hepia

Smartbag

Système de gestion d'affaires

Adrien Taboada

06/03/2017

TABLE DES MATIERES

INTRODUCTION

Pour commencer, ce projet n'est pas uniquement un projet de semestre, il continuera en projet de Bachelor. De ce fait, on peut dire que le projet de semestre est un travail de recherche sur la faisabilité du projet. Pour cela il a fallu faire beaucoup de recherches.

La particularité de ce projet, est que nous travaillons en binôme avec mon collègue Axel Collet. Et bien sûr que ce projet est un projet de Bachelor.

PROJET CHIC

Le CHIC (China Hardware Innovation Camp) est un projet organisé par l'EPFL et Swissnex China. Swissnex est une fondation de la confédération helvétique pour promouvoir l'ingénierie suisse dans le monde, dans notre cas la Chine. Le projet CHIC consiste à développer en 12 semaines un objet connecté et innovateur de préférence.

Nous sommes organisés en différentes équipe selon notre localisation. Chaque équipe est composée d'étudiants de différentes orientations (business, design, ingénierie). Tout a commencé en novembre 2016 au cours d'un week-end d'idéation.

LA TEAM GENÈVE

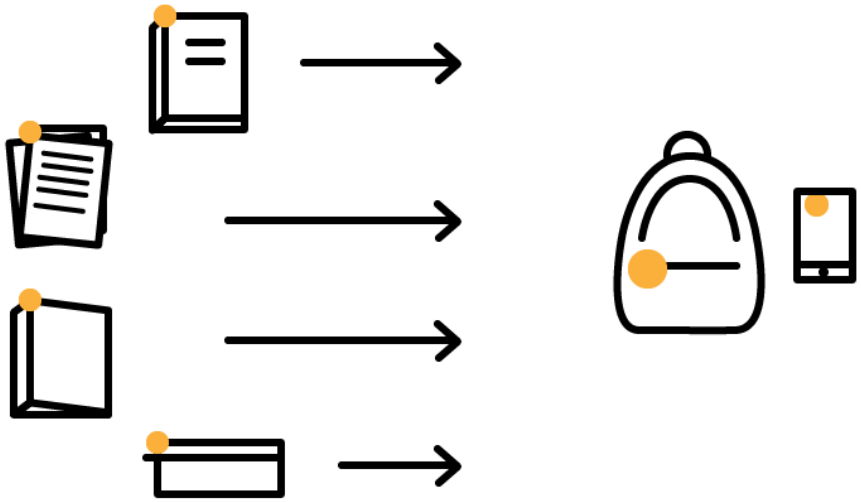
Dans la team Genève nous sommes cinq étudiants : Deux en ingénierie des technologies de l'information en orientation matérielle à l'HEPIA, une en international business management à l'HEG et deux en design à la HEAD.

Mon collègue Axel et moi sommes les ingénieurs qui s'occuperons de la partie technique du projet, Tabea s'occupera de toute la partie économique, Julia de l'interaction avec l'utilisateur et Loïc du design et des parties mécaniques.

GÉNÉRALITÉS

Le concept que nous avons tous choisi est celui du smartbag. Le problème que nous avons relevé, c'est l'oubli. Tout le monde a déjà connu la situation où il leurs manquait une affaire importante ! Avec ce projet nous allons nous attaquer à cette problématique en créant un objet qui aidera tout le monde.

Cela se composera d'une centrale autonome qui se trouvera dans un sac et de tags à positionner sur nos affaires. Le tout sera connecté avec un smartphone qui permettra de gérer toutes les affaires et surtout de programmer notre agenda pour que le sac vérifie automatiquement, au départ par exemple, que toutes nos affaires sont présentes dans le sac pour la journée selon l'agenda.



Ce sera comme une gestion d'inventaire mais couplé avec un agenda. Ce qui veut dire que l'utilisateur devra tagguer toutes les affaires qu'il veut suivre et qu'ensuite il insère dans l'agenda son emploi du temps et quelles affaires il aura besoin pour tel jour.

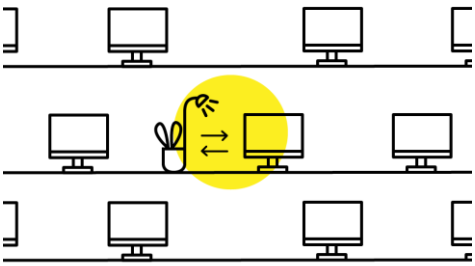
Ce qui veut dire que chaque affaire devra avoir un tag RFID sur elle. Nous allons utiliser des tags passifs, ce qui veut dire qu'il n'y a pas de batterie intégrée et de ce fait un tag sera compact, fin et pas cher.

HISTORIQUE DES IDÉES

Avant d'avoir notre idée, nous sommes passés par plusieurs autres idées. Ces trois idées ont été soumises à nos coordinateurs et entre nous tous, les coordinateurs et l'équipe, nous avons tous voté pour le projet qui nous intéressait le plus, notre smartbag. Mais voici les autres projets :

SMART GARDEN

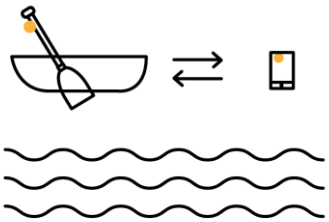
Ceci était notre première idée. Nous avions commencé à penser à faire un jardin connecté, mais ce produit existant déjà, nous l'avons modifié.



L'itération suivante était d'avoir un pot connecté équipé de différents capteurs, dont un qui mesurait la qualité de l'air. Le but était d'avertir l'utilisateur quand son air n'avait plus assez d'oxygène.

Le problème de cette idée était que nous les ingénieurs n'avions vraiment pas de challenge.

PADDLE



Une autre idée venant cette fois ci d'Axel était de rendre un paddle connecté. La cible principale étant les clubs de paddle pour pouvoir avoir un suivi continu de leurs clients quand ils sont en sortie. Sachant qu'il y a un dispositif se fixant à la cheville pour éviter de perdre son paddle en tombant, c'était notre cible : un bracelet de jambe connecté.

Ses fonctions étaient de localiser le sportif, lui envoyer une alerte s'il sortait d'une zone prédéfinie, de recevoir des alertes météo en temps réel pour lui dire de rentrer et possiblement de faire une fonction SOS.

Le problème de cette idée étant que ce n'était pas novateur et que les designers ne se retrouvaient pas dedans.

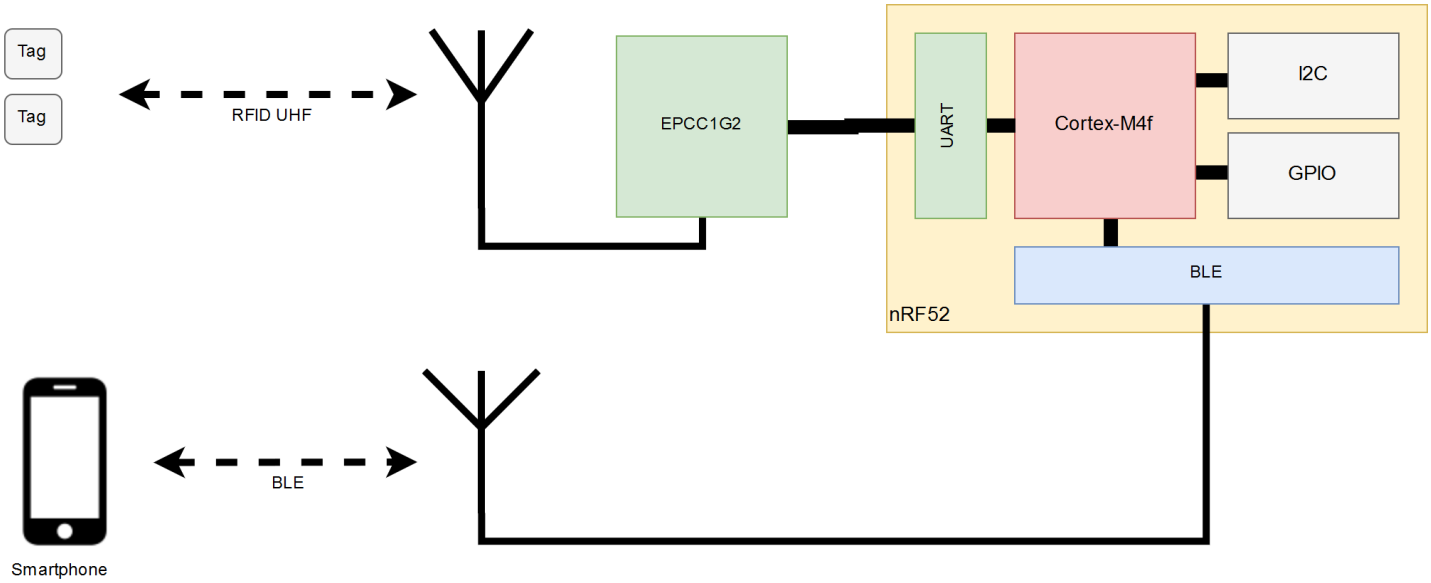
TRAQUEUR DE SKI

Cette dernière idée étant de moi, cela aurait été de créer un traqueur de ski pour pouvoir enregistrer toute une journée de ski. Ce projet aurait impliqué beaucoup de traitement de signal.

Par contre ce n'était vraiment pas une idée innovante, il y a déjà plusieurs traqueurs pour le ski existant, sur ce constat nous avons abandonné cette idée.

ARCHITECTURE

Ce projet comporte une centrale qui contient toute la partie hardware que nous allons concevoir. Cette centrale sera composée par un microcontrôleur avec un périphérique Bluetooth intégré, d'un périphérique RFID et de toute la partie alimentation.



Sachant que la partie BLE est un périphérique intégré au SOC, nous lui communiquerons via des registres. Par contre, vu que le périphérique RFID est extérieur au SOC, il faudra communiquer avec lui à travers le bus UART.

Ce schéma bloc n'est bien sûr pas définitif, car il manque plusieurs éléments pour le moment : l'accéléromètre et l'alimentation. Mais pour le moment nous n'avons pas approfondi ces parties, donc dans un souci de simplification, elles n'apparaissent pas encore ici.

MICROCONTROLLEUR

La base de ce projet est un microcontrôleur car la consommation de l'ensemble est un critère important. Nous avons besoin d'une unité centrale qui contrôlera l'ensemble. Ce qui laisse le choix entre un microcontrôleur ou un FPGA. Mais la consommation devant être la plus faible possible, le FPGA est directement hors-jeu.

COMPARATIF

Nous avons comparé différents microcontrôleurs pour trouver celui qui nous siéra le mieux. Nous nous sommes basé sur plusieurs critères pour le sélectionner : Pour commencer nous avons recherché un microcontrôleur qui possède le BLE et le RFID directement intégré. Il n'y avait jamais un assemblage des deux technologies sur une seule puce.

Sur ce constat nous nous sommes tourné sur les SOC qui possèdent soit l'un soi l'autre. Par contre, autant ce n'est pas un problème pour le Bluetooth, autant c'en est un pour le RFID. Les puces permettant de faire du RFID étaient compatible uniquement en ~13MHz, ce qui est problématique vu que nous avons besoin d'une portée minimale. Donc nous avons recherché uniquement les microcontrôleurs possédant une partie Bluetooth.

Nous avons commencé à chercher chez NXP car nous avions déjà de l'expérience dessus. Chez NXP nous avons trouvé le QN9020 qui est assez gourmand pour le service qui rend. Et aussi le KW31Z qui lui est tout de suite plus intéressant côté consommation, mais il n'était pas encore entré en phase de production.

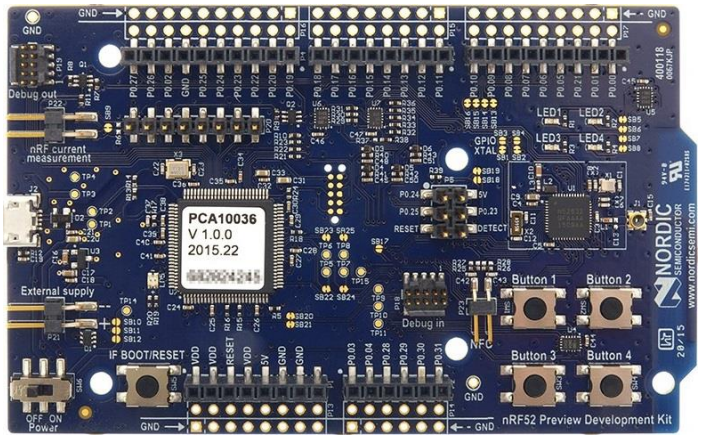
Ensuite nous sommes allés voir chez Texas Instrument qui possède le CC2540 avec Bluetooth intégrée, mais il n'est pas basé sur un cœur ARM mais sur un 8051. Ce qui implique des limites du côté de la mémoire et surtout qu'il ne fait pas le poids comparé aux solutions basée sur ARM.

Pour finir nous avons trouvé un autre fabricant : Nordic Semi qui est spécialisé dans des solutions tout intégré pour le sans-fil. Ils proposent le nRF51 et le nRF52 qui sont deux SOC avec le Bluetooth intégré. Ils sont les deux de la même famille mais ne sont pas de la même génération. Les améliorations du nRF52 sont qu'il possède un cœur Cortex-M4f à la place du Cortex-M0 et il y a surtout la finesse de gravure qui change, elle est bien plus fine. Ce qui implique une baisse de consommation drastique.

UContrôleur	Processeur	Conso. 0dB	Conso. veille	Commentaires
nRF51	Cortex-M0	8mA	2.6uA	Prédécesseur nRF52
nRF52	Cortex-M4f	5.3mA	2.7uA	Meilleur sur la consommation
ti CC2540	8051 (8bits)	27mA	235uA	Consommation trop élevée
NXP QN9020	Cortex-M0	8.8mA	3uA	Rapport puissance/conso. faible
NXP KW31Z	Cortex-M0+	6.1mA	N/C	Début de production

Sur ce constat, nous avons choisi d'utiliser le nRF52 de Nordic Semi.

NRF52



Ce microcontrôleur est basé sur un cœur Cortex-M4f et possède tous les périphériques utiles sur un microcontrôleur (I2C, UART, SPI, etc…). Il possède en plus un contrôleur 2.4GHz qui supporte le Bluetooth Smart, ANT et de la radio propriétaire. Il y a aussi un contrôleur NFC.

La partie Bluetooth est compatible Bluetooth 4.2 et Nordic met à disposition une pile Bluetooth Low Energy nommée SoftDevice.

Nous avons commandé le kit de développement chez Mouser. Chaque kit vient avec cinq puces nRF52 sur une bande et une antenne NFC.

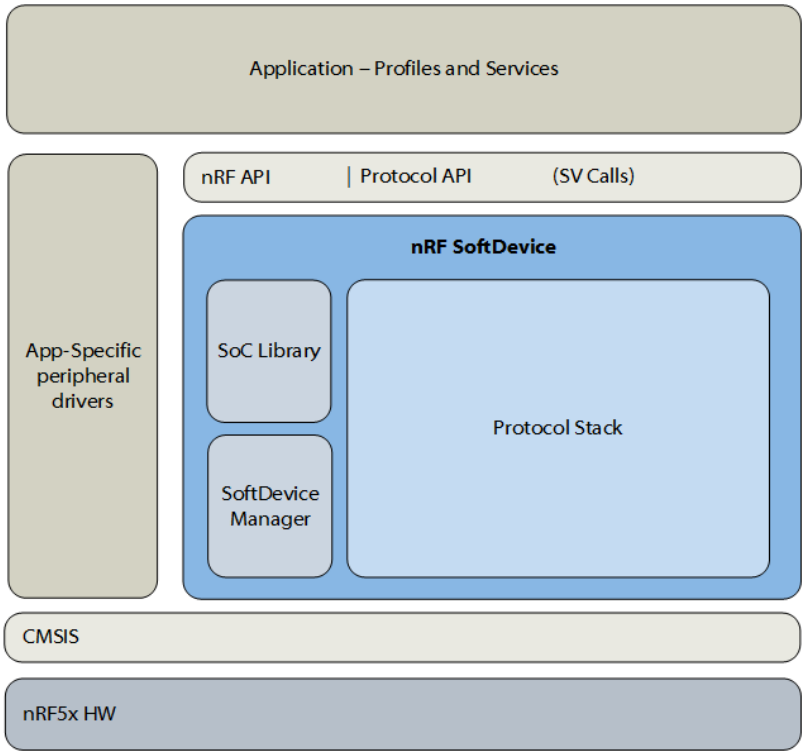
Les interfaces qui seront utilisé seront la partie radio pour le Bluetooth et l'UART pour le RFID. L'ajout d'un accéléromètre est possiblement une prochaine étape, ce qui rajoutera surement une interface I2C ou SPI.

SOFTDEVICE S132

Cette pile Bluetooth est la dernière version en date pour les nRF. Comme son nom l'indique, cette pile est vue comme un périphérique supplémentaire sur le microcontrôleur, malgré que ce ne soit pas du matériel.

L'utilisation de cette pile est conseillée pour que l'application soit indépendante des évolutions du Bluetooth. Ce qui veut dire qu'il est possible d'ajouter de mettre à jour le SoftDevice sans changer l'application.

Nous allons utiliser cette pile dans son rôle de Peripheral pour que notre produit soit vu comme un esclave par les smartphones.

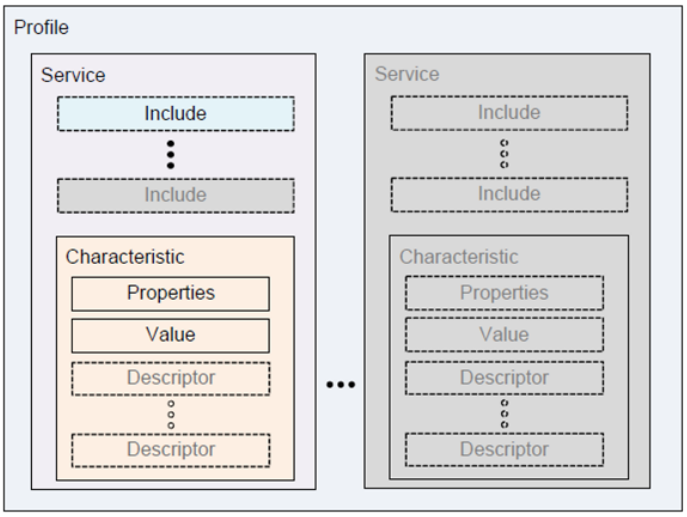


BLUETOOTH LOW ENERGY (BLE)

Le smartbag aura une connectivité *Bluetooth Low Energy* pour pouvoir communiquer avec un smartphone. De ce fait, nous allons utiliser le périphérique Radio 2.4GHz intégré dans le nRF52 avec la pile Bluetooth *SoftDevice S132* de *Nordic Semi*. Ce *SoftDevice* est une pile complète supportant le *Bluetooth 4.2* avec plusieurs rôles *BLE* intégré.

GENERIC ATTRIBUTE PROFILE (GATT)

Le GATT est une structure de données utilisé pour le BLE qui définit les messages que les deux périphériques Bluetooth vont s'envoyer.



On peut voir que cette structure complète représente un profil. Et que chaque profil propose des services. Et chaque service possède ses propres caractéristiques.

Un périphérique Bluetooth à l'obligation de s'annoncer, et dans cette annonce il y a l'ID de son GATT mais pas le contenu du profil. Ce qui implique que pour communiquer avec d'autres périphériques, ils doivent avoir aussi le bon profil.

De ce fait, nous allons créer un profil spécifique pour ce projet, ce qui impliquera qu'il faudra implémenter le GATT dans le maitre et l'esclave.

PROFIL BLE

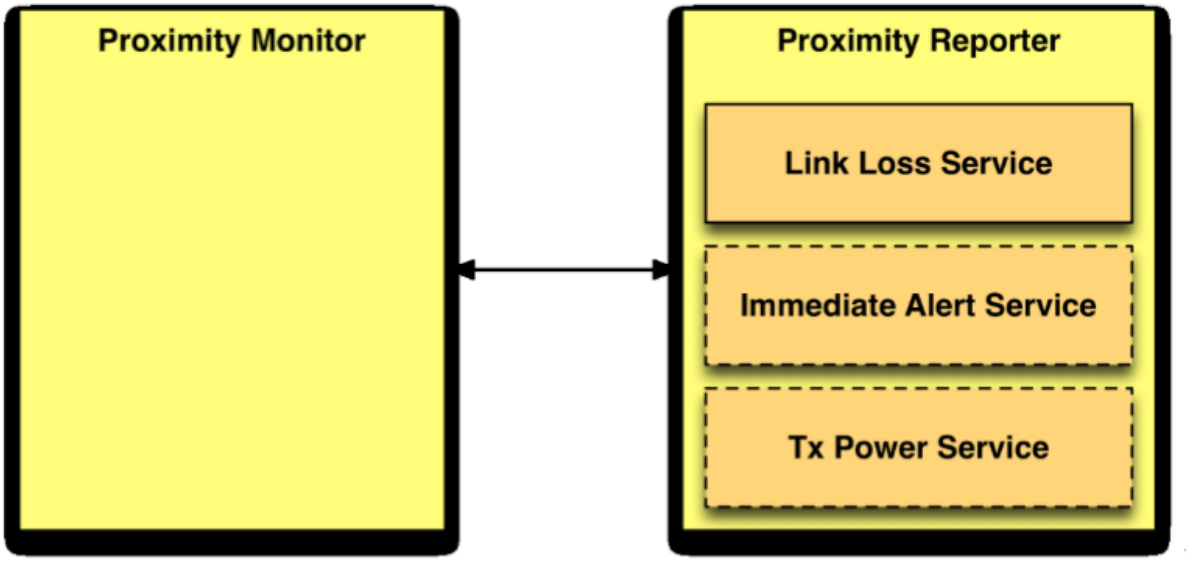
Le BLE possède plusieurs profils adoptés pour différentes utilisations. Il n'y a malheureusement pas de profil qui prends en charge tout selon dont nous avons besoin.

Attardons-nous sur une particularité du BLE, les profils adoptés existent uniquement pour avoir un standard pour que différents produits qui effectuent les mêmes taches puissent communiquer ensemble. Mais si nous ne trouvons pas le profil qui colle à nos besoins, nous sommes libres de créer notre propre profil et c'est ce que nous allons faire là.

Par contre nous n'allons pas créer un profil complet mais plutôt en modifier un existant pour lui donner les fonctions qui lui manque. Le profil de base que nous allons utiliser est le Proximity (PXP).

PROXIMITY (PXP)

Le profil Proximity est un profil qui permet d'alerter un maitre si l'esclave ne se trouve plus dans sa zone d'émission. La base de ce profil nous sera utile si on implémente une fonction qui alertera l'utilisateur si son sac se retrouve loin de lui.



Ce profil possède deux rôles : Le proximity Monitor et le Proximity Reporter. Il y a deux rôles qui ont chacun une cible différente.

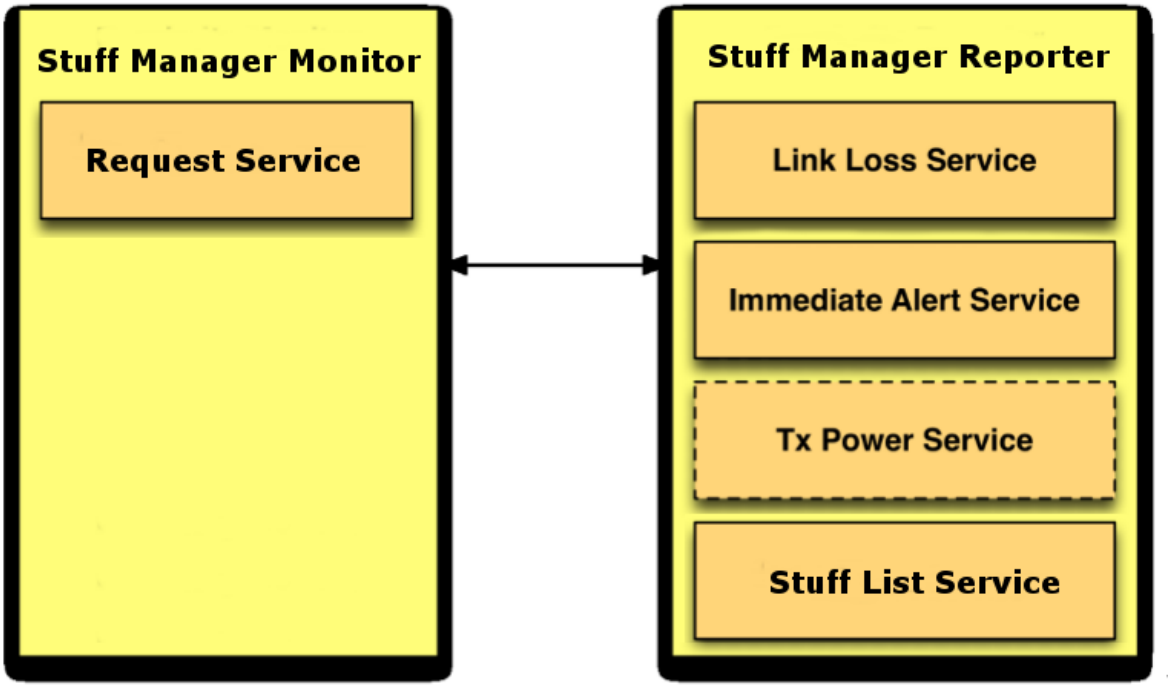
Le Proximity Monitor est dédié au maitre. Et comme on peut le voir ci-dessus, il n'y a aucun service du côté Bluetooth. La seule fonction du Monitor étant de prévenir quand il y a une perte de connexion avec le Reporter. Ce qui veut dire qu'après l'appairage il n'y a qu'une communication unidirectionnelle entre les périphériques.

Et le Proximity Reporter est dédié à l'esclave. Son rôle principal est d'envoyer des paquets à intervalles régulières au Monitor pour lui annoncer qu'il est toujours présent. Il possède un service obligatoire : le Link Loss Service. Les autres services étant optionnels.

Un petit résumé des services proposé. Le Link Loss est celui qui permet d'annoncer sa présence au Monitor. L'Immediate Alert est un service qui permet d'envoyer une alerte (possibilité d'avoir des alertes différentes). Et pour finir le Tx Power Service qui permet d'envoyer la consommation du Reporter pendant la transmission.

NOTRE PROFIL

Notre profil va rajouter plusieurs services. Pour le moment son petit nom est le service Stuff Manager (STF). Voici son architecture (pas définitive) :



Les changements par-apport au profil Proximity son que le Monitor puisse envoyer des commandes au Reporter. Pour par exemple lui demander de faire un scan des objets aux alentours.

Par contre le plus important c'est l'ajout d'un service spécifique pour communiquer la liste des affaires détectés par la base, le Stuff List Service. Il sera obligatoire car c'est la base du profil.

Il y a tous les services de base du PXP, mais avec un petit changement : l'Immediate Alert sera obligatoire car la base doit envoyer une alerte au smartphone s'il manque un objet pour la journée.

RFID

Ce projet comporte aussi une grande partie qui est le RFID. La base communiquera avec les tags apposés sur les affaires que nous souhaitons monitorer.

Ces tags sont de type passif sans batterie comme on peut en trouver dans les forfaits de ski. Ce sera un challenge énergétique car si les tags ne possèdent pas de source d'énergie, c'est la base qui doit la leur fournir ! Je prévois que cette partie fera partie des plus grands consommateurs.

Cette partie du travail c'est mon collègue Axel Collet qui s'en occupe. Pour plus de précisions regardez son rapport.

ALIMENTATION

La base devra être alimentée par une source d'énergie. De ce fait, il faut lui adjoindre une batterie ou des piles.

Au début nous avions pensé à une alimentation avec des piles (alcaline, lithium) au vu de la caractéristique de ne pas devoir à changer les piles trop souvent. Mais tout a été supplanté par une batterie lithium. La batterie possède quasiment que des avantages comparés aux piles :

Piles	**Batteries lithium**
+ Standard (facilité d'acquisition)	+ Meilleurs encombrement/capacité
- Non rechargeable	+ Toute forme possible
- Capacité limitée	+ Rechargeable
- Forme imposée	- Difficulté à remplacer

De ce fait nous allons utiliser une batterie lithium. Nous n'avons pas décider de quelle technologie nous allons utiliser entre les *li-ion* et les *li-po*. Il y a encore un choix à faire sur la batterie, sera-t-elle remplaçable par l'utilisateur ou intégrée dans la base ?

L'autonomie visée pour la base est de 2 semaines au minimum. Devoir recharger un objet en plus dans sa vie quotidienne est plus qu'embêtant, de ce fait avoir l'autonomie la plus longue possible dans un format compact sera un critère déterminant dans ce projet.

Pour recharger cette batterie, nous allons utiliser un connecteur standard qui sera compatible avec tous les chargeurs USB que l'on peut retrouver chez nous. De ce fait, nous allons utiliser un connecteur USB type-C. Nous n'avons pas choisi le micro-USB car nous sommes dans une phase de transition entre les deux connecteurs, en faveur du nouveau type-C.

Micro-USB	**USB type-C**
+ Démocratisé	+ Réversible
+ Bon marché	+ Transition vers le type-C
- Transition vers le type-C	- Non démocratisé
	- Prix conséquent encore

PROBLÈMES RENCONTRÉS

ENVIRONNEMENT DE TRAVAIL

Les premiers problèmes que nous avons eus étaient la mise en place de l'environnement de travail. Le nRF52 possède bien évidement un SDK et tout un tas d'utilitaire pour pouvoir faire notre application sur le microcontrôleur.

Leurs outils peuvent se séparer en deux branches : la branche Keil et la branche GCC.

Nous avions commencé par utiliser GCC car, contrairement à Keil, il est open-source. En plus de ça Nordic propose un tutoriel pour pouvoir mettre en place un environnement Eclipse-GCC. Et il y a aussi pleins d'exemples de projet pour commencer. Toute la chaine de compilation était gérée par des makefile assez complexes.

C'est de la que vient les premiers problèmes, pour je ne sais quelle raison, je n'ai jamais réussi à faire fonctionner Eclipse correctement avec le SDK. Ce qui fait qu'il ne pouvait pas y avoir la génération automatique du makefile, sachant que leurs complexités pour les nRF était haute.

De ce fait nous avons testé la branche Keil avec uVision qui lui est un IDE complet de programmation pour tout type de processeurs ARM. Son défaut est qu'il est propriétaire, mais tout est simplifié pour la compilation et l'upload de programmes.

CAPTURE DE PAQUETS BLE

Un autre problème rencontré était de pouvoir capturer les bons paquets Bluetooth pour les analyser. Mais il y a beaucoup de périphériques Bluetooth de toute sorte qui communiquent en continu. De ce fait il a fallu filtrer les paquets selon leurs addresses MAC. Mais il y a encore un autre problème, le smartphone Android (en tout cas le miens) envois des paquets tout le temps !

CONCLUSION

Pour conclure ce travail, il est parfaitement possible de créer ce SmartBag. Toutes les technologies existent déjà, il faut juste les assembler ensemble et créer un environnement complet.

La plus grande problématique sera, bien sûr, l'autonomie. Mais pour avoir une idée de l'autonomie, le seul moyen est d'effectuer des tests de consommations. Ensuite nous pourrons penser à la capacité de la batterie qui lui sera adjointe. Mais là encore il faut faire des compromis, car la base ne devra pas être trop lourde ou trop imposante.

Par contre, ce qui me dérange, c'est que je n'ai pas de code à montrer. Je n'ai rien de fonctionnel et correct à montrer. Etant habitué à rendre toujours un programme avec chaque travail, ici c'est singulièrement différent. La faute au fait de la nature du projet, nous n'avons sélectionné le projet uniquement à partir de Décembre 2016. Ce qui fait que pour le moment nous n'avons eu que trois mois pour effectuer nos recherches et commencer à programmer.

Il est très intéressant de travailler avec des personnes venant d'horizon différent. On peut voir les divergences de point de vue, comme par exemple les priorités entre les ingénieurs et les designers ne sont clairement pas les mêmes.

SUITE DU TRAVAIL

Les étapes suivantes à effectuer sont les suivantes :

- Continuation et finition de la partie Bluetooth
- Continuation et finition de la partie RFID
- Mesure de la consommation
- Calcul de l'autonomie
- PCB custom
- Intégration du tout dans une base (travail avec les designers)

hepia

Smartbag - annexes

Système de gestion d'affaires

Adrien Taboada

06/03/2017

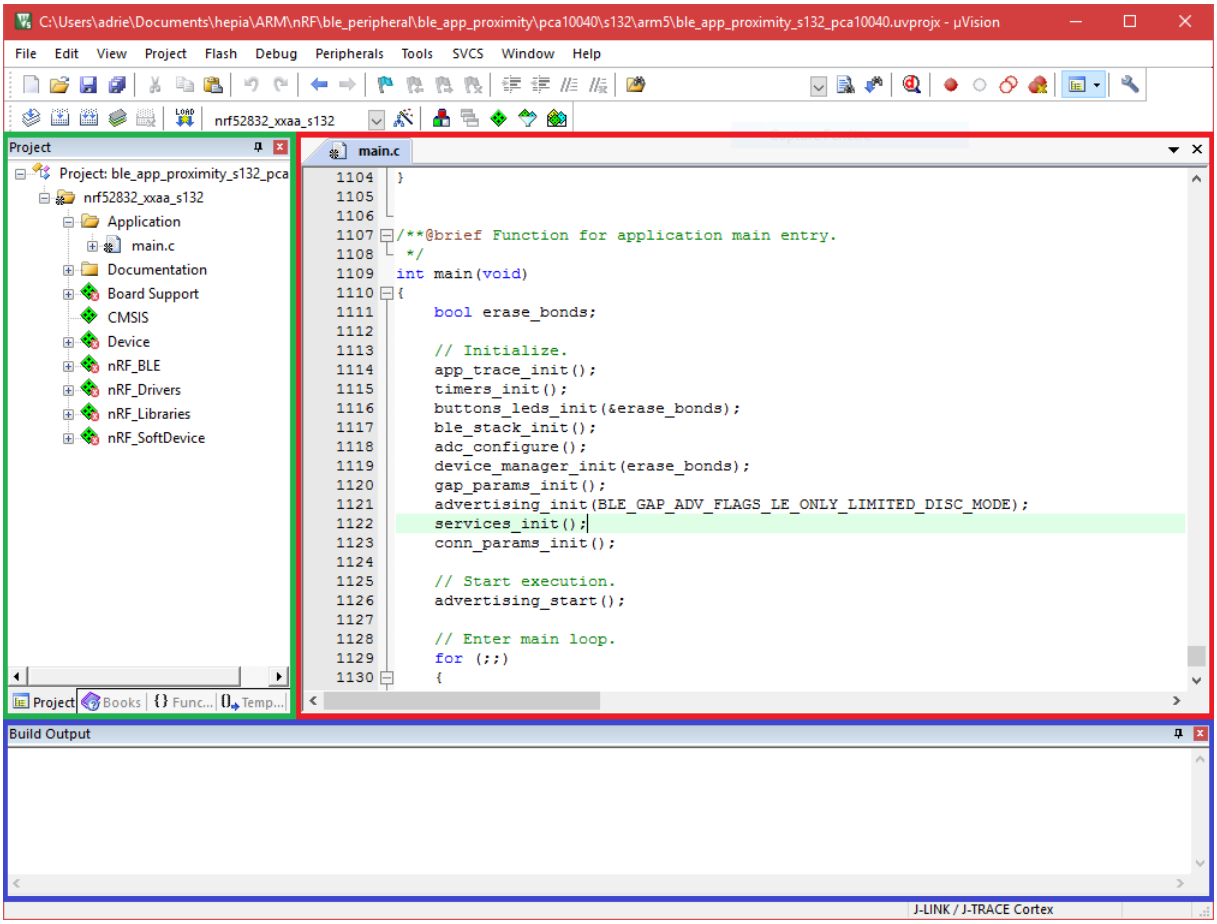
TABLE DES MATIERES

ENVIRONNEMENT DE DÉVELOPPEMENT

La carte étant compatible J-Link. Elle est vue comme un périphérique de stockage de masse si aucun driver spécifique n'est installé sur l'ordinateur. Cela permet de programmer la carte avec un fichier binaire sans avoir besoin de programme spécifique. Mais pour une plus grande simplicité, nous allons utiliser les programmes suivants :

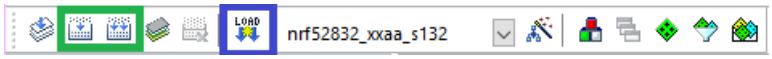
KEIL UVISION 5

L'environnement de développement utilisé pour développer sur la carte est uVision 5 de Keil. C'est un IDE uniquement sur Windows qui prend en charge différents processeur ARM.

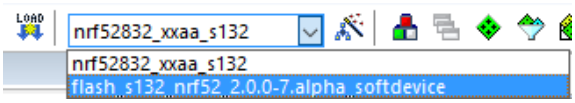


L'environnement se présente avec trois fenêtres différentes : **L'édition du code**, **l'arborescence du projet** et la **console**.

COMPILATION ET UPLOAD DU PROGRAMME



L'étape suivante est de compiler le programme. Pour le compiler il faut utiliser le **bouton Build**. Ensuite il faut uploader le programme sur la carte de développement, cela s'effectue avec le **bouton Download**.



Par contre il y a une étape supplémentaire si on utilise le SoftDevice. Si le Bluetooth doit être utiliser, il faut qu'il y ait un SoftDevice sur la carte. Et pour cela il faut le flasher avant de flasher le programme. Pour effectuer cette opération, il faut sélectionner le SoftDevice dans la combobox et utiliser le **bouton Download**. Et ensuite on peut uploader notre programme.

SI LE MESSAGE SUIVANT APPARAIT :

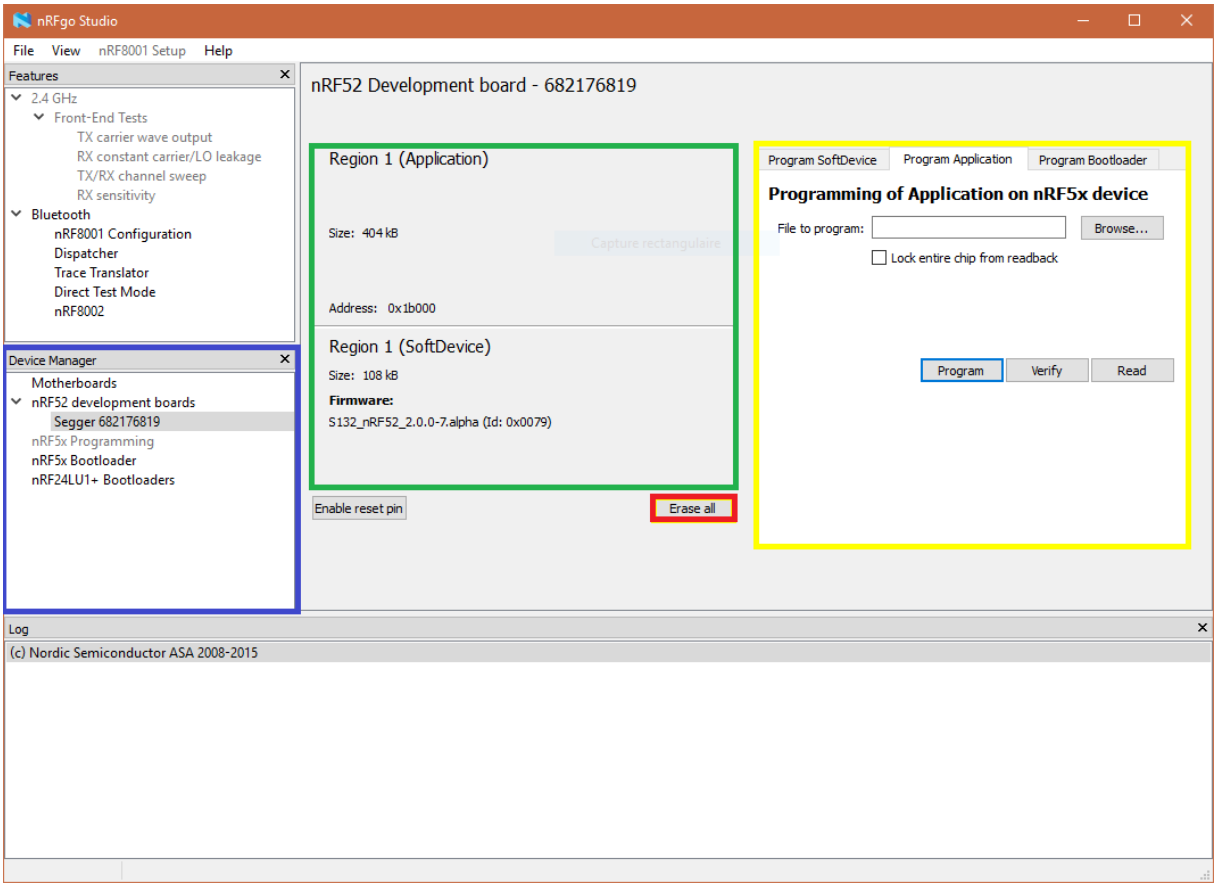
ERROR: FLASH DOWNLOAD FAILED - "CORTEX-M4"

CELA VEUT DIRE QU'IL Y A UN SOFTDEVICE SUR LA CARTE. IL FAUT DONC EFFACER LA FLASH DE LA CARTE AVEC L'UTILITAIRE NRFGOSTUDIO.

NRFGOSTUDIO

Le microcontrôleur possède un utilitaire qui permet d'effectuer plusieurs actions sur la carte. Ce programme étant pour plusieurs produits de Nordic, nous n'utiliserons de loin pas toutes les fonctions.

La seule partie du programme qui nous intéresse est celle intitulé **Device Manager**.



Dans cette fenêtre, on peut voir une **représentation de la mémoire flash de la carte**. Sur l'exemple ci-dessus, un programme utilisant la pile Bluetooth est chargé, donc il y a le SoftDevice et l'application.

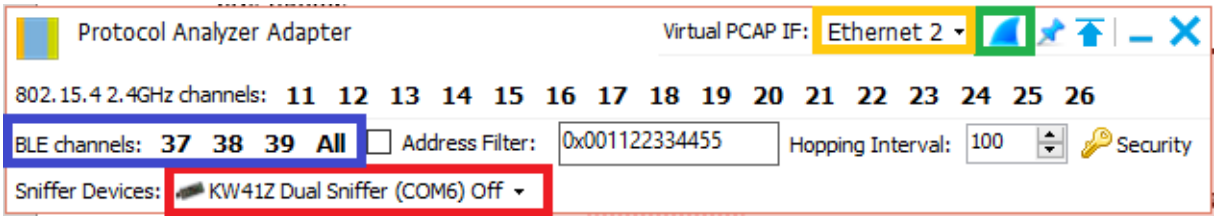
Il est possible d'effacer complètement la mémoire de la carte, ce qui est obligatoire pour supprimer le SoftDevice. Car il est impossible d'uploader un programme ne l'utilisant pas s'il est présent. Pour cela il suffit juste d'appuyer sur **Erase all**.

Et pour finir il est possible de programmer la carte directement **ici** avec un programme précompilé, un SoftDevice ou encore un BootLoader.

KINETIS PROTOCOL ANALYZER

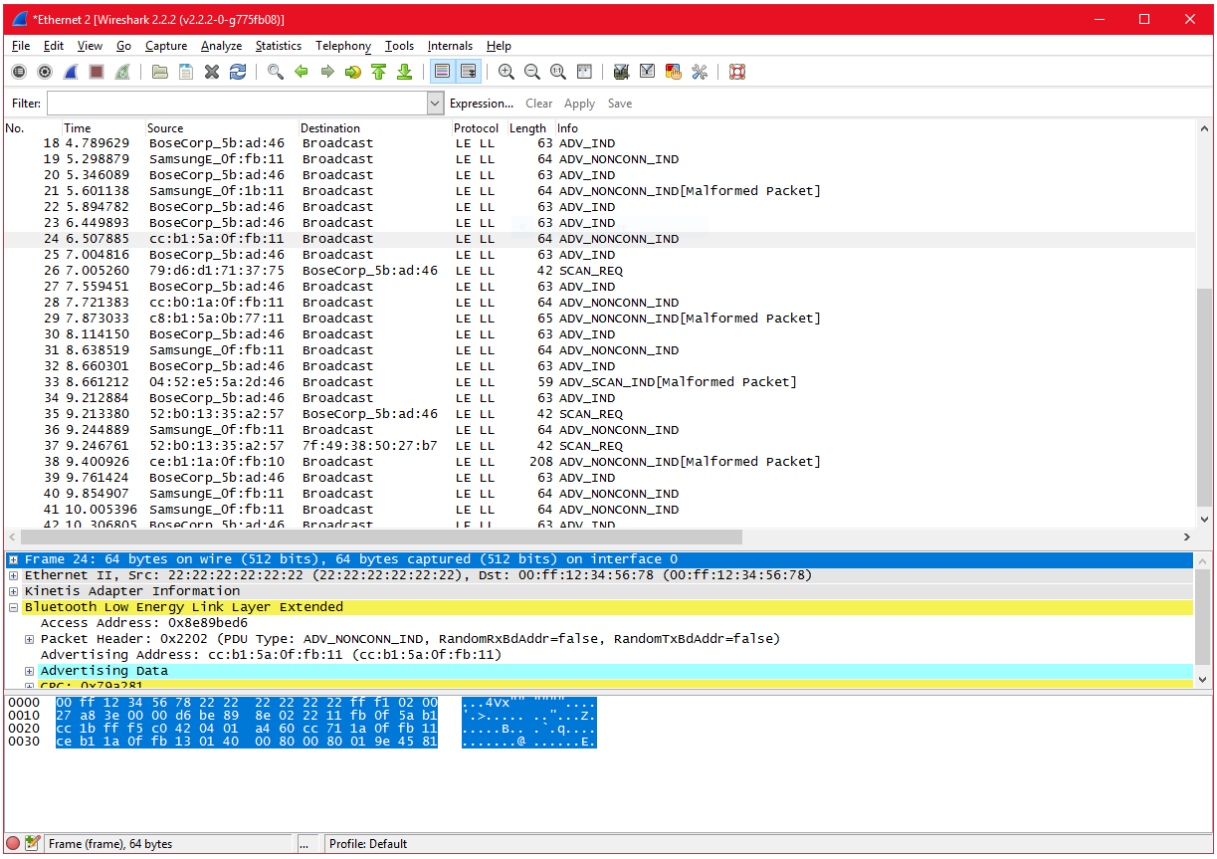
Un autre matériel très utile, c'est une clé Bluetooth USB associé à Wireshark. Ça permet de capturer les paquets Bluetooth pour pouvoir les analyser.

Il faut tout d'abord lancer le Kinetis protocol analyzer adapté qui se présente comme ça :



Ce programme doit détecter tout d'abord la **clé associée**. Ensuite il faut sélectionner les canaux que l'on veut capturer. On peut capturer les canaux 802.15.4 ou plus précisément ceux spécifique au BLE. Pour ça il faut sélectionner les **BLE channels**. Quand tout est configuré, on peut lancer la capture Wireshark avec le **bouton Wireshark**.

Ensuite c'est un Wireshark tout a fait standard qui se lance. Et pour lancer la capture il faut sélectionner la bonne carte réseau (Ici **Ethernet 2**).



Ce Wireshark est composé de trois fenêtres principales. Il y a tout d'abord la liste des paquets capturés. Ensuite, en sélectionnant un paquet nous pouvons voir les détails du paquet, en décomposé ou en brut.

ABRÉVIATIONS

BLE	Bluetooth Low Energy
RFID	Radio Frequency Identification
NFC	Near Field Communication
UART	Universal Asynchronous Receiver Transmitter
CHIC	China Hardware Innovation Camp

SOURCES

Ce document possède des images venant de sources différentes.

Le schéma du SoftDevice vient de la spécification SoftDevice S132.

Le schéma du GATT et du profil PXP viennent du Bluetooth SIG.

BLUETOOTH

Profils BLE adopté

Bluetooth GATT

NRF52

Spécification nRF52

Spécification SoftDevice S132