Une image contenant texte

Description générée automatiquement

Antonin CHACHKINE

Julien JANVIER

Morgan LE ROUX

Documentation Objet Connecté

Mölkky Connecté

Table des matières

[Présentation du projet 3](#_Toc125937754)

[Contexte du projet 3](#_Toc125937755)

[Cahier des charges 3](#_Toc125937756)

[Solution développée 3](#_Toc125937757)

[Quilles intelligentes 3](#_Toc125937758)

[Panier connecté 3](#_Toc125937759)

[Présentation du hardware 5](#_Toc125937760)

[Choix des composants : 5](#_Toc125937761)

[Philosophie de conception 6](#_Toc125937762)

[Réalisation de la carte 6](#_Toc125937763)

[Les quilles 7](#_Toc125937764)

[Le panier 8](#_Toc125937765)

[Fusion 9](#_Toc125937766)

[Système de recharge 9](#_Toc125937767)

[Problèmes rencontrés 9](#_Toc125937768)

[Solutions proposées 10](#_Toc125937769)

[Pistes d’amélioration 11](#_Toc125937770)

[Présentation du software 12](#_Toc125937771)

[Overview 12](#_Toc125937772)

[Pinout 12](#_Toc125937773)

[projet-nrf52 13](#_Toc125937774)

[config\_perso.h 13](#_Toc125937775)

[config.h 13](#_Toc125937776)

[main.c 13](#_Toc125937777)

[molkky\_pin.c 14](#_Toc125937778)

[molkky\_basket 15](#_Toc125937779)

[ble\_app\_uart 17](#_Toc125937780)

[Améliorations 18](#_Toc125937781)

# Présentation du projet

## Contexte du projet

Le Mölkky Club d’Anjou cherche une solution innovante permettant de simplifier le déroulement d’une partie en comptant les point et en permettant une intégration dans un système de gestion de tournoi.

## Cahier des charges

La solution technique doit répondre aux contraintes suivantes :

* Le dispositif doit avoir une autonomie permettant de jouer plusieurs matchs sans recharge
* Les dimensions doivent être proches de celle du panier d’origine
* Le poids et l’équilibre des quilles doivent rester inchangés
* Les quilles doivent pouvoir être remplacées facilement
* Plusieurs jeux de Mölkky doivent pouvoir être utilisés en même temps sans interférences
* La chute des quilles doit être détectée
* L’état de charge des batteries doit pouvoir être consulté
* La connexion au jeu doit être simple

## Solution développée

La solution technique développée comporte 2 parties :

### Quilles intelligentes

Embarquant une carte permettant la détection de chute, leur rôle est d’envoyer un message en ‘’broadcast’’ lorsqu’une chute est détectée par l’accéléromètre (mpu6050). Ce message est envoyé par le microcontrôleur (NRF 52832) en utilisant le protocole *enhanced shockburst (esb)*. L’esb est un protocole propriétaire de communication basse consommation développé par Nordic sur la bande 2.4 GHz. La communication se faisant en broadcast, la quille émet son message ‘’à qui veut l’entendre’’ et n’a pas besoin de rester connectée à la cible du message.

### Panier connecté

Composé de deux cartes, le panier a pour mission de recevoir les messages des quilles, de les filtrer en fonction de leur appartenance au jeu du panier et du type de message qu’elles envoient. Une fois filtrés, les messages sont transmis au téléphone via le protocole *Bluetooth Low Energy* (BLE).

Afin de remplir cette mission, une première carte s’occupe de la réception des messages envoyés en esb et de leur tri. Elle transmet ensuite ces messages en UART à une deuxième carte qui se chargera de les envoyer au téléphone via BLE.

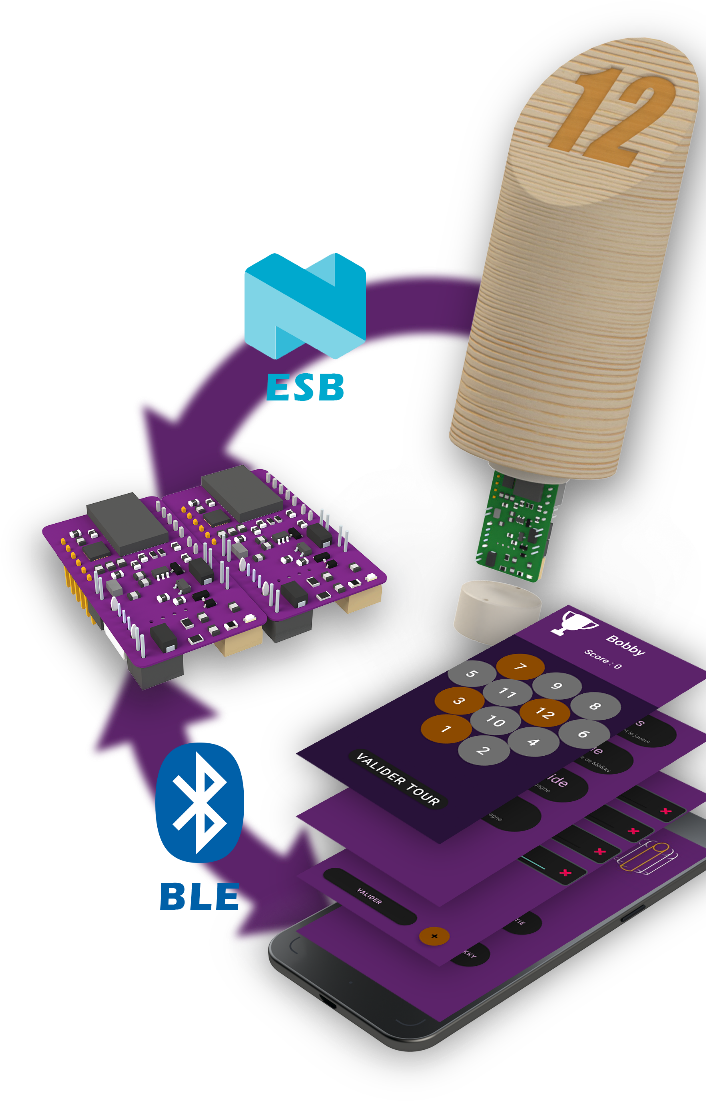


Schéma de fonctionnement de la solution technique

# Présentation du hardware

## Choix des composants :

Besoin : Communication de chaque quille vers une passerelle qui retransmettra les informations vers le téléphone.

Très peu de données seront envoyées de la quille vers la passerelle.

Choix du protocole de communication : ESB pour la communication entre les quilles et la passerelle car permet une diffusion de l’information en ayant très peu de coût énergétique.

Choix du microcontrôleur : le protocole ESB est un protocole proposé par Nordic. Nous avons donc recherché un microcontrôleur capable de communiquer avec ce protocole pour un coût raisonnable.

Le nrf 52832 correspond à ces critères. Il s’alimente à 3v3. Il possède une RAM suffisante.

<https://www.mouser.fr/ProductDetail/Adafruit/4077?qs=vLWxofP3U2wwdn9WwvfdQA%3D%3D>

Choix de l’accéléromètre : Nous souhaitions un accéléromètre capable de communiquer en protocole i2C. Nécessitant une alimentation de 3V3 pour correspondre à notre microcontrôleur.

Nous avons choisi le MPU6050 que l’on trouve au magasin de l’ESEO.

<https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>

Choix de la batterie : Nous souhaitions une batterie avec une capacité supérieure à 100mAH pour garantir une autonomie de plusieurs parties sans nécessité de recharge. Avec une tension nominale supérieure à 3V3. Une batterie de 3.7V de tension nominale est parfaite puisque sa tension variera de 3V à 4.2V en fonction de sa charge.

[https://www.amazon.fr/300mAh-Li-ION-Batterie-Chargeur-Quadcopter/](https://www.amazon.fr/300mAh-Li-ION-Batterie-Chargeur-Quadcopter/dp/B09QM2WGRX/ref=sr_1_1_sspa?__mk_fr_FR=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=2KXE2ODB2UPW8&keywords=batterie+3.7v+300mah&qid=1668674800&qu=eyJxc2MiOiIzLjY1IiwicXNhIjoiMy41MiIsInFzcCI6IjIuNjYifQ%3D%3D&sprefix=batterie+3.7v+300mah%2Caps%2C113&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1&smid=A31GY6G2YYT5K6)

Choix du régulateur de la tension de la batterie : Comme notre courant de sortie est relativement faible, nous avons décidé d’utiliser un LDO plutôt qu’un Buck pour fournir notre tension de 3V3 en permanence à nos composants.

Nous avons choisi un LDO capable de sortir 3v3 en sortie avec une tension d’entrée comprise entre 3V et 4.2V pour être efficient peu importe l’état de charge de notre batterie. Le LDO : TPS76333DBVT correspond.

<https://www.mouser.fr/ProductDetail/Texas-Instruments/TPS76333DBVT?qs=tJ5HNKWh3OU6kt%2FvF%2F5hDQ%3D%3D>

Choix du Boost :

Pour la recharge de nos quilles, nous avons besoin d’un boost

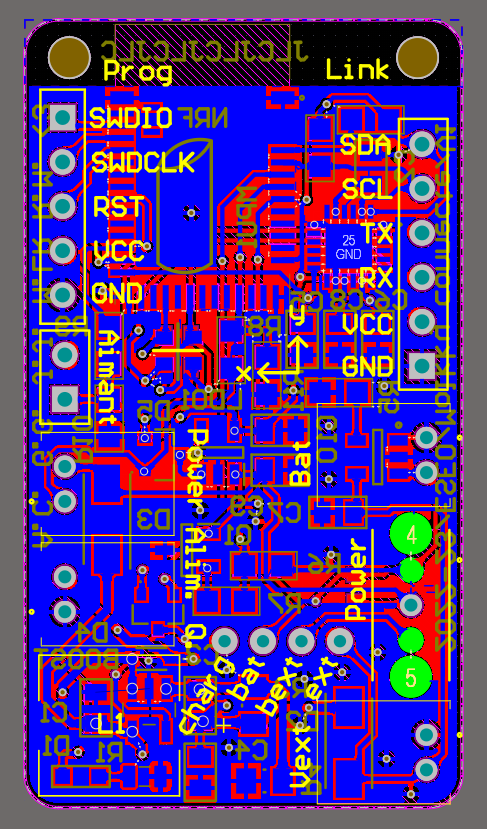
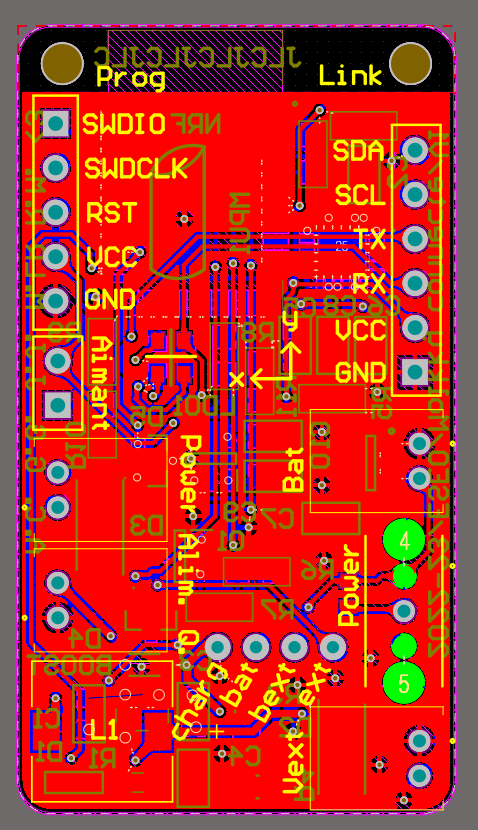
## Philosophie de conception

Afin de réduire au maximum les frais de production de notre jeu de molkky, nous avons fait le choix de ne réaliser qu’un seul type de carte, afin d’en commander plus et de réduire le cout unitaire de chaque carte. Ce choix découle du constat que les différents éléments de notre système (panier et quille) nécessitaient globalement les mêmes composants pour fonctionner.

Nous avons également été contraint par la place disponible. Les quilles ont une taille fine qui nous impose une taille maximum de PCB. Nous avons fait le choix de partir sur un PCB plus fin et plus long que celui réalisé précédemment. L’ajout des composants ne servant qu’a la partie panier ne représente pas une partie importante de la place de notre PCB.

Finalement, nous avons choisis de simplifier au maximum le système et de déporter l’intelligence sur l’application, afin de simplifier le PCB et de limiter sa consommation.

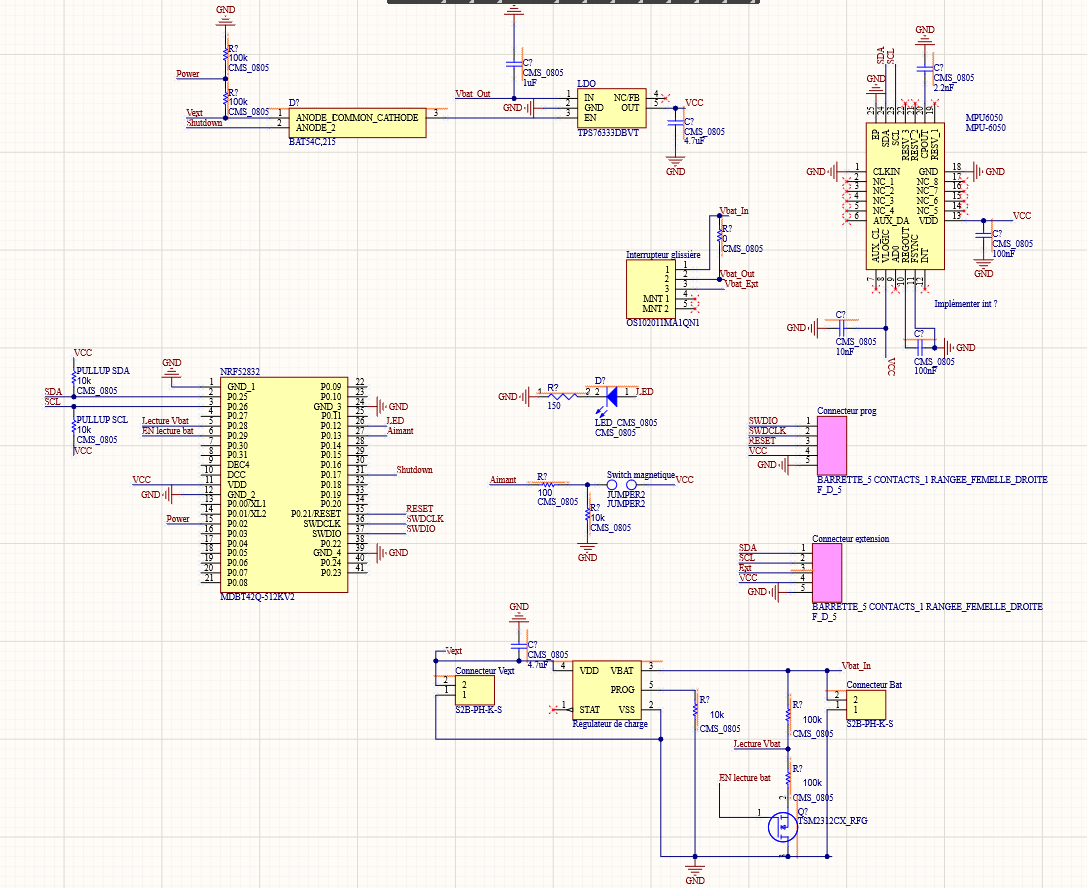
## Réalisation de la carte



La carte a été réalisée suite à la conception de deux schémas : un schéma du panier et un de la quille. Le fonctionnement des deux parties de ce système sont très similaires : les deux systèmes doivent gérer le chargement et déchargement d’une batterie, et ont des périphériques similaires (aimant / bouton d’alimentation), led …

Pour faciliter le débogage et la programmation de la carte, il a été décidé de laisser les bus I2C et UART accessible via des broches d’extension disponible d’un côté de la carte. Nous avons également décidé de faire passer VCC par ce même connecteur afin d’alimenter entièrement la carte du panier grâce à un LDO.

### Les quilles



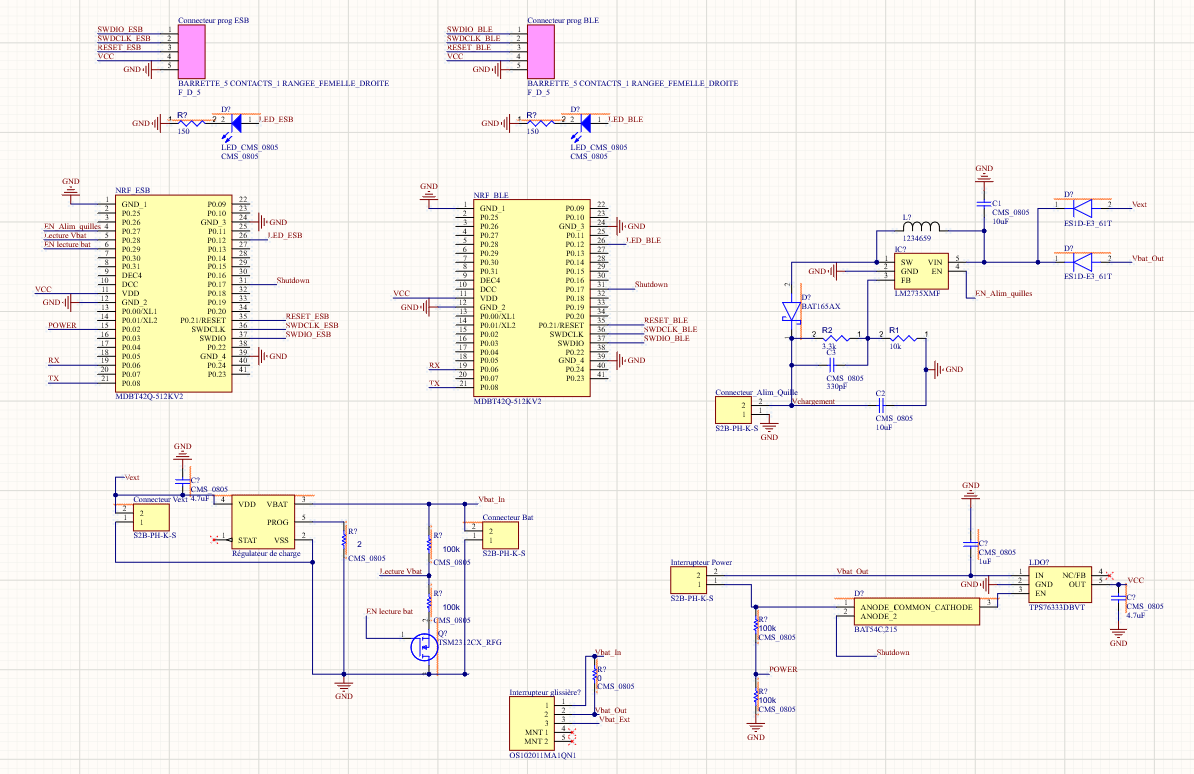
La conception de la quille est assez simple, elle fonctionne de façon indépendante et n’a accès qu’a deux capteurs extérieurs : l’interrupteur magnétique et le port de chargement. L’idée est de pouvoir démarrer la quille une fois que cette dernière est sortie du panier ou que le panier est démarré. La quille est donc en réveillé par un signal qui démarre son LDO. Cette dernière démarre et active un signal shutdown qui vient la maintenir en vie. Lorsque cette dernière décide qu’il faut arrêter de consommer, elle peut couper ce signal et se suicider pour couper sa consommation.

La lecture de la batterie se fait via un mosfet qui fait passer la tension de la batterie dans un pont de résistances. La tension est ensuite lue par un ADC de la carte. Cette solution est intéressante pour sa simplicité et sa taille faible.

Pour économiser de l’espace, les connecteurs ont été soudé vers l’intérieur. Un interrupteur permet de déconnecter la batterie du système et d’utiliser une batterie externe (connecteur bext). Afin d’économiser la place de cet interrupteur une résistance de 0 Ohm permet de bloquer cet interrupteur à la position fermé.

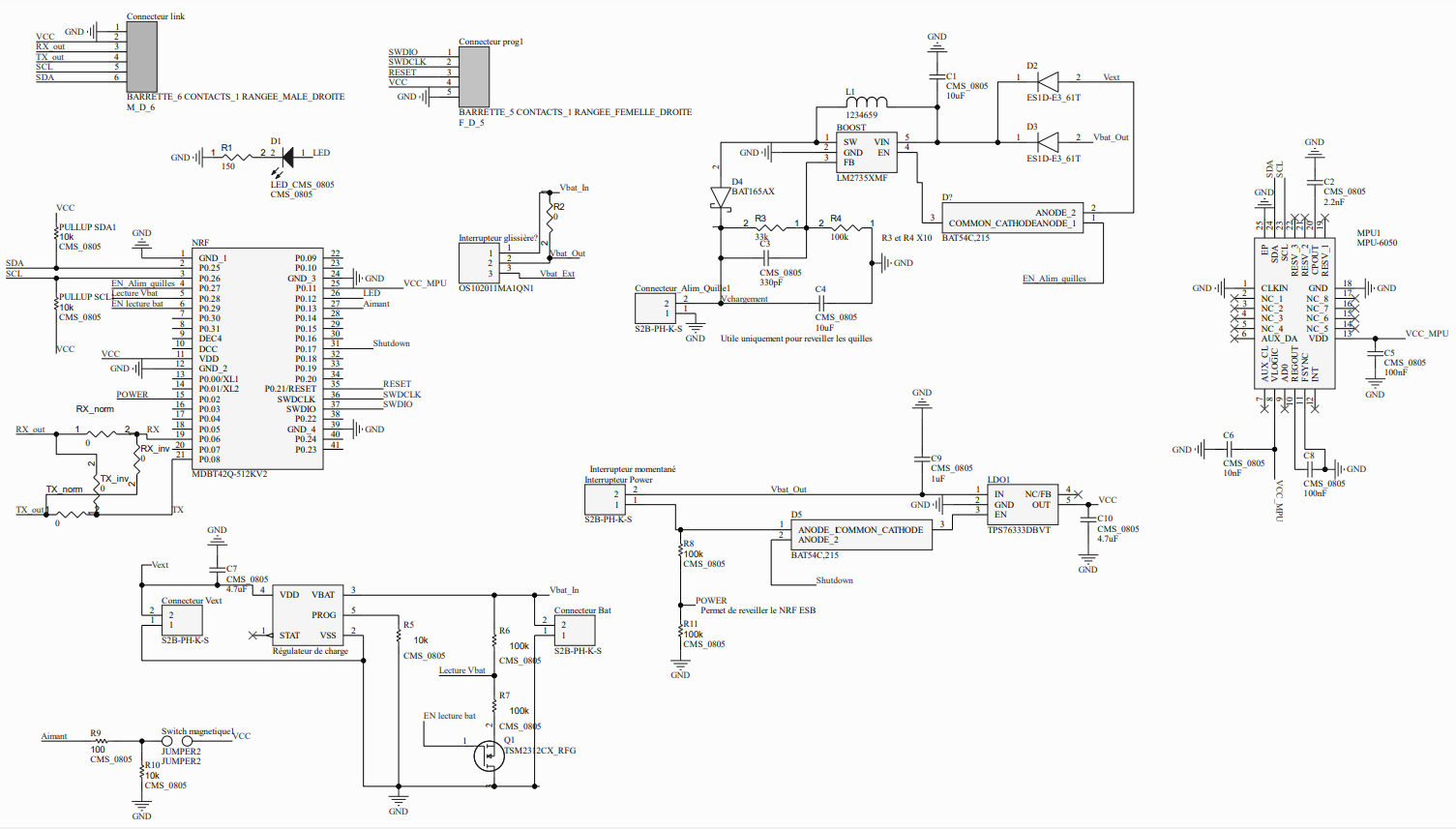
Il a également été décidé de placer le MPU le plus haut possible afin de maximiser les valeurs d’accélération lues et de rendre nos mesures plus fiables.

### Le panier



Le panier est constitué de deux cartes afin de nous permettre de communiquer en ESB et en BLE. La communication entre ces deux cartes est assurée par une liaison UART. Pour inverser les pins RX et TX des résistances de 0 ohms sont soudées d’un côté ou de l’autre du PCB. Par choix, la carte principale du panier, qui gère la batterie et le rechargement des quilles via le BOOST est la carte de l’ESB. La carte du BLE est donc très peu équipée et ne sert que de relai. Afin de monter les deux cartes cote à cote, la carte BLE à son connecteur Link soudé de d’autre coté que celui de la carte ESB. Ainsi les deux cartes assemblées en un bloc et peuvent être monter simplement.

### Fusion



La conception du PCB fusionné reprend les points précédents. Il sera noté que ce schematic reprend quelques modifications.

## Système de recharge

Dans une optique de durabilité, nous avons fait initialement le choix de recharger les quilles par induction. Cependant au vu des contraintes du projet et pour simplifier la réalisation d’un premier prototype, vous avons utilisé une prise Jack plus simple à implémenter. Bien que cette solution ne soit pas étanche, nous avons mis en place des systèmes de protection afin de ne pas craindre de court-circuit. Ce connecteur est plus robuste et plus fiable que le système de pistes proposé précédemment. Du point de vue de la carte l’utilisation d’un barrel Jack ou d’un chargeur par induction est invisible du côté de la carte, qui ne reçoit qu’un signal Vext. Afin d’implémenter un chargement par induction, il est simplement nécessaire de refaire le bas du housing et le support du panier.

Il a été décidé de mettre en place un système de guidage afin de centrer les quilles sur le support de chargement. Le design de ce support permet de placer la quille dans n’importe quelle position.

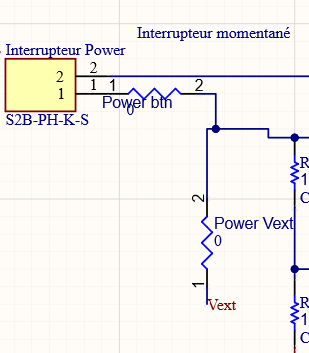
## Problèmes rencontrés

Une fois la réalisation de la carte, nous nous sommes rendu compte de deux problèmes :

* Un ilot de masse n’était pas relié au niveau de la capa de découplage C4.
* Il n’est pas possible de réveiller les cartes à l’aide de Vext, comme initialement prévu, à cause d’une erreur lors de la fusion des deux PCB.

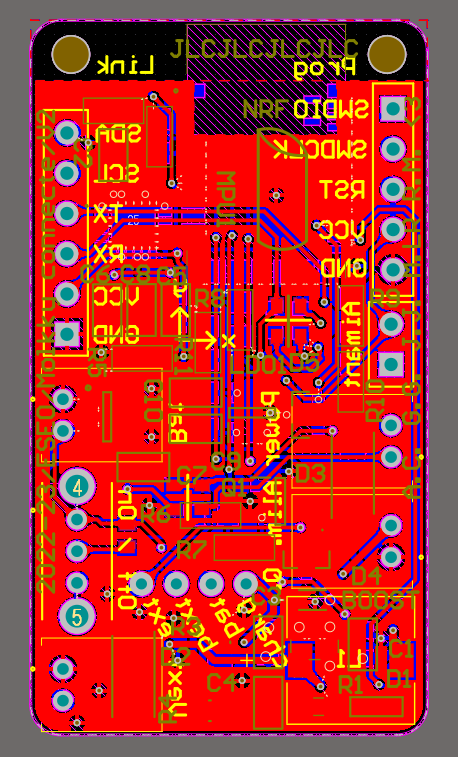
## Solutions proposées

Afin de résoudre ces problèmes, il à été ajouté deux résistances de 0 Ohms qui permettent de relier Vext ou le bouton d’alimentation au LDO, pour pouvoir démarrer les quilles de deux façons différentes.



Un via a également été ajouté pour palier l’ilot de masse.

Le routage des dernières modifications est effectué, la carte est prête à partir en production.



## Pistes d’amélioration

Les rectifications ayant été effectuées, il reste quelques pistes d’amélioration à creuser :

* Mettre en place le chargement par induction
* Repenser le rôle de l’interrupteur magnétique et trouver une alternative plus durable
* Séparer les cartes des quilles et du panier / explorer la piste d’un PCB sécable
* Multiplexer le BLE et l’ESB afin de n’avoir qu’un NRF dans le panier
* Concevoir une antenne afin de réduire le cout des NRF

# Présentation du software

## Overview

La solution développée comportant 3 types de cartes avec des fonctions différentes il y a donc 3 codes :

2 codes présents dans le ’’projet-nrf52’’ (projet général utilisé pour des projets à l’école) et l’exemple ble\_app\_uart du SDK.

Le code embarqué dans la quille a pour mission de :

* Gérer l’alimentation de la quille
* Effectuer des mesures d’angles à intervalles réguliers grâce au MPU6050
* Mesurer l’état de charge de la batterie
* Envoyer des messages en ESB
* Détecter la présence d’un aimant pour déclencher un mode appairage

Le code embarqué dans la première carte du panier a pour mission de :

* Stocker les identifiants des quilles appartenant au panier
* Réceptionner les messages ESB
* Filtrer ces messages en fonction de leur expéditeur
* Transmettre ces messages à la 2ème carte via UART
* Recevoir les messages UART (appairage)
* Déclencher l’allumage des quilles via EN\_Alim\_quilles

Le code embarqué dans la deuxième carte du panier a pour mission de :

* Réceptionner les messages UART
* Transmettre ces messages en BLE
* Réceptionner les messages en BLE (messages d’appairage)
* Transmettre ces messages en UART à la première carte du panier

## Pinout

|  |  |  |  |
| --- | --- | --- | --- |
| **Fonction** | **Pin carte quille** | **Pin carte panier** | **Pin carte de test (Bluefruit)** |
| Enable LDO | 17 | NA | x |
| Enable lecture Vbat | 29 | NA |  |
| Lecture Vbat | 28 | NA |  |
| Lecture Power | 2 | NA |  |
| UART Rx | 6 | 6 / 8 |  |
| UART Tx | 8 | 8 / 6 |  |
| SDA | 25 | NA |  |
| SCL | 26 | NA |  |
| Lecture aimant | 13 | NA |  |
| LED | 26 | 26 |  |
| Enable alim quilles | NA | 27 |  |

## projet-nrf52

Le projet ‘’projet-nrf52’’ se veut généraliste il possède donc différents modules qui doivent être configurés en fonction de nos besoins.

Les fichiers principaux communs à tous les objets sont :

* config\_perso.h
* config.h
* main.c

Les fichiers concernant nos objets sont :

* molkky\_pin.c
* molkky\_basket.c

### config\_perso.h

Le fichier config\_perso.h est un simple fichier de configuration qui permet de modifier des paramètres globaux en fonction de notre utilisation du projet.

Ici, on peut choisir l’objet sur lequel on travaille, l’activation ou non des leds ainsi que le type de carte sur laquelle sera exécuté le code (carte de test bluefruit feather ou carte custom).



### config.h

Le fichier conf.h est le fichier de configuration principal du projet qui permet d’activer les différents modules nécessaires pour chaque objet

Une image contenant texte

Description générée automatiquement

### main.c

Le fichier main.c est le fichier qui permet de faire le lien avec les machines à états de tous les objets. On trouve aussi dans ce fichier une partie d’initialisation commune à tous les objets (GPIO, timers).

Le µcontrôleur devant activer lui-même le LDO permettant d’alimenter la quille, on met à l’état haut la broche ENABLE\_LDO au début de cette initialisation

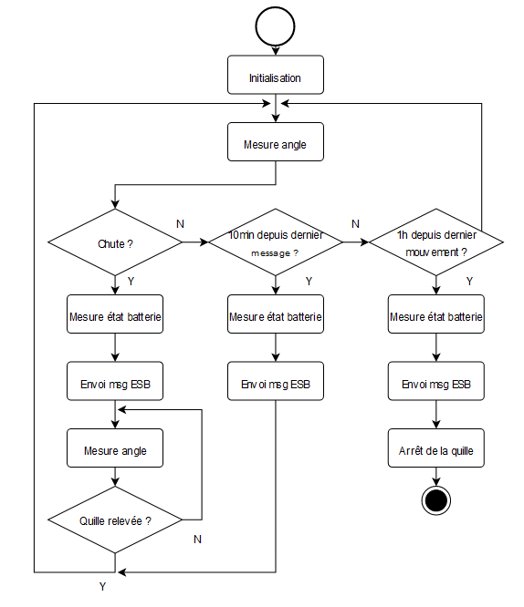
Une image contenant texte

Description générée automatiquement

(Rappel de fonctionnement : Le panier alimente la quille qui démarre, c’est ensuite au µcontrôlleur de choisir de rester allumé ou non, il a ce contrôle via la broche ENABLE\_LDO)

### molkky\_pin.c

Ce fichier contient la machine à états principale pour la quille, avant de détailler le code, voici un diagramme de fonctionnement simplifié de ce que fait actuellement la quille :



Notre machine à états est constituée de 7 états :

* START : Etat d’init, on active les GPIOs, l’ESB, le MPU, etc…
* IDLE : On attend de détecter une chute, on passe ensuite dans l’état CHECK\_FALL. On regarde aussi depuis combien de temps on n’a pas bougé ou envoyé de messages. En fonction du résultat on peut choisir d’envoyer un message simple avec le niveau de batterie ou de s’éteindre.
* CHECK\_FALL : On refait plusieurs mesures afin de confirmer la chute, on passe ensuite dans l’état FALLEN, sinon on retourne en IDLE.
* FALLEN : On remplis les champs de notre trame esb puis on l’envoie NB\_MSG\_MAX fois. On passe ensuite à l’état WAIT.
* WAIT : Comme dans IDLE mais on attend que la quille soit relevée, on passe ensuite à CHECK\_UP
* CHECK\_UP : On refait plusieurs mesures afin de confirmer que la quille est relevée, on passe ensuite dans l’état IDLE, sinon on retourne en WAIT.
* ERROR : Etat permettant de notifier d’une erreur (ici on fait juste clignoter la led)

L’envoi des message [esb](https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/ug_esb.html) se base sur les codes d’exemple du SDK *esb\_ptx* et *esb\_prx* dont les explications sont [ici](https://developer.nordicsemi.com/nRF_Connect_SDK/doc/2.2.0/nrf/samples/esb/README.html)

La charge utile (payload) de notre message est constituée de 5 octets (uint8\_t). Elle est de la forme

Une image contenant texte

Description générée automatiquement

La quille peut être amenée à envoyer 4 types de messages :

* PIN\_INIT : mode appairage lorsqu’un aimant est détecté
* PIN\_FALLEN : lorsque la quille vient de tomber
* PIN\_DEATH : lorsque la quille va s’éteindre
* PIN\_BATT : lorsque la quille envoie simplement son état de charge au panier pour lui indiquer qu’elle est encore ‘’en vie’’

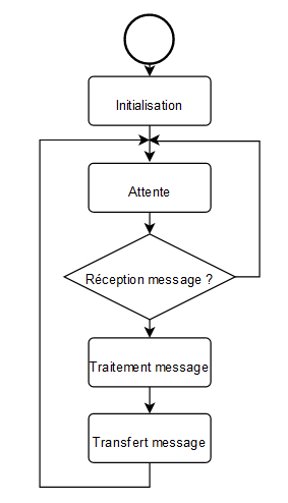
Une image contenant texte

Description générée automatiquement

Actuellement, pour démarrer une quille, il faut passer l’interrupteur vers le ‘’haut’’ (vers le µcontrôleur) puis passer un aimant près du switch magnétique

### molkky\_basket

Ce fichier contient la machine à états principale pour la première carte du panier, avant de détailler le code, voici un diagramme de fonctionnement simplifié de ce que fait actuellement cette carte :



Ici les messages sont reçus via esb et transmis via UART.

Ce fichier est basé sur les codes d’exemple du SDK [esb\_prx](https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v15.0.0%2Fuart_example.html) et [uart](https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v15.0.0%2Fuart_example.html).

Ici, la machine à états principale n’est constituée que de 2 états : un état d’initialisation et un état d’attente. Le traitement des données reçues se fait lors des interruptions provoquées par la réception de messages esb ou uart.

Lors du traitement des données, le panier fait les actions suivantes :

* Vérification de la taille de la payload reçue, elle doit être de 5 octets
* Identification de la quille (son ID unique correspond-il à celui d’une des quilles déjà enregistrées)
* Calcul de la distance séparant la quille du panier grâce au RSSI
* Transfert de la trame en UART en ajoutant la distance mesurée et le charactère ‘\n’ indiquant la fin de la trame

## ble\_app\_uart

Ce projet [d’exemple](https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v15.0.0%2Fuart_example.html) présent dans le sdk permet de transférer en BLE un message reçu en uart.

Afin de le faire fonctionner, il est nécessaire d’installer la pile bluetooth ([SoftDevice](https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v15.0.0%2Fuart_example.html)) sur la cible.

Pour faire cela en utilisant Eclipse, deux ‘’debug configuration’’ sont nécessaire. Une première permettant de téléverser le .hex du softdevice, un deuxième pour téléverser notre programme :

*Configuration pour le Softdevice :*

Une image contenant texte

Description générée automatiquement

Une image contenant texte, capture d’écran, intérieur

Description générée automatiquement

*Configuration pour le programme :*

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

A chaque nouveau téléversement, il faut bien faire d’abord celui de Sofdevice puis celui de notre programme.

## Améliorations

La solution technique actuelle permet de détecter la chute de plusieurs quilles mais elle nécessite d’allumer manuellement les quilles (mettre l’interrupteur vers le µcontrôleur puis passer un aimant au-dessus du switch magnétique).

Afin d’en faire une solution utilisable au quotidien, des modifications / améliorations sont nécessaires :

**Améliorations de la quille :**

1. Amélioration de la mesure de la batterie

Actuellement, la mesure d’état de la batterie est implémentée mais elle ne fonctionne pas.

1. Amélioration de la mise hors tension de la quille

Actuellement, la quille est capable de s’éteindre lorsque la batterie est faible ou lorsqu’elle est immobile trop longtemps. La mesure de la batterie ne fonctionnant pas, la mise hors tension a été désactivée. Une fois réactivée, il sera nécessaire de vérifier que l’extinction fonctionne correctement

1. Amélioration des mesures du MPU6050

Actuellement, un problème avec les MPU oblige de les réinitialiser avant chaque mesure ce qui prends du temps et réduit la fréquence d’échantillonnage. Une fois les MPU fonctionnant correctement, il faudra enlever les fonctions d’init du MPU présentes à chaque mesure. Il faudra aussi augmenter le nombre de mesures nécessaires pour valider la chute d’une quille

1. Optimisation de la batterie

Actuellement, le µcontrôleur fonctionne en permanence. Il pourrait être intéressant de le placer en ‘’deep-sleep’’ entre chaque mesure avec l’accéléromètre afin d’économiser de la batterie.

1. Ajout de la fonction d’appairage,

Il sera nécessaire d’ajouter une fonction d’interruption permettant d’envoyer des messages d’appairage (message avec comme type PIN\_INIT) à intervalles réguliers lorsqu’un aimant sera détecté sur l’entrée ‘’ Lecture aimant’’

1. Automatisation de l’ID

Actuellement, l’ID des quilles est donné à la main pour chaque quille, il serait intéressant d’utiliser [l’ID unique](https://devzone.nordicsemi.com/f/nordic-q-a/22089/specific-unique-id-for-device-detection) du µcontrôleur à la place.

**Améliorations de la carte réception esb du panier :**

1. Ajout de l’appairage

Actuellement, le panier ne transmet que les messages de quilles qu’il connait. Il serait intéressant de transmettre aussi les messages d’appairage venant d’autres quilles.

1. Gestion dynamique des quilles enregistrées

Actuellement un tableau contenant les ID des quilles du panier ainsi que leur numéro (compris entre 1 et 12) est généré à chaque démarrage et n’est pas stocké en dur ce qui empêche une gestion dynamique des quilles. Il faudrait stocker ce tableau de façon permanente et gérer les messages d’appairage venant du téléphone afin de permettre la gestion dynamique des quilles.

1. Amélioration de la mesure de distance

Actuellement, les mesures de distances sont fausses, il faudrait retravailler les calculs de distance à partir du RSSI

1. Alimentation des quilles pour les démarrer

Actuellement, le panier ne permet pas d’activer les quilles, il faudrait ajouter l’allumage en utilisant le pin ‘’Enable alim quille’’

**Si des informations sont manquantes, envoyez vos questions à morgan.leroux@reseau.eseo.fr**