|  |
| --- |
| hepia |
| Smartbag |
| Système de gestion d’affaires |

|  |
| --- |
| Adrien Taboada  06/03/2017 |

Table des matières

[Projet CHIC 2](#_Toc477255125)

[La team 2](#_Toc477255126)

[Généralités 2](#_Toc477255127)

[Composants 2](#_Toc477255128)

[Architecture 2](#_Toc477255129)

[Microcontrolleur 2](#_Toc477255130)

[Comparatif 2](#_Toc477255131)

[nRF52 2](#_Toc477255132)

[Environnement de développement 2](#_Toc477255133)

[Keil uVision 5 2](#_Toc477255134)

[nRFGoStudio 2](#_Toc477255135)

[Kinetis Protocol Analyzer 2](#_Toc477255136)

[Bluetooth Low Energy (BLE) 3](#_Toc477255137)

[Generic Attribute Profile (GATT) 3](#_Toc477255138)

[Profil BlE 3](#_Toc477255139)

[Proximity (PXP) 3](#_Toc477255140)

[Notre profil 4](#_Toc477255141)

[SoftDevice S132 4](#_Toc477255142)

[RFID 5](#_Toc477255143)

[Alimentation 5](#_Toc477255144)

[Sources 6](#_Toc477255145)

[Bluetooth 6](#_Toc477255146)

Smartbag

# 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.

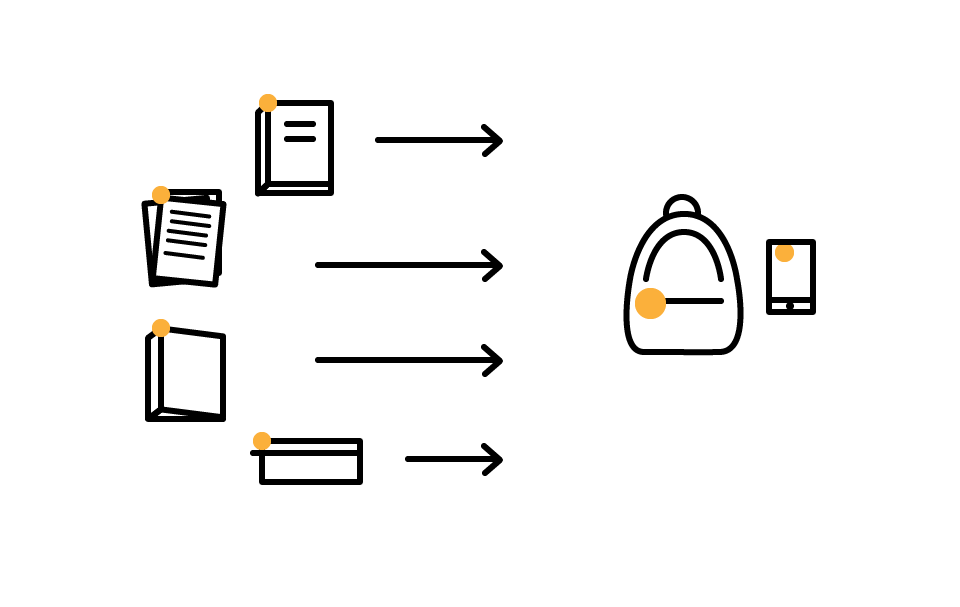


Mon collègue Axel et moi sommes les ingénieurs qui s’occuperons de la partie technique du projet, Tabea s’occupera de toute la partie économique, Julia de l’interaction avec l’utilisateur et Loic du design et des parties mécaniques.

# 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.



## Historique des idées

Avant d’avoir notre idée, nous sommes passé par plusieurs autres idées :

### Smart garden

### Paddle

### Tracker de ski

## Composants

<liste composant>

## 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.

<schéma bloc>

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.

# 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.

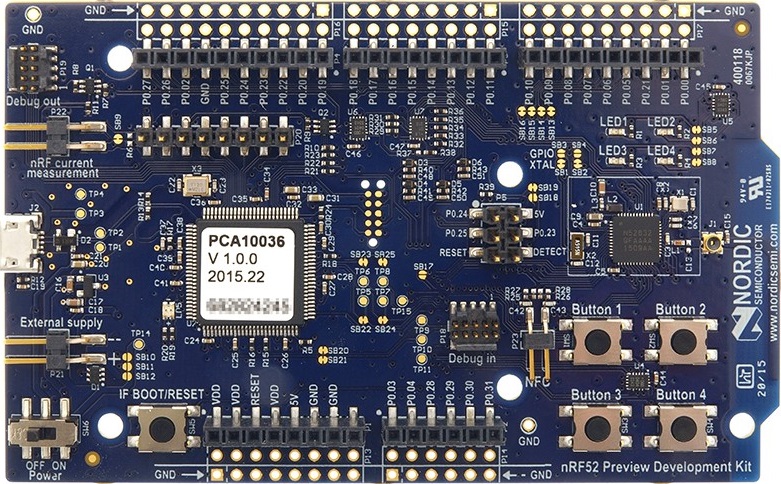
Ensuite nous sommes allés voir chez Texas Instrument qui possède le CC2540 avec Bluetooth intégrée, mais il n’est pas basé sur un cœur ARM mais sur un 8051. Ce qui implique des limites du côté de la mémoire et surtout qu’il ne fait pas le poids comparé aux solutions basée sur ARM.

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.

# 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.

Attention !

Le Bluetooth Low Energy est pas mal différents du Bluetooth 2.1EDR

## Generic Attribute Profile (GATT)

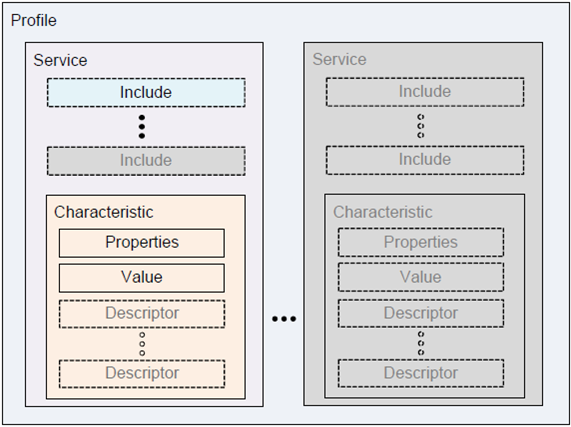
Le *GATT* est une structure de données utilisé pour le *BLE* qui définit les messages que les deux périphériques Bluetooth vont s’envoyer.

Figure 1 GATT

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.

Nous n’allons pas créer un profil complet mais plutôt modifier un profil 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.

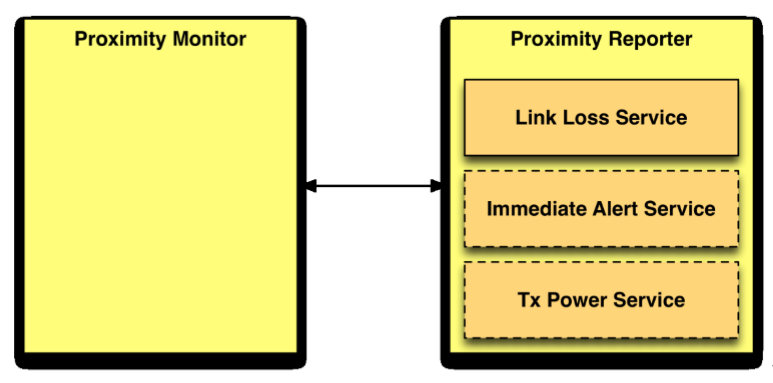


Figure 2 PXP

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 le *Proximity Reporter* est dédié à l’esclave. <…> Ce rôle possède un service obligatoire : le Link Loss Service

### Notre profil

Notre profil va rajouter plusieurs services. Concrètement le maitre va devoir envoyer certaines commandes à la base, ce qui fait que nous allons ajouter un service qui enverra des commandes. < ?>

Et l’esclave va devoir répondre avec une liste d’objets, ce qui fait que nous allons rajouter un service de liste d’objet.

## SoftDevice S132

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

# RFID

<collet>

# 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)  - Non rechargeable  - Capacité limitée | + Meilleurs encombrement/capacité  + Toute forme possible  + Rechargeable  - 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  + Bon marché  - Transition vers le type-C | + Réversible  + Transition vers le type-C  - Non démocratisé  - Prix conséquent encore |

# Sources

## Bluetooth

<https://www.bluetooth.com/specifications/adopted-specifications>

# TODO

* Changer style doc
* Faire les tutos nrf
* Faire intro ! xD