PROJETS DEUXIEME ANNEE : JALON 3/ S8

Date : 06/06/2017 Numéro de Projet : **P5C-404**

Titre du Projet : **Spectacle de magie**

Encadrant du projet : Laurent Cabaret (*Remplaçant Hanane Meliani)*

Nom des élèves :

Gael Colas

Sylvestre Prabakaran

Sommaire

0. Livrables

Notre projet se centre sur l'application d'un point de vue d'ingénieur à la conception et à la réalisation d'un tour de magie technologique : « Color Pen Prediction ». Tout ce projet a été réalisé en suivant notre « check-list de conception d'un tour de magie » du point de vue d'un ingénieur. Ce projet a permis de valider cette check-list.

Nous avons imaginé une façon de réaliser cet effet de mentalisme, sans complice. Ceci étant uniquement rendu possible grâce à l'utilisation des technologies actuelles tirant partie de connaissances variées :

- Systèmes embarqués ;
- Programmation : Arduino et Android ;
- Micro-électronique ;
- Modélisation 3D.

Notre livrable principal est donc le **dispositif lui-même**, permettant de réaliser le tour :

- **Le circuit électronique** ;
- **Le pot** imprimé en 3D dissimulant ce circuit électronique ;
- **L'application Android** associée.

Mais comme notre projet s'intéressait à la démarche d'ingénieur appliqué à la conception d'un tour de magie, notre tour s'accompagne de toute une documentation qui permet de le reproduire dans d'autres conditions, de l'utiliser et de le généraliser.

Les livrables complémentaires pour le tour lui-même sont :

- Les **plans** du dispositif : sous format SpaceClaim, ils fournissent le support de base pour réaliser un pot en impression 3D ;
- Un **manuel d'utilisation** : il explique comment utiliser le dispositif. Le manuel donne notamment la liste des actions que doivent effectuer le magicien pour que le tour soit réussi.

Les livrables associés au circuit électronique sont les suivants :

- Le **fichier Altium** : ce fichier permet d'imprimer la carte du circuit imprimé ;
- Les **data sheets** des différents composant utilisés ;
- Le **code Arduino** : c'est celui qui commande la détection des feutres et la transmission Bluetooth de l'information ;
- La **bibliothèque Hx711** : cette bibliothèque Arduino modifiée permet d'utiliser le capteur de force avec la carte Radon ;
- Le **code Android** : c'est celui qui commande le traitement de l'information envoyée par l'Arduino et sa conversion en code tactile.

Tous ces livrables sont disponibles dans le fichier compressé « CPP.zip ».

Certains de ces documents sont également fournis en « Annexe ».

1. Choix de l'effet magique

L'effet magique principal que nous voulons présenter est : « prédire le futur ». Cet effet défie le principe de causalité : on ne peut prévoir l'effet d'une action alors que les causes n'ont pas encore eu lieu.

Le tour que nous souhaitons présenter est un tour de mentalisme : ce tour permet de prédire exactement à l'avance le coloriage d'un dessin en *n* couleurs. Chaque couleur étant affectée à une zone du dessin.

Du point de vue du spectateur, le tour est impossible grâce aux 2 informations-clés indispensables :

1) Le choix du coloriage est « libre » ;
2) Le magicien n'a pas connaissance des couleurs choisies avant la fin du tour.

Dans ces conditions, le magicien aurait une probabilité d'avoir la bonne combinaison de couleurs :

$$P = \frac{1}{n!}$$

Pour donner cette impression, notre tour doit donc permettre au magicien de réaliser seul l'effet magique tout en remplissant les exigences associées :

1) Le choix des feutres est libre ;
2) Le magicien ne peut voir les couleurs choisies avant la fin du tour.

2. Recherche sur l'Etat de l'Art

Monde de la magie

Notre projet doit permettre de reproduire un tour existant grâce à l'utilisation d'un système embarqué. Ce tour est un tour de mentalisme, il s'appelle : « Color Pen Prediction ». Il utilise un gimmick à 200€ : http://www.paris-magic.com/magasin-magie-paris-magic-mentalisme-color-pen-prediction-2-0.htm

Notre dispositif doit permettre de réaliser le tour présenté au show américain « The Ellen DeGeneres Show » par le magicien Justin Willman.

Référence : https://www.youtube.com/watch?v=i2_MQekyIXk

Recherches sur l'électronique

Pour tous les systèmes de détection, Il faut tout de même ajouter au dispositif une carte pour commander les systèmes de détection et de communication munie d'une alimentation adaptée.

La solution technique envisagée doit a priori répondre à des problèmes d'encombrement. C'est pourquoi nous allons utiliser des cartes électroniques de faible encombrement. Nous avons pensé à deux solutions possibles :

1) TEENSY LC

Dimensions : 18x36 mm
Nombre d'entrées et sorties : 34
Masse : 7 g
Tension d'opération : 3.3 V
Fréquence d'horloge : 48 MHz
Mémoire Flash : 16 K

Pour les premiers tests, nous avons opté pour l'utilisation de la Teensy LC, recommandée par notre encadrant Mr Cabaret, qui présente un double-avantage : plus faible encombrement et meilleures performances.

Pour apprendre à utiliser et à programmer la carte Teensy, nous avons suivi la formation sur GitLab de notre encadrant : https://gitlab.centralesupelec.fr/2005cabaretl/FormationArduinoTeensy

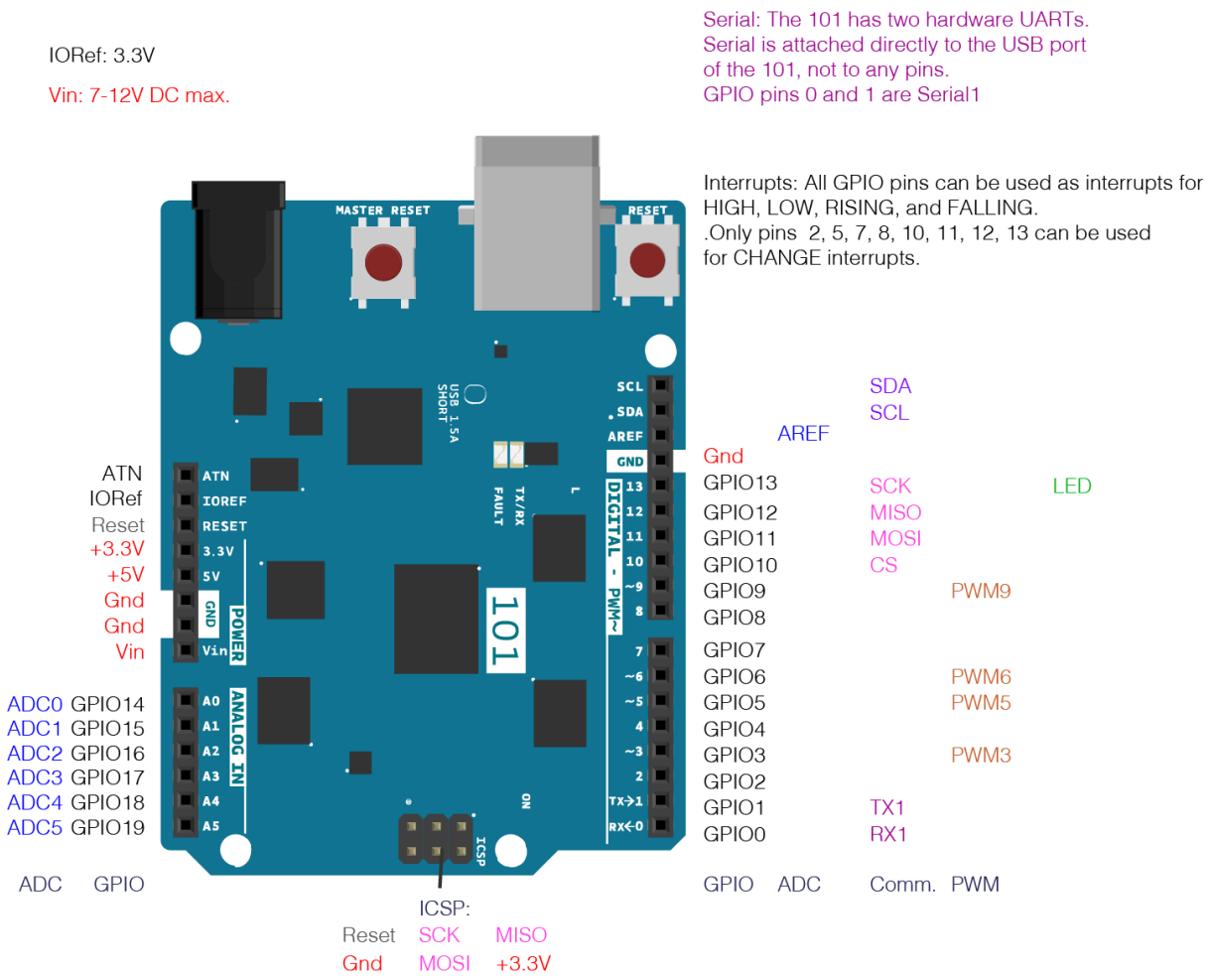
2) Arduino RADON

Dimensions : 18x36 mm
Nombre d'entrées et sorties : 26
Masse : 10 g
Tension d'opération : 3.3 V / 5V
Fréquence d'horloge : 48 MHz
Mémoire Flash : 16 K

Cette carte Arduino n'est pas encore disponible sur le marché. Mais elle possède globalement les caractéristiques de sa grande sœur *la Genuino 101 Intel Curie*.
Pour programmer la Radon, il faut donc :

- Utiliser les bibliothèques compatibles avec la *Genuino 101* ;
- Sélectionner « Arduino/Genuino 101 » dans l'IDE.

Voici le détail des différents pins de la Genuino 101 et par extension de la Radon :



Arduino 101 Pinout Diagram

C'est cette carte que nous avons retenu pour 2 raisons :

- Performance : elle possède le nombre de pin adapté pour utiliser le capteur de force ;
- Encombrement : elle est de taille très réduite et possède l'immense avantage de posséder un module Bluetooth intégrer ce qui limite encore l'encombrement.

Bibliothèques utilisées

Les bibliothèques sont à décompresser dans le dossier :
C:\Users\YourName\Documents\Arduino\libraries

- Hx711

Il s'agit de la bibliothèque permettant d'utiliser le capteur de force.

Cette bibliothèque a été développée par un contributeur de la communauté Arduino. La fonction principale :

getGram()

permet d'obtenir directement des résultats en grammes à partir des valeurs de résistance envoyées par le capteur.

bibliothèque bien qu'utilisable telle quelle pour nos premiers tests sur la Teensy LC a dû être modifiée et adaptée pour être compatible avec la Radon. Cette bibliothèque « Hx711 » modifiée est fournie dans nos livrables.

- CurieBLE

Il s'agit de la bibliothèque permettant d'utiliser le module Bluetooth intégré à la carte Radon.

Cette bibliothèque a été développée par Intel pour leur carte Arduino Genuino 101 Intel Curie possédant un tel module Bluetooth.

Cette bibliothèque n'a pas été modifiée et est installée automatiquement lors de l'installation du plugin « Arduino Genuino 101 Intel Curie » sur l'IDE.

Systèmes embarqués

Etude comparative des systèmes de détection :

Système de détection	Détection directe	Description	Fonctionnement	Encombrement		Discrétion	Rapidité	Fiabilité	Coût
				Pot à feutres	Feutre				
Complice	Oui	Un complice se trouve proche de la scène avec vue sur le pot de feutres	1) Le complice voit la couleur choisie par le spectateur ; 2) Il code la couleur ; 3) Il envoie le code au magicien par téléphone.	Nul : on utilise un pot normal	Nul : on utilise des feutres normaux	Faible : l'intensité des vibrations d'un téléphone est difficilement maîtrisable. Elles pourraient être entendues par le spectateur.	Lent : le complice doit coder lui-même la couleur puis l'envoyer par téléphone	Dépendante : du complice qui doit être capable de voir et de coder rapidement la couleur	Nul (Sauf si le complice est rémunéré)
Bouton poussoir	Oui	Le pot est décomposé en compartiments individuels. Dans chaque compartiment, un feutre est posé sur un bouton poussoir (de type languette métallique).	1) Le spectateur prend un feutre 2) La languette se soulève 3) Le contact électrique ne se fait plus 4) Le feutre est détecté	Important : a priori, les boutons poussoirs peuvent être très petits, mais il faut autant de compartiments que de feutres	Nul : on utilise des feutres normaux	Forte : le pot opaque dissimule le dispositif électronique complètement	Rapide : calculé et envoyé quasi-instantanément par le dispositif	Moyenne : au cours du temps, la languette pourrait se déformer sous le poids du feutre et le contact se faire même une fois le feutre ôté.	Faible : les boutons poussoirs sont peu chers
Capteur lumineux	Oui	Le pot est décomposé en compartiments individuels. Dans chaque compartiment, une LED éclaire un capteur lumineux placé en face, capteur dissimulé quand le feutre est inséré.	1) Le spectateur prend un feutre 2) Le capteur dissimulé par le feutre est maintenant éclairé par la LED 3) Le feutre est détecté	Important : chaque compartiment doit recevoir une LED et un capteur lumineux	Nul : on utilise des feutres normaux	Moyenne : la lumière des LED pourrait être détectée par le spectateur	Rapide : calculé et envoyé quasi-instantanément par le dispositif	Moyenne : selon la position, le feutre pourrait ne dissimuler que partiellement le capteur, celui-ci pourrait donc envoyer des feux-positifs	Faible : les LED et les capteurs lumineux sont peu chers
RFID	Oui	Le pot possèderait un lecteur RFID. Chaque feutre possèderait sa puce passive RFID unique : son marqueur. La détection se fait de la manière suivante : le lecteur envoie un signal, le marqueur déforme le signal, la déformation permet de détecter de quel marqueur il s'agit.	1) Le spectateur prend un feutre 2) La puce RFID du feutre est hors de portée du lecteur 3) Le feutre est détecté	Moyen : le capteur RFID peut être fin	Faible : les puces RFID font quelques mm, les étiquettes RFID sont plates	Forte : le pot opaque dissimule le dispositif électronique complètement	Rapide : calculé et envoyé quasi-instantanément par le dispositif	Moyenne : risque de collision entre les différents marqueurs (communication brouillée par l'activité simultanée des marqueurs)	Moyen : - Lecteur RFID : 20€ - Puce RFID : 5€ http://www.robotshop.com/ca/fr/pieces-robot-rfid.html?p=2
Mesure du poids des feutres	Non	Le pot de feutres serait une balance de précision qui mesurerait le poids total des feutres. Chaque feutre aurait un poids légèrement différent.	On pourrait détecter quel feutre est ôté de son pot en calculant simplement la différence entre le poids précédent et le poids actuel.	Moyen : le capteur de pression peut être fin	Très faible : il faut juste modifier le poids des feutres	Moyenne : le spectateur peut remarquer que tous les feutres ne sont pas faits de la même façon à cause de leur différence de poids.	Rapide : calculé et envoyé quasi-instantanément par le dispositif	Dépendante : de la précision de la balance (de l'ordre de 0.01 g)	Faible : environ 20€ pour une balance de précision Prix capteur résistif : 15€

3. Sélection de la solution technique

Pour répondre aux exigences de l'étape 1, notre dispositif doit donc remplir 3 fonctions distinctes :

1) Récupérer l'information : le dispositif doit détecter de façon directe ou indirecte quel est le feutre qui vient d'être pris par le spectateur.
   Ainsi, le choix du feutre peut être laissé libre : l'exigence 1 est respectée.

Méthode directe : le dispositif détecte la présence/absence de tous les feutres individuellement.

Méthode indirecte : le dispositif détecte la présence du groupe de feutres présent. Lorsque cette information est modifiée (un feutre est pris), il détermine quel feutre a été pris par soustraction entre l'état précédent et l'état actuel.

2) Transmettre l'information au magicien : le dispositif doit communiquer au magicien le feutre qui d'être pris par le spectateur.
   Ainsi, le magicien n'a pas besoin de prendre connaissance directement par lui-même de la couleur choisie : l'exigence 2 est respectée.

3) Etre indétectable : le dispositif ne doit pas être visible du spectateur ; problème d'encombrement : miniaturisation du dispositif dans le pot et les feutres.
   Cette fonction va de soit pour un dispositif magique : s'il est détecté par le spectateur, il n'y a pas d'effet magique.

On peut donc décomposer notre dispositif électronique en 3 parties, chacune devant être indétectable :

1) Détection

L'étude comparative des différents systèmes de détection présentée dans la partie précédente nous permet de choisir le système le plus adapté à tester :

- Le capteur résistif : 1 **capteur de force** est nécessaire ;

Un pot pesant : 100g ;
Un feutre pesant : 15g ;

Il nous faut un capteur possédant les caractéristiques suivantes :
- Plage de mesure : [0g ; 500g] ;
- Précision : 0.5 g.

2) Transmission

Pour transmettre l'information, on pourrait utiliser :

- Un système wifi ;
- Un système Bluetooth.

Nous avons choisi d'utiliser un module **Bluetooth** comme émetteur car le système Bluetooth est plus fiable sur la distance qui nous intéresse : environ 1m de distance entre l'émetteur (le pot) et le récepteur (le téléphone).

Quant au récepteur, il s'agira d'un téléphone sous Android (possédant évidemment un module Bluetooth).

3) Affichage

L'exigence 2 nous restreint quant aux dispositifs de communication utilisables. Puisque le magicien ne doit pas pouvoir prendre directement connaissances des couleurs de feutres choisis, on ne peut pas utiliser de communication :

- Visuelle : le magicien aura les yeux bandés ;
- Auditive : un système de code sonore pourrait être compris par des spectateurs dans la salle ; de plus, le magicien ne doit pas avoir d'oreillettes.

Le type de communication que nous allons utiliser est donc de type tactile : un buzzer placé sur le corps du magicien l'informera du feutre choisi grâce à un code simple.

Une **application Android**, développée par nos soins, s'occupera de traduire les données reçues par le module Bluetooth de la carte Arduino dans ce code tactile grâce au **vibreur** intégré au téléphone.

Comme chacune de ces parties devra être indétectable du spectateur, l'étape finale de notre projet sera :

- **Intégration**

Il nous faudra tout d'abord miniaturiser tout le circuit électronique en concevant un circuit imprimé connectant sur une même plaque : la carte Radon, son alimentation, le capteur de force et son convertisseur A/N. La réalisation de la carte se fera grâce au logiciel de PCB : *Altium.*

Enfin, il nous faudra modéliser en 3D sous SpaceClaim le pot afin de pouvoir l'imprimer ensuite en 3D. Le socle du pot devra remplir les fonctions suivantes :

- **Dissimuler** le circuit électronique ;
- **Laisser accessible** : l'alimentation (pour pouvoir la changer), le « Master Reset Button » (si la carte doit être manuellement réinitialisée), la prise microUSB (si un nouveau programme doit être téléversé).
- **Etre amovible** : s'il faut réparer le circuit électronique.

Code retenu

Le code que nous avons élaboré est basé sur le principe du Morse pour qu'il soit à la fois simple et rapidement déchiffrable. Dans ce code, il y aura 2 types de vibrations :

- Courtes : « 0 »
- Longues : « 1 »

Chaque couleur étant codé sur 2 caractères dans cet alphabet, on peut coder jusqu'à 6 couleurs. Nous n'en utiliserons que 5.

- Violet : 0
- Bleu : 1
- Vert : 00
- Jaune : 01
- Rouge : 10

Présentation du "mode dégradé"

Une contrainte importante de notre dispositif, comme nous l'avons souligné en introduction, est qu'il doit permettre de réaliser un **tour de magie de scène**. La représentation du tour n'est pas filmée, mais consiste bien en une présentation « en live » devant un public. Notre dispositif doit donc remplir les exigences associées telles que définis en introduction :

1) Le choix des feutres est libre ;
2) Le magicien ne peut voir les couleurs choisies avant la fin du tour.

Et cela même en cas d'aléas, inhérents à la performance en direct.

Notre système repose sur un dispositif électronique, que ce soit pour la détection ou la communication. La partie la plus sensible aux aléas est donc encore une fois la partie électronique : malgré des tests de fonctionnement juste avant d'entrer sur scène, il n'est pas inenvisageable que le dispositif électronique ne fonctionne pas une fois sur scène ; cela peut être dû à un câble coupé, une soudure qui lâche, ou un moteur qui rend l'âme. Pourtant il faut être en mesure de continuer la représentation, c'est pourquoi nous avons appliqué nos techniques d'ingénieur pour concevoir un « mode dégradé » de notre tour.

Ce « mode dégradé » permet de remplir les deux exigences précédentes, même en cas de défaillances de la partie électronique, avec une légère concession pour la deuxième exigence. En effet, notre mode dégradé utilise un complice caché dans les spectateurs ou dans un endroit proche de la scène.

Si le magicien lui-même ne voit pas les couleurs choisies, c'est le complice qui est chargé de les voir, puis de les transmettre au magicien. Pour ce qui est de la communication, on peut utiliser le vibreur du portable en remplacement du buzzer en conservant le code précédent :

1. Une vibration courte sera obtenue en envoyant un SMS au magicien ;
2. Une vibration longue en appelant le portable du magicien.

Dans ce mode dégradé, nous sommes donc en mesure de présenter notre tour en cas de défaillances de la partie électronique, et l'effet magique reste « réussi » même s'il est bien évidemment amoindri :

3. Le magicien ne peut plus réaliser ce tour seul ;
4. Ce système est plus lent : le complice doit faire lui-même toute la succession d'étapes détection de la couleur du feutre pris - codage de la couleur - envoi du code par téléphone. Le complice doit être très rapide dans ces actions pour que le rythme du tour ne soit pas suspect, entre le moment où le spectateur prend le feutre et le moment où le magicien lui dit quel élément colorier.

Mais ce mode nous permet de présenter un produit robuste dans toutes les conditions d'utilisations, ce qui est un critère important pour juger de la qualité d'un projet d'ingénieur.

4. Description précise du déroulement du tour

Déroulement du tour (point de vue du spectateur) :

1) Le magicien annonce qu'il a réalisé une prédiction qui se trouve dans une enveloppe scellée ;
2) Le magicien donne l'enveloppe à un spectateur qui la conserve jusqu'à la fin du tour (dans un endroit inaccessible du magicien) ;
3) Le magicien présente le matériel, apparemment banal, au spectateur :
    a. Un dessin en noir et blanc avec plusieurs éléments distincts à colorier
    b. Un pot de feutre de couleurs
4) Le magicien explique au spectateur qu'il va devoir colorier le dessin avec les feutres ;
5) Le magicien se bande les yeux ;
6) Le spectateur prend un feutre au hasard dans le pot ;
7) Le magicien lui dit de colorier un élément spécifique du dessin ;
8) Le spectateur colorie cet élément de cette couleur ;
9) Le spectateur se débarrasse du feutre (en le posant par terre par exemple ;
10) L'opération est répétée tant qu'il reste des feutres dans le pot et que le dessin n'est pas totalement colorié ;
11) Le magicien enlève le bandeau ;
12) Le magicien demande au spectateur de sortir la prédiction de l'enveloppe ;
13) La prédiction et le dessin du spectateur correspondent parfaitement au niveau du choix des couleurs.

5. Conception des plans techniques

La partie électronique

- **1ère partie : la détection**

Nous avons choisi comme système de détection **le capteur de force**. Celui-ci mesure le poids global du système : pot + feutres.

S'il détecte une variation **de ce poids supérieure à un seuil de**

$$\boldsymbol{mMin = 5\ g}$$

la carte Radon en déduit qu'un feutre a été enlevé du pot. Elle conserve en mémoire :

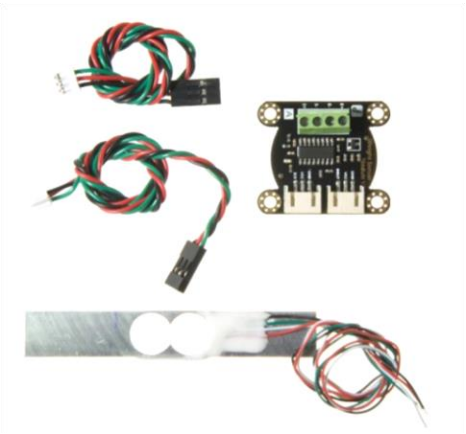
- L'ancienne valeur du poids global ;
- La nouvelle valeur du poids global.

Tous les feutres ont des poids différents : l'écart entre 2 poids de feutres est de l'ordre de 0.5g.

En faisant la différence entre l'ancienne valeur du poids global et la nouvelle valeur, elle en déduit quel feutre a été enlevé et donc quelle couleur va être utilisée pour le dessin.

Pour répondre aux caractéristiques présentées dans la partie précédente, voici le capteur de force que nous avons sélectionné :

ENSEMBLE DFRobot : http://www.gotronic.fr/art-capteur-de-force-1-kg-interface-sen0160-22707.htm#complte_desc



Capteur de force à pont de Wheatstone :
- Plage de valeur : [0g ; 1kg]

Fiche technique

CAN HX711 de précision à gain sélectionnable (32, 64 ou 128 bits)

Fiche technique

La précision de l'ensemble est de : 0.1g.

Ce système présente 2 avantages :
- Il répond aux exigences (plage de valeur + précision) définies dans la partie précédente ;
- Il est directement compatible avec la bibliothèque Arduino *Hx711*.

- **2ème partie : la transmission**

Pour transmettre les données de la carte Radon vers le téléphone du magicien, nous utilisons le module Bluetooth intégré à la Radon.

La bibliothèque Arduino *BLECurie* nous permet d'échanger des données entre les deux appareils selon le protocole Bluetooth. Une fois connecté à la carte Radon, le téléphone peut :

- *Read* : lire les données à sa disposition ;
- *Write* : envoyer des données à la Radon ;
- *Be Notify* : être averti quand une donnée change de valeur.

- **3ème partie : l'affichage**

Les données, une fois reçues par le téléphone, sont traitées par l'application Android et converties en code vibratoire.

- **Intégration**

Le logiciel de PCB Altium nous permet de concevoir numériquement la plaque de circuit imprimé qui accueillera et reliera tous les composants de notre système :

- La carte Arduino Radon : le microcontrôleur ;
- Son alimentation externe **:** une pile 9V ;
- Le capteur de force ;
- Son convertisseur A/N.

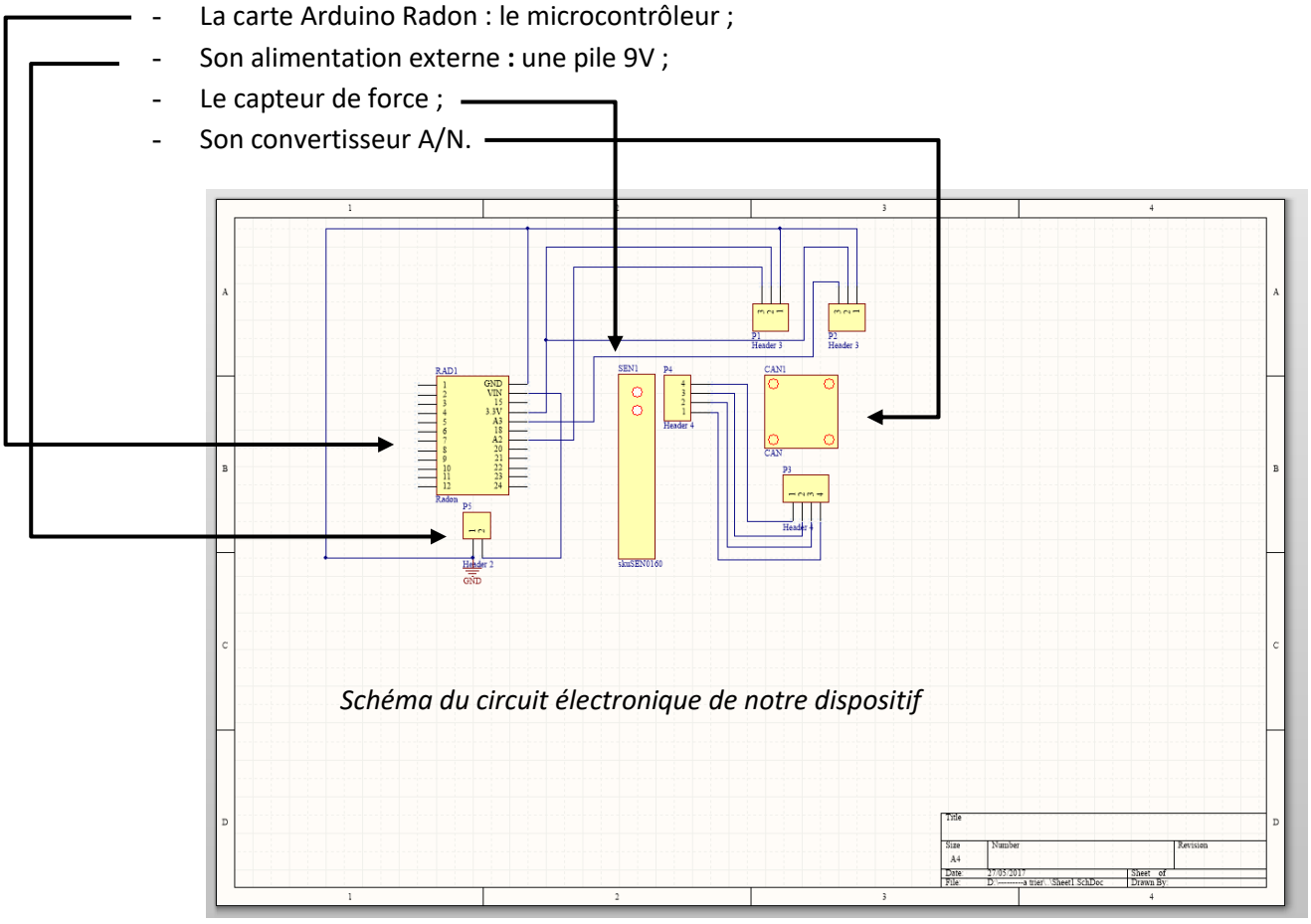
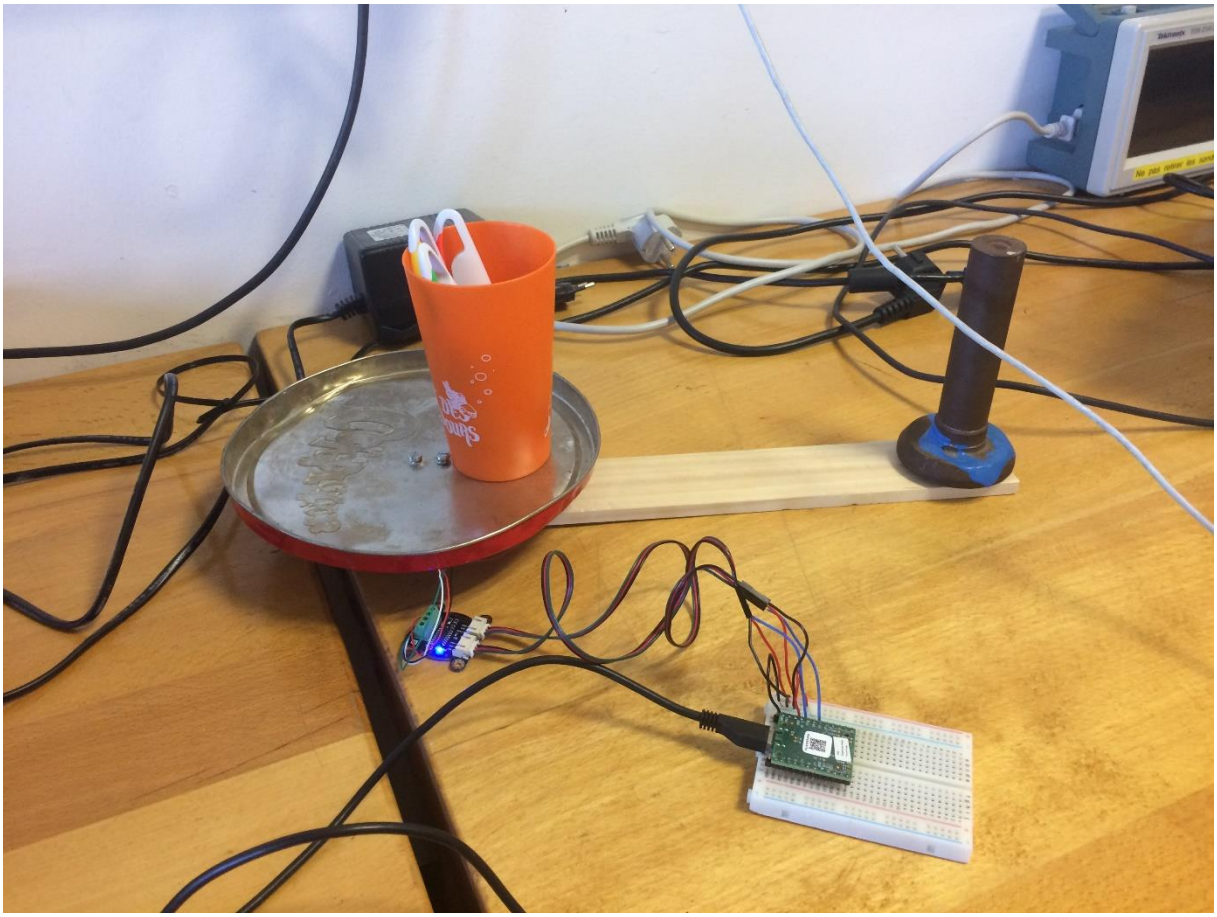


Schéma du circuit électronique de notre dispositif

Voici le prototype que nous avons utilisé pour réaliser :

- Les tests de validation de concept ;
- Les premiers tests de dimensionnement.



Photographie du prototype de dispositif de balance connectée

On y retrouve tous les éléments du circuit électronique décrit plus haut.

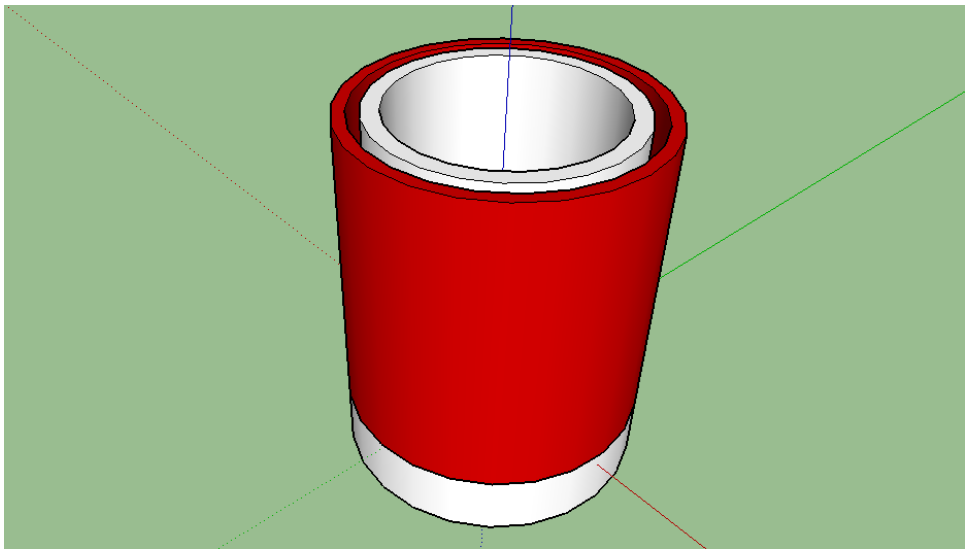
La partie mécanique

- **Le pot**

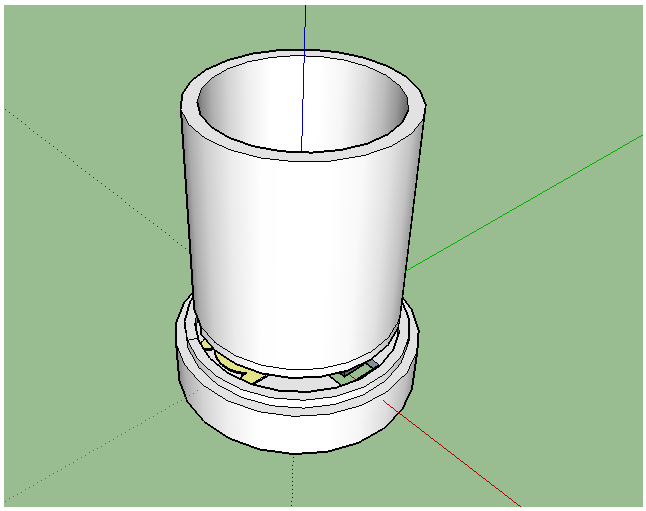
Pour recevoir la partie électronique, nous avons modélisé sous SketchUp un modèle de pot. Ce modèle pourra être ensuite converti sous SpaceClaim pour fournir les fichiers 3D nécessaires à une impression 3D.

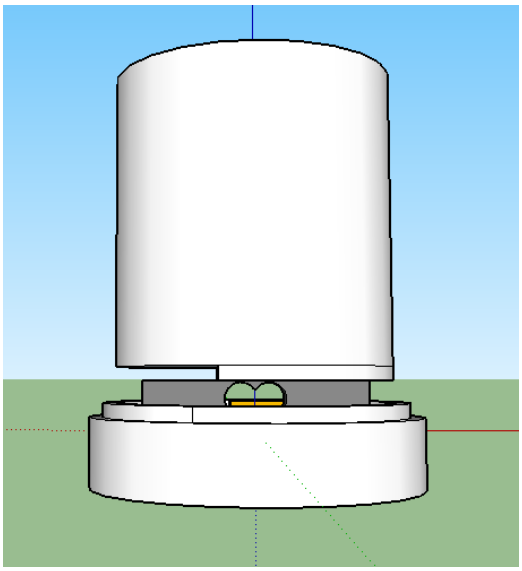
Notre pot se décompose en 3 parties :

1. Un grand pot fixe : il s'agit en fait d'un cylindre englobant toute la partie interne du pot (petit pot + base). Il permet de dissimuler aux yeux des spectateurs le système électronique interne et de donner au pot global un aspect normal ;



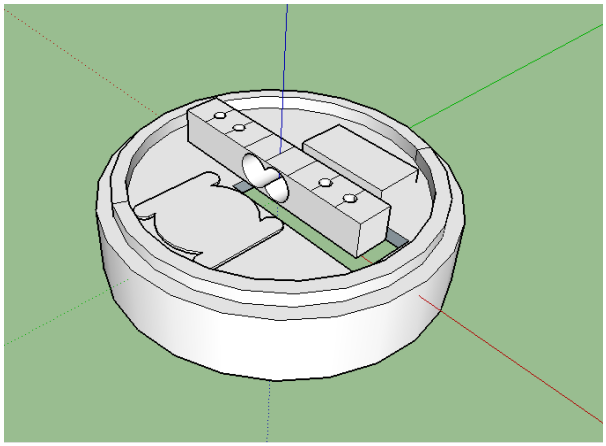
2. Un petit pot mobile : cette partie mobile qui reçoit l'ensemble des feutres est celle qui sera effectivement pesée par le capteur de force. Grâce à ce pot interne, les feutres n'interagissent à aucun moment avec la partie fixe. On s'assure ainsi que le poids total des feutres est effectivement mesuré.

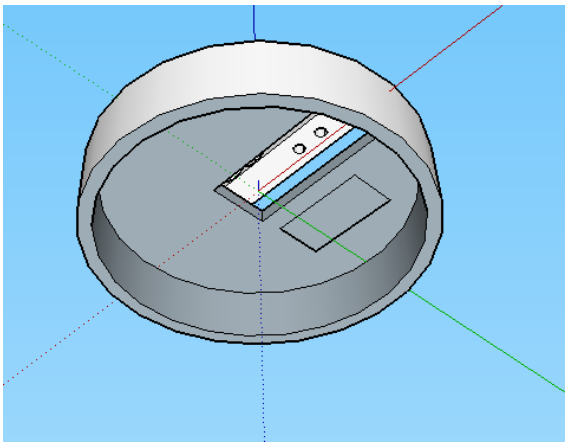




Ce pot interne repose sur la jauge de contrainte.

3. Une base (le fond du pot) : celle-ci doit contenir l'ensemble des composants du système électronique :
    - La carte Teensy ;
    - Le module wifi
    - L'ensemble capteur de force (jauge de contrainte + convertisseur éventuel)
    - La batterie





La base sert de support pour tous les composants électroniques. Au-dessus de la base, se trouve la jauge de contrainte avec son convertisseur et la plaque Teensy associée. (Voir figure à gauche). En dessous de la base, il reste de la place pour y installer la batterie ainsi que le module Wifi.

Contrainte : **miniaturisation**. Il faudra minimiser au maximum le volume de la base (ou donner l'impression que ce volume est très faible depuis l'extérieur) afin que l'on ne se doute pas qu'il ne s'agit pas d'un pot normal.

- **Les feutres**

Les feutres utilisés sont des feutres ordinaires : pour que le dispositif de détection fonctionne, il faut que ces feutres aient des masses différentes.

L'enjeu était que l'écart entre la différence de masse entre 2 feutres réponde à la double contrainte :

1) Etre « suffisamment grande » pour que le système attribue avec succès chaque différence de masse au bon feutre ;
2) Etre « suffisamment petite » pour que le spectateur ne remarque pas de différence de masse entre les feutres.

Pour quantifier numériquement ce que signifiaient « suffisamment grande » et « suffisamment petite », nous avons réalisé 2 expériences :

1) Avec le capteur

Nous avons mesuré plusieurs échantillons dont nous connaissions la masse avec notre capteur de force : en réalisant plusieurs charges et décharges.

Ces tests nous ont permis de confirmer que le capteur est précis à $\Delta m = \pm 0.1\ g$.

L'écart minimal entre la masse de 2 feutres doit donc être, avec une marge de sécurité, de :

$$delta = 0.5\ g$$

2) Avec le spectateur

Nous avons disposé une série de feutres d'aspect extérieur identique mais de masses différentes. Les yeux bandés, le spectateur devait discerner pour chaque paire quel était le feutre le plus lourd.

Cette expérience nous permet de conclure qu'un spectateur est incapable de distinguer une différence de masse de $\Delta m = 3\ g$ entre des feutres de masse moyenne $m_0 = 15\ g$: soit une différence de masse de 20%.

Conclusion

Pour des feutres de **masse moyenne $\boldsymbol{m_0 = 15\ g}$**, on peut donc utiliser jusqu'à

$$\mathbf{max = 6\ feutres}$$

pour que **l'écart minimal entre la masse de 2 feutres** soit d'au moins :

$$\boldsymbol{delta = 0.5\ g}$$

6. Réalisation du dispositif

Rédaction du code Arduino

Le détail du code est disponible en « Annexes ».

- **Paramètres**

Voici les différents paramètres utilisés dans le programme : ils ont été déterminés par expérimentation dans la partie précédente.

Paramètre	**Valeur**	**Signification**
max	6	Nombre maximal de feutres utilisables dans le tour
mMin (en g)	5	Borne inférieure des masses de feutres mesurées par le capteur
delta (en g)	0.5	Ecart minimal autorisé de masses entre 2 feutres
mEqual (en g)	0.1	Ecart maximal de masses pour que celles-ci soient considérées égales
tMes (en ms)	50	Intervalle de temps entre 2 mesures du capteur
nMes	10	Nombre de mesures comparées entre elles pour vérifier la stabilité de la mesure

Remarque : la data-sheet du capteur indique : « 80SPS (= *sample per second*) output data rate »

Le temps de rafraichissement du capteur est donc : $t_{raf} = \frac{1}{80} = 0.0125\ s = 12.5\ ms$.

On a bien : $t_{raf} < tMes$.

- **Variables**

Voici les différentes variables utilisées dans le programme :

Variable	**Unité**	**Signification**
n	vide	Nombre de feutres utilisés dans le tour
mPot	g	Masse du pot
mPen[max]	g	Tableau des masses des feutres
mCurrentPen	g	Masse du feutre qui vient d'être retiré du pot
currentPen	vide	Numéro du feutre qui vient d'être retiré
mTot	g	Masse totale mesurée par le capteur

La brique élémentaire du programme Arduino est le sous-programme :

- ***isWeightDifferent()***

Ce programme renvoie *« true »* si la masse totale du système a changée et si cette variation s'est stabilisée.

1) Le programme attend que la masse *scale.getGram()* soit stabilisée ;

Pour cela il compare nMes mesures de masse prises à *tMes* l'une de l'autre. Si le maximum des valeurs absolues des 3 différences des mesures est inférieure à *mEqual*, alors le programme considère que la masse est stabilisée et passe à l'étape 2 :

2) Le programme teste si la masse totale a changée.

Pour cela il compare la nouvelle masse totale *scale.getGram()* à l'ancienne masse totale *mTot*. Si la différence est supérieure à *mMin*, alors un feutre a été ôté : le programme renvoie « *true* ». Sinon le programme renvoie « *false* ».

Voici les trois autres sous-programmes utilisés dans la boucle principale :

- ***parameters()***

Ce programme récupère les valeurs des différents paramètres définies par l'utilisateur (voir le tableau « Paramètres » page précédente).

Cette récupération est automatique : elle est réalisée par un dialogue Bluetooth entre la carte Arduino et le téléphone de l'utilisateur, dès que la connexion entre les 2 appareils est établie. La caractéristique « *parametersCharacteristic »* est utilisée pour l'échange de données.

- ***initialization()***

Ce programme permet d'initialiser le tour : le programme enregistre les différentes masses des feutres pesés l'un après l'autre.

1) Le programme reçoit de l'utilisateur le nombre *n* de feutres utilisés dans le tour ;

2) Le programme attend qu'un feutre soit posé sur la balance ;

Pour cela, il attend que la fonction *isWeightDifferent()* renvoie « *true* ».

3) Le programme enregistre les masses de chaque feutre ;

A chaque feutre est associé un numéro entre 0 et *n*. La masse de chaque feutre est stockée dans le tableau *mPen[max]* à la place correspondante.

4) Le programme s'assure que les masses des feutres sont différentes.

A chaque ajout de feutre, si la masse de ce feutre est proche de celle d'un autre feutre à moins de *delta*, alors le programme le signale à l'utilisateur. L'utilisateur doit alors changer de feutre et le repeser.

Après l'initialisation, la fonction suivante est utilisée pendant le tour proprement dit, tant qu'il reste des feutres dans le pot.

- ***takenPen()***

Ce programme renvoie le feutre qui vient d'être retiré du pot par le spectateur.

1) Le programme attend qu'un feutre soit retiré ;

Pour cela, il attend que la fonction *isWeightDifferent()* renvoie « *true* ».

2) Le programme détermine quel feutre vient d'être retiré.

Pour cela, il compare la différence de masse totale *mCurrentPen* avec les masses des différents feutres stockées dans le tableau *mPen[max]*. S'il trouve une masse de feutre proche à moins de *delta/3* de *mCurrentPen* il renvoie le numéro associé : *currentPen*.

- **Communication Bluetooth**

Pour transmettre les données à l'utilisateur, on utilise le module Bluetooth de la carte Radon. Pour cela, nous avons créé un service : *penService* auquel un numéro UUID a été attribué : a7b2fe81-25c5-4ca5-adfc-77e4f3834bea. Pour être certain de ne pas lui attribuer un numéro déjà utilisé, un générateur de numéro UUID unique a été utilisé.

La carte Radon attend que le téléphone se connecte. Puis une fois la connexion établie, échange des données avec lui selon 3 protocoles différents :

- *Read* : lire les données à sa disposition ;
- *Write* : envoyer des données à la Radon ;
- *Be Notify* : être averti quand une donnée change de valeur.

Les caractéristiques prennent en argument des entiers et renvoie des entiers codés sous forme de nombre en Hexadécimal codé sur 2 caractères (1 byte) : les valeurs sont donc codées entre $00 = 0$ et $FF = 255$.

En particulier, cela signifie que la masse du feutre doit être inférieur à 25.6g.

$$(int)10 * mPen < 256 \Rightarrow \boldsymbol{mPen} < 25.6\ g$$

3 types de données sont échangées entre la carte Radon et le téléphone Android. A chaque type est associée une caractéristique différente.

Voici les données transmises par les différentes caractéristiques :

- *ParametersCharacteristics*

Donnée échangée	Plage de valeurs acceptées	Valeur reçue	Par défaut	Protocole
mEqual	$0\ g \leq mEqual \leq 5.1\ g$	(int)50*mEqual	5	Write
mMin	$0\ g \leq mMin \leq 255\ g$ (valeurs entières)	(int)mMin	5	Write
delta	$0\ g \leq delta \leq 25.5\ g$	(int)10*delta	5	Write
tMes	$0\ s \leq tMes \leq 2550\ s$ (valeurs entières par pas de 10)	tMes/10	5	Write
nMes	$0 \leq nMes \leq 255$ (valeurs entières)	nMes	10	Write
n	$0 \leq n \leq max$ (valeurs entières)	n	3	Write

- switchWeightCharacteristic

Donnée échangée	Plage de valeurs acceptées	Valeur transmise	Protocole
mPen	$0\ g \leq mPen < 25.6\ g$	(int) 10*mPen	Notify

Cette caractéristique remplit également 1 fonction :

- Pendant ***initialization()*** : informer l'utilisateur si le feutre qui vient d'être posé peut être utilisé pour le tour (envoi d'un '1') ou s'il doit être changé (envoi d'un '0').

- switchPenCharacteristic

Donnée échangée	Plage de valeurs acceptées	Valeur transmise	Protocole
currentPen	$0 \leq currentPen \leq n$ (valeurs entières)	currentPen	Notify

Cette caractéristique remplit également 1 fonction :

- Pendant ***parameters()*** : informer l'utilisateur de la bonne réception des données.

Circuit imprimé

Pour accueillir l'ensemble des composants de notre circuit électronique, nous avons réalisé une plaque de circuit imprimé.

Cette plaque a été modélisée, puis imprimée, grâce au logiciel de PCB *Altium Designer*. Pour nous former à ce logiciel, nous avons suivi la formation dispensée dans le polycopié du Lisa disponible sur GitLab : https://gitlab.centralesupelec.fr/2005cabaretl/PolyAltium.

Obtenir un circuit imprimé remplit un double objectif :

- Support : fournir un support plan à notre circuit électronique, facilement intégrable ensuite dans un pot ;
- Miniaturisation : relier nos différents composant en limitant la taille des connexions (fils) volumineuses.

Voici les différentes étapes de la conception de notre circuit imprimé sous Altium :

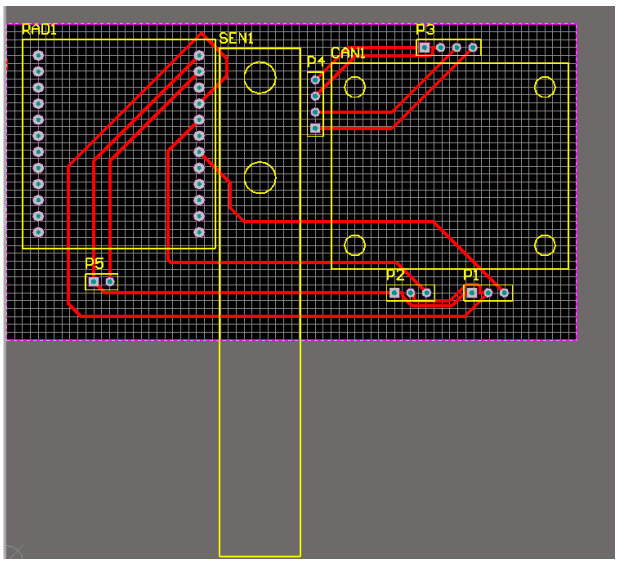
1) Création d'une nouvelle bibliothèque de composants.

Pour chaque composant de notre circuit (Radon, CAN et capteur de force), comme ceux-ci n'existaient dans aucune bibliothèque prédéfinie de Altium, nous avons dû créer :

- Le schéma du composant (son symbole sur le circuit électronique) associé à :
- Son empreinte de PCB : ses dimensions (surface de contact du composant avec la plaque, épaisseur), ses pattes (position, nombre, diamètre), ses trous (de fixation).

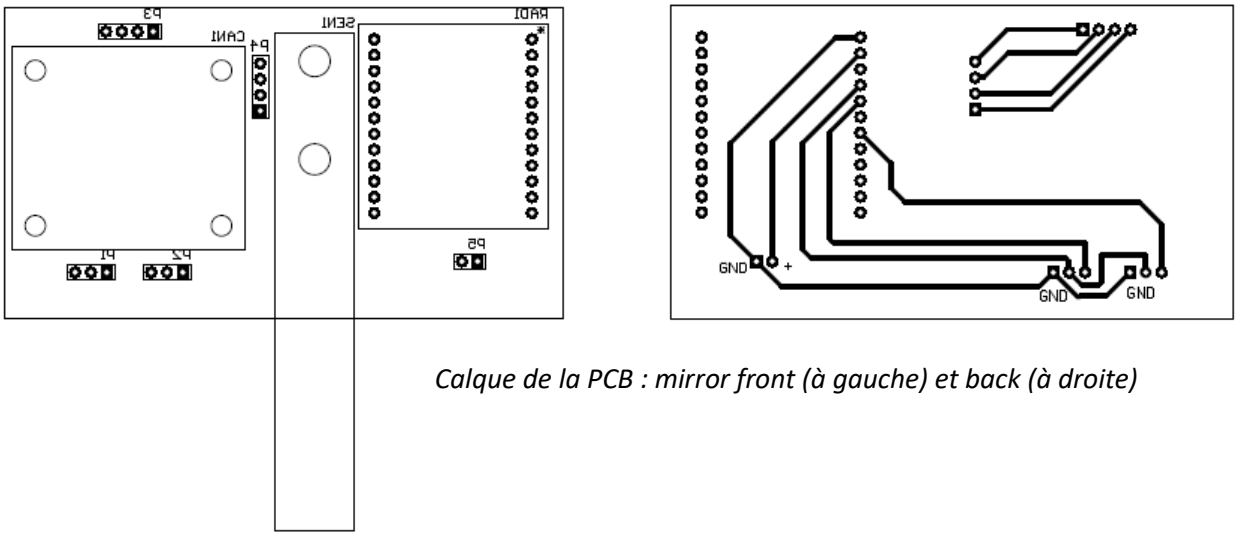
2) Réalisation du schéma du circuit électronique (présenté dans la partie précédente).

3) Création de la PCB : agencement des composants, connexion des composants pour respecter le schéma.



Modèle Altium de la PCB du dispositif

4) Impression du typon



Calque de la PCB : mirror front (à gauche) et back (à droite)

5) Fabrication de la carte

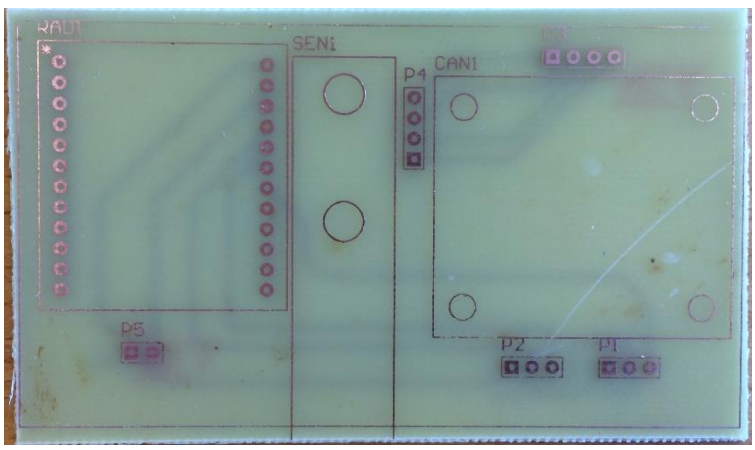
On part d'une plaque d'époxy recouverte d'une feuille de cuivre, elle-même recouverte d'une pellicule photosensible. Puis on applique le protocole suivant :

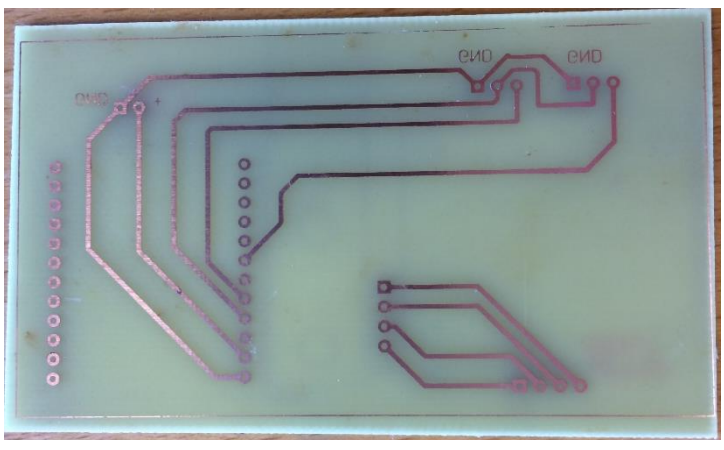
- Placer la plaque entre les 2 feuilles du calque ;
- Exposer la plaque aux rayonnements UV sur les 2 faces (3 min) ;

La partie de la couche photosensible non protégée par le calque reçoit un rayonnement UV.

- Placer la plaque dans le liquide révélateur : la partie exposée aux UV réagit avec le liquide révélateur, les pistes apparaissent ;
- Placer la plaque dans une solution qui va attaquer le cuivre non protégé par la pellicule photosensible (5 min à 32°C) ;
- Nettoyer la plaque dans une solution de soude.

On obtient finalement notre plaque sur laquelle on retrouve uniquement les pistes de cuivre présentes sur le typon.

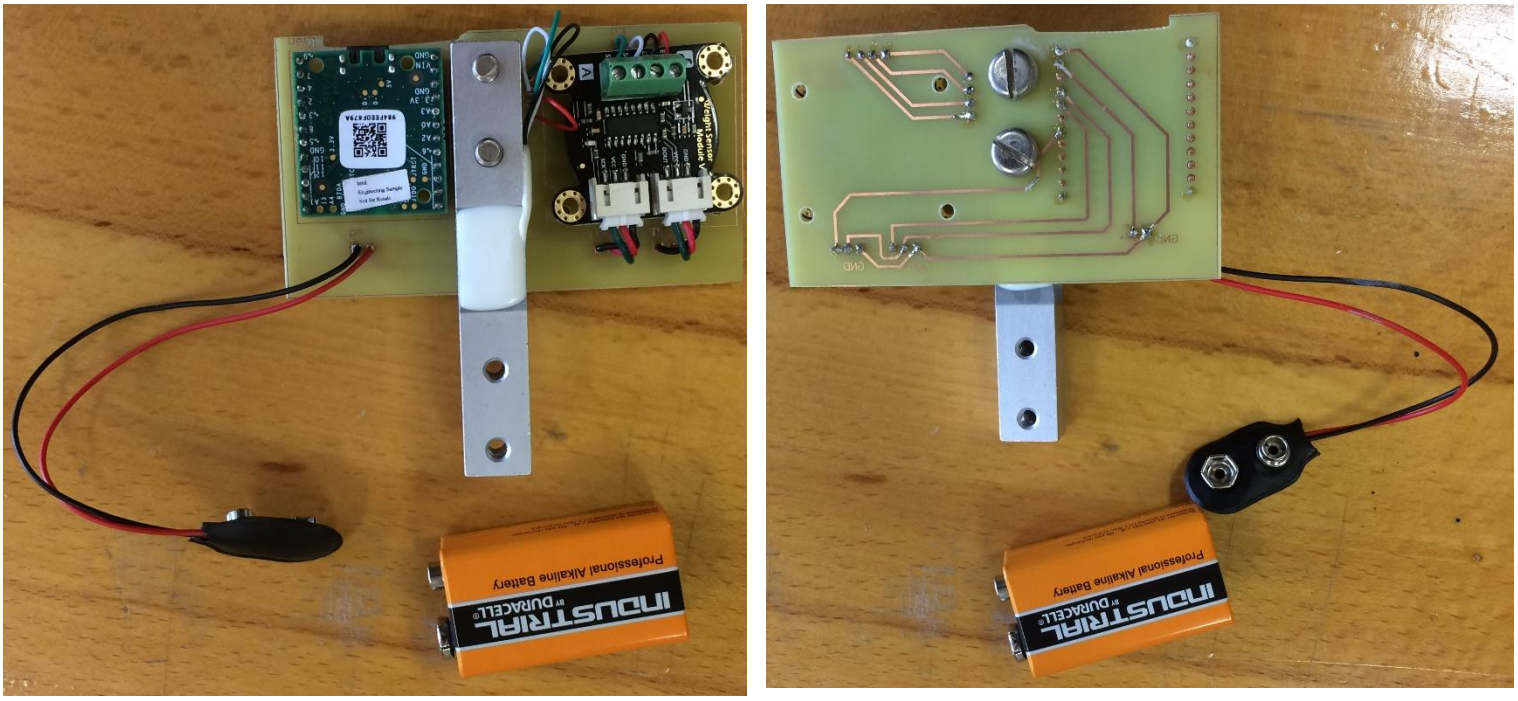




6) Câblage physique des composants

On câble ensuite le circuit en suivant le schéma électronique définit sous Altium dans la partie précédente.

On obtient finalement notre circuit imprimé.



Circuit imprimé final : front (à gauche) et back (à droite)

Rédaction du code Android

Objectif principal

Le dispositif permettant de détecter les feutres qui sont pris dans le pot et de transmettre l'information, il a fallu réaliser l'autre partie du projet qui a pour but d'assurer ces deux fonctions principales :

- Recevoir les données transmises par le pot.
- Transmettre discrètement ces informations au magicien (à travers des vibrations).

A ces deux fonctions principales, nous avons également ajouté les fonctions suivantes qui sont importantes pour assurer un déroulement optimal du tour :

- Permettre l'initialisation du tour en guidant le magicien durant chaque étape.
- Permettre la personnalisation des paramètres afin de trouver les données optimales pour une détection des feutres fiable et des vibrations clairement identifiables.

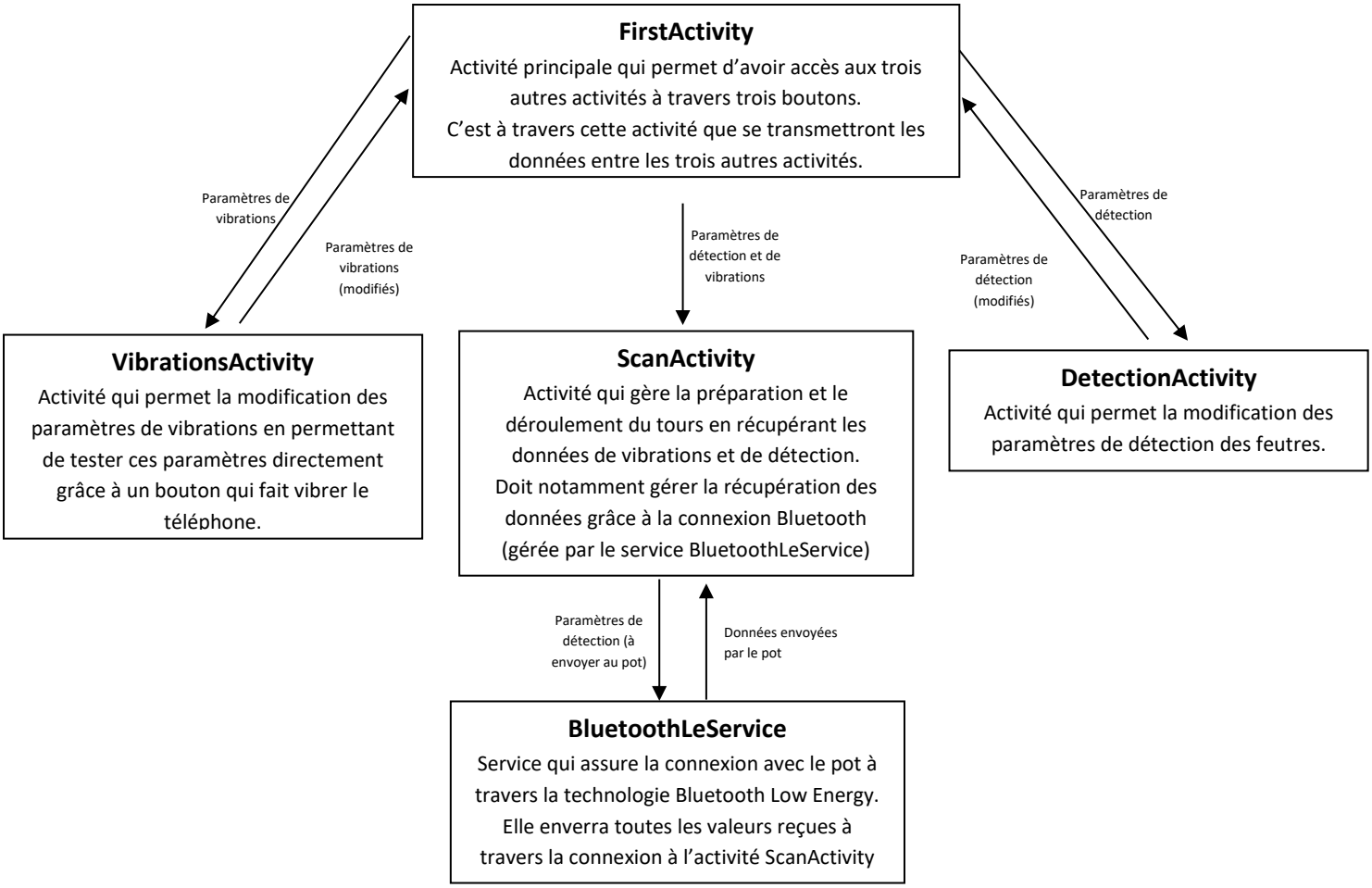
Fonctionnement global de l'application

Après plusieurs tests en implémentant itérativement de nouvelles fonctionnalités, il a été décidé que l'application serait axée sur 3 vues :

- Une partie de réglage des paramètres de vibrations où l'utilisateur peut choisir les paramètres du signal rectangulaire envoyé par l'application. Il faudra notamment jauger entre un signal plus long pour que celui-ci soit plus facilement identifiable et un signal plus court pour avoir une meilleure réactivité.
- Une partie de réglage des paramètres de détection des feutres où l'utilisateur peut choisir tous les paramètres présentés dans la partie « Rédaction du code Arduino » qui permettent d'avoir une détection fiable et la plus rapide possible.
- Une partie principale pour gérer le déroulement global du tour avec les différentes étapes (De la connexion de l'application jusqu'à la fin du tour).

Architecture de l'application

A partir des trois grandes parties que doit gérer l'application, on obtient l'architecture finale de l'application qui est la suivante :



L'application fonctionne donc à travers 5 classes dont 4 activités et 1 service. Tous ces fichiers seront présentés plus en détail par la suite.

Sur le schéma ci-dessus, toutes les flèches représentent les informations qui sont échangées entre chaque activité ou service. Ces échanges de données se font à travers des objets de la classe Intent en utilisant la méthode putExtra.

Rédaction du code

Le projet Android est constitué de plusieurs types de fichiers :

- Le fichier MANIFEST qui contient les informations générales de l'application et notamment toutes les permissions qui sont accordées à l'application. Les permissions demandées dans le cadre de notre application sont les suivantes :
    - Contrôle du vibreur
    - Utilisation du Bluetooth
- Les fichiers sources en JAVA qui exécutent toutes les opérations qui doivent être faites par chacune des activités et service. Ils forment le contrôleur de l'application.
- Les fichiers XML correspondant aux layouts qui permettent d'établir l'interface de chaque activité en y ajoutant les widgets correspondant. Ils forment la vue de l'application.
- Les fichiers XML correspondant aux ressources de l'application, ils contiennent entre autres les couleurs et les textes utilisés dans l'application.

Pour chacune des activités sont associés un fichier source et un fichier layout.

Création de chacune des activités

- ***First Activity***

  Comme présenté précédemment, cette activité doit permettre de rediriger vers les trois autres activités. Elle est donc composée seulement de trois boutons (Button) :

  Chacun de ces boutons lance un Intent qui ouvre l'activité en question et envoie à cette activité toutes les informations nécessaires à travers la fonction putExtra.

  Pour lancer les activités DetectionActivity et VibrationActivity, on utilise la fonction startActivityForResult qui attend en retour un résultat afin de mettre à jour tous les paramètres.

- ***Detection Activity***

  Cette activité permet de régler tous les paramètres liés à la détection. Ces paramètres sont au nombre de 6 et ont été présentés dans le tableau correspondant à *Parameters Characteristic* de la partie de la rédaction du code Arduino.

  Le réglage de chacun de ces paramètres se fait grâce à des boutons coulissants (SeekBar) qui mettent à jour des zones de texte (TextView) à chaque fois que la position d'un bouton est changée.

Il y a également deux boutons :

- Un bouton qui permet de valider ces paramètres et de retourner à l'activité FIrstActivity
- Un bouton qui permet de rétablir les paramètres par défauts qui ont été renseignés dans la partie électronique.

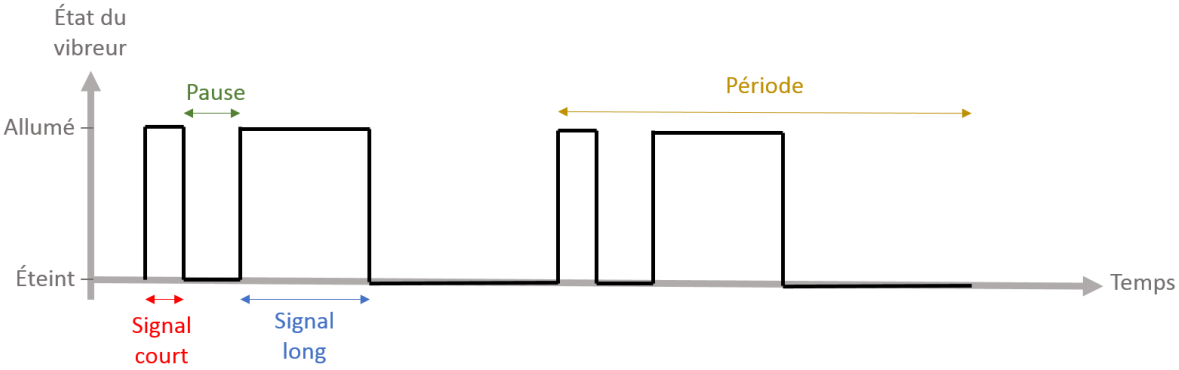
En ce qui concerne les SeekBar qui mettent à jour les zones de textes, on utilise la methode setOnSeekBarChangeListener qui permet de personnaliser les actions qui ont lieu lorsqu'on touche à ce widget. On modifie notamment la méthode onProgressChanged.

- ***VibrationsActivity***

Cette activité a un comportement très similaire de celui de l'activité DetectionActivity. En effet, elle permet également de régler des paramètres : ceux qui sont liés aux vibrations.

Il a été décidé que le signal de vibrations sera envoyé deux fois au magicien. Ces vibrations reprennent celles qui ont été présentées en début de rapport avec les signaux longs et les signaux courts. Il y a donc 4 paramètres à régler (voir le schéma ci-dessous qui représente l'envoi du signal correspondant au chiffre 4) :

- La durée d'un signal court
- La durée d'un signal long
- La durée de la pause entre un signal court et un signal long
- La période liée à un signal. La durée totale des deux signaux vaudra donc deux périodes.



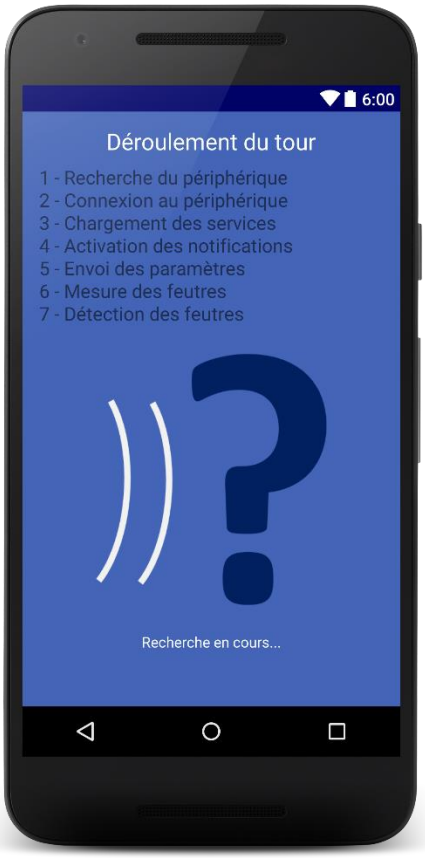
On doit donc avoir 4 objets SeekBar qui permettent de régler ces 4 paramètres. Tout comme dans DetectionActivity, il y a deux boutons : un pour validé ces paramètres et un autre pour remettre les paramètres par défaut. À cela, on ajoute un bouton qui permet de tester ces vibrations pour tous les chiffres possibles en le choisissant dans une zone de texte. On obtient donc l'interface suivante :

Cette interface gère aussi une particularité : les 4 paramètres doivent respecter certaines contraintes entre elles, et des conditions permettent de bloquer les SeekBars lorsqu'une de ces relations n'est plus respectée :

- signalCourt < signalLong
- 2*signalLong+pause < periode

- ***ScanActivity***

Cette activité est la plus importante, c'est celle qui s'occupe du déroulement de tout le tour Pour gérer la connexion, cette activité fait appel à un service qui a également été créé : BluetoothLeService. La procédure à suivre a été décomposée en 7 étapes dans l'applications :



1. Recherche du périphérique

   Tout d'abord, l'activité recherche les périphériques environnants en utilisant la fonction startScan d'un objet BluetoothLeScanner. Celui-ci recherchera tous les périphériques Low Energy et dès qu'il trouvera un périphérique ayant la même UUID que celle qui a été renseigné pour le pot, alors l'activité fera appel au service BluetoothLeService en lui envoyant les informations de ce périphérique.

2. Connexion au périphérique

   Dès que le BluetoothLeService est appelé, il va lancer la connexion et en cas de succès, envoie un message à l'activité (à travers un Intent) pour mettre à jour l'affichage sur la

vue du téléphone. A partir de maintenant, dès que le périphérique se déconnectera, l'activité se fermera automatiquement pour revenir à l'écran de départ.

3. Chargement des services

   L'objet BluetoothLeService va maintenant rechercher les services Bluetooth disponibles : dans notre cas, il n'y en a que 1. Une fois que celui-ci est chargé, on envoie un Intent à l'activité pour que la vue soit mise à jour.

4. Activation des notifications

   Il faut ensuite activer les notifications pour les deux caractéristiques switchWeightCharacteristic et switchPenCharacteristic. Pour cela on fait encore appel au service BluetoothLeService qui va mettre en place ces deux caractéristiques.

5. Envoi des paramètres

   Une fois les notifications activées. On peut envoyer tous les paramètres de détection au pot en utilisant la fonction writeCharacteristicValue dans le BluetoothLeService. On se sert ici des notifications pour à chaque fois être sûr que l'envoi a été fait.

6. Mesure des feutres

   Cette étape correspond à la pesée des feutres un par un. Pour cela, on lit les valeurs indiquées par la caractéristiques switchWeightCharacteristic. Pour chaque feutre, cette caractéristique change deux fois de valeurs : une première fois pour indiquer la masse du feutre, puis une deuxième fois pour indiquer si cette masse est correcte ou si elle est trop proche d'un autre feutre auquel cas il faut changer de feutre.

7. Détection des feutres

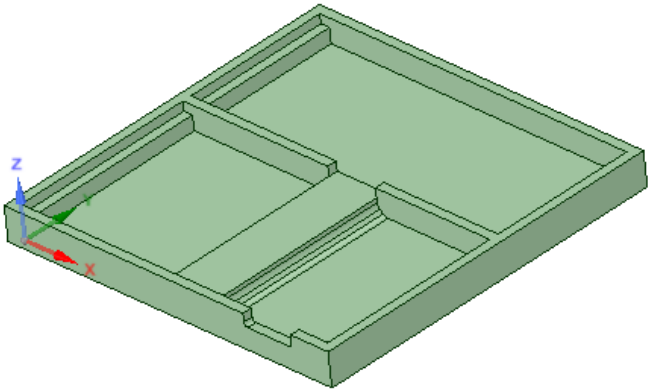
   C'est cette étape qui correspond au tour : le magicien met le portable dans sa poche afin de pouvoir sentir les vibrations et à chaque fois que le téléphone détecte qu'un feutre a été enlevé, il va vibrer en envoyant un signal correspondant au numéro du feutre, permettant ainsi au magicien de dire au spectateur quelle zone du coloriage colorier.

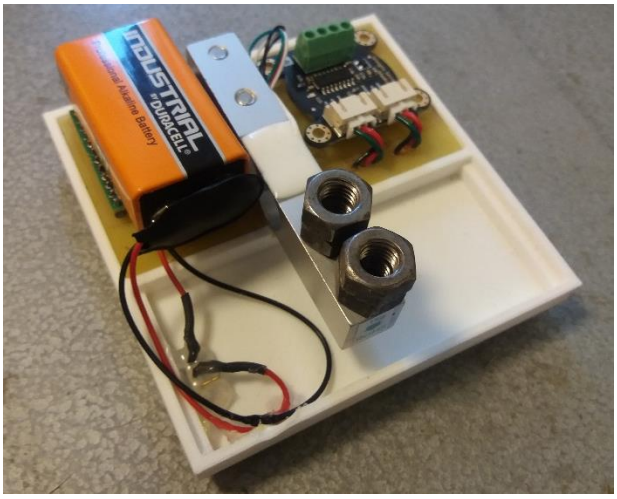
Réalisation du pot

La base du pot doit satisfaire 2 contraintes :

- Recevoir le circuit imprimé sans jeu : afin que le capteur soit bien fixe et que les mesures ne soient pas faussées ;
- Etre stable : la base doit être en contact de façon stable avec le sol pour éviter les erreurs de mesures.

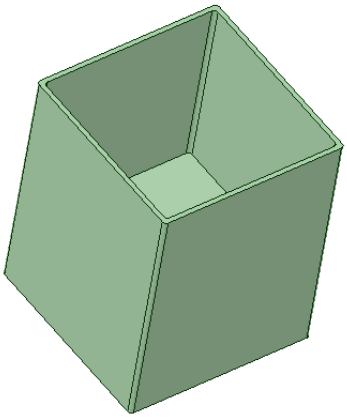
La solution que nous avons retenue pour satisfaire ces contraintes est l'impression 3D.





Le design de la base du pot sous Space Claim

Pour ce qui est du revêtement du pot, nous souhaitions également l'imprimer en 3D, ci-dessous une image du fichier Space Claim associé.



Le design du revêtement du pot sous Space Claim

Malheureusement, cette impression était irréaliste du fait de 2 inconvénients majeurs :

- Temps d'impression = 10h ;
- Gâchis de matière : énormément de support utilisé pour l'impression de la base du revêtement.

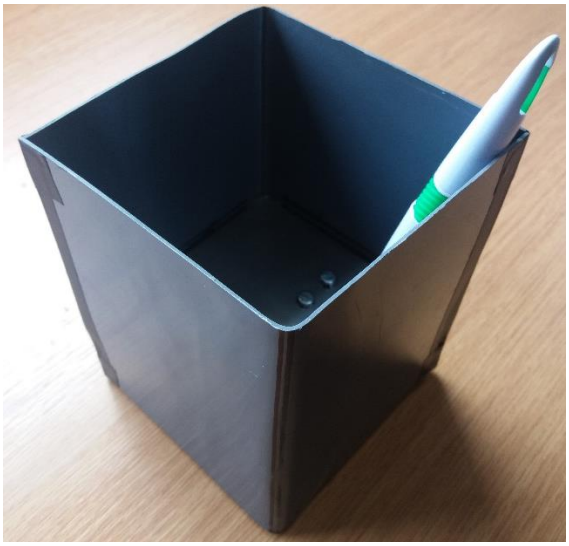
Nous nous sommes donc rabattus sur une solution alternative : un revêtement en plastique thermo-plié, avec une base en aluminium.

Le revêtement thermo-plié : vue de haut (à gauche), vue de dessous (à droite)

Le plateau a été réalisé en aluminium car il devait satisfaire 2 contraintes :

- Stabilité : le plateau devait être suffisamment lourd pour que l'ajout/l'enlèvement de feutres correspondent à de faibles variations de masses relatives. Dans ces conditions, le système oscille moins longtemps et converge plus vite ;
- Rigidité : le plateau devait être suffisamment rigide pour ne pas se déformer sous le poids des feutres, afin que le revêtement ne soit pas en contact avec le support, sinon cela aurait faussé les mesures.

Voici le pot une fois l'assemblage final effectué :



Le revêtement permet de dissimuler efficacement le circuit électronique mais sans toucher ni la base ni la table : son seul point de contact est l'autre extrémité du capteur.

Tests

Reverse-engineering

L'objectif de notre projet était de pouvoir reproduire le tour « Color Pen Prediction » présenté par le magicien Justin Willman au « Ellen Degeneres Show ».

Pour juger si notre dispositif permet de remplir cet objectif, il nous faut comparer s'il est aussi performant que l'original. Pour cela, nous allons comparer les temps de réponses des 2 dispositifs :

$$tps(réponse) = tps(détection) + tps(codage\ vibratoire) + tps(décodage)$$

Ce temps peut être mesuré sur la vidéo de la performance originale comme l'intervalle de temps entre le moment où Ellen prend le feutre et le moment où Justin lui dit quelle zone du dessin colorier.

Feutre	Jaune	Orange	Rouge	Vert	Bleu
Tps de réponse (en s)	3.1	4.0	2.9	2.9	2.9

Le temps de réponse théorique moyen est donc : $t_{rep,th} = \frac{3.1+4.0+2.9+2.9+2.9}{5}$

$$\boldsymbol{t_{rep,th} = 3.16\ s}$$

Résultats

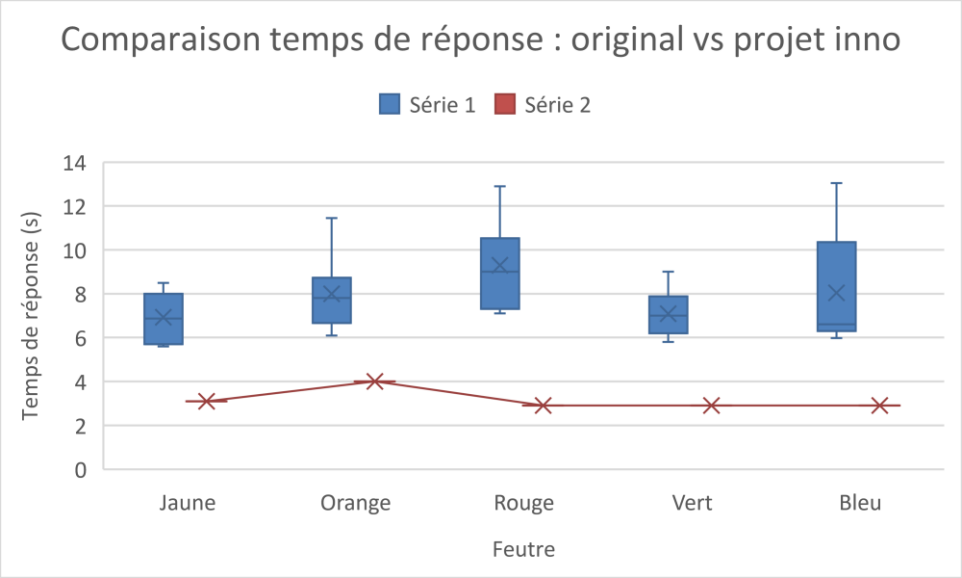
Pour comparer le dispositif original avec notre dispositif, **20 tests successifs ont été réalisés** sur notre dispositif. Chaque test constituait une reproduction complète du **tour de magie à 5 feutres** : de la pesée des feutres lors de l'initialisation au tour de mentalisme à proprement parlé.

- Fiabilité

Notre système montre une fiabilité de 100% : lors des tests, à chaque fois qu'un feutre était pris par le spectateur, il était effectivement détecté par le système qui envoyait le bon numéro associé.

- Temps de réponse

Pour comparer numériquement les deux systèmes, nous nous sommes basés sur le seul critère mesurable sur la vidéo : le temps de réponse. Nous avons donc mesuré pour chaque test le temps de réponse associé à chaque feutre. Ci-dessous la synthèse de nos résultats.



La courbe orange représente les temps de réponse du dispositif original, mesurés sur la vidéo pour chaque feutre. Les histogrammes bleus représentent la distribution, à un écart-type près, autour de la moyenne des temps de réponses par feutres, pour notre dispositif.

On remarque que **notre dispositif est globalement plus lent** : **son temps de réponse moyen** est de $\mathbf{7.76}\ \boldsymbol{s}$, contre seulement $3.16\ s$ pour le dispositif original.

Bilan des coûts du prototype :

Partie	**Objet**	**Prix**
Electronique	Radon	40 € (Genuino 101)
	Capteur de force DFRobot	26.85 €
	PCB	5 €
Mécanique	Feutres	5 €
	Base du pot (impression 3D)	1.65 €
	Revêtement du pot	5 €

Le coût de fabrication de notre prototype est : **82.5 €**.

Bien évidemment, en cas de production en série, ces coûts diminueraient drastiquement dans les secteurs :

- PCB : maintenant que le calque est disponible, il serait possible d'automatiser la réalisation du circuit imprimé à partir de plaques directement prédécoupées aux bonnes dimensions ;
- Base du pot : un moule de la base du pot pourrait être réalisé. Cela permettrait de remplacer l'impression 3D, coûteuse et surtout lente, par un moulage plastique, peu cher et rapide. De même pour le revêtement du pot, un moulage plastique est la solution la plus économique et rapide.

Enfin, le coût individuel de tous les composants pourrait être réduit par de grosses commandes groupées.

On devrait pouvoir arriver à un coût du dispositif à l'unité de l'ordre de 60 €. Ce coût est bien plus faible que le prix à la vente du gimmick original du tour « Color Pen Prediction » présenté en introduction : 200 € l'unité.

Conclusion

En partant de la vidéo du tour réalisé avec le gimmick original, considéré comme une « boîte noire », l'objectif de notre projet était de réaliser un dispositif permettant de présenter le même effet magique de mentalisme selon 2 exigences :

1) Le choix des feutres est libre ;
2) Le magicien ne peut voir les couleurs choisies avant la fin du tour.

Cet objectif est atteint : nous pouvons présenter un prototype fiable qui remplit ces 2 exigences.

Pour juger plus en détail de la réussite de notre projet, il est nécessaire de comparer notre dispositif avec le dispositif original.

Points négatifs de notre dispositif :

- Dispositif plus lent

On a pu voir dans la partie « Tests » que notre dispositif avait un temps de réponse plus élevé que le gimmick original. Toutefois, cela n'est pas gênant pour la performance du tour de magie : un temps de réponse de l'ordre de 7s ne rend pas le tour suspect du point de vue du spectateur.

Malgré tout, si l'on souhaite améliorer ces performances, il pourrait être intéressant de réaliser une étude comparative de capteurs de force, précis à 0.1g près, pour en choisir un qui converge plus rapidement que celui que nous utilisons.

- Design inadapté

Notre pot présente en effet un design inadapté : il est trop gros et ne ressemble pas à un pot normal. Ce défaut est dû au manque de temps pour réaliser le revêtement du pot.

Pour résoudre ce problème, il faudrait réaliser un design plus complexe de revêtement du pot à imprimer en 3D. Un revêtement circulaire et évasé permettrait par exemple de diminuer drastiquement le volume du pot tout en dissimulant le capteur et la base du pot.

Points positifs de notre dispositif :

- Prix à la vente moins élevé

On a pu voir dans la partie « Bilan des coûts du prototype » que notre dispositif, s'il était produit en série, coûterait de l'ordre de 60 € l'unité soit bien moins que le prix de vente du gimmick original : 200 € l'unité.

Notre prix de vente bien moins élevé constituerait un atout important pour concurrencer le gimmick existant sur le marché.

- Dispositif plus versatile

Contrairement au gimmick original, notre dispositif permet de réaliser ce tour avec des feutres ordinaires et un nombre de feutres qui n'est pas fixé à l'avance à 5 : pouvoir réaliser ce tour avec les feutres du spectateur par exemple rajoute encore à l'effet magique.

De plus, notre dispositif peut permettre de réaliser d'autres tour de mentalisme que le tour de Color Pen Prediction. Par exemple, si l'on remplit le pot de bonbons, la détection de différences masse peut permettre de connaître le nombre de bonbons qui viennent d'être pris par le spectateur.

Remerciements

Nous tenons à remercier d'abord le laboratoire du LISA : Laurent Cabaret et Hanane Meliani, nos encadrants, pour nous avoir guidé par leurs questions et leurs conseils tout au long de ce semestre. Merci également à Didier Coudray qui a toujours réussi à répondre à nos (nombreuses !) questions et demandes. Leur aide nous a permis de pousser ce projet plus loin que ce que nous pensions possible au départ.

Merci ensuite à la fondation ECP dont l'aide financière nous a permis de réaliser ce projet. Sans cette subvention la réalisation de ce prototype n'aurait pas été possible.

Annexes

Bibliothèque Hx711

Les bibliothèques sont à décompresser dans le dossier :
C:\Users\YourName\Documents\Arduino\libraries

Les deux fichiers ci-dessous : *Hx711*.h et *Hx711.cpp* sont à enregistrer dans un dossier *Hx711* dans le dossier *libraries*.

- Fichier Hx711.h

```
/* Arduino library for digital weight scale of hx711
 *
 * hardware design: syyyd
 * available at http://syyyd.taobao.com
 *
 * library design: Weihong Guan (@aguegu)
 * http://aguegu.net
 *
 * modified for Arduino Radon : Gael Colas
 *
 * library host on
 * https://github.com/aguegu/Arduino
 *
 *  Created on: Oct 31, 2012
 *
 *  Modified on : May 07, 2017
 */

#ifndef HX711_H_
#define HX711_H_

#include "Arduino.h"

class Hx711
{
public:
		Hx711(uint8_t pin_din, uint8_t pin_slk);
		Hx711();
		virtual ~Hx711();
		void begin(uint8_t pin_din, uint8_t pin_slk);
		long getValue();
		long averageValue(byte times = 25);
		void setOffset(long offset);
		void setScale(float scale = 1992.f);
		float getGram();

private:
		uint8_t _pin_dout;
		uint8_t _pin_slk;
		long _offset;
		float _scale;
};

#endif /* HX711_H_ */
```

- Fichier Hx711.cpp

```
/*
 * Hx711.cpp
 *
 *  Created on: Oct 31, 2012
 *      Author: agu
 */

#include "Hx711.h"

Hx711::Hx711(uint8_t pin_dout, uint8_t pin_slk)
{
	begin(pin_dout, pin_slk);
}

Hx711::Hx711()
{
}

Hx711::~Hx711()
{

}

void Hx711::begin(uint8_t pin_dout, uint8_t pin_slk) {

	_pin_dout = pin_dout;
	_pin_slk = pin_slk;
	pinMode(_pin_slk, OUTPUT);
	pinMode(_pin_dout, INPUT);

	digitalWrite(_pin_slk, HIGH);
	delayMicroseconds(100);
	digitalWrite(_pin_slk, LOW);

	averageValue();
	this->setOffset(averageValue());
	this->setScale();
}

long Hx711::averageValue(byte times)
{
	long sum = 0;
	for (byte i = 0; i < times; i++)
	{
		sum += getValue();
	}

	return sum / times;
}

long Hx711::getValue()
{
	byte data[3];

	while (digitalRead(_pin_dout))
		;

	for (byte j = 0; j < 3; j++)
	{
		for (byte i = 0; i < 8; i++)
		{
```

```
                digitalWrite(_pin_slk, HIGH);
                bitWrite(data[2 - j], 7 - i, digitalRead(_pin_dout));
                digitalWrite(_pin_slk, LOW);
            }
        }

        digitalWrite(_pin_slk, HIGH);
        digitalWrite(_pin_slk, LOW);

        return ((long) data[2] << 16) | ((long) data[1] << 8) | (long) data[0];
}

void Hx711::setOffset(long offset)
{
        _offset = offset;
}

void Hx711::setScale(float scale)
{
        _scale = scale;
}

float Hx711::getGram()
{
        long val = (averageValue() - _offset);
        return (float) val / _scale;
}
```

Code Arduino

```
#include <CurieBLE.h> //library for BLE communication
// examples available here : https://www.arduino.cc/en/Reference/CurieBLE

BLEPeripheral blePeripheral; // BLE Peripheral Device (the board you're programming)
BLEService penService("a7b2fe81-25c5-4ca5-adfc-77e4f3834bea"); // BLE Pen Service

// BLE Parameters Characteristic
BLEByteCharacteristic parametersCharacteristic("a7b2fe81-25c5-4ca5-adfc-77e4f3834beb", BLERead | BLEWrite);

// BLE Current Weight Characteristic
BLEByteCharacteristic switchWeightCharacteristic("a7b2fe81-25c5-4ca5-adfc-77e4f3834bec", BLERead | BLENotify);
// remote clients will be able to get notifications if this characteristic changes
// the characteristic is 2 bytes long as the first field needs to be "Flags" as per BLE specifications
// CAREFUL : the weight must not exceed 25.6g (because the date is stored in Hex on 2 characters : 16² = 256 = 25.6*10)

// BLE Current Pen Characteristic
BLEByteCharacteristic switchPenCharacteristic("a7b2fe81-25c5-4ca5-adfc-77e4f3834bed", BLERead | BLENotify);
// remote clients will be able to get notifications if this characteristic changes
//
https://developer.bluetooth.org/gatt/characteristics/Pages/CharacteristicViewer.aspx?u=org.bluetooth.characteristic.heart
_rate_measurement.xml

// useful hex-to-decimal converter : http://www.binaryhexconverter.com/hex-to-decimal-converter

#include <Hx711.h> //library for the weight sensor

Hx711 scale ; //the weight sensor
/* library design: Weihong Guan (@aguegu)
 * library host on : https://github.com/aguegu/ardulibs/tree/3cdb78f3727d9682f7fd22156604fc1e4edd75d1/hx711
 */

double mPot ;//the weight of the pot (in g)

const int max = 6 ; //the maximum number of pens that the sensor can detect
float mPen [max] ; //the array of the weights of the different pens (in g)

float mCurrentPen ; //the weight of the pen which has just been taken (in g)
int currentPen ; //the number of the pen which has just been taken

float mTot ;//the total weight measured by the sensor (in g)

int n = 0 ;//the number of pen used in the trick
float mEqual = 0.1 ; //the maximal difference in weight for which 2 different weight are considered equal (in g)
float mMin = 5 ;//the lower boundary of weight measured by the sensor (in g)

float delta = 0.5 ;//the minimal difference of weight allowed between 2 pens (in g)
int tMes = 50 ;//the time between 2 measurement of the sensor (in ms)
int nMes = 10 ;//the number of measurements taken to ensure that the measurement is stable

void setup() {
  Serial.begin(9600) ;
  // Hx711.DOUT - pin #A2
  // Hx711.SCK - pin #A3
  scale.begin(A2, A3) ;

  // set advertised local name and service UUID:
  blePeripheral.setLocalName("CPP");
  blePeripheral.setAdvertisedServiceUuid(penService.uuid());
```

```
  // add service and characteristic:
  blePeripheral.addAttribute(penService);
  // CAREFUL : you have to add the characteristics in the SAME order you defined them
  blePeripheral.addAttribute(parametersCharacteristic);
  blePeripheral.addAttribute(switchWeightCharacteristic);
  blePeripheral.addAttribute(switchPenCharacteristic);

  // set the initial value for the characteristics:
  parametersCharacteristic.setValue(0);
  switchWeightCharacteristic.setValue(0);
  switchPenCharacteristic.setValue(0);

  // begin advertising BLE service:
  blePeripheral.begin();
}

void loop() {

  // wait until a central is connected to peripheral:
  while(!blePeripheral.central()) {
     blePeripheral.central();
  }
  Serial.print("Connected to central: ");
  // print the central's MAC address:
  Serial.println(blePeripheral.central().address());

  // poll peripheral
  blePeripheral.poll();

  parameters() ;

  initialization();

  while (true){
    Serial.print("Le feutre qui vient d'être pris est le feutre numéro : ");
    Serial.println(takenPen());
    switchPenCharacteristic.setValue(currentPen);  // notify the user of the current pen which has been taken by the public
  }
}

boolean isWeightDifferent () {
  float difference [nMes] ;
  difference [0] = scale.getGram() - mTot ;
  difference [9] = difference[0] + 0.11 ;
  delay(tMes) ;

  int counter = 1 ;

  //compare nMes measurement taken at tMes interval to ensure that the measurement is stable
  while ((maxAbsDiffVec(difference, nMes) > mEqual) || (abs(difference[0]) > 30)){
    difference[counter] = scale.getGram() - mTot ;
    delay(tMes);

    counter = (counter+1)%10 ;
  }

  //return "true" if the difference between the old and the new value of weight is superior to mMin : a pen has been
taken/dropped
  float result = (abs(difference[0]) > mMin) ;
  return result ;
}

// get the set of parameters from the user
```

```
void parameters() {
  // wait until the parameter is received
  while(parametersCharacteristic.value() == 0){
  }
  // the first parameter received is (int)50*mEqual
  mEqual = parametersCharacteristic.value()/50. ; //the maximal difference in weight for which 2 different weight are
considered equal (in g)
  Serial.println(mEqual) ;

  // reset the characteristic to 0
  parametersCharacteristic.setValue(0) ;
  //notify the user that the parameter has been received
  switchPenCharacteristic.setValue(1) ;

  // wait until the parameter is received
  while(parametersCharacteristic.value() == 0){
  }
  // the second parameter received is (int)mMin
  mMin = parametersCharacteristic.value() ; //the lower boundary of weight measured by the sensor (in g)
  Serial.println(mMin) ;

  // reset the characteristic to 0
  parametersCharacteristic.setValue(0) ;
  //notify the user that the parameter has been received
  switchPenCharacteristic.setValue(2) ;

  // wait until the parameter is received
  while(parametersCharacteristic.value() == 0){
  }
  // the third parameter received is (int)10*delta
  delta = parametersCharacteristic.value()/10. ;//the minimal difference of weight allowed between 2 pens (in g)
  Serial.println(delta) ;

  // reset the characteristic to 0
  parametersCharacteristic.setValue(0) ;
  //notify the user that the parameter has been received
  switchPenCharacteristic.setValue(3) ;

  // wait until the parameter is received
  while(parametersCharacteristic.value() == 0){
  }
  // the forth parameter received is tMes/10
  tMes = 10*parametersCharacteristic.value() ;//the time between 2 measurement of the sensor (in ms)

  // reset the characteristic to 0
  parametersCharacteristic.setValue(0) ;
  //notify the user that the parameter has been received
  switchPenCharacteristic.setValue(4) ;
  Serial.println(tMes) ;

  // wait until the parameter is received
  while(parametersCharacteristic.value() == 0){
  }
  // the fifth parameter received is nMes
  nMes = parametersCharacteristic.value() ;//the number of measurements taken to ensure that the measurement is stable
  Serial.println(nMes) ;

  // reset the characteristic to 0
  parametersCharacteristic.setValue(0) ;
  //notify the user that the parameter has been received
  switchPenCharacteristic.setValue(5) ;
}
```

```
// initialize the trick : register the number of pens, their weights
void initialization() {

 Serial.print("Combien de feutres voulez-vous mettre dans le pot ? \n") ;
 n = parametersCharacteristic.value() ; // nb of pens
 while (n == 0){// any value other than 0
  n = parametersCharacteristic.value() ; // nb of pens
 }
 Serial.print(n) ;
 Serial.print(" feutres vont être utilisés pour ce tour. \n") ;

 // reset the characteristic to 0
 switchPenCharacteristic.setValue(0) ;

 mTot = scale.getGram() ;

 //register the weight of the different pens in the mPen array
 for (int i=0 ; i<n ; i++) {
  Serial.print("Posez le feutre ") ;
  Serial.print(i+1) ;
  Serial.print(" sur la balance. \n") ;
  while (!isWeightDifferent()){
  }
  mPen[i] = scale.getGram() - mTot ;
  mTot = scale.getGram() ;
  Serial.print("Le feutre ") ;
  Serial.print(i+1) ;
  Serial.print(" a une masse de mFeutre = ") ;
  Serial.print(mPen[i], 1) ;
  Serial.print(" g. \n") ;

  // transmit mPen[i] rounded to the next 0.1 to the user
  float mPenRound = ((int)(mPen[i]*10)) ;
  switchWeightCharacteristic.setValue(mPenRound);  // notify the user of the weight of the pen which has just been added

  boolean alright = true ;

  for (int j = 0 ; j<i ; j++){
   if (abs(mPen[j] - mPen[i]) <= delta){
    Serial.print("Le feutre ") ;
    Serial.print(i+1) ;
    Serial.print(" a la même masse que le feutre ") ;
    Serial.print(j+1) ;
    Serial.print(".\n");
    Serial.print("Changez de feutre ") ;
    Serial.print(i+1) ;
    Serial.print(", puis repesez le.\n");

    // there is a problem
    alright = false ;

    while (mTot - scale.getGram() < mMin){ // wait until the user remove the pen
    }
    mTot = scale.getGram() ;
    i--;
    j = i ; //on sort de la boucle
   }
  }
  if (alright){
   // notify the user that the weight of the pen which has just been added is different from the weight of the other pens
   switchWeightCharacteristic.setValue(1); // the pen is fine
  }else{
   // notify the user that the weight of the pen which has just been added is similar to the weight of an other pen
```

```
     switchWeightCharacteristic.setValue(0); // the pen must be changed
    }
   }
}

//return the number "currentPen" of the pen which has just been taken
int takenPen(){
    currentPen = 0 ;

    while (!isWeightDifferent()){
    }
    mCurrentPen = - scale.getGram() + mTot ;

    for (int i=0 ; i<n ; i++){
     if (abs(mCurrentPen - mPen[i]) <= delta/3){
       currentPen = i+1;
     }
    }
    mTot = scale.getGram() ;
    Serial.println(mCurrentPen) ;
    return currentPen ;
}

// compute the maximum of absolute values of the difference of 2 terms of vec
float maxAbsDiffVec (float vec [], int len){
  float max = abs(vec[0]- vec[1]) ;

  for (int i = 0; i < len; i++){
    for (int j = i+1; j < len; j++){
       if (abs (vec[i]-vec[j]) > max){
         max = abs (vec[i]-vec[j]) ;
       }
    }
  }
  return max ;
}
```