ROCHE Clément
COLLURA Matthieu

Projet Main robotisé

26/05/2017 1ière séance

Objectifs :
-Première prise en main de la main robotisé et de l'ensemble des éléments
-Comprendre le fonctionnement de tous les éléments (carte capteur, carte d'interfaçage etc…)
-Faire le mapping entre les 16 canaux du shield servo-moteur et les capteurs pour tester le programme et avoir une visibilité sur l'avancement du projet

•Prise en main des différents éléments du robot :

La **carte capteur**, il s'agit du gant qui possède l'ensemble des capteurs (accéléromètre) et la carte de développement NXP KL25Z.

L'ensemble **d'interfaçage**, il s'agit d'une carte munie du composant PCA9685, qui permet de générer des signaux PWM sur 16 canaux pour contrôler les servo-moteurs (Adafruit PWM/Servo Driver). Et d'une carte NXP KL25Z.

Les **modules Xbee**, sont les modules présents sur le gant et la carte d'interfaçage pour établir la communication entre ces deux éléments.

Nous avons donc, dans un premier temps, câblé tous le système (alimentation USB pour la carte capteur, puis alimentation 5V pour les servo-moteurs) puis connecter quelques pins sur la carte d'interfaçage pour tester la communication entre le gant et la main robotisé.
Nous avons pu constater le bon fonctionnement du système et la communication entre le gant et la main robotisé en manipulant quelques doigts. Nous n'avons pas eu le temps de faire le mapping souhaité, car nous avons rencontré un problème.

•Problème d'alimentation de la carte d'interfaçage

Nous avons constaté au niveau du bornier d'alimentation du driver PWM/Servo Adafruit, un faux contact entre le 5V et le 0V, due aux fils dénudés sur plus d'1cm. Nous nous sommes rendu compte de ce souci plus tard pendant un test, la carte NXP KL25Z de l'ensemble d'interfaçage a donc pris un coup.
Nous avons donc sécurisé l'ensemble des connectiques défaillantes puis récupérer une carte Nucléo F401RE pour remplacer la carte NXP KL25Z.
Nous avons fait quelques modifications au niveau du driver PWM/Servo Adafruit pour que le shield soit adapté à la carte Nucleo F401RE (notamment les ports I2C pour s'interfacer avec le composant

PCA9685, puis les ports série pour la communication Xbee).

Suite à cela, nous ne pouvions plus faire fonctionner le système, notamment au niveau de la communication entre les modules Xbee, et les communications séries.
Nous avons décidé de contacter Kevin JOUAN qui a travaillé sur le sujet l'année dernière, s'il pouvait nous éclaircir sur les détails du système. Directement disponible, il s'est rendu à l'IUT pour que l'on puisse se rencontrer lors de la prochaine séance, pour nous expliquer les détails du projet.

26/05/2017 2ième séance

Objectifs :
-Comprendre l'ensemble des détails du système avec Kevin
-Récupérer les codes sources à jour
-Valider la partie capteur avec l'aide de Kevin
-Comprendre d'où peut venir le problème de communication entre les deux modules Xbee et les ports séries

•Validation de la partie capteur

En utilisant un terminal sur le port série de la carte capteur. Nous avons pu rapidement validé le fonctionnement de la carte capteur et de l'ensemble (capteurs, carte capteur, module Xbee). Nous avons bien observé des commandes envoyées via le port série.

•Comprendre le problème de communication entre les deux modules Xbee et les ports séries.

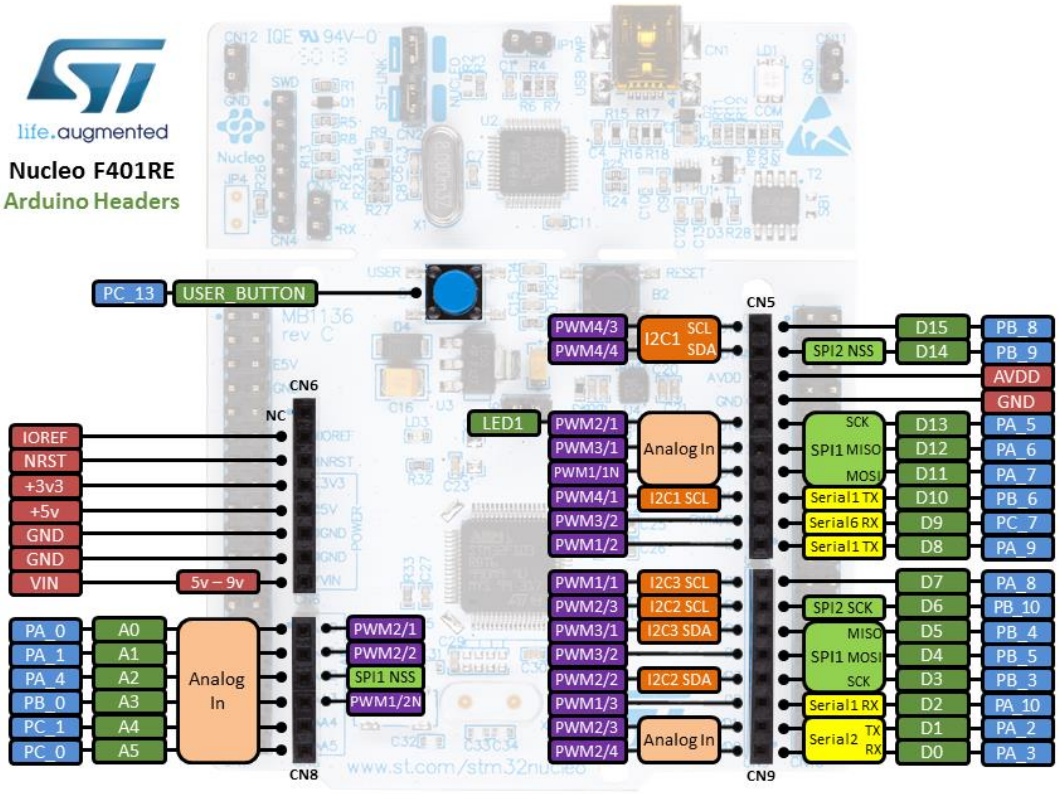
Comme expliqué précédemment, après avoir changé de carte d'interfaçage suite au souci du faux contact, nous avons changé la carte pour une Nucleo F401RE. Même après fait tous les changements nécessaires que ce soit au niveau du shield PWM/servo, ou du code source, pour faire correspondre les ports I2C et ports séries des deux cartes, impossible de faire fonctionner le système après ça.

Avec l'aide de Kevin, nous avons testé que ce soit avec l'oscilloscope ou un terminal l'ensemble des signaux des différents ports série et des modules Xbee.

Nous avons en fin de séance finalement trouver le problème. Il s'agissait en réalité d'un problème au niveau du choix du bus série au niveau de la carte Nucléo. Nous avions choisi le bus Serial2 avec Tx sur la pin PA_2 et Rx sur la pin PA_3. Ce bus est réservé pour un port COM par défaut.
En prenant le Bus Serial1 avec Tx sur la pin PB_9 et Rx sur la pin PA_10, le système a pu fonctionner correctement, nous avons finalement validé la communication entre le gant et la carte d'interfaçage.

Objectifs pour les prochaines séances :

-Recommencer les tests complets avec la nouvelle carte Nucléo
-Faire le mapping des 16 canaux avec l'ensemble des capteurs et des servo-moteurs
-Evaluer les axes d'amélioration et poursuivre le travail de Kevin (faire bouger tous les doigts,



rendre les mouvements fluides etc..)
-Faire fonctionner le module Bluetooth pour faire communiquer le système avec une application de supervision
-Etablir un concept pour commencer la partie application Android pour le système de supervision (imaginé une interface etc…)

31/05/2017 3ième séance :

Objectifs :

-Modifier le shield PWM Adafruit pour faire fonctionner correctement le bus série de la nouvelle carte (Nucleo)
-Faire tous les tests concernant le changement de bus (au niveau de la communication série)

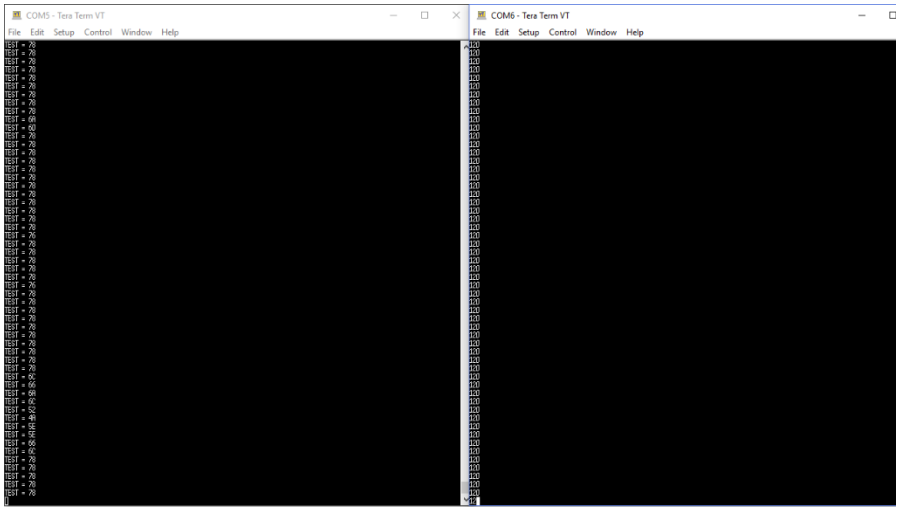
-Faire fonctionner correctement les deux modules Xbee (au niveau de la synchronisation)
-Tester tous le système en condition réelle

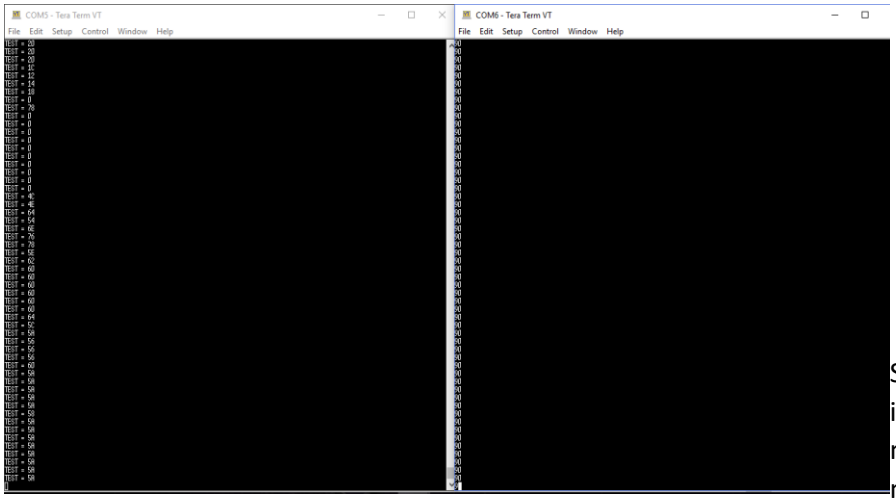
•Modification du shield PWM

Le shield était initialement adapté pour la carte KL25Z, comme nous avons dû changer de carte pour une carte Nucléo, nous avons dû modifier le shield au niveau des soudures pour faire fonctionner le bon bus série.

•Test concernant le changement de bus

Après avoir modifié le shield, nous avons testé avec un terminal la communication série du nouveau bus. Tout est OK de ce côté-là. Nous avons donc mis en œuvre les modules Xbee connecté sur ce bus série. Nous avons pu tester le fonctionnement à l'aide d'un terminal :





Sur le port COM 5, les informations reçues par le module Xbee de la carte moteur (figure de gauche). Sur le port COM 6, les informations envoyées par le modules Xbee de la carte gant (figure de droite).

On remarque, que lorsque le gant envoie « 90 » on reçoit bien 0x5A. La communication entre le gant

et la carte moteur fonctionne bien.

•Test en condition réelle

Nous avons chargé les derniers programmes et pu faire fonctionner correctement deux doigts avec la nouvelle carte et les modifications sur le shield effectuées. Nous avons pu commencer à faire le mapping des bornes PWM de la carte moteur.

Objectifs pour les prochaines séances :

-Modifier le code pour faire fonctionner les 3 accéléromètres indépendamment (pour rendre le mouvement du doigt plus fluide)
-Modifier le code pour faire fonctionner tous les doigts
-Modifier le hardware côté carte gant, pour le module bluetooth
-Commencer l'application Android pour récupérer des informations sur un port série.

02/06/2017 4ième séance :

•Modification du code pour faire fonctionner tous les doigts

Après avoir fait fonctionner un seul doigt, nous avons modifié les codes sources pour pouvoir faire fonctionner l'ensemble des doigts simultanément. En rajoutant notamment les commandes capteurs et moteurs correspondantes.
Nous avons testé le bon fonctionnement du système en faisant correctement bouger les doigts.
Nous avons remarqué cela dit un appel de courant lorsque tous les doigts sont sollicités en même temps, ce qui conduit à un fort ralentissement du système.
Pour le moment, nous avons seulement relevé le problème, aucune action n'est prévue pour l'instant.
Nous avons aussi rencontré un problème sur le pousse, qui utilise un plot PWM différent de ce prévue par la carte PWM. Nous testerons la sortie de ce plot pour voir si la PWM est ok.

•Modification du code pour faire fonctionner les 3 accéléromètres indépendamment

Nous avons effectué des modifications sur le code capteur pour que les valeurs de chaque accéléromètre du gant soit pris en compte indépendamment. A l'origine, la valeur des accéléromètres du bout des doigts ne sont pas pris en compte, c'est la valeur du moteur à la base du doigt qui est propagé sur les autres moteurs.

Lors de nos tests, nous avons rencontré un problème de blocage des moteurs a une position intermédiaire. Nous allons investiguer plus en détail la partie accéléromètre/SPI ainsi que la communication Xbee et l'ensemble des informations qui transitent entre la carte capteur/moteur

pour mieux comprendre le problème.
L'analyse du code nous a permis de beaucoup mieux comprendre les différentes parties logicielles, nous pensons pouvoir résoudre ce problème pour la séance prochaine

•Module Bluetooth

Nous n'avons pas pu attaquer la partie hardware pour le module Bluetooth qui nous servira pour la supervision du système.
Nous avons choisi la solution de positionner le module Bluetooth sur la carte capteur, et d'ouvrir un second port série. Nous copierons l'ensemble des données séries présentes en « Input » / « Output » de l'Xbee.

Objectifs pour la prochaine séance :

-Tester la PWM sur le plot où est connecté le pouce
-Refaire des tests/modifications dans le code capteur pour faire fonctionner correctement tous les accéléromètres
-Faire les modifications hardware pour le module Blueetooth
-Commencer l'application (nous n'avons pu commencer cette séance)

19/06/2017 5ième séance :

•Test de la PWM sur le plot du pouce

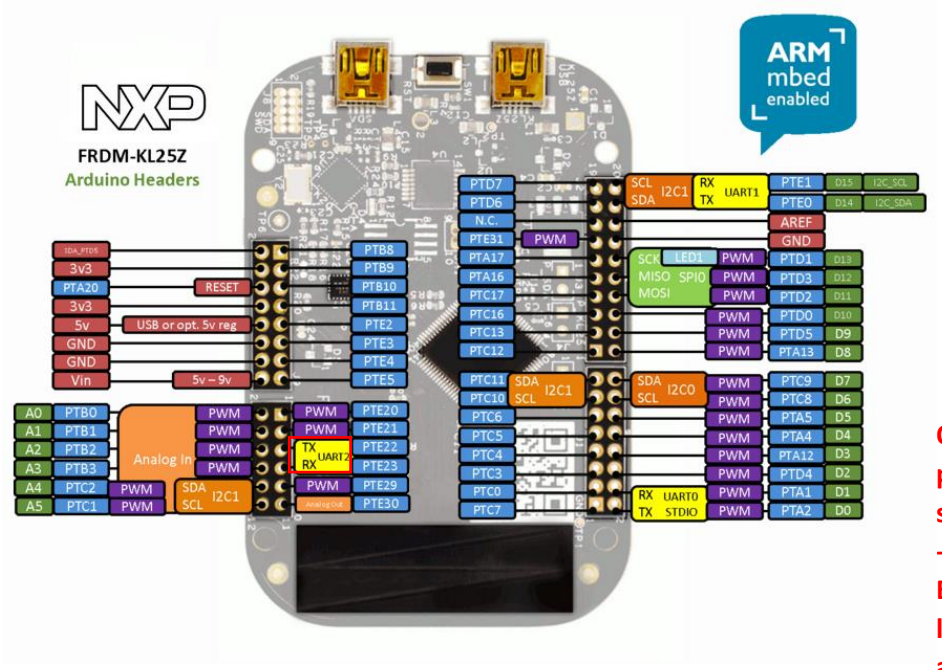
Nous avons retesté a l'oscilloscope l'ensemble des sorties PWM. Le plot où est connecté le pouce ne fonctionne clairement pas. Il s'agit d'un plot qui a été rajouté car il n'y avait plus de chanel disponible sur le shiel PWM. Nous allons prévoir une modification mais ce n'est pas prioritaire face au développement de l'application Android.

•Modification du code pour faire fonctionner tous les accéléromètres

Nous avons modifié le code pour envoyer les données de tous les accéléromètres indépendamment, pour rendre le mouvement plus fluide pour les différentes phalanges. Nous avons fait quelques tests : que ce soit sur 4 doigts ou bien un seul doigt, lorsque l'on envoi toutes les données de chaque accéléromètre, le doigt se bloque dans une position intermédiaire. Nous avons quelques pistes : la quantité de données à transmettre est beaucoup plus importante en envoyant les valeurs de tous les accéléromètres.
Nous pensons aussi que ce problème peut venir de la fonction d'interruption côté module Xbee

•Modification hardware pour le module Bluetooth

Nous avons constaté que la modification hardware pour accueillir le module Bluetooth avait déjà été mis en place. Ce module sera placé sur la carte capteur (gant), sur un bus série UART. Le bus UART1 étant déjà utilisé pour le module Xbee, nous utiliserons le bus série UART de la carte FRDM-KL25Z. Nous avons fait quelques modifications dans le code pour pouvoir utiliser le module.



la module testé module téléphone -Commencer l'application Android

Objectifs pour prochaine séance : -Récupérer un Bluetooth et l'appairage du avec un

•Test de l'appairage du module Bluetooth

Après avoir positionné le module Bluetooth sur la carte capteur, on constate bien que le module Bluetooth est correctement alimenté. On remarque une LED qui clignote rapidement sur le module. Après quelques recherches, cette signalétique indique que le module est prêt à être appairé avec un téléphone par exemple.
Actuellement, nous n'arrivons pas à nous appairer au module Bluetooth. Nous avons des pistes, notamment concernant la broche « Enable » du module HC-06. Nous allons aussi préparer un montage pour tester rapidement le module Bluetooth sans utiliser le shield capteur. Peut-être que notre problème vient de la carte capteur.

•Application Android

Nous avons très rapidement commencé l'application Android pour pouvoir récupéré des données avec l'aide du module Bluetooth. Nous allons dans un premier temps, essayer de récupérer des données avec « Bluetooth Terminal »

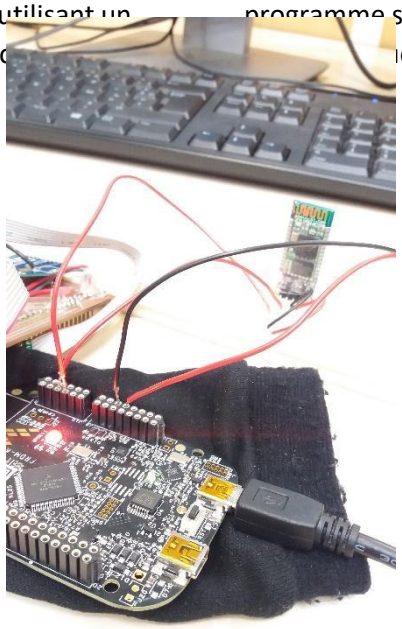
Objectifs pour la prochaine séance :

-Faire le montage pour tester le module Blueetooth sans la carte capteur (gant)
-Tester la transmission de données avec le module Bluetooth sur la carte capteur
-Modifier le code pour envoyer les données des accéléromètres avec la liaison Bluetooth

20/06/2017 6ième séance :

•Faire le montage pour tester le module Bluetooth sans la carte capteur

Lors de la dernière séance, nous avons eu du mal à mettre en place le module Bluetooth, notamment au niveau de l'appairage du module avec un téléphone. Nous avons donc fait le montage suivant, en utilisant que la carte FRDM-KL25Z (sans le shield capteur où est positionné le plot pour le Bluetooth). Nous avons par la même occasion changer de module Bluetooth). En utilisant un programme simple chargé dans la carte, et avec l'aide d'un terminal Bluetooth, on peut voir que l'on reçoit bien des données :



•Tester la transmission de données avec la carte capteur

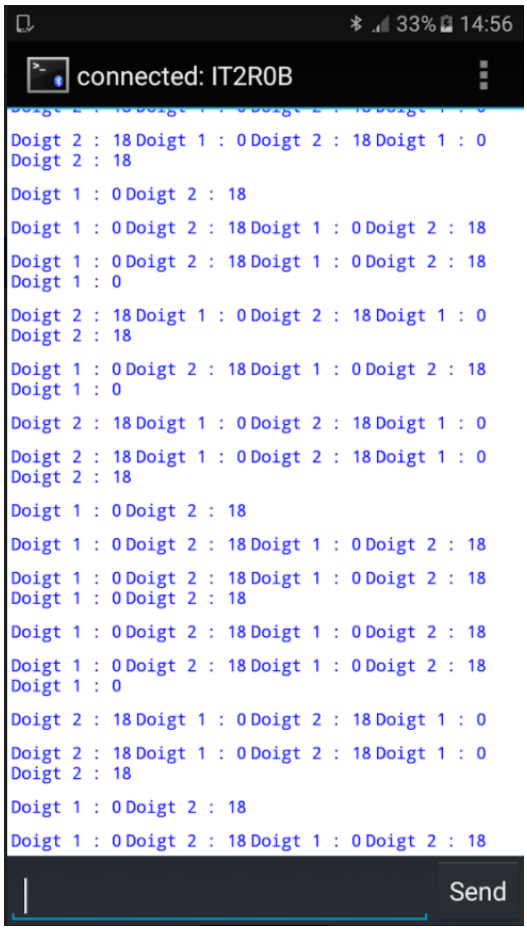
En utilisant le même programme que précédemment, nous avons pu faire les mêmes observations.

Nous arrivons à transmettre des données avec le module Bluetooth. Notre problème venait donc bien du module Bluetooth défaillant.

•Modifier le code pour envoyer les données des accéléromètres avec la liaison Bluetooth

Nous avons pu faire des modifications pour envoyer des données sur un terminal Bluetooth. C'est donc un bon début car notre module fonctionne correctement et est capable de transmettre les bonnes données. Nous avons eu des problèmes à mettre en oeuvre cette transmission de donnée car elle rentrait en conflit avec la transmission Zigbee du contrôle des doigts.

Nous avons finalement résolu le problème et le contrôle des doigts est toujours possible tandis que le module bluetooth transmet les bonnes informations. Il faut maintenant réceptionner et traiter ces informations via une application android sur portable, et non via un terminal bluetooth.



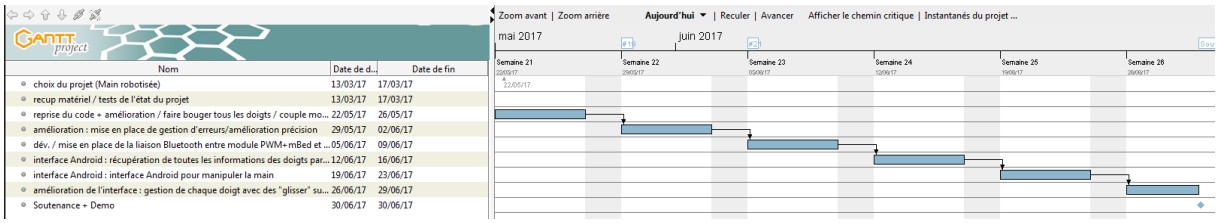
← Réception des données de plusieurs doigts via bluetooth.

•Créer une application Android Studio

Nous avons commencé à mettre en oeuvre une application permettant de réceptionner le bluetooth de notre module. Pour l'instant, l'interface est simple et ne comporte qu'un simple bouton ON/OFF qui active ou désactive le bluetooth en demandant l'autorisation à l'utilisateur.

Nous avons également mis en oeuvre la recherche des appareils disponibles et la manière de s'appairer mais nous n'avons pas eu le temps de le tester.

Vérification des tâches effectuées par rapport au diagramme de Gantt établi en début de projet



Nom	Date de d...	Date de fin
choix du projet (Main robotisée)	13/03/17	17/03/17
recup matériel / tests de l'état du projet	13/03/17	17/03/17
reprise du code + amélioration / faire bouger tous les doigts / couple mo...	22/05/17	26/05/17
amélioration : mise en place de gestion d'erreurs/amélioration précision	29/05/17	02/06/17
dév. / mise en place de la liaison Bluetooth entre module PWM+ mBed et ...	05/06/17	09/06/17
interface Android : récupération de toutes les informations des doigts par...	12/06/17	16/06/17
interface Android : interface Android pour manipuler la main	19/06/17	23/06/17
amélioration de l'interface : gestion de chaque doigt avec des "glisser" su...	26/06/17	29/06/17
Soutenance + Demo	30/06/17	30/06/17

Par rapport au diagramme de Gantt, nous sommes légèrement en retard. Conformément à ce que nous avions mis, nous avons réussi à reprendre en main le projet, et à faire bouger tous les doigts à l'exception du pouce.

Nous avons également amélioré la précision afin que les doigts en mouvement ne rentrent pas en collision les uns avec les autres. En revanche, la gestion d'erreurs reste assez vague.

Pour finir, nous avons mis en place la liaison bluetooth après une étude approfondie de la carte et commencé l'application Android permettant de communiquer.

Nous sommes donc en retard par rapport à nos plans, mais nous devrions avoir largement le temps de terminer cette partie application android.

Objectifs pour la prochaine séance :

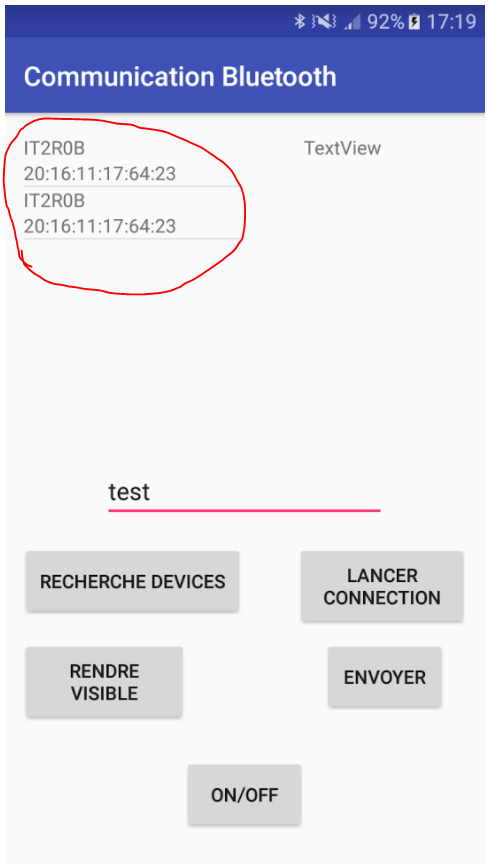
- Terminer l'application android et établir une première communication permettant de recevoir les données du module bluetooth.

- **Commencer à établir une liaison permettant d'envoyer des ordres au module pour commander la main.**
- **Travailler l'ergonomie de la main afin d'avoir un environnement de travail pratique et sans danger.**

21/06/2017 7^{ième} séance :

•Application Android

Nous avons bien progressé dans l'application Android et l'interface est maintenant assez complète. La recherche d'appareils disponibles fonctionne bien et nous sommes capables de nous appairer avec le module bluetooth. En revanche, nous n'avons pas réussi à lire les données envoyées par le module à travers l'application.



← en rouge, le module bluetooth recherché qui envoie les données.

Soupçon de pourquoi ça ne marche pas : la doc Android concernant le bluetooth inclut une fonction UUID qui est en lien avec un com client-serveur. Cet UUID doit être déterminé non aléatoirement côté serveur, donc application, car on utilise un module bluetooth série. Or UUID se génère tout seul aléatoirement.

Notre problème vient sans doute de là.

•Hardware

Nous avons repris bon nombre de soudure afin de simplifier la main. Nous avons commencé à créer des extensions de câbles pour éviter de tirer les fils lors de l'utilisation de la main. Cela simplifiera beaucoup son utilisation à l'avenir.

Objectifs pour la prochaine séance :

- Etablir une première communication permettant de recevoir les données du module bluetooth.

- **Commencer à établir une liaison permettant d'envoyer des ordres au module pour commander la main.**
- **Terminer les extensions**
- **Commencer à créer une doc digne de ce nom à grands renforts de schémas électriques sur chaque carte.**

22/06/2017 8ième séance :

•Application Android

Nous avions précédemment un problème sur les UUID lors des connexions bluetooth avec Android. Ce sont des identifiants utilisés et nécessaires pour établir une connexion entre 2 appareils. Ce problème a été résolu et nous as permis d'utiliser le programme Sample contenu dans Android. Maintenant, nous disposons d'une application capable d'établir une connexion entre deux appareils bluetooth, d'appairer et d'afficher les informations du module. Elle fait office de terminal.

Maintenant, nous nous sommes heurtés à une difficulté dû au fait que le module établit une communication série et que nos caractères sont difficiles à traiter en tant que tel, puisque le module n'envoie que des caractères et non des variables déterminés.

Il va donc falloir parser notre flux de données indistincts afin de pouvoir les traiter enfin comme des int et extraire les données de chaque doigt.

Objectifs pour la prochaine séance :

- **Etablir un bon parseur pour traiter le flux de données.**
- **Commencer à établir une liaison permettant d'envoyer des ordres au module pour commander la main.**
- **Commencer à créer une doc digne de ce nom à grands renforts de schémas électriques sur chaque carte.**

23/06/2017 8ième séance :

Pour cette séance, nous nous sommes séparés pour traiter les deux parties importantes de notre projet, à savoir la partie commande de la main, plus hardware, et la partie application Android.

•Commande de la main

Les accéléromètres fonctionnent. Nous avons pu les tester un à un, et chacun donnait un bon résultat. Malheureusement, à cause de la colle qui obstrue la carte, nous n'avons pas pu vérifier totalement l'état de la carte, car nous avons remarqué que certains fils avaient tendance à rompre, tout simplement. Ainsi, j'ai réussi à mettre en place les trois accéléromètres pour un doigt, mais avec un succès tout relatif. C'est à dire que les capteurs envoyaient les bonnes données, et que les moteurs bougeaient bien en conséquence. Mais certains détails devaient être améliorés, car lorsque nous baissons notre doigt tout en le gardant droit (donc une seule phalange en action), les positions des 3 accéléromètres sont modifiés tout de même, et la main recroquevillait entièrement son doigt.

Malheureusement, la main s'est mis en croix pour une raison absolument inconnue. Ce genre de panne lui arrive souvent et il n'est dès lors plus possible de lui arracher le moindre mouvement. Ce n'est pas la première fois que ça lui arrive et nous sommes constamment obligés de réinjecter nos programmes de back-up pour récupérer le contrôle. Même en effaçant toutes les modifications et en revenant au programme de base, ça ne marche pas.

Nous ne savons pas du tout pourquoi cette panne se produit. Néanmoins, un doigt a bien fonctionné tel qu'il l'avait été prévu par ses créateurs, et c'est sur ce programme là que l'on se basera pour continuer.

Du reste, afin de limiter au maximum les à-coups et vibrations des doigts, j'ai introduit des masques afin de réduire la précision des mesures des accéléromètres. Cette réduction de précision est bénéfique et permet une meilleure fluidité.

23/06/2017 9ième séance :

•Commande de la main

La main s'est finalement remise en marche normalement sans avoir eu à changer de programme, ce qui rend d'autant plus inexplicable ce problème.

Finalement, le problème du trop-plein de données et des incohérences de plusieurs accéléromètres est dû à un simple problème spatial. L'accéléromètre à l'intégralité de ses valeurs comprise dans un angle entre 0 et 90°. Lorsque l'accéléromètre est à 90°, il renvoie la valeur 120 qui signifie que le doigt est levé. Lorsque l'accéléromètre est à 0°, il renvoie la valeur 0. Ceci est le résultat du traitement du code mbed que nous avons repris.

Or quand nous plions notre doigt, cela fonctionne pour l'accéléromètre de la phalange de base car nous ne pouvons naturellement pas faire plus ou à peine plus que 90°. Par contre, les deux phalanges plus hautes peuvent accomplir de plus grands angles, au-delà des 90°. Or, au-delà des 90°, l'accéléromètre est perdu et renvoie… 120, donc l'exact opposé de ce qu'il doit faire.

Comme la main fonctionnait intégralement sur la base de cette première phalange, sur laquelle ce problème ne s'appliquait pas, l'erreur n'avait pas été repérée plus tôt.

Il y a donc un nouveau traitement à faire prenant en compte ce nouveau paramètre qui n'était pas traité auparavant. Cela m'a permis de commencer à esquisser une documentation à ce sujet, qu'il faudra que je rédige plus proprement.

Malheureusement, un nouveau problème est survenu. La carte Adafruit a subitement cessé de fonctionner pour une raison à nouveau inconnue. La carte Nucleo a grillé également. Après une rapide recherche de panne, il est apparu que le GND et le V+ de la carte Adafruit étaient reliés. Pourtant, aucun point de soudure ne fait de pont. C'est donc le composant PCA9685 qui a dysfonctionné et entraîné le grillage de la carte Nucleo, ce que nous déplorons. Heureusement, nous avions pris nos précautions et il nous restait une carte Freescale à exploiter. J'ai donc refait les soudures et les mappings pour adapter la nouvelle carte à son usage.

En revanche, nous n'avons pas de carte Adafruit de rechange, ce qui est très problématique, à une semaine de la soutenance.

C'est vraiment dommage, car nous commencions à en voir le bout.

•Application Android

Nous avons pu commencer la partie traitement des données que nous recevons actuellement sur notre application. Nous avons dans un premier temps construit un ensemble de méthode pour mettre en place un **parser** de flux de données, pour récupérer les différentes informations des doigts. Nous avons passé du temps notamment avec Mme Maillefert, sur une problématique de traitement des données que nous avons mis en place. En effet, il nous était dans un premier temps, impossible de correctement traiter les informations.
Le problème venait globalement du fait, que les données n'étaient pas correctement formatées côté mbed (donc envoyées avec une logique qui n'était pas pertinente).

Notre second problème vient du fait que nous avons essayé de mettre ne place des méthodes et des classes en programmant de la même façon qu'en C. Nous devrons donc par la suite, essayer d'utiliser plutôt des fonctions et des classes Java, afin d'établir un traitement pertinent.

Objectifs pour la prochaine séance :

- **Trouver un moyen de remplacer la carte Adafruit.**
- **Ecrire un code propre pour tous les accéléromètres, prenant en compte le problème évoqué afin de l'esquiver.**
- **Continuer à renseigner la documentation esquissée.**
- **Formater les données envoyées**
- **Remplacer/Modifier les méthodes de parsing**

26/06/2017 10ième séance :

•Application Android

Nous avons commencé par formater les trames envoyées par le mbed pour clarifier l'envoi des données. En utilisant des identifiant de types clés-valeurs, nous avons simplifié l'envoi des données :

Trame typique : avec I, M et A, les clés pour Index, Majeur et Annuaire et les valeurs de la position de chaque doigt et X et Y les délimiteurs de fin de valeurs.

I{Value1} X M{Value2} Y A{Value3}

Nous avons donc ensuite remplacé les méthodes mises en place précédemment par l'utilisation de fonctions et de classes Java. Notamment en utilisant l'outil :

```
Int= Integer.parseInt(String str);
```

Nous avons à présent, avec ce système, correctement parsé et traité les données, notamment avec la détection des délimiteurs des clés/valeurs (utilisation de l'outil substring).
Nous avons donc globalement dans notre classe qui gère la construction des messages :

```
mConversationArrayAdapter.add(readMessage); //créer le flux de données

recup_index = readMessage; //récupération du flux dans une chaine
recup_index = recup_index.substring((recup_index.indexOf("I"))+1,
recup_index.indexOf("X")); //détection des délimiteurs
Log.d(getClass().getName(), "valeur index char = " + recup_index);
index = Integer.parseInt(recup_index); //parser la chaine pour transformer
les trames en valeurs entières
```

Nous rencontrons toutefois un problème de synchronisation entre les données envoyées par le mbed et le traitement côté Android, nous avons encore quelques plantages de l'application lors de ces problèmes de désynchronisation. Nous allons donc lors de la prochaine séance, régler l'ensemble de ces problèmes, en améliorant une seconde fois le format des trames envoyées et le traitement appliqué.

Objectifs pour la prochaine séance :

- **Modifier le formatage des trames pour gérer le problème de synchronisation**
- **Modifier l'algorithme de traitement pour gérer le problème de synchronisation**
- **Continuer le développement et les tests sur la création d'un Intent et la communication des données traitées avec une progressBar**

27/06/2017 11ième séance :

•Application Android

L'objectif de cette séance était de pouvoir mettre en place une communication entre les différentes classes du projet Android. Nous voulions pouvoir récupérer le flux de données pour créer des variables contenant en temps réel la position du doigt, puis alimenter une progressBar, pour donner un ordre d'idée de la position du doigt sur l'UI.

Nous avons fait différents tests :

-Essai de la création de plusieurs Intent et de méthodes onReceive pour transmettre ces données : après avoir mis en place l'ensemble des outils que nous avons vu en TP, nous nous sommes rendu compte d'un problème. Nous n'avons pas réussi à transmettre les données. Après plusieurs recherches, cette solution est utile uniquement si on veut transmettre des données entre plusieurs Activités. Dans notre cas, nous n'avons qu'une seule Activité.

Main Activity :

```
final RecepteurAngle capteur = new RecepteurAngle();
IntentFilter(RecepteurReponse.ACTION_REPONSE));
registerReceiver(capteur, new IntentFilter(RecepteurAngle.ACTION_REPONSE));
```

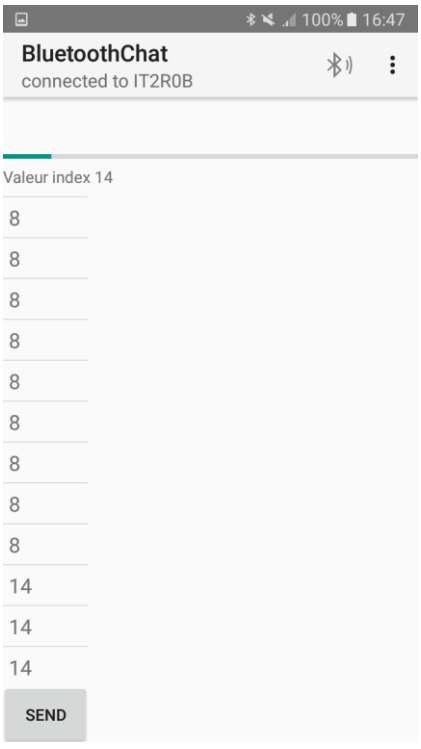
```
public class RecepteurAngle extends BroadcastReceiver{
    public static final String ACTION_REPONSE = "Message passé";

    private int valeur_index;
    private int valeur_index_status=0;
    final ProgressBar index_progress = (ProgressBar)
findViewById(R.id.progressBar3);
```

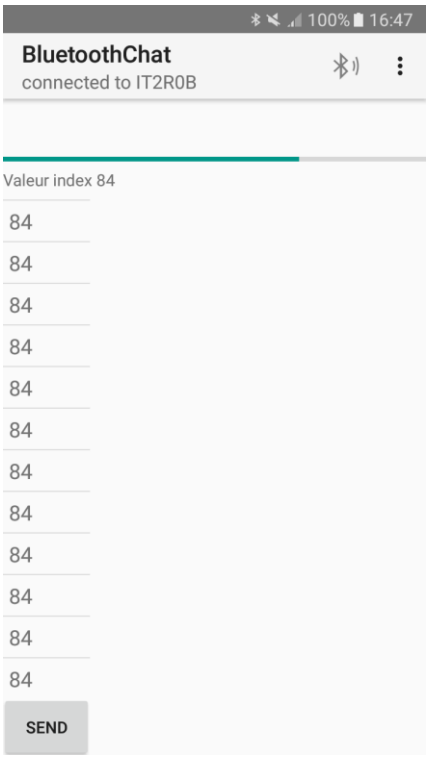
```
    @Override
    public void onReceive(Context context, Intent intent){

        valeur_index =
intent.getIntExtra(BluetoothChatFragment.index_reel,-1); //modifier pour
récupérer l'entier "index" de la classe BlueetoothChatFragment
        valeur_index_status = (int)valeur_index;
        resultat_index.setText("Index="+ Integer.toString(valeur_index));
        index_progress.setProgress(valeur_index_status);
        Log.d(getClass().getName(), "Valeur index status = " +
valeur_index_status);
        Log.d(getClass().getName(), "Valeur index = " + valeur_index);
    }
}
```

Classe :

```
broadcastIntent.setAction(MainActivity.RecepteurAngle.ACTION_REPONSE);
broadcastIntent.putExtra(PARAM_OUT_MSG, index_reel);
context.sendBroadcast(broadcastIntent);
```

-Nous avons appris et pu expérimenter, que des objets Android (type textView, progressBar etc…) peuvent être déclarés en ***static.*** C'est-à-dire, en créant des objets textView et progressBar en static, nous pouvons manipuler l'objet directement à partir de la classe qui nous intéresse, où est stockée la variable.
Nous affichons donc cette variable sur l'interface, elle représente la valeur de la ProgressBar en direct. Il nous suffisait à ce moment-là de mettre en place la ProgressBar telle que nous l'avions vu en cours d'Android pour l'incrémenter.
La ProgressBar est donc fonctionnelle pour un doigt et informe visuellement de la valeur d'un accéléromètre, ici celui de base de l'index.





Objectifs pour la prochaine séance :

- **Modifier le formatage des trames pour gérer le problème de synchronisation**
- **Modifier l'algorithme de traitement pour gérer le problème de synchronisation**
- **Ajouter des progressBar pour les autres doigts**
- **Modifier l'UI**