

Rapport de Projet IMA4

Vêtements Intelligents



Table des matières

Introduction	2
I) Application Android.....	2
1) Environnement.....	2
2) Présentation de l'application.....	3
1.1) Menu	3
.....	3
1.2) Paramètres	3
1.3) Entraînement.....	3
1.4) Historique.....	7
.....	7
1.5) Activité Stat	8
II) Tee-shirt Connecté	9
1) Présentation	9
2) Capteur de température TMP36GZ	10
3) Accéléromètre MMA8452Q	11
4) Puce BLE nrf8001.....	14
5) Réalisation du Tee-Shirt IMAmotive.....	16
Conclusion.....	17
Annexes.....	18

Introduction

Le but de notre projet est de concevoir un système d'interaction avec des objets connectés sur les vêtements ou le corps. Ce sujet nous a particulièrement plu car les systèmes embarqués sur les humains sont de plus en plus présent dans notre quotidien comme les montres connectées, les lunettes Google Glass ou bien encore les bracelets connectés permettant de récolter des informations sur les nageurs pendant leur entraînement. Nous réaliserons une plateforme basée sur la technologie BLE (Bluetooth Low Energy ou Bluetooth 4.0) et mettant en œuvre deux parties. La première est un système embarqué compact intégré à un T-shirt permettant de recevoir des informations et en transmettre. La deuxième sera de créer une application Android permettant de communiquer avec ce système.

I) Application Android

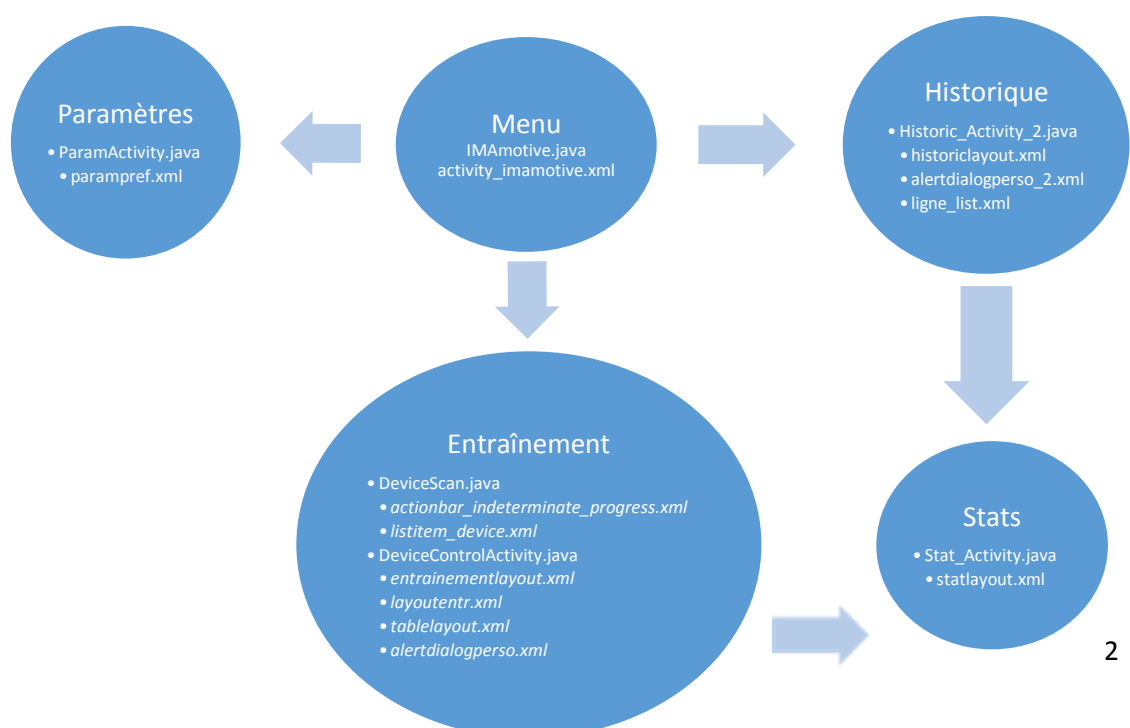
1) Environnement

Nous avons décidé de travailler avec l'IDE **Eclipse**. Pour que nos fichiers Java s'exécutent et pour avoir un ensemble d'outils nous permettant de compiler et déboguer nous avons besoins de la JDK (Java Development Kit) « Java SE 8u45 ». Afin de développer une application Android nous avons besoins également du SDK Android (Software Development Kit) disponible sur leur site. L'application ne pourra être utilisée que par des smartphones supportant l'API 18 ou plus, c'est-à-dire la version Android 4.3.1 car c'est à ce niveau-là que les fonctions permettant d'utiliser la technologie Bluetooth Low Energy ont été ajoutées.

Notre environnement est composé de fichier Java et html, les fichiers Java servent aux calculs et les fichiers Xml à la présentation finale (**cf Annexes Environnement**).

A chaque classe Java est associé un ou plusieurs fichiers XML qu'on appellera par la suite layout.

Voici comment sont représentés les liens entre les différentes activités dans notre projet :



2) Présentation de l'application

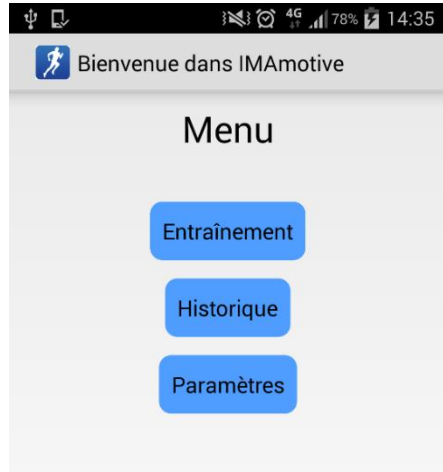
1.1) Menu

Le menu est constitué de trois boutons simples menant à d'autres activités :

- Entraînement
- Historique
- Paramètres

Il permet une navigation simple entre les différentes activités proposé.

Pour effectuer une action lors d'un clic sur le bouton il suffit de mettre un Listener sur celui-ci. Ici on passe à une autre activité (**cf Annexe IMAmotive.java**). Il faudra en plus ajouter l'activité dans le manifest.xml (**cf Annexes manifest.xml**) pour prévenir le compilateur que la classe java se comporte comme une activité.



1.2) Paramètres

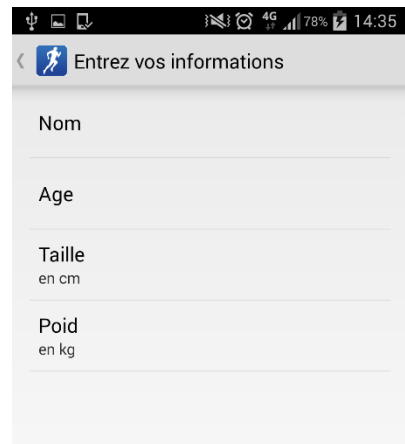
Nous avons créé une activité « Paramètre » afin d'avoir des informations sur l'utilisateur, le plus important est de savoir sa taille et son poids. Cela servira par exemple à effectuer quelques calculs comme le nombre de calories brûlé à chaque pas effectué. Cependant ces données n'ont pas été exploitées.

Cette activité est ce que l'on appelle une *préférence partagé* (son extension s'appelle « *PreferenceActivity* ») elle est utilisée pour conserver des données même si l'application est arrêtée ou tuée. Ces données pourront être utilisé depuis n'importe qu'elle autre activité en utilisant cette ligne :

```
String taille = preferences.getString(keyTextTaille, "160");
```

Le premier argument étant l'identifiant de la valeur, le deuxième est la valeur par défaut au cas où il n'y aurait pas eu d'association.

Pour ce faire le fichier Xml doit être placé dans le dossier « xml » se trouvant dans le dossier « res » (**cf Annexes Environnement et parampref.xml**).



1.3) Entraînement

Cette partie comporte deux activités se succédant. La première sert à établir la connexion Bluetooth avec le T-Shirt, c'est l'activité « DeviceScanActivity » (extension de la classe « ListActivity »). La deuxième « DeviceControlActivity » (extension de la classe « Activity ») gère les différents

événements comme la connexion, déconnexion, envoi et réception de données, en parallèle elle enregistre les données reçues dans un fichier texte.

1.3.1) DeviceControlActivity

1.3.1.1) Connexion Bluetooth Low Energy

Cette partie a été le plus compliqué à comprendre car il a fallu s'aider de code déjà établi sur le net comme celle disponible en open source sur le site Android, nous nous sommes également inspiré du code de l'application nrfUART2.0 disponible en open source également sur gitHub.

Afin d'utiliser le Bluetooth dans notre application nous devons indiquer la permission de l'utiliser dans le fichier manifest.xml, il suffit d'y ajouter ces trois lignes :

```
<uses-feature android:name="android.hardware.bluetooth_le" android:required="true"/>
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
```

Avant de pouvoir communiquer avec le T-shirt nous devons nous assurer que le Bluetooth peut être supporté.

Dans la méthode «onCreate» de l'activité «DeviceScanActivity» nous devons ajouter ces lignes :

```
// Use this check to determine whether BLE is supported on the device. Then you can
// selectively disable BLE-related features.
if (!getPackageManager().hasSystemFeature(PackageManager.FEATURE_BLUETOOTH_LE)) {
    Toast.makeText(this, R.string.ble_not_supported, Toast.LENGTH_SHORT).show();
    finish();
}
// Initializes a Bluetooth adapter. For API level 18 and above, get a reference to
// BluetoothAdapter through BluetoothManager.
final BluetoothManager bluetoothManager = (BluetoothManager) getSystemService(Context.BLUETOOTH_SERVICE);
mBluetoothAdapter = bluetoothManager.getAdapter();

// Checks if Bluetooth is supported on the device.
if (mBluetoothAdapter == null) {
    Toast.makeText(this, R.string.error_bluetooth_not_supported, Toast.LENGTH_SHORT).show();
    finish();
    return;
}
```

Une fois l'activité créée nous devons nous assurer que le Bluetooth est activé sur le téléphone, pour cela on écrit dans la méthode « onResume » :

```
// Ensures Bluetooth is enabled on the device. If Bluetooth is not currently enabled,
// fire an intent to display a dialog asking the user to grant permission to enable it.
if (!mBluetoothAdapter.isEnabled()) {
    if (!mBluetoothAdapter.isEnabled()) {
        Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
    }
}
```

1.3.1.2) Recherche des appareils Bluetooth

Nous devons à présent rechercher notre appareil, appelé *Tshirt-IMA*. Nous créons une liste qui répertorie tous les appareils détecté. Elle sera définie à partir d'une interface nommée *LeDeviceListAdapter* (Cf Annexe DeviceScanActivity.java).

Pour trouver les Devices nous utilisons les méthodes startLeScan() et stopLeScan() qui vont respectivement débiter le scan des Devices ou l'arrêter. Ces méthodes se trouvent dans la classe BluetoothAdapter.

Elles prennent en paramètre *BluetoothAdapter.LeScanCallback* qu'il faudra implémenter dans le fichier *DeviceScanActivity.java* :

```
// Device scan callback.
private BluetoothAdapter.LeScanCallback mLeScanCallback = new BluetoothAdapter.LeScanCallback() {
    @Override
    public void onLeScan(final BluetoothDevice device, int rssi, byte[] scanRecord) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                mLeDeviceListAdapter.addDevice(device);
                mLeDeviceListAdapter.notifyDataSetChanged();
            }
        });
    }
};
```

Lorsque la méthode *startLeScan()* trouve un Device elle l'ajoute à la *ListView* «*mLeDeviceListAdapter*» et l'affiche directement sur le téléphone. Nous aurions pu passer en paramètre un tableau *UUID[]* afin de ne sélectionner que notre T-shirt pour s'assurer que l'utilisateur ne sélectionnera pas un autre appareil.

Pour pouvoir agir sur les Devices détecté on place un Listener sur les Items de la liste. Lorsque l'utilisateur en sélectionne un on passe l'adresse du Device en paramètre pour que l'activité *DeviceControlActivity* puisse connaître le Device en question. (cf Annexes *DeviceScanActivity.java*)

1.3.2) DeviceControlActivity

1.3.2.1) Communication Bluetooth

Une fois le Device « *Tshirt-IMA* » sélectionné l'utilisateur voit apparaître un bouton *Start*. La communication débutera lorsqu'il appuiera dessus.

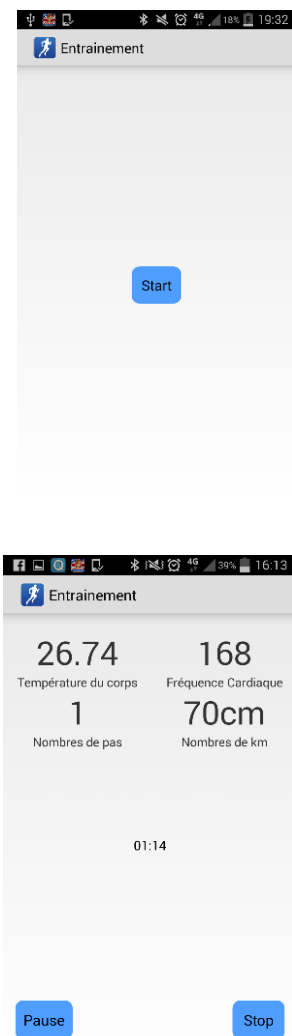
Dans le Listener du bouton *Start* on récupère l'adresse du Device, on envoie un message « 1 » au T-shirt pour qu'il commence à nous envoyer ces données et on débute le chronomètre.

L'utilisateur peut ensuite décider de faire une pause en appuyant sur le bouton correspondant, cela va déconnecter le téléphone du T-shirt pour que celui-ci ne lui envoie plus de donnée et arrêter le chronomètre. Le bouton « Reprendre » prendra la place du bouton « pause » pour reprendre la connexion et le chronomètre.

```
//Reprise des calculs
reprendre.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        reprendre.setVisibility(View.GONE);
        pause.setVisibility(View.VISIBLE);

        focus.setBase(SystemClock.elapsedRealtime() + timeWhenStopped);
        focus.start();
        mBluetoothLeService.connect(mDeviceAddress);
    }
});
```

La gestion des événements comme la connexions/déconnexions, la découverte d'un service ou l'envoi et la réception de donnée se fait à l'aide du Handler *mGattUpdateReceiver* de type « *BroadcastReceiver*» (cf Annexes *DeviceControlActivity.java*)



A chaque évènement `ACTION_GATT_CONNECTED` on envoie un message « 0 » au T-Shirt pour qu'il nous envoie ses données.

1.3.2.2) Traitement des données

Dans le Handler `mGattUpdateReceiver` l'évènement `ACTION_DATA_AVAILABLE` va traiter les paquets de byte reçu. Les paquets sont sous la forme de tableau de byte.

Il faut appeler cette fonction pour le récupérer :

```
intent.getByteArrayExtra(BluetoothLeService.EXTRA_DATA);
```

Nous avons défini un protocole afin de savoir quel bit correspond à quelle information. La trame a une taille de 9 bits, elle commence par un bit de Start « C0 » et se termine par un bit de stop « C1 », si ce n'est pas le cas la trame n'est pas traitée. Les deux premiers bits correspondent à la température (un pour les unités/dizaines et l'autre pour les dixièmes/centièmes), le 3^{ème} la fréquence, et les 4 derniers pour le nombre de pas (resp. millième, centaines, dizaines, unités). Le temps est récupéré à partir du chronomètre.

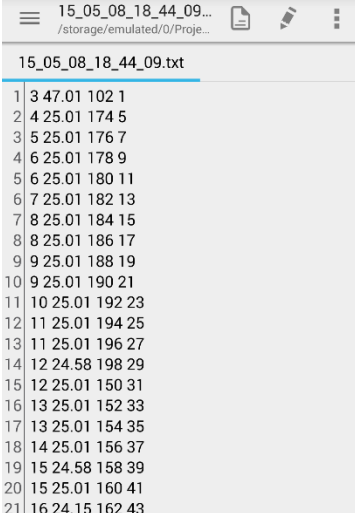
Nous traitons directement ces valeurs en les affichant à l'écran et en les stockant dans un **fichier**. Pour pouvoir lire et écrire dans ces fichiers nous devons spécifier la permission dans le manifest (**cf Annexe manifest.xml**) et nous utilisons les fonctions écrites dans la classe « `FileClass.java` ».

Ce fichier est créé lorsque l'utilisateur clique sur le bouton *Start*, son nom est déterminé en fonction de la date et de l'heure à laquelle il a appuyé au format « `yy_MM_dd_HH_mm_ss` », avec cette méthode il est impossible d'avoir deux fichiers avec un nom identique.

Tous les fichiers sont rangés dans un dossier « `Projet_Tshirt` », s'il n'existe pas il est automatiquement créé.

Les données sont écrites dans l'ordre suivant (exemple ci-contre):

Temps | Température | Fréquence cardiaque | Nombre de pas



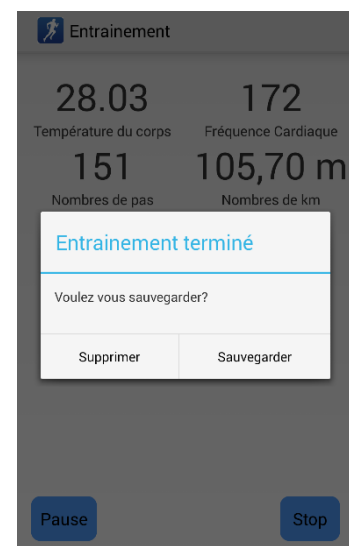
Line	Time	Temp	Freq	Steps
1	3	47.01	102	1
2	4	25.01	174	5
3	5	25.01	176	7
4	6	25.01	178	9
5	6	25.01	180	11
6	7	25.01	182	13
7	8	25.01	184	15
8	8	25.01	186	17
9	9	25.01	188	19
10	9	25.01	190	21
11	10	25.01	192	23
12	11	25.01	194	25
13	11	25.01	196	27
14	12	24.58	198	29
15	12	25.01	150	31
16	13	25.01	152	33
17	13	25.01	154	35
18	14	25.01	156	37
19	15	24.58	158	39
20	15	25.01	160	41
21	16	24.15	162	43

1.3.2.3) Arrêt de l'entraînement

Lorsque l'utilisateur décide d'arrêter son entraînement il doit appuyer sur *Stop*, dès lors une `alertDialog` s'affichera et lui demandera s'il veut supprimer ou sauvegarder ses performances.

L'`alertDialog` est généré depuis le layout « `alertyDialogperso` », elle est composée de deux boutons : un « `PositiveButton` » et un « `NegativeButton` ».

Le « `NegativeButton` » va supprimer le fichier qui vient d'être créé tandis que le « `PositiveButton` » appelle l'activité « `Stat_Activity` » et lui met en paramètre le nom du fichier qu'il vient de traiter.



1.4) Historique

Cette activité est lancée depuis le menu en appuyant sur le bouton du même nom, la classe associée est `Historic_Activity`, c'est une extension de la classe `ListActivity`. Elle répertorie tous les fichiers du dossier « `Projet_Tshirt` » à l'aide d'une `ListView` et permet soit d'accéder aux détails des entraînements précédents soit de les supprimer (plusieurs `Item` peuvent être sélectionnés en même temps).

Pour récupérer la liste des fichiers créés nous avons écrit une fonction `recupFichiers()` dans notre classe « `FileClass` » qui renvoie les fichiers existant sous forme de tableau de type `File`. Nous parcourons ensuite tous les fichiers pour extraire la date de leur nom (**cf Annexes `Historic_Activity.java`**).

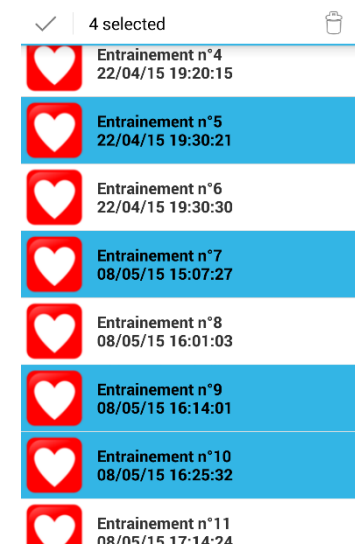
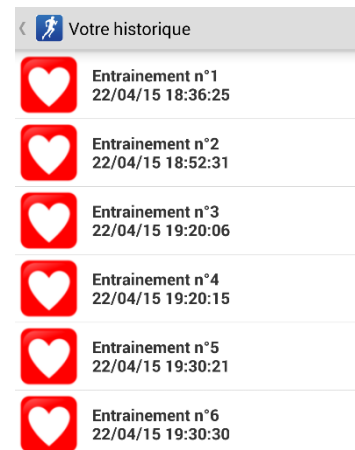
Nous créons ensuite la `ListView` à partir de l'interface `SelectionAdapter` (définie dans la classe `Historic_Activity`). Choisir ce type d'interface a été assez difficile car il a fallu faire le choix entre une `ListView` proposant une interface plus riche en détails (plusieurs `Layout` dans la même ligne) et `ListView` pouvant sélectionner plusieurs `Item` en même temps mais n'offrant qu'un seul `Layout` à la fois.

Nous avons opté pour la `ListView` permettant de choisir plusieurs `Item` à la fois, en effet il est plus pratique pour l'utilisateur de sélectionner plusieurs `Entrainement` s'il décide d'en supprimer une grosse partie.

Pour cela la `ListView` doit être en mode « `CHOICE_MODE_MULTIPLE_MODAL` ».

Puis on ajoute 3 `Listener` :

- `getListView().setOnItemLongClickListener` : Cette fonction est appelée lorsque l'on effectue un clic long sur l'item. Dans cette fonction on appelle `getListView().setItemChecked` pour indiquer que l'on passe en mode « sélection multiple ». Cette fonction appelle la fonction `getListView().setMultiChoiceModeListener` qui permet de définir ce que l'on veut faire lorsque l'on sélectionne ou désélectionne plusieurs `Item`.
- `getListView().setMultiChoiceModeListener` : On change l'`ActionBar` en appelant le `layout` de type « menu » `menuhistoric` (situé dans `res/menu`). Ci-contre vous pouvez constater le changement de l'`ActionBar`, elle dispose d'un item « `item_delete` » représenté par une petite poubelle. Lorsque l'on clique sur cet item la fonction `onActionItemClicked` est appelée. On enregistre alors toutes les positions dans un tableau de type « `Set<Integer>` » (`SetPos`) puis on fait apparaître une `AlertDialog` dont le `layout` est



alertdialogperso_2 (cf ci-contre). Si on clique sur « Oui » elle supprimera les fichiers sélectionnés à l'aide du tableau SetPos.

- *getListView().setOnClickListener* : cette fonction est appelé lorsque l'on effectue un clique simple sur l'Item, elle mémorise le nom de l'Item sélectionné pour le passer en paramètre avant d'appeler l'activité « Stats » (de la même façon que dans l'activité DeviceControlActivity.

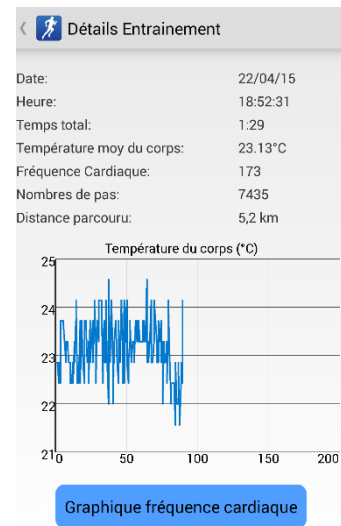
1.5) Activité Stat

1.5.1) Récupération des données

Cette activité est appelée après l'activité *Historic_Activity* et *DeviceControlActivity*, sa classe associée se nomme *Stat_Activity*. Elle récupère le nom du fichier à l'aide de la fonction *secondeActivite.getStringExtra(EXTRAS_ID_FILE)*.

La principale fonction de cette activité est de lire dans ce fichier, on utilise alors les fonctions défini dans la classe « *FileClass* ».

L'idée est de lire les lignes du fichier une par une et de les mettre dans un tableau de string (on utilise la fonction *LireFichierE* de la classe *FileClass*). Puis on « split » chaque case du tableau (qui sont des strings) afin de récupérer les différentes colonnes, de là le traitement des données se fait assez facilement et on récupère toutes celles dont on a besoins (cf **Annexes Stat_Activity.java**) et on les affiche à l'aide de *TextView*.



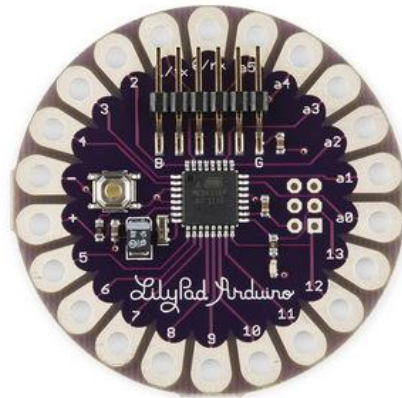
1.5.2) GraphView

Nous avons décidé d'afficher deux graphiques, un représentant la fréquence cardiaque en fonction du temps et l'autre la température du corps. Pour cela nous avons installé la bibliothèque *GraphView* disponible sur internet. Malheureusement nous n'exploitons pas toutes les ressources de cette bibliothèque car nous n'avons pas trouvé les codes le permettant, nous aurions pu par exemple effectuer un zoom sur le graph ou afficher certains points lorsque l'on clique sur la courbe. L'utilisateur peut décider de switcher entre ces deux graphiques en appuyant sur le bouton « Graphique » en bleu situé en bas du graphique. (cf **Annexes Stat_Activity.java**)

II) Tee-shirt Connecté

1) Présentation

Nous avons décidé de créer un Tee-shirt qui envoie des données en continu. Ce Tee-shirt doit pouvoir récupérer la température du corps et le nombre de pas de l'utilisateur puis l'envoyer en Bluetooth sur l'application Android. Le Tee-shirt inclut donc un capteur de température, un accéléromètre utilisé en podomètre et une puce Bluetooth. Pour contrôler tous ces éléments, nous allons utiliser un microprocesseur Atmega328p intégré dans un Lilypad. Le Lilypad a l'avantage de pouvoir être cousu sur le tee-shirt avec ses broches très espacées.



L'ensemble de nos programmes sur le microprocesseur sont écrits en langage C. Nous avons donc commencé par nous documenter sur les différents capteurs et les protocoles de communication utilisés. Voici les capteurs que nous avons utilisés :

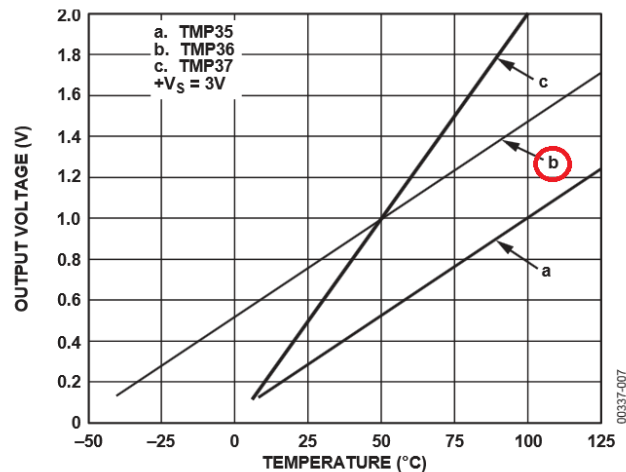
- Capteur de température TMP36GZ
- Accéléromètre MMA8452Q
- Emetteur Bluetooth Low Energy nrf8001

Nous allons vous présenter chacun de ses capteurs.

2) Capteur de température TMP36GZ



Le TMP36GZ est un capteur de température analogique. La tension sur sa broche de sortie est proportionnelle à sa température. Ce capteur peut mesurer une température comprise entre -40°C et $+125^{\circ}\text{C}$ avec une précision de $\pm 0.5^{\circ}\text{C}$. Voici ses caractéristiques tension/température :



Nous avons choisi ce capteur car sa tension d'alimentation doit être entre 2.V et 5.5V, comme le Lilypad délivre une tension de 3.3V, cela convient parfaitement. Pour convertir la tension récupérée en température, il faut utiliser la courbe ci-dessus et les caractéristiques du convertisseur analogique numérique (CAN) du Lilyad. Le CAN est un convertisseur 8bits qui récupère une tension en fonction de la tension d'alimentation du capteur. Ici notre capteur est alimenté en 3.3V. Le Lilypad va donc nous récupérer une valeur sur 8bits que nous allons convertir en volt puis en $^{\circ}\text{C}$.

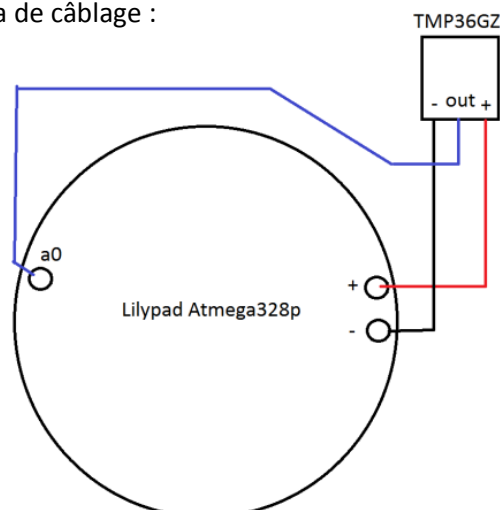
Conversion Bit -> Volt :

$$\text{Temperature} = \frac{3.3V * \text{Valeur lue}}{255} \text{ en Volt}$$

Conversion Volt -> $^{\circ}\text{C}$:

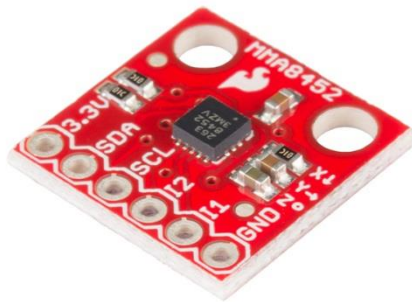
$$\text{Temperature} = \frac{\text{Temperature en Volt} * 25^{\circ}\text{C}}{0.75V} \text{ en } ^{\circ}\text{C}$$

Grâce à ses deux formules, nous pouvons récupérer la valeur de la température du corps de l'utilisateur. Voici le schéma de câblage :



3) Accéléromètre MMA8452Q

L'accéléromètre est ici utilisé comme podomètre pour compter le nombre de pas effectués par l'utilisateur. Nous allons regarder la valeur d'accélération sur l'axe vertical, si nous dépassons une certaine valeur d'accélération, c'est qu'il y a eu un choc et donc un pas. Le problème avec les accéléromètres analogique, c'est qu'il faut lire la valeur en continue alors que notre programme fait une multitude de chose et donc il ne peut pas lire en continue les valeurs de l'accélération. Nous avons donc décidé de prendre l'accéléromètre MMA8452Q car cet accéléromètre peut générer des interruptions avec plusieurs modes de fonctionnement. En câblant l'accéléromètre sur le LilyPad en utilisant les broches d'interruptions, nous pourrions ainsi détecter chaque pas fait par l'utilisateur.



L'accéléromètre MMA8452Q est un accéléromètre numérique 3-axes sur une résolution de 12bits. Il peut être configuré pour détecter une accélération de $\pm 2g/\pm 4g/\pm 8g$. Il possède deux broches d'interruptions que l'on peut programmer suivant 6 modes de détection et est alimenté avec une tension comprise entre 1.95V et 3.6V. Cet accéléromètre utilise le protocole I2C pour communiquer.

Le protocole I2C utilise un bus série synchrone bidirectionnel. Avec un bus, il peut y avoir plusieurs maîtres et plusieurs esclaves. Les échanges ont toujours lieu entre un seul maître et un (ou tous les) esclave(s), toujours à l'initiative du maître (jamais de maître à maître ou d'esclave à esclave). Cependant, rien n'empêche un composant de passer du statut de maître à esclave et réciproquement.

La connexion est réalisée par l'intermédiaire de 2 lignes :

- SDA (Serial Data Line) : ligne de données bidirectionnelle,
- SCL (Serial Clock Line) : ligne d'horloge de synchronisation bidirectionnelle.

Il ne faut également pas oublier la masse qui doit être commune aux équipements.

Les 2 lignes sont tirées au niveau de tension V_{DD} à travers des résistances de pull-up (R_p).

Le nombre maximal d'équipements est limité par le nombre d'adresses disponibles, 7 bits pour l'adresse et un bit pour définir si on écrit ou on lit, soit 128 périphériques, mais il dépend également de la capacité (C_b) du bus (dont dépend la vitesse maximale du bus). Il faut savoir que des adresses sont réservées pour diffuser des messages en broadcast et que de nombreuses adresses sont déjà attribuées par les fabricants ce qui limite grandement le nombre d'équipements (une variante d'adressage sur 10 bits existe également).

Le message peut être décomposé en 2 parties :

- Le maître est l'émetteur, l'esclave est le récepteur :
 - émission d'une condition de START par le maître (« S »),
 - émission de l'octet ou des octets d'adresse par le maître pour désigner un esclave, avec le bit R/W à 0 (voir la partie sur l'adressage ci-après),
 - réponse de l'esclave par un bit d'acquiescement ACK (ou de non-acquiescement NACK),
 - après chaque acquiescement, l'esclave peut demander une pause (« PA »).
 - émission d'un octet de commande par le maître pour l'esclave,
 - réponse de l'esclave par un bit d'acquiescement ACK (ou de non-acquiescement NACK),
 - émission d'une condition de RESTART par le maître (« RS »),
 - émission de l'octet ou des octets d'adresse par le maître pour désigner le même esclave, avec le bit R/W à 1.

- Le maître devient récepteur, l'esclave devient émetteur :
 - émission d'un octet de données par l'esclave pour le maître,
 - réponse du maître par un bit d'acquiescement ACK (ou de non-acquiescement NACK),
 - (émission d'autres octets de données par l'esclave avec acquiescement du maître),
 - pour le dernier octet de données attendu par le maître, il répond par un NACK pour mettre fin au dialogue,
 - émission d'une condition de STOP par le maître (« P »).



Le MMA8452Q possède 6 sources d'interruptions :

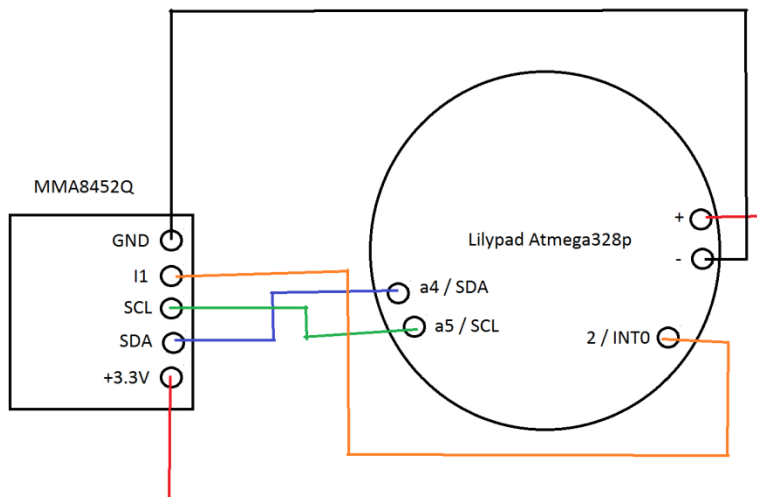
- Data-Ready Interrupt
- Singl-Pulse Interrupt
- Double-Pulse Interrupt
- Directionnal-Pulse Interrupt
- Freelfall Interrupt
- Motion Interrupt

Nous utiliserons la dernière source d'interruption, la détection de mouvement. Le principe est très simple, lorsque qu'une certaine valeur d'accélération des dépassé, l'accéléromètre génère une interruption sur la broche I1.

Nous avons décidé qu'un pas (de course) provoque une augmentation d'accélération de 1g. Il ne faut pas oublier que sur l'axe verticale, nous avons une accélération constante de 1g dû à la pesanteur. Nous allons donc initialiser les registres pour respecter tous les paramètres du cahier des charges :

- Activation des interruptions pour la détection de mouvement
- Routage de cette interruption sur la broche I1
- Activation de la résistance de pull-up
- Interruption sur front descendant
- Détection de l'accélération sur l'axe verticale
- Valeur à dépasser pour générer une interruption : 2g

Il ne nous reste plus qu'à câbler notre accéléromètre de cette façon :



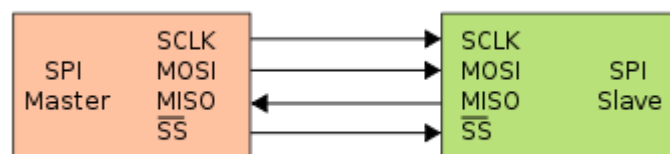
4) Puce BLE nrf8001

Pour envoyer les données des capteurs, nous utilisons un émetteur BLE nrf8001. Cet émetteur à l'avantage de s'alimenter également en 3.3V et est utilisé dans des montres, des capteurs médicaux, des capteurs pour le fitness/sport ou bien encore pour permettre la commande d'autres dispositifs. Il convient donc parfaitement à notre application.



Une fois la communication établie avec l'application Android, nous allons envoyer nos données en continue. Pour communiquer avec le microprocesseur, le nrf8001 utilise le protocole SPI :

Une liaison SPI (pour Serial Peripheral Interface) est un bus de données série synchrone qui opère en mode Full-duplex. Les circuits communiquent selon un schéma maître-esclaves, où le maître s'occupe totalement de la communication. Plusieurs esclaves peuvent coexister sur un même bus, dans ce cas, la sélection du destinataire se fait par une ligne dédiée entre le maître et l'esclave appelée chip select :



Le bus SPI utilise 4 signaux logiques :

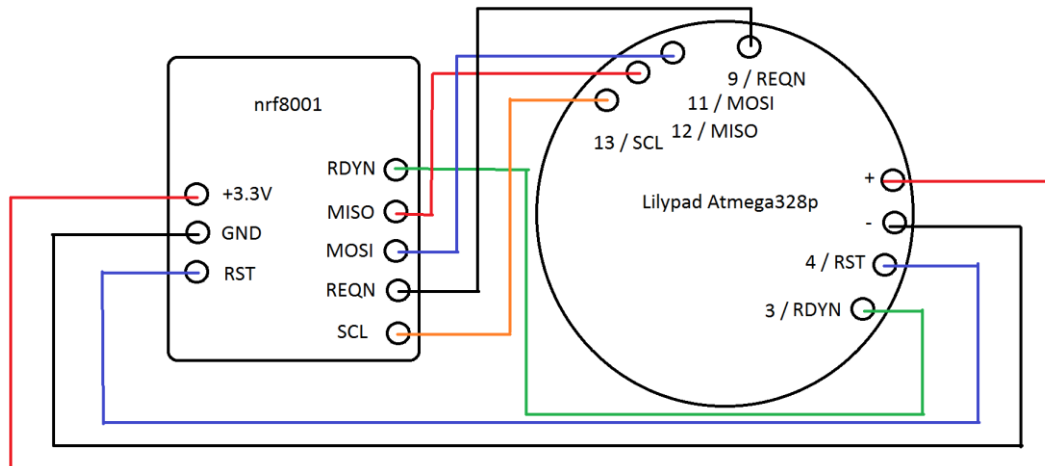
- **SCLK** — Serial Clock, Horloge (généré par le maître)
- **MOSI** — Master Output, Slave Input (généré par le maître)
- **MISO** — Master Input, Slave Output (généré par l'esclave)
- **SS** — Slave Select, Actif à l'état bas (généré par le maître)

Pour notre projet, il n'y a qu'un seul esclave, nous n'aurons donc pas besoin de gérer la broche esclave. Le principe de fonctionnement est simple, les données échangées sont des octets. La transmission s'effectue sur 2 fils monodirectionnels (nommés MOSI, MISO). Une horloge indépendante fixée par le maître synchronise les échanges (en général sur front). Il n'y a pas d'adressage des esclaves (comme sur un bus i2C par exemple). L'esclave devient actif au moyen d'une ligne de sélection dédiée (généralement active à l'état bas).

Le nrf8001 possède également deux autres broches : RDYN et REQN. La broche RDYN permet de savoir lorsque la puce BLE est prête à envoyer/recevoir des informations et la broche REQN permet le réveil du BLE.

Pour notre programme, nous avons utilisé la bibliothèque construite par NordicSemiConductor que nous avons adapté pour notre mode de fonctionnement. En effet, nous rangeons toutes nos valeurs avec le protocole que nous avons choisi dans une liste FIFO (first input, first ouput) puis les fonctions internent permettent d'envoyer les données sur notre application.

Voici le schéma de câblage :



5) Réalisation du Tee-Shirt IMAmotive

Il nous reste à présent à créer le Tee-shirt connecté. Pour se faire nous avons cousu tous les composants directement sur le Tee-shirt. Nous avons donc utilisé un fil conducteur de courant pour relier les broches entre elles comme sur les schémas des parties précédentes. Voici à quoi ressemble notre tee-shirt :



On eut voir que nous avons utilisé du fil en cuivre pour relier notre accéléromètre. En effet, le fil conducteur utilisé est très résistif, il nous a donc fallut l'utiliser sur de courtes distances. L'accéléromètre étant plus éloigné que les autres composants, nous avons utilisé des fils en cuivre pour pallier la résistivité des fils conducteurs.

Vous trouverez le code C de notre programme pour le microprocesseur en annexe.

Conclusion

Ce projet nous a permis de comprendre l'enjeu des objets connectés dans notre société. Aujourd'hui, nous en sommes entourés et le fait d'en avoir créé un, nous a permis de comprendre les principaux problèmes.

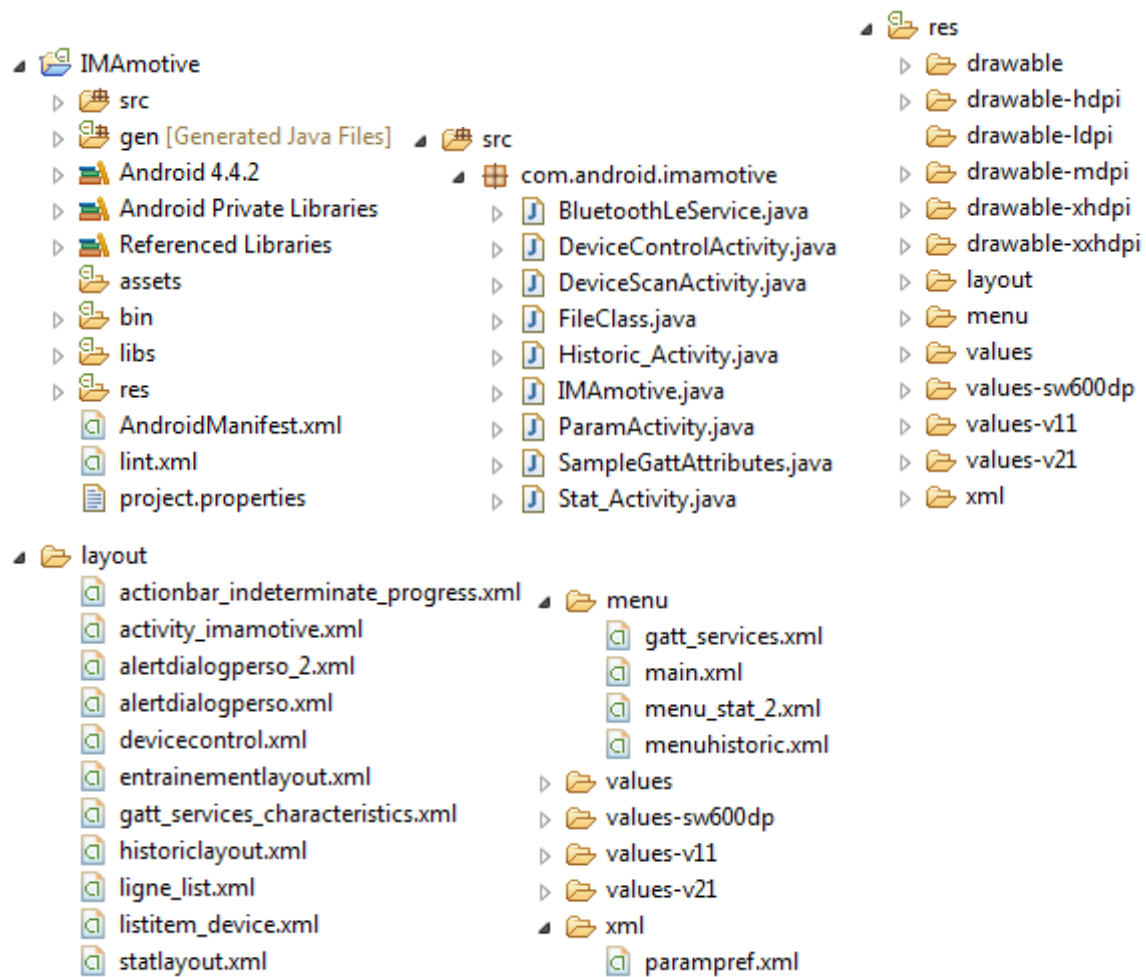
En effet, nous avons été confrontés à de nombreux problèmes tels que la communication Bluetooth, le traitement des informations... L'utilisation de capteur numérique est très pratique car elle permet de réaliser de nombreuses fonctions mais cela complique encore le traitement des informations. Le plus gros problème reste la consommation électrique de ce genre d'équipements. Les capteurs utilisés ainsi que le microprocesseur consomment beaucoup d'énergie qui empêche la portabilité du tee-shirt pendant un trop long moment.

Malgré tous ses problèmes, nous avons réussi à faire communiquer notre tee-shirt avec notre application et voir en temps réel la réception des données.

Ce projet aurait dû inclure un cardio-fréquencemètre pour avoir les battements du cœur, malheureusement nous n'avons pas reçu notre cardio-fréquencemètre dans les temps.

Annexes

Environnement



Manifest.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.imamotive"
    android:versionCode="1"
    android:versionName="1.0">
<?xml version="1.0" encoding="UTF-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.imamotive"
    android:versionCode="1"
    android:versionName="1.0">

    <uses-sdk android:minSdkVersion="18"/>

    <uses-feature android:name="android.hardware.bluetooth_le" android:required="true"/>

    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

    <application
        android:allowBackup="true"
        android:label="@string/app_name"
        android:icon="@drawable/ic_launcher"
        android:theme="@android:style/Theme.Holo.Light">

        <activity android:name=".IMAmotive"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity
            android:name=".DeviceScanActivity"
            android:label="Connection au T-Shirt" >
        </activity>
        <activity
            android:name=".DeviceControlActivity"
            android:label="Entraînement">
        </activity>

        <service
            android:name=".BluetoothLeService"
            android:enabled="true"/>

        <activity
            android:name=".Historic_Activity"
            android:label="@string/titleHistorique" >
        </activity>
        <activity
            android:name=".ParamActivity"
            android:label="@string/mesinfos" >
        </activity>

        <activity
            android:name=".Stat_Activity"
            android:label="Détails Entraînement" >
        </activity>

    </application>
</manifest>
```

Permission Bluetooth

Permission lecture/écriture

Déclaration activité Principale

Déclaration d'une activité

Parampref.xml

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >

    <EditTextPreference
        android:key="keyTextName"
        android:dialogTitle="Entrez votre nom"
        android:positiveButtonText="Valider"
        android:negativeButtonText="Annuler"
        android:title="@string/name"/>

    <EditTextPreference
        android:key="keyTextAge"
        android:dialogTitle="Entrez votre âge"
        android:positiveButtonText="Valider"
        android:negativeButtonText="Annuler"
        android:title="@string/age"/>

    <EditTextPreference
        android:key="keyTextTaille"
        android:dialogTitle="Entrez votre taille en m"
        android:positiveButtonText="Valider"
        android:negativeButtonText="Annuler"
        android:title="@string/taille"
        android:summary="en cm"/>

    <EditTextPreference
        android:key="keyTextPoid"
        android:dialogTitle="Entrez votre Poid en m"
        android:positiveButtonText="Valider"
        android:negativeButtonText="Annuler"
        android:title="@string/poid"
        android:summary="en kg"/>

</PreferenceScreen>
```

IMAmotive.java

```
public class IMAmotive extends Activity {
    private Button Entrainement_act = null, Param_act = null, /*Connect_act = null,*/ Historic_act = null;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_imamotive);
        getActionBar().setTitle("Bienvenue dans IMAmotive");
        Entrainement_act = (Button) findViewById(R.id.Button_Entr);
        Param_act = (Button) findViewById(R.id.button_Param);
        Historic_act = (Button) findViewById(R.id.button_Hist);

        Entrainement_act.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                // Le premier paramètre est le nom de l'activité actuelle
                // Le second est le nom de l'activité de destination
                Intent secondeActivite = new Intent(IMAmotive.this, DeviceScanActivity.class);

                // Puis on lance l'intent !
                startActivity(secondeActivite);
            }
        });
        Param_act.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v){
                Intent secondeActivite = new Intent(IMAmotive.this, ParamActivity.class);
                startActivity(secondeActivite);
            }
        });
        Historic_act.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent secondeActivite = new Intent(IMAmotive.this, Historic_Activity.class);

                startActivity(secondeActivite);
            }
        });
    }
}
```

DeviceScanActivity.java 1/3

```

public class DeviceScanActivity extends ListActivity {
    private LeDeviceListAdapter mLeDeviceListAdapter;
    private BluetoothAdapter mBluetoothAdapter;
    private boolean mScanning;
    private Handler mHandler;
    public static final String TAG = "DeviceListActivity";
    private static final int REQUEST_ENABLE_BT = 1;
    // Stops scanning after 10 seconds.
    private static final long SCAN_PERIOD = 10000;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getActionBar().setTitle("Connection au T-Shirt");
        getActionBar().setDisplayHomeAsUpEnabled(true);
        mHandler = new Handler();

        // Use this check to determine whether BLE is supported on the device. Then you can
        // selectively disable BLE-related features.
        if (!getPackageManager().hasSystemFeature(PackageManager.FEATURE_BLUETOOTH_LE)) {
            Toast.makeText(this, R.string.ble_not_supported, Toast.LENGTH_SHORT).show();
            finish();
        }
        // Initializes a Bluetooth adapter. For API level 18 and above, get a reference to
        // BluetoothAdapter through BluetoothManager.
        final BluetoothManager bluetoothManager = (BluetoothManager) getSystemService(Context.BLUETOOTH_SERVICE);
        mBluetoothAdapter = bluetoothManager.getAdapter();

        // Checks if Bluetooth is supported on the device.
        if (mBluetoothAdapter == null) {
            Toast.makeText(this, R.string.error_bluetooth_not_supported, Toast.LENGTH_SHORT).show();
            finish();
            return;
        }
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        if (!mScanning) {
            menu.findItem(R.id.menu_stop).setVisible(false);
            menu.findItem(R.id.menu_scan).setVisible(true);
            menu.findItem(R.id.menu_refresh).setActionView(null);
        } else {
            menu.findItem(R.id.menu_stop).setVisible(true);
            menu.findItem(R.id.menu_scan).setVisible(false);
            menu.findItem(R.id.menu_refresh).setActionView(R.layout.actionbar_indeterminate_progress);
        }
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.menu_scan:
                mLeDeviceListAdapter.clear();
                scanLeDevice(true);
                break;
            case R.id.menu_stop:
                scanLeDevice(false);
                break;
            case android.R.id.home:
                onBackPressed();
                return true;
        }
        return true;
    }
}

```

DeviceScanActivity.java 2/3

```
@Override
protected void onResume() {
    super.onResume();

    // Ensures Bluetooth is enabled on the device. If Bluetooth is not currently enabled,
    // fire an intent to display a dialog asking the user to grant permission to enable it.
    if (!BluetoothAdapter.isEnabled()) {
        if (!BluetoothAdapter.isEnabled()) {
            Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
        }
    }
    // Initializes list view adapter.
    mLeDeviceListAdapter = new LeDeviceListAdapter();
    setListAdapter(mLeDeviceListAdapter);
    scanLeDevice(true);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // User chose not to enable Bluetooth.
    if (requestCode == REQUEST_ENABLE_BT && resultCode == Activity.RESULT_CANCELED) {
        finish();
        return;
    }
    super.onActivityResult(requestCode, resultCode, data);
}

@Override
protected void onPause() {
    super.onPause();
    scanLeDevice(false);
    mLeDeviceListAdapter.clear();
}

//Passage à l'autre activité lorsque l'on clique sur un item de la liste
@Override
protected void onItemClick(ListView l, View v, int position, long id) {
    final BluetoothDevice device = mLeDeviceListAdapter.getDevice(position);
    if (device == null) return;
    final Intent intent = new Intent(this, DeviceControlActivity.class);

    if (device == null) return;
    final Intent intent = new Intent(this, DeviceControlActivity.class);
    intent.putExtra(DeviceControlActivity.EXTRAS_DEVICE_NAME, device.getName()); //Passage des paramètres à l'autre activité
    intent.putExtra(DeviceControlActivity.EXTRAS_DEVICE_ADDRESS, device.getAddress());
    if (mScanning) {
        mBluetoothAdapter.stopLeScan(mLeScanCallback);
        mScanning = false;
    }
    startActivity(intent); //Appel de l'autre activité
}

/*
As soon as you find the desired device, stop scanning.
Never scan on a loop, and set a time limit on your scan. A device that was previously
available may have moved out of range, and continuing to scan drains the battery.
*/
private void scanLeDevice(final boolean enable) {
    if (enable) {
        // Stops scanning after a pre-defined scan period.
        mHandler.postDelayed(new Runnable() {
            @Override
            public void run() {
                mScanning = false;
                mBluetoothAdapter.stopLeScan(mLeScanCallback);
                invalidateOptionsMenu();
            }
        }, SCAN_PERIOD);

        mScanning = true;
        mBluetoothAdapter.startLeScan(mLeScanCallback);
    } else {
        mScanning = false;
        mBluetoothAdapter.stopLeScan(mLeScanCallback);
    }
    invalidateOptionsMenu();
}
}
```


DeviceScanActivity.java 3/3

```
// Adapter for holding devices found through scanning.
private class LeDeviceListAdapter extends BaseAdapter {
    private ArrayList<BluetoothDevice> mLeDevices;
    private LayoutInflater mInflater;

    public LeDeviceListAdapter() {
        super();
        mLeDevices = new ArrayList<BluetoothDevice>();
        mInflater = DeviceScanActivity.this.getLayoutInflater();
    }
    public void addDevice(BluetoothDevice device) {
        if(!mLeDevices.contains(device)) {
            mLeDevices.add(device);
        }
    }
    public BluetoothDevice getDevice(int position) {
        return mLeDevices.get(position);
    }
    public void clear() {
        mLeDevices.clear();
    }
    @Override
    public int getCount() {
        return mLeDevices.size();
    }
    @Override
    public Object getItem(int i) {
        return mLeDevices.get(i);
    }
    @Override
    public long getItemId(int i) {
        return i;
    }
    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        ViewHolder viewHolder;
        // General ListView optimization code.
        if (view == null) {
            view = mInflater.inflate(R.layout.listitem_device, null);
            viewHolder = new ViewHolder();
            viewHolder.deviceAddress = (TextView) view.findViewById(R.id.device_address);
            viewHolder.deviceName = (TextView) view.findViewById(R.id.device_name);
            view.setTag(viewHolder);
        } else {
            viewHolder = (ViewHolder) view.getTag();
        }
        BluetoothDevice device = mLeDevices.get(i);
        final String deviceName = device.getName();
        if (deviceName != null && deviceName.length() > 0)
            viewHolder.deviceName.setText(deviceName);
        else
            viewHolder.deviceName.setText(R.string.unknown_device);
        viewHolder.deviceAddress.setText(device.getAddress());
        return view;
    }
}
// Device scan callback.
private BluetoothAdapter.LeScanCallback mLeScanCallback = new BluetoothAdapter.LeScanCallback() {
    @Override
    public void onLeScan(final BluetoothDevice device, int rssi, byte[] scanRecord) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                mLeDeviceListAdapter.addDevice(device);
                mLeDeviceListAdapter.notifyDataSetChanged();
            }
        });
    }
};
};
```

DeviceScanActivity.java 1/5

```

public class DeviceControlActivity extends Activity {
    private final static String TAG = DeviceControlActivity.class.getSimpleName();
    public static final String EXTRAS_DEVICE_NAME = "DEVICE_NAME";
    public static final String EXTRAS_DEVICE_ADDRESS = "DEVICE_ADDRESS";
    private String mDeviceAddress;
    private BluetoothLeService mBluetoothLeService;
    //Ecriture lecture fichier
    final String LOG_TAG = "SDactivity memoire externe";
    String monFichier;
    //Entraînement layout
    RelativeLayout rl;
    Chronometer focus;
    Button start, stop, pause, reprendre;
    TextView mTemperature, mCardiaque, mNbPas, mNbKm;
    TableLayout mtablelayout;
    long timeWhenStopped = 0;
    // Code to manage Service lifecycle.
    private final ServiceConnection mServiceConnection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName componentName, IBinder service) {
            //On affiche le bouton start une fois connecte????
            start.setVisibility(View.VISIBLE);
            mBluetoothLeService = ((BluetoothLeService.LocalBinder) service).getService();
            if (!mBluetoothLeService.initialize()) {
                Log.e(TAG, "Unable to initialize Bluetooth");
                finish();
            }
            // Automatically connects to the device upon successful start-up initialization.
            mBluetoothLeService.connect(mDeviceAddress);
        }
        @Override
        public void onServiceDisconnected(ComponentName componentName) {
            mBluetoothLeService = null;
        }
    };
    private final BroadcastReceiver mGattUpdateReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            final String action = intent.getAction();

            if (BluetoothLeService.ACTION_GATT_CONNECTED.equals(action))
            {
                //Code nrf
                runOnUiThread(new Runnable() {
                    public void run() {

                        Log.d(TAG, "UART_CONNECT_MSG");

                        byte[] value;
                        String message = "0";
                        try {
                            value = message.getBytes("UTF-8");
                            mBluetoothLeService.writeRXCharacteristic(value);
                        } catch (UnsupportedEncodingException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        }
                    }
                });
            }
            else if (BluetoothLeService.ACTION_GATT_DISCONNECTED.equals(action))
            {
                runOnUiThread(new Runnable() {
                    public void run() {
                        Log.d(TAG, "UART_DISCONNECT_MSG");
                        mBluetoothLeService.close();
                    }
                });
                invalidateOptionsMenu();
            }
        }
    }
}

```

DeviceActivity.java 2/5

```
else if (BluetoothLeService.ACTION_GATT_SERVICES_DISCOVERED.equals(action))
{
    Log.d(TAG, "UART_ACTION_GATT_SERVICES_DISCOVERED");
    mBluetoothLeService.enableTXNotification();
}
else if (BluetoothLeService.ACTION_DATA_AVAILABLE.equals(action))
{
    Log.d(TAG, "ACTION_DATA_AVAILABLE");
    final byte[] txValue = intent.getByteArrayExtra(BluetoothLeService.EXTRA_DATA);
    runOnUiThread(new Runnable() {
        public void run() {
            try {
                int bitStart = (int) txValue[0] & 0xff; //Conversion en unsignedint
                int bitStop = (int) txValue[8] & 0xff;
                long time = focus.getBase() - SystemClock.elapsedRealtime();

                int hours = (int) (time / 3600000);
                int minutes = (int) (time - hours * 3600000) / 60000;
                int seconds = (int) (time - hours * 3600000 - minutes * 60000) / 1000;
                int timeTotal = (hours*3600) + (minutes*60) + seconds; //allez savoir pourquoi la valeur total est négative

                float temperature = 0;
                int pas = 0;

                if(bitStart == 192 && bitStop == 193 )
                {
                    float bit1 = (int) txValue[1] & 0xff; //température unité
                    float bit2 = (int) txValue[2] & 0xff; //température centième
                    int freq = (int) txValue[3] & 0xff; //fréquence
                    int bit4 = (int) txValue[4] & 0xff; //pas millièmes
                    int bit5 = (int) txValue[5] & 0xff; //pas centaine
                    int bit6 = (int) txValue[6] & 0xff; //pas dizaine
                    int bit7 = (int) txValue[7] & 0xff; //pas unite

                    temperature = bit1+(bit2/100);
                    pas = bit4*1000+bit5*100+bit6*10+bit7;

                    String donnee = String.valueOf(-timeTotal) + " " + String.valueOf(temperature)+ " " + String.valueOf(freq) + " " + String.valueOf(pas);

                    FileClass.ecrireFichierE(monFichier + ".txt", donnee);

                    mTemperature.setText(String.valueOf(temperature));
                    mCardiaque.setText(String.valueOf(freq));
                    mNbPas.setText(String.valueOf(pas));

                    int nbCm = pas*70;

                    int nbKm = nbCm/100000;
                    int nbM = (nbCm-nbKm*100000)/100;

                    String Distance;
                    if(nbKm>0)
                        Distance = Integer.toString(nbKm) + "," + Integer.toString(nbM) + " km";
                    else if(nbM>0)
                        Distance = Integer.toString(nbM)+ " " + Integer.toString(nbCm-(nbM*100)) + " m"; - " m";
                    else
                        Distance = Integer.toString(nbCm)+"cm";
                    mNbKm.setText(Distance);
                }
            } catch (Exception e) {
                Log.e(TAG, e.toString());
            }
        }
    });
}
if (action.equals(BluetoothLeService.DEVICE_DOES_NOT_SUPPORT_UART)){
    showMessage("Device doesn't support UART. Disconnecting");
    mBluetoothLeService.disconnect();
}
};
}
```

DeviceActivity.java 3/5

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.entraînementlayout);
    getActionBar().setTitle("Entraînement");

    rl = (RelativeLayout) findViewById(R.id.layoutentr);

    start = (Button) findViewById(R.id.button_start);
    stop = (Button) findViewById(R.id.button_stop);
    pause = (Button) findViewById(R.id.button_pause);
    reprendre = (Button) findViewById(R.id.button_reprendre);
    mTemperature = (TextView) findViewById(R.id.text_temperature);
    mCardiaque = (TextView) findViewById(R.id.text_cardiaque);
    mNbPas = (TextView) findViewById(R.id.text_nbpas);
    mNbKm = (TextView) findViewById(R.id.text_km);
    mtablelayout = (TableLayout) findViewById(R.id.tablelayout);

    focus = new Chronometer (DeviceControlActivity.this);

    RelativeLayout.LayoutParams params = new RelativeLayout.LayoutParams ((int) LayoutParams.WRAP_CONTENT, (int) LayoutParams.WRAP_CONTENT);
    params.addRule(RelativeLayout.CENTER_IN_PARENT);

    focus.setLayoutParams(params);

    rl.addView(focus);

    //On cache tout puis on affiche start lorsque c'est connecté

    start.setVisibility(View.VISIBLE);
    focus.setVisibility(View.GONE);
    stop.setVisibility(View.GONE);
    pause.setVisibility(View.GONE);
    reprendre.setVisibility(View.GONE);
    mtablelayout.setVisibility(View.GONE);

    Intent gattServiceIntent = new Intent(this, BluetoothLeService.class);
    bindService(gattServiceIntent, mServiceConnection, BIND_AUTO_CREATE);

    //AlertDialog//
    //On instancie notre layout en tant que View
    LayoutInflater factory = LayoutInflater.from(this);
    final View alertDialogView = factory.inflate(R.layout.alertdialogperso, null);
    //Création de l'AlertDialog
    final AlertDialog.Builder adb = new AlertDialog.Builder(this);
    //On affecte la vue personnalisée que l'on a créée à notre AlertDialog
    adb.setView(alertDialogView);
    //On donne un titre à l'AlertDialog
    adb.setTitle("Entraînement terminé");
    //On affecte un bouton type "OK" à notre AlertDialog et on lui affecte un évènement
    adb.setPositiveButton("Sauvegarder", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            //Création de l'intent pour passer à l'activité "Stat_Activity"
            Intent secondeActivite = new Intent(DeviceControlActivity.this, Stat_Activity.class);
            //On indique quelle fichier doit traiter l'activité en lui passant son nom en paramètre
            secondeActivite.putExtra(Stat_Activity.EXTRAS_ID_FILE, monFichier+".txt");
            startActivity(secondeActivite);
            DeviceControlActivity.this.finish();
        }
    });
    //On crée un bouton de type "Annuler" à notre AlertDialog et on lui affecte un évènement
    adb.setNegativeButton("Supprimer", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            //On supprime le fichier qui vient d'être créé
            File sdLien = Environment.getExternalStorageDirectory();
            File Fichier = new File(sdLien + File.separator + "Projet_Tshirt"+ File.separator + monFichier+".txt");
            Log.d("Fichier SD", "path de fichier:" + monFichier);
            if(!Fichier.exists()) {
                throw new RuntimeException("Fichier inexistant sur la carte sd");
            }
            Fichier.delete();
            //Lorsque l'on cliquera sur annuler on quittera l'application
            finish();
        }
    });
}

```

DeviceActivity.java 4/5

```
start.setOnClickListener(new View.OnClickListener() {
    @SuppressWarnings("SimpleDateFormat") @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        stop.setVisibility(View.VISIBLE);
        pause.setVisibility(View.VISIBLE);
        focus.setVisibility(View.VISIBLE);
        mtablelayout.setVisibility(View.VISIBLE);
        start.setVisibility(View.GONE);

        mTemperature.setText("---");
        mCardiaque.setText("---");
        mNbPas.setText("---");
        mNbKm.setText("---");

        reprendre.setVisibility(View.GONE);
        timewhenStopped = 0; //securite
        focus.setBase(SystemClock.elapsedRealtime());
        focus.start();

        //connection au T-shirt
        registerReceiver(mGattUpdateReceiver, makeGattUpdateIntentFilter());
        if (mBluetoothLeService != null) {
            final boolean result = mBluetoothLeService.connect(mDeviceAddress);
            Log.d(TAG, "Connect request result=" + result);
        }

        //Récupération de l'adresse du Device
        final Intent intent = getIntent();
        mDeviceAddress = intent.getStringExtra(EXTRAS_DEVICE_ADDRESS);
        mBluetoothLeService.connect(mDeviceAddress);

        //Envoie d'un message au T-shirt pour lui demander l'envoi de ces données
        String message = "1";
        byte[] value;
        try {
            //Envoie des datas au service
            value = message.getBytes("UTF-8");
            mBluetoothLeService.writeRXCharacteristic(value);
        } catch (UnsupportedEncodingException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        //Création du fichier en nommant le String "monFichier"
        SimpleDateFormat formatter = new SimpleDateFormat("yy_MM_dd_HH_mm_ss");
        String currentDateTimeString = formatter.format(new Date());
        //Ecrire la donnee recu dans une variable string globale pour ensuite ecrire dans le fichier une fois la connection arrete
        monFichier = currentDateTimeString;
        Log.i(LOG_TAGE, "nom de fichier:" + monFichier); //current date string sera le nom du fichier
    }
});

//Reprise des calculs
reprendre.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        reprendre.setVisibility(View.GONE);
        pause.setVisibility(View.VISIBLE);

        focus.setBase(SystemClock.elapsedRealtime() + timewhenStopped);
        focus.start();
        mBluetoothLeService.connect(mDeviceAddress);
    }
});
```

DeviceControlActivity.java 5/5

```
pause.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        pause.setVisibility(View.GONE);
        reprendre.setVisibility(View.VISIBLE);

        timeWhenStopped = focus.getBase() - SystemClock.elapsedRealtime();
        focus.stop();
        mBluetoothLeService.disconnect();
    }
});
//Arret des calculs, on demande à l'utilisateur si souhaite arreter l'entrainement puis go Stat_activity
stop.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        timeWhenStopped = focus.getBase() - SystemClock.elapsedRealtime();//On recupere le temps final pour l'envoyer dans la bdd
        focus.stop();
        focus.setBase(SystemClock.elapsedRealtime());
        //Disconnect button pressed
        mBluetoothLeService.disconnect();
        adb.show();
    }
});
}
@Override
protected void onResume() {
    super.onResume();
    Log.d(TAG, "onResume");
    registerReceiver(mGattUpdateReceiver, makeGattUpdateIntentFilter());
    if (mBluetoothLeService != null) {
        final boolean result = mBluetoothLeService.connect(mDeviceAddress);
        Log.d(TAG, "Connect request result=" + result);
    }
}
@Override
protected void onPause() {
    super.onPause();
    unregisterReceiver(mGattUpdateReceiver);
}
@Override
protected void onDestroy() {
    super.onDestroy();
    unbindService(mServiceConnection);
    mBluetoothLeService = null;
}
private static IntentFilter makeGattUpdateIntentFilter() {
    final IntentFilter intentFilter = new IntentFilter();
    intentFilter.addAction(BluetoothLeService.ACTION_GATT_CONNECTED);
    intentFilter.addAction(BluetoothLeService.ACTION_GATT_DISCONNECTED);
    intentFilter.addAction(BluetoothLeService.ACTION_GATT_SERVICES_DISCOVERED);
    intentFilter.addAction(BluetoothLeService.ACTION_DATA_AVAILABLE);
    return intentFilter;
}
}
```

Historique_Activity.java 1/4

```
public class Historique_Activity extends ListActivity {
    File[] mesFichiers;
    ListView maListViewPerso;
    String idFichier;
    View alertDialogView;
    AlertDialog.Builder adb;
    private SelectionAdapter mAdapter;
    Set<Integer> setPos;
    int Taille;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.historiclayout);

        getActionBar().setDisplayHomeAsUpEnabled(true);
        getActionBar().setTitle("Votre historique");
        //On instancie notre layout en tant que View
        LayoutInflater factory = LayoutInflater.from(this);
        alertDialogView = factory.inflate(R.layout.alertdialogperso_2, null);

        //Création de l'AlertDialog
        adb = new AlertDialog.Builder(this);
        //On affecte la vue personnalisé que l'on a crée à notre AlertDialog
        //((ViewGroup) alertDialogView.getParent()).removeView(alertDialogView);
        adb.setView(alertDialogView);
        //On donne un titre à l'AlertDialog
        adb.setTitle("Voulez vous vraiment supprimer le(s) fichier(s)?");

        //On affecte un bouton "OK" à notre AlertDialog et on lui affecte un évènement
        adb.setPositiveButton("Oui", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                Log.i("AlertDialog", "OUI");
                //On parcour toutes les positions sélectionné à l'aide du tableau setPos
                for(int posItem:setPos)
                {
                    Log.i("Set", Integer.toString(posItem));
                    int i=0;
                    //Puis on parcour tout les fichiers
                    for(File monFichier : mesFichiers)
                    {
                        //Et on match si ce sont les même positions
                        if(i==posItem)
                        {
                            monFichier.delete();
                        }
                        i++;
                    }
                }
            }
        });
        onResume();
    }
}

//On crée un bouton "supprimer" à notre AlertDialog et on lui affecte un évènement
adb.setNegativeButton("Non", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
        //On ne fait rien
    }
});
}
```

Historique_Activity.java 2/4

```
public void onResume() {
    super.onResume();
    Log.i("TEST","On resume");
    //On récupère les fichiers du dossier "Projet_Tshirt"
    mesFichiers = FileClass.recupFichiers();
    Log.i("TEST","FileClass");

    //Pour trier le tableau de File:
    Arrays.sort(mesFichiers);

    Taille = 0;
    for(File monFichier : mesFichiers)
    {
        Taille++;
    }
    String[] donnee = new String[Taille];
    int i = 0;
    for(File monFichier : mesFichiers)
    {
        int id = i+1;
        String name = "Entrainement n°" + id;
        String date = (monFichier.getName() != null) ? monFichier.getName().substring(0,monFichier.getName().indexOf('.')):""; //Enlève le ".txt"
        Log.i("date",date);
        String[] dateSplit;
        dateSplit = date.split("_");

        date = dateSplit[2] + "/" + dateSplit[1] + "/" + dateSplit[0] + " " + dateSplit[3] + ":" + dateSplit[4] + ":" + dateSplit[5];

        donnee[i] = name + "\n" + date;
        Log.i("Donne",donnee[i]);
        i++;
    }
    mAdapter = new SelectionAdapter(this,R.layout.ligne_list, R.id.textView1, donnee);
    setListAdapter(mAdapter);
    getListView().setChoiceMode(ListView.CHOICE_MODE_MULTIPLE_MODAL);

    getListView().setMultiChoiceModeListener(new MultiChoiceModeListener() {
        private int nr = 0;
        @Override
        public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
            Log.i("GetListView","ListView 1");
            // TODO Auto-generated method stub
            return false;
        }
        @Override
        public void onDestroyActionMode(ActionMode mode) {
            // TODO Auto-generated method stub
            Log.i("GetListView","ListView 2");
            mAdapter.clearSelection();
        }
        @Override
        public boolean onCreateActionMode(ActionMode mode, Menu menu) {
            // TODO Auto-generated method stub
            Log.i("GetListView","ListView 3");
            nr = 0;
            MenuInflater inflater = getMenuInflater();
            inflater.inflate(R.menu.menuhistoric, menu);
            return true;
        }
        @Override
        public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
            Log.i("GetListView","ListView 4");
            // TODO Auto-generated method stub
            switch (item.getItemId()) {
                case R.id.item_delete:
                    Log.i("Delete","delet item check");
                    //setPos.clear();
                    setPos = mAdapter.getCurrentCheckedPosition();
                    adb.show();
                    if ((ViewGroup)alertDialogView.getParent() != null)
                    {
                        ((ViewGroup)alertDialogView.getParent()).removeView(alertDialogView);
                    }
                    nr = 0;
                    mAdapter.clearSelection();
                    mode.finish();
            }
            return false;
        }
    });
}
```


Historique_Activity.java 3/4

```

@Override
public void onItemCheckedStateChanged(ActionMode mode, int position,
    long id, boolean checked) {
    Log.i("GetListView", "ListView 5");
    // TODO Auto-generated method stub
    if (checked) {
        nr++;
        mAdapterer.setNewSelection(position, checked);
    } else {
        nr--;
        mAdapterer.removeSelection(position);
    }
    mode.setTitle(nr + " selected");
}
});
getListView().setOnItemLongClickListener(new OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(AdapterView<?> arg0, View arg1,
        int position, long arg3) {
        // TODO Auto-generated method stub

        getListView().setItemChecked(position, !mAdapter.isPositionChecked(position));
        return false;
    }
});
getListView().setOnItemClickListener(new OnItemClickListener(){
    @Override
    public void onItemClick(AdapterView<?> a, View v, int position, long id) {
        Log.i("OnClick", (String) getListView().getItemAtPosition(position));
        //on récupère la HashMap contenant les infos de notre item (titre, description, img)
        Long positionItem = getListView().getItemIdAtPosition(position);
        Log.i("OnClick", "i: " + Long.toString(positionItem));
        int i=0;
        String iden;
        for(File monFichier : mesFichiers)
        {
            if(i==positionItem)
            {
                iden = monFichier.getName();
                Intent secondeActivite = new Intent(Historic_Activity.this, Stat_Activity.class);
                //Envoie du nom du fichier en paramètre
                secondeActivite.putExtra(Stat_Activity.EXTRAS_ID_FILE, iden);
                startActivity(secondeActivite);
            }
            i++;
        }
    }
});
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch(item.getItemId()) {
        case android.R.id.home:
            onBackPressed();
            return true;
    }
    return super.onOptionsItemSelected(item);
}
}

```

Historique_Activity.java 4/4

```
private class SelectionAdapter extends ArrayAdapter<String> {
    private HashMap<Integer, Boolean> mSelection = new HashMap<Integer, Boolean>();

    public SelectionAdapter(Context context, int resource, int textViewResourceId, String[] objects) {
        super(context, resource, textViewResourceId, objects);
    }
    public void setNewSelection(int position, boolean value) {
        mSelection.put(position, value);
        Log.i("setNewSelection", Integer.toString(position));
        notifyDataSetChanged();
    }
    public boolean isPositionChecked(int position) {
        Boolean result = mSelection.get(position);
        return result == null ? false : result;
    }
    public Set<Integer> getCurrentCheckedPosition() {
        return mSelection.keySet();
    }
    public void removeSelection(int position) {
        mSelection.remove(position);
        notifyDataSetChanged();
    }
    public void clearSelection() {
        mSelection = new HashMap<Integer, Boolean>();
        notifyDataSetChanged();
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View v = super.getView(position, convertView, parent); //let the adapter handle setting up the row views
        v.setBackgroundColor(getResources().getColor(android.R.color.background_light)); //default color

        if (mSelection.get(position) != null) {
            v.setBackgroundColor(getResources().getColor(android.R.color.holo_blue_light)); // this is a selected position so make it red
        }
        return v;
    }
}
```

Stat_Activity.java 1/3

```
public class Stat_Activity extends Activity{
    public static final String EXTRAS_ID_FILE = "ID_FILE";
    String Id_File;
    Button button_Temp,button_Card;
    TextView mDate, mHeure, mTime, mTemp, mCard, mNbPas, mnbKm;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.statlayout);
        getActionBar().setDisplayHomeAsUpEnabled(true);

        mDate = (TextView) findViewById(R.id.text_date);
        mHeure = (TextView) findViewById(R.id.text_heure);
        mTime = (TextView) findViewById(R.id.text_time);
        mTemp = (TextView) findViewById(R.id.text_temperature);
        mCard = (TextView) findViewById(R.id.text_cardiaque);
        mNbPas = (TextView) findViewById(R.id.text_nbpas);
        mnbKm = (TextView) findViewById(R.id.text_km);

        button_Temp = (Button) findViewById(R.id.button_temp);
        button_Card = (Button) findViewById(R.id.button_card);

        //récupération du fichier à analyser:
        final Intent secondeActivite = getIntent();
        Id_File = secondeActivite.getStringExtra(EXTRAS_ID_FILE);

        String[] dateSplit;
        String Id_ , date, heure;

        Id_ = Id_File.substring(0,Id_File.indexOf('.'));

        dateSplit = Id_.split("_");

        date = dateSplit[2] + "/" + dateSplit[1] + "/" + dateSplit[0] ;
        heure = dateSplit[3] + ":" + dateSplit[4] + ":" + dateSplit[5];
        mDate.setText(date);
        mHeure.setText(heure);
        //On récupère le nombre de ligne
        int nbLineText = FileClass.countLineFichierE(Id_File);
        Log.i("Stat","Nombres de ligne du fichier: " + nbLineText);

        String[] textColonne,textLigne; //Tableau de string pour chaque ligne et chaque colonne
        //On récupère chaque ligne du fichier
        textLigne = FileClass.lireFichierE(Id_File);

        int cardMoy = 0;
        float tempMoy = 0;

        String timeTotal="", nbPasTotal="";

        int[] timeTable = new int[nbLineText]; //Tableau regroupant toutes les valeur de temps lu dans le fichier
        float[] tempTable = new float[nbLineText]; //Toutes les valeurs de température
        int[] cardTable = new int[nbLineText]; //Valeurs de la fréquence cardiaque

        //Boucle pour parcourir le fichier ligne par ligne
        for (int x=0; x<nbLineText; x++)
        {
            //On récupère les colonnes en splitant le string ligne
            String ligne = textLigne[x];
            textColonne = ligne.split("\\s");

            //TEMPS
            timeTable[x] = Integer.parseInt(textColonne[0]);
            Log.i("Colonne","Time : " + timeTable[x]);

            //TEMPERATURE
            tempMoy += Float.parseFloat(textColonne[1]);
            tempTable[x] = Float.parseFloat(textColonne[1]);
            Log.i("Colonne","Temperature : " + tempTable[x]);

            //FREQUENCE CARDIAQUE
            cardMoy += Integer.parseInt(textColonne[2]);
            cardTable[x] = Integer.parseInt(textColonne[2]);
            Log.i("Colonne","textColonne : " + cardTable[x]);
        }
    }
}
```

Stat_Activity.java 2/3

```
//DERNIER LIGNE
if(x==(nbLineText-1))
{
    timeTotal = textColonne[1];
    nbPasTotal = textColonne[3];
}
}

String Time = ConverTime(timeTotal); // Cette fonction est définie plus bas dans le fichier.

mTime.setText(Time);
mNbPas.setText(nbPasTotal);

int nbCm = Integer.parseInt(nbPasTotal)*70;

int nbKm = nbCm/100000;
int nbM = (nbCm-nbKm*100000)/100;

String Distance;
if(nbKm>0)
    Distance = Integer.toString(nbKm) + "," + Integer.toString(nbM) + " km";
else if(nbM>0)
    Distance = Integer.toString(nbM) + "," + Integer.toString(nbCm-(nbM*100)) + " m";
else
    Distance = Integer.toString(nbCm)+"cm";

mnbKm.setText(Distance);

//Température et rythme cardiaque moyenne:
tempMoy = tempMoy/nbLineText;
String tempMoyString = Float.toString(tempMoy);

tempMoyString = tempMoyString.substring(0,5);

mTemp.setText(tempMoyString + "°C");

cardMoy = cardMoy/nbLineText;
String cardMoyString = Integer.toString(cardMoy);
mCard.setText(cardMoyString);

//Graph de la fréquence par rapport au temps
final GraphView graph_card = (GraphView) findViewById(R.id.graph_card);
LineGraphSeries<DataPoint> series_card = null;
DataPoint[] dataPoint = new DataPoint[nbLineText];
for (int x=0; x<nbLineText; x++)
{
    dataPoint[x] = new DataPoint(timeTable[x],cardTable[x]);
}
series_card = new LineGraphSeries<DataPoint>(dataPoint);
graph_card.setTitle("Fréquence cardiaque");
graph_card.addSeries(series_card);
graph_card.setVisibility(View.VISIBLE);

//Graph de la température par rapport au temps
final GraphView graph_temp = (GraphView) findViewById(R.id.graph_temp);
LineGraphSeries<DataPoint> series_temp = null;
DataPoint[] dataPoint2 = new DataPoint[nbLineText];
for (int x=0; x<nbLineText; x++)
{
    dataPoint2[x] = new DataPoint(timeTable[x],tempTable[x]);
}
series_temp = new LineGraphSeries<DataPoint>(dataPoint2);
graph_temp.setTitle("Température du corps (°C)");
graph_temp.addSeries(series_temp);

graph_temp.setVisibility(View.GONE);
button_Card.setVisibility(View.GONE);
button_Card.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        graph_card.setVisibility(View.VISIBLE);
        graph_temp.setVisibility(View.GONE);
    }
});
```

Stat_Activity.java 3/3

```
        button_Card.setVisibility(View.GONE);
        button_Temp.setVisibility(View.VISIBLE);
    }
});
button_Temp.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        graph_card.setVisibility(View.GONE);
        graph_temp.setVisibility(View.VISIBLE);
        button_Card.setVisibility(View.VISIBLE);
        button_Temp.setVisibility(View.GONE);
    }
});
}
//Fonction permettant de convertir le temps "00:00" en seconde de type int
//Attention, l'exercice doit durer moins d'une heure!!
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch(item.getItemId()) {
        case android.R.id.home:
            onBackPressed();
            return true;
    }
    return super.onOptionsItemSelected(item);
}
public String ConverTime(String time)
{
    String timeString;
    int secondes = Integer.parseInt(time);
    int heure, minute;
    heure = secondes/3600;
    minute = (secondes-(heure*3600))/60;
    secondes = secondes - (heure*3600) - (minute*60);
    if (heure > 0)
        timeString = Integer.toString(heure)+":"+Integer.toString(minute)+":"+Integer.toString(secondes);
    else if (minute > 0)
        timeString = Integer.toString(minute)+":"+Integer.toString(secondes);
    else
        timeString = Integer.toString(secondes)+ " secondes";
    return timeString;
}
}
```

Programme Lilyypad

```
/**
 * Initialise la puce BLE avec le header "service.h"
 */
#include "services.h"

/**
 * Variables pour la communication Bluetooth
 */
#ifdef SERVICES_PIPE_TYPE_MAPPING_CONTENT
static services_pipe_type_mapping_t
services_pipe_type_mapping[NUMBER_OF_PIPES] = SERVICES_PIPE_TYPE_MAPPING_CONTENT;
#else
#define NUMBER_OF_PIPES 0
static services_pipe_type_mapping_t * services_pipe_type_mapping = NULL;
#endif

static const hal_aci_data_t setup_msgs[NB_SETUP_MESSAGES] PROGMEM = SETUP_MESSAGES_CONTENT;
static hal_aci_evt_t aci_data;
static volatile byte ack = 0;

/* Buffer pour l'envoi et la reception de donnees */
#define MAX_TX_BUFF 64
static uint8_t tx_buff[MAX_TX_BUFF];
static uint8_t tx_buffer_len = 0;

#define MAX_RX_BUFF 64

static uint8_t rx_buff[MAX_RX_BUFF+1];
static uint8_t rx_buffer_len = 0;
static uint8_t *p_before = &rx_buff[0] ;
static uint8_t *p_back = &rx_buff[0];

/* Define how assert should function in the BLE library */
void __ble_assert(char *file, uint16_t line)
{
    print("ERROR ");
    print(file);
    print(": ");
    print_int(line);
    while(1);
}

/* Variables Globales pour le comptage du nombre de pas*/
uint16_t pas;
int mode_connect =0;

void setup(void)
{
    // Initialisation de l'interruption pour le nombre de pas
    EICRA |= (1 << ISC00);
    EIMSK |= (1 << INT0);

    /**
     * Initialise tout les services du BLE grace au header service.h
     */
    if (NULL != services_pipe_type_mapping)
    {
        aci_state.aci_setup_info.services_pipe_type_mapping = &services_pipe_type_mapping[0];
    }
    else
    {
        aci_state.aci_setup_info.services_pipe_type_mapping = NULL;
    }
    aci_state.aci_setup_info.number_of_pipes = NUMBER_OF_PIPES;
    aci_state.aci_setup_info.setup_msgs = (hal_aci_data_t*)setup_msgs;
    aci_state.aci_setup_info.num_setup_msgs = NB_SETUP_MESSAGES;
}
```

```

/**
 * Initialise les broches de l'atmega
 */
aci_state.aci_pins.board_name = BOARD_DEFAULT;
aci_state.aci_pins.reqn_pin = 9;
aci_state.aci_pins.rdyn_pin = 3;
aci_state.aci_pins.mosi_pin = 11;
aci_state.aci_pins.miso_pin = 12;
aci_state.aci_pins.sck_pin = 13;

aci_state.aci_pins.spi_clock_divider = SPI_CLOCK_DIV8;

aci_state.aci_pins.reset_pin = 4;
aci_state.aci_pins.active_pin = UNUSED;
aci_state.aci_pins.optional_chip_sel_pin = UNUSED;
aci_state.aci_pins.interface_is_interrupt = false;
aci_state.aci_pins.interrupt_number = 3;

lib_aci_init(&aci_state, false);
}

/**
 * Detecte les evenement du BLE
 */
void aci_loop()
{
    static bool setup_required = false;
    int i=0;

    if (lib_aci_event_get(&aci_state, &aci_data))
    {
        aci_evt_t * aci_evt;
        aci_evt = &aci_data.evt;
        switch(aci_evt->evt_opcode)
        {
            case ACI_EVT_DEVICE_STARTED: //BLE Connecte
            {
                aci_state.data_credit_total = aci_evt->params.device_started.credit_available;
                switch(aci_evt->params.device_started.device_mode)
                {
                    case ACI_DEVICE_SETUP: //BLE Initialise
                        setup_required = true;
                        break;
                    case ACI_DEVICE_STANDBY:
                        if (aci_evt->params.device_started.hw_error)
                        {
                            _delay_ms(20);
                        }
                        else
                        {
                            lib_aci_connect(0/* in seconds : 0 means forever */, 0x0050 /* advertising interval
50ms*/);
                        }
                        break;
                    default:
                        break;
                }
            }
            break; //BLE pret a la reception et l envoi de donnees

            case ACI_EVT_CMD_RSP:
                if (ACI_CMD_GET_DEVICE_VERSION == aci_evt->params.cmd_rsp.cmd_opcode)
                {
                    lib_aci_set_local_data(&aci_state, PIPE_DEVICE_INFORMATION_HARDWARE_REVISION_STRING_SET,
                    (uint8_t *)&aci_evt->params.cmd_rsp.params.get_device_version), sizeof
(aci_evt_cmd_rsp_params_get_device_version_t));
                }
                break;

            case ACI_EVT_CONNECTED: //BLE connecte au telephone
                mode_connect = 1;
                aci_state.data_credit_available = aci_state.data_credit_total;
                lib_aci_device_version();
                break;

            case ACI_EVT_PIPE_STATUS:
                if (lib_aci_is_pipe_available(&aci_state, PIPE_UART_OVER_BTLE_UART_TX_TX))
                {
                    lib_aci_change_timing_GAP_PPCP();
                }
                break;
        }
    }
}

```

```

case ACI_EVT_TIMING:
    lib_aci_set_local_data(&aci_state,
        PIPE_UART_OVER_BTLE_UART_LINK_TIMING_CURRENT_SET,
        (uint8_t *)&(aci_evt->params.timing.conn_rf_interval),
        PIPE_UART_OVER_BTLE_UART_LINK_TIMING_CURRENT_SET_MAX_SIZE);

    break;

case ACI_EVT_DISCONNECTED: //BLE deconnecte
    //Attente d'une nouvelle connection
    lib_aci_connect(0/* in seconds */, 0x0050 /* advertising interval 50ms*/);

    break;

case ACI_EVT_DATA_RECEIVED: //Reception de Donnee
    if (PIPE_UART_OVER_BTLE_UART_RX_RX == aci_evt->params.data_received.rx_data.pipe_number)
    {
        if(rx_buffer_len == MAX_RX_BUFFER)
        {
            break;
        }
        else
        {
            if(p_back == &rx_buff[MAX_RX_BUFFER])
            {
                p_back = &rx_buff[0];
            }
            *p_back = aci_evt->params.data_received.rx_data.aci_data[i];
            rx_buffer_len++;
            p_back++;
        }
    }
    break;

case ACI_EVT_DATA_CREDIT:
    aci_state.data_credit_available = aci_state.data_credit_available + aci_evt->params.data_credit.credit;
    ack=1;

    break;

case ACI_EVT_PIPE_ERROR:
    if (ACI_STATUS_ERROR_PEER_ATT_ERROR != aci_evt->params.pipe_error.error_code)
    {
        aci_state.data_credit_available++;
    }
    break;

case ACI_EVT_HW_ERROR:
    //Erreur lors de la connection, attente d'une nouvelle connection
    lib_aci_connect(0/* in seconds, 0 means forever */, 0x0050 /* advertising interval 50ms*/);
    printf("Advertising started. Tap Connect on the nRF UART app");
    break;
default:
    break;
}
}
else
{
    /**
    *Pas d'evenement, attente d'un nouvel evenement
    */
}

if(setup_required)
{
    if (SETUP_SUCCESS == do_aci_setup(&aci_state))
    {
        setup_required = false;
    }
}
}

```



```

/**
 * Fonction pour envoyer les donnees
 * Les donnee sont stockes dans un tableau de taille 64 puis envoyer par le BLE avec la fonction ble_do_events
 */
void ble_write(unsigned char data)
{
    if(tx_buffer_len == MAX_TX_BUFF)
    {
        return;
    }
    tx_buff[tx_buffer_len] = data;
    tx_buffer_len++;
}

void ble_write_bytes(unsigned char *data, uint8_t len)
{
    int i=0;
    for (i = 0; i < len; i++)
        ble_write(data[i]);
}

/**
 * Lit les donnees recu par le BLE
 * Les donnees recut sont stockes dans un tableau de taille 64
 * Il suffit de faire un appel a la fonction pour recuperer les donnees recut
 */
int ble_read()
{
    int data;
    if(rx_buffer_len == 0) return -1;
    if(p_before == &rx_buff[MAX_RX_BUFF])
    {
        p_before = &rx_buff[0];
    }
    data = *p_before;
    p_before ++;
    rx_buffer_len--;
    return data;
}

unsigned char ble_connected()
{
    return mode_connect;
}

/**
 * Cette fonction permet de generer l envois de
 * toutes les donnees stockes dans le tableau de la fonction ble_write()
 */
void ble_do_events()
{
    if (lib_aci_is_pipe_available(&aci_state, PIPE_UART_OVER_BTLE_UART_TX_TX))
    {
        if(tx_buffer_len > 0)
        {
            unsigned char Index = 0;
            while(tx_buffer_len > 20)
            {
                lib_aci_send_data(PIPE_UART_OVER_BTLE_UART_TX_TX, &tx_buff[Index], 20);
                tx_buffer_len -= 20;
                Index += 20;
                aci_state.data_credit_available--;
                ack = 0;
                while (!ack)
                    aci_loop();
            }

            lib_aci_send_data(PIPE_UART_OVER_BTLE_UART_TX_TX,& tx_buff[Index], tx_buffer_len);
            tx_buffer_len = 0;
            aci_state.data_credit_available--;
            ack = 0;
            while (!ack)
                aci_loop();
        }
    }
    aci_loop();
}

```

```

/**
 * Fonction pour recuperer la temperature du capteur TMP36GZ
 * Ajoute les valeurs du capteur dans le buffer a envoyer
 */
void send_temp(void) {
    unsigned int value=0;
    float temp =0;

    ad_init(0);
    value=ad_sample(); //temperature sur 10bits
    temp=(3.3*value)/255; //temperature en volt
    temp=(temp*25)/0.75; //temperature en °C

    ble_write((int)temp);

    temp=temp-(int)temp;
    temp=temp*100;

    ble_write((int)temp);
}

/**
 * Cette fonction simule le cardiofrequencemetre que nous devons ajouter a notre tee-shirt
 */
void send_frequence(uint8_t* freq) {
    *freq+=2;
    if(*freq==200) *freq=150;
    ble_write(*freq);
}

/**
 * Ajoute les valeurs du nombre pas dans le buffer sous la forme : millier, centaine, dizaine, unite
 */
void send_pas() {
    int millier=0,centaine=0,dizaine=0,unit=0;
    millier=pas/1000;
    centaine=(pas-millier*1000)/100;
    dizaine=(pas-(millier*1000)-(centaine*100))/10;
    unit=pas-(millier*1000)-(centaine*100)-(dizaine*10);

    ble_write(millier);
    ble_write(centaine);
    ble_write(dizaine);
    ble_write(unit);
}

/**
 * Fonction d'interruption qui recupere l interruption engendre par l accelerometre et incremente le compteur pas
 */
ISR (INT0_vect)
{
    pas++;
    cli();
}

int main() {

    //Definition des variables
    uint8_t frequence=100;
    pas=0;

    //Initialisation des registres de l accelerometre
    setup_MMA8452();

    //Initialisation du BLE
    setup();

    //Boucle infini pour l envoi des donnees
    while(1) {
        sei();

        //Recupere les evenement du BLE
        aci_loop();

        /**
         * Si le BLE est connecte, alors on envoit les donnees
         * Pour l'envoi des donnees, on commence par envoyer 0xC0
         * On envoit ensuite la temperature sur deux octets, le premier octets correspond aux entiers et le deuxieme
         aux decimales
         * Puis la frequence sur un octet
         * Le nombre de pas sur 4 octets avec les milliers, les centaines, les dizaines puis les unites
         * Puis on finit par un octet 0xC1
         */
    }
}

```

```
if(ble_connected()) {
    ble_write(0xC0);
    send_temp();
    send_frequence(&frequence);
    send_pas();
    ble_write(0xC1);
    //On appelle ble_do_events pour realiser l envoi du buffer contenant toutes les donnees
    ble_do_events();
}
}
return 0;
}
```
