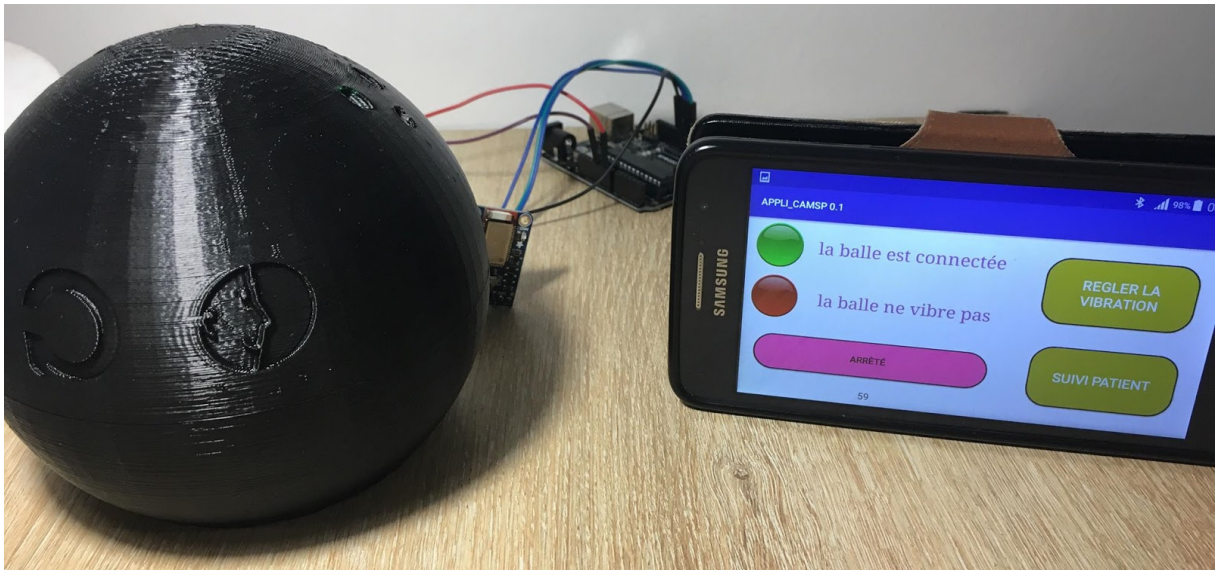
Rapport de Projet IMA4

Projet 15 : Balle vibrante connectée pour enfants sourds



Réalisé par : CATTELAIN Thibault
CUNIN Thomas

Encadrants école : ASTORI Rodolphe
BOÉ Alexandre
VANTROYS Thomas

Sommaire

Remerciements :

Nous tenons à remercier le fabricarium de Polytech'Lille pour nous avoir apporté de précieux conseils dans la conception de la balle et nous avoir permis d'utiliser les imprimantes 3D, Mr FLAMEN et Mr BOE pour leur aide dans la réalisation du PCB ainsi que Mr REDON et Mr VANTROYS qui ont bien voulu user de leur temps pour nous imprimer nos prototypes et nous donner de précieux conseils.

Introduction :

Un certain nombre d'enfants atteints de surdité dès la naissance ne peuvent pas être appareillés. Cela signifie qu'il est impossible de leur implanter un appareil auditif leur permettant d'avoir une ouïe partielle ou totale. Cela pose un problème lorsque l'enfant est atteint de surdité depuis la naissance car il n'a jamais pu faire le lien entre le mouvement des lèvres d'une personne et un son qui est émis.
La partie du cerveau devant initialement gérer cela est donc prise pour effectuer d'autres tâches. Lors de la rééducation de l'enfant, cela pose donc des problèmes. Pour pallier à cela, lors des séances de rééducation d'enfants ayant ce problème, les praticiens font passer un objet vibrant sur l'enfant lorsqu'ils parlent. Ainsi l'enfant peut faire un lien entre un mouvement de lèvres du praticien et les vibrations. Seul problème, ces praticiens n'ont pas d'outil leur permettant de faire cela d'une manière simple et efficace (en effet, pour l'instant ils utilisent une brosse à dent électrique pour effectuer les vibrations).

Notre projet réalisé en partenariat avec le Centre d'action médico-sociale précoce (CAMSP) de Montfort consiste donc à leur créer un objet permettant de répondre à leur besoin afin de rendre les séances de rééducation plus confortables pour le praticien, et plus efficaces pour l'enfant.

I) Elaboration du produit

1) Définition du besoin

Dans l'introduction, nous avons énoncé assez succinctement ce dont avait besoin le CAMSP. Nous allons reprendre cela en détaillant plus précisément.

Nous n'avons pas eu l'occasion de rencontrer le personnel du CAMSP, mais ces derniers avaient laissé des directives assez claires par écrit. Avec ces documents, nous avons pu définir un cahier des charges pour identifier clairement le besoin et ne pas développer un produit qui ne correspondait finalement pas à leurs attentes :

- L'institut a besoin d'un objet pouvant être facilement tenu par des enfants (les patients seront âgés de 0 à 6 ans) et facilement lavable comme un objet du type ours en peluche ou balle.

- Cet objet doit être capable de détecter le son et de vibrer en conséquence. Une vibration modulable selon l'intensité du son est également souhaitable.

- La liaison entre le micro captant le son et l'objet doit être obligatoirement sans fil.

- L'objet doit avoir une autonomie d'au moins 1 journée.

- Une application Android doit pouvoir être associée à cet objet. Nous devons pouvoir à partir de l'application :
    - Régler le niveau d'intensité de la vibration.
    - Accéder à un suivi des patients.
    - Pouvoir activer ou désactiver la balle.

2) Définition du cahier des charges

A partir des besoins énoncés par le CAMSP, nous avons pu décider que notre objet serait une balle pour plusieurs raisons :

- Il sera facile d'insérer tout l'électronique à l'intérieur de la balle.
- Une balle est bien plus facile à nettoyer qu'un ours en peluche.

Nous avons donc réalisé le cahier des charges en accord avec les besoins énoncés ci-dessus.

Celui-ci est disponible dans la partie documents rendus de notre wiki.

II) Création du modèle

1) Elaboration du prototype et répartition du travail

Une fois le cahier des charges terminé, nous avons pu déterminer la manière dont nous allions créer la balle, la rendre vibrante, et après réflexion, nous avons décidé d'adopter le schéma suivant :

- La balle sera conçue via l'impression 3D. Le logiciel de CAO OnShape permettra de réaliser un modèle 3D, puis nous pourrons obtenir un modèle en PLA via les imprimantes 3D à notre disposition au sein de Polytech.

- Pour assurer une vibration homogène dans toute la balle, celle-ci devrait contenir 6 vibreurs placés sur toute la surface de la balle.

- La connexion entre l'application et la balle se fera via une liaison Bluetooth. Nous avons choisi un module Bluetooth Low Energy (BLE) pour sa faible consommation en courant, le faible débit ainsi que la faible distance de notre utilisation.

- La balle sera équipée d'un microprocesseur Atmega328p ainsi que d'un module Bluetooth Low Energy pour récupérer les données envoyées de l'application Android via Bluetooth. L'Atmega328p contrôlera également les 6 vibreurs via des contrôleurs moteurs. Tout cela sera alimenté par une batterie placée également dans la balle. Il faudra également placer un boost converter pour stabiliser la tension en sortie de la batterie.

- Pour la recharge de la batterie, nous avons opté pour un système de recharge par induction. Ce choix a été motivé par le fait que ce type de recharge ne nécessite aucun fil, ou port USB (qui ne serait pas étanche). Il faudra également intégrer un chargeur Li-ion pour charger la batterie (on ne peut pas charger une batterie li-on en injectant simplement une tension constante).

- La balle ne pourra fonctionner que grâce à l'application. Cette dernière va gérer le son du microphone du smartphone, calculer son intensité puis enverra par Bluetooth une trame indiquant la puissance à laquelle devra vibrer le moteur. L'application pourra également demander à l'utilisateur de changer le niveau d'intensité de vibration. En effet, certains enfants étant plus sensibles que d'autres aux vibrations, il faut pouvoir faire en sorte de changer la vibration suivant les patients.

Il y avait donc 3 grands points à traiter :

- Création de la coque de la balle via impression 3D.
- Création de l'application Android.
- Création d'une carte électronique et implémentation de tout le système mécatronique nécessaire au bon fonctionnement de la balle.

A la vue de tout le travail à fournir, nous avons décidé de nous répartir les tâches pour pouvoir finir notre prototype et avoir une application présentable dans les délais impartis. Une personne s'est donc occupée de la création de la balle en impression 3D ainsi que de la création de l'application Android tandis que l'autre a pris en charge la création du PCB qui irait dans la balle, du développement des programmes embarqués dans la balle ainsi que du montage final de la balle.
Bien qu'il y ait eu une répartition des tâches, chacun de nous est resté totalement informé sur ce que l'autre faisait, des réflexions en commun étaient faites régulièrement, et une vérification du travail était toujours faite afin de déceler des soucis à côté desquels nous serions passés.
Nous allons donc décrire les différentes parties de la création de la balle.

2) Création de l'application Android

i) Présentation de l'application

Pour lancer l'application, nous avons pensé que quand l'utilisateur l'ouvrirait, il arriverait sur une page principale ayant les fonctionnalités énoncées plus tôt, à savoir :

-Un bouton pour activer ou désactiver les vibrations de la balle.
-Un bouton permettant d'accéder à une page suivi patient.
-Un autre bouton permettant d'accéder à la page réglant la vibration.

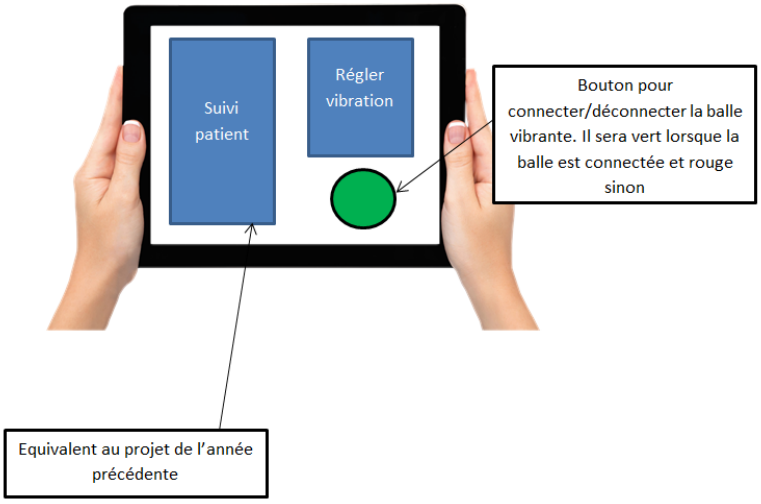


Figure 1 : simulation de la page principale de l'application.

Cette première page (ou activité) serait donc assez épurée car elle ne contiendrait au final que 3 activités. Voici à quoi elle pourrait ressembler :

En cliquant sur le bouton régler vibration, nous arriverions sur une nouvelle page ou nous pourrions régler le niveau de vibration que nous souhaitons pouvoir atteindre. Celle-ci devrait ressembler à cela :

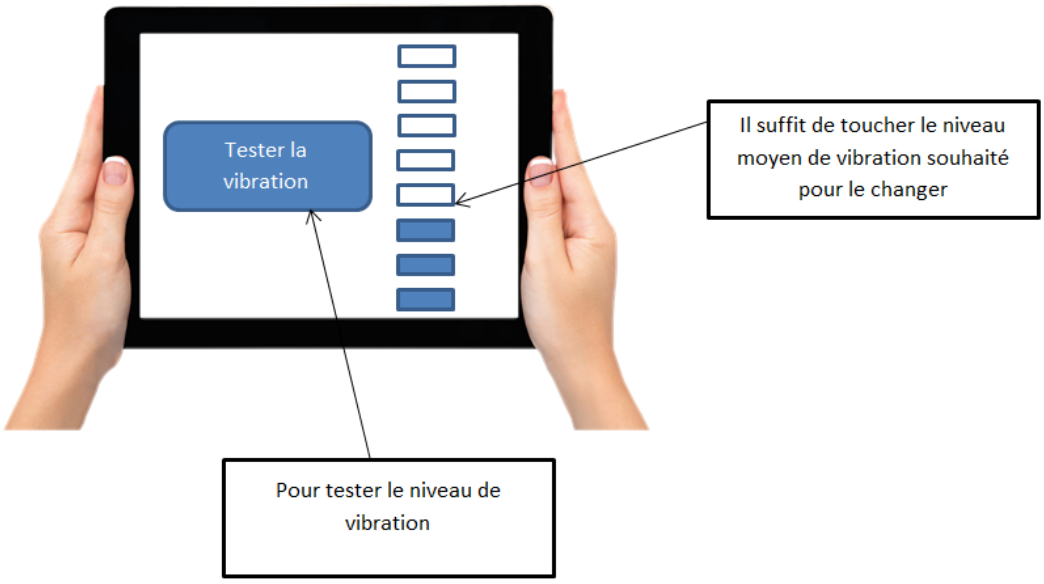


Figure 2 : simulation de la page permettant de régler la vibration de l'application.

Le fait d'augmenter ou de diminuer la vibration permettra de changer la plage de vibration de la balle. Par exemple, si l'utilisateur décide de régler la vibration sur très bas, alors pour obtenir des niveaux de vibration importants de la balle, il faudra que le micro capte un son très important, inversement, si la vibration est réglée sur un niveau important, de faibles sons impliquent néanmoins une vibration importante.

La page consacrée au suivi patient n'a quant à elle pas été programmée. Après concertation avec notre tuteur, nous avons décidé de ne pas nous en occuper car nous n'avons pas pu obtenir assez d'informations du CAMSP concernant ce qu'il fallait faire précisément.

Une fois que nous savions ce que nous voulions obtenir, il ne restait plus qu'à le coder.

La première chose àaété de faire toute la partie dite Front-end.

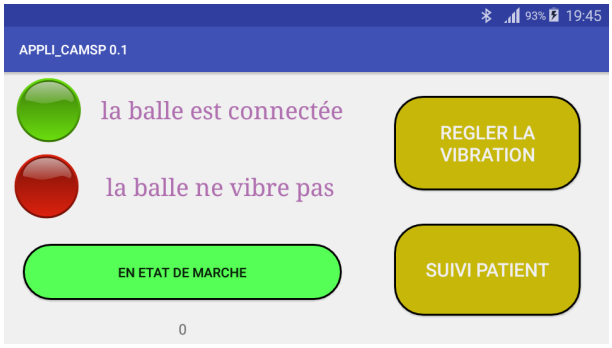
ii) Programmation xml pour le front-end

Par programmation front-end, nous voulons dire toute la programmation qui n'est destinée qu'à créer l'interface avec l'utilisateur. Mais toute la partie gestion de l'audio, du Bluetooth ou de l'interaction entre ces composants n'est pas traitée ici mais durant le développement back-end.

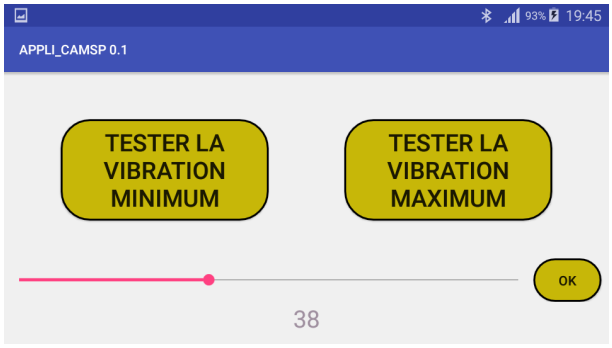
Pour réaliser tout cela, nous allons développer notre application via Android studios. Le choix de cet IDE a été motivé par le fait que c'est de très loin le plus utilisé et le plus puissant. D'autres logiciels plus rapides à prendre en main et qui ne nécessitent pas forcément de connaître des langages comme le java ou le XML existent, mais ceux-ci sont assez restreints et ne permettent pas de gérer de manière triviale des choses comme le Bluetooth ou l'audio.

La programmation de toute l'interface se fait donc en XML. Ne connaissant pas du tout ce langage, il nous a fallu un temps d'apprentissage avant de nous lancer. Des tutoriels comme ceux que l'on peut trouver sur OpenClassroom nous ont permis de nous former rapidement au langage XML mais également au langage et à la programmation java JEE.

Nous avons donc commencé à programmer l'application, et nous sommes rendu compte qu'il serait utile de rajouter un témoin indiquant lorsque la balle vibre ou non. Nous l'avons donc ajouté à notre interface. Après plusieurs essais et configurations différentes, nous avons obtenu ces designs pour les deux activités fonctionnelles de l'application.

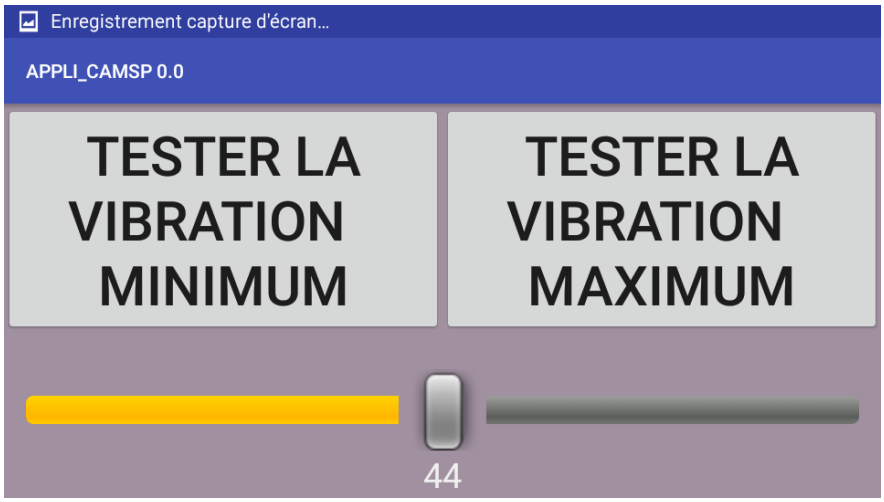


Version finale de l'activité principale



Version finale de l'activité réglage vibration

Il ne faut pas croire que nous avons obtenu cela au bout d'un essai. Il nous a fallu beaucoup de temps et surtout de pratique avant de pouvoir afficher quelque chose de correct. Voici par exemple une des versions de l'activité réglage vibration :



Version intermédiaire de l'activité réglage vibration

Cette version est déjà très aboutie, nos premières versions n'étaient même pas présentables.

Après que le design de l'application ait été créé, il a fallu s'attaquer au coeur du problème, la programmation back-end.

iii) Programmation java pour le back-end

Coder en utilisant le langage XML ne représentait pas un réel défi technique, en revanche, les choses sont devenues beaucoup plus compliquées lorsqu'il a fallu réaliser le corps de l'application. Comme expliqué précédemment, pour réaliser les interactions entre les widgets ou pour programmer la liaison Bluetooth entre la balle et l'application, nous avons utilisé le langage Java.

Nous avons commencé par le plus simple, à savoir lier les activités entre elles, faire en sorte que l'appui sur un bouton provoque une réaction de l'application…

Une fois cela fait nous étions un peu plus sûrs de nous et un peu plus chevronnés à la programmation Android, nous avons programmé la lecture du micro.

L'application doit récupérer l'intensité sonore grâce au micro puis envoyer périodiquement sa valeur au module Bluetooth.

Nous avons donc commencé par initialiser un Timer qui appellerait une méthode toutes les 250ms. Cette méthode serait en charge de déterminer l'intensité sonore, de la coupler au coefficient de vibration défini par l'utilisateur (via réglage vibration) puis d'appeler

la méthode qui envoie les données via Bluetooth. Il n'est pas forcément utile d'ajouter ici des bouts de code qui hors de leur contexte n'auraient pas vraiment de sens, mais vous pourrez retrouver l'ensemble de l'application dans la rubrique documents rendus du wiki.

Initialiser la connexion Bluetooth entre le smartphone et la balle et envoyer des données a été une des choses les plus complexes que nous avons dû gérer lors de la création de l'application. D'autant que comme expliqué plus haut, nous utilisons un Bluetooth Low energy, qui a une consommation beaucoup plus faible (chose très importante vu que nous voulions que la balle ait l'autonomie la plus grande possible) mais qui se programme d'une manière totalement différente d'un Bluetooth classique. De plus, il y a peu de documentation viable sur la façon de la programmer.

Pour faire en sorte que deux appareils puissent communiquer, nous pouvons procéder de deux méthodes différentes :

- En utilisant la méthode de l'advertising. Un appareil va envoyer des trames de données de façon régulière. Tous les appareils se trouvant dans le secteur vont pouvoir recevoir et lire ces trames. Dans ce mode, l'appareil qui émet des données ne peut pas en recevoir et vice-versa.
- En utilisant le mode connecté. On crée une connexion entre les deux appareils que l'on veut voir communiquer, puis ces deux appareils peuvent librement échanger des données, et ils ne peuvent communiquer qu'entre eux. Il est même possible de crypter les trames envoyées (nous n'irons pas jusque-là dans notre application).

Principalement pour des raisons matérielles (le smartphone que nous utilisions ne supportait pas la technologie de l'advertising) nous avons choisi d'adopter la méthode connectée pour envoyer nos données (pour information, nous nous sommes rendu compte que le téléphone n'était pas compatible après avoir déjà commencé à programmer la première méthode ce qui nous a fait perdre quelques jours de programmation). Après beaucoup de recherches sur le fonctionnement et l'initialisation de cette connexion, nous sommes finalement parvenus à obtenir une connexion puis un échange de données entre les deux appareils !

Comme pour la programmation de l'audio, il n'est pas vraiment utile d'afficher ici tout le code ayant permis d'établir une connexion Bluetooth, d'autant que celui-ci est plutôt volumineux et découpé. Mais une fois de plus, il est disponible dans la rubrique documents rendus de notre wiki.

Une fois que tout cela a été créé dans l'application, il a encore fallu corriger les très nombreux bugs que nous avons détecté. Cela a pris un temps considérable, mais nous avons pour finir une application qui se révèle fonctionnelle. Bien évidemment, il reste encore des axes d'amélioration.

3) Création de la carte électronique

Afin de réaliser ce projet, nous avons eu besoin de plusieurs “blocs électroniques” :

-Un bloc de commande et de traitement de l’information : l’Atmega328p.

-Un bloc de communication Bluetooth : le MDBT40.

-Un bloc de contrôle des moteurs : le L293DD.

-Un bloc de recharge de la batterie li-ion : le MCP73831T.

-Un bloc de stabilisation de la tension de la batterie (boost converter) : le TPS61090.

Les blocs principaux étant choisis, il est nécessaire de choisir et dimensionner les composants les accompagnants.

i) Le chargeur de batterie li-ion (voir Schéma 1)

La batterie utilisée étant de type Lithium-Ion, cette dernière nécessite un chargeur afin d'assurer le cycle de charge et de décharge. La charge est assurée selon les étapes suivantes :

-Courant maximum constant et tension minimale (3.2v) au début de la charge.

-Augmentation de la tension jusqu'à la tension maximale de charge (5v).

-Abaissement du courant de charge jusqu'à 0A.

-Le chargeur MCP73831T (U1) permet d'assurer ce cycle tout en pouvant être supervisé par des leds :

-Une led orange (D2) témoignant de la charge en cours.

-Une led verte (D1) témoignant d'une charge complétée.

-La tension d'entrée est également découplée (C1).

ii) Le boosteur de tension (voir Schéma 1)

Le module choisi pour maintenir la tension à 5v est le TPS61090. La tension de sortie de ce module dépend de la valeur des résistances R8, R9, R3, R7. Elles sont calculées selon les équations suivantes fournies dans la datasheet :

Équation 1:

$$R3 = R7 \times (\frac{Vo}{Vfb} - 1)$$

Avec R7 = 200kΩ, Vfb = 500mV d'après la datasheet. Vo, la tension de sortie souhaitée est de 5V.

Après calcul, R3=1.8MΩ.

Équation 2:

$$R8 = R9 \times (\frac{Vbat}{Vlb} - 1)$$

Ici, R9 = 340kΩ, Vlb = 500mV toujours d'après la datasheet. Vbat est la tension minimale de la batterie à partir de laquelle l'indication de batterie faible se déclenche. Cette tension est selon la datasheet de la batterie de 2.75V, mais nous préférons éviter tout risque : en cas de non utilisation de la batterie pendant une longue période, la batterie continue de se décharger à cause de courants de fuite. Si la tension de la batterie passe sous les 2.75V, elle deviendra inutilisable, c'est pourquoi nous minimisons les risques en fixant la tension de batterie faible Vbat = 3.2V.

Après calcul, R8 = 1.8V.

Équations 3 & 4:

Le boost converter nécessite un "boost inductor" afin de stocker l'énergie durant la conversion. Afin de pouvoir déterminer la valeur de cette inductance (L1), il est nécessaire d'estimer le courant moyen Il traversant l'inductance :

$$Il = Iout \times \frac{Vout}{Vbat \times 0.8}$$

On fixe ici Iout à 500mA, ceci donnera une charge plus lente, mais la balle ne chargeant que durant la nuit, le temps de charge n'est pas un problème, de plus ceci prolongera la durée de vie de la batterie. Concernant Vbat, on se place dans un cas très défavorable, avec une batterie extrêmement déchargée à Vbat = 1.8V. La tension de sortie désirée reste Vout=5V.

Après calcul, Il = 1.750A.

Avec ces données, nous pouvons maintenant dimensionner L1 :

$$\frac{Vbat \times (Vout - Vbat)}{\Delta Il \times f \times Vout}$$

Avec ΔIl = 0.2 * Il, f = 600kHz.

Après calcul, on trouve une inductance Il = 5.5uH au minimum.

Les capacités en entrée, C2 et C3, sont des capacités servant à améliorer le comportement transitoire du montage, et sont conseillées respectivement à 10uf et 0.1uf d'après la datasheet.

La capacité C5 en sortie sert à limiter le ripple (résidus de variation de tension), elle est conseillée à 100uF pour une sortie de 5V.

iii) Le module Bluetooth (voir Schéma 1)

Le module Bluetooth choisi fonctionne en 3.3v, or la seule tension d'alimentation présente est 5V, il faut donc un convertisseur 5v vers 3.3v. Nous nous sommes orientés vers un montage avec le MIC5225-3.3 (U5). Des capacités de découplage (C7 et C8) sont également ajoutées ainsi qu'une diode Schottky afin de protéger le circuit.

En revanche, une erreur dans la commande nous a amené à travailler avec un module MIC5225 qui, avec le schéma de câblage d'un MIC5225-3.3 (soit pate 4 inutilisée) délivrait 0V. Il a donc été nécessaire de rajouter deux résistances, R18 entre la patte 5 et la masse, R17 entre la patte 4 et la patte 5.
Le dimensionnement des résistances se fait de la manière suivante :

$$V_{out} = V_{ref}(1 + \frac{R17}{R18})$$

Avec Vref = 1.24V et Vout=3.3V

iv) Schéma électrique complet et PCB

Grâce aux calculs précédents, nous avons pu dimensionner les composants et obtenir un chargeur de batterie, une tension stable à 5V et une autre tension stable à 3.3V.
Les 5V serviront d'alimentation pour l'Atmega328P et les deux contrôleurs moteur. Le 3.3V servira d'alimentation uniquement pour le MDBT40.

La carte doit rester évolutive, nous avons donc fait en sorte de rendre disponible beaucoup d'entrées et sorties supplémentaires, en plus de celles servant à programmer et déboguer.

Nous avons donc décidé de rendre disponibles des entrées analogiques, en cas d'évolution vers un micro intégré à la balle, des entrées/sorties digitales venant de l'Atmega328P mais aussi une E/S venant du module Bluetooth. Cette dernière servira de témoin de vibration.

Concernant les pins servant à la programmation, nous avons décidé dans un premier temps de graver le bootloader de l'atmega avec des câbles soudés temporairement afin de ne pas surcharger le PCB, ceci était une erreur. En effet, les pins D11 et D12 de l'Atmega328P ont été utilisés afin de programmer le bootloader mais aussi afin de téléverser les programmes en raison du manque d'un FTDI.

Enfin, il n'était initialement pas prévu de programmer le module Bluetooth, mais il s'est avéré que cela était nécessaire. Dans une prochaine révision du PCB, il sera nécessaire de mettre un header supplémentaire sur les pins DATA et CLOCK du MDBT40 (voir Schéma 2) afin de pouvoir le programmer avec des simples câbles Dupont.

Enfin, pour un confort supplémentaire au moment du débogage, il peut être intéressant de mettre aussi un header sur les GPIO 18 et 19 (voir Schéma 2). Ces GPIO sont celles configurées en tant que RX/TX. Une interface série banchée sur ce header permet de

visionner les messages envoyés à l'Atmega328P par le MDBT40.
Le schéma final est visible en annexe, Schéma 1.

4) Programmation des périphériques embarqués

i) L'Atmega328P

Le rôle de l'Atmega328P dans ce circuit est de recevoir des instructions du module Bluetooth et de commander les contrôleurs moteur selon ce qui a été reçu.
La première chose à faire est de configurer la liaison entre le MDBT40 et l'Atmega328P. Nous allons donc configurer une liaison série à 115200 bauds.
Pour cela nous utilisons la librairie SoftwareSerial.h qui nous permet de configurer une liaison série sur des pins digitaux de l'Atmega.

Nous définissons alors RX sur D9 et TX sur D10, puis initialisons les pins :
SoftwareSerial mySerial(TXD_PIN, RXD_PIN);

Nous pouvons ensuite initialiser la vitesse de la connexion :
mySerial.begin(115200).

La liaison est maintenant prête, il n'y a plus qu'à surveiller l'état du drapeau :
mySerial.available().
Afin de savoir si une donnée est disponible sur le port série, et ensuite la lire avec :
mySerial.read().

Maintenant que la liaison série avec le module Bluetooth est prête, il faut maintenant gérer la réception des données.
Nous avons dans un premier temps utilisé des données sous la forme aXXX, bXXX et cXXX afin de contrôler respectivement les paires de moteurs a, b et c, avec XXX une valeur comprise entre 0 et 255. Le problème étant qu'à un instant t nous devons contrôler les trois moteurs en même temps, or jusque-là nous devons envoyer trois commandes distinctes, ce qui implique qu'entre l'envoi de la première commande et le traitement de la dernière s'est écoulé un temps non négligeable.

Nous avons donc décidé d'optimiser ce temps en réduisant les trois commandes en une seule : aXXX, bXXX et cXXX devient aXXXbXXXcXXX en un seul message. Le fait de garder les lettres b et c est un choix. Grâce à ça, il est possible d'envoyer le message a85b74c96 par exemple sans avoir à rajouter les 0.

Le code est donc fait de telle sorte que la donnée reçue doit être un a afin de commencer à regarder la suite de la donnée.
Lorsqu'un a est détecté, la donnée suivante est stockée dans une variable, puis la suivante additionnée à 10 fois la valeur stockée précédemment, puis stockée à son tour, etc, de la manière suivante :

data=data*10+c.

Lorsque b est détecté, data est stocké dans data1, et la valeur après b sera à son tour stockée dans data, et ainsi de suite jusqu'à la valeur de c.

A la fin du processus, data1, data2 et data3 contiendront les valeurs des moteurs a, b et c, pour être appliquées aux sorties PWM D3, D5 et D6.

La dernière fonction de ce programme est la vérification si une des trois valeurs est supérieure à 50 (valeur au-delà de laquelle les moteurs commencent à tourner, l'énergie peut commencer à vaincre l'inertie des moteurs). En cas de succès, D2 est passée à l'état haut, et en cas d'échec à l'état bas.
De cette manière une led s'allumera en façade de la balle si les moteurs vibrent, et s'éteindra si ils ne vibrent pas.

ii) Le module Bluetooth

Le module Bluetooth n'était pas pré-programmé lorsque nous l'avons reçu. Cela se manifestait par une absence de réponse de sa part. En effet, il n'était pas détecté par les périphériques Bluetooth, et n'allumait aucune LED. En revanche il n'y avait pas de signe de dommages, de mauvaise soudure ou de surchauffe.

Nous avons donc essayé de le reprogrammer via le port SWD d'une carte Nucleo F401RE. Dans un premier temps via le logiciel OpenOCD via lignes de commande.

Nous avions à notre disposition tout le SDK du MDBT40, ce qui nous donne l'accès à des programmes d'exemple tels que blinky ou uart_test.

Nous avons réussi à faire fonctionner le programme précompilé blinky fourni dans le SDK, faisant juste clignoter une LED. Ceci nous indiquant que le module n'était pas endommagé.

En revanche, le programme précompilé faisant fonctionner le Bluetooth restait non fonctionnel.

Le problème venait de l'horloge. En effet, le nRF51 qui équipe le MDBT40 peut fonctionner correctement sans horloge externe. L'horloge interne avait donc été correctement configurée, comme le prouve le fait que le programme blinky clignotait à la bonne fréquence.

En revanche, le softdevice, responsable de la communication Bluetooth se devait d'être configuré aussi.

Pour ce faire, nous avons utilisé le logiciel Keil qui permet de reprendre les sources du SDK du module Bluetooth, ainsi que les exemples afin de les modifier et de les recompiler.
Nous avons donc ouvert l'exemple fourni permettant de configurer une liaison UART. Dans la fonction ble_stack_init du main.c, les lignes suivantes devaient être présentes :

```
nrf_clock_lf_cfg_t clock_lf_cfg;
 clock_lf_cfg.source = NRF_CLOCK_LF_SRC_RC;
 clock_lf_cfg.rc_ctiv = 16;
 clock_lf_cfg.rc_temp_ctiv = 2;
 clock_lf_cfg.xtal_accuracy = 0;

 // Initialize SoftDevice.
 SOFTDEVICE_HANDLER_APPSH_INIT(&clock_lf_cfg, true).
```

Une fois cette correction apportée, nous avons pu accéder à notre module Bluetooth via un périphérique android pour la première fois.

Le programme de communication UART fonctionnait correctement et nous permettait de communiquer avec l'Atmega328P, c'est pourquoi les modifications que nous lui avons apporté sont peu nombreuses.

Nous avons changé le nom du module trop générique par [IMA4] Prj15.

Nous avons également configuré le module pour que la led connectée sur le GPIO 18 s'allume lorsqu'un périphérique est connecté et s'éteigne le cas échéant. Pour cela il suffisait d'initialiser la led dans le fichier pca10028.h de keil, et d'utiliser les fonctions bsp_board_led_off et bsp_board_led_on lorsqu'un événement de connexion et de déconnexion était lancé.

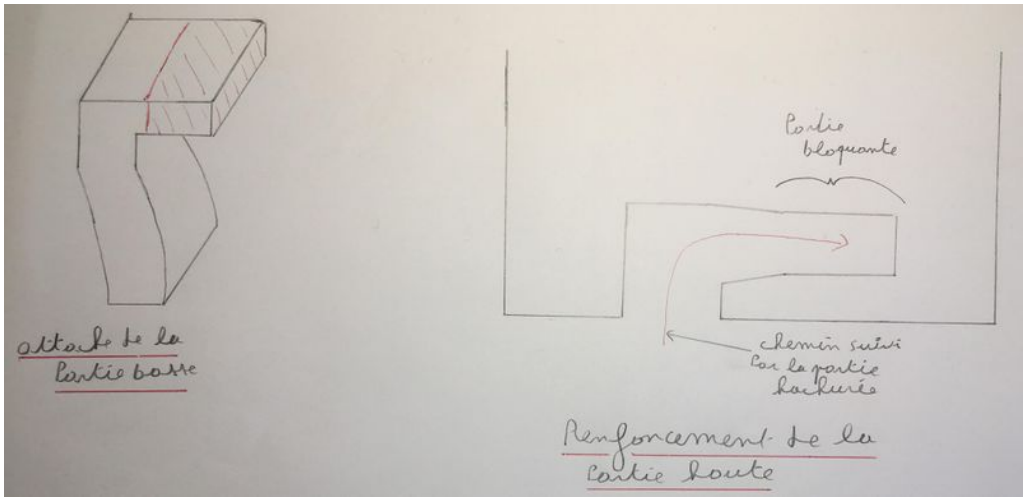
5) Création du modèle 3D

i) Définition du besoin :

Pour réaliser le modèle 3D, nous avons dû au préalable déterminer à quels critères il devait répondre, à savoir :

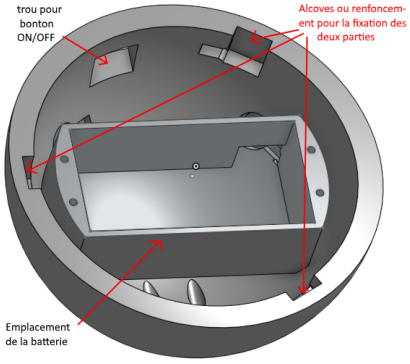
- Avoir un diamètre de moins de 15 cm pour que l'enfant ou le praticien puisse la manipuler avec aisance.
- Fixer la carte électronique et la batterie de manière à ce que ces dernières ne bougent pas lors des manipulations de la balle vibrante.
- Avoir des emplacements pour pouvoir poser des vibreurs (6 au total).
- Avoir des trous pour pouvoir placer des LEDs et un interrupteur.
- Avoir une base plate pour pouvoir poser la bobine à inductance.

En fonction de toutes ces contraintes, nous avons pu modéliser sur OnShape notre premier modèle. Le principal intérêt de logiciels de CAO comme OnShape réside dans le fait qu'il est possible d'assembler toutes les parties afin de tester si tout se place comme nous le souhaitions. La réalisation de l'assemblage ne nous a néanmoins pas épargné le fait de devoir faire deux prototypes car le premier avait un système de fixation des parties hautes et basses de la balle beaucoup trop fragile. Nous avons donc opté pour un système de fixation par coulissement. Le principe est d'avoir des embouts mâles dans la partie basse qui viendront se fixer dans les embouts femelles de la partie haute. Le principe est détaillé dans le croquis ci-dessous :

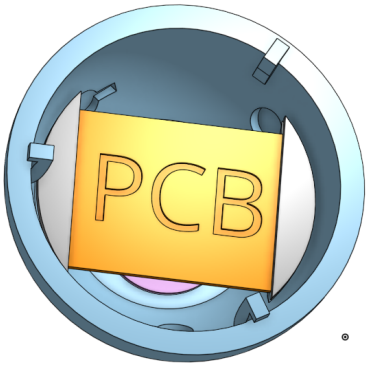


Description du système d'accroche des deux parties

Voici à quoi ressemble notre modèle :

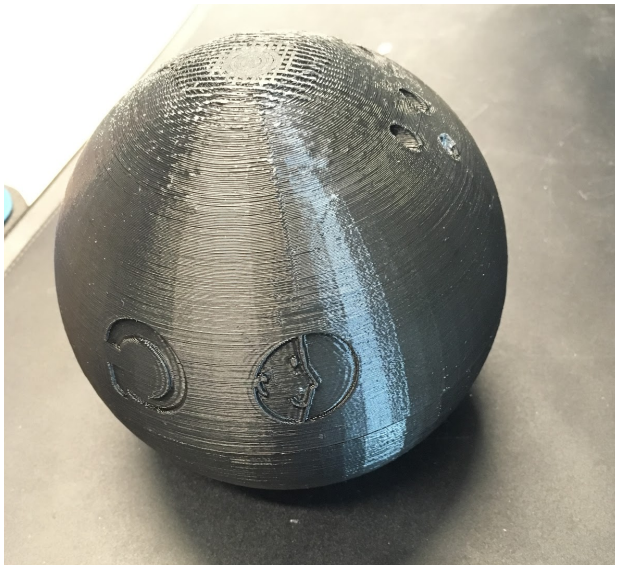


Partie basse



Partie haute

Nous avons ensuite imprimé nos différentes parties puis les avons assemblées. Comme expliqué précédemment, la première impression fut un échec car les clips étaient trop fragiles, mais la seconde fut une réussite. Tout s'est emboîté comme prévu entre la partie haute et la partie basse. Le système de fixation devant prendre en tenaille le PCB quant à lui n'a pas vraiment rempli les exigences. Nous avions prévu des trous dans lesquels des tétons devaient venir se coincer, mais ces derniers trop fragiles se sont cassés. Nous avons une seule des deux attaches du PCB qui a tenu le choc, pour remplacer l'autre, nous avons dû utiliser une vis et un boulon qui ont fait pression et ont maintenu le PCB en place.



Balle imprimée et assemblée

6)Assemblage et finalisation du produit

La balle a été conçue afin de garantir une facilité de démontage/remontage dans le but de faciliter son développement futur. Pour ce faire, nous avons fait le choix de conserver sur la carte les headers et de souder sur tous les périphériques des connecteurs Dupont femelles. De plus, certains fils soudés sur la carte n'ont pas été coupés, c'est le cas :

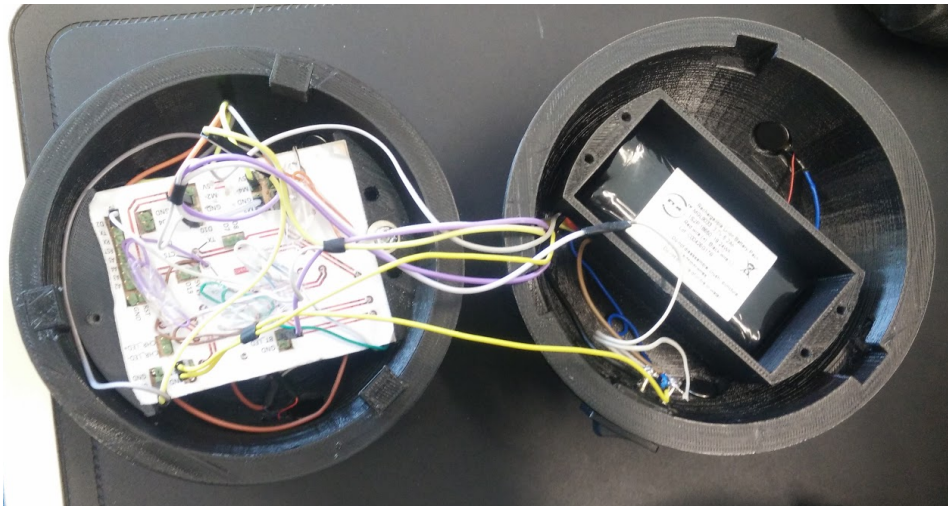
-Des fils sur les broches D11 et D12 de l'arduino (afin de pouvoir le programmer avec un arduino en ISP).

-Des fils sur les GPIO configurés en RX et TX du module Bluetooth (afin de pouvoir visualiser les communications entre le module Bluetooth et l'arduino).

-Des fils sur les broches data et clock du module Bluetooth (afin de pouvoir programmer le module en SWD).

Dans une future version du PCB, ces fils devront être remplacés par des headers.

Nous avons donc soudé sur les connecteurs Dupont les 3 leds, les 6 moteurs, l'interrupteur (directement sur la masse de la batterie), le + de la batterie, et la bobine de recharge sans fil.



Balle entièrement assemblée et ouverte

Comme expliqué plus haut, vous pouvez constater sur la boule gauche que nous avons dû recourir à un peu de débrouille afin de pouvoir faire tenir le PCB dans la balle. Nous nous sommes servis d'une vis et d'une rondelle pour faire pression sur le PCB et le maintenir en place.

Les câbles connectés à un moteur ont été rassemblés avec une gaine thermorétractable, dans un souci de clarté lors des branchements. Idem pour les câbles connectés aux leds et à la batterie. De plus, leur longueur est suffisamment longue pour pouvoir poser les deux demi-sphères sur une table sans rien débrancher, mais également assez courte pour ne pas venir se prendre dans les fixations lors de la fermeture.

III) Limites du projet

Comme nous avons pu le constater tout au long de ce rapport, nous avons pu répondre au cahier des charges et délivrer un produit fini et fonctionnel dans le temps qui nous était imparti. Néanmoins, la balle comme l'application ne sont pas du tout parfaites, et il reste de nombreuses améliorations à apporter à l'ensemble, notamment :

- Lorsque l'application est connectée à la balle et que cette dernière s'éteint puis se rallume, l'application n'est pas capable de se reconnecter automatiquement à la balle, il faut éteindre puis rallumer manuellement l'application. Nous n'avons pas eu le temps de nous pencher assez sur ce problème pour le résoudre.

- Le système de détection de voix fonctionne, mais lorsqu'il y a du bruit autour, la balle est facilement parasitée et vibre alors que l'utilisateur n'a pas forcément émis de son, il conviendrait de régler également ce problème. Il sera nécessaire d'affiner le système de détection de son.

- Au niveau de l'application, le mode suivi des patients n'a pas été codé par manque d'informations. Cela apporterait un vrai plus (mais demanderait également un travail conséquent en base de données notamment).

- Nous pourrions encore réduire de beaucoup la taille de la balle ce qui impliquerait une prise en main plus conviviale, en commençant par réaliser une nouvelle carte qui serait gravée en TOP et en Bottom, ce qui réduirait potentiellement la surface par deux. Sur notre première réalisation, les gravures en bottom sont restées marginales. Créer un PCB qui serait rond serait aussi une possibilité pour réduire la taille de la balle.

- L'utilisation d'un Atmega328P s'est avérée inutile. En effet, notre module Bluetooth est un microcontrôleur ARM, programmable et doté d'entrées et de sorties digitales, de sorties PWM et même d'un ADC. Il aurait pu entièrement remplacer l'Atmega328P, mais nous nous sommes rendu compte de son potentiel lors des recherches sur comment le programmer, le projet était donc déjà trop avancé pour être modifié.

Conclusion

Pour conclure, nous pouvons dire que la balle ainsi que l'application sont toutes deux fonctionnelles. Nous avons donc su répondre aux exigences qui nous étaient demandées et ce dans le temps imparti. Nous espérons que ce que nous avons produit est à la hauteur de ce que l'on peut attendre d'un travail d'étudiant ingénieur en fin de quatrième année. Nous avons pris un grand plaisir à créer ce produit. La pluridisciplinarité du projet a également été grandement appréciée (nous avons pu faire de la mécanique, de l'électronique, de l'informatique…) et nous a permis d'apprendre énormément. Nous espérons que nos travaux seront une bonne base ne demandant qu'à être améliorée afin d'aboutir à un produit fini pouvant servir au CAMSP et à la rééducation des enfants.

Annexes

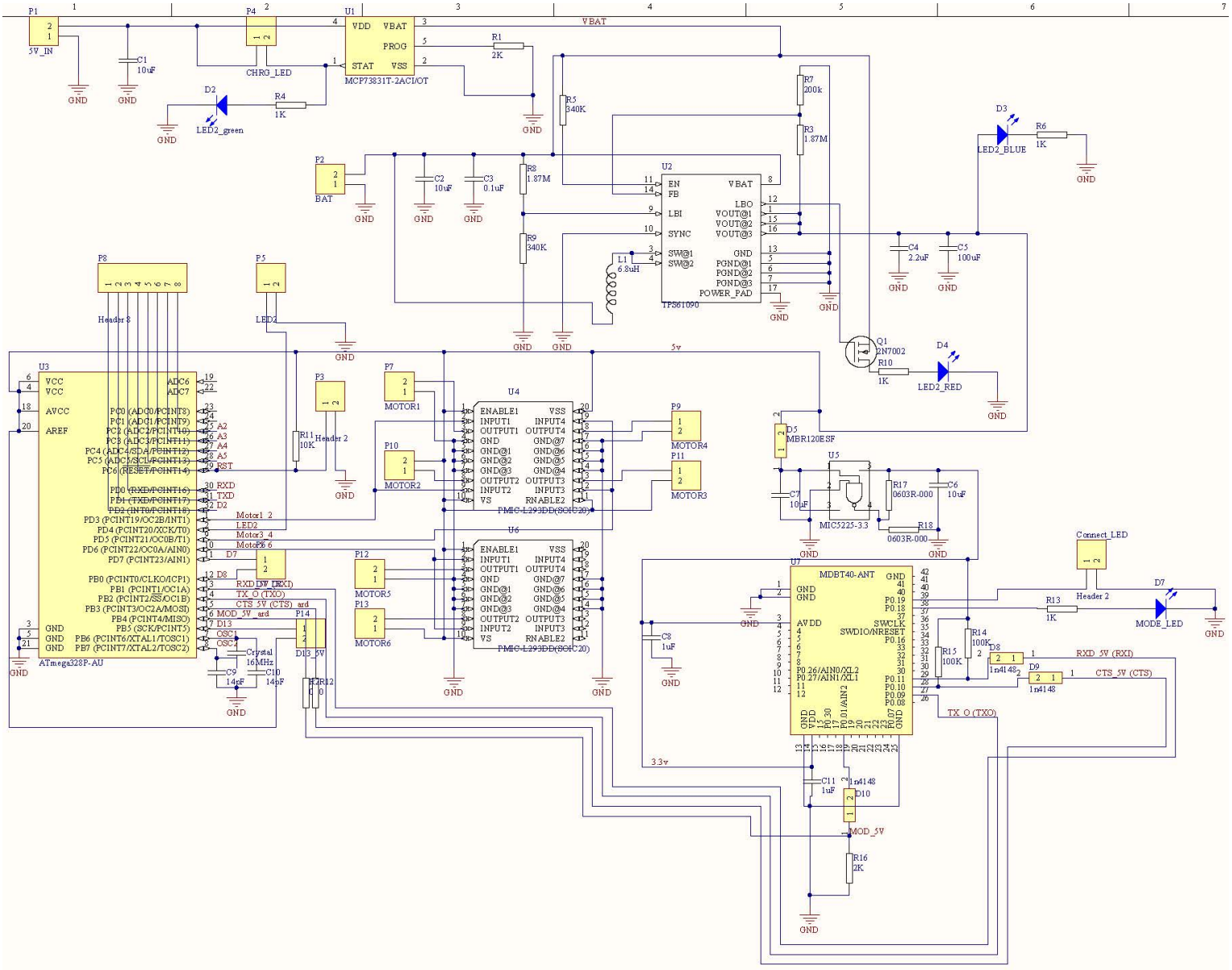


Schéma 1

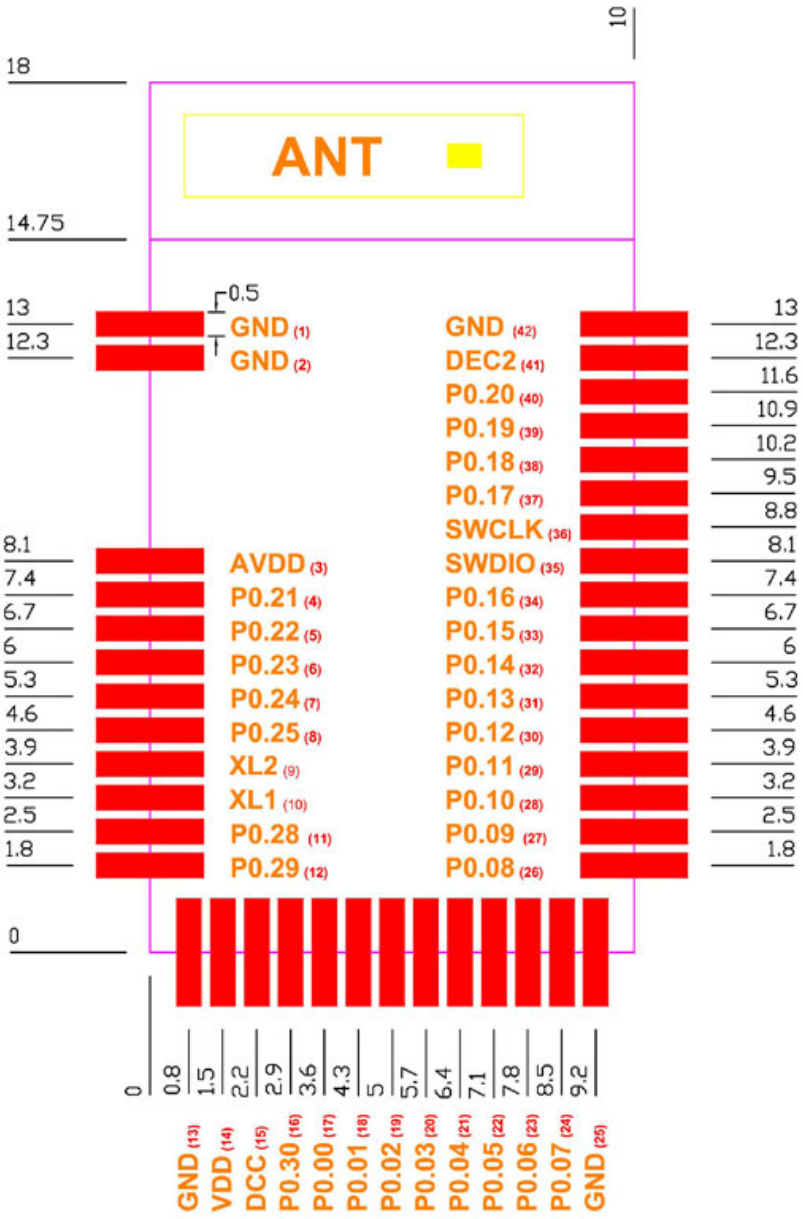

ANT
GND (1)
GND (2)
AVDD (3)
P0.21 (4)
P0.22 (5)
P0.23 (6)
P0.24 (7)
P0.25 (8)
XL2 (9)
XL1 (10)
P0.28 (11)
P0.29 (12)
GND (42)
DEC2 (41)
P0.20 (40)
P0.19 (39)
P0.18 (38)
P0.17 (37)
SWCLK (36)
SWDIO (35)
P0.16 (34)
P0.15 (33)
P0.14 (32)
P0.13 (31)
P0.12 (30)
P0.11 (29)
P0.10 (28)
P0.09 (27)
P0.08 (26)
GND (13)
VDD (14)
DCC (15)
P0.30 (16)
P0.00 (17)
P0.01 (18)
P0.02 (19)
P0.03 (20)
P0.04 (21)
P0.05 (22)
P0.06 (23)
P0.07 (24)
GND (25)

Schéma 2

CAHIER DES CHARGES PRODUIT

TABLE DES MATIÈRES

1. DESCRIPTION GENERALE

1.1 Objet du programme de R&D

L'objet sera une balle capable de vibrer lorsqu'un son est émis. Cette vibration doit être plus ou moins puissante et réglable. Des leds seront également présentes afin d'afficher lorsque la balle vibre, est connectée ou alors est en recharge. Il y aura également un bouton ON/OFF pour allumer ou éteindre la balle.

1.2 Produits associés

Une application pour smartphone Android est associée à la balle.

2. FONCTIONS À RÉALISER

2.1 Fonctions générales du produit

- L'application doit utiliser le microphone du smartphone pour capter le son, l'application enverra ensuite par Bluetooth une trame ordonnant la vibration à une certaine intensité (suivant le niveau de son capté) de la balle.
- L'application doit être facile de prise en main, conviviale et fonctionnelle.
- L'application doit être équipée d'un mode permettant un suivi des patients.
- L'application doit être équipée d'un mode permettant un réglage de l'intensité de la vibration.
- La balle doit être équipée de leds s'allumant lorsque la balle est connectée par Bluetooth au smartphone, lorsqu'elle vibre et lorsqu'elle se recharge.
- La balle doit être équipée d'un bouton ON/OFF.

2.2 Interfaces Produit

La connexion entre la balle et l'application Android doit se faire via Bluetooth.

3. ENVIRONNEMENT DU PRODUIT

3.1 Contrainte d'environnement

La balle doit être partiellement étanche (résistante à la bave d'un enfant).

3.2 Couleur

Aucune spécification de couleur n'a été faite.

4. PERFORMANCES DU PRODUIT

4.1 Capacité / Endurances

La balle fonctionnera sur batterie et devra avoir une autonomie de 100 minutes minimum (calculée sur la base de 10 minutes d'utilisation par heure et d'une journée d'utilisation).

4.2 Autres performances

5. MISE EN ŒUVRE DU PRODUIT

5.1 Installation mécanique

La balle doit être assez résistante et pouvoir être démontable à souhait.

5.2 Installation électrique

Pas d'installation électrique particulière.

6. CONTRAINTES DE RÉALISATION

6.1 Contraintes de taille et de poids

La balle doit avoir un diamètre extérieur de 15 cm maximum et un poids suffisamment faible pour qu'un enfant puisse la tenir (soit moins de 1kg).

7. CONTRAINTES DE SÉCURITÉ

7.1 Contraintes de sécurité

- La balle ne doit pas chauffer.
- Aucune partie ne doit être détachable pour la sécurité de l'enfant (risque d'ingérer des composants).