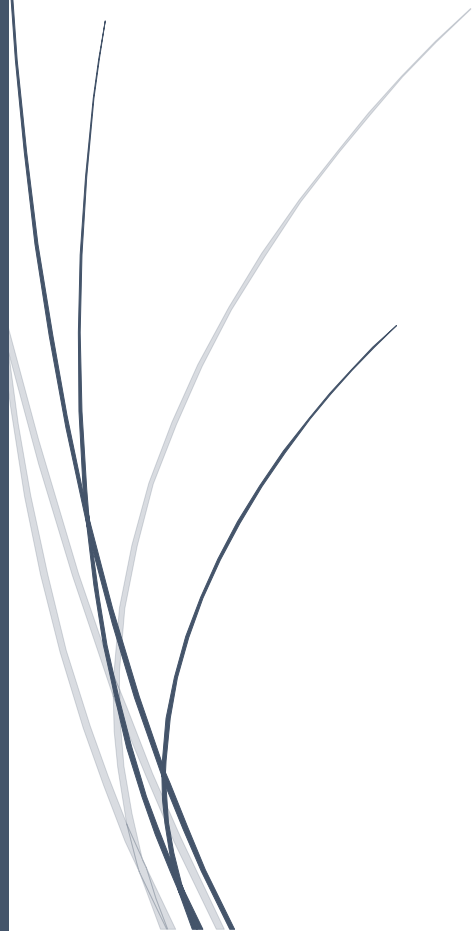
Université de Lille 1

École Polytechnique Universitaire de Lille
Département Informatique-Microélectronique-Automatique

11/05/2015

Surveillance passive du sommeil

Rapport de projet de quatrième année



Élèves :
Jérôme BAILET & Manouk SIMON

Encadrants :
Alexandre BOE & Thomas VANTROYS

Table de matières

Remerciements

Tout d’abord, nous tenons à remercier, toute l’équipe pédagogique de Polytech’Lille et les responsables de la formation Informatique-Microélectronique-Automatique de nous avoir enseigné les bases pour réaliser ce projet de quatrième année dans de bonnes conditions.

Nous souhaitons également remercier et témoigner toute notre reconnaissance à Monsieur Alexandre BOE, notre responsable de projet, pour son aide précieuse et sa disponibilité tout au long du projet, ainsi que Monsieur Thomas VANTROYS.

Enfin, nous remercions Monsieur Xavier REDON pour nous avoir fourni tout le matériel nécessaire et pour s’être rendu disponible pour répondre à toutes nos questions, de manière rapide et pertinente.

Introduction

Dans le cadre de notre 4ème année de formation d'ingénieur en Informatique-Microélectronique-Automatique, à Polytech Lille, nous avons eu à réaliser un projet qui se déroule le long d'un semestre et d'un quota minimal de quarante heures.

C'est l'occasion pour nous, étudiants, de mettre en application des connaissances acquises depuis le début de notre formation et de les perfectionner autour d'un sujet complet, de l'élaboration du cahier des charges à une éventuelle réalisation finale.

Notre projet a comme sujet "la surveillance passive de sommeil". L'objectif est de concevoir un capteur passif qui permettrait de suivre et d'analyser le sommeil de son utilisateur.

Nous équiperons ce capteur de plusieurs modules (carte SD, Bluetooth, application Android…) qui permettront d'exploiter du mieux possible les données de ce prototype, ainsi que de le compléter afin de faciliter son utilisation par l'opérateur.

Présentation du projet

Contexte du projet

Nous avons tous fait l'expérience d'une bonne hygiène de sommeil ou d'un manque de sommeil et des répercussions sur notre corps et notre psychisme. Nous ressentons les conséquences de notre sommeil dès le matin au levé.
Le rôle primordial du sommeil sur la santé de l'être humain est reconnu et prouvé scientifiquement. Le sommeil a de multiples effets sur le fonctionnement de notre organisme et favorise et régule certaines fonctions : défenses immunitaires, régulation hormonale,....
De nombreuses études ont démontré les perturbations physiologiques dues à un manque chronique de sommeil : fatigue physique, manque d'énergie, de vigilance et de concentration, perturbation de l'humeur ainsi que de la mémorisation et des apprentissages, … jusqu'à l'apparition de pathologies graves comme l'obésité ou l'hypertension…
Il est établi que nous consacrons un tiers de notre existence à dormir. Ce tiers est très important pour notre équilibre physique et psychique.

Or, depuis l'essor des nouvelles technologies, de nouvelles habitudes viennent perturber sérieusement notre sommeil.
Les téléphones portables, tablettes et télévisions se retrouvent jusque dans nos chambres, divertissant notre cerveau et excitant nos yeux, là où la lecture d'un livre conduisait par une fatigue naturelle des yeux à l'endormissement. De surcroît, le rythme de plus en plus infernal de notre vie quotidienne et professionnelle, augmente le stress voire l'anxiété des personnes dont le système nerveux est mis à mal. Tout ceci entraîne facilement des troubles du sommeil.

Selon plusieurs études, les Français dorment en moyenne moins de 7 heures par nuit, tandis que l'OMS recommande huit heures de sommeil quotidien. Ajouter à cela, 64% des personnes se sentent toujours fatiguées après leur nuit de sommeil.

Description du projet

Ce projet a pour but d'améliorer l'expérience et la qualité du sommeil, en permettant à l'utilisateur du Dreamer (nom du capteur passif) d'analyser son sommeil à travers le SleepWatcher (nom de l'application Android associée au capteur passif).

L'analyse de son sommeil par l'utilisateur lui permet de se responsabiliser et de se rendre actif pour tenter de réguler au mieux ses périodes de sommeil et de retrouver un cycle de vie plus équilibré.

Cahier des charges

L'objectif du projet est de concevoir un capteur passif Dreamer qui permettrait de suivre et d'analyser le sommeil de son utilisateur, avec plusieurs modes de fonctionnement:

- Un mode silencieux (sans transmission de données et donc pas d'émission radio) plus destiné aux personnes sensibles aux ondes électromagnétiques.
- Un mode alarme (pas de transmission sauf dans le cas d'une alarme programmée) afin de surveiller les enfants et d'alerter les parents en cas de problème. (à étudier en fonction des premiers résultats de surveillance obtenus)
- Un mode actif (transmission des données régulière vers un smartphone) pour traiter et analyser en temps réel le sommeil

Le contrôle de ce capteur et ses différents modes s'effectue à travers une application Android SleepWatcher. L'application permet également de stocker les données enregistrées et d'analyser sous forme de graphe le sommeil.

Étapes du projet

Après étude du sujet et du cahier des charges nous avons divisé le travail en plusieurs étapes. Nous avons privilégié d'abord le fonctionnement du système avant de se pencher sur l'étude du sommeil et l'interprétation des données du capteur.

Étape 1 :

- Schéma du système avec les différents modules à implémenter
- Liste des composants nécessaires pour le projet

Étape 2 :

- Programmation de l'Arduino : récupération des données du capteur sur la carte SD (mode silencieux)
- Réalisation des 1ers tests (sauts sur le lit, position levé/couché, allongé/assis, mouvements rotatifs, test du bruit, test sur toute la nuit).

Étape 3 :

- Développement de l'application Android pour le contrôle du capteur
- Programmation du mode de fonctionnement : alarme et actif
- Recherche d'un algorithme pour l'étude du sommeil à partir des données enregistrées par le capteur
- Suivi du sommeil dans l'application :
    - Synchronisation des données avec graphes indiquant la qualité du sommeil
    - Durée du sommeil
- Tests des trois modes en situation simulé et réel

Présentation de la réalisation

Cette première partie renseigne sur les différents éléments qui composent le capteur de surveillance de sommeil et présente les différentes fonctionnalités du Dreamer.
De plus, une brève notice de l'application Android SleepWatcher sera rédigée.

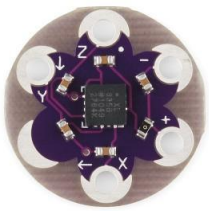
Description de la partie hardware : Dreamer

Le prototype fonctionnel

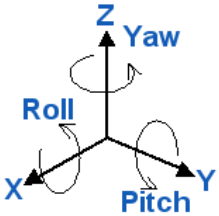
Nous avons réussi à mettre au point un prototype tout à fait fonctionnel. Ce prototype est composé de plusieurs éléments.

L'Arduino Mega, basée sur le microcontroleur ATmega1280 de chez Atmel, est le coeur du prototype. L'ensemble du frimware du Dreamer que nous avons développé, est flashé sur cet Arduino.

Le capteur de sommeil correspond à une shield accéléromètre 3 axes de chez LilyPad, utilisé comme un capteur de mouvements.



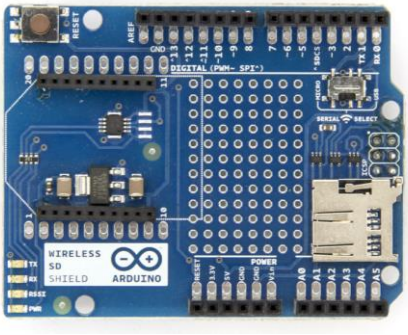
Cette shield est basé sur l'accéléromètre ADXL335 qui émet des signaux analogiques entre un 0V à 3V sur chacun des X, Y et Z axes. Pour avoir des mesures précises, nous conversons cette tension analogique en une unité de gravité. A partir des régles de trigonométrie, nous récupérons les angles pitch et roll :



La shield MOD-RTC de chez Olimex est interfacée avec l'Arduino via la communication Wire. Cette shield, utilisant une horloge temps réel PCF8563, permet de mémoriser l'heure en temps réel au moment où le capteur de sommeil a effectué un échantillon de mesures. Ainsi, l'exactitude des données est fidélisée au maximum.

La date et l'heure du RTC s'initialisent automatiquement à la date et à l'heure de la compilation du programme, lors de la fonction setup de l'Arduino.

L'ensemble des données (mesure de l'accéléromètre et heure du RTC) est stocké dans une carte SD de 2 Go. La carte SD est introduite dans une shield Wireless SD de chez Arduino.



Enfin, le dernier élément qui compose le prototype est une shield BLE, le MOD-nRF8001 d'Olimex. C'est l'élément central pour la communication avec l'utilisateur. Le prototype peut être commandé à partir de l'application Android SleepWatcher, installé sur le smartphone de l'utilisateur. A partir de ce dernier, l'utilisateur peut contrôler le Dreamer et transférer les fichiers présents sur la carte SD.

Il est à noter que la shield BLE et la shield Wireless SD sont interfacées avec l'Arduino à travers une communication I^2C.

En fin de projet, nous avons ajouté en complément deux boutons poussoirs et deux LEDs sur une breadboard.
Cet ajout permet à l'utilisateur de commander directement le prototype sans passer par la case de l'application smartphone.

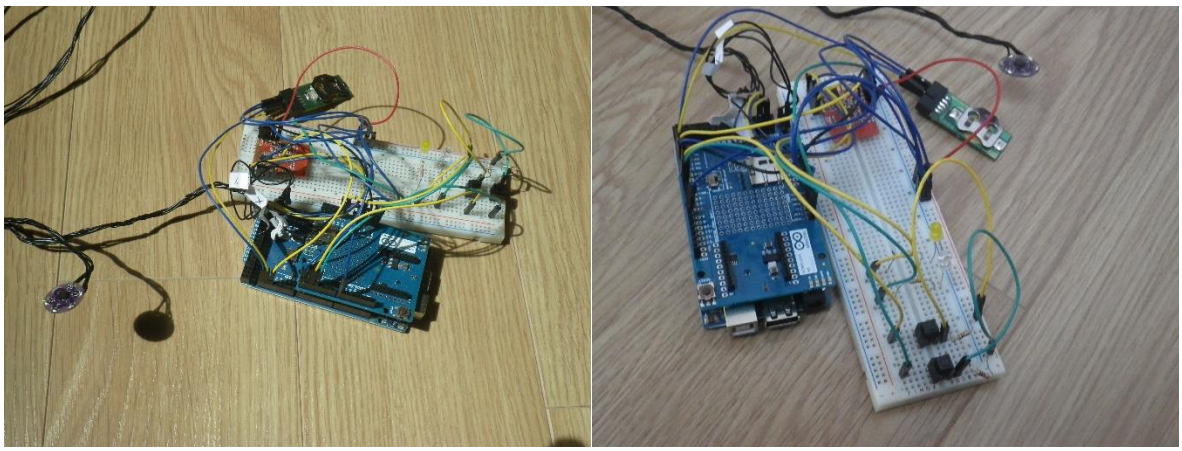
Chaque bouton possède la fonction ON/OFF du composant auquel il est associé : un bouton commande la mise en arrêt ou en marche du BLE, l'autre la mise en arrêt ou marche du Dreamer dans le mode silence.
Cela simplifie à la fois le circuit et évite d'augmenter la taille du prototype.

Les boutons sont directement connectés sur les pins d'interruption du BLE.

Chaque LED indique respectivement l'état de marche ou d'arrêt du BLE et du Dreamer (quelque soit le mode activé) : si la LED est allumée, le BLE ou Dreamer est allumé, sinon il est éteint.

Le prototype final

Une fois les modules associés et interfacés à la platine de développement l'Arduino Mega, nous nous retrouvons avec un prototype fonctionnel assez volumineux.



Initialement, nous avions souhaité ajouter de nouvelles fonctionnalités au capteur et à la documentation sur la surveillance de sommeil, afin d'améliorer le prototype.
Cependant, une bonne partie du projet a été consacrée à la recherche des capteurs de sommeil des entreprises concurrentes, comme l'Aura de chez Withings.

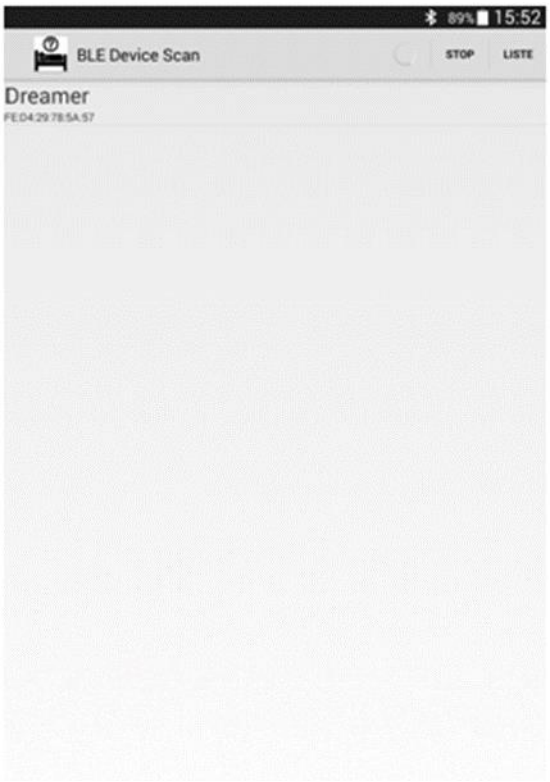
Nous n'avons donc pas eu le temps d'intégrer l'ensemble de ces fonctionnalités au sein d'une carte PCB de taille plus petite, qui aurait permis de réduire de manière significative l'encombrement de l'appareil et ainsi de le rendre le plus discret possible.

Afin de protéger le circuit électronique, il était prévu de recouvrir le prototype final sous 1 cm de mousse polyuréthane et de renfermer l'ensemble par un tissu étanche.
L'alimentation de Dreamer aurait été assurée par une prise secteur, avec une longueur de fil de 1,5 mètres.

Description de l'application Android : SleepWatcher

L'application Android SleepWatcher est complémentaire de la partie Arduino Dreamer. Il est possible à l'aide de cette application de contrôler le capteur et de récupérer les données pour obtenir les courbes de sommeils. La communication entre l'application et le capteur s'effectue à travers le protocole Bluetooth Low Energy.

L'application se divise en 4 activités chacune ayant un rôle spécifique :



A l'ouverture de l'application, l'application va tout d'abord demander à l'utilisateur d'allumer le Bluetooth. Nous avons développé SleepWatcher pour des versions Android 4.4 KitKat afin d'avoir une meilleure performance et stabilité lors de l'utilisation de Bluetooth Low Energy.

Une fois le Bluetooth activé, l'appareil va scanner les environs et détecter le capteur Dreamer (plus précisément le composant BLE nRF8001 qui broadcast).

Dans la barre de menu, il est possible d'arrêter le scan ou de le reprendre, à noter que le scan s'arrête automatiquement après 10 secondes pour ne pas consommer trop d'énergie. L'on peut également accéder à la liste des enregistrements dont nous parlerons plus loin.

Une fois le capteur détecté, il suffit de cliquer dessus pour accéder à l'interface de contrôle.

Nous accédons à l'interface de contrôle. Dans la barre de menu il est possible à tout moment de se connecter/déconnecter du capteur.

Trois checkbox permettent de choisir le mode de fonctionnement : le mode Actif, le mode Silence ou le mode Alarme. Une fois l'un des 3 modes choisi, le bouton Start permet de lancer une acquisition. Le bouton Stop permet d'arrêter l'acquisition en cours. Lorsque des données sont reçues, elles s'affichent dans l'espace Réception.

Le bouton Stats permet d'accéder à la liste des acquisitions sur l'appareil portable.

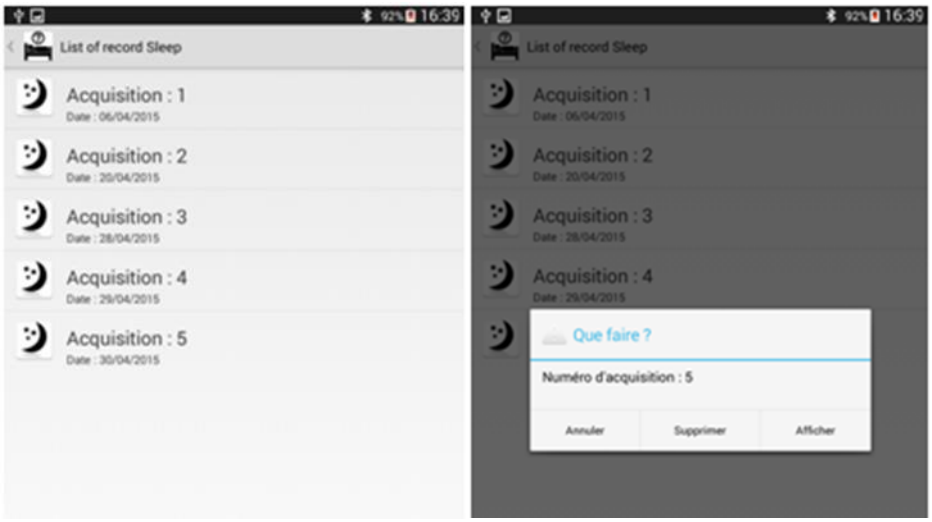
En appuyant sur Transfert, les données contenues dans la carte SD du Dreamer sont transférées au téléphone.

Spécificité des modes :

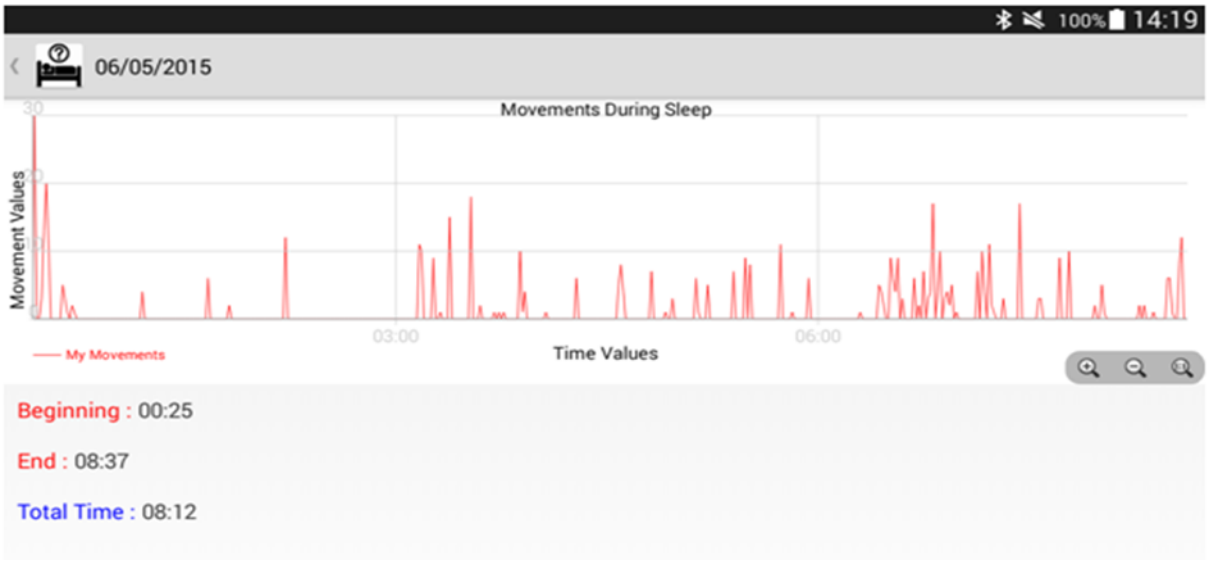
- En mode Actif les données acquises par le Dreamer sont transmises directement vers le téléphone et enregistrées sur ce dernier.
- En mode Silencieux il n'y a aucune transmission de données par Bluetooth. Ce mode est plus destiné aux personnes sensibles aux ondes électromagnétiques. Les données sont enregistrées sur la carte SD du Dreamer. Il n'y a donc aucune émission radio. Une fois l'acquisition terminée, le bouton Transfert permet de transférer les données de la carte SD du capteur au téléphone.
- Le mode Alarme, plus destiné à surveiller les enfants, est similaire au mode Actif. La différence est que lorsque l'enfant ne bouge plus pendant une certaine durée, une alarme se déclenche sur le téléphone pour avertir les parents.

A l'appui du bouton Stats dans l'interface de contrôle ou du menu Liste lors du scan, l'on accède à la liste des acquisitions sur le téléphone. Elles sont rangées par date, de la plus ancienne à la plus récente. Lorsque l'on clique sur un des éléments de la liste, l'application va demander à l'utilisateur de choisir entre :

- Afficher la courbe
- Supprimer les données pour cette date
- Ne rien faire donc retourner à la liste



La courbe de sommeil à une date donnée correspond dans l'application SleepWatcher aux nombres de mouvements par minute tout au long du sommeil. Sur l'axe des abscisses figurent le temps, et sur l'axe des ordonnées le nombre de mouvements. Il est possible de zoomer/dézoomer sur la courbe. Comme informations complémentaires : le début et la fin de l'acquisition, ainsi que la durée totale.



Présentation de la partie Software

Ce chapitre a pour but de donner un aperçu du code développé et d'expliquer les choix techniques pour réaliser notre capteur de sommeil.
Il se décompose en deux parties. D'une part, nous argumenterons la partie logicielle qui compose le Dreamer. D'autre part, nous présenterons le code développé qui a servi à mettre au point l'application Android, SleepWatcher.

Le firmware de l'Arduino

La conception du firmware de l'Arduino s'est déroulée, chronologiquement, en plusieurs étapes.
L'interfaçage d'une nouvelle shield avec l'Arduino et le développement du code associé à cette shield a conduit automatiquement à la création d'une librairie spécifique à la shield et à ses fonctionnalités.

Le firmware se traduit donc par un fractionnement du code en 3 fichiers, chacun regroupant les fonctions relatives à un domaine :

- La librairie du RTC (gestion du fonctionnement du "Real Time Clock") ;
- La librairie du BLE (gestion des évènements du "Bluetooth Low Energy") ;
- La librairie du Dreamer (gestion des différents modes).

Il est bon à savoir que nous avons choisi de développer le code sous le langage Arduino (C++) afin d'exploiter au maximum les nombreux programmes libres sur GitHub. Pour des fonctions élémentaires, nous avons programmé en C.

Sur GitLab, se trouve l'ensemble du code qui permet le bon fonctionnement du Dreamer. Chaque fonction de l'ensemble des librairies est commentée.

La communication Bluetooth

La communication Bluetooth entre le Dreamer et l'application SleepWatcher est un élément essentiel de notre projet.
L'utilisateur commande le Dreamer à travers cette communication : il choisit un des trois modes de fonctionnement (Actif, Alarme ou Silencieux) et commande la mise en fonctionnement et la mise en arrêt du Dreamer.
Mais aussi, le Bluetooth permet l'envoi des données du Dreamer vers l'application SleepWatcher afin que les données soient directement stockées et accessibles sur le smartphone de l'utilisateur, n'importe quand et n'importe où.

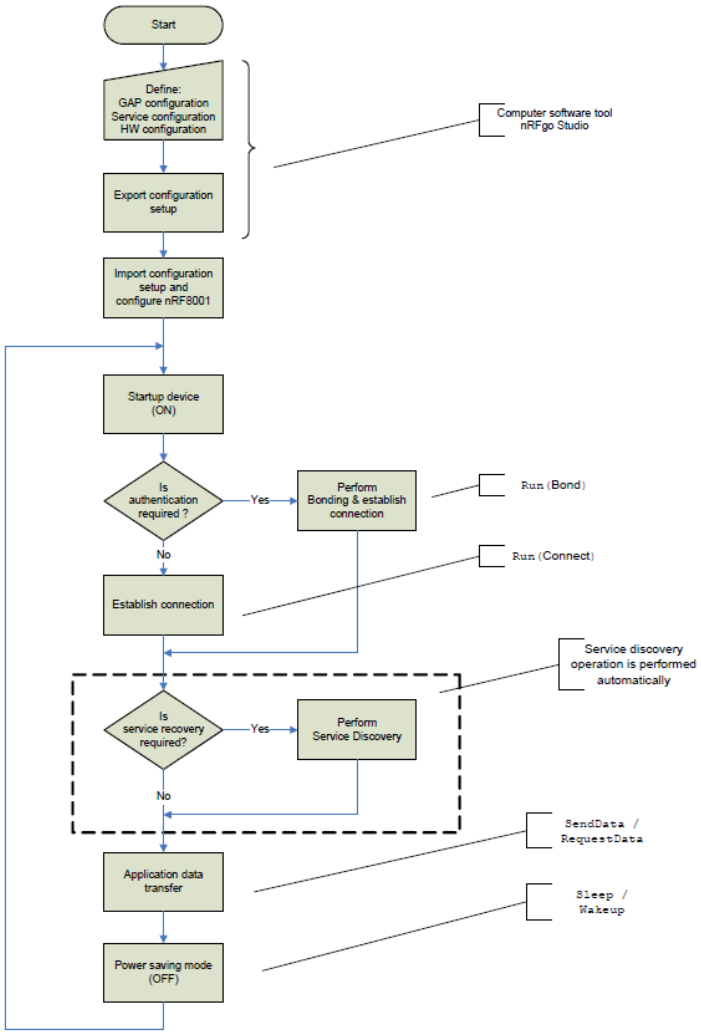
Nous avons récupéré un code libre et accessible sur Internet sur le service web d'hébergement et de gestion de développement de logiciels, GitHub.

Le BLE peut être mis sous différents états :

- Setup mode : état initial dès le démarrage du BLE ou après un RESET. Il permet la configuration des paramètres hardware et GAP ainsi que les services GATT.

- Sleep mode : le BLE est à l'état de sommeil. La consommation en courant est aux alentours de 6 microampères. Les paramètres de configuration du BLE et des données sont stockés en mémoire.
- Active mode : le BLE est à l'état actif. Il peut avoir trois différents niveaux d'activité :
    - Connected : appareillé à un smartphone et émission radio (pour transmettre des données entre les deux appareils) ;
    - Advertising : non-appareillé à un smartphone et émission radio (pour se connecter à un appareil) ;
    - Standby : pas d'émission radio.
- Test mode : teste le fonctionnement du BLE (nous ne l'utiliserons pas dans notre cas).

Nous avons joint ci-dessous deux schémas afin d'expliquer au mieux la procédure du BLE nRF8001et ses différentes configurations :



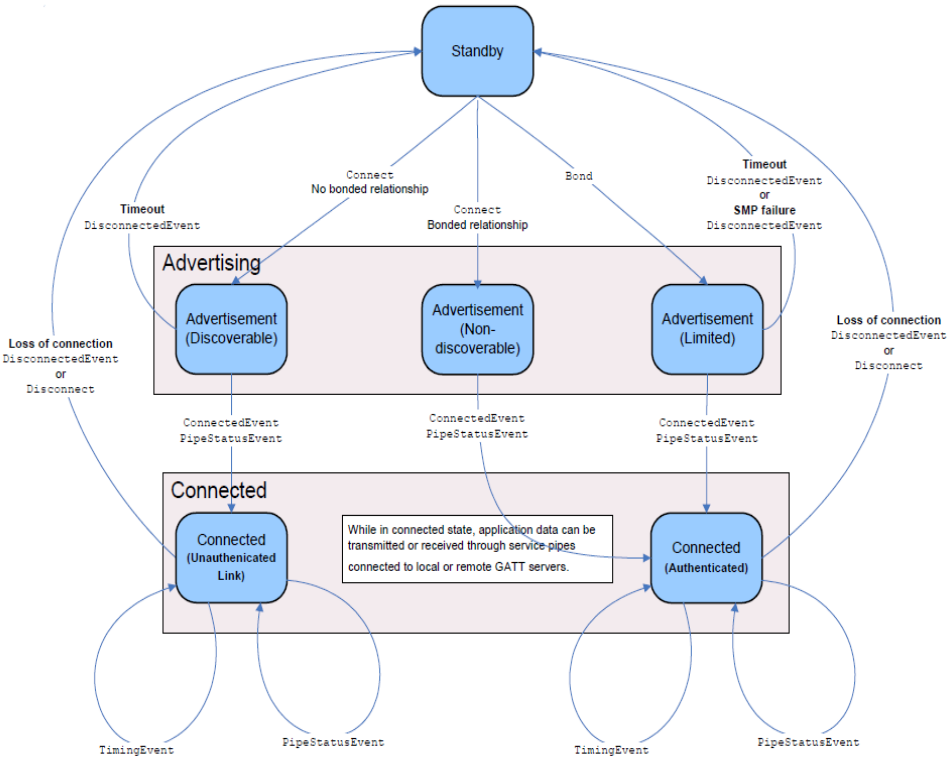
Pour communiquer avec le nRF8001, nous utilisons le protocole ACI (Application Controller Interface) qui utilise l'interface standard SPI pour échanger les données entre l'Arduino et le BLE.

Les échanges de données sur l'interface ACI sont divisés en deux types :

- Commands : échanges de données qui sont lancés par l'Arduino vers le nRF8001.
- Events : échanges de données qui sont lancés par le nRF8001 vers l'Arduino.

Par exemple, lors de l'envoi d'une mesure vers l'application, il va y avoir un "Commands" tandis que lors de la réception d'une demande de connexion par le smartphone ou lors de l'envoi d'une commande par le smartphone (start/stop), il y aura un "Events".

Voici, un aperçu des "Events" pour la connexion et déconnexion entre deux appareils :

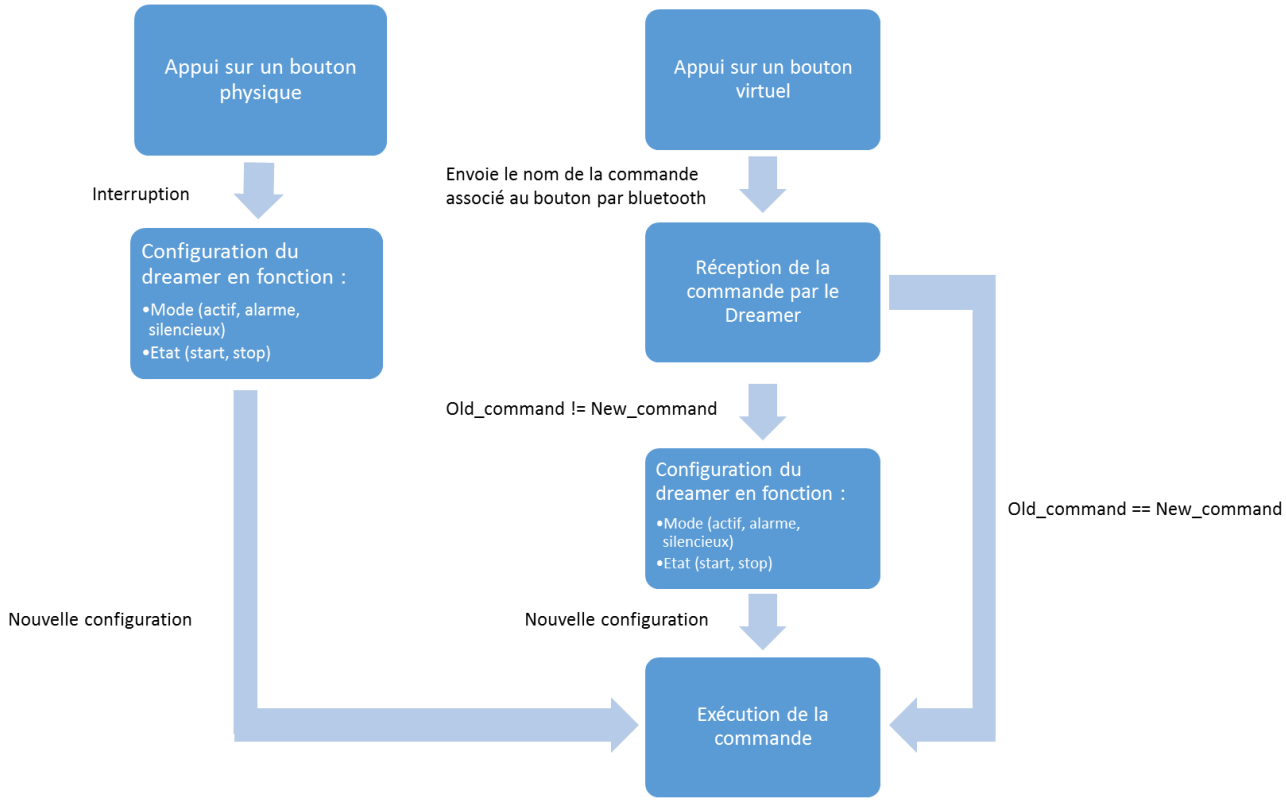


A partir de cette base de travail, nous avons modifié le programme, à l'aide de la datasheet du nRF8001, pour l'adapter à notre cahier des charges et le compléter selon nos besoins : envoi de la date, envoi de 3 mesures d'un échantillon, traitement de la commande.

Le traitement des commandes de l'utilisateur

L'utilisateur a la possibilité de commander le Dreamer, soit à travers l'application SleepWatcher soit à l'aide des boutons physiques sur le Dreamer. (Cette partie est réalisée entièrement par le programme main).

Nous pouvons schématiser le traitement de commandes de la manière suivante :



Via l'application SleepWatcher

L'appui sur un bouton virtuel entraîne l'envoi par liaison bluetooth d'une chaîne de caractères associé à la commande du bouton. Un "Event" se produit au sein du BLE dès réception de cette chaîne, qui reconfigure les paramètres du Dreamer.

Via les boutons physiques

L'appui sur un bouton entraîne une interruption qui modifie une variable booléenne. Lors de l'appel à la fonction de gestion des boutons, au début de chaque itération dans le programme main (dans une boucle infinie pour rappel), cette variable va modifier la configuration des paramètres du Dreamer, selon le bouton appuyé et la configuration actuele du Dreamer : arrêt/marche du BLE ou arrêt/marche du Dreamer.

Exécution de la commande

La fonction d'exécution de la commande se fait toujours par le programme main. Des conditions sont définies pour éviter que le Dreamer effectue plusieurs opérations en même temps (exemple : transfert de fichier et mode actif).

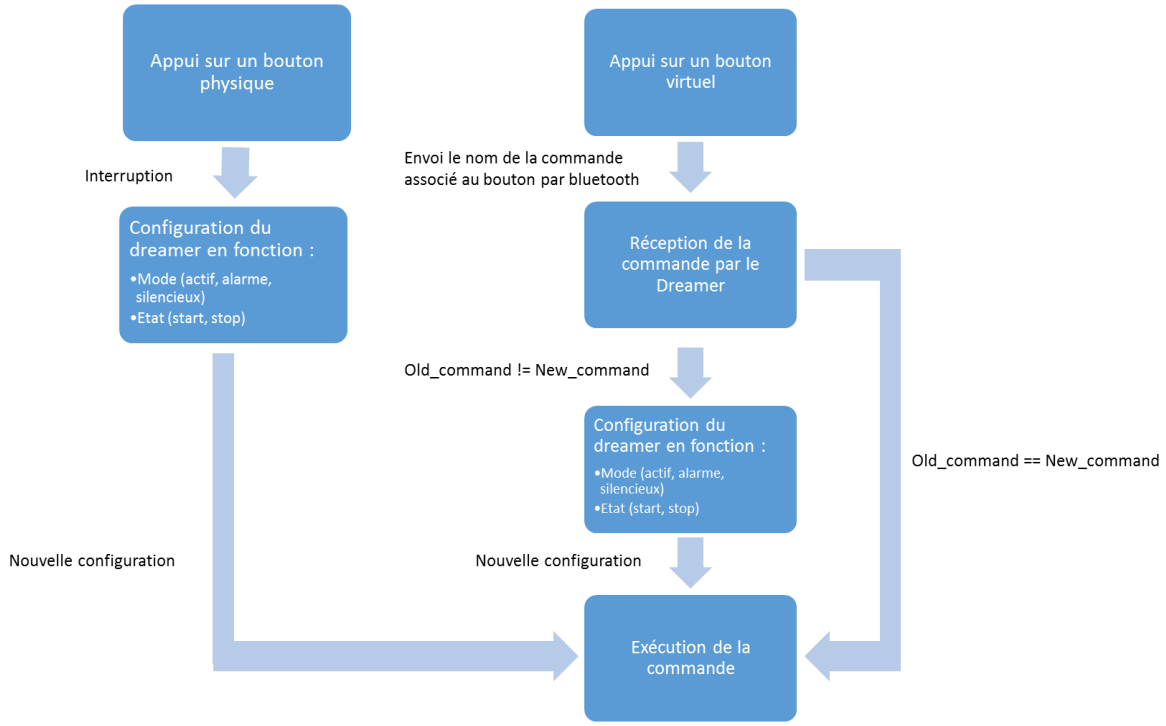
La gestion des différents modes

Les 3 modes du Dreamer fonctionnent seulement lorsqu'un des modes est paramétré via une commande par Bluetooth ou par bouton physique (le mode est actif par défaut) et que le Dreamer est lancé (start).
Une fois la commande stop reçue, le Dreamer n'effectue plus l'une des trois opérations.

Un schéma récapitulatif de la procédure spécifique à chacun des modes est donné.

Le mode Actif et le mode Alarme

Le mode actif et le mode alarme utilise en réalité le même protocole. Le mode alarme est géré dans l'application SleepWatcher.



Le module Bluetooth ne peut envoyer seulement que 20 octets de données. Or, nous devons envoyer 8 valeurs différentes sachant que :

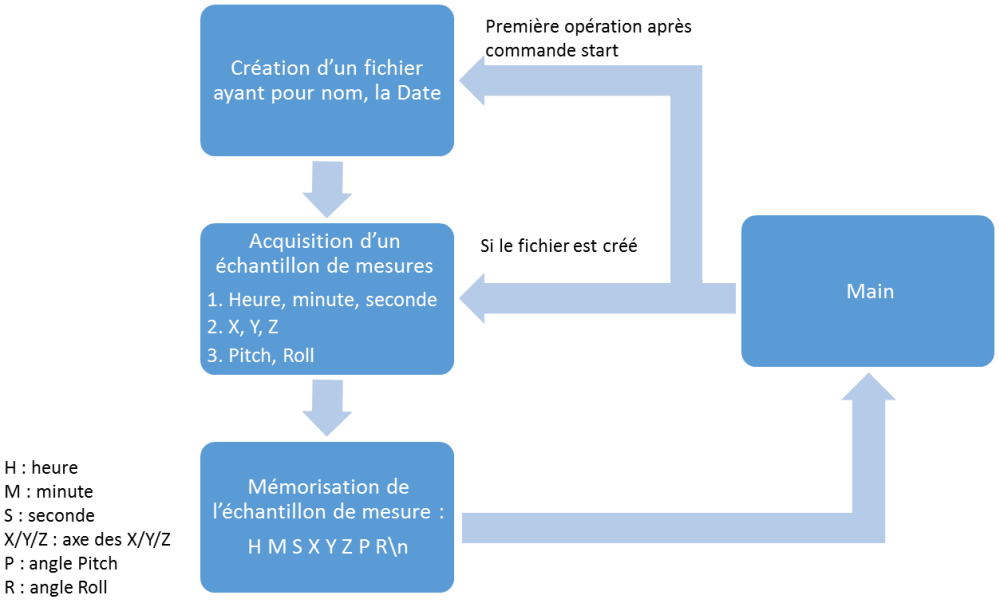
- Les valeurs de temps (heure, minute et seconde) occupent chacune 2 octets ;
- Les valeurs des axes (X, Y et Z), qui sont au départ des mesures flottantes inférieures à 1 et converties en entier avec une troncature au millième, peuvent avoir une taille de 5 octets en prenant en compte le signe positif et négatif ;
- Les valeurs des angles Pitch & Roll, qui ont subi la même conversion que les valeurs des axes, ont une taille plus élevée (jusqu'à 6 octets) du fait qu'elles peuvent varier à la dizaine.

Un maximum de 20 octets est donc facilement dépassé. Nous avons donc séparé un échantillon de mesure en trois et effectué 3 envois, l'un à la suite des autres.

Lors d'une commande d'envoi, le BLE va introduire les données à envoyer dans un buffer. Pour éviter des erreurs et un engorgement du buffer, l'appel à la fonction qui gère les évènements de la communication Bluetooth est nécessaire. Pour cela, on effectue une nouvelle itération de la boucle infinie dans le programme main.
Suite au bon fonctionnement de l'envoi de données, nous n'avons pas trouvé utile d'activer l'accusé de réception, afin de ne pas ralentir le temps de transmission.
Éventuellement sur une transmission longue distance, il sera peut être nécessaire de l'activer.

Le mode Silencieux

Le mode silencieux doit permettre à l'utilisateur de pouvoir stocker dans un fichier de la carte SD les mesures réalisées durant son sommeil, sans que le bluetooth du Dreamer ne soit activé.



Nous avons décidé de nommer le nom de chaque fichier créé par sa date de création. Cela a l'avantage de permettre au dormeur d'utiliser le Dreamer plusieurs fois dans la journée. Par exemple, dans la même journée, la mémorisation des données acquises lors de la deuxième utilisation du Dreamer va se faire à la suite de la dernière mémorisation des données acquises lors de la première utilisation.
Lors de la visualisation des mesures, l'utilisateur aura une vue de l'ensemble de la journée.

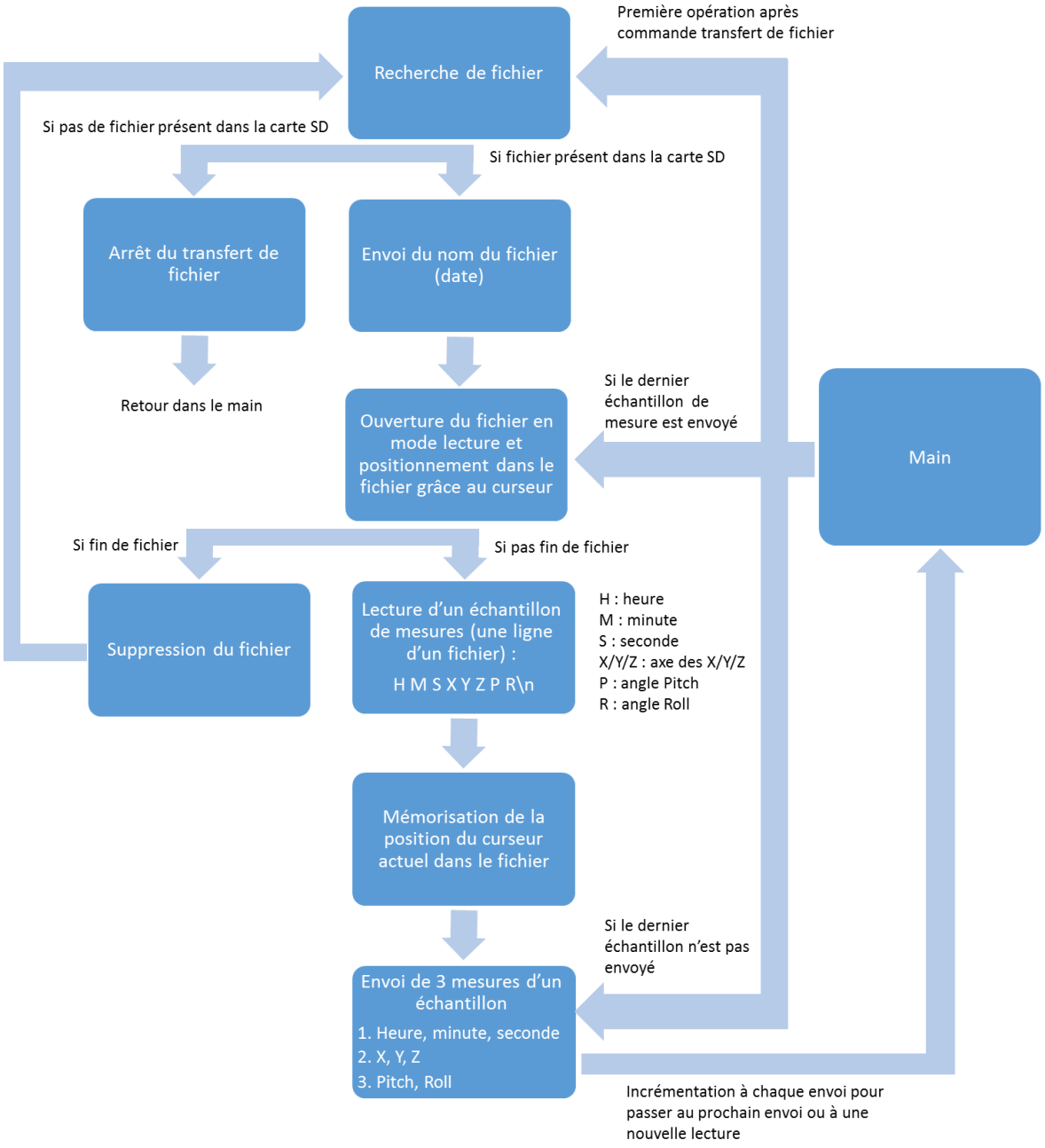
Le transfert de fichiers stockés sur la carte SD

Le mode silencieux nous a obligé à réfléchir sur un moyen pour l'utilisateur de récupérer les données sur la carte SD du Dreamer et de les visualiser sur son smartphone.

Une fonction transfert de fichier a été créée, qui s'effectue seulement si le Dreamer n'est pas activé sur l'un des différents modes, si le BLE est activé et si l'utilisateur l'a demandé à travers un bouton virtuel sur l'application SleepWatcher.

Il est bon de savoir que l'envoi des données stockées dans un fichier de la carte SD s'effectue exactement de la même manière que pour le mode actif et alarme.

Ici, nous avons eu a géré, de surcroît, la recherche de tous les fichiers stockés sur la carte SD et leur transfert.



Pour rappel, chaque ligne correspond à un échantillon de mesure et se décompose de la manière suivante : Heure Minute Seconde AxeX AxeY AxeZ Pitch Roll.
La difficulté a été de retenir le numéro de la ligne L qui permettrait de se positionner à la suivante lors de la prochaine lecture après l'envoi de la ligne L.
Lorsque le curseur se situe à la fin du fichier, celui-ci est automatiquement supprimé puis une nouvelle recherche de fichier dans la carte SD est effectuée.
Le programme s'arrête lorsque la carte SD ne contient plus de fichiers.

Une autre difficulté que nous avons dû gérer, est l'existence d'un fichier "SYSTEM~" dans la carte SD. En effet, le formatage de la carte SD au format FAT génère ce fichier "SYSTEM~".
La carte SD doit être sous le format FAT afin qu'elle puisse être utilisé par l'Arduino.

Le firmware de l'application Android

L'application SleepWatcher a été développée en Java sous le logiciel Android Studio. Son développement s'est déroulé en plusieurs étapes :

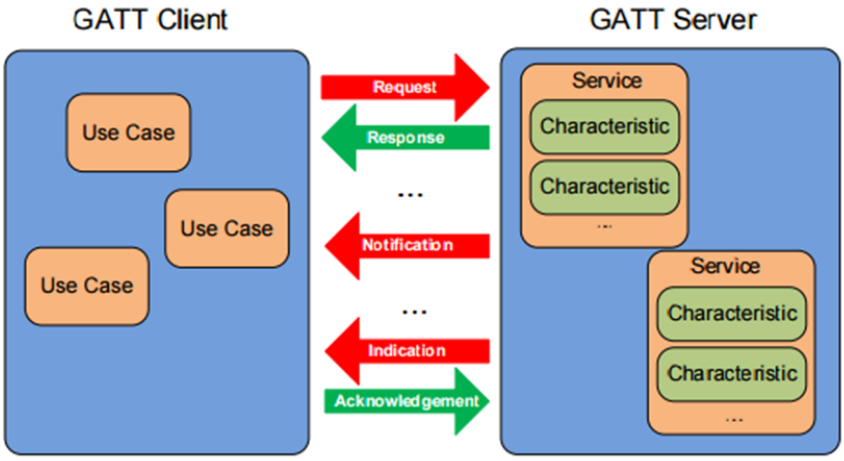
- Mise en place de la communication Bluetooth Low Energy entre le smartphone et le composant nrf8001 (module BLE du « Dreamer »)
- Développement de l'interface de contrôle, des différents modes et de l'enregistrement des données
- Affichage des courbes

La communication Bluetooth Low Energy

La mise en place de la communication entre le smartphone et le composant fut l'une des étapes les plus fastidieuses. Nous avons récupéré le code opensource disponible sur le site officiel *developer.android.com*.

A la différence du Bluetooth classique, le Bluetooth Low Energy est un protocole de communication rapide et de faible consommation énergétiques. L'application joue le rôle *Central* pour communiquer avec le composant *Peripheral* (le nrf8001 dans notre cas). Toute communication BLE est construite selon un profil *GATT* (acronyme de Generic *Attribute* Profile). Ce profil définit la manière dont les périphériques transmettent leur donnée aux appareils. L'application est le *GATT Client* tandis que l'Arduino joue le rôle de *GATT Server*. A noter que avec le profil GATT, les connections sont exclusives. C'est-à-dire que les périphériques BLE comme le nrf8001 ne peuvent être connecté qu'à un seul smartphone à la fois.

Les périphériques sont vus comme des *Services* et des *Characteristics*. Les services peuvent contenir une collection de Characteristics. Les characteristics sont des flags de propriété ou les valeurs de capteurs. Le nfr8001 possède 1 service propre et plusieurs Characteristics (RX, TX et Client Configuration). Chaque Service et Characteristic a son identifiant (UUID) propre. Le schéma ci-dessous reprend visuellement les explications.



Les échanges de données s'effectuent par Notification et non par Indication avec Acknowledgement, afin de ne pas ralentir la communication.

Concrètement dans l'application lorsque l'on choisit un mode ou lorsque l'on appuie sur Start, Stop ou Transfert ; une commande (chaine de caractère) est envoyée sur le Characteristic RX du GATT Server. La commande envoyée est reconnue par l'Arduino et est traitée. Lorsque le Dreamer envoi régulièrement des données, le smartphone va lire les données disponibles dans les Characteristics et les récupérer.

Les différents modes : Actif, Silencieux et Alarme

Afin d'éviter les bugs et conflits, lors de l'appui de Start il n'est pas possible d'effectuer un Transfert, et l'on a empêché la possibilité d'appuyer plusieurs fois le bouton Start à la suite.

Mode Silencieux :
Pour le mode Silencieux il suffit juste de cocher le mode Silencieux et de lancer Start. Le Dreamer va se mettre en veille et il faut se déconnecter de l'application pour ne pas perturber son fonctionnement.
Lorsque l'on veut récupérer les fichiers contenus sur la carte SD du capteur, l'on appuie sur le bouton Transfert.

Le *BroadcastReceiver* est l'objet qui par sa fonction onReceive va recevoir les events de l'application. Lorsqu'un event spécifiques, par exemple dans notre cas lorsque une donnée est disponible par Bluetooth, le code correspondant du BroadcastReceiver s'exécute. Dans le BroadcastReceiver l'on va recevoir tout d'abord le nom du fichier en question, et un fichier portant le même nom sera créé dans un emplacement spécifique du téléphone (/sdcard/SleepWatcherDataFiles).

Ensuite seront reçu successivement :
- Heure, Minute, Seconde
- X, Y, Z
- Pitch et Roll

Ces valeurs sont écrites dans le fichier sur le téléphone dans l'ordre exact, avec un espace entre chaque valeur et un saut à la ligne après écriture de Roll (voir Annexe 3).

Lors de l'envoi des valeurs de X, Y, Z, Pitch et Roll par Bluetooth, ces valeurs sont multipliées par 100 afin de les transformer en Int. Cette conversion était nécessaire dû à la taille limitée du buffer de 20 octets. Dans le BroadcastReceiver de l'application ces valeurs sont reconverties puis écris dans le fichier.

Mode Actif :

En mode Actif les données sont envoyées régulièrement du Dreamer au SleepWatcher. Le BroadcastReceiver va recevoir en premier la date du jour et crée un fichier depuis cette date. De même que le mode Silencieux, il y a reconversion des valeurs et les données sont rangées dans l'ordre définit.

Mode Alarme :
Le mode Alarme suit à peu de choses près la même procédure que le mode Actif (création de fichier à partir de la date du jour, stockage des valeurs dans un ordre précis avec reconversion de ces dernières).
Cependant une alarme retentit lorsque le dormeur ne bouge plus au bout d'un certain temps. Comment est-il possible de reconnaitre une absence de mouvement ? Ou plutôt comment peut-on reconnaitre un mouvement ? Cette partie est complémentaire de la recherche d'un algorithme du sommeil.
Avec les données de l'accéléromètre et de son bruit (environ 0.03g pour X, Y, Z et 1.5 degrés pour Pitch et Roll) nous avons approximé à partir de plusieurs tests qu'une variation de 0.05g (pour X, Y, Z) ou 3 degrés (pour Pitch, Roll) correspondaient à un mouvement. On peut ainsi compter le nombre de mouvements pour chaque minute.
Alors pour le mode Alarme, le BroadcastReceiver va comparer chaque valeur et après un certain temps, s'il n'y a plus de mouvements, l'alarme retentit.

Les courbes de sommeil

Afin de générer des courbes dans l'application Android, nous avons installé la librairie opensource *AChartEngine*. Cette API utilise le mécanisme d'Intent, des messages configurables qui permettent d'interagir avec différents composants Android d'une application. Elle ne nécessite aucune connexion internet.

Nous nous sommes alors poser la question de comment exploiter les valeurs contenues dans les fichiers enregistrées afin de générer une courbe explicite sur la qualité du sommeil.

Après analyse des articles de recherches sur le sommeil fournis par M.Boé nous avons déterminé un élément commun : le nombre de mouvements effectués par le corps par minute tout au long de la nuit.

Lors des premières semaines nous avons effectué plusieurs tests, et notamment sur le bruit de l'accéléromètre (voir Annexe 1 et 2). On observe une variation de 0.03g pour les valeurs d'accélération de X, Y et Z alors que l'accéléromètre est posé à plat, sans dormeur et sans aucune pression. Pour les angles Pitch et Roll on observe une variation de 1.5 degrés environ. Nous avons alors approximé à partir de plusieurs tests qu'une variation de 0.05g entre deux valeurs consécutives de X, Y, ou Z ou un écart de 3 degrés pour Pitch et Roll correspond à un mouvement du dormeur.

A partir de cette hypothèse il a été possible de créer la courbe du sommeil. A partir des valeurs dans un fichier texte (voir Annexe 3) il suffit de comparer successivement les valeurs correspondantes. S'il y a un écart de 0.05g entre 2 valeurs (soit entre deux lignes) de X, Y, ou Z ou bien 3 degrés de différence entre Pitch et Roll alors on ajoute un mouvement pour la minute correspondante. On effectue la même opération pour chaque minute. On a évité l'ajout excessif de mouvements : par exemple s'il y a déjà eu une incrémentation d'un mouvement à cause d'une variation de X pour 2 lignes successives ; alors on ne prend pas en compte les écarts entre Y, Z, Pitch et Roll pour ces 2 mêmes lignes. Une seule incrémentation entre deux lignes successives suffit.

On obtient alors une courbe avec en abscisse le temps et en ordonnée le nombre de mouvements. Cette courbe est assez explicite, on observe une alternance entre des pics de mouvements, phase de vide et regroupements de mouvements de valeurs moyennes à faibles.

On peut remarquer des pics de mouvements environ toutes les 90 minutes ce qui correspond à la durée d'un cycle de sommeil qui alterne micro-réveil (beaucoup de mouvements), phase de sommeil léger (mouvements réguliers), sommeil profond (peu de mouvements voir nul), sommeil paradoxal (peu de mouvements).

Bilan

L’étude sur le sommeil

L’étude sur le sommeil s’est résumée dans ce projet à la quantité de mouvements pour chaque minute. Nous avons réussi à obtenir des résultats exploitables malgré une simple approximation pour la détection de mouvements. Les courbes nous informent tout de même sur la qualité du sommeil. Un regroupement de pics de mouvements trop fréquents indique une nuit agitée, tandis que des pics bien espacés d’environ toutes les 90 min (soit un cycle de sommeil) correspondent à une meilleure qualité de sommeil.

Par manque de temps et de connaissances nous n’avons pas développé le code pour distinguer les différentes phases du sommeil : l’hypnogramme. Il aurait fallu approximer de nouveau les valeurs afin de déterminer ces différentes phases. Il aurait été peu rigoureux de se baser seulement sur un accéléromètre. La fréquence cardiaque, l’activité électrique du cerveau (avec émission des ondes thétas et delta) et le mouvement des globes oculaires sont plus révélateurs des différents stades du sommeil. Cependant utiliser ses caractéristiques aurait été à l’encontre du sujet qui est “la surveillance **passive** du sommeil”.

Les difficultés rencontrées

Bien que le prototype de surveillance passive de sommeil soit fonctionnel, quelques erreurs subsistent et pourraient être corrigées :

Du côté de l’Arduino

Des erreurs de lecture du temps ou de date, générées par le MOD-RTC, sont assez régulières malgré que le MOD-RTC soit bien initialisé.
Le remplacement du MOD-RTC par un autre module devrait résoudre le problème.

De plus, nous sommes partis du fait que le fonctionnement du Bluetooth serait stable une fois la connexion établie entre le Dreamer et le smartphone. Malheureusement, lors des simulations réelles (test sur une nuit complète de sommeil), le Dreamer peut se déconnecter du smartphone à un moment aléatoire.
Pour y remédier, nous avons eu l’idée que le Dreamer se reconnecte automatiquement au smartphone dès que celle-ci est perdue entre eux. Cependant, cela empêcherait l’utilisateur de déconnecter volontairement son smartphone du Dreamer

Du côté de l’application Android

Le code Android a été adapté au fur et à mesure des bugs rencontrés. L’application a été développée tout au long du projet à l’aide de tutoriel et code disponible sur le net, et n’a donc pas le mérite d’être aussi performante et stable qu’une application professionnelle. En effet la gestion du cycle de vie, la mise en arrière-plan, la gestion de mémoire, l’amélioration de la performance n’ont pas été assez approfondies. Pour chaque problème des solutions provisoires ont été apporté. Nous avons tout de même abouti à un prototype fonctionnel

pour le projet. Malgré cela des bugs persistent pouvant provoquer des crashs dont la liste ci-dessous n'est pas exhaustive.

On remarque pour le RTC que la valeur de l'heure rencontre des bugs entre chaque 40ième seconde à 59ième seconde. Il s'affiche une heure improbable (voir Annexe 3). Lorsque les premières valeurs du temps dans le fichier sont correctes il n'y a aucun problème pour l'affichage de la courbe. Cependant si la 1ère ligne du fichier débute par une valeur improbable c'est à dire que l'acquisition débute entre la 40ième seconde et la 59ième seconde, l'application va crasher pour afficher la courbe. Il faut dans ce cas corriger manuellement la ligne du fichier.

Dans l'interface de contrôle, une succession d'appui répétitif des boutons Start et Stop peut amener à créer des fichiers de date improbable. Le RTC n'arrive pas à gérer correctement plusieurs envois de la date du jour. On limite donc dans l'application une seule acquisition par jour. Si vraiment l'on veut effectuer une acquisition dans la même journée, il faut supprimer le fichier correspondant dans la liste des acquisitions.

Lors de l'enregistrement des données dans un fichier du téléphone, nous avons fait en sorte que le fichier se termine par une ligne complète. Le temps de latence entre l'envoi de la commande Stop et son traitement par le Dreamer n'étant pas négligeable, des données supplémentaires sont envoyées pour quelques secondes. Nous supprimons automatiquement la dernière ligne afin d'éviter des crashs lors de l'affichage de la courbe; et nous arrêtons l'écriture des données lors de l'appuie sur le bouton Stop.

Pour le mode Silencieux, lorsque le Dreamer tente de déconnecter son Bluetooth, l'application va automatiquement chercher à se reconnecter et donc impossible pour le capteur de passer en mode veille. Pour remédier à cela il faut quitter l'application. Cette astuce ne règle pas le problème mais permet tout de même d'exécuter la fonctionnalité.

L'affichage de la courbe d'une acquisition qui a duré toute la nuit, peut prendre quelques secondes et bloquer l'interface utilisateur UI. Lors d'une rotation de l'écran, l'application va recréer la courbe et donc bloquer à nouveau l'UI pour quelques secondes.

Enfin dernier problème qui peut mener à des crashs : si les données envoyés du Dreamer à SleepWatcher ne sont pas envoyés dans le bon ordre (HH MM SS X Y Z Pitch Roll), l'application plante. Cela peut être dû à une mauvaise lecture par le capteur ou une déconnection/reconnexion imprévue.

Les développements à apporter

Notre projet de surveillance passive de sommeil est un sujet d'actualité. Depuis peu de temps, le monde connaît un essor d'objets connectés en particulier dévolus au domaine de la santé.
A l'heure actuelle encore peu d'entreprises commercialisent des objets de surveillance de sommeil.
D'un point de vue purement technique et fonctionnel, le Dreamer peut déjà se démarquer du Withings Aura, grâce à son mode silencieux : la connexion Bluetooth peut être désactivée en même temps que le capteur de sommeil continue de collecter les données.

Par ailleurs, de nombreuses fonctionnalités peuvent encore être ajoutées :

- Ajout d'un module contenant un élément vibrant et sonore afin d'enrichir le prototype d'un réveil.
- Création d'un site web conçu comme base de données. Chaque utilisateur aurait un compte associé et retrouverait l'ensemble de ses données sur ce site web. Le transfert de données se réaliserait par Wifi à partir d'une shield Wifi intégrée au prototype.

Conclusion

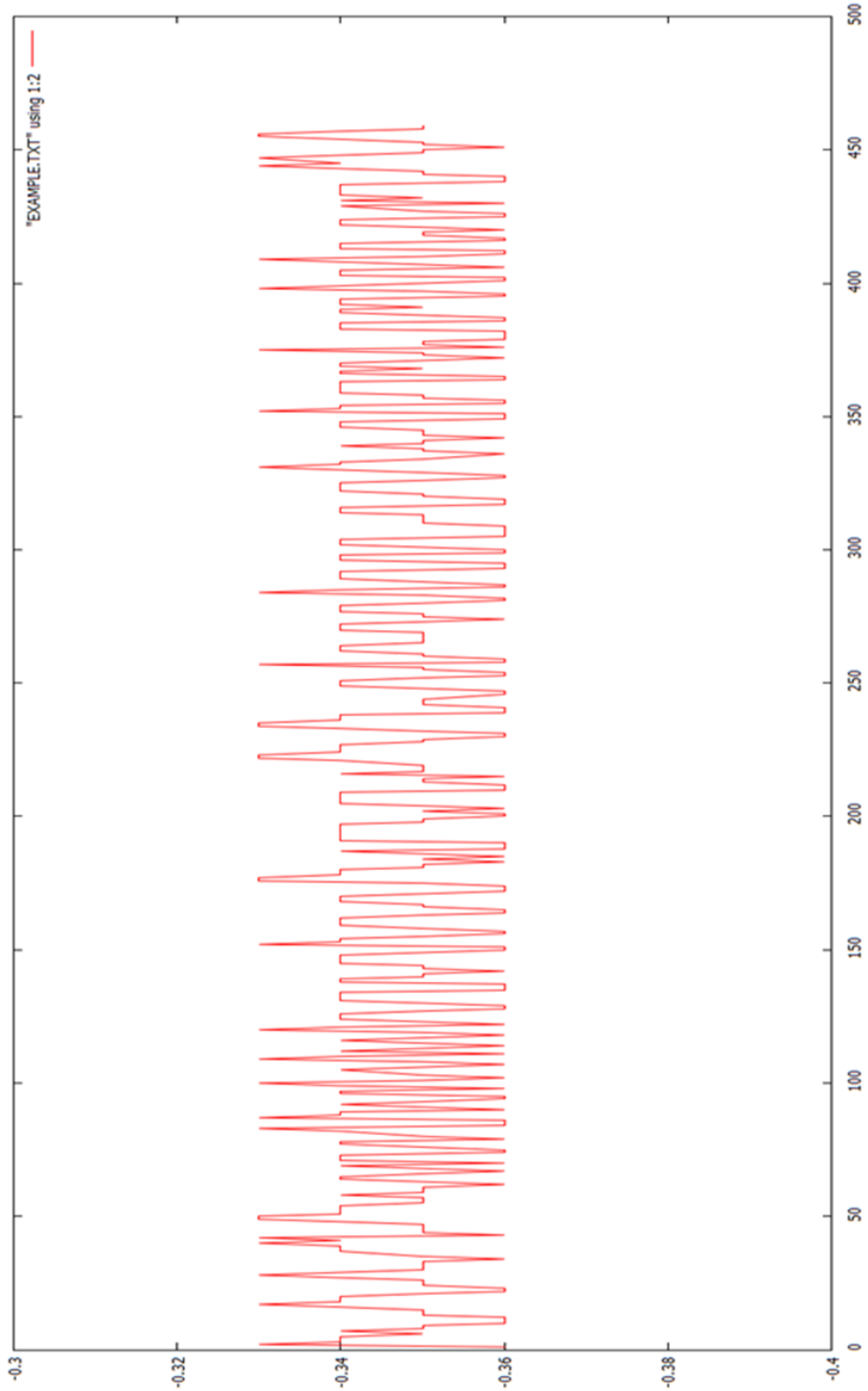
Nous avons effectué ce projet dans le cadre de notre quatrième année dans la spécialité Informatique Micro-électronique et Automatique à Polytech'Lille.

Notre projet consistait à la surveillance passive de sommeil. L'objectif était de concevoir un capteur passif qui permet de suivre et d'analyser le sommeil de son utilisateur. Nous avons répondu à la problématique du système dans sa globalité par l'aboutissement d'un prototype. Malheureusement nous n'avons pas pu intégrer les fonctionnalités au sein d'une carte PCB. Le développement de chacun des modules séparément a été rapide cependant le fonctionnement dans son ensemble et la communication entre chaque module a demandé un effort plus important et beaucoup de rigueur.

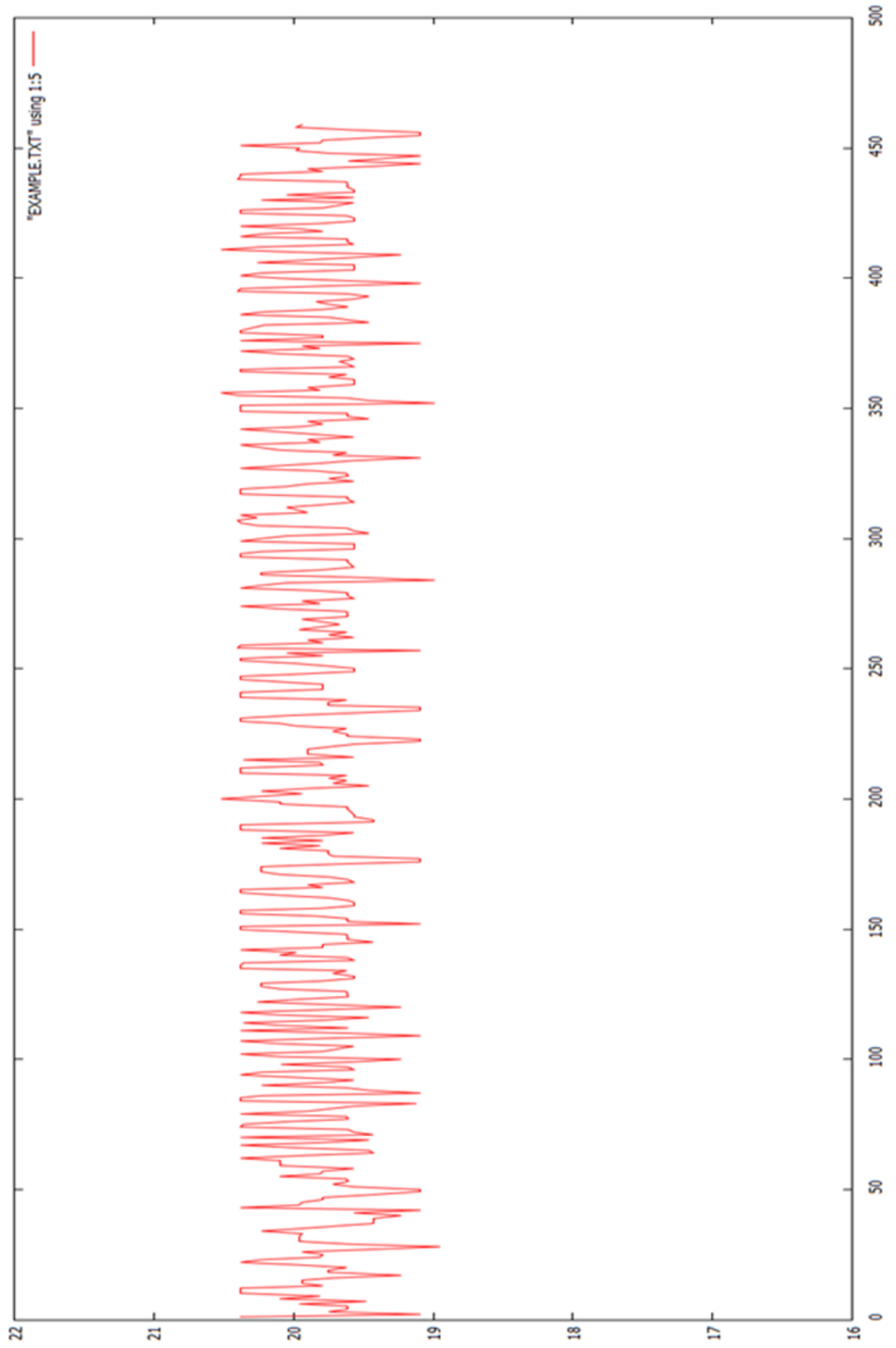
Au fur et à mesure de l'avancée du projet, nous avons pu mettre en pratique nos connaissances acquises durant la formation et d'en développer de nouvelles. Cette expérience nous a apportée énormément sur le plan des connaissances ainsi que sur la gestion de projet. Cette expérience nous a appris également le travail en équipe ainsi qu'à faire face à la difficulté.

Annexes

Annexe 1 : Courbe du bruit sur les valeurs d'accélération de X (accéléromètre ADXL335)



Annexe 2 : Courbe du bruit sur l'angle Pitch (accéléromètre ADXL335)



Annexe 3 : Exemple de fichier contenant les données d'une acquisition le 06/05/2015

- Colonne 1 à 3 : le temps selon le format heure, minute, seconde
- Colonne 4, 5, 6 : respectivement valeurs d'accélération de X, Y et Z
- Colonne 7 et 8 : respectivement valeurs des angles Pitch et Roll

File Edit Search View Tools Options Language Buffers Help

1 06052015.TXT

```
0 25 16 -0.31 -0.16 0.95 18.02 -9.04
0 25 17 -0.31 -0.17 0.95 17.99 -9.54
0 25 19 -0.31 -0.17 0.95 17.99 -9.54
0 25 20 -0.30 -0.16 0.95 17.53 -9.07
0 25 21 -0.31 -0.16 0.95 18.02 -9.04
0 25 23 -0.31 -0.16 0.95 18.02 -9.04
0 25 24 -0.31 -0.17 0.94 18.17 -9.63
0 25 25 -0.30 -0.16 0.95 17.53 -9.07
0 25 27 -0.30 -0.16 0.95 17.53 -9.07
0 25 28 -0.30 -0.16 0.95 17.53 -9.07
0 25 29 -0.31 -0.17 0.95 17.99 -9.54
0 25 31 -0.30 -0.17 0.95 17.51 -9.57
0 25 32 -0.30 -0.16 0.95 17.53 -9.07
0 25 33 -0.31 -0.16 0.96 17.85 -8.96
0 25 35 -0.31 -0.17 0.95 17.99 -9.54
0 25 36 -0.31 -0.17 0.95 17.99 -9.54
0 25 37 -0.31 -0.17 0.94 18.17 -9.63
0 25 39 -0.31 -0.17 0.95 17.99 -9.54
40 25 40 -0.31 -0.16 0.95 18.02 -9.04
40 25 41 -0.30 -0.16 0.95 17.10 -9.09
40 25 43 -0.31 -0.16 0.96 17.85 -8.96
40 25 44 -0.30 -0.16 0.96 17.37 -8.98
40 25 45 -0.31 -0.16 0.96 17.85 -8.96
40 25 47 -0.31 -0.17 0.95 17.99 -9.54
40 25 48 -0.31 -0.17 0.95 17.99 -9.54
40 25 49 -0.31 -0.17 0.95 17.99 -9.54
40 25 51 -0.30 -0.16 0.94 17.70 -9.15
40 25 52 -0.31 -0.16 0.95 18.02 -9.04
40 25 53 -0.31 -0.16 0.95 18.02 -9.04
40 25 55 -0.31 -0.17 0.95 17.99 -9.54
40 25 56 -0.30 -0.16 0.96 17.37 -8.98
40 25 57 -0.31 -0.15 0.95 18.04 -8.54
40 25 59 -0.32 -0.14 0.95 18.55 -8.01
0 26 0 -0.32 -0.15 0.95 18.53 -8.51
0 26 1 -0.32 -0.15 0.94 18.71 -8.59
0 26 3 -0.32 -0.16 0.95 18.50 -9.02
0 26 4 -0.32 -0.15 0.94 18.71 -8.59
0 26 5 -0.31 -0.15 0.95 18.04 -8.54
0 26 7 -0.32 -0.15 0.95 18.53 -8.51
0 26 8 -0.31 -0.14 0.95 18.07 -8.03
```