INSA TOULOUSE

PROJET TUTEURE

Mesure de la fréquence cardiaque sur un bonnet de natation et transmission sans fil aquatique

4ème Année AE/SE & AE/IS

-

Réalisé par :
Ghazal ADNANI Julien CHOUVET
Yacine ESCALANTE Laure LAMBERT

Projet proposé par : Mrs. Pascal ACCO & Xavier TOLZA
Tuteur bibliothécaire : Mme. Maëlle CADIOU

Sommaire

Remerciements

Nous tenons à remercier toutes les personnes ayant contribué à l'avancement de ce projet tuteuré.

Nous souhaitons de plus remercier chaleureusement M. Di Mercurio dont l'aide a été très précieuse pour ressouder les fils de la carte Newton et ce, toujours avec un grand sourire.

Ensuite, nous souhaitons également remercier Mr Grisolia qui nous a fourni le module Bluetooth Low Energy et nous a fait confiance.

Enfin, nous remercions nos tuteurs Mr Acco et Mr Tolza pour leur confiance, leur disponibilité et leur soutien dans ce projet.

Introduction

Ce rapport de projet tuteuré a été produit à la suite de la demande de l'entreprise Swimbot souhaitant ajouter une fonctionnalité à son boitier de natation.

Le but de ce dernier étant de corriger le nageur lors de divers exercices, il est ainsi composé d'un boîtier à positionner sous le bonnet derrière la tête, et de deux écouteurs qui s'accrochent au bonnet au niveau des oreilles. Ainsi, après avoir calibré quelques paramètres au début de l'entraînement, le nageur reçoit un appel sonore lui indiquant si sa position de tête est incorrecte et lui dictant des exercices.

Le projet proposé est alors de perfectionner ce dispositif en y ajoutant un capteur de fréquence cardiaque dont le résultat devra s'afficher sur le boitier via une application Android. Dès lors, plusieurs problématiques majeures apparaissent : créer un capteur capable d'être immergé sous l'eau, créer une chaîne d'acquisition de données permettant la communication entre le capteur et l'accessoire Swimbot, intégrer au niveau logiciel la mesure de la fréquence cardiaque au sein du système déjà existant.

Dans l'ensemble, ce projet permet une application des connaissances obtenues dans les cours de programmation, électronique et de gestion de projet.

Enfin, vous trouverez joint à ce rapport un dossier contenant les codes développés sur Arduino et en Java Androïd pour la réalisation de ce projet.

I. Cahier des charges

I.1. Contexte

Travaillant au sein de l'entreprise SWIMBOT, Mr TOLZA, doctorant au LAAS, a proposé ce sujet dans le but d'ajouter une fonctionnalité supplémentaire au système embarqué Swimbot. En effet, ce projet avait pour objectif final d'aboutir à la réalisation d'un relevé de mesures de fréquence cardiaque en milieu aquatique afin de l'afficher par la suite sur l'écran d'un système embarqué nommé « Swimbot » placé dans un bonnet de bain. Ceci permettant alors au nageur de connaitre sa fréquence cardiaque moyenne à la fin d'un exercice proposé par le boitier Swimbot ou à l'entraîneur de connaître la fréquence cardiaque de son nageur en temps réel via une application mobile en interaction avec le Swimbot.

Au vue de ces problématiques, le sujet était décomposé en deux grandes parties : une orientée hardware sur la conception d'un capteur capable d'aller dans l'eau, et une orientée sur la partie software, c'est-à-dire gérant le lien entre le capteur et le système embarqué.

Nous avons alors fait le choix de nous orienter vers cette deuxième partie car nous souhaitions apprendre à développer une liaison sans fil et nous familiariser avec le développement sous Android.

I.2. Résultats des recherches bibliographiques

Nous avons réalisé des recherches bibliographiques sur l'ensemble du sujet en nous penchant d'avantage sur la connexion sans fil. Dans un premier temps, nous avons travaillé sur la communication sous l'eau. Toutefois, comme nous pouvons le lire sur le graphique, les ondes sont absorbées sous l'eau. En effet, d'après la figure ci-dessous, nous constatons que plus la fréquence de transmission n'augmente, plus l'absorption est importante.

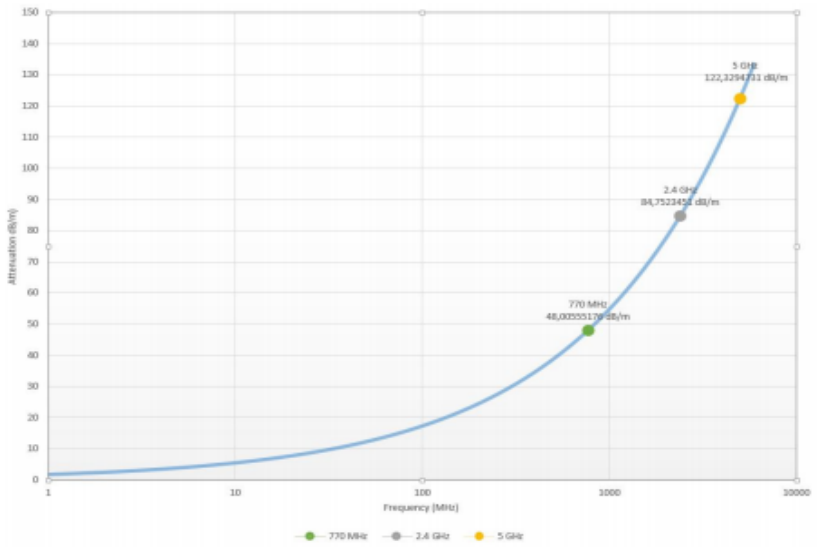


Figure 1 - Graphique de l'atténuation dans un milieu aquatique en fonction de la fréquence d'émission

La seule possibilité qui s'offrait à nous était donc l'utilisation du protocole LoRA car il utilise une basse fréquence d'émission au contraire du Wi-Fi ou des ondes radio. En effet, cette dernière est autour des 866 MHz et n'est donc potentiellement pas absorbée.

L'interrogation principale de cette communication était alors de savoir si le débit envoyé était suffisant pour permettre à un entraîneur d'accéder aux fréquences cardiaques de son nageur en temps réel via une application Androïd exécutée sur smartphone.

I.3. Objectifs et finalités

Pour mener au mieux ce projet, nous avons décidé de définir des objectifs croissants à réaliser et ne pas commencer par la mise en place de ce protocole en premier. En effet, ce protocole demande une connexion sans fil et n'ayant aucune connaissance dans l'établissement d'une connexion, nous avons décidé de commencer par une connexion filaire.

Néanmoins, cette première étape nous a permis d'analyser et de comprendre le fonctionnement des éléments communs aux deux types de connexion tels que la carte de développement Newton, le capteur de fréquence cardiaque.

La suite logique aura été de partir sur une connexion sans fil « aérienne » pour assurer une preuve de conception avant de passer sur du LoRA pour s'orienter sur du milieu aquatique.

II. Description des éléments majeurs

Pour mettre à bien ce projet, nous avons eu à notre disposition une carte Newton 2, carte étant spécialement adaptée pour programmer des codes destiné à des applications Android. Le but a donc été de développer notre code sur cette carte, pouvant être considérée comme l'équivalent de l'appareil Swimbot, mais avec des contraintes d'accès en moins. En effet, en utilisant la carte Newton, nous avons eu la possibilité de nous connecter directement sur des interfaces de communication interne.

II.1. La carte Newton 2

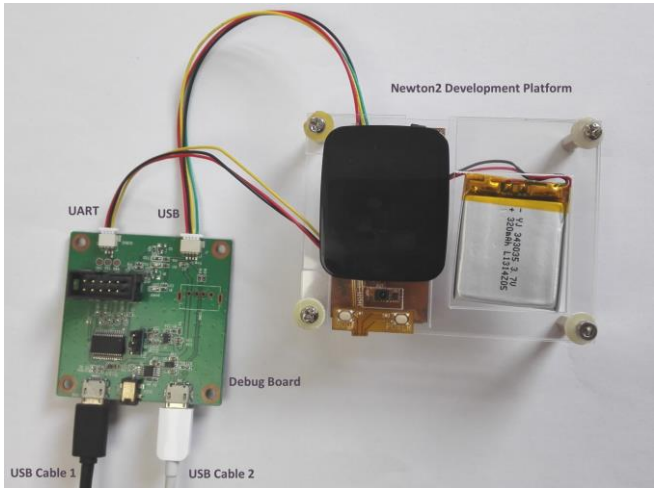


Figure 2 - Carte de Newton 2

Cette carte de développement possède deux accès : un par un port USB standard et un autre en mode DEBUG. Ce dernier nous permettant d'accéder au terminal de la carte.

Cette carte est ainsi composée de deux éléments :

- Un écran tactile de petite dimension à travers lequel nous pouvons naviguer, exécuter des applications Androïd, …
- Une carte de débogage permettant l'accès au terminal interne de l'appareil via une connexion en série.

Cependant, le bash de cette carte de développement disposant de peu d'applications et de commandes, nous avons été limités dans l'utilisation des commandes shell.

Nous avons été ainsi surpris de constater que les autorisations de lecture/écriture ainsi que celles d'exécution ne nous étaient accordés que sur quelques rares dossiers. De même, la commande *chmod* n'étant pas reconnue, nous n'avons pas pu l'utiliser. Plus généralement, nous avons du « composer » tout au long de notre projet avec de nombreuses spécificités de ce bash, qui, très souvent, nous limitaient dans sa pleine utilisation.

Nous avons alors du nous adapter à ces contraintes et spécificités voire les contourner lorsqu'elles étaient trop fortes, et ce dans le but d'atteindre les objectifs fixés.

II.2. Le langage orienté objet

Afin de développer une application Android, nous avons fait appel à l'IDE Android Studio (IDE pour Integrated Development Environment), développé par Google et pour lequel nous avons du apprendre un nouveau langage de programmation, le java, et nous former aux librairies Android.

Ce langage de programmation, différent du Java classique, nous a demandé de comprendre la logique permettant de réaliser une application mobile et de s'adapter à une nouvelle manière de coder. Le site *Android Developper* fut notre site de référence pour coder notre application. En effet, ce dernier regroupant toute la documentation nécessaire à la compréhension des activités, méthodes, etc. de ce langage, il nous fut d'une précieuse aide pour réaliser nos applications finales.

II.3. Le capteur

Notre choix ayant été de nous orienter vers la partie software, nous avons décidé, notamment lors de notre recherche bibliographique, de commander un capteur de fréquence cardiaque : le Pulse Sensor, qui à l'aide d'une LED infrarouge est capable de détecter le pouls.

L'un des avantages à faire appel à ce capteur était la présence d'un code en open source, c'est-à-dire accessible par tous, fourni par les concepteurs de ce capteur. Cela a ainsi été d'une grande aide et d'un gain de temps précieux dans la conception de notre code pour l'acquisition et le traitement des données du capteur.

III. Acquisition du pouls

Nous avons vu précédemment que le capteur Pulse Sensor avait été choisi pour l'acquisition des mesures. Ce dernier s'appuie sur un principe de l'optoélectronique pour mesurer la fréquence cardiaque.

Pour ce faire, il fait appel à une LED qui envoie un signal lumineux. Puis, un capteur de lumière mesure l'intensité de la lumière reçue après réflexion du signal envoyé sur le sang présent dans le doigt. En effet, selon l'effort physique effectué, notre cœur va pomper du sang plus ou moins fort et la quantité de sang présente dans notre doigt variera en fonction de ce pompage. Ainsi, plus il y aura de sang présent dans le doigt et plus la lumière envoyée sera absorbée et l'intensité lumineuse reçue par le capteur faible. La valeur du

signal analogique envoyée par ce capteur est ainsi directement liée à cette intensité (plus elle sera faible et plus la valeur analogique envoyée sera forte).

De plus, ce capteur présentait deux avantages :

- Une facilité de prise en main.
- Sa compatibilité avec l'Arduino et notamment la présence d'un code déjà implémenté gérant le filtrage du signal reçu et sa conversion d'analogique à numérique (et donc en battements par minute).

L'une des premières étapes du projet a alors été d'analyser et de comprendre ce code afin de s'en inspirer dans notre acquisition.

Nous avons ainsi analysé les deux codes fournis par Pulse Sensor. L'un ayant pour but de faire clignoter une LED en fonction du battement cardiaque mesuré. Le second, lui, affichant en temps réel sur un moniteur série mais aussi graphiquement sur un traceur série, les mesures de la fréquence de l'utilisateur.

Sur la figure 3, nous pouvons observer les résultats affichés par le moniteur série :

1. la première le Signal qui, après conversion, donne la mesure en bpm.
2. la deuxième valeur l'IBI qui correspond à l'intervalle de temps qu'il y a eu entre deux bonnes mesures.
3. la troisième valeur renvoyant la BPM, c'est-à-dire le nombre de battements par minute (ici 86 et 94 bpm).

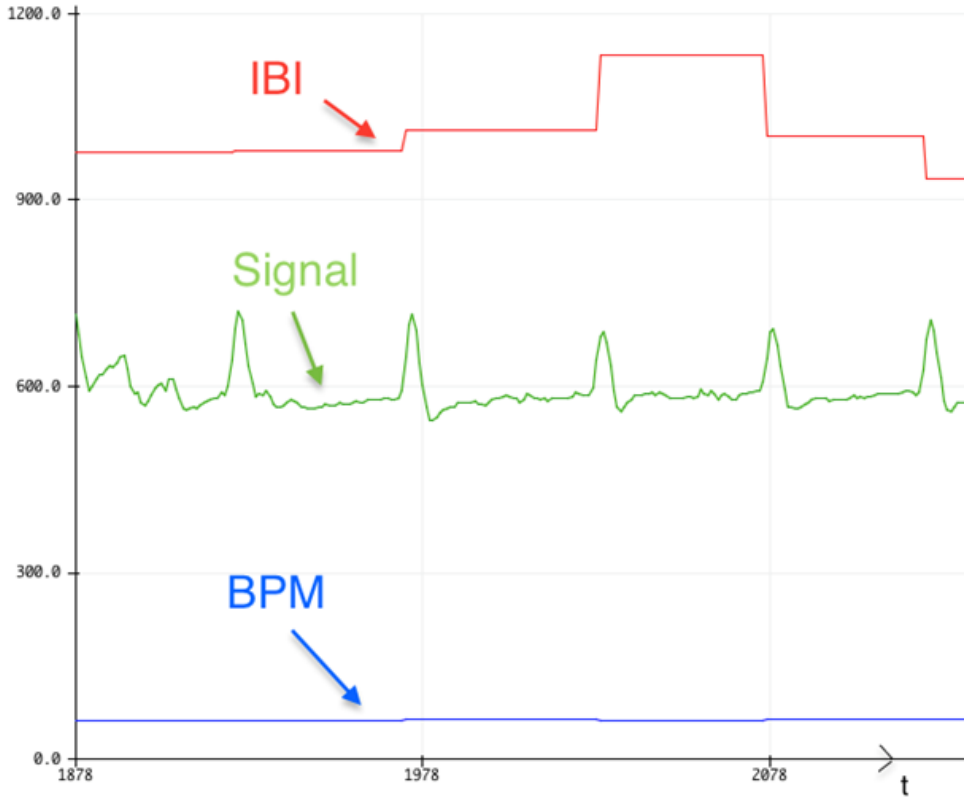


Figure 3 - Tracé Série obtenu

Une fois la compréhension complète du fonctionnement de ce capteur, nous nous en sommes inspirés pour la réalisation de notre propre code pour lequel seule la valeur en BPM est renvoyée. Nous avons alors implémenté un calcul de conversion identique de celui

présenté ci-dessus afin d'envoyer (en connexion filaire ou sans fil) la valeur de la fréquence en bpm grâce à une variable nommée « BPM ».

IV - Conception de la connexion, évolutive face aux contraintes rencontrées

Pour l'envoi et le traitement des données de l'Arduino au Swimbot, nous avons défini des objectifs croissants au fur et à mesure du projet et ce dans le but d'obtenir une meilleure efficience dans la répartition du travail au sein de notre équipe et des résultats concrets et validés étape par étape.

Ainsi, pour commencer, le code précédent nous ayant prouvé qu'une récupération via une liaison série était possible, nous avons opté pour un envoie en connexion série entre l'Arduino et le Swimbot (plus précisément à la carte Newton 2). Cependant, les difficultés rencontrées au niveau de la Newton, et notamment concernant les droits de lecture/écriture de fichiers ainsi que ceux d'exécution de script nous ont vite poussé à chercher un deuxième moyen d'envoie et notre choix fut alors porté sur le Bluetooth.

Toutefois, une connexion Bluetooth entre deux appareils est une implémentation compliquée et délicate. Nos connaissances, à ce moment en java Android étaient trop limitées. Ainsi, une troisième solution a été conçue afin de garantir une solution opérationnelle.

De manière générale, l'évolution de nos prises de décision et la répartition du travail et des tâches réalisées dans le temps sont représentées sur le diagramme suivant :

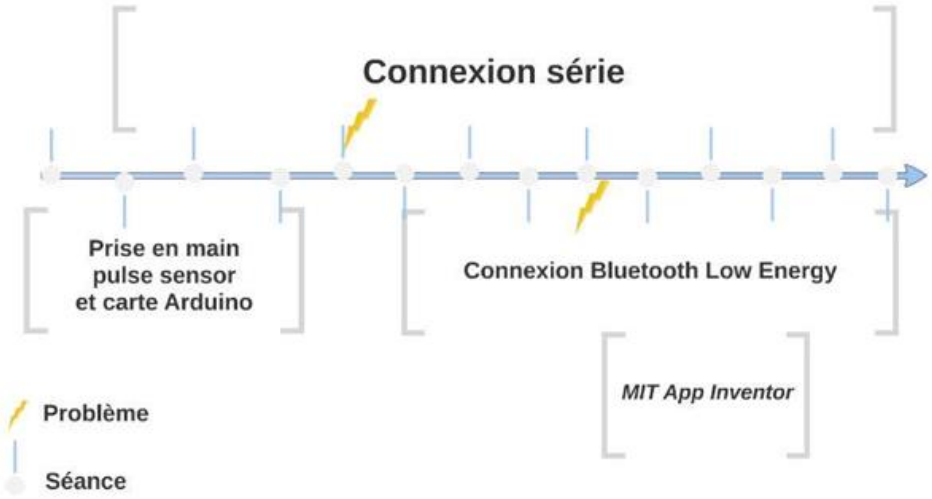


Figure 4 - Chronogramme de la réalisation du projet

Malgré ces problèmes rencontrés, nous avons finalement réussi à implémenter un affichage de la fréquence cardiaque mesurée par le capteur sur application Androïd de la Newton et d'un smartphone. La suite de ce rapport va ainsi vous illustrer ces trois méthodes, à commencer par la transmission par liaison série.

V- Transmission des données par liaison série

Notre premier choix s'est orienté vers une communication filaire entre l'Arduino et la carte Newton afin de transmettre les données par liaison série puis les traiter au travers d'une application sur le Swimbot.

V.1. Analyse de la chaîne d'acquisition

En nous rappelant de l'objectif principal de ce projet qui est d'afficher la fréquence cardiaque de l'utilisateur sur une application Android, nous avons d'abord explicité la chaîne d'acquisition suivante, qui décrit les interactions entre le capteur, l'Arduino et la Newton :

Capteur FC — *Arduino Uno ⇔ CAN* — *Swimbot ⇔ Traitement des données*

Figure 5 - Schéma de principe de la chaîne d'acquisition du signal

Tout d'abord, après avoir analysé du code en open source fournit par le concepteur du Pulse Sensor, nous avons remarqué que l'Arduino pouvait lire les valeurs du capteur via l'un de ses ports analogiques et nous avons fait le choix de brancher le fil violet du capteur (transportant le signal) au port A0 de l'Arduino.

Concernant l'interaction entre l'Arduino et le Swimbot, l'utilisation de la liaison série pour la transmission des données passe par les ports RX (pour la réception de données) et TX (pour leur transmission) des deux systèmes. Dans notre cas, les données transmises passent donc du port TX de l'Arduino au port RX du Swimbot.

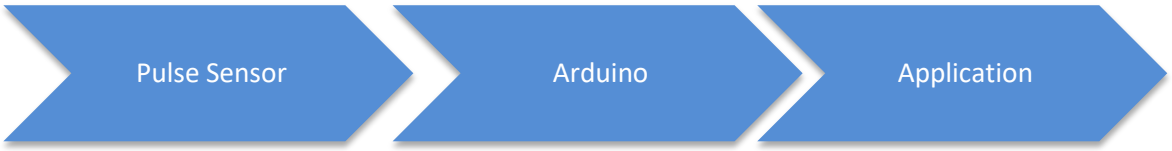
V.2. Les contraintes et les problématiques rencontrées lors de la réalisation

Une première problématique fut la question du stockage des données envoyées par le capteur à l'Arduino puis par l'Arduino au Swimbot. Nous avons remarqué que ces dernières arrivaient sur l'entrée standard du shell de la carte Newton.

Afin de pouvoir traiter ces données, nous avons alors cherché à les stocker dans un fichier texte (« .txt ») enregistré dans la carte mémoire de la Newton 2. Pour cela, nous avons redirigé ces valeurs, lorsqu'elles arrivaient sur le shell, dans ce fichier texte, à l'aide de commandes bash écrites sur un script.

Une seconde problématique concernait la lecture des données écrites. En effet, afin de récupérer ces dernières, notre application Android devait avoir les droits et surtout être capable de lire ce fichier.

V.3. Processus général



- Fait les mesures de pouls

- Lit les mesures et les convertit en bpm
- Stocke ces mesures dans un fichier .txt qu'il crée sur la carte mémoire de la Newton

- Lit le fichier .txt
- Calcule et affiche la fréquence moyenne de l'utilisateur

A. Envoi et stockage des données

Pour gérer l'acquisition des données, nous avons implémenté un code Arduino dans lequel nous récupérons les valeurs analogiques du signal mesuré par le capteur, les convertissons en numérique et les stockons dans la variable « BPM » afin d'obtenir des valeurs en battements par minute.

Comme cela a été dit précédemment, nous avons fait le choix de stocker ces valeurs dans un fichier texte préalablement créé par l'exécution d'un script dans le shell de la Newton.

L'application CoolTerm, qui est une application pour accéder à un terminale via un port série. Ainsi, nous avons accès facilement au terminal de la Newton depuis n'importe quel ordinateur. Nous avons d'abord pu tester notre script en l'exécutant « à la main », c'est-à-dire directement depuis le terminal de la Newton.

Nous avons ainsi eu une confirmation rapide de la bonne création du fichier texte mais aussi du bon fonctionnement du stockage de ces données ligne par ligne.

Cependant, ce choix soulevait une nouvelle problématique dans l'optique d'une automatisation de l'enregistrement de ces données afin de rendre notre système lus autonome : comment créer et remplir ce fichier texte de façon automatique ?

Trois options s'offraient alors à nous pour l'exécution, « automatique » ou à la demande, de ce script :

- Au démarrage de la Newton (à l'aide d'un de ses fichiers d'initialisation Linux).
- Lors du lancement de l'application Android.
- Par l'intermédiaire de la carte Arduino, qui au travers de sa connexion série avec la Newton, a le pouvoir d'écrire sur la sortie standard de celle-ci.

- **<u>Au démarrage de la Newton :</u>**

En effet, lors de la mise en route d'un système Linux comme l'est la Newton, un fichier d'initialisation (généralement nommé « init.d ») gère l'initialisation du système nécessaire à son bon fonctionnement. Ainsi, l'idée ici était d'intégrer l'exécution de ce script dans ce fichier pour qu'il s'exécute à chaque démarrage de la Newton.

Cependant, les droits nécessaires pour modifier ce fichier système ne nous étaient pas attribués. De plus, une erreur dans la manipulation de ce fichier aurait pu engendrer des conséquences néfastes au bon fonctionnement de la Newton et cette hypothèse fut donc abandonnée.

- **Lors du lancement de l'application Android :**

Cette méthode paraissait être la plus idéale pour notre système. En effet, l'idée était, ici, de laisser à l'application ce rôle pour que cette exécution ne se fasse qu'en cas de besoin, c'est-à-dire lorsqu'un utilisateur souhaite mesurer sa fréquence cardiaque. Cependant, dû à une restriction des droits d'accès d'une application extérieure au shell du système, seules des commandes de lecture (type « ls ») et non d'écriture étaient autorisées.

- **Par l'intermédiaire de l'Arduino :**

C'est donc cette dernière option que nous avons retenue, malgré le fait qu'elle ne soit pas la plus optimale. En effet, l'Arduino étant connecté en série avec la Newton, il a le pouvoir d'exécuter des commandes sur l'entrée standard de celle-ci.

Malheureusement, lorsque le script est exécuté par l'Arduino, il nous est impossible de l'arrêter sans envoyer un signal de type « SIGINT » en écrivant manuellement sur l'entrée standard de l'Arduino. L'envoi des valeurs du capteur est alors sans fin.

Ainsi, nous avons intégré les deux lignes de commande suivantes dans le code Arduino qui génère l'envoie des données sur la liaison série :

```
//Demarrage, sur la Newton, du script
//de sauvegarde des valeurs de frequences cardiaques
Serial.println("su root");
Serial.println("su -c sh \"/data/scriptFiltre.sh\"");
```

Figure 6 - Code Arduino du lancement du script

Ces deux lignes assurent ainsi l'exécution du script « scriptFiltre.sh » qui permet de créer un fichier texte nommé « HeartRateData.txt » dans lequel il écrit les valeurs mesurées ligne par ligne.

B. Lecture et traitement des données

Deux options de lecture du fichier étaient envisageables dans l'implémentation de notre application. La première correspond à une lecture interne, c'est-à-dire à la lecture d'un fichier stocké dans la mémoire interne de la Newton. Dans la seconde option, il s'agit d'une lecture externe, c'est-à-dire à la lecture d'un fichier stocké sur une mémoire externe (une carte SD par exemple).

Pour les mêmes problèmes de droit rencontrés précédemment, nous ne pouvions pas créer de fichiers dans la mémoire interne de la Newton. De plus, même si cela avait été possible, il est généralement préférable de stocker ce genre de fichier sur la mémoire externe afin de ne pas encombrer la mémoire interne du périphérique.

Ainsi, nous avons fait le choix de sauvegarder notre fichier texte « HeartRateData.txt » dans un dossier placé à la racine de la carte SD de la Newton (« /sdcard/HeartRateMeasures.txt »).

Par conséquent, l'application est capable de lire l'ensemble des données contenu dans ce fichier grâce à un algorithme de lecture ligne par ligne. Une fois les valeurs récupérées, le calcul de la moyenne totale de ces mesures est effectué puis affiché sur l'écran de carte Newton. L'utilisateur peut ainsi lire à l'écran sa fréquence moyenne.

V.4. Utilisation de l'application

Pour qu'un utilisateur puisse connaître sa fréquence cardiaque, il doit, dans un premier temps, réaliser le « téléversement » du code Arduino dans l'Arduino. Puis, sur l'écran de navigation de la carte Newton, il peut lancer l'application *« Cavaswimmer ~~ »*.

A. Solution optimale souhaitée

L'objectif premier que nous nous étions fixés était un affichage en temps réel de la fréquence cardiaque avec la possibilité que l'utilisateur affiche la moyenne lorsqu'il le désirait.

Malheureusement, une utilisation en temps réel de l'application, c'est-à-dire un affichage en direct et en continu des fréquences cardiaques mesurées, n'a pas été possible. En effet, la communication entre l'Arduino et la carte Newton étant en série, il nous est impossible d'écrire et de lire simultanément dans le même fichier.

Une solution envisageable fut alors d'envoyer le signal de fermeture du processus « SIGINT » en tapant sur l'entrée standard de la Newton « CTLR+C ». Toutefois, aucune commande de type action ne pouvait être envoyé depuis l'application Java. La lecture du fichier texte passe donc obligatoirement par une manipulation à la main de l'utilisateur via le logiciel CoolTerm pour stopper l'écriture des données dans ce fichier et libéré la liaison série.

B. Solution proposée

Il s'agit ici d'une application composée de deux pages, dont les modes opérationnels sont décrits dans le tableau suivant :

Bouton *MOYENNE*	Ouvre la nouvelle fenêtre qui affiche la moyenne de la fréquence cardiaque
Bouton *NOUVEAU TEST*	Renvoie sur la première fenêtre, et redémarre une nouvelle acquisition des données du capteur de fréquence cardiaque
Bouton *EXIT*	Quitte l'application, retour sur l'écran principal du système

Ainsi, la première fenêtre sur laquelle arrive l'utilisateur, indique à ce dernier que la capture de son pouls est en cours.

Pour obtenir la moyenne de sa fréquence cardiaque, il doit alors interrompre l'écriture dans le fichier texte, c'est-à-dire débrancher l'Arduino (qui est la solution la plus simple). En appuyant sur le bouton *MOYENNE*, la seconde fenêtre apparait alors, et affiche la valeur moyenne de la fréquence cardiaque.

L'utilisateur a ensuite la possibilité de quitter l'application, ou alors il peut refaire une série de mesures pour établir une nouvelle moyenne de sa fréquence cardiaque. Pour cela, il doit rebrancher la carte Arduino pour relancer le système.

Ci-dessous, un exemple d'utilisation de notre application. Nous pouvons ainsi voir que la personne ayant utilisé le capteur avait une fréquence cardiaque moyenne de 73,09 bpm ce qui correspond aux valeurs usuelles d'une personne au repos.

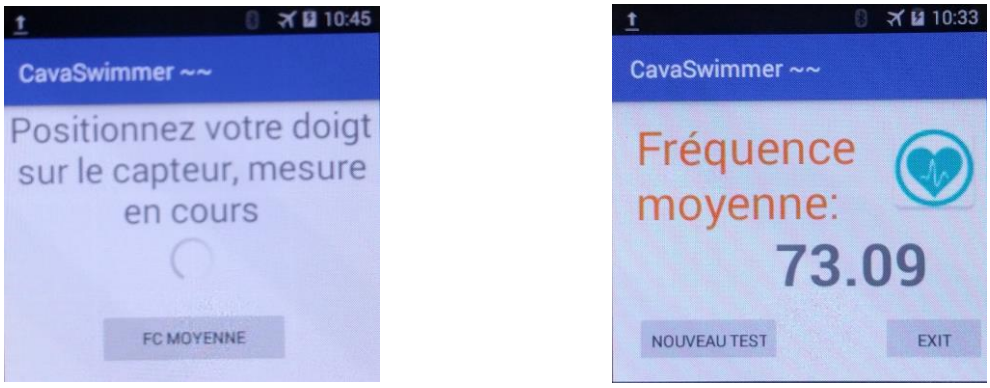


Figure 7 - Ecrans de l'application « CavaSwimmer ~~ »

En conclusion, nous avons réussi à mettre en place le cheminement d'un relevé de fréquence cardiaque à partir d'un capteur jusqu'à l'affichage fait par une application Android par une communication filaire. Nous avons alors souhaité passer par une connexion sans fil et si possible récente dans le but d'obtenir un affichage en temps réel.

VI- Etablissement d'une connexion Bluetooth Low Energy

Afin de pouvoir établir une connexion sans fils entre l'Arduino et la Newton, nous avons décidé de mettre en place une liaison par Bluetooth. En effet, cette technologie étant déjà présente sur la carte Newton, il nous suffisait alors de l'implémenter sur la carte Arduino.

Néanmoins, les ondes émises par le Bluetooth sont absorbées par l'eau. En suivant notre logique d'appréhender notre projet par objectifs croissants, un succès de transmission par Bluetooth constituerait alors une preuve de conception pour une communication sans fil. Ce travail pourra alors constituer un élément de base, par la suite, au développement du protocole LoRA.

Pour cela, nous nous sommes équipés du module Bluetooth HC-05 compatible avec Arduino. Ce dernier est en réalité un module Bluetooth Low Energy (BLE) et possède l'avantage de pouvoir, dans une phase de connexion, jouer le rôle du client ou du serveur.

VII.1. Paramétrage du module HC-05 sur l'Arduino

Le rôle du module Bluetooth est de récupérer, par l'intermédiaire de son port RX, les données envoyées par la carte Arduino, au travers de son port TX, afin de les diffuser par Bluetooth. Pour cela, nous avons donc dû connecter le HC-05 à l'Arduino, puis coder en langage Arduino l'initialisation du module et l'envoie des données.

La connexion se fait rapidement et simplement. Le module Bluetooth est alimenté en 5V par la carte Arduino. Ensuite, afin de permettre un échange de données bidirectionnel, le port RX de l'Arduino doit être connecté au port TX du HC-05 et le port TX de l'Arduino au port RX du HC-05.

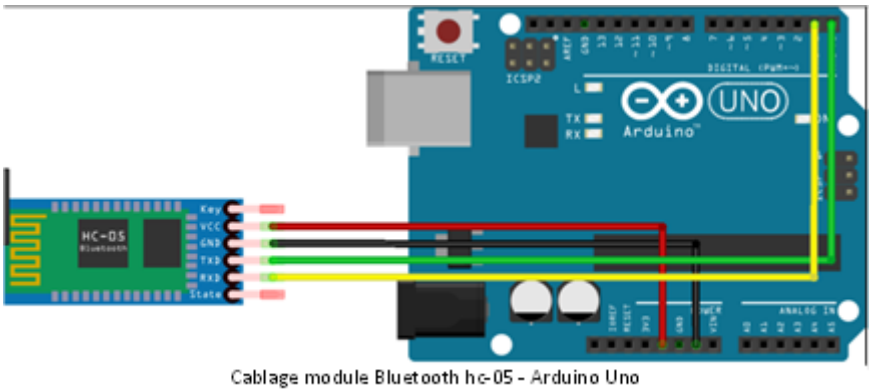


Figure 8 - Représentation du câblage entre le module BLE et la carte Arduino

Une fois les deux appareils reliés, il faut initialiser la liaison série RX/TX, puis envoyer les données provenant du capteur vers le module Bluetooth afin qu'il les diffuse sous forme d'ondes radio. Les modifications apportées au code Arduino sont les suivantes :

```
//Fonction d'envoie des données
void sendDataToSerial(char symbol, int data ){
    //Envoie des données par port USB
    Newton.print(symbol);
    Newton.println(data);
    //Envoie des données par BLE
    BTserial.println(data);

}
```

Figure 9 - Fonction d'envoi des données du capteur en code Arduino

VII.2. Implémentation de la connexion par BLE sur la Newton

Afin de permettre à la carte Newton de recevoir les données envoyées par la carte Arduino, nous avons développé une application Android permettant l'établissement de la connexion entre les deux appareils.

La connexion par BLE entre deux dispositifs est standard et suit les étapes suivantes :

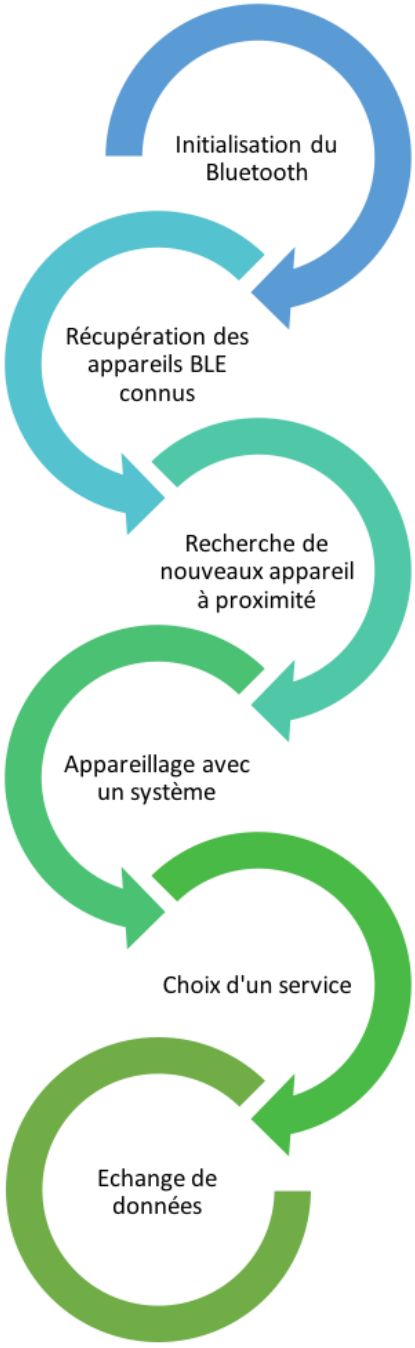


Figure 10 - Processus d'établissement d'une connexion par BLE

1 – Initialisation : Dans un premier temps, il faut vérifier si le dispositif sur lequel est lancé l'application supporte ce type de Bluetooth. Si ce n'est pas le cas, le dispositif ne pourra alors pas communiquer avec le module HC-05. En revanche, s 'il le supporte, on s'assure alors que le Bluetooth est actif, et sinon on demande à l'utilisateur de le rendre visible.

2 – Recherche des appareils BLE connus : Une fois le Bluetooth activé, on effectue une recherche récupère les appareils auxquels on s'est déjà connecté auparavant et qui sont à proximité de l'appareil exécutant l'application. Cette étape est facultative.

3 – Recherche de nouveaux appareils : Il s'agit ici de faire un « scan » pour détecter les systèmes BLE à proximité. Ces systèmes représentent l'ensemble des appareils auquel il possible de se connecté.

On affiche ensuite à l'utilisateur l'ensemble des dispositifs connus et détectés auxquels il peut se connecter pour qu'il en sélectionne un.

4 – Appareillage : Une fois que l'utilisateur a choisi un appareil auquel se connecter (dans notre cas le module HC-05), l'application va alors se connecter à cet appareil, plus précisément à son *GATT server* dans le cas du BLE. On distingue alors le *client* (la Newton), qui effectue une demande de connexion auprès du *serveur* (module HC-05) qui l'accepte ou non.

5 – Choix d'un service : Le client étant maintenant connecté au serveur, il va pouvoir récupérer une liste de services, contenant un ou plusieurs attributs (*characteristic*) offerts par ce dernier. Il faut ensuite choisir l'attribut correspondant à l'utilisation que l'on veut faire. Dans notre cas, l'attribut correspond à l'envoie de données de fréquence cardiaque.

6 – Echange de données : La connexion étant établis et le service choisis, le client peut alors recevoir des données du serveur.

Nous avons donc essayé de développer une application à implanter sur la Newton permettant l'établissement de cette connexion. Cependant, le développement de ces fonctionnalités demande une connaissance poussée de la stack Android. Or, par manque de temps, nous n'avons pas pu nous former suffisamment sur ce langage de programmation. Après plusieurs essaies, nous n'avons malheureusement pas réussis à créer une application établissant correctement la connexion entre l'Arduino et la Newton.

Pour remédier à ce problème, nous avons décidé de nous appuyer sur le code d'une application effectuant cette connexion. Après l'avoir analysé, nous l'avons modifié et adapté pour qu'il réponde à nos besoins : appareillage avec le module HC-05 et choix de l'attribut permettant le transfert des données de fréquence cardiaque.

Enfin, nous avons rajouté un bouton qui permet, une fois la connexion établie, de passer sur les activités gérant le traitement des données de fréquence cardiaque.

D'après le chronogramme, lors de la réalisation de cette connexion, nous travaillions aussi en parallèle sur la connexion série. Afin d'optimiser nos tests, nous avons donc décidé de les effectuer sur le téléphone, avec un système Android, de l'un d'entre nous. Ainsi, les images du design de notre application ont le format d'un écran de téléphone mobile.

VII.3. Présentation de l'application développée

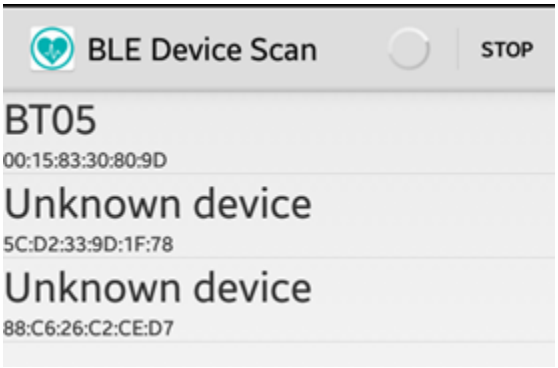


Figure 11 - Affichage des appareils trouvés à proximité

Recherche des appareils à proximité

Lorsque l'on démarre l'application, l'écran d'accueil présente le résultat du scan des appareils supportant le BLE, à proximité de la Newton.

L'utilisateur choisis ensuite le dispositif il souhaite se connecter. Ici le module HC-05 correspond à « BT05 »

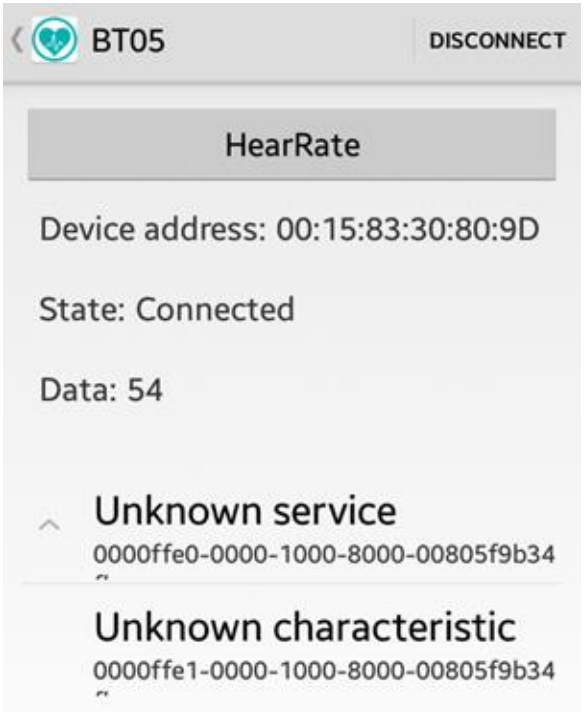


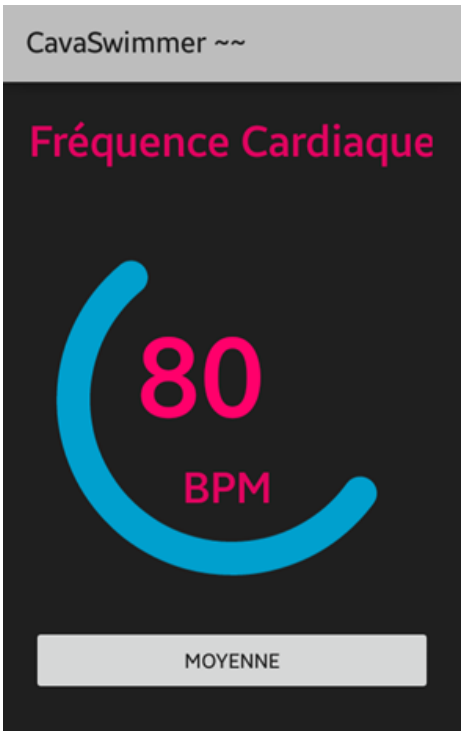
Figure 12 - choix d'un service et d'un attribut

Choix d'un service et d'un attribut

Après avoir sélectionné le dispositif auquel on souhaite se connecter, un écran apparaît renseignant l'état de la connexion, l'adresse du module Bluetooth de l'appareil et les services/attributs disponibles.

Nous avons ici limité l'affichage des attributs à ceux correspondant à l'envoie des données de fréquence cardiaque. L'utilisateur sélectionne alors ce dernier pour activer l'acquisition des données envoyées par le module HC-05.

Enfin, le bouton « HeartRate » permet de passer sur l'écran d'affichage de la fréquence cardiaque.



Affichage en temps réel de la fréquence cardiaque

Cette activité permet l'affichage en temps réel de la fréquence cardiaque (en BPM) de l'utilisateur.
Une barre de progression (*progressBar)* tournante indique que l'acquisition des données est en cours.
En bas de l'écran, le bouton « Moyenne » permet à l'utilisateur de passer sur l'écran suivant.

Figure 13 - Affichage de la fréquence cardiaque en direct sur l'application

Affichage de la moyenne

Cette activité affiche la fréquence cardiaque moyenne de l'utilisateur.
Le bouton « Nouvelle Mesure » permet à l'utilisateur de revenir à l'écran précédent et de commencer une nouvelle séquence de mesure.

Figure 14 - Affichage de la moyenne

En conclusion, nous avons réussi à mettre en place une transmission sans fil, de type Bluetooth Low Energy entre deux appareils. Par conséquent, nous répondons au besoin que peut avoir un entraineur sportif, en lui offrant une application lui permettant de connaitre la fréquence cardiaque de son sportif en temps réel.

Toutefois, n'étant pas sûrs de pouvoir finir sa réalisation à temps, nous avons pris l'initiative de proposer, toujours une transmission par BLE, mais en créant l'application par une interface différente.

VII-Mise en place d'une connexion Bluetooth Low Energy par l'intermédiaire de l'interface MIT App Inventor

Rencontrant des problèmes dans l'écriture du code pour établir la connexion Bluetooth Low Energy, nous avons décidé entre temps de nous assurer de présenter un résultat avec une connexion non filaire.

VII.1. Fonctionnement de l'interface

Il s'agit d'un IDE disponible en ligne, sur lequel l'utilisateur ajout tous les composants qu'il souhaite sur un écran virtuel de téléphone avant d'aller sur la page « Blocks » pour coder les fonctionnalités. Enfin, il lui suffit de générer un code QR qu'il scanne avec l'application MIT App Inventor, au préalable installée sur le téléphone. Ainsi, sa propre application apparait sur l'écran et il peut l'utiliser.

Cette interface de développement a été conçue par des ingénieurs du Massachusetts Institut of Technology. Elle a pour but de permettre aux développeurs amateurs de créer leur propre application plus aisément qu'en un environnement de développement normal.

En effet, le concepteur n'a plus besoin d'écrire en langage de programmation orienté objet. Le concept est simple: il repose sur des graphiques. Il s'agit d'associer des blocs de contrôle sous la forme de puzzle imbriqué afin d'ajouter des fonctionnalités aux composants de l'écran. Nous retrouvons la même logique des fonctions en programmation (« when…do », ou « if …else ») mais simplifiée.

Par exemple, l'établissement de la connexion est plus facile à concevoir. Une fonction telle que celle qui permet de trouver d'autres utilisateurs de Bluetooth LE autour existe déjà. Comme nous pouvons le voir ci-dessous, elle est faite par « call BlutoothLE1.StartScanning »:

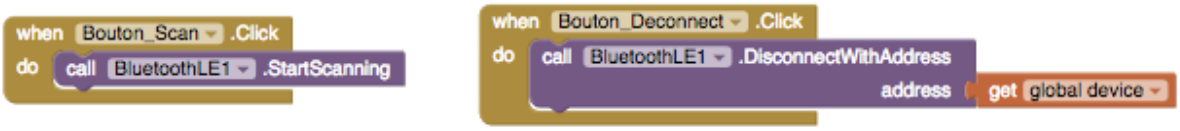


Figure 15 - Exemple des blocs pour coder une application sous MIT Inventor

L'utilisateur doit alors assembler les blocs ensemble pour développer son application. Le code final que nous avons créé est en annexe 1.

VII.2. Utilisation de l'application

Lorsque l'application est lancée, l'utilisateur voit cette page apparaître sur son Swimbot :

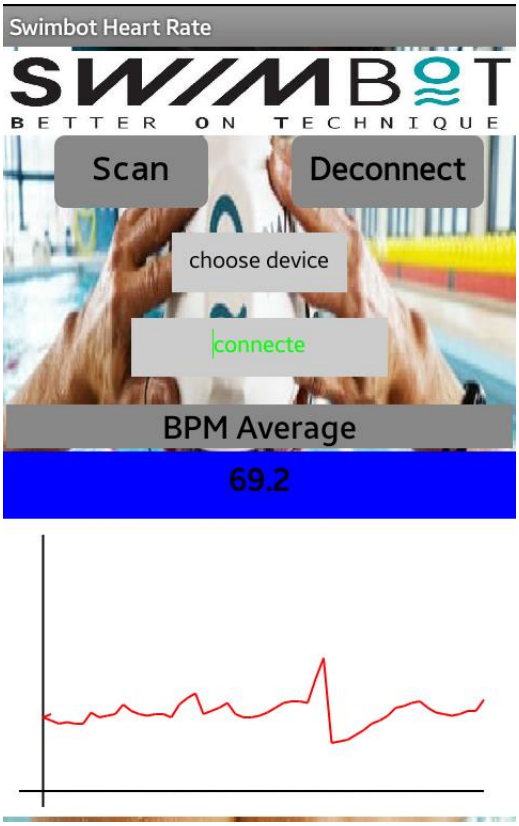


Figure16 - Graphisme de l'application par MIT Inventor

Les modes opérationnels de cette application sont décrits dans le tableau suivant :

Bouton *Scan*	Analyse les appareils avec le Bluetooth actif autour (à lancer en premier)
Bouton *choose device*	Affiche les appareils trouvés au scan. Sélectionner celui concerné (ici BT 05). Un message connecté remplace non connecté
Bouton *Deconnect*	Couper la connexion Bluetooth avec l'appareil concerné
Graphe	Affiche en temps réel les valeurs reçues lors de l'acquisition. En cliquant dessus, réinitialise et redémarre une nouvelle acquisition

Après avoir établi la connexion, l'application va afficher en continu la valeur qu'elle reçoit du capteur. Au bout d'environ 55 mesures, le graphe est alors complètement tracé et la valeur moyenne calculée apparait dans le carré bleu, sous le texte « BPM Average ».

Par conséquent, cette solution nous a permis de mieux nous approprier le processus d'établissement d'une connexion Bluetooth. Elle est en complément de celle proposée précédemment, qui répond davantage à un travail d'ingénieur.

Conclusion

Nous avons donc développé trois méthodes différentes afin d'afficher la fréquence cardiaque d'une personne sur l'écran d'un appareil Android au travers d'une application, fréquence mesurée par un capteur. Potentiellement, un entraineur peut même suivre en direct l'évolution cardiaque de son sportif lors d'un entrainement grâce à nos applications connectées en Bluetooth.

La suite logique de ce projet aurait été de développer notre code pour une application au réseau LoRA. Ce dernier suivant une logique très proche de celle abordée sur une transmission par Bluetooth, il rentre dans la logique de notre processus d'objectifs croissants et aurait répondu au besoin d'une communication sans fil dans un milieu aquatique.

Ce projet Tuteuré était ainsi le premier projet pour lequel nous étions entièrement autonomes et indépendants. Il s'agit donc d'une expérience nouvelle, assez éloignée de nos Travaux Pratiques ou des Bureaux d'Etudes que nous avons eu dans le cadre de nos cours.

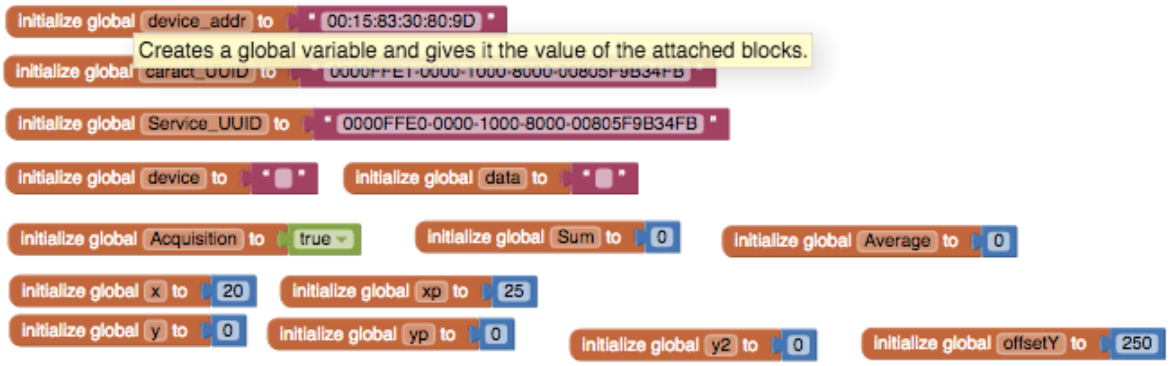
En effet, dans ce projet, il nous a été demandé de concevoir une solution du début à la fin, et nous avons du faire face aux problèmes rencontrés lors de la réalisation sans avoir un professeur connaissant la solution pour nous aider.

En parallèle, nous avons mis en place une organisation de travail au sein du groupe pour mener le projet à bien. Pour cela, nous avons définis des tâches définies et nous nous les sommes répartis avec une mise en place fréquente de réunions nous permettant d'avancer toujours dans la même direction.

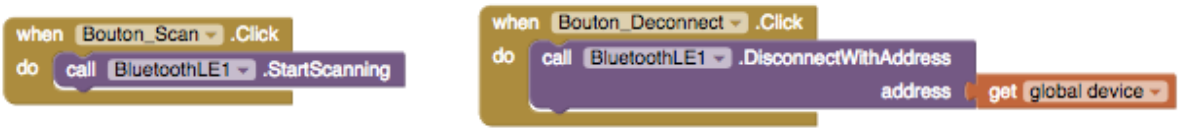
Il a s'agit ainsi d'une expérience très enrichissante pour l'ensemble de l'équipe et nous a donné un aperçu de ce qui peut nous attendre en entreprise.

Annexe

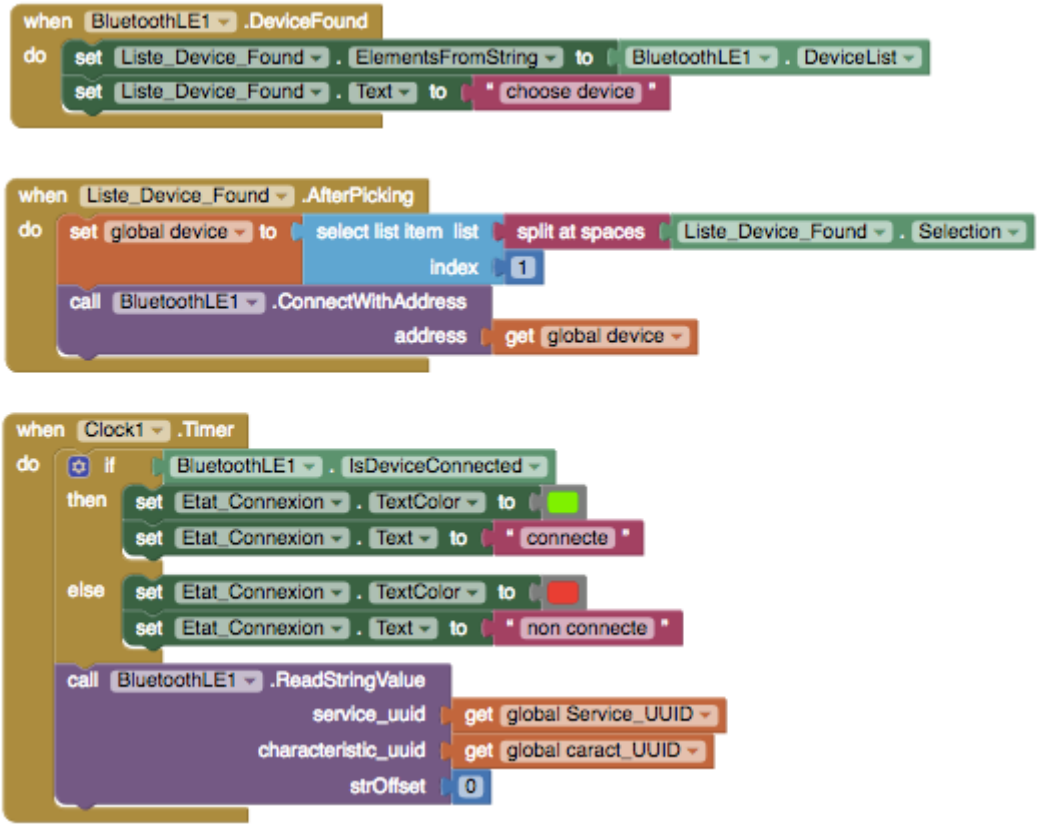
Annexe 1 : Code programmation fait sous MIT App Inventor



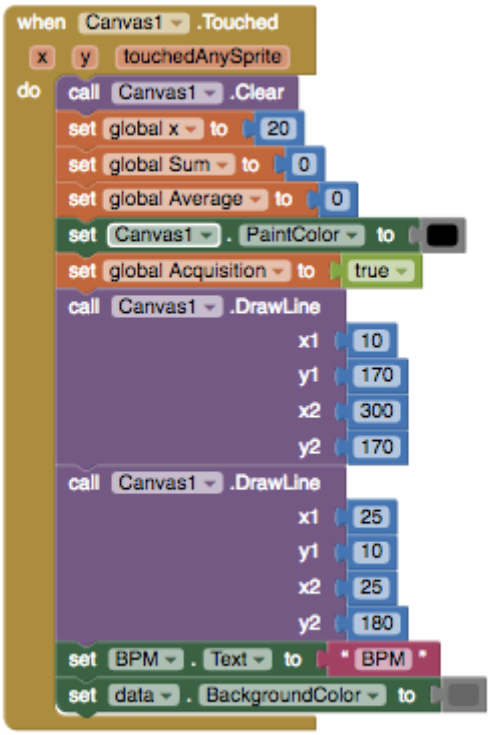
Initialisation des variables



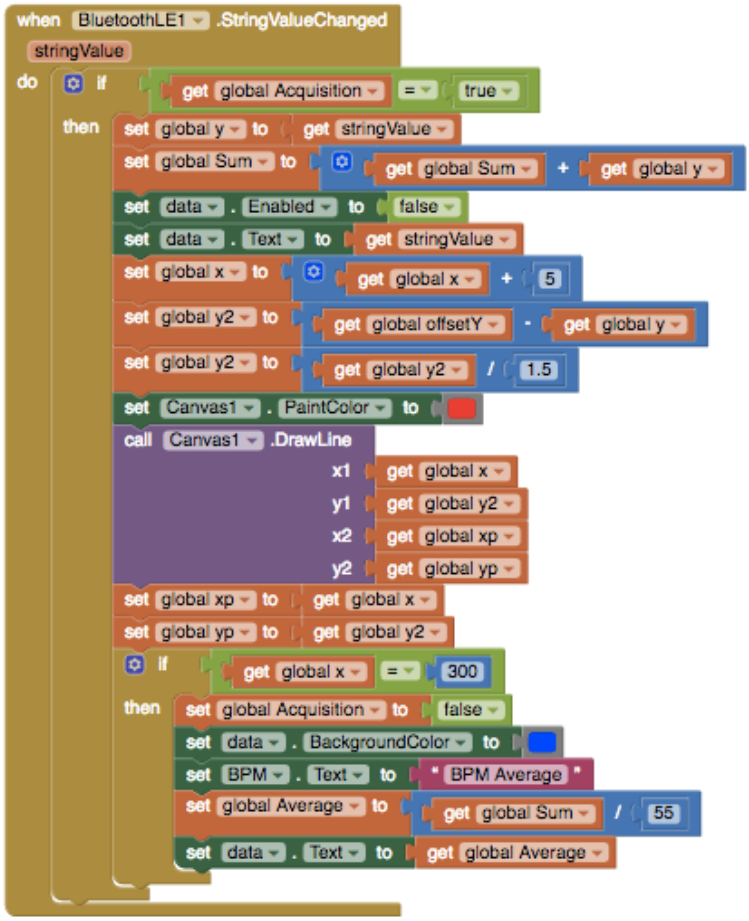
Création des boutons *Scan* et *Deconnect*



Affichage des devices supportant le BLE et établissement de la connexion avec le device sélectionné



Démarrage d'une nouvelle séquence de mesure par toucher du graphique



Affichage en temps réel des valeurs de fréquence cardiaque et affichage du le graphique