans le développement logiciel en gardant un certain recul avec les consignes auxquelles je m'étais fixer.

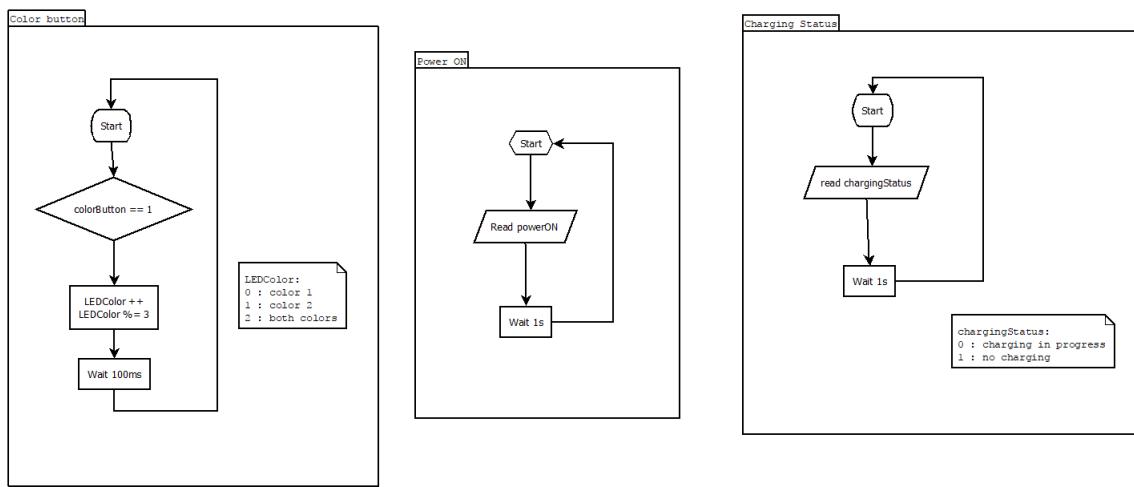


FIGURE 3.17 – Quelques algorigrammes du software de la worklamp

Chapitre 4

Analyse des résultats obtenus

Chapitre 5

Apports

Table des figures

1.1	Bureau à la frontière Chine - Hong-Kong	4
2.1	Dessins 3D de prototypes de Looping	5
3.1	Différents types de connecteurs USB	7
3.2	Exemple d'une négociation USB PD (simplifiée)	8
3.3	PCB du projet Looping	8
3.4	Schéma électrique de la gestion de batterie de Looping	9
3.5	Byte SMBus visualisé avec un oscilloscope	9
3.6	3 bytes SMBus visualisés avec un analyseur logique	10
3.7	Extrait du <i>BQ40Z60 Technical Reference Manual</i>	10
3.8	Registres du BQ40Z60 dans le logiciel BQ studio	11
3.9	Avant/après du site vu d'un smartphone	12
3.10	Router wifi compatible VPN avec une clé USB LTE	13
3.11	Dessin 3D et photo de la worklamp	13
3.12	PCBs des versions AC et DC de la worklamp	14
3.13	Schéma électrique de la worklamp	14
3.14	Définition de pins sur le processeur EFM8BB1x-QSOP24	15
3.15	Illustration sortie « Open drain » et « Push-pull »	16
3.16	Algorithme de charge d'une batterie lithium-ion	16
3.17	Quelques algoritgrammes du software de la worklamp	17