Bordeaux INP
ENSEIRB
MATMECA

Systèmes embarqués
Projet de fin d'études
Rapport

Le développement d'une sonde intelligente

Élève:	*Tuteur pédagogique:*	*Maître de stage:*
BELOUARGA Mohamed	Valéry LEBRET	Jérémy LAUVIGE

06 Août 2021

Table des matières

Table des figures

Liste des tableaux

1 Remerciements

En premier lieu, je tiens à remercier toute l'équipe pédagogique de l'ENSEIRB-MATMECA de Bordeaux pour le savoir-faire, la méthodologie et l'esprit d'ingénieur qu'ils nous ont transmis durant mes trois années de formation dans le domaine de l'électronique.Il m'est agréable de saisir cette occasion pour remercier M. Patrice KADIONIK, le responsable de l'option Systèmes Embarqués, pour son suivi et son effort pour nous former dans le domaine des systèmes embarqués.

Je suis aussi profondément reconnaissant envers mon maître de stage, M. Jérémy LAUVIGE, qui m'a accordé beaucoup de son temps pour m'expliquer les différents aspects techniques de mon stage de fin d'études. Son expertise, ses remarques constructives et ses connaissances dans le domaine de l'embarqué m'ont aidé à progresser énormément.

Je tiens aussi à remercier M. Emilien BLATEAU, Manager logiciel embarqué, qui m'a ouvert les portes de l'entreprise, et pour son aide à bien intégrer l'entreprise et faire connaissance aux différents personnels de l'entreprise.

Ensuite, j'adresse mes remerciements à toute l'équipe de l'embarqué de Sauermann Group pour leur accueil et leur aide, durant toute la période de stage, plus particulièrement :

M. Mathieu THIRIAU, le responsable de produit, pour ses conseils et ses clarifications à réaliser des tâches techniques.

M. Pierre-Luc GUEGAN, ingénieur du logiciel embarqué, pour son soutien et ses conseils que ce soient techniques ou logiciels.

M. Antoine CALLEC, ingénieur du logiciel embarqué, pour l'aide qui m'a accordé à finir des tâches et pour l'explication des différents aspects techniques.

M. Emmanuel TRAMESON, ingénieur du logiciel embarqué, pour ses remarques et ses explications.

Mlle. Hind LOUMARI, stagiaire et ma collègue pour avoir partagé cette expérience professionnelle.

Enfin, je remercie mon tuteur pédagogique, Valéry LEBRET, qui était le premier à nous présenter le fonctionnement des microcontrôleurs au sein de l'école. Je remercie également toutes les personnes qui m'ont aidé pendant la période de mon stage pour arriver à ce stade.

2 Résumé

Dans le cadre de mon stage de troisième année à l'ENSEIRB-MATMECA, j'ai effectué un stage de fin d'études de six mois à Sauermann Group, afin de mettre en pratique mes compétences acquises durant mes études à l'ENSEIRB-MATMECA. Cette expérience professionnelle reste cohérente avec ma formation puisqu'elle s'inscrit dans le domaine des systèmes embarqués, et précisément, construire un système embarqué sur un microcontrôleur.

Ce stage a comme but la conception et la réalisation d'un firmware d'une sonde intelligente, cette sonde ferait partie des produits de Sauermann Group, qui est un groupe spécialisé dans la conception et la production des instruments de mesure de la qualité d'air et les pompes de relevage de condensats.

Abstract

As part of my third year at my actual engineering school ENSEIRB-MATMECA, I did a six-month internship period at Sauermann Group, in order to use in practice my skills acquired during my studies at ENSEIRB-MATMECA. This professional experience is consistent with my studies since it remains in the embedded systems field, and specifically, building an embedded program for a microcontroller.

The goal of this internship's project is to design and produce a firmware for an intelligent probe that will be a commercialized product by Sauermann Group, which is a group specializing in the design and manufacturing of air quality measuring instruments and condensate removal pumps.

3 Introduction

Pour avoir des environnements industriels sécurisés, ou pour interdire l'air de sortir de la chambre d'un patient atteint de la covid-19, il est important d'avoir des instruments de mesure précis et très développés. Sauermann assure des instruments de mesure de la qualité d'air, ses instruments, qui mesurent la pression, la température, le pourcentage de CO2 ..., permettent à leurs clients d'assurer leurs environnements du travail, ou l'efficacité de leurs chaudières et être plus compétitifs dans certains domaines.

C'est dans ce cadre parvient mon stage, le développement d'une sonde intelligente, cette sonde va permettre aux clients de faire des mesures très précises, d'utiliser une sonde sur différents sites, cette sonde peut être configurée via le transmetteur, l'appareil qui affiche les mesures et les stocke au cloud. L'avantage de cette sonde, c'est qu'elle va être autonome dans la prise et l'ajustage des mesures. L'ajustage est un processus industriel qui assure la fiabilité de la mesure. Quand le client va vouloir ajuster l'environnement des mesures, il sera amené à envoyer seulement la sonde au laboratoire pour la faire ajuster, contrairement au modèle actuel, qui consiste à envoyer la sonde et le transmetteur au laboratoire pour les ajuster. Cela va apporter au client des économies, et va accélérer le processus d'ajustage.

4 Présentation de l'entreprise

4.1 Sauermann Group

Le groupe Sauermann est un groupe d'origine française, devenu un groupe international, implanté sur quatre continents (Europe, Asie, Amérique du nord et l'Australie). C'est un groupe spécialisé dans la fabrication des instruments de relevage de condensats et des instruments de mesure de la qualité d'air. Le groupe Sauermann a connu plusieurs étapes d'évolution pour améliorer sa présence sur la scène internationale, et pour maintenir une forte proximité avec ses clients partout sur la planète.

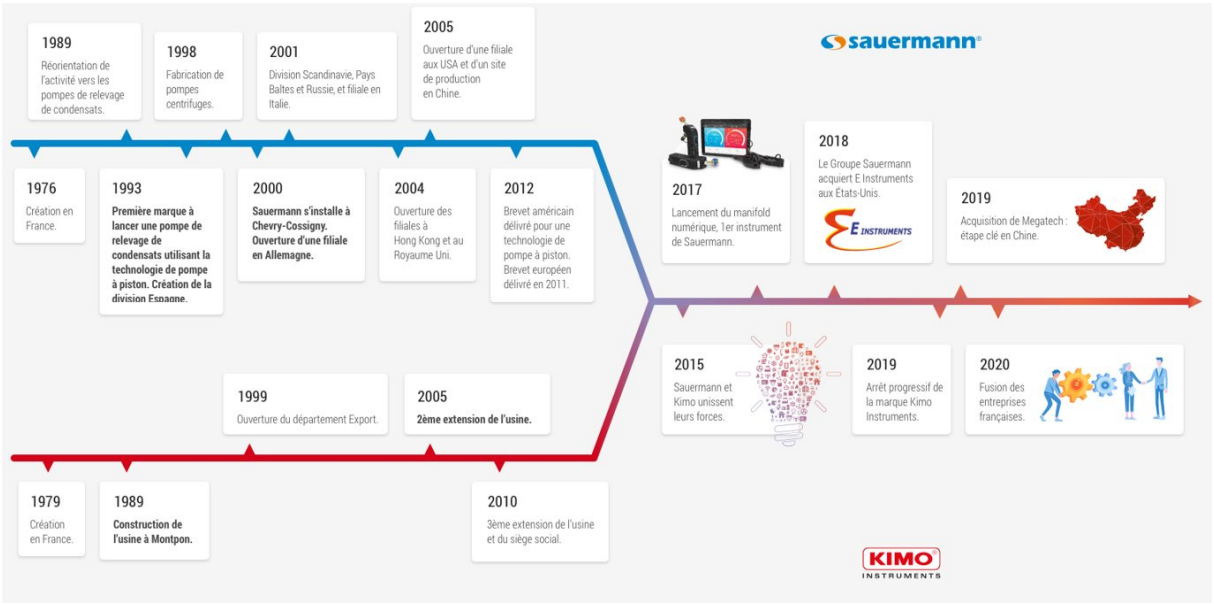

FIGURE 1 – L'évolution du groupe Sauermann

Le groupe Sauermann a trois buts principaux : faire partie des 3 principaux fabricants mondiaux HVACR (Heating, Ventilation, Air-Conditioning and Refrigeration), s'imposer dans le domaine de la qualité de l'air et du pharmaceutique, et repousser les limites de l'innovation dans le design ergonomique, le numérique et les objets connectés. Pour cela, le groupe Sauermann a décidé de ne garder qu'une seule marque, ainsi faire disparaître la marque Kimo Instruments, peu connue sur la scène internationale, d'autre part, le groupe investit énormément pour la conception de nouveaux appareils de mesure et de relevage de condensats et dans ses laboratoires très avancés pour améliorer les mesures prises par les appareils.

4.1.1 Sauermann

Sauermann est une entreprise française, et un fabricant de pompes de relevage de condensats, les appareils de climatisation se débarrassent de ce liquide polluant, les condensats, et les pompes le drainent afin d'augmenter leur durée de vie. Sauermann est concentrée sur le marché international et très avancée numériquement, ce qui lui a permis d'avoir une clientèle internationale.

4.1.2 Kimo Instruments

Fondée en 1979 en Dordogne, par deux cofondateurs M. Bernard MOULINET et M. Calogero TERMINI, l'entreprise Kimo Instruments était spécialisée dans la fabrication des instruments de mesure avec services après vente. Kimo Instruments était concentrée sur le marché français, ce qui lui a permis d'être le leader en France dans ce domaine, d'autre part, au niveau mondial,

Kimo Instruments n'est pas connue, ce qui a poussé sa croissance à stagner à partir de 2011, Figure 2.

Aujourd'hui, Kimo Instruments est devenue Sauermann France, ainsi la disparition de la marque Kimo Instruments.

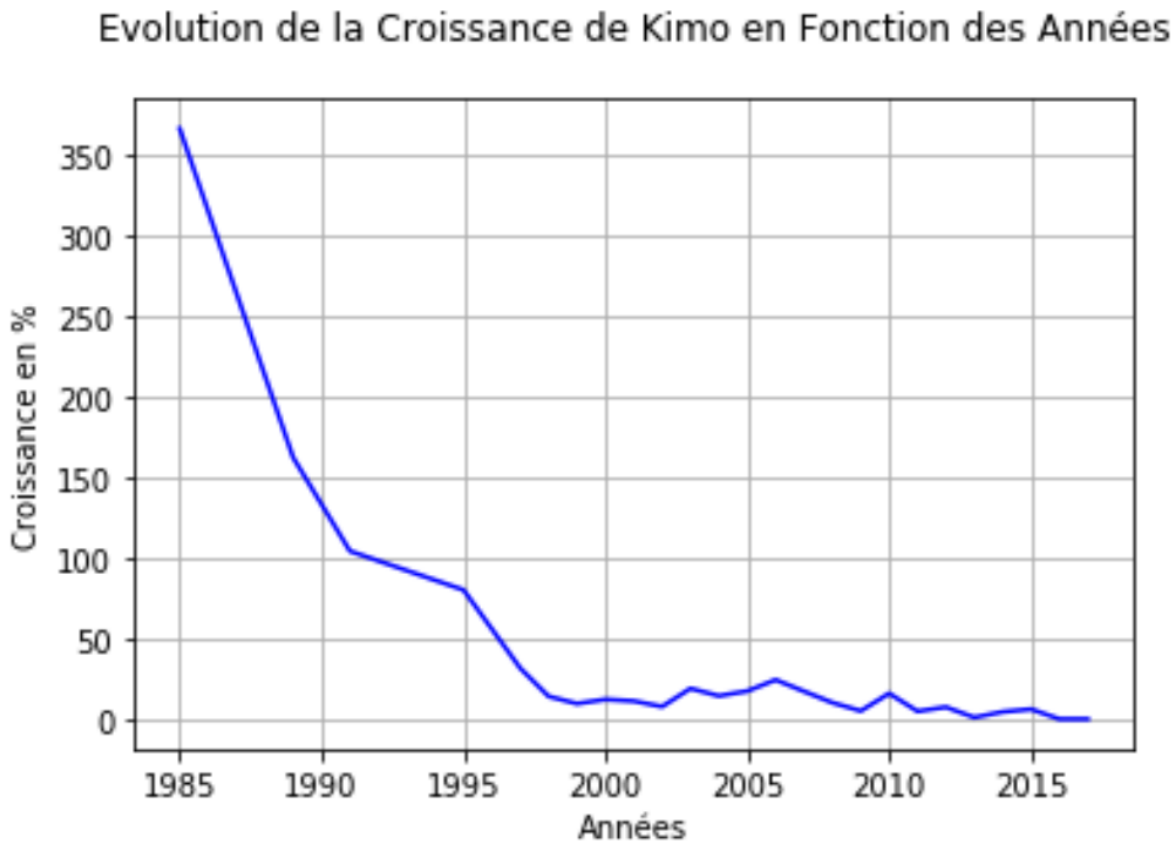

FIGURE 2 – L'évolution de la croissance de Kimo Instruments

4.1.3 E instruments

E instruments est un distributeur américain des pompes de relevage de condensats et les instruments de mesure de la qualité d'air. Toujours dans la recherche d'améliorer sa présence mondiale, le groupe Sauermann a acquis E instruments en 2018 afin de pouvoir conquérir des parts dans le marché américain.

4.1.4 Megatech

Pour renforcer la présence du groupe dans le marché chinois, et asiatique en général, Sauermann a acquis, en 2019, le distributeur Megatech, qui était le distributeur exclusif des instruments de Kimo Instruments en Chine. Cette acquisition a permis au groupe de conquérir des nouveaux marchés asiatiques qui n'étaient pas exploités par l'entreprise.

4.2 Implantation

Sauermann dispose de plusieurs sites, mais mon stage s'est déroulé aux bureaux de coworking de Wellio à la cité numérique, cependant, j'avais la chance de visiter le site de production basé

à Montpon-Ménestérol, et qui dispose des laboratoires pour l'ajustage des appareils :

FIGURE 3 – Le site de production de Montpon-Ménestérol

4.3 Sauermann en chiffres

Le groupe Sauermann est implanté sur quatre continents, et il dispose de 12 sites, dont quatre unités de production. Le département de recherche et développement se compose de 20 ingénieurs et de 10 techniciens. Le nombre de salariés de Sauermann dépasse aujourd'hui 400, de plus de 25 nationalités, et le chiffre d'affaire est en constante évolution.

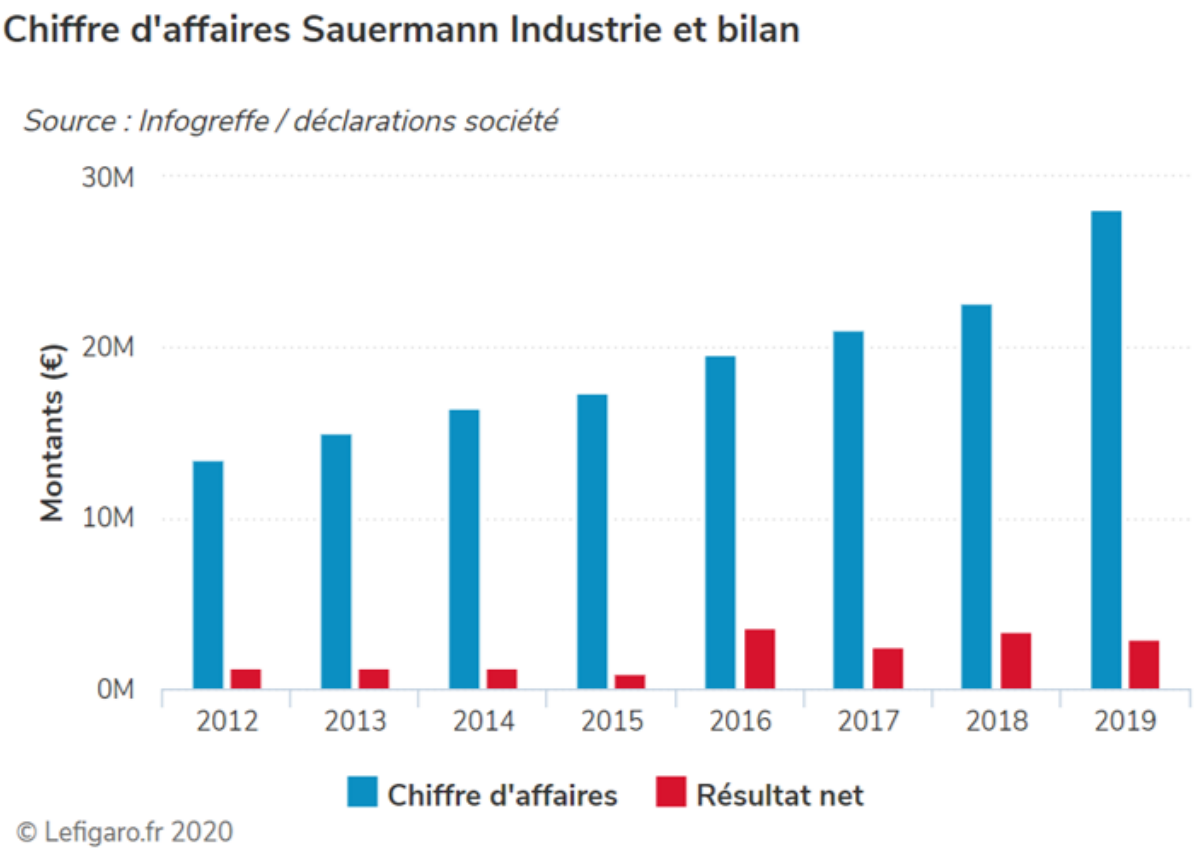

FIGURE 4 – L'évolution du chiffre d'affaire de Sauermann

4.4 Le département R&D

Le département "recherche et développement" dirigé par M.Vincent AYMA, est le département qui conçoit et développe les nouveaux produits qui seront commercialisés par l'entreprise.

Ce département contient principalement 6 équipes, chacune a son domaine de spécialisation. Ces équipes sont coordonnées entre elles à travers le chef du projet.

L'équipe du logiciel embarqué

L'équipe de l'embarqué est une équipe composée par un manager M.Emilien BLATEAU, le lead-tech M.Jérémy LAUVIGE qui est mon tuteur de stage, et trois autres ingénieurs. Son travail est de concevoir et développer les firmwares des produits de l'entreprise. L'autre stagiaire et moi de l'ENSEIRB-MATMECA, Hind LOUMARI étions intégrés à cette équipe, et nous avions des réunions quotidiennes matinales pour discuter de l'avancement du projet, de sprint ...

L'équipe électronique

C'est l'équipe qui conçoit et développe les cartes électroniques de futurs produits, elle choisit également les composants et le microcontrôleur qui seront utilisés. L'équipe de l'embarqué est en contact permanent avec cette équipe pour toute demande particulière, comme l'ajout d'une mémoire externe, ou d'un ADC ...

L'équipe test et validation

L'équipe test et validation est l'équipe qui teste toutes les releases du software qu'elles soient applicatives ou embarquées. Elle teste aussi toutes les nouvelles fonctionnalités, et détecte les bogues de software ou le dysfonctionnement d'une fonctionnalité.

L'équipe application

C'est l'équipe qui développe des applications Android, iPhone ou bureau pour les produits de l'entreprise, ces applications permettent de stocker la data des utilisateurs, les visualiser ou les analyser. Elle gère aussi les serveurs de l'entreprise, et veille sur la sécurité des applications de toute tentative de vol de la data.

L'équipe mécanique

L'équipe mécanique est l'équipe qui conçoit les supports physiques et les boîtiers des appareils.

L'équipe métrologique

L'équipe qui calibre les appareils de mesure, et mènent des recherches pour mesurer le plus précisément possible. Les laboratoires de Sauermann, où l'ajustage se déroule, sont accrédités ISO/CEI 17025 qui est une norme d'exigences générales concernant la compétence des laboratoires d'étalonnages et d'essais.

Le P.O

M.Mathieu THIRIAU est le Product Owner de l'entreprise, il veille sur le développement d'un produit, et définit les priorités des nouvelles fonctionnalités. Il est le responsable de la définition des tâches et de la conception des produits.

4.5 Equipe Marketing

L'équipe marketing est une équipe internationale qui définit les caractéristiques des produits qui vont réaliser un succès. Le résultat de son travail est un document qui s'appelle MRD (Market Requirement Document). Ce document contient les normes que les produits doivent respecter, et leurs caractéristiques. Pour mon stage, le MRD contenait les caractéristiques de toute une gamme de produits, cela veut dire les transmetteurs, les sondes intelligentes et les sondes normales. Nous avons commencé par extraire les informations liées aux sondes intelligentes, du MRD, pour rédiger le cahier des charges.

L'équipe Marketing analyse aussi le produit après sa conception pour faire des retours ou pour demander des changements sur le produit.

5 Le sujet de stage

5.1 La nouvelle gamme de sondes : les sondes intelligentes

La nouvelle gamme des sondes sont des sondes qui peuvent fonctionner sur plusieurs types de transmetteurs et qui peuvent communiquer avec d'autres appareils pour leur transmettre des informations liées aux mesures qu'elles peuvent effectuer.

Les sondes sont des éléments qui viennent se brancher sur le transmetteur, et donc permettent au transmetteur d'afficher les mesures prises et les envoyer au cloud pour le traitement. Les transmetteurs sont des appareils qui pouvant afficher les mesures, communiquer avec les sondes, et lancer des mises à jour du firmware des sondes par exemple.

FIGURE 5 – Deux appareils de type transmetteur de Kimo Instruments

5.2 Les besoins fonctionnels

Les principales caractéristiques de la sonde intelligente sont :

Interchangeabilité

La sonde intelligente doit être interchangeable, ce qui signifie qu'elle doit fonctionner sur plusieurs types de transmetteurs.

Communication

Au début de la connexion, la sonde doit communiquer au transmetteur des informations qui la définissent :

- Le modèle de la sonde que le transmetteur va afficher.
- Le numéro de série
- La version du firmware
- Les informations de ajustage de la sonde
- La dernière date d'ajustage
- La durée d'utilisation

— Les types de mesures
— Les statuts des capteurs
— ....

Ajustage

L'ajustage est un processus industriel permettant la fiabilité de la mesure en compensant les défauts de chaque capteur. La sonde intelligente doit ajuster ses mesures, contrairement aux sondes précédentes qui ne le faisaient pas, et dont le transmetteur faisait à la place. Contrairement aux sondes normales qui devaient être envoyées avec leurs transmetteurs, les sondes intelligentes sont envoyées seules aux laboratoires de Sauermann pour l'ajustage.

Management des erreurs

La sonde doit reporter les erreurs détectées au transmetteur, par exemple : Le capteur ne fonctionne plus.

5.3 Arbre décisionnel

Le fruit de ce travail d'analyse est la conception de l'arbre décisionnel de la sonde intelligente. cet arbre est illustré dans la figure suivante :

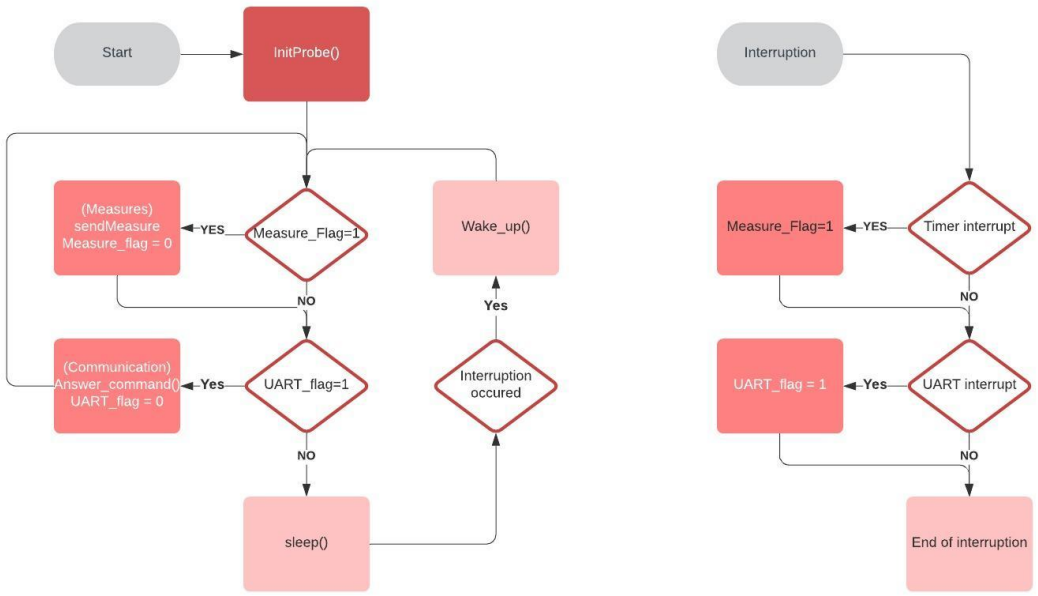

FIGURE 6 – L'arbre décisionnel des sondes intelligentes

Cet arbre décisionnel décrit le fonctionnement du firmware, la sonde après son démarrage initialise les périphériques du microcontrôleur, l'ADC ... Si le transmetteur fait une commande, la sonde doit l'exécuter. et après chaque seconde la sonde soit prendre une mesure et l'envoyer au transmetteur.

Afin de ne pas rester bloquer dans une tâche pendant l'interruption, nous utilisons des flags pour indiquer si une interruption est arrivée. Les interruption servent principalement à réveiller le microcontrôleur. L'interruption serait servie en mode de fonctionnement normal et pas en mode interruption, cela permettra au microcontroleur le fait de ne pas rater une interruption.

6 Le hardware

6.1 La gamme STM8

STM8 est une famille de microcontrôleurs fabriqués par la société ST Microelectronics. Ce sont des microcontrôleurs low-cost. La série choisie pour les sondes intelligentes est STM8L qui est une famille low-power, ce qui signifie qu'elle est très basse en consommation énergétique.

L'équipe électronique a décidé de changer la référence du microcontrôleur et a opté, après, pour un STM32 à la place. Compte tenu de la méthode du développement de l'entreprise, la conception du firmware n'a pas été trop impactée.

6.2 Périphériques

Les microcontrôleurs STM8L contiennent aussi d'autres composants à côté du processeur, de la mémoire flash et la RAM. Ces composants s'appellent des périphériques, comme les 5 timers qui permettent de gérer le temps, les périphériques de communication (I2C, USART(3 unités) et SPI(2 unités)), les deux horloges internes, la mémoire EEPROM et 54 GPIO.

7 L'environnement du développement

7.1 IAR

IAR Embedded Workbench IDE est l'IDE qui sera utilisé pour le développement du projet, il contient toute la toolchain, c'est à dire un compilateur, un débogueur et un éditeur de liens entre les différents fichiers. C'est un logiciel propriétaire, mais il peut être utilisé sans licence pour des firmwares qui ne dépassent pas 32Kb. La toolchain établie par l'IAR prend en charge l'architecture STM8, et elle est très performante au sens où elle prend en charge plusieurs architectures. Sa compilation est rapide et son débogueur soutient pleinement le STM8.

Figure 7 – L'environnement IAR Embedded Workbench

7.2 Le programmeur ST-LINK

ST-Link est à la fois un programmeur et un débogueur des microcontrôleurs STM8 et STM32. Il a deux interfaces : une de type SWIM et l'autre de type SWD. Pour le projet des sondes intelligentes, c'est l'interface SWIM qui est utilisée. L'IAR Embedded Workbench IDE a besoin du programmeur ST-LINK pour qu'il fonctionne. Le ST-LINK représente l'intermédiaire entre un ordinateur et les microcontrôleurs de type STM, il soutient les trois OS les plus utilisés Windows(7,8 et 10), Linux(64-bits) et macOS.

FIGURE 8 – Le programmeur ST-LINK

Interface SWIM

Pour programmer le microcontrôleur, il fallait se servir de l'interface SWIM, qui utilise les quatre pins GND, VDD, RESET et la pin SWIM. Le microcontrôleur suit un protocole de communication appelé SWIM, qui est spécifié par le «User Manuel» UM0470.

8 Les méthodes de développement

Pour le développement du projet, plusieurs outils étaient utilisés pour faciliter notre travail.

8.1 Azure

Azure est une plateforme d'hébergement et de services numériques proposée par la société américaine Microsoft. Cette plateforme sera utilisée pour :

- Héberger le code de notre projet dans le cloud et récupérer les nouvelles fonctionnalités développées par les autres membres de l'équipe.
- Planifier et organiser entre les membres de l'équipe, les tâches de développement et de correction des bogues.
- Faire des demandes d'intégration du code au projet.
- Faire des révisions et des tests pour vérifier si les demandes d'intégration du code respectent les règles de développement de Sauermann.

8.2 Git

Git est un logiciel de gestion de versions du code, qui est déployé pour partager le code entre les membres de l'équipe, et la plate-forme Azure propose une solution d'hébergement Git. L'équipe logiciel embarqué a fixé une règle afin d'utiliser cette solution correctement.

Cette règle est d'utiliser un arbre Git avec une seule branche officielle, et tout le développement doit être fusionné à cette branche. Elle permet d'avoir une seule version du code commune entre toute l'équipe et de ne pas avoir des conflits de versions très délicats à résoudre.

Gitk

Gitk est une interface graphique de Git permettant, entre autres, la visualisation de l'historique des dépôts Git. Au lieu d'utiliser les commandes "git log" et "git diff", Gitk, qui est un logiciel open-source, permet de visualiser les commits et les fusions précédents, les changements effectués. Il permet aussi de voir les commentaires, les dates de fusion et autres informations importantes.

L'interface graphique de gitk est représentée dans la figure suivante :

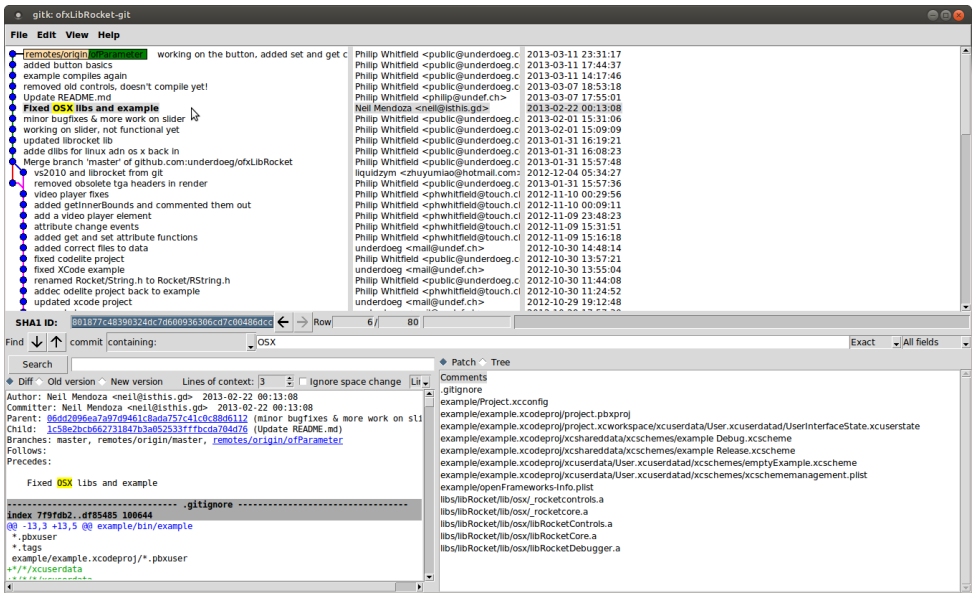

FIGURE 9 – L'interface graphique Gitk (source : internet)

8.3 Intégration continue

Pour avoir un code compréhensible facilement par tous les membres de l'équipe du logiciel embarqué, certaines règles ont été fixées :

— Un modèle des fichiers .h et .c est fixé. Ce modèle précise l'emplacement où l'ajout des autres librairies prennent place, l'emplacement des variables privées et publiques et l'emplacement des fonctions privées et publiques.
— Des règles pour nommer les variables privées et publiques, les fonctions privées et publiques et aussi les noms des nouveaux types déclarés.
— Des règles pour utiliser un seul style lors du développement, comme le nombre de caractères maximum dans une seule ligne, le nombre maximum d'arguments d'une fonction, le

nombre d'espaces blancs pour une tabulation...
— Des règles pour optimiser la mémoire et le temps d'exécution.
— L'obligation de valider les nouveaux changements par un ou deux membres de l'équipe.

L'équipe du logiciel embarqué a automatisé la vérification de ces règles en utilisant :

— Uncrustify, un embellisseur de code, qui va gérer la partie styliste des règles.
— CPP-check, un analyseur statique du code, qui va détecter des erreurs d'optimisation ou des simples erreurs comme :
  - Dépassement de la taille d'un tableau
  - Les fuites de mémoires (allouer une partie de la mémoire sans la libérer)
  - Des erreurs de performances, comme la déclaration d'une variable dans le mauvais endroit.
— Azure, où les script vont s'exécuter pour vérifier si les règles de développement sont respectées en utilisant Uncrustify et CPP-check.

8.4 Établissement des tâches

La plate-forme Azure était utilisée pour planifier le développement de notre projet. Une étude a été menée pour préciser et planifier le développement du projet, et ainsi déclarer les tâches nécessaires au développement du projet. Après la réalisation de cette étape, une réunion est organisée pour déchiffrer les tâches, ce qui veut dire, préciser le temps nécessaire pour la réalisation de chaque tâche.

Les tâches étaient attribuées entre l'autre stagiaire et moi, selon nos préférences. Les tâches principales du projet des sondes sont illustrées dans la figure suivante :

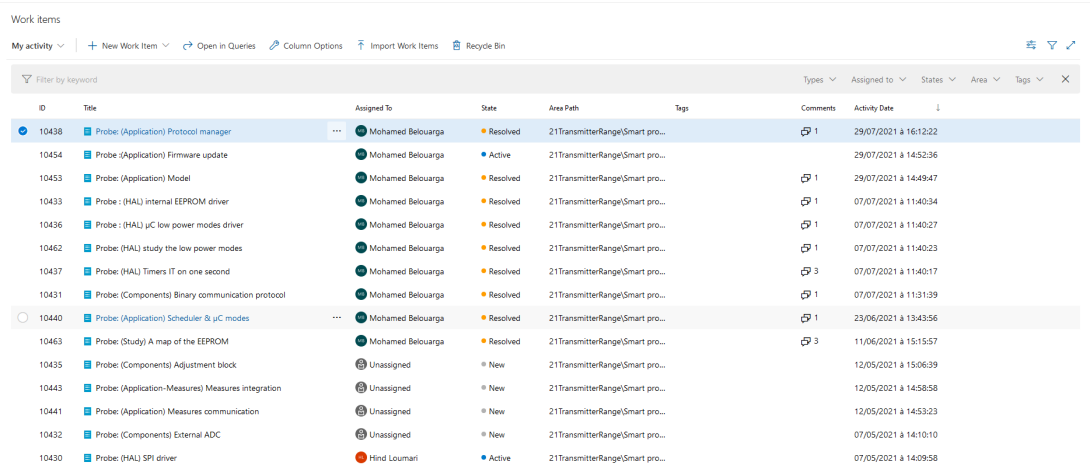

FIGURE 10 – Les tâches principales du projet des sondes intelligentes

8.5 Générateurs de documentation

Les générateurs de documentation sont des programmes utilisant les commentaires des fichiers sources pour générer la documentation d'un projet ou d'une librairie, en forme de HTML, pdf, latex ...

L'entreprise Sauermann m'a confié une tâche supplémentaire de trouver une solution pour générer la documentation de leurs projets.

8.5.1 *Les technologies présentes*

Il y a deux types de générateurs de documentation :

— Des générateurs à partir des langages de balisages comme markdown et reStructuredText : qui ne sont pas utiles pour notre usage comme MkDocs, Docsify, Couscous ... vu que nous sommes intéressés par ceux qui fonctionnent à partir des commentaires du code source.

— Des générateurs de documentation à partir du code source sont moins nombreux que les précédents. Il existe :

- Rdoc : destiné au langage Ruby
- GTK-Doc : utilisé par GTK+ et GNOME spécialisé en langage C.
- Natural Docs : qui prend en charge 21 langages, y compris le langage C, cependant, il génère seulement une documentation en HTML.
- HeaderDoc : un générateur de documentation d'Apple qui prend en charge plusieurs langages de programmation, mais il ne fonctionne pas sur Windows.

8.5.2 *Doxygen*

Après mon étude de plusieurs solutions et la réalisation d'une comparaison entre elles, mon équipe a choisi Doxygen, un générateur très utilisé qui génère une documentation à partir des fichiers en langages de balisage et des fichiers du code source.

Doxygen prend en charge le langage C sur tous les systèmes d'opération, son seul inconvénient, qui s'avère aussi un point fort, est l'obligation de le configurer avant l'utiliser.

Pour configurer Doxygen, Doxywizard est un programme qui génère le fichier de configuration Doxyfile à partir d'une interface graphique (Figure 11), et qui lance aussi Doxygen pour générer la documentation. Avec Doxywizard, Doxygen peut être configuré pour contenir des graphes qui montrent la liaison entre les différentes fonctions et les différents fichiers.

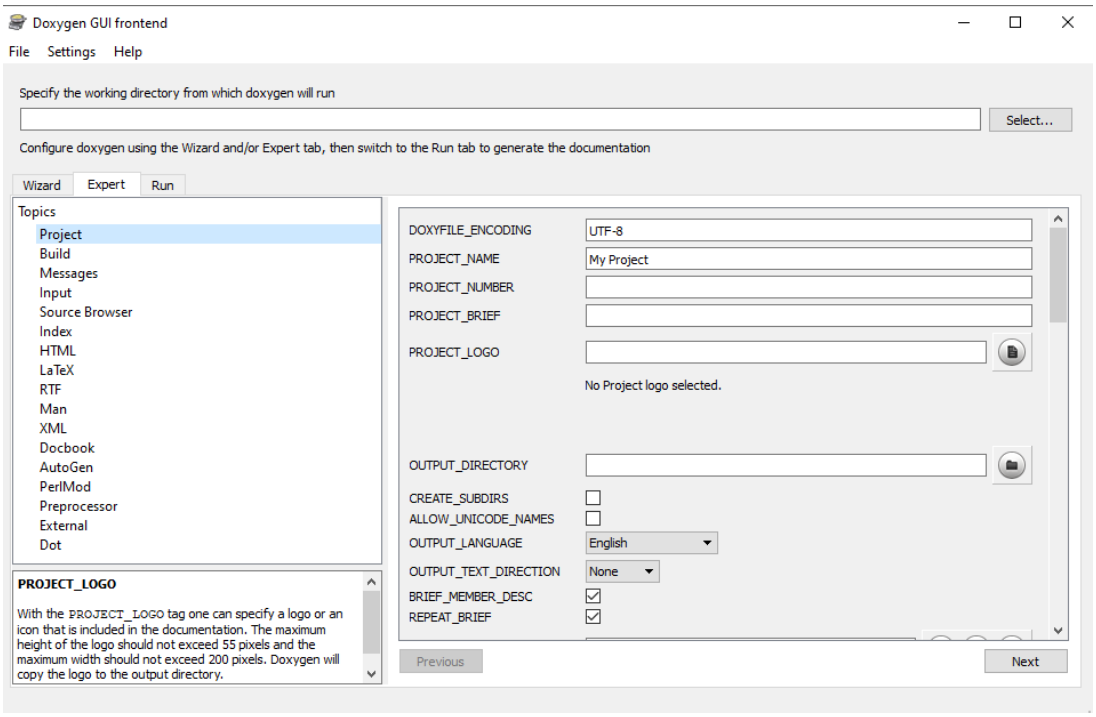

FIGURE 11 – L'interface graphique Doxywizard

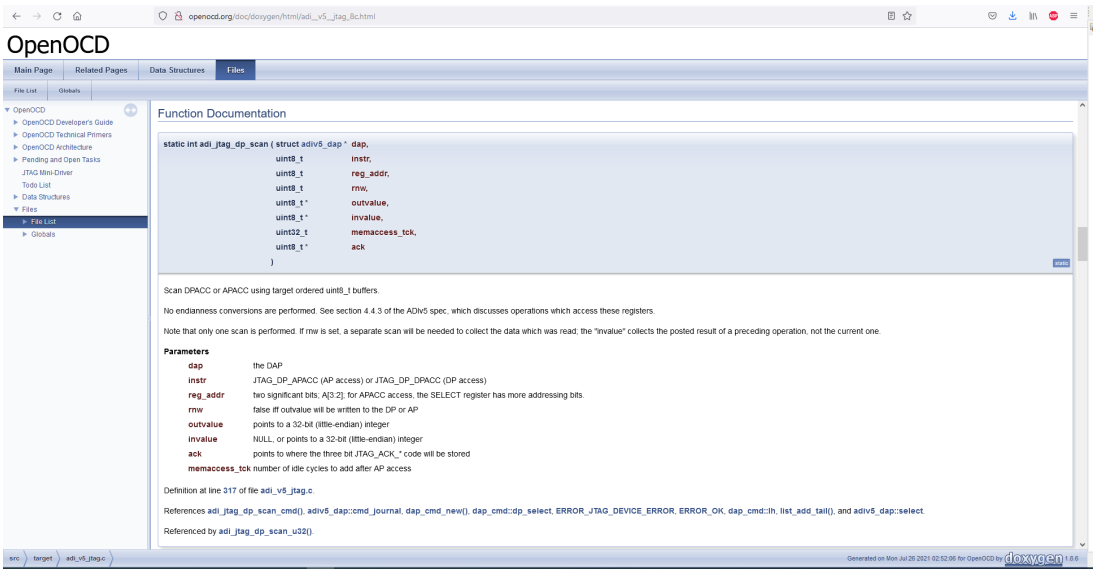

FIGURE 12 – Un exemple d'une documentation générée par Doxygen

8.5.3 JavaDoc

Pour que Doxygen génère la documentation, il faut que les commentaires respectent le format javaDoc, et qu'ils utilisent les tags javaDoc.

Pour illustrer l'utilisation de ces tags, voici un exemple d'une fonction documentée :

```
/**
 * @brief  This function permits to add two integers
 *
 * @param  a     The first integer
 * @param  b     The second integer
 * @return int   The result of the addition
 */
int Add(int a, int b)
{
    return a + b;
}
```

Les tags javaDocs sont très nombreux pour les citer tous dans le rapport.

8.5.4 Python pour des changements en masse

L'équipe du logiciel embarqué n'utilisait pas les commentaires JavaDoc pour commenter leurs projets, et ces commentaires étaient très nombreux pour les changer à la main, d'où le besoins d'utiliser un script pour changer le format de ces commentaires.

J'ai réalisé un script python afin de finir cette tâche, ce script devrait prendre en charge plusieurs exceptions et les gérer, parce que les fichiers n'étaient pas documentés de la même manière.

Le script python nous a permit de changer le format des centaines de commentaires dans une courte durée.

8.5.5 *Les règles de documentation*

La réalisation de cette tâche a abouti à la déclaration des règles de documentation, qui sont, en bref :

— Tous les fichiers doivent être documentés.
— Toutes les fonctions doivent être documentées.
— Les commentaires doivent être en format javaDoc.

9 Organisation de software

Vu le grand nombre des périphériques à gérer, les nombreux besoins logiciels et le besoin de pouvoir utiliser ces petit logiciels pour d'autres projets, une méthode de développement a été fixée.

9.1 Programmation modulaire

La programmation modulaire est une méthode de programmation qui consiste à utiliser plusieurs fichiers .c et .h pour construire des blocs de software indépendants, ainsi découper une grosse application en plusieurs modules. Cette méthode de développement permet le découpage du projet, de distribuer des tâches indépendantes entre les programmeurs, d'utiliser le même code pour plusieurs projets et de tester les différents blocs indépendamment.

Cette méthode de développement nécessite un makefile ou un programme de construction logicielle comme cmake, cependant, pour notre projet, l'IAR fait cette construction automatiquement.
Cette méthode de développement permettra à l'entreprise de réduire le temps et le coût de développement, grâce à la réutilisation des modules pour d'autres projets.

9.2 Architecture logicielle

Pour les projet des sondes, trois répertoires Git étaient utilisés, un qui s'appelle HAL_8 (HAL : Hardware Abstraction Layer), le deuxième s'appelle Components, ce dépôt git contient tous les modules qui seront possiblement utilisés pour d'autres projets, et le dernier s'appelle Applications, ce dernier Git contient les modules qui seront pas utilisés pour d'autres projet parce qu'ils ne sont pas indépendants, et leurs caractères ne permettent pas la réutilisation de ces modules.
Un autre répertoire qui s'appelle Drivers est fourni par la société ST Microelectronics, et qui contient des fonctions basiques pour manipuler les registres du microcontrôleur.

L'organisation des dépôts Git était d'ajouter le dépôt Drivers au dépôt HAL_8, d'ajouter le dépôt HAL_8 aux Components puisqu'il peut être réutilisé pour un autre projet. et d'ajouter le répertoire Components au dépôt Applications qui contient aussi les fichiers de configuration du projet.

La programmation modulaire est implémentée en utilisant cette arborescence :

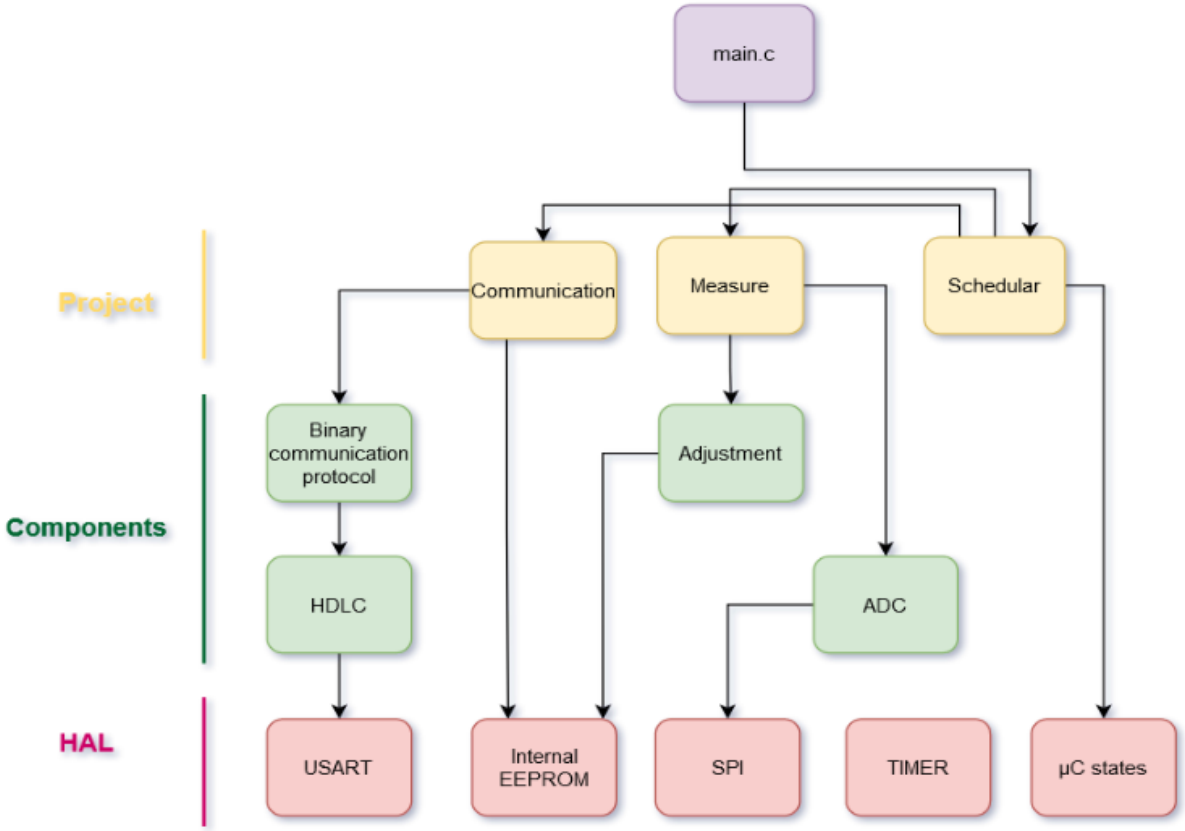

FIGURE 13 – L'architecture du projet des sondes intelligentes

Drivers

Les drivers sont des modules qui gèrent les périphériques du microcontrôleur ou qui font des petites fonctionnalités, en manipulant leurs registres, comme la création d'un octet dans la mémoire EEPROM, le changement d'un bit dans un registre...

HAL

C'est la couche d'abstraction du matériel qui permet la manipulation du hardware. C'est la seule couche qui peut accéder au matériel pour l'utiliser, toutes les autres couches doivent appeler cette couche quand elle doivent utiliser un des périphériques du microcontrôleur.

Cette couche permet une séparation entre le matériel et le logiciel, et cacher les détails techniques du matériel.

Components

Les Components seront des composants logiciels qui seront utilisés pour d'autres projets, comme HAL_8 qui peut être utilisé pour un autre projet à base de STM8, cette couche ne contient aucun composant spécifique à un projet. Cette couche va permettre à l'entreprise de réduire le temps et le coût de développement d'un autre projet.

Les blocs de ce dépôt sont développés pour qu'ils soient configurables pour un projet, les fichiers de configurations se situent dans le bloc Applications.

Applications

La partie Applications contient tous les modules qui ne sont pas réutilisables pour d'autres

projets, ces modules sont haut-niveau et sont spécifiques au projet.

10 Le développement du projet

Pendant la période de stage j'étais amené à développer plusieurs blocs pour le projet des sondes intelligentes. Certains de ces blocs sont indépendants et réutilisables pour d'autres projets, et certains blocs étaient spécifiques au projet des sondes.

10.1 Liaison série

Avant de commencer le développement, des études ont été réalisées afin de spécifier les besoins logiciels et les périphériques qui seront utilisés. Parmi ces études, une étude était faite pour proposer un protocole de communication.

Le protocole de communication a été proposé pour qu'il soit multicouche, la première couche va être le support physique. Afin de choisir un support physique, une étude a été réalisée aussi pour voir les supports que le microcontrôleur contient. La famille STM8L utilisée contient trois supports physiques :

I2C

I2C ou I²C est l'abréviation de son nom en anglais : Inter-Integrated Circuit (IIC), est un support de communication half-duplex. la liaison I2C est illustrée dans la figure suivante :

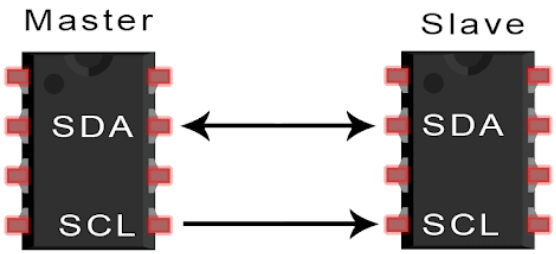

FIGURE 14 – La liaison I²C (sans la masse)

Le signal SDA est pour envoyer la data, et le signal SCL est imposé par le maître pour synchroniser la communication, en plus, il y a la masse qui lie les deux composants.

La forme d'un byte envoyé est illustrée dans la figure suivante :

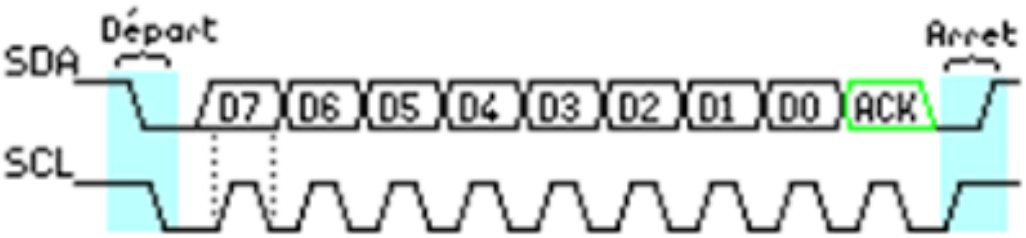

FIGURE 15 – L'architecture d'un byte envoyé avec l'I²C

Quand le slave reçoit un byte, il doit envoyer un bit de voltage bas pour faire une reconnaissance (ACK = acknowledgment).

Ce protocole est un protocole entre un "maître" et des "esclaves", quand cette relation n'est pas vraie, un mécanisme d'arbitrage doit être établi pour éviter les conflits entre les deux composants. Ce support était exclu comme choix, car il n'est pas adapté à notre cas.

SPI

SPI est l'abréviation du nom de support en anglais : Serial Peripheral Interface, ce support permet une communication duplex intégrale. La liaison SPI est représentée dans la figure suivante :

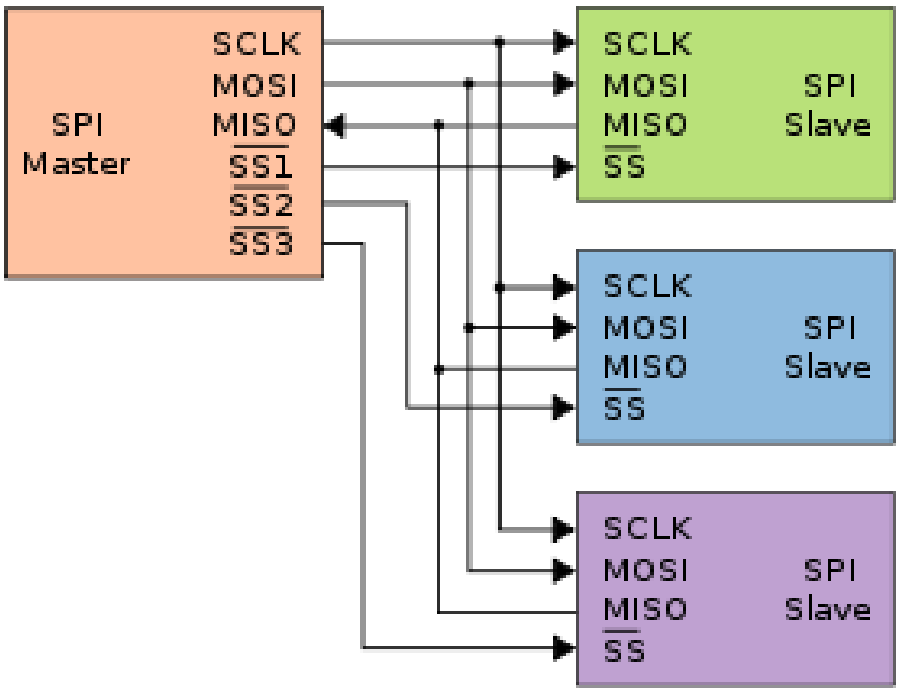

FIGURE 16 – La liaison SPI

Ce support est désigné pour des relations maître-esclaves, le maître peut dialoguer avec plusieur slaves en désignant quel composant va communiquer avec lui(via le mécanisme de "slave select"), or pour notre cas, nous n'avons pas cette relation, alors ce choix est aussi exclu.

USART

USART est une abréviation du nom de support en anglais : Universal Synchronous & Asynchronous Receiver Transmitter, ce support est un support de communication en duplex intégral, ce qui permet à deux équipement d'envoyer des trames et d'écouter en même temps, sans aucun conflit.

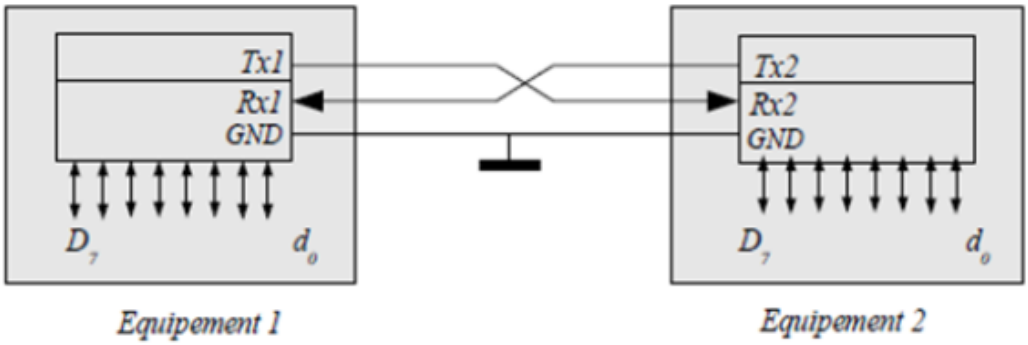

FIGURE 17 – La liaison USART (Asynchrone)

Tx est le signal de transmission, Rx est celui de la reception, et GND représente la masse. Ce support peut contenir une autre liaison pour synchroniser la communication(UART Synchrone) qui est l'horloge.

La forme d'un byte envoyé à travers l'USART est :

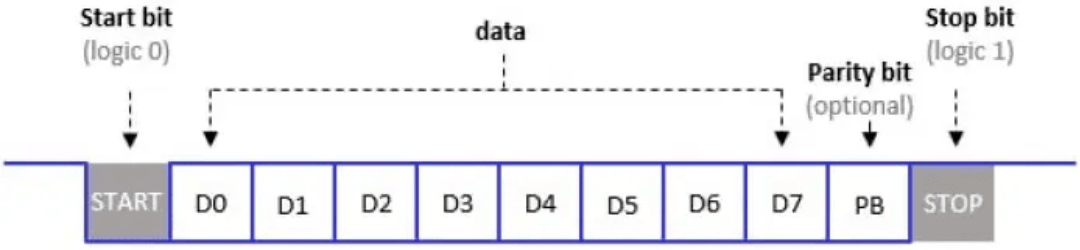

FIGURE 18 – L'architecture d'un byte envoyé avec l'USART

Remarque : le Stop bit peut être un bit, un bit et demi ou 2 bits selon la configuration de l'USART.

Le bit de parité (Parity bit) est optionnel. C'est un outil pour vérifier que le byte est bien reçu par le récepteur. Il doit aussi être configuré pour que le nombre des bits à 1 soit pairs ou impairs.

L'USART était choisi comme le support du protocole de la communication vu qu'il est duplex intégral, simple et très utilisé pour des usages pareils. Le bit de parité était désactivé, la taille de bit d'arrêt est un seul bit et la vitesse de transmission est 9600 baud.

10.2 EEPROM

Afin d'enregistrer la configuration de la sonde et les informations liées à une sonde, un bloc EEPROM était développé. Le microcontrôleur utilisé contient trois types de mémoire :

— 64KB de la mémoire Flash
— 4KB de la mémoire RAM
— 256 bytes d'EEPROM

La mémoire RAM est une mémoire volatile, cela veut dire quand elle est éteinte, elle perd toutes ses données. L'accès à cette mémoire est très rapide par rapport à la mémoire flash ou l'EEPROM.

La mémoire flash et la mémoire EEPROM sont deux mémoire non volatiles, et qui sont assez similaires, les différences principales sont :

— Le nombre d'écriture sur l'EEPROM est très grand par rapport à la mémoire flash, cela veut dire que la mémoire EEPROM peut supporter 100k nombre d'écriture et 3k pour la mémoire flash.
— Le changement d'un byte sur la mémoire EEPROM est possible, cependant, pour la mémoire flash les changement sont en blocs (pages de mémoire).

La mémoire flash est conçue pour les grosses quantités des données, tandis que l'EEPROM est conçue pour les petites quantités.

Pour écrire dans la mémoire EEPROM, il fallait respecter un mécanisme de protection qui protège l'EEPROM. Ce mécanisme est composé de deux étapes, qui sont écrire deux valeurs différentes dans le registre de protection de l'EEPROM, ces deux valeurs s'appellent les clés de MASS (Memory Access Security System) qui sont : 0xAE et 0x56. La lecture de la mémoire n'est pas protégée. Ce mécanisme est illustré dans la figure suivante :

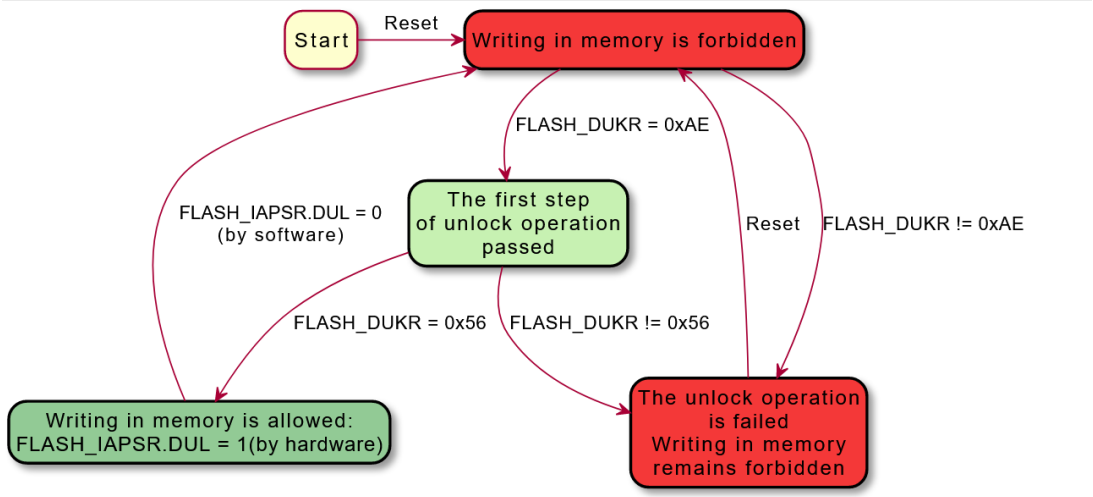

FIGURE 19 – Le mécanisme d'écriture dans la mémoire EEPROM (générée avec UML)

Organisation de l'EEPROM

Pour utiliser correctement l'EEPROM, un fichier d'organisation est indispensable (EEPROM mapping). Ce fichier est un fichier .h qui contient des defines pour les emplacements des paramètres de configuration de la sonde intelligente, ci-dessous un tableau qui résume ce que le fichier d'organisation de l'EEPROM contient :

Bloc d'EEPROM	Paramètre enregistré	la taille en bytes
Système	La version du fichier de mapping	1
Communication	Le numéro du modèle	10
	Le numéro de série	12
	Les types des capteurs	2
	L'intervalle des mesures	8
Mesures	Le type des mesures	1
Ajustage	Le tableau d'ajustage	8*5 = 40
	la date du premier ajustage	2
	la date du dernier ajustage	2
	Le temps d'usage	4
	La température d'ajustage	4
	La pression d'ajustage	4
	Le nombre d'ajustage	1
Configuration d'utilisateur	Le gain d'utilisateur	4
	L'offset de l'utilisateur	4
Checksum	Le checksum de l'EEPROM	2
Total	Le total	101

TABLE 1 – Le tableau d'organisation de l'EEPROM

UML

Un outil qui s'avère être très intéressant à l'usage, surtout pour faire la documentation du code est le langage UML qui m'a servi à générer la figure 19 pour faire la documentation du bloc qui gère l'EEPROM (Readme.md), ce langage de modélisation graphique est un langage qui est très utile pour faire des graphes, des diagrammes d'états, des schémas ...
Le site avec lequel j'ai compilé le code UML et généré les graphes est : http ://www.plantuml.com
Il est plus pratique de compiler les graphes en ligne qu'établir un environnement local pour le faire.

10.3 Ètude des modes de fonctionnement low-power

La sonde intelligente ne doit pas être gourmande en termes de consommation énergétique, vu qu'elle sera utilisée sur des enregistreurs de données (data loggers), qui sont des appareils fonctionnant avec des batteries, et qui collectent les données de mesures.

La famille SMT8L est une famille de microcontrôleurs de basse consommation, ils ont 3 types de fonctionnement, mode normal, mode basse consommation(low-power mode), mode d'arrêt(halt mode).

Après l'analyse de la datasheet du microcontrôleur, nous trouvons que pour passer d'un mode à l'autre, il y a soit des instructions assembleur à être appelées, soit des séquences logicielles ou des interruptions qui peuvent faire passer le microcontrôleur d'un mode à l'autre.

Pour bien comprendre la suite, deux points doivent être éclairés :

1. Le microcontrôleur contient deux horloges internes :

   — HSI(High Speed Internal clock) : La fréquence de cette horloge est 16MHz, et elle peut être divisée par un diviseur de 8 bits.
   — LSI(Low Speed Internal clock) : est une horloge de basse consommation, sa fréquence est 38kHz.

2. Le microcontrôleur peut se réveiller grâce à :

   — Une interruption : Quand une interruption arrive (si elle est activée), le microcontrôleur applique un mécanisme dit enregistre/restaure qui lui permet de sauter pour servir l'interruption et restaurer après le contexte précédent pour continuer le programme.
   — Un événement (Event) : La différence entre un évènement et une interruption est que les évènements n'appliquent pas le mécanisme enregistre/restaure, alors, quand un événement arrive le microcontrôleur se réveille et continue l'exécution du programme. Ce programme doit vérifier quel évènement l'a réveillé, par polling. Cette propriété permet au microcontrôleur d'économiser de l'énergie.

le diagramme d'état de la famille STM8L est :

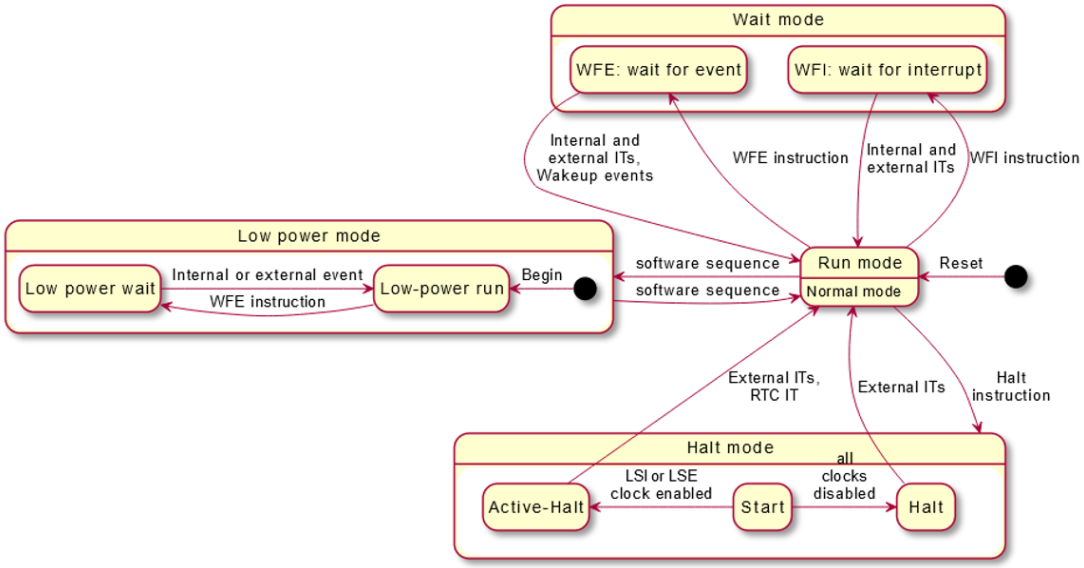

FIGURE 20 – Le diagramme d'état du microcontrôleur(généré avec UML)

La figure précédente montre trois états principaux du microcontrôleur, et qu'après un reset logiciel ou hardware, le micrôcontrôleur démarre en mode fonctionnement normal.

Pour entrer en état low-power, une séquence logicielle doit être appelée, ce mode va être expliqué dans la partie 10.3.2 .

Pour entrer en état d'arrêt ("halt"), une instruction assembleur est appelée(asm("halt")), cet état permet l'éteinte du processeur et de tous les périphériques, ce qui permet une économie d'énergie considérable si les interruptions internes ne sont pas nécessaires. Cet état contient deux modes, pour sélectionner le mode "active-halt", il faut activer une horloge low-power(interne ou externe) avant d'entrer en état d'arrêt ce qui active aussi les interruptions du RTC.

La différence entre WFE et WFI est que la première sert à attendre un "event", pendant que la deuxième sert à attendre une interruption. Dans les deux modes le processeur est éteint, ce qui permet d'économiser de l'énergie.

10.3.1 Comparaison entre les différents modes

Les différents modes de fonctionnement ont des caractéristiques différentes l'un de l'autre, une étude a été menée afin d'extraire ces caractéristiques et pour savoir quels sont les modes les plus adaptés au projet des sondes, les trois tableaux suivants montrent les caractéristiques des différents modes :

Mode	Horloge	CPU	Régulateur de voltage
Run	HSI	actif	actif
Low power run	LSI	actif	inactif

TABLE 2 – La comparaison entre les deux modes où le CPU est actif

Mode	Horloge	CPU	Régulateur de voltage	Les sources de réveil
Attente d'une interruption (WFI)	HSI	inactif	actif	Les interruptions
Attente d'un événement (WFE)	HSI	inactif	actif	Les évènements et les interruptions
Attente en basse consommation (low power wait)	LSI	inactif	inactif	Les évènements

TABLE 3 – La comparaison entre les trois modes de veille

Mode	Horloge	CPU	Périphériques	Les sources de réveil	Régulateur de voltage
Arrêt(halt)	LSI	inactif	inactif (sauf RTC*)	Interruptions externes (ou RTC)	inactif
Arrêt actif (active halt)	inactif	inactif	inactif	Interruptions externes	inactif

TABLE 4 – La comparaison entre les deux modes d'arrêt

*RTC : (Real Time Clock) horloge de temps réel.

En se basant sur les trois tableaux et selon les besoins des sondes en termes de périphériques, de sources d'interruption, les modes de fonctionnement seront choisis.

10.3.2 Implémentation

Les sondes sont des appareils communicants, alors, l'horloge LSI (38kHz) ne peut pas répondre aux besoins de la sonde quand elle communique avec les transmetteurs, vu que la vitesse de transmission ou de la réception est de 9600 baud. D'autre part, la sonde doit minimiser la consommation énergétique pour pouvoir fonctionner avec des transmetteurs à base de batteries. Vu que la sonde doit se réveiller chaque seconde pour envoyer les mesures, les modes d'arrêt ne sont pas envisagées, parce qu'ils ne permettent pas un réveil à partir d'un timer, qui est un périphérique interne. Les modes choisis sont le fonctionnement normal (normal run), et le mode

d'attente en basse consommation (low-power wait).

Pour implémenter ces deux modes, la data-sheet indique qu'il faut suivre un protocole, ce protocole est simplifié dans la figure suivante :

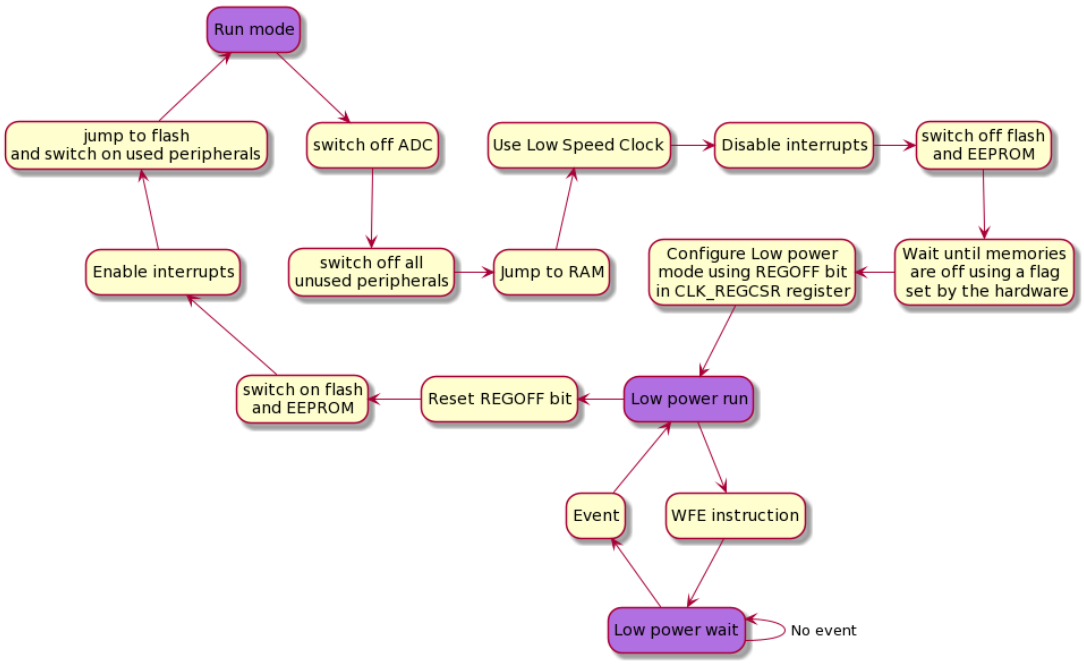

FIGURE 21 – Le processus pour passer entre les différents modes de fonctionnement (généré avec UML)

La figure précédente montre les deux séquences logicielles qu'il faut appeler pour passer du mode fonctionnement normal au mode "low-power run" et l'inverse. elle montre aussi comment passer du mode "low-power run" au mode "low-power wait". Ce qu'on peut comprendre de cette figure est que le microcontrôleur économise de l'énergie en fonctionnant avec une horloge lente (38kHz) et en éteignant la plupart des périphériques.

En mode "low-power run", le microcontrôleur exécute le code à partir de la RAM pendant que la mémoire RAM et la mémoire EEPROM sont éteintes. Les interruptions sont aussi désactivées pendant ce mode.

10.3.3 Performances

Afin de spécifier les performances des deux modes choisis, des tests de consommation et de la durée de réveil étaient menés, la diagramme en bâtons suivant montre la consommation énergétique du microcontrôleur :

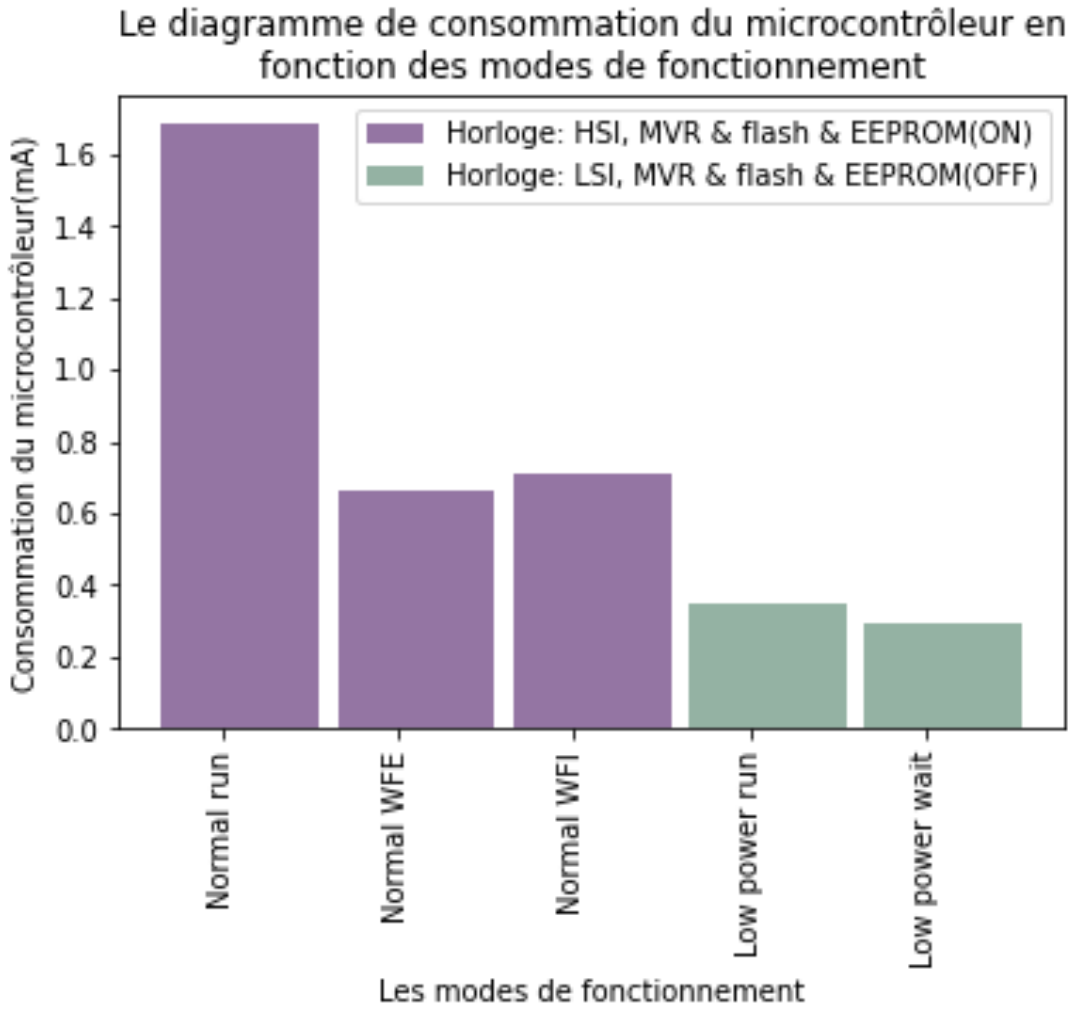

FIGURE 22 – Résultats des mesures sur les différents modes choisis (généré avec python)

Remarque : les conditions de ces tests étaient :
— Les périphériques sont en modes d'arrêt (leurs horloges ne sont pas activées).
— La carte de test contient quelques petits composants électroniques, cependant leur consommation est négligée devant la consommation du microcontrôleur.

Les résultats des tests montrent bien que les modes à basse consommation ne consomment pas beaucoup d'énergie en les comparant aux modes de fonctionnement normaux, ce qui montre l'efficacité de l'utilisation de ces modes.

Les durées de réveil ont été mesurées aussi, à l'aide d'un analyseur logique de type Ikalogic SQ200, cet analyseur logique permet de voir les variations des entrées/sorties du microcontrôleur à l'aide d'un logiciel dont le nom est ScanaStudio :

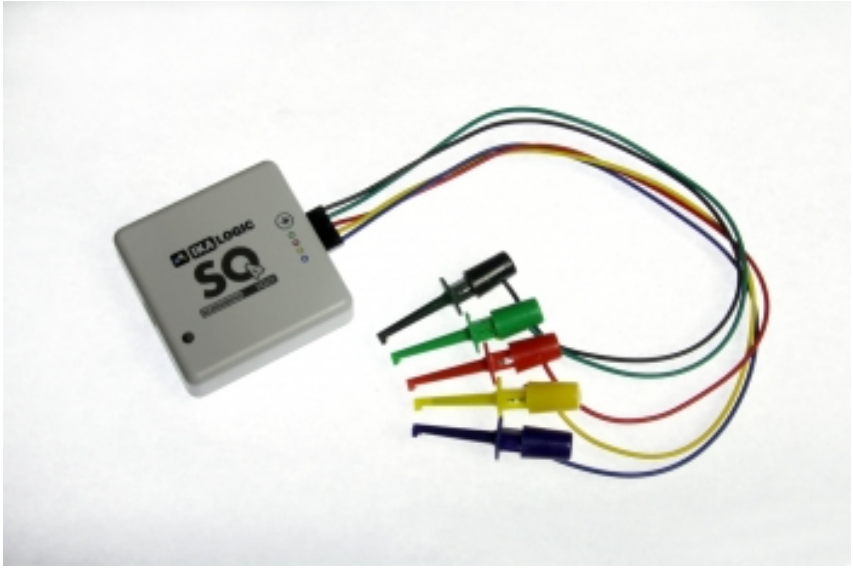

FIGURE 23 – L'analyseur logique ScanaQuad

En connectant cet analyseur à un ordinateur, et en utilisant le logiciel ScanaStudio, le microcontrôleur a été réveillé puis il était programmé pour qu'il change l'état d'une pin directement, cela va permettre de mesurer la durée de réveil, voici un exemple de ce que ScanaStudio nous montre, après une campagne de mesures :

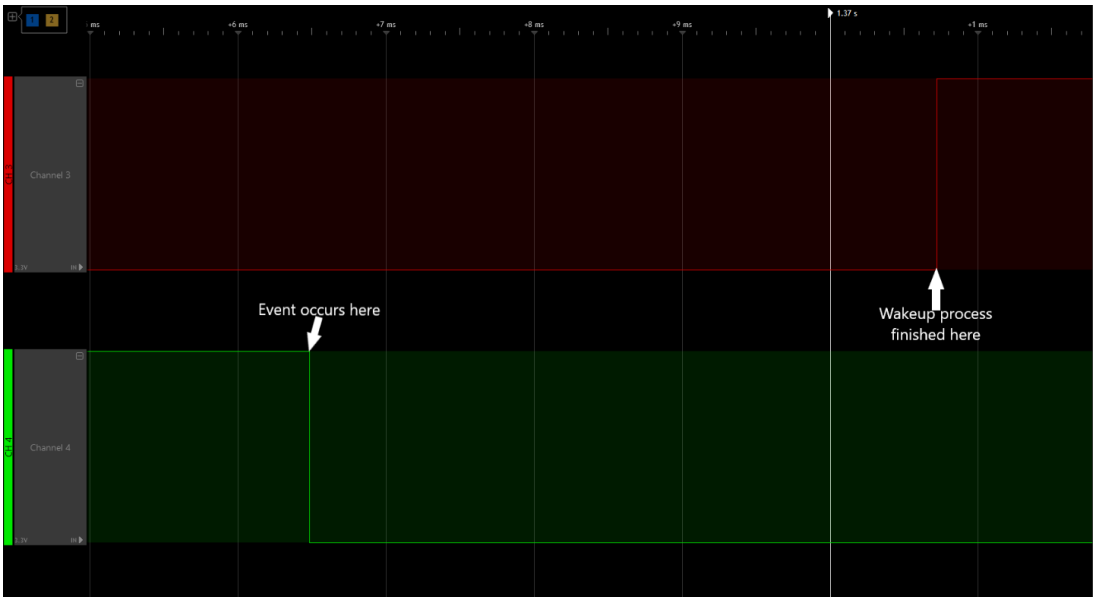

FIGURE 24 – Les résultats de ScanaStudio

Le tableau des résultats est le suivant :

Mode	La durée de réveil (µs)
WFI ou WFI	6
Attente en basse consommation	4246

TABLE 5 – La durée de réveil du microcontrôleur en fonction des modes de réveil

Ces derniers résultats montrent bien que la durée de réveil du mode "Attente en basse consommation" est très longue par rapport aux modes d'attente normaux, cela est dû à l'utilisation de l'horloge LSI (38kHz) contre HSI (2MHz). Cependant cela ne pose aucun souci pour répondre au cahier des charges.

10.4 Timers

Les timers sont des périphériques très importants, et qui seront utilisés pour différents buts. Les timers permettent de mesurer le temps à partir de l'horloge interne du microcontrôleur et en incrémentant un registre. Ces périphériques vont servir le microcontrôleur pour mesurer la durée entre deux tâches, par exemple : envoyer les mesures chaque seconde.

Ils peuvent aussi servir le microcontrôleur à sortir d'une boucle while grâce au calcul des tiques, par exemple, quand le microcontrôleur attend la modification d'un registre par le hardware dans une boucle while, si après 1000 tiques, le registre ne change pas, cela veut dire qu'un problème est arrivé, et il ne faut pas rester bloqué dans cette boucle. Cette technique d'utilisation des délais est très importante afin de ne pas rester bloqué dans des boucles sans fin.

En réalité, je n'ai pas utilisé les tiques, mais le temps en ms, cela était possible lors de l'initialisation du timer. En récupérant la fréquence de l'horloge, le timer est initialisé afin d'incrémenter son registre chaque ms.

10.5 Protocole de communication

Les spécifications du protocole de communication entre la sonde et le transmetteur sont les résultats de l'étude mentionnée dans la partie 10.1 page 21, ce protocole de communication était conçu pour qu'il soit multicouche en s'inspirant du modèle OSI(Open Systems Interconnection). Ces couches doivent être indépendantes, et chacune a sa propre tâche.

10.5.1 Architecture

Ce protocole de communication, qui est nommé BCP (Binary communication protocole), contient trois couches :

- USART : Le support physique de la communication.
- HDLC : La structure des trames.
- BCP Applicatif : l'organisation des échanges d'informations entre la sonde et le transmetteur.

10.5.2 HDLC

HDLC (High-Level Data Link Control) est un protocole de deuxième niveau du modèle OSI, il spécifie comment les paquets sont échangés, et la structure des paquets.

La structure des trames HDLC

La structure standard des trames HDLC est montrée dans la figure suivante :

Fanion de début	Adresse	Commande	Données	Frame Check Sequence	fanion de fin
8 bits (01111110)	8 bits	8 bits	...	16/32 bits	8 bits (01111110)

FIGURE 25 – La structure de la trame HDLC

Chaque élément de la structure HDLC a son propre rôle :

— Fanion : C'est le byte 0x7E qui déclare le début de la trame ou la fin de la trame si une trame est en cours d'envoie.
— Adresse : L'adresse de la destination du trame.
— Commande : pour Spécifier le type de la trame, plusieurs peuvent être définis.
— Données : Les données qui sont destinées à l'autre dispositif.
— Le checksum : est une somme pour détecter si la trame est bien arrivée, sans aucune erreur.

En prenant en compte les besoins des sondes, cette structure était personnalisée, afin de garder les éléments qui seront utilisés. L'adresse est abandonnée vu que l'échange sera seulement entre deux dispositifs, la partie commande est abandonnée aussi. La nouvelle structure est :

Flag	Information	FCS (CRC)	Flag
1 byte	N bytes	2 bytes	1 byte

FIGURE 26 – La structure de la trame HDLC personnalisée

CRC

La somme de contrôle(Checksum) est une courte séquence de bytes calculée à partir d'une somme de données, il est utilisé pour assurer l'intégrité des données et vérifier qu'aucun changement n'est arrivé. Elle est conçue pour que le moindre changement au niveau de ces données engendre un grand changement au niveau de la somme de contrôle, cette somme est utilisée pour le transfert des fichiers, des trames ...

Parmi les techniques de calcul de la somme de contrôle, le CRC (Contrôle de redondance cyclique) est très utilisé pour vérifier l'intégrité des trames et qu'aucune erreur de transmission n'est arrivée. Le CRC permet de détecter d'une manière efficace si un bit ou deux ont été changés. L'Ethernet par exemple utilise un CRC de 32 bits.

Mécanisme d'échappement

Le mécanisme d'échappement est utilisé pour éviter d'avoir le byte 0x7e au milieu de la trame, ce qui peut être considéré comme la fin de la trame. Ce mécanisme consiste à utiliser un byte d'échappement qui est 0x7d, donc si la trame doit contenir 0x7d, il est remplacé par deux bytes 0x7d 0x5d. Et pour échapper au byte 0x7e, il est remplacé par 0x7d 0x5e.

10.5.3 Le protocole de communication binaire

Le protocole de communication binaire manipule la partie information après l'encapsulation ou la décapsulation par le bloc HDLC. Ce protocole suit les règles suivantes :

1. Toutes les trames envoyées par le transmetteur doivent avoir une réponse de la part de la sonde, sauf les trames qui contiennent une erreur de CRC ou de mécanisme d'échappement.
2. Les réponses doivent avoir un des statuts spécifiés dans le tableau 8.
3. Toutes les trames doivent avoir un ID, et une réponse à une trame doit avoir son ID, cet ID est nommé requestID.
4. Les trames de réponses qui contiennent le statut "Arguments invalides"(voir le tableau 8) ne doivent pas contenir le requestID, parce que l'identification du requestID n'est pas précise.

5. La sonde peut aussi envoyer des trames, qui sont nommées des trames spontanées (Les erreurs et les mesures).
6. La structure des trames du transmetteur est :

Commande	Objet	Clé	RequestID	charge utile
1 byte	1 byte	1 byte	1 byte	N bytes

TABLE 6 – La structure d'une trame transmetteur avant l'encapsulation en HDLC

7. La structure des trames, de type réponse, de la sonde est :

Type de trame	Statut	requestID	charge utile
1 byte	1 byte	1 byte	N bytes

TABLE 7 – La structure d'une trame sonde avant l'encapsulation en HDLC

Si la trame est spontanée, comme les erreurs et les mesures, les deux bytes de statut de de requestID sont supprimés.

Byte de représentation	Description
0x01	Succès : la commande était exécutée avec succès
0x02	Arguments invalides : La commande est corrompue (syntaxe n'est pas respectée, nombre d'argument incorrecte)
0x03	La commande n'est pas attendu (clé ou objet introuvable)
0x04	Accès interdit : Les droits d'accès pour exécuter la commande ne sont pas accordés
0x05	Le temps d'exécution de la commande est trop long
0x06	Erreur interne : une erreur non documentée est arrivée

TABLE 8 – Les statuts des réponses de la sonde

Remarques :
— Les commandes utilisées sont : "GET"(0x01) et "SET"(0x02).
— Les objets utilisés sont : "Sensor"(0x01), "Device"(0x02) et "Adjustment"(0x03).
— Les clés sont spécifiées dans le tableau 1, plus la clè "la version du firmware".

10.5.4 Les tests du protocole de communication

Afin de tester la communication, et en prenant en compte que le transmetteur n'est pas encore fabriqué, un script python simulant le transmetteur a été utilisé. Ce script manipule un composant qui s'appelle FTDI, voir la figure 27, permettant d'envoyer et de recevoir des trames via l'UART. Ce script envoie des trames au microcontrôleur et attend la réponse du microcontrôleur pour analyser cette réponse. Ces tests m'ont aidé à détecter et corriger plusieurs bogues et mal-fonctionnements.

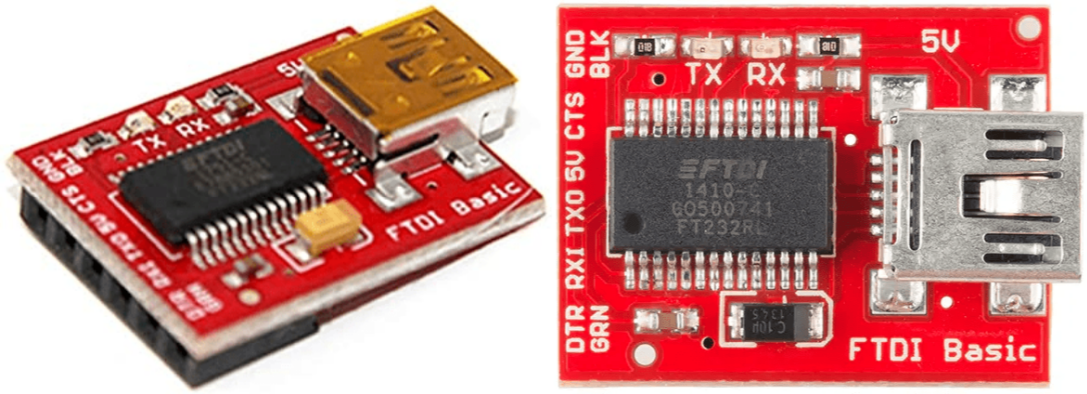

FIGURE 27 – Le FTDI utilisé pour envoyer et recevoir des trames via l'UART

La réalisation de ces tests a permit la conception de trois librairies python qui sont :
— FTDI : Une librairie qui gère le FTDI
— HDLC : Une librairie qui gère l'encapsulation et la décapsulation des trames en python.
— pdfGeneration : une librairie pour générer les une pdf avec les résultats des tests.

Ces librairies seront utilisées pour les prochains tests de l'entreprise. La figure suivante montre un exemple de pdf généré après les tests avec les résultats de ces tests :

Test Results

tested parameter	command	status of the answer	Test result
Mapping version test	Set	Success	SUCCESS
Mapping version test	Get	Success	SUCCESS
Firmware version test	Set	access is not permitted	SUCCESS
Firmware version test	Get	Success	SUCCESS
Sensor type test	Set	Success	SUCCESS
Sensor type test	Get	Success	SUCCESS
Number of adjustOps test	Set	access is not permitted	SUCCESS
Number of adjustOps test	Get	Success	SUCCESS
non-existent	Set	object or key not found	SUCCESS
non-existent	Get	object or key not found	SUCCESS

28/07/2021 13:44:42

FIGURE 28 – Un exemple des résultats des tests en python

Les tests ne sont pas tout le temps en succès, il arrive parfois où des erreur de CRC se produisent, et le pdf de tests généré à la fin de ce stage contient 3 pages avec plus de paramètres testés.

10.5.5 La mise à jour en USART

L'équipe électronique a décidé de changer le microcontrôleur utilisé, elle a opté pour un STM32. Grâce à l'architecture logicielle sur laquelle le projet est basé (HAL, components, application), la migration de STM8 vers STM32 n'était pas compliquée. Seulement la partie HAL qui a été remplacée par un répertoire Git de HAL pour STM32 développé par l'entreprise pour un autre projet. L'établissement de tout le protocole de communication de nouveau a été facile et les blocs HAL, pour STM8, développés précédemment seront utilisés pour un autre projet que l'entreprise a déjà entamé.

Les sondes intelligentes ne sont pas connectées aux serveurs de l'entreprise, c'est pourquoi, il faut déployer une solution pour mettre à jour leur firmware. La solution était, après avoir prouvé le concept par une étude et des tests, que le transmetteur, qui est connecté aux serveurs de l'entreprise, fasse la mise à jour de la sonde via l'UART. Il s'est avéré que les microcontrôleurs STM32 ont un bootloader qui peut être activé ou désactivé (AN2606), ce bootloader a son propre protocole de communication spécifié par l'«Application Note» AN3155 de la sociétéST Microelectronics.

Le bootloader peut être utilisé via SPI, I2C ou UART pour les dernières versions, et il permet de faire des commandes pour supprimer une partie de la mémoire flash, écraser toute la mémoire, écrire dans la mémoire flash ...

J'ai commencé à concevoir un bloc qui va gérer cette partie du côté de transmetteur, mais vu l'arrivée de la fin de mon stage je n'ai pas pu la finir.

11 Résultats

Le développement modulaire, l'intégration continue et l'organisation du projet sont tous des facteurs qui nous ont aidés à bien faire avancer le projet. D'autre part, j'ai bien réalisé une montée en compétences techniques, en compétences d'organisation et de communication, et aussi le développement de mon esprit ingénieur.

En ce qui concerne mon niveau technique, je suis arrivée à développer des petits blocs software indépendants, cela m'a permit de développer mes capacités en langage C, et de voir la développement embarqué d'une autre manière, en plus, j'ai réussi à utiliser le langage python pour des buts plus concrets, soit pour faire des tests ou pour faire des tâches précises en masse. J'ai développé d'autres capacités techniques, comme la conception logicielle et de ne pas voir un programme embarqué comme un seul bloc. J'ai aussi compris la particularité du code embarqué, un code embarqué doit être minimal et ne doit pas boguer complètement et rester bloqué dans une fonction.

J'ai bien développé aussi mon esprit d'ingénieur, précisément, ne pas attendre une information prête à utiliser mais aller chercher l'information et devenir autonome dans mes propres tâches. Cela n'exclut pas d'aller demander de l'aide à d'autres collègues afin de ne pas rester bloqué sur un point précis, mais une phase de recherche est importante pour comprendre toutes les particularités d'une tâche.

L'organisation du travail est un point très important que je négligeais avant de voir, au sein de Sauermann, à quel point elle peut faire avancer un projet, ou le retarder voir ne le finir jamais. L'organisation d'un projet de l'embarqué commence par la réalisation du cahier des charges afin de ne pas se perdre au milieu du projet, l'établissement et la configuration de la toolchain,

l'organisation logicielle, le découpage du projet en blocs et en tâches, mais aussi des faire des recherches sur des aspects techniques pour monter en compétences et essayer de déployer autres solutions plus optimisées. Tout cela m'a permis aussi de voir et comprendre tout le processus de la conception d'un nouveau produit.

Les résultats obtenus à la fin de ce stage, en terme de projet, et qui sont liés au cahier des charge, vu que le but était dès le début le respect du cahier des charges, ce sont la réalisation de toute la partie communication de la sonde avec le transmetteur, l'interchangeabilité est aussi assurée numériquement via le protocole de la communication binaire conçu. Une partie du management des erreurs était faite par le protocole de communication qui renvoie un statut à chaque trame reçue.

Pour finir le projet, il reste à développer la partie d'ajustage des mesures, et gérer les retours de l'équipe Marketing et l'équipe Test & Validation.

12 Difficultés rencontrées

Lors de ce stage, j'ai rencontré plusieurs problèmes, qui sont majoritairement techniques mais aussi méthodologique.

Parmi ces problèmes, j'ai rencontré des difficultés à m'habituer à Git, et comprendre comment il marche, cependant à force de l'utiliser, j'ai compris mieux son fonctionnement et j'arrivais à l'utiliser aisément.

J'ai aussi rencontré des problèmes lors de l'implémentation des modes de basse consommation, je n'ai trouvé aucun exemple sur internet de cette implémentation. la documentation était claire, mais techniquement le changement des modes n'était pas si facile. Le but était de faire des fonctions qui changent le mode de fonctionnement du microcontrôleur (voir la figure **??**), j'ai rencontré plusieur problèmes pour effectuer les différentes étapes de changement du mode de fonctionnement, par exemple, je ne savais pas comment sauter et exécuter du code à partir de la RAM. Après avoir lu "IAR C/C++ Development Guide" j'ai trouvé la solution qui est l'utilisation de l'intrinsèque "__ramfunc", qui est un des intrinsèques de l'IAR, et le compilateur fait en sorte qu'une fonction de type "__ramfunc" est copiée en RAM au démarrage du microcontrôleur, et elle s'exécute à partir de la mémoire RAM quand elle est appelée. Un autre problème c'est quand on est en RAM, nous ne pouvons pas appeler les fonction de la mémoire flash, alors pour passer au mode WFE, via la fonction wfe(), j'étais obligé de la faire directement en assembleur (asm("wfe")).

Un autre problème, vu que j'avais pas fait attention à la pin SWIM, qui est la pin 1, j'ai essayé de réveiller le microcontrôleur en utilisant une interruption externe sur cette pin, ce qui n'a pas marché, parce que en mode débogage, cette pin est utilisée tout le temps par le débogueur ST-LINK. Donc, si cette pin est utilisée pour réveiller le microcontrôleur en mode débogage, le microcontrôleur va se réveiller directement par le débogueur ST-LINK.

J'avais aussi des problèmes lors de l'extraction des informations de la datasheet, cette datasheet est pleine d'informations, que nous ne voulons pas forcément. Cependant parfois je sautais des informations utiles à mon application, ou je n'accordais pas suffisamment d'attention à cette information, ce qui m'a fait perdre beaucoup du temps, surtout pour des applications particulières comme l'implémentation des modes de basse consommation.

Le temps aussi était très limité, un projet de tel ampleur, ne peut pas être fait en six mois, je l'ai trouvé très grand par rapport à ce que j'ai fait à l'école, mais j'ai réalisé aussi que la majorité des projets commerciaux sont des très grands projets, même pour toute une équipe.

J'ai rencontré un autre problème lors de la présentation de mes demandes d'intégration du code, je n'arrivais pas à bien présenter mes demandes et les expliquer, les autres développeurs ne connaissaient pas forcément le contexte de ma demande, et j'ai commis des erreurs comme expliquer le code à la place d'expliquer l'utilité de ce code. Mon maître de stage et le manager de l'équipe m'ont donné des conseils afin d'améliorer mes présentations, et m'ont invité à voir d'autres présentations. J'ai travaillé sur ce problème, et j'ai eu des bons retours de la part de mon maître de stage, mais il faut que je travaille encore sur ce problème.

13 Conclusion

Grâce aux équipes du département Recherche & Développement, le groupe Sauermann continue à développer des nouveaux appareils et pousser la limite de la technologie dans le domaine des instruments de mesure. Quant à mon stage, la communication est totalement faite, l'interchangeabilité sera assurée via l'utilisation du même protocole de communication sur les transmetteurs, ce qui reste c'est le développement de la partie mesure, et l'amélioration de la communication, en ajoutant plus de paramètres échangés et plus de commandes. Les tests en python pourront être développés pour assurer un bon fonctionnement, et générer un rapport de tests plus détaillé.

Personnellement, je comprends mieux l'utilité de l'organisation du développement, soit entre les membres d'une équipe, ou même l'organisation du code(les règles de développement, les commentaires ...). j'ai pu aussi développer mon niveau en langage C et python, et j'ai appris le langage UML pour générer des graphes. D'autre part, j'ai compris l'importance des langages de script comme python dans le développement embarqué.

Maintenant, je comprends mieux le fonctionnement des microcontrôleurs, comme j'ai eu l'occasion de développer plusieurs blocs pour exploiter les périphériques. J'ai découvert aussi l'importance de l'automatisation des tâches répétées, et la méthode Agile/Scrum.

Ce stage m'a permit d'élargir mes connaissances et de comprendre le développement embarqué des microcontrôleurs en particulier, et le développement dit bare-metal en général.

Références

[1] AN2606. *STM32 microcontroller system memory boot mode*. URL : http://www.st.com/resource/en/application%5C_note/cd00167594-stm32-microcontroller-system-memory-boot-mode-stmicroelectronics.pdf.

[2] AN3147. *Power management in STM8L and STM8AL*. URL : www.st.com/resource/en/application_note/cd00263631-power-management-in-stm8l-and-stm8al-stmicroelectronics.pdf.

[3] AN3155. *USART protocol used in the STM32 bootloader*. URL : www.st.com/resource/en/application_note/cd00264342-usart-protocol-used-in-the-stm32-bootloader-stmicroelectronics.pdf.

[4] CPP-CHECK. *Analyseur statique du code*. URL : http://cppcheck.sourceforge.net/.

[5] DOXYGEN. URL : https://www.doxygen.nl/index.html. Générateur de documentation.

[6] FTDI. *Convertisseur USB vers USART*. URL : https://ftdichip.com/.

[7] Sauermann GROUP. *Le site de Sauermann*. URL : https://sauermanngroup.com/fr-FR.

[8] HDLC. *High-Level Data Link Control*. URL : https://datatracker.ietf.org/doc/html/rfc4349.

[9] IAR. *IAR embedded workbench*. URL : https://www.iar.com/.

[10] IAR IDE. *IAR C/C++ Development Guide*. URL : https://wwwfiles.iar.com/arm/webic/doc/EWARM_DevelopmentGuide.ENU.pdf.

[11] PYSERIAL. *La librairie python pour utiliser le FTDI*. URL : https://pythonhosted.org/pyserial/.

[12] REPORTLAB. *La librairie python pour générer des pdf*. URL : https://www.reportlab.com/opensource/.

[13] UM0470. *STM8 SWIM communication protocol and debug module*. URL : https://www.st.com/resource/en/user_manual/cd00173911-stm8-swim-communication-protocol-and-debug-module-stmicroelectronics.pdf.

[14] UNCRUSTIFY. *Embellisseur du code*. URL : https://github.com/uncrustify/uncrustify.