Projet fin d'étude : Le Sportif Augmenté
Rapport final

DELOBELLE Matthieu

26 février 2019

Encadrants :
Thomas VANTROYS - Alexandre BOE - Xavier REDON

Table des matières

Introduction

Afin de conclure ma cinquième année ainsi que mon cursus scolaire au sein de la formation Informatique Microélectronique et Automatique, spécialité Système Communiquant, à Polytech Lille, il m'a été demandé de réaliser un projet de fin d'étude.

L'objectif de ce projet est d'allier l'électronique et l'informatique, les deux matières phares de ma formation afin de pouvoir proposer un ensemble d'objets connectés, ici dans le cadre du sport.

Je présenterai dans un premier temps le contexte du projet, en introduisant les objectifs et le cahier des charges initial. Puis je traiterai des différents travaux majeurs réalisés durant le projet, avant de conclure par une rétrospective sur l'année et les pistes d'amélioration possibles du projet.

Remerciements

Je tiens à remercier Alexandre BOE et Thomas VANTROYS pour leur aide, notamment dans la mise en place du projet à ses débuts. Merci également pour les diverses entre-vues.

Merci Xavier REDON pour son aide et le prêt de matériel express qui m'a permis de travailler jusqu'à la fin du projet dans de bonnes conditions.

Grand merci à Thierry FLAMEN, pour son aide et ses précieux conseils dans la réalisation des différents PCB, ainsi que pour son temps et sa confiance en cette fin de projet.

1 Contexte du projet

1.1 Présentation

De nos jours, les objets connectés destinés aux sportifs sont de plus en plus répandus. Qu'il s'agisse des équipements de pointes des grands clubs de sports ou des petits bracelets calorimétriques destinés aux particuliers, le marché du gadget sportif est en plein essor.

Cependant les produits de pointes seront inaccessibles au commun des mortels, et des équipements comme l'AppleWatch offrant la possibilité de mesurer l'activité physique ne permettent toujours pas d'évaluer les performances de l'utilisateur. De plus cette montre n'est compatible qu'avec les produits estampillés d'une pomme, et coûte assez cher..

FIGURE 1 – Montre connecté de chez Apple

Des alternatives existent, comme les bracelets FitBit qui proposent un relevé des dépenses calorifiques ou du rythme cardiaque pendant l'effort. Mais encore une fois, très peu d'informations quant aux performances.

1.2 Objectifs

L'objectif du projet est alors de proposer une alternative à ces produits, le plus ouvert et générique possible afin de pouvoir l'inter-connecter avec différents capteurs et périphériques. Et ce, dans le but de rendre le produit accessible et compatible à tous les sports. Permettre une certaine liberté quant aux capteurs connectés afin d'être capable de mesurer l'accélération ou la position de certains muscles durant l'effort, ce qui permettrait de juger les performances de l'utilisateur.

Il sera donc nécessaire de mettre en place diverses communications sans fils, afin de ne pas gêner les performances de l'utilisateur, en proposant une autonomie suffisante à un match ou une séance d'entraînement. L'affichage

de ses données sera disponible sur un terminal sur le bord du terrain (PC/-Tablette Android) afin que l'entraîneur puisse définir la suite des exercices et corriger les défauts techniques des joueurs.

1.3 Définition du cahier des charges initial

Pour réaliser ce projet, certaines tâches devront être réalisées :

- Concevoir une architecture réseau permettant la récolte des données jusqu'au PC
  - Définition de la typologie du(es) réseau(x) utilisé(s)
  - Définition des technologies utilisées
  - Définition du protocole de communication et des trames
- Réaliser les différentes cartes nécessaires au fonctionnement de l'architecture réseau
  - Cartes de prototypage afin de valider le modèle et faire fonctionner les différents modules de communication
  - Conception des PCB intégrant les modules
  - Fabrication des packagings
- Proposer une application sur PC afin de pouvoir visualiser les données

2 Travail réalisé

2.1 Définition de l'architecture réseau

Afin de mettre en place le système, il est nécessaire de définir dans un premier temps la typologie des réseaux ainsi que les technologies utilisées. Un terrain réglementaire en compétition internationale mesure 105*68m, ainsi en supposant que l'on place le moniteur sur le côté du terrain, en son milieu, les données ont dans le pire cas, environ 85m à parcourir.

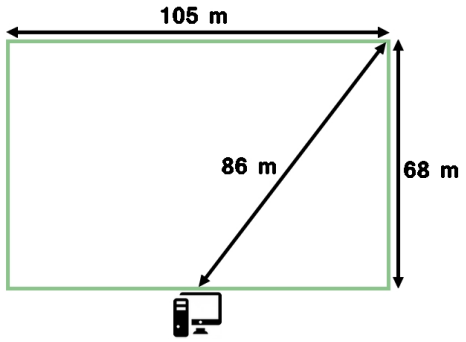

FIGURE 2 – Terrain de foot réglementaire

Ainsi il faut un dispositif sans fil permettant de couvrir une telle portée. Plusieurs choix s'offrent à nous. Le choix d'une communication avec Wifi ou HiperLAN est vite écartée car leur consommation énergique est assez élevé. Le module Xbee de chez Digi International propose une communication jusqu'à 100m en champ ouvert (ce qui est le cas d'un terrain de foot). On se focalisera donc sur cette technologie, du moins pour la communication longue portée.

En effet, qui dit plus haute portée de communication, dit plus haute consommation énergique, et qui dit donc besoin d'une plus grande capacité de batterie, qui sera donc plus grosse et plus lourde. On ne peut se permettre d'embarquer une batterie trop imposante sur chaque capteur présent sur le joueur. Ainsi je décide donc de séparer la récolte de données et l'émission au moniteur.

De ce fait, une base mobile sera installée dans le dos du joueur (entre les omoplates) afin de gêner le moins possible ses performances. C'est celle-ci qui fera l'émission vers le moniteur, mais elle communiquera également avec les différents capteurs via un réseau en étoile utilisant une technologie

à plus faible portée. Je souhaitais au départ utiliser la technologie ANT+ afin de rendre le système compatible avec certains dispositifs déjà existant. Les difficultés à trouver de la documentation complète ou des modules de communication facilement utilisables sans nécessiter un périphérique propriétaire, ainsi que les problèmes de commandes m'ont poussé à partir vers une technologie plus ouverte, à savoir le Bluetooth Low Energy.

Ainsi, si l'on suit cette logique, on peut schématiser le réseau global comme suit :

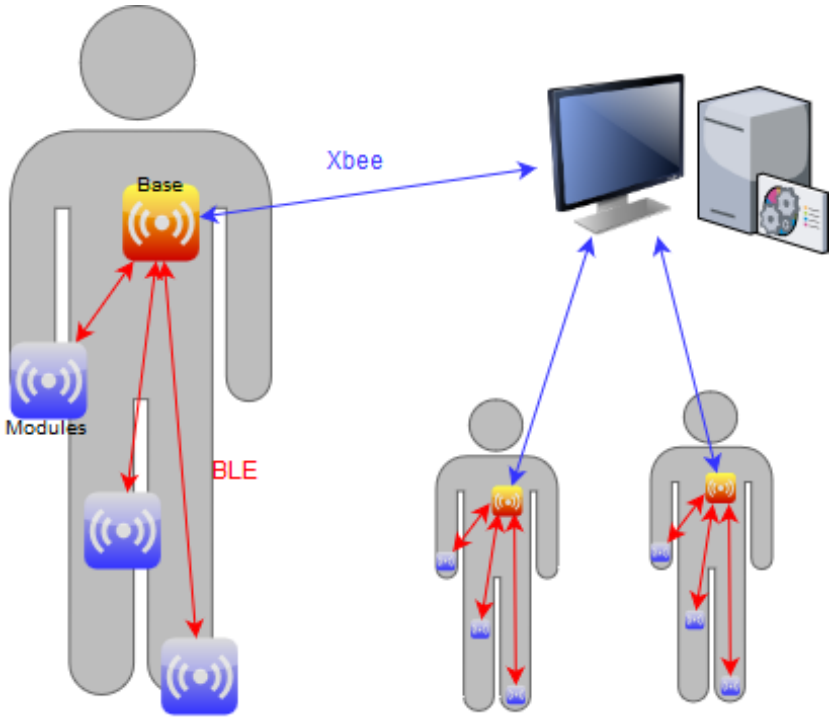

FIGURE 3 – Schema global du réseau

2.2 Essais d'utilisation des modules BLE

La mise en place du système implique l'utilisation de module BLE. Cependant il s'agit d'une technologie peu utilisée durant nos projets. Il est donc nécessaire de se faire la main dessus. J'ai pu commander un lot de module BLE de chez Cypress, des CYBLE-212023. Il s'agit de modules assez peu consommateurs d'énergie et assez petits (14*19mm) qui embarque directement un Cortex-M0.

L'objectif est alors d'essayer de les faire fonctionner, et de pouvoir les reprogrammer, ou les interfacer avec un autre Cortex-M0 afin de pouvoir réaliser les communication. Pour ce faire, il faut d'abord être capable de se brancher dessus. Il a donc fallut réaliser une carte de connexion car la connectique du composant n'est pas standard, et qu'il n'y a pas de librairie incluant ce composant sur le net.

2.2.1 Réalisation du PCB

J'ai dans un premier temps réaliser une librairie Altium permettant d'utiliser ces modules BLE. L'objectif n'était pas de simplement faire une connectique vers des headers, mais bien de fournir un composant utilisable sur Altium permettant de placer le module BLE facilement sur les différentes cartes que j'aurai a faire.

Ainsi j'ai pu faire un connecteur permettant d'alimenter le module et de pouvoir le tester.

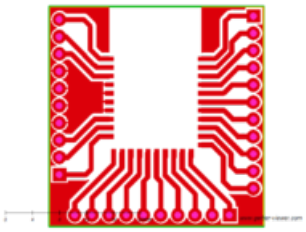

FIGURE 4 – PCB de branchement du CYBLE-212023

2.2.2 Essais d'utilisation du module

J'ai pu donc essayer de faire fonctionner mes modules en les alimentant en 3V3 (j'ai pour cela utilisé une carte NUCLEO)

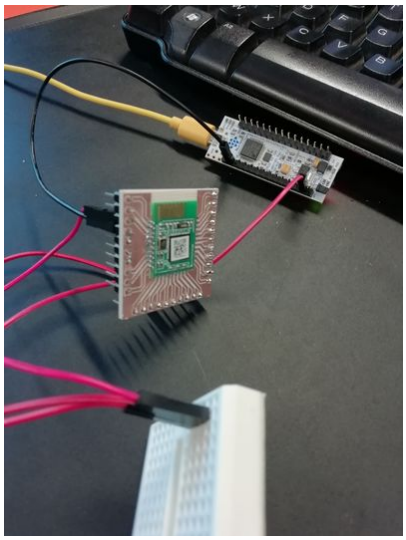

FIGURE 5 – Montage utilisant le CYBLE

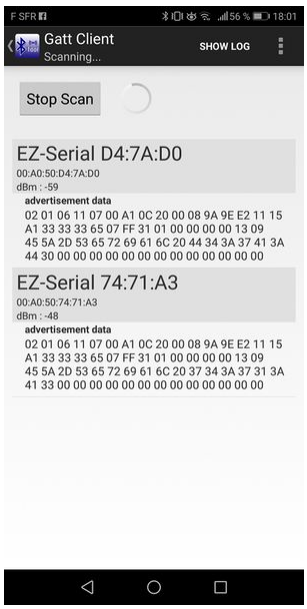

FIGURE 6 – Client BLE sur Android

Les modules sont donc fonctionnels, s'alimentent correctement et se broadcastent en tant que GATT Serveur, leur mode par défaut. J'ai d'ailleurs pu les détecter grâce à un client BLE installé sur mon téléphone. Cependant malgré les essais de changement de mode des modules (pour en passer un en tant que GATT Client, afin de permettre une communication) avec les logiciels CySmart et Psoc Programmer du constructeur, pas moyen de changer leur fonctionnement. De plus, je n'ai pas réussi à récupérer les informations du Cortex M0 interne grâce à un STLINK-V2 connecté aux broches SWD du module..

Par manque de temps et étant donné qu'il reste pas mal de travail, je décide de mettre le BLE de côté afin de me concentrer sur le reste du projet. A des fins de démonstration celui-ci sera remplacé dans un premier prototype par du Xbee (que je maîtrise déjà un peu plus). Il ne restera plus qu'à retravailler sur le BLE par la suite en l'intégrant dans le système complet.

2.3 Réalisation d'un duplicata de NUCLEO-L031K6

Afin de réaliser mes différentes cartes, j'utiliserai des contrôleur de type Cortex de chez ST. J'ai choisi comme référence le STM32L031K6 (Cortex M0) pour les capteurs embarqués et STM32L152RE (Cortex M3) pour la base mobile. J'ai pu donc travailler sur les cartes NUCLEOs vendu par le fabricant qui embarquent ces même contrôleur.

Cependant à l'instar des cartes Arduino pour les ATMega, beaucoup d'éléments supplémentaires sont présent sur les cartes NUCLEOs qui permettent la protection, le téléversement, ou l'inter-connection du MCU. Cependant tous ces éléments prennent de la place et, dans le cadre d'un capteur embarqué dans le domaine du sport, il n'est pas envisageable de travailler avec de trop gros périphériques, car ils risquent de gêner l'athlète.

Il est donc nécessaire de dupliquer les cartes NUCLEOs en n'extrayant que la partie incluant le MCU embarqué, ainsi que son environnement passif primordial à son fonctionnement. Pour ce faire ST propose les schematics des cartes de prototypages en open-source. Il a donc été possible de récupérer un exemple de l'environnement du MCU. Et en comparant cet exemple à la datasheet du composant, il a été possible de réaliser un schematic, puis un PCB de la carte.

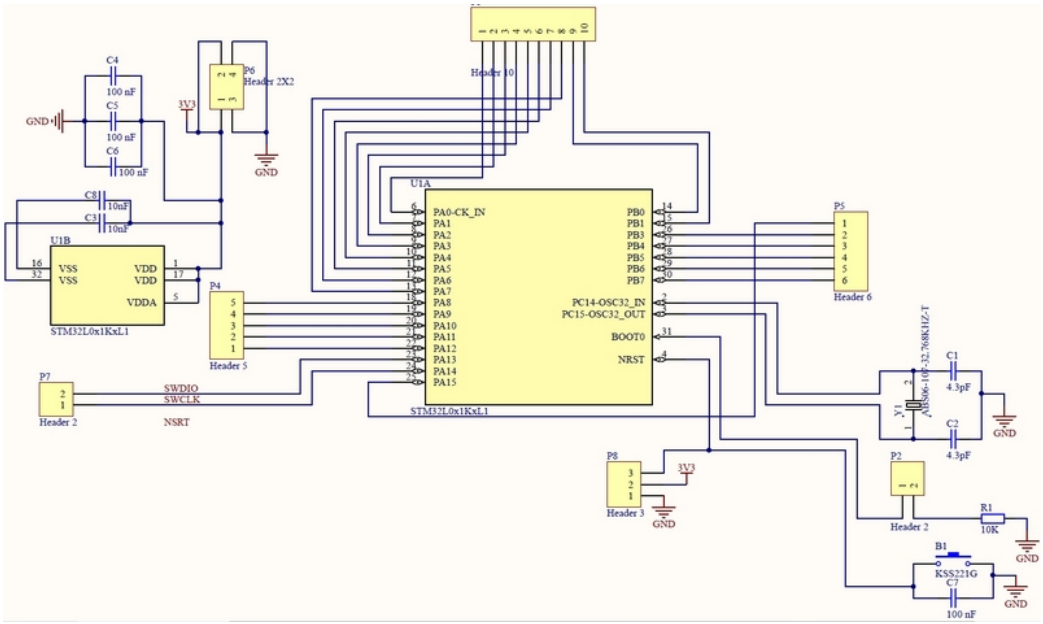

FIGURE 7 – Schéma d'un prototype d'utilisation du STM32L031K6

Ce schéma électronique prend en compte plusieurs considérations, en fonction de la datasheet ou des schémas de la NUCLEO :

— Il n'est pas possible de brancher une horloge HSE (du style 16MHz), il faut donc brancher un cristal 32.768KHz qui servira de clock de référence au multiplicateur d'horloge interne au MCU..
— Un découplage de 100nF au borne des deux pins VDD et du pin VDDA est nécessaire
— Le BOOT_0 est relié à la masse via une résistance 10kOhm (on utilisera ici un header 2*1 afin de changer permettre de changer l'état du BOOT_0 si nécessaire..)
— La programmation du MCU doit se faire en SWD (Serial Wire Debug), car le MCU n'intègre pas les fonctionnalités JTAG. Il faut donc ressortir les pins d'alimentation, le NRST et SWDIO/SWCLK vers des broches afin de pouvoir programmer le composant

En essayant d'obtenir un placement des composants optimisés permettant de limiter la taille des pistes, et en mettant l'horloge et le découplage au plus proche du controleur, j'ai réussi a obtenir une version acceptable du PCB :

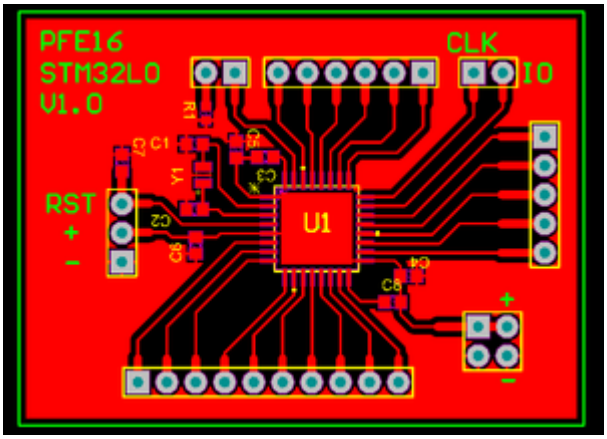

Figure 8 – PCB du STM32L031K6

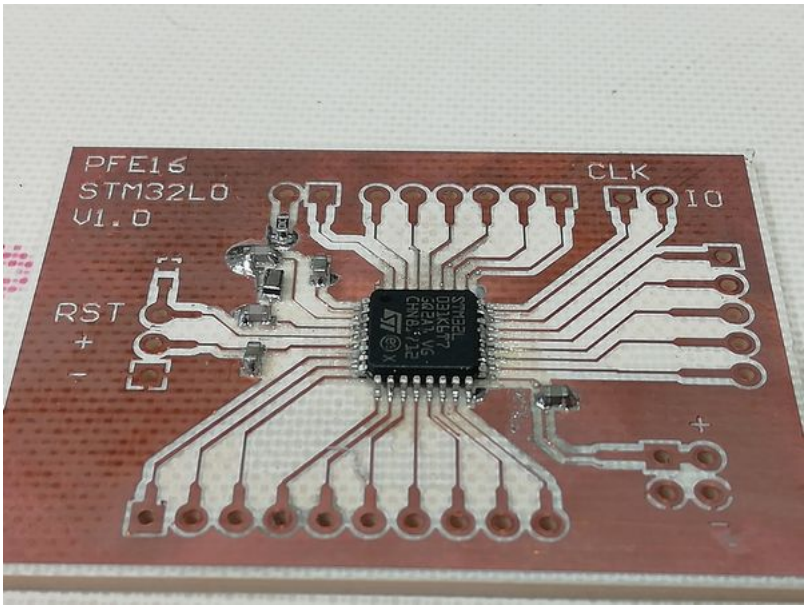

FIGURE 9 – Carte en sortie de four

Malgré des contacts électriques assurés, et les essais d'utilisation de la carte en utilisant un JTAG STLINK-V2 ouun JLINK de chez Segger, je n'ai réussi a détecter ma carte afin de la programmer. Elle a donc été confiée a M. BOE pour qu'il l'expertise.

En attendant je me suis donc concentré sur la partie informatique du projet, notamment sur la programmation des cartes (en utilisants les NUCLEOs) et la mise en place des trames Xbee.

2.4 Définition des trames Xbee

Afin de définir tout le système de communication, il est important de définir clairement les trames qui serviront à l'échange. Dans la mesure ou les échanges entre les module et la base mobile associée, et les bases et le PC se feront tous en Xbee, il est nécessaire de mettre en place un système stable permettant d'éviter les collisions de message.

Ainsi chaque trame se décompose sous la forme suivante :

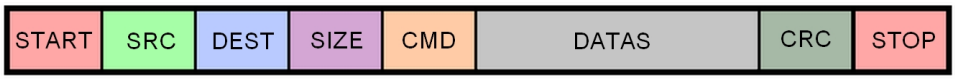

FIGURE 10 – Structure typique d'une trame

Un octet de start et un de stop permettent de délimiter les trames pour vérifier que la chaîne est bien arrivée en entière. Un octet pour l'ID de la source et de la destination du message (Le nombres d'élément par canal Xbee ne devrait pas être aberrant, un seul octet suffit amplement).

La taille de la trame qui permettra aussi de vérifier qu'aucune données n'a été perdu en route. Il s'agit du même usage pour le CRC.

Puis vient la commande indiquant la raison du message, suivie des données l'accompagnant. voici la liste possible des commandes :

0. Envoi d'un ACK
1. Initialisation d'une base mobile
2. Déclaration d'un nouveau module
3. Fin d'initialisation d'une base mobile
4. Rendre une base mobile active
5. Rendre une base mobile inactive
6. Initialisation d'un module
7. Demande de données
8. Réponse à une demande de données
9. Envoi de données au moniteur

Étant donné que l'ordinateur sera branché également sur le réseau Xbee, il ne pourra donc écouter que dans un canal à la fois. Chaque joueur aura son propre canal (soit un maximum de 15 joueurs) afin d'éviter les collisions de données et l'ordinateur n'écoutera donc qu'une seule base à la fois. Afin de limiter la consommation de batterie, une seule base sera donc "active" à la fois, tandis que toutes les autres seront inactives. Tant quelles sont inactives, les bases n'émettent aucun messages, ce qui limite leur consommation énergétique.

2.5 Réalisation de boucliers pour les NUCLEOs

Afin de connecter proprement les cartes NUCLEOS aux modules XBee, il fut nécessaire de créer différents shield, un pour chacun des modèles. L'avantage des NUCLEOS est qu'elles respectent le écartement standard, et que les convention de placement des broches sont les mêmes sur que les Arduinos (Uno pour la NUCLEO L152RE et Nano pour la L031K6). Il est donc assez aisé de créer des bouclierS.

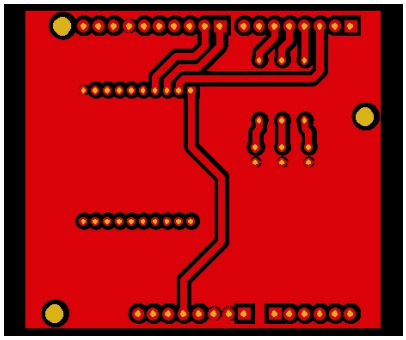

FIGURE 11 – Shield de la L152RE

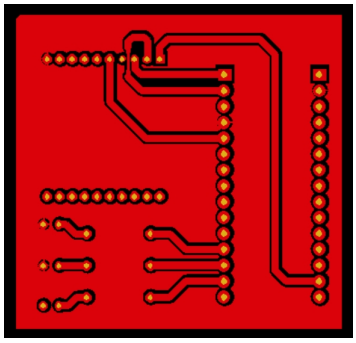

FIGURE 12 – Shield de la L031K6

La connectique minimale nécessaire aux modules Xbee dans un usage UART classique ne se limite qu'à l'alimentation (3V3) du module, ainsi que la connexion des broches TX/RX. Afin de permettre un certain contrôle du module en cas de problème j'ai également relié la broche de Reset à une sortie digitale. Je relie également trois autres sorties digitales à des LEDs afin de proposer un affichage visuel du fonctionnement des cartes

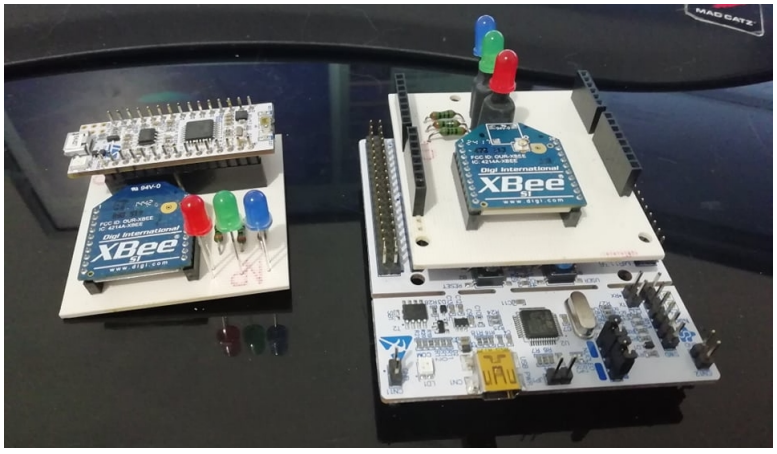

FIGURE 13 – A gauche un module, à droite la base mobile

2.6 Programmation des cartes

La programmation des cartes a été réalisé grâce à Mbed OS, un système d'exploitation open-source destiné à l'IOT, dont le projet est porté par ARM ainsi que ses partenaires techniques (Samsung, ST...). Cet OS consiste en une librairie centrale qui permet l'accès aux drivers, interruption et mise en réseau de ses différents éléments. De plus beaucoup d'extention existe à cette librairie, notamment la librairie XBeeLib, créée et publiée par Digi Internationale, les fabricants des modules que j'utilise.

Bien que je pensais que cette librairie aller me faire gagner beaucoup de temps, le fait qu'elle n'ai pas fonctionné comme prévu m'en a fait plus perdre qu'autre chose, et j'ai donc eu l'obligation de recoder en partie la librairie à ma sauce afin de la faire fonctionner avec mon système. De plus cela à permis d'éviter de tout réécrire, dans la mesure ou la plupart des commandes AT ne me serviront pas, j'ai pu me concentrer sur les commandes suivantes :

— +++ - Entrer en mode AT Command
— ATCH X - Changer le canal actuel vers le canal X
— ATID X - Changer le PanID du périphérique
— ATMY X - Changer l'ID du périphérique
— ATWR - Sauvegarder les changements
— ATCN - Quitter le mode AT Command

Cette liste de commande me suffit à faire fonctionner le système, mais il est tout a fait possible d'en écrire plus, car les fonctions se ressemblent grandement.

Dans un soucis d'économie d'énergie durant l'utilisation des systèmes, la plupart des commandes est géré en tant d'interruption matérielle sur le port série. Ainsi la fonction main de chacune des cartes ressemble à la suivante :

```
10
11 int main() {
12     init();
13     while(1){
14         sleep();
15     }
16 }
17
```

FIGURE 14 – Fonction main() des cartes

De ce fait, les périphériques sont en train de dormir la plupart du temps. Seul la base active fonctionne en quasi-permanence de manière active, car elle prospecte tous ses modules avant d'envoyer les informations au moniteur.

Les fichiers sont divisés suivant l'arborescence suivante :
- main.cpp - Fichier principal incluant la fonction main() du programme
- network.cpp - Défini le déroulement du système. C'est notamment ici qu'on analyse le contenu des trames reçues
- trame.cpp - Permet la vérification des trames entrantes, afin de filtrer les trames erronées
- xbee.cpp - Regroupe les fonctions faisant interraction avec le module (envoi de message ou commande AT)
- led.cpp - Défini les differentes fonctions permettant de gerer les leds
- trameGenerator.cpp - Permet la création de trame qui seront envoyé par le module Xbee

L'ensemble des programmes est disponible sur le gitlab de l'école, avec le lien vers celui-ci dans mon wiki.

2.7 Mode d'emploi du système

Afin de faire fonctionner l'ensemble du système il est important de suivre un protocole de démarrage assez strict, mais qui est accompagné de signaux lumineux sur les différentes cartes. Voici donc le mode d'emploi de l'ensemble du système :

1. Démarrez l'ordinateur qui recevra les données, branchez le périphérique de communication et lancez le logiciel
2. Une fois le logiciel lancé, appuyer sur le bouton [Initialisation] qui démarrera la connexion avec le périphérique de communication.
3. Une fois la communication assurée, démarrez une première base mobile. Attendre que celle-ci clignote bleu avant de procéder à l'étape suivante.
4. Appuyer sur le bouton [Ajouter un joueur] sur le logiciel. Ceci enverra un message d'initialisation à la base. attendre que celle-ci clignote vert avant de procéder à l'étape suivante
5. Démarrez un à un les différents modules que l'on souhaite connecter au joueur. Ces modules doivent clignoter bleu lorsqu'ils sont initialisés, puis vert lorsqu'ils sont détectés par la base mobile du joueur.
6. Une fois que vous avez connecté tous vos modules. Cliquez sur le bouton [Fin préparation] après avoir rentré le nom de votre joueur. Tous les modules et la base inter-connectés obtiendront un code couleur unique permettant de reconnaître le joueur.
7. Vous pouvez alors au choix : commencer à récupérer les données de votre joueur en allant dans l'onglet portant le son nom, ou vous pouvez ajouter un joueur supplémentaire en reprenant à partir de l'étape 3.

Des photos d'aide sont disponibles en annexes de ce rapport.

3 Bilan du projet et ouvertures

3.1 Tâches accomplies et retours sur le cahier des charges

Beaucoup de briques du projet ont été posées, la partie programmation fonctionne quasiment comme voulu au jour ou j'écris ce rapport, et elle devrait être achevée pour la présentation. Beaucoup de recherches sur les differentes technologies ainsi que sur la programmation des STM32 ont pu être réalisées, et j'ai pu apprendre beaucoup sur ces outils.

Bien que le travail a été compliqué de par mon travail en solo sur ce projet, une grande partie du cahier des charges a pu être réalisé :

— L'architecture réseau ainsi que les trames d'échanges sont clairement définies
— Les shields des cartes NUCLEOs sont fonctionnels et les échanges se font correctement
— Il est possible de récupérer les données sur l'ordinateur

Bien que la carte n'a pas fonctionné, la conception d'un premier PCB pour les modules embarqué a été réalisé. Et le module BLE que je dispose est prêt à l'emploi sur Altium. Un packaging simple existe aujourd'hui qui n'est la que pour protéger les cartes et proposer un exemple.

3.2 Travaux à revoir

Une partie majeure du projet consistait en l'utilisation d'une technologie de communication à courte portée, d'abord ANT+ puis BLE, je n'ai pas réussi à travailler comme je le souhaitais avec ces deux réseaux. Le premier à cause d'un défaut de commande et le peu de documentation, le second par manque de temps et une erreur lors de l'achat des modules (Ces modules sont aujourd'hui obsolètes)

Il sera clairement intéressant de connaître la cause de l'échec du PCB embarquant le STM32L0, nous n'avons que peu l'occasion de produire de telle carte en dehors de nos projets de 4 et 5e année, et bien que cela reste un exercice difficile, il n'en reste pas moins intéressant, surtout de par notre formation.

3.3 Pistes d'amélioration

Le prototype actuel est absolument améliorable, ne serait-ce que par l'usage d'une technologie de communication moins énergievore entre la base et les modules (BLE au départ).

Il est également important de se concentrer sur la sécurité du système. J'ai pu mettre à mal le système assez facilement avec mon ordinateur. Il est assez sensible au DDOS et se met "rapidement" en l'air en cas d'attaque. De plus le protocole Xbee en lui même n'est que très peu sécurisé. Il est très facile d'intercepter les messages avec un simple sniffer. Le système Mbed propose une implantation d'un chiffrement SSL/TLS. Il peut être intéressant de voir s'il est possible de l'utiliser dans notre cas.

Une interface graphique plus attrayante pourra également être mise en place afin d'aider l'utilisateur à mieux s'en servir, tout en proposant un peu plus de liberté quant au mode d'emploi du système. En proposant par exemple l'ajout après coup d'un module supplémentaire sur un joueur déjà initialisé.

Mais il est primordial de miniaturiser les systèmes ! Les prototypes sont encore trop gros et risqueraient de perturber le sportif dans son activité. De plus, les modules s'exposent à des dommages lors de contacts trop violents (Rugby) mais pourraient également blesser les joueurs.

Conclusion

Un prototype quasi fonctionnel est disponible à la fin de ce projet. Il est certain qu'il aurait été fortement intéressant de proposer des modèles réduits si le PCB embarquant le Cortex M0 avait fonctionné. Il reste beaucoup d'amélioration au projet, mais une grande partie du travail a été réalisé.

Il est toujours très compliqué de s'atteler à un projet aussi grand, reliant de l'informatique embarqué et de la conception électronique alors que l'on est seul dessus. Mais c'est un défi que j'ai voulu relevé, car comme pour mon projet de 4e année, cela a été grandement formateur.

J'aurai à nouveau la possibilité de travailler sur ces micro-contrôleurs lors de mon stage chez BLUEGRiot, un bureau d'étude spécialisé dans les objets connectés. J'espère ainsi pouvoir valoriser mes compétences acquises durant mes différents projet et ma formation au sein de Polytech Lille