



Développement d'une plateforme de surveillance de l'état de santé d'une personne

Laboratoire Bordelais de Recherche
Informatique

Du 02/06/14 au 26/09/14

Simon ARMAND
Etudiant en deuxième année électronique
ENSEIRB-MATMECA

Tuteur : Francine Krief (Enseignant-Chercheur)
Maître de stage : Omessaad Hamdi (Enseignant-Chercheur)

Sommaire

Remerciements	3
I. Introduction.....	4
II. Le LaBRI	5
A. Présentation générale	5
B. Activités.....	5
C. La recherche	5
III. Le projet	7
A. Le contexte du projet	7
B. Le sujet du stage.....	8
C. Matériel mis à disposition	9
D. Cahier des charges	10
E. Planification du projet.....	11
IV. Réalisation du projet	13
A. Etat de l’art.....	13
a. Dans la recherche	13
b. Produits commercialisés	13
B. Matériel retenu, corrélations envisagées	14
a. Corrélations envisagées et capteurs retenus	14
b. Matériel supplémentaire	15
c. Bilan	16
C. Conception générale	16
a. Partie Arduino.....	17
b. Application mobile	19
D. Conception détaillée	22
a. Partie Arduino.....	22
b. Application	27
E. Tests pratiques	32
a. Test du système	32
b. Consommation.....	32
F. Validation du système	34
V. Conclusion	35
Annexe 1 : Description des capteurs.....	36
Annexe 2 : Références.....	42

Annexe 3 : Diagramme de classe de l'application.....	43
Annexe 4: Fonctions importantes de la classe e-Health	44
Annexe 5 : Fonctions importantes de la classe Monitor.....	46

Remerciements

Je tenais tout d'abord à remercier toute l'équipe du LaBRI pour m'avoir si bien accueilli et sans qui ce stage n'aurait pas pu être possible.

I. Introduction

Lors du stage de deuxième année, j'attendais de découvrir le monde de l'entreprise ou de la recherche afin de réellement savoir ce qui me correspond le mieux. Je portais également beaucoup d'importances à ce que l'on me donne un vrai problème à résoudre, en autonomie et non un projet scolaire sans grand intérêt. Par ailleurs, le domaine de la bioélectronique m'intéresse au plus haut point. En effet, je voulais absolument trouver un stage dans ce domaine afin de consolider mon intention d'y travailler. J'ai toujours aimé la biologie et la médecine. Par conséquent, trouver un stage ou un métier qui associe ce que je fais, c'est-à-dire l'électronique et la biologie était pour moi l'idéal. De plus, je pense que la bioélectronique et la télémédecine sont des domaines porteurs qui ne vont pas cesser de se développer. Le sujet du stage s'inscrit complètement dans ce domaine puisqu'il mélange biologie, électronique et informatique. C'est pourquoi il m'a de suite intéressé.

Aujourd'hui, grâce aux capteurs d'activités (capteurs accrochés à la ceinture, les montres, etc...) et aux capteurs médicaux (fréquence cardiaque, saturation en oxygène, etc...) et les capteurs environnementaux (capteurs d'humidité, de température, de pression...), il est par exemple possible de mesurer l'activité quotidienne d'une personne, pour lutter contre l'obésité, ou prévenir certaines maladies comme les maladies cardiovasculaires. L'objectif du stage était de mettre en place une plateforme de surveillance de l'état de santé d'une personne basé sur le principe précédent. A partir de la donnée issue de capteurs qu'il fallait choisir préalablement, il s'agissait de surveiller certaines grandeurs afin de mettre en évidence des corrélations avec un problème physique. En cas de problème, il fallait également prévenir la personne en émettant une alerte ou un conseil. Trois grandes questions se posent alors : quelles corrélations rechercher et avec quels capteurs? Comment détecter ces corrélations et avec quel matériel ? Comment prévenir l'utilisateur qu'un problème est survenu ?

Nous allons appréhender ce stage à travers trois aspects. Tout d'abord, nous allons nous intéresser au Laboratoire Bordelais de Recherche Informatique (LaBRI), laboratoire dans lequel le stage a été effectué. Nous le présenterons généralement avant de parler des différentes activités menées en son sein et de l'équipe dans laquelle je me trouvais. Dans un second temps, nous nous intéresserons au sujet du stage afin de mettre en évidence les enjeux d'un tel sujet, de décortiquer en profondeur le sujet du stage pour arriver à dresser un cahier des charges réaliste, en accord avec la problématique du sujet. Enfin, dans un troisième temps, nous nous intéresserons à la conception technique du système.

II. Le LaBRI

A. Présentation générale

Le LaBRI, **L**aboratoire **B**ordelais de **R**echerche **I**nformatique est une unité de recherche associée au CNRS (UMR 5800), à l'Université de Bordeaux et également à l'INP Bordeaux. Il s'agit donc d'un laboratoire public. Depuis 2002, il est également partenaire de l'INRIA.

Par ailleurs, le LaBRI est un laboratoire dynamique qui ne cesse de se développer. En effet, les activités menées au sein du LaBRI sont à la pointe des technologies du numérique. En conséquence, ses effectifs se sont accrus de façon importante ces dernières années. Ainsi, en mars 2014, il réunit près de 320 personnes, dont 112 enseignants chercheurs (Université de Bordeaux, INP Bordeaux), 37 chercheurs (CNRS, INRIA), 23 personnels administratifs et techniques (Université de Bordeaux, INP Bordeaux, CNRS, INRIA) et plus de 140 doctorants, post-doctorants et ingénieurs contractuels. Ce développement est en partie dû au soutien du Conseil Régional d'Aquitaine qui a participé à l'extension du bâtiment et à l'augmentation des équipements et des bourses de thèse et post-doctorants.

B. Activités

Les missions du LaBRI s'articulent autour de trois axes principaux :

- La recherche (théorique et/ou appliquée), dont les principaux domaines seront explicités plus tard
- La valorisation et le transfert de technologie,
- La formation, avec notamment bon nombre d'enseignants chercheurs enseignant à l'université ou à l'INP Bordeaux.

C. La recherche

Le laboratoire s'articule autour de six équipes thématiques alliant recherche fondamentale, recherche appliquée et transfert technologique :

- *Combinatoire et algorithmique*
- *Image et son*
- *Méthodes formelles*
- *Modèles et algorithmes pour la bio-informatique et la visualisation d'informations*
- *Supports et algorithmes pour les applications numériques hautes performances.*
- *Programmation, réseaux et systèmes*

L'équipe dans laquelle je me trouvais était composée de quatre personnes et travaillait sur ce dernier thème. En particulier, leurs travaux s'articulaient autour de qualité de service et la sécurité dans les réseaux de nouvelles générations. Plus précisément, l'axe de recherche de l'équipe tournait autour de trois aspects : les architectures de communication, les protocoles de signalisations et les algorithmes de routage. Le sujet de stage a été déterminé dans une

volonté de travailler sur les capteurs médicaux et constitue donc le premier projet dans ce domaine.

III. Le projet

A. Le contexte du projet

Tout d'abord, ce sujet de stage s'inscrit dans une période de fort développement des applications de « e-santé ». En effet, le nombre d'applications destinées aux particuliers qui permettent de mesurer l'activité physique (exemple des applications de fitness, de monitoring...) ne cesse d'augmenter. De plus, smartphones, tablettes et autres accessoires électroniques sont devenus incontournables et à la portée de tout le monde. Désormais, pour beaucoup, il paraît inconcevable de vivre sans smartphone et pour la majorité, il se trouve sans cesse à proximité ou dans l'une de nos poches. Il est donc légitime de penser à développer une application qui œuvre à notre bien-être et à notre santé physique, qui puisse collecter en continu des données, les analyser et nous renvoyer un compte-rendu, une alerte, un conseil à tout moment de la journée puisqu'il est désormais possible d'être sans cesse connecté à son smartphone.

De plus, le développement de la « e-santé » est favorisé par des conditions dans le domaine de la santé qui se dégradent : pénurie de personnel, évitement de l'hospitalisation, préférence pour les opérations en ambulatoire. Les applications de « e-santé » répondent donc à ces nouveaux besoins en terme de santé et permettent donc de limiter les coûts puisqu'il devient possible de surveiller, voir soigner une personne à son domicile, sans qu'il soit nécessaire de venir à l'hôpital.

Par ailleurs, des facteurs sociaux-démographiques comme le vieillissement de la population, l'augmentation des naissances, le développement de l'obésité et des maladies cardiovasculaires favorisent le développement de systèmes permettant de surveiller, aider, accompagner ce genre de personnes. En effet, que ce soit pour les personnes âgées, les nourrissons ou les personnes sujettes à des problèmes cardiovasculaires, un problème peut survenir à tout moment. On peut donc imaginer qu'une application les surveillant à tout moment et alertant leur entourage ou du personnel médical pourrait leur être bénéfique. Mais de telles applications pourraient également intéresser des personnes n'ayant pas particulièrement de problèmes de santé mais étant soucieux de leur bien-être et qui veulent avoir un aperçu de l'évolution de leurs constantes (exemple : prise de tension une fois par jour afin d'avoir un suivi rigoureux de cette dernière).

Enfin, la démocratisation des capteurs d'activité, des capteurs médicaux et environnementaux ainsi que la miniaturisation de ces capteurs permettent d'envisager de créer de tels systèmes, s'adressant à des particuliers. En effet, ces capteurs sont facilement trouvables et peu coûteux. La conception des systèmes est donc relativement peu coûteuse en matériel.

B. Le sujet du stage

1. Description du sujet

L'objectif du stage est de créer un système capable de surveiller en continu, à partir de données issues de capteurs médicaux, l'état de santé d'une personne. Pour cela, j'ai eu accès à différents capteurs médicaux qui seront détaillés dans une prochaine partie comme par exemple une sonde oxymétrique, un capteur d'airflow qui permet de mesurer l'activité respiratoire de la personne, un capteur de position ou d'activité galvanique (mesure du taux de sueur). A partir de ces différents capteurs, reliés en filaire à une plateforme prévue à cet effet, il s'agissait de récolter les données de ces capteurs, les analyser, chercher des corrélations entre ces mesures et une possible anomalie et avertir l'utilisateur en cas de problème. Les corrélations à chercher n'étant pas définie, il a fallu également chercher des liens entre la mesure d'une grandeur physique (exemple le pouls) et un problème de santé. Par ailleurs, la collecte des informations issues des capteurs a été effectuée sur une carte Arduino Uno, qui permettait d'intégrer la plate-forme de capteurs aisément et de faire des calculs sur les données recueillies assez facilement. De plus, le deuxième objectif consiste à prévenir l'utilisateur en cas d'anomalie détectée, immédiatement en temps réel grâce à son application smartphone qu'il a fallu également créer. Dans ce cas, un conseil lui sera également prodigué. Enfin, un travail préalable de recherche de l'état de l'art a été dressé afin de savoir la valeur ajoutée d'un tel projet, quelles corrélations ou quelles architectures avaient déjà été traitées ou adoptées.

Pour résumer, le sujet de stage se composait donc de cinq étapes fondamentales :

- Rechercher des produits et des architectures déjà existantes, c'est l'état de l'art.
- Choisir un ensemble de capteurs cohérent, se complétant et permettant d'établir des corrélations entre les performances physiques d'une personne et certaines constantes vitales comme par exemple la fréquence cardiaque ou la saturation en oxygène.
- Concevoir un module de raisonnement capable de détecter une anomalie ou un risque pour la santé. Cette détection sera basée sur les informations collectées par les capteurs médicaux précédemment choisis.
- Créer une application mobile permettant d'alerter l'utilisateur en cas de problème. Elle servira également à visualiser les données en temps réel, sauvegarder les différentes alertes afin de dresser un historique des différents problèmes survenus.
- Tester le système et le valider.

Ces cinq points seront développés dans la partie IV du rapport.

2. Finalité du projet

Il est à noter que le système fonctionnel n'est pas une finalité en soit. Ce projet constitue d'abord une prise en main des capteurs et servira de base à d'autres projets. En effet, les capteurs médicaux n'ayant jamais été manipulés par l'équipe, il a fallu dans un premier temps connaître leur fonctionnement mais également savoir à quoi ils servaient. Par ailleurs, ce projet sera en particulier, réutilisé à des fins pédagogiques. Le système sera donc amélioré, complété par des étudiants qui travailleront par exemple sur des questions de sécurité : l'application sauvegarde des données personnelles sur l'utilisateur, comment garantir que ces données ne vont pas être piratées ? Comment éviter qu'une autre personne se connecte aux

capteurs et récupèrent les données de ces derniers ? De plus, la communication ainsi que les questions de green vont être étudiées : comment améliorer les communications afin de garantir une transmission sans perte des données ? Comment améliorer les communications sachant que ce sont elles qui consomment le plus en termes d'énergie ? De plus, les étudiants pourront également s'intéresser à l'aspect généricité du projet, c'est-à-dire à modifier le système pour le rendre plus modulaire. Cela permettrait de pouvoir changer de capteurs, de modifier une brique de l'architecture du système sans avoir à modifier les autres. Enfin, une multitude d'autres capteurs peuvent être ajoutés afin de surveiller d'autres paramètres et établir d'autres corrélations. Un projet plus ambitieux, cherchant à dépister de nouvelles corrélations peut être mis au point afin de se démarquer des produits existants déjà sur le marché.

C. Matériel mis à disposition

Afin de réaliser ce projet, j'avais à disposition une plateforme de capteurs, un capteur d'activités sous forme de montre ainsi qu'une tablette sur laquelle l'application mobile a été déployée. Une Arduino Uno a également été achetée par la suite afin de superposer la plateforme à celle-ci. En effet, la plateforme de capteurs « e-health sensor platform » est un shield Arduino, c'est-à-dire une extension de l'Arduino qui se superpose à cette dernière.



Figure 1: Plateforme de capteurs "e-Health sensor platform"

Neuf capteurs différents sont reliés à cette plateforme :

- **Une sonde oxymétrique** qui permet de mesurer la saturation en oxygène en %, c'est-à-dire la quantité d'oxygène dissout dans le sang. Elle permet également de mesurer le pouls en bpm.
- **Des électrodes à électrocardiogramme** qui permettent de mesurer l'activité électrique du cœur grâce à 3 électrodes placées au niveau du cœur : une au niveau du cœur (+), une de l'autre côté du cœur (-) et une sous le cœur (N).
- **Un capteur d'airflow** qui permet de mesurer l'activité respiratoire. Il retourne une valeur analogique entre 0 et 1023. La valeur est positive lors d'une expiration. Lors d'une inspiration, la valeur est égale à 0.
- **Un thermomètre** qui permet de mesurer la température du corps humain.

- **Un tensiomètre automatique** qui permet de mesurer la tension.
- **Un capteur de position** qui contrôle 5 positions : Debout/Assis, couché sur le dos, couché sur le ventre, couché latéralement sur la gauche ou la droite.
- **Un capteur d'activité galvanique** qui permet de mesurer la conductance électrique de la peau qui varie en fonction de son niveau d'humidité (en fait la sueur).
- **Un glucomètre** pour mesurer le taux de glucose dans le sang.
- **Des électrodes à électromyogramme** qui mesure l'activité électrique d'un muscle grâce à des électrodes placés sur celui-ci.

La description complète du matériel est donnée en Annexe 1. Le détail des différentes grandeurs physiques comme la saturation en oxygène ou le pouls ainsi que les conséquences d'une variation de ces grandeurs sera explicité dans la partie IV. De plus, les capteurs retenus seront également présentés dans cette dernière.

D. Cahier des charges

Le système doit répondre à plusieurs critères afin de remplir les objectifs du projet. Tout d'abord, on rappelle que l'utilisateur doit sans cesse être surveillé. Par ailleurs, le deuxième objectif est de détecter un problème lorsque les mesures recueillies ne sont pas « normales ». Le système doit donc être capable de trier les bonnes et mauvaises mesures, puis en fonction des mauvaises mesures, établir un diagnostic. C'est-à-dire qu'à chaque série de mauvaises mesures, une corrélation doit être établie avec un problème physique. Le troisième objectif est de déclencher une alerte lors de la détection d'un problème et de la rendre lisible, compréhensible par l'utilisateur. Quatrièmement, il faut que l'utilisateur puisse visualiser les données issues des capteurs ainsi que les alertes détectées. Il faut également qu'il soit alerté en cas de problème grâce à son application mobile.

D'où le cahier des charges suivant :

1. Le système doit acquérir en continu les valeurs issues des capteurs choisis.
2. Le système doit être capable de déceler les mauvaises mesures et établir une corrélation avec un problème physique.
3. Alerter l'utilisateur en cas de problème sur sa tablette.
4. Le système doit pouvoir envoyer les différentes données issues des capteurs ainsi que les alertes sur la tablette afin de les afficher grâce à l'application mobile. En conséquence, il faut que la tablette puisse recevoir les différentes données venant de l'Arduino. Un moyen de communication entre l'Arduino et la tablette doit donc être trouvé.
5. L'utilisateur doit être capable de voir ses différentes constantes et être alerté, le tout en temps réel.
6. Les contraintes de consommation doivent également être prises en compte. En effet, l'Arduino, alimentée par une batterie ou une pile est destinée à être embarquée sur l'utilisateur toute la journée. Un système consommateur d'énergie n'est donc pas envisageable.

E. Planification du projet

Afin de respecter le cahier des charges et répondre aux objectifs du projet, j'ai partagé mon travail en 4 grandes parties. La première, « état de l'art, recherche des corrélations possibles et choix des capteurs à retenir » a duré le premier mois. La deuxième phase portait sur l'étude des capteurs, l'acquisition des données, la mise en place des corrélations et l'envoi des données vers la tablette : la partie Arduino en générale. Cette phase contenait également les premiers tests sur l'Arduino afin de contrôler le travail effectué. Cette phase a duré 4 semaines. La troisième partie portait sur la création de l'application mobile et a constitué la partie la plus longue du stage puisqu'elle a duré quasiment 5 semaines. En effet, comme je n'avais jamais fait d'application auparavant, le développement de l'application m'a pris beaucoup de temps. Enfin, dans la dernière phase, l'interfaçage de la partie Arduino et de l'application a été faite. Les tests sur le système global et le réglage des derniers détails ont également été effectués. Cette partie a duré les trois dernières semaines.

Par ailleurs, si on rentre un peu plus dans les détails, je me suis fixé, chaque semaine un objectif à atteindre comme le montre le planning suivant. Cela permettait de ne pas prendre trop de retard, savoir le travail effectué et ce qu'il restait à faire. J'ai ainsi pu mieux m'organiser. Cependant, j'ai été obligé de mélanger la partie Arduino et application pour des questions de matériel. En effet, étant donné que du matériel a été commandé en début de stage, notamment un module Bluetooth Arduino, j'ai commencé par développer l'application puis j'ai arrêté le développement de l'application pour concevoir la partie Arduino à l'arrivée du module Bluetooth.

Echéance	Tâche	Commentaires
02/06 – 26/06	Etat de l'art + recherche des corrélations + choix des capteurs à retenir	Travail de documentation
26/06 – 11/07	Installation logiciels, plugins + recherche renseignements sur le développement d'applications Android	Logiciels, plugins nécessaires au déploiement d'applications Android
14/07 – 25/07	Développement de l'interface Bluetooth de l'application	Beaucoup de problèmes de matériels
28/07 -01/08	Acquisition des données + mise en place des corrélations au niveau de l'Arduino	Fin de la partie détection problèmes + diagnostic
04/08 – 08/08	Test sur l'Arduino	Tests de la partie précédente

11/08 – 15/08	Elaboration de l'interface Bluetooth de l'Arduino	
18/08 – 22/08	Test Interface Bluetooth Arduino/Tablette + Récupération des données	
25/09 – 12/09	Mise en place des alertes au niveau de l'application tablette	Développement de la plus grosse partie de l'application
15/09 – 26/09	Test du système complet, amélioration au niveau des performances (robustesse, consommation) et de l'application.	Finalisation du projet

Figure 2: Planning du stage

IV. Réalisation du projet

A. Etat de l'art

a. Dans la recherche

1. Architectures existantes

Dans la recherche, plusieurs papiers ont été publiés concernant des projets de monitoring à distance. Je ne vais pas tous les citer mais dresser un compte-rendu des projets se rapprochant le plus de celui-ci.

Tout d'abord, il existe des architectures basées sur la communication entre un WSN (Wireless Sensors Network) et des professionnels de santé. Dans [1], le WSN collecte des données sur le patient (pouls, oxymétrie, activités physiques) et les stockent sur un serveur personnel grâce au protocole ZIGBEE. Le serveur communique ensuite par Bluetooth avec un ordinateur qui envoie les données aux professionnels de santé via Internet. Dans un projet plus récent datant de 2012 [2], le WSN envoie les données récoltées sur un cloud. Les professionnels de santé ayant un droit d'accès peuvent alors consulter ces données et stocker à leur tour un diagnostic médical, un commentaire sur ce même cloud. Le papier propose également des solutions de sécurité d'accès au serveur qui garantissent l'intégrité des données envoyées stockées sur le cloud.

Dans [3], le système est composé de capteurs environnementaux (pression, température..), de mouvements, de présence, de sons ainsi qu'une webcam, reliés sans fil à un ordinateur. Ces capteurs vont donner des informations qui peuvent être utilisées pour localiser le patient dans la maison, détecter l'activité dans laquelle il est engagé afin de prévoir un moyen pour que l'utilisateur termine cette activité si à un moment donné il est distrait. Ce système est donc destiné aux personnes souffrant de liaisons cérébrales qui peuvent subir par moments des pertes de mémoire. Il permet d'aider ces personnes à faire des tâches quotidiennes dans un temps imparti. Par exemple, imaginons que l'utilisateur soit en train de cuisiner et qu'il reçoive un coup de téléphone. Le système va détecter le coup de téléphone et accepter une stratégie pour que l'utilisateur n'oublie pas d'éteindre le feu. Ce projet ressemble à celui-ci dans le fait qu'il y a une analyse des données issues des capteurs, un comportement à adopter et un retour à l'utilisateur en fonction de ce qui a été décidé.

Un autre système, présenté [4], propose un système de détection des apnées du sommeil en étudiant les signaux d'oxymétrie et d'électrocardiogramme. Ces derniers sont envoyés et analysés sur un smartphone afin de garantir un diagnostic temps réel. Les résultats sont ensuite envoyés par e-mail chez le praticien, médecin traitant le patient. Comme notre projet, les données sont donc traitées localement sur un smartphone. Cependant, le système est figé à deux sortes de capteurs (oxymètre et électrocardiogramme) et l'utilisateur n'est pas directement, en temps réel d'un nouveau problème car, bien que les données soient préalablement analysées, ce sont les praticiens qui établissent le diagnostic final.

b. Produits commercialisés

Outre les projets de recherche, certains produits sont déjà commercialisés. Que ce soit pour les hôpitaux ou le grand public, des systèmes de monitoring de patient, connectés existent.

Par exemple, pour les hôpitaux, une entreprise nommée Capsule [5] crée des solutions pour relier n'importe quel appareil biomédical à n'importe quel système d'informations (smartphone, tablette, ordinateurs...). Mais ces solutions s'adressent avant tout au milieu hospitalier et ne nous concerne donc pas vraiment puisque notre projet s'adresse aux particuliers. Pour ces derniers, il existe beaucoup de produits concernant la mesure de l'activité respiratoire qui permettent de détecter les apnées du sommeil, faire des tests respiratoires à domicile pour les personnes asthmatiques par exemple ou des tests d'efforts pour les personnes ayant un problème respiratoire. Encore une fois, l'avantage de ces produits réside dans le fait que tous ces tests précédents peuvent être pratiqués par le patient, à son domicile sans aller voir un spécialiste. Pour n'en citer que quelques-uns, une entreprise nommée MIR (Medical International Research [6]) commercialise des appareils dédiés à la spirométrie qui est la mesure des volumes d'air inspirés et expirés ainsi que les débits s'y rattachant). Ainsi, un des appareils permet de surveiller les désaturations, synonyme d'apnées en continu grâce à une sonde oxymétrique. L'utilisateur peut surveiller en temps réel ces constantes, ici sa saturation en oxygène sur un petit appareil de la taille d'un portable. Il peut également faire différents tests respiratoires et envoyer les résultats par e-mail à son médecin. Bien qu'ici les capteurs ne soient pas rattachés au smartphone, l'idée de surveillance en continu d'une constante et de détection d'une anomalie se retrouve. Une autre entreprise du nom de TWITOO [7] commercialise différents équipements médicaux comme un tensiomètre, un oxymètre, une balance, un thermomètre, un spiromètre et des capteurs ECG. Tous ces appareils sont synchronisés automatiquement par Bluetooth sur une plate-forme qui transmet à son tour sur un espace médical sécurisé les différentes mesures. Ces dernières sont donc accessibles à tout moment par tous ceux qui ont les droits d'accès. Les données ne sont donc pas consultables directement sur un smartphone comme ce présent projet mais via un serveur sécurisé.

B. Matériel retenu, corrélations envisagées

a. Corrélations envisagées et capteurs retenus

Le premier travail a été de mettre en évidence des corrélations entre la mesure de capteurs et un problème physique. Pour cela, j'ai cherché dans la littérature médicale des informations sur les différents capteurs afin de pouvoir isoler les capteurs les plus importants, ceux qui sont indispensables pour le diagnostic de certains problèmes de santé. Ainsi, avec la panoplie de capteurs présentés précédemment, il a été possible d'établir les corrélations suivantes entre :

- les apnées du sommeil et la mesure de la période respiratoire bien évidemment mais également avec la mesure de la saturation en oxygène ($SPO_2 < 94\%$). Il existe deux sortes d'apnée : l'apnée sans désaturation (sans baisse de la saturation en oxygène) et celle avec désaturation. Une apnée est déclarée à partir de 10 secondes sans respiration. Elle est déclarée pathologique à partir de 30 événements par nuit et anormal à partir de 5 par heure d'après [8].
- Un état de stress ou d'émotion forte et la mesure de l'activité galvanique (augmentation de la sueur), d'une hyperventilation (augmentation de la période respiratoire) et du pouls (augmentation du rythme cardiaque).

- Un évanouissement et la mesure de l'activité galvanique (augmentation de la transpiration), d'une hyperventilation, d'une bradycardie (diminution du rythme cardiaque) et de l'oxymétrie (diminution du SPO2).

Il reste maintenant à savoir de combien ces constantes physiques doivent varier afin d'établir la bonne corrélation. Ceci pose un gros problème. En effet, chaque être humain étant différent, ces constantes vont varier différemment, auront des valeurs au repos différentes selon les individus. Par exemple, un test dans l'équipe a montré que sur 4 personnes, au repos personne n'avait le même pouls, et la variance était très grande (valeurs allant de 50 à plus de 100 bpm autour d'une moyenne de 80). Certains avaient même un pouls qui était considéré comme anormal dans la littérature médical. De plus, certains capteurs comme le capteur d'activité galvanique est très difficile à tester et à démontrer son utilité dans les cas présentés précédemment. En effet, en ce qui concerne par exemple la détection d'un état de stress, des tests émotionnels sur plusieurs patients auraient pu montrer son importance et quantifier la variation de l'activité galvanique. Malheureusement, ces tests n'ont pas eu lieu. Ainsi, bien que la mesure de ce capteur soit prévue dans l'architecture, je ne l'ai pas implémenté car non vérifiable.

Par ailleurs, ce travail de documentation a tout de même permis de déterminer les « seuils » normaux de certaines constantes vitales comme le pouls, la saturation en oxygène ou la période respiratoire. Encore une fois, ces seuils varient en fonction de la personne, ce sont donc des valeurs moyennes. Ainsi, au repos, les différentes constantes doivent rester entre les valeurs suivantes :

- Pour le pouls : la fréquence cardiaque (FC) doit être comprise entre 50 et 100 bpm et ne doit pas dépasser la valeur 220-AgePersonne. Au-dessus de 100 bpm, l'individu se trouve en tachycardie alors qu'en dessous de 50, il se trouve en bradycardie d'après [9].
- Pour la respiration : au repos, un individu « normal » se situe aux alentours de 15 ou 20 respirations par minute. On rappelle qu'une respiration se compose d'une inspiration et d'une expiration. Au-dessus de cette valeur, la personne est en hyperventilation. En activité, le rythme respiratoire monte entre 40 et 50 respirations par minute. Ces valeurs sont encore une fois une moyenne d'après [10].
- Pour la saturation en oxygène (SPO2), la SPO2 doit rester entre 95 et 99%. En-dessous, c'est une désaturation en oxygène. En revanche, une SPO2 de 100 % peut correspondre à un empoisonnement au monoxyde de carbone.

b. Matériel supplémentaire

Du matériel supplémentaire était nécessaire. Tout d'abord une carte Arduino Uno, détaillée *Annexe 1* a dû être achetée afin de pouvoir connecter la plateforme et ainsi pouvoir récupérer les valeurs des capteurs. En effet, la plateforme étant un shield Arduino, il fallait obligatoirement une carte Arduino Uno pour l'utiliser. De plus, il a fallu trouver un moyen de communication entre la carte Arduino avec les capteurs et l'application mobile. Plusieurs choix se sont alors offerts. Tout d'abord, le WiFi aurait pu être utilisé comme moyen de communication. Cependant, cette technologie est consommatrice en énergie, notamment par rapport à la technologie Bluetooth. De plus, elle est plus souvent utilisée sur des distances plus longues. Ensuite, la technologie ZIGBEE semble être déjà plus appropriée par rapport au WiFi car beaucoup moins consommatrice et dédiée à de plus courtes distances (100m

maximum). Cependant, la plupart des smartphones et des tablettes n'ont pas cette technologie incorporée. On ne peut donc pas l'utiliser. Enfin, la technologie Bluetooth semble être la plus appropriée. En effet, au niveau de la consommation, elle se situe entre la technologie ZIGBEE et WiFi. De plus, la plupart des appareils connectés à un smartphone fonctionne avec le Bluetooth (casque audio, montre...). C'est donc une technologie parfaitement adaptée à la communication entre un périphérique (ici l'Arduino) et une tablette. Il a donc fallu acheter un module Bluetooth afin de le connecter à l'Arduino. Le détail du module HC-05 est donné en *Annexe 1*.

c. Bilan

Pour résumer, seulement trois capteurs ont tout d'abord été retenus : la sonde oxymétrique, le capteur d'airflow et le capteur d'activité galvanique qui, pour faute de test et de vérification de son fonctionnement n'a pas été utilisé dans le projet. Ces trois capteurs sont censés contrôler la saturation en oxygène, le pouls et l'activité respiratoire mais également déceler les apnées, avec ou sans désaturation, un état de stress ou de forte émotion ou un risque d'évanouissement. Par ailleurs, afin de pouvoir manipuler ces capteurs, une carte Arduino Uno a été achetée ainsi qu'un module Bluetooth HC-05 pour l'envoi des données sur la tablette. Nous allons maintenant nous intéresser à la conception du système, d'une façon générale, puis nous rentrerons dans les détails.

C. Conception générale

En restant d'un point de vue général, le schéma suivant récapitule l'architecture retenue. Les choix d'implémentation seront expliqués plus en détail dans les prochaines parties.

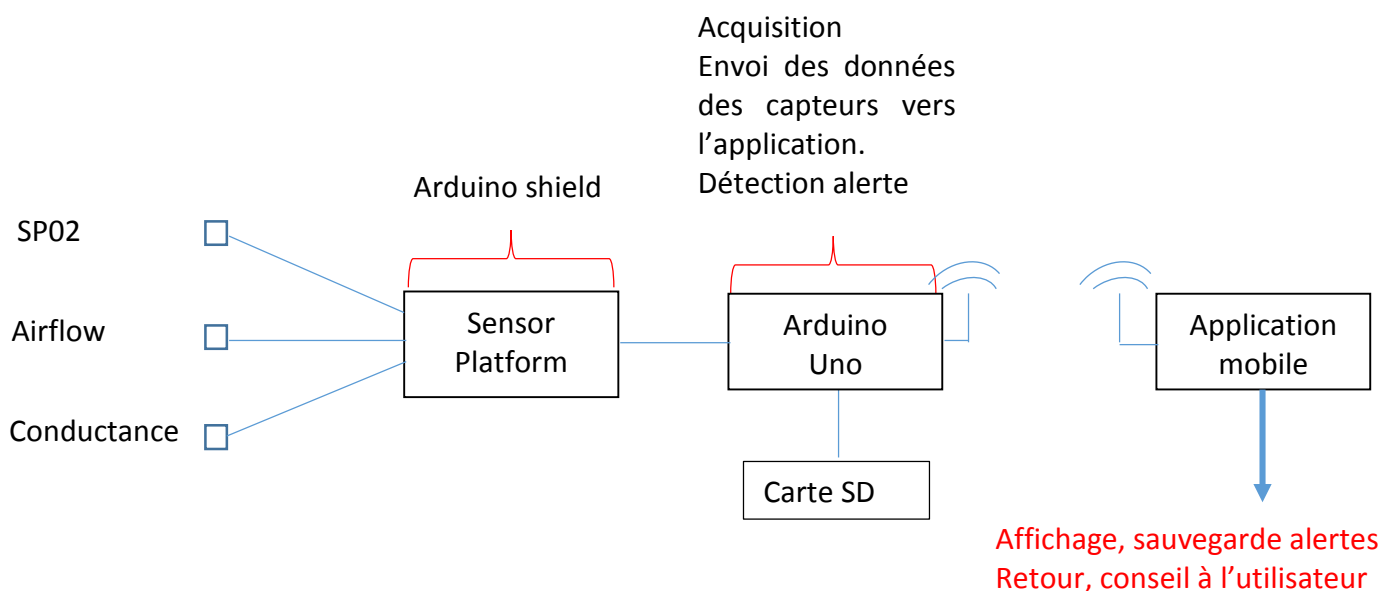


Figure 3: Architecture générale du système

La carte Arduino récupère à intervalles réguliers la valeur des différents capteurs. Ces données sont analysées, toujours dans l'Arduino et envoyées par Bluetooth sur l'application qui va les

afficher. Si l'analyse des données détecte une anomalie, celle-ci est envoyée sur l'application afin de prévenir l'utilisateur du problème. L'application, à son tour permettra d'afficher la valeur actuelle des capteurs, afficher les alertes et les sauvegarder afin de pouvoir dresser un historique des alertes. Nous allons maintenant zoomer sur la partie Arduino + plateforme et la partie application mobile.

a. Partie Arduino

1. Architecture

Cette partie se compose de 4 blocs. Le premier est un bloc de récupération des données issues des capteurs. Il va acquérir de nouvelles mesures à intervalles réguliers (exemple toutes les secondes). Le deuxième va comparer ces mesures aux seuils trouvés précédemment afin de savoir si ces valeurs sont normales ou pas. Le troisième, le bloc de corrélation va permettre d'établir un diagnostic en fonction des valeurs des capteurs. Enfin, le quatrième, le bloc Bluetooth permet d'envoyer des données à la tablette. Il est à noter que le bloc de comparaison et de corrélation auraient également pu se trouver du côté de l'application. En effet, les données provenant des capteurs auraient pu être envoyées directement sur l'application qui, elle, se serait chargée d'établir les corrélations. Cependant, dans ce cas, en cas de perte de connexion entre l'application et l'Arduino, les corrélations ne pourraient plus se faire et on perdrait alors de détecter des problèmes. En corrélant localement, sur la carte Arduino, il est possible de détecter des alertes et même de les enregistrer sur une carte SD.

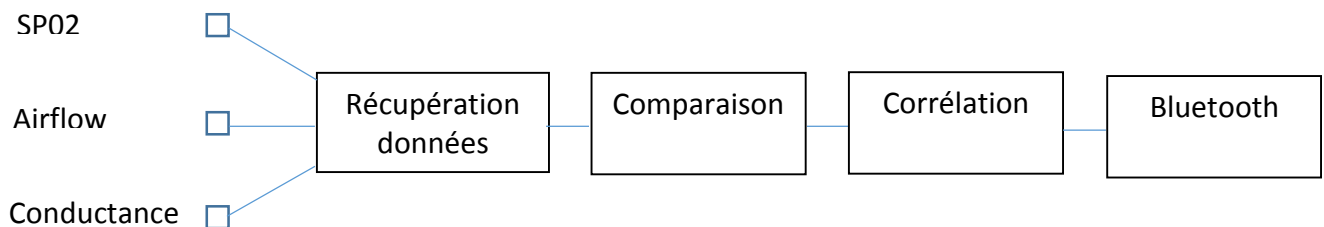


Figure 4: Schéma bloc de la partie Arduino

2. Principe général

Chaque seconde (valeur qui peut être modifiée pour diminuer la consommation ou augmenter la précision des mesures), le système se réveille (grâce à une interruption du TIMER 2) pour prendre une nouvelle mesure des 4 grandeurs les plus importantes :

- Rythme cardiaque
- SPO2
- Airflow
- Activité galvanique

A la suite de ces nouvelles mesures, on regarde si ces mesures sont « normales » en les comparant à leurs valeurs seuils trouvées plus haut. Si elles sont normales, on les envoie par Bluetooth vers l'application afin qu'elles soient affichées et on attend la prochaine

interruption. Dans le cas contraire (mesures qui ne sont pas dans les normes), on va augmenter la fréquence d'échantillonnage à 3 mesures par seconde (fréquence qui peut être réduite ou augmentée) afin d'avoir un meilleur suivi des valeurs et on les envoie. Si au bout de 10 secondes les valeurs ne sont toujours pas bonnes, on rentre dans le bloc de corrélation et on établit le diagnostic. Les 10 secondes d'enregistrement permettent de savoir si c'est une vraie alerte ou si ce sont seulement des mesures marginales dues à la défaillance d'un capteur ou un artefact. Après le diagnostic, les valeurs et le diagnostic sont envoyés par Bluetooth à l'application afin de donner l'alerte. L'envoi s'effectue sous forme de trame qu'il faudra ensuite décoder.

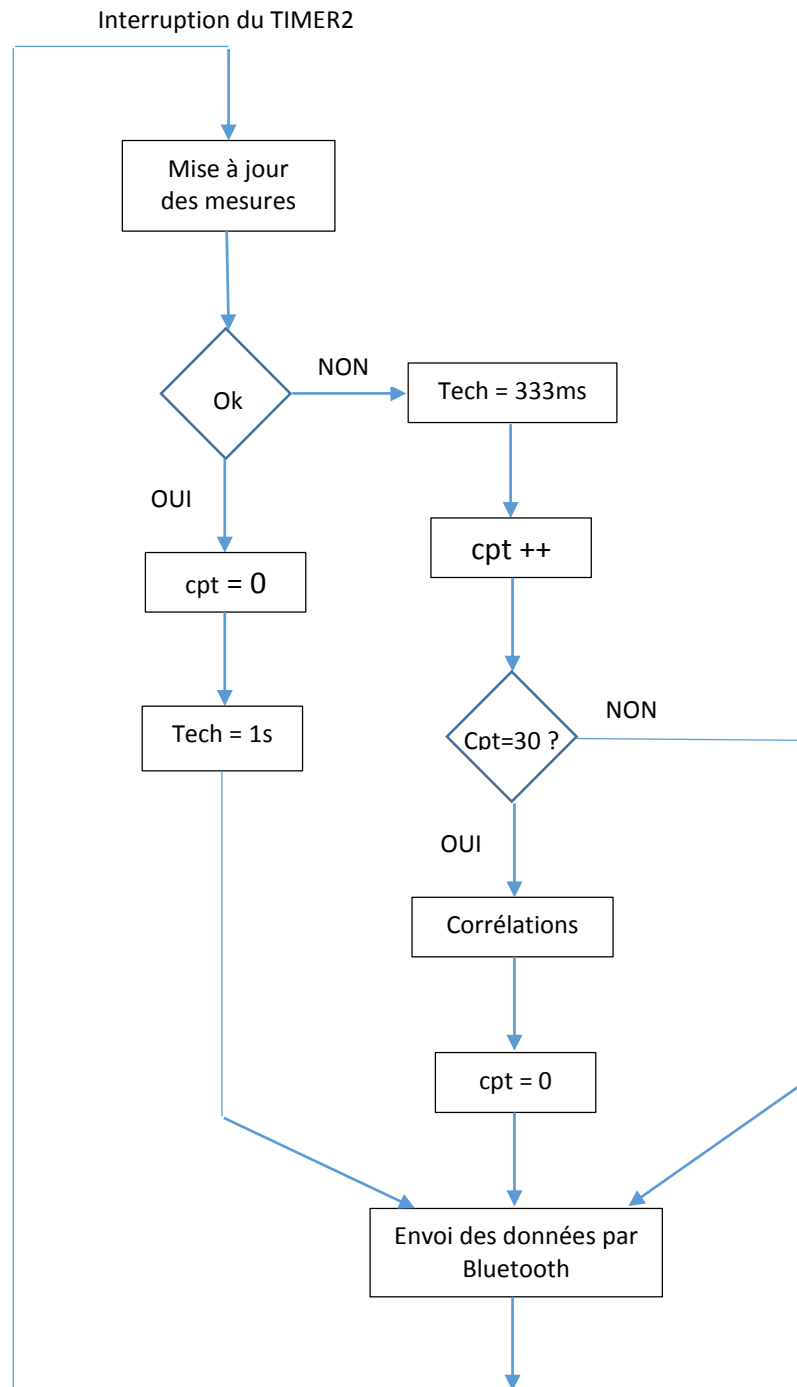


Figure 5: Schéma bloc de la partie Arduino

Les données envoyées sont donc : soit les mesures seules lorsqu'aucune alerte n'a été détectée, soit les mesures et le diagnostic lorsqu'une corrélation a été trouvée. De plus, *cpt* est un compteur qui permet de savoir si on a atteint les 10 secondes d'enregistrement après la première mesure d'une valeur anormale. Les 10 secondes sont atteintes lorsque *cpt* est égal à 30 puisque la période d'échantillonnage est de 333 ms.

b. Application mobile

1. Développement de l'application

L'application a été créée avec Qt Creator grâce à QtQuick en langage C++ associé au langage Qml et javascript. Le C++ est utilisé pour les calculs dans le programme principal alors que le qml qui est un langage déclaratif est utilisé seulement pour l'interface graphique et l'affichage. Qt Creator est un IDE qui permet entre autre de déployer des applications sur quasiment toutes les cibles : Android, IOS, Symbian bien que celle choisie fut Android. L'application a été développée pour les « API level 14 » ou plus, ce qui veut dire que les appareils antérieur à l'Api level 14 ou qui n'ont pas fait la mise à jour ne pourront pas installer l'application.

2. Fonctionnement général

Au lancement de l'application, celle-ci va demander à se connecter par Bluetooth à l'Arduino. La connexion établie, cette dernière va commencer à envoyer des données par Bluetooth sous forme de trame comme vue précédemment. A la réception d'une nouvelle trame, celle-ci va être décodée afin d'extraire les valeurs des capteurs et le type de l'alerte si un problème a été détecté. Les différentes valeurs extraites vont alors être transférées du côté du qml pour être affichées. Par contre, si une alerte est décodée, le détail de celle-ci est enregistré avant d'être affiché dans l'historique des alertes grâce au qml. Le détail des trois grandeurs (pouls, saturation et airflow) avant et après l'alerte correspondante, est alors tracé sous forme de courbes.

3. Détail des classes et affichage

Huit classes ont été utilisées afin de réaliser l'application. Le diagramme de classe est donné en *Annexe 3*. Par ailleurs, trois fichiers qml, correspondants aux différentes fenêtres de l'application ont été créés. Voici la présentation des différentes classes et fichiers qml :

- *RemoteSelector* :

Permet de rechercher les services Bluetooth environnant afin de sélectionner le service distant procuré par l'Arduino (service de « Serial Port Profile ») afin de s'y connecter.

- *Client* :

Permet de se connecter à un service.

- *Bluetooth* :

Regroupe les deux précédentes classes afin de se connecter à l'Arduino, initialise le Bluetooth de la tablette, permet de recevoir et envoyer des données.

- *DataMemory* :

Regroupe les méthodes permettant de sauvegarder les alertes, de lire la mémoire des alertes ou d'aller chercher une information en mémoire

- *QmlData* :

Classe contenant les informations relatives à une alerte. Une alerte est composée du type de l'alerte, de son numéro (alerte 1, 2, 3...), des valeurs de pouls, SPO2 et airflow au moment de l'alerte et de trois buffers contenant les valeurs de pouls, SPO2 et airflow avant et après l'alerte afin de retracer l'historique de l'alerte.

- *BtData* :

Permet de trier les données reçues par le Bluetooth, de décoder la trame reçue. Elle fait le lien entre les données reçues, les données à sauvegarder et celle à envoyer vers le qml pour l'affichage comme par exemple la valeur du pouls, du SPO2 et d'Airflow en temps réel.

- *ScreenDetails* :

Contient les informations relatives à l'écran : sa taille (hauteur et largeur) ou son orientation.

- *Display* :

Cette classe contient les éléments visuels de l'application comme la taille de l'écran ou un modèle pour l'affichage des alertes.

Au niveau de l'affichage et donc des fichiers qml, trois fenêtres ont été créées :

- *Accueil.qml* :

Correspond à la page d'accueil de l'application.

- *UserInterface.qml*

Correspond à la page où les données ainsi que les courbes temps réelles sont affichées.

- *UserAlertes* :

Correspond à la page où l'historique des alertes est affiché. Contient également, lorsque l'on clique sur une alerte, son détail ; c'est-à-dire ce qui s'est passé avant et après l'alerte.

4. Diagramme de séquence

Les diagrammes suivants montrent l'enchaînement des actions au sein de l'application. Le premier représente le cas où aucune alerte n'a été envoyée de la part de l'Arduino. En revanche, le second montre l'enchaînement des actions dans le cas où une alerte a été envoyée et reçue par l'application.

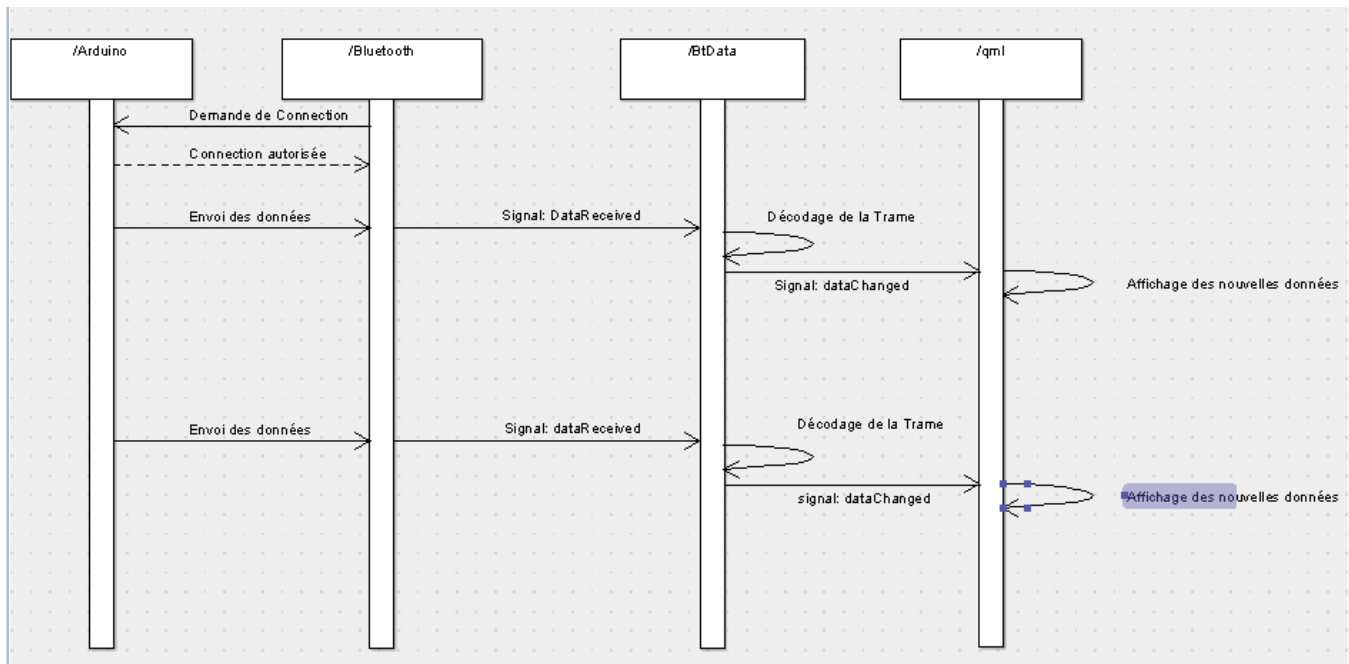


Figure 6: Diagramme de séquence lorsqu'une alerte a été envoyée

Comme vu précédemment, l'application demande à se connecter. Lorsque la connexion est établie, l'Arduino commence à envoyer des trames de données. A la réception de ces dernières, elles sont alors décodées dans la classe BtData. Comme aucune anomalie n'a été décodée, les anciennes valeurs de pouls, saturation et airflow à afficher sont rafraîchies avec les nouvelles grâce au signal dataChanged(). Enfin la partie qml affiche les nouvelles données.

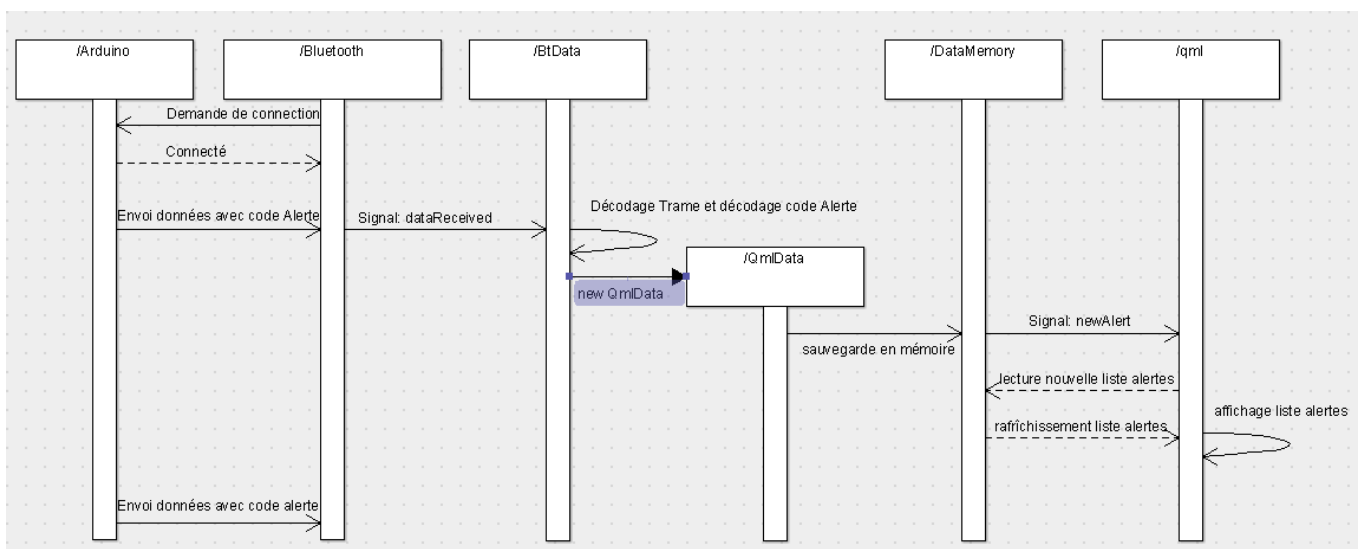


Figure 7: Diagramme de séquence dans le cas où une alerte a été envoyée

Dans ce cas, l'Arduino envoie une trame contenant un code alerte. A la réception de la trame, elle est décodée dans la classe BtData qui détecte une alerte. Un nouveau « QmlData » qui correspond à la description d'une alerte est alors créé puis immédiatement sauvegardé. La

partie Qml affiche alors, en lisant la nouvelle liste des alertes en mémoire le nouvel historique des alertes qui a été mis à jour.

D. Conception détaillée

Nous allons rentrer encore dans les détails des deux grosses parties en étudiant chaque bloc et chaque algorithme important.

a. Partie Arduino

1. Récupération des données

Tout d'abord, nous allons nous intéresser au fonctionnement des deux capteurs qui nous intéressent : la sonde oxymétrique et le capteur d'airflow. Le premier se connecte sur la plateforme grâce à une connectique spécialement adaptée (cf Annexe 1). Mais en fait, ce capteur se compose de 8 pins numériques qui se connectent aux pins numérique 6 à 13 de l'Arduino. Ce module fonctionne par interruption sur front montant de la Pin 6. A chaque interruption, la valeur du pouls et de la saturation est rafraîchie. Ces deux valeurs sont obtenues par conversion de l'afficheur 7 segments de la sonde en nombre. En effet, la sonde est fabriquée de telle sorte que les deux valeurs s'affichent sur le petit écran de la sonde. A chaque interruption, l'un après l'autre, chaque nombre de l'écran est envoyé sur les pins 7 à 13 sous forme de sept segments. Il faut alors recréer le nombre en fonction des segments allumés. Le détail de la fonction se trouve ci-dessous :

```
void eHealthClass::readPulsioximeter(void)
{
    uint8_t digito[6];

    uint8_t A = 0;
    uint8_t B = 0;
    uint8_t C = 0;
    uint8_t D = 0;
    uint8_t E = 0;
    uint8_t F = 0;
    uint8_t G = 0;

    for (int i = 0; i<6 ; i++) { // read all the led's of the module
        A = !digitalRead(13);
        B = !digitalRead(12);
        C = !digitalRead(11);
        D = !digitalRead(10);
        E = !digitalRead(9);
        F = !digitalRead(8);
        G = !digitalRead(7);

        digito[i] = segToNumber(A, B, C ,D ,E, F,G);
        delayMicroseconds(3100);
    }

    SPO2 = 10 * digito[3] + digito[2];
    BPM  = 100 * digito[5] + 10 * digito[1] + digito[0];
}
```

Quant au capteur d'airflow, il est également connecté sur la plateforme avec deux broches (+ et masse) comme le montre l'Annexe 1. Mais ces deux broches sont également connectées sur l'entrée analogique A1 de l'Arduino pour le + et la masse de l'Arduino pour la masse du

capteur. Il suffit ainsi de lire la broche A1 grâce aux fonctions Arduino « `analogread()` » afin d'obtenir la valeur du capteur comprise entre 0 et 1023. La plateforme étant livrée avec la bibliothèque *E-Health* définie en *Annexe 4*, il suffit d'appeler les fonctions « `getBPM()` », « `getOxygeneSaturation()` » et « `getAirflow()` » afin d'obtenir les différentes mesures.

Par ailleurs, afin de récupérer en permanence ces données, j'ai utilisé une interruption sur le `TIMER2`. L'interruption est déclenchée à chaque overflow du `TIMER2` qui est un compteur 8 bits. Pour configurer le processeur `ARM328P` présent sur l'Arduino, j'ai utilisé une bibliothèque Arduino déjà existante : *MsTimer2*. En quelques mots, pour faire fonctionner le `Timer2`, il faut jouer sur les registres `TIMSK2` pour autoriser l'interruption par overflow du `TIMER2` et `TCCR2A` et `TCCR2B` pour définir le type de génération du signal (normale, PWM..). Par ailleurs, il faut lui associer une fonction d'interruption grâce au vecteur `TIMER2_OVF_VECT`. La fonction associée remet à 255 – N ms le `TIMER2` afin qu'il déborde tous les N ms et lève un flag afin de commencer l'acquisition des nouvelles mesures.

2. Bloc de comparaison

A chaque interruption, de nouvelles valeurs sont acquises. Ces valeurs vont alors être comparées à des valeurs de référence afin de savoir si elles sont bonnes ou mauvaises. Pour les blocs de comparaison et de corrélation, j'ai créé la classe *Monitor* qui regroupe les méthodes nécessaires à ces deux blocs. En particulier, les méthodes *monitorHeart*, ci-dessous et *monitorSPO2* permettent de comparer la valeur du pouls et de l'oxymétrie à leur valeur de référence.

```
byte Monitor::monitorHeart(int bpm)
{
    //tachycardie
    if (bpm >= 110)    //!< condition à modifier en fonction de chaque
individu (rythme cardiaque normal propre)
    {
        nbTachycardie ++;
        return 1;          //!< Problème
    }

    else if (bpm <= 40)
    {
        nbBradycardie ++;
        return 1;          //!< Problème
    }

    else
    {
        return 0;          //!< Pas de problème
    }
}
```

Ici, les deux valeurs de références sont 100 et 40 bpm. Si la nouvelle mesure se trouve entre ces deux valeurs, on retourne le code 0, ce qui signifie que la mesure est normale. Dans le cas contraire, on incrémente un compteur et on retourne le code 1, ce qui signifie que la mesure n'est pas normale et qu'il faut commencer ou continuer l'acquisition des mesures à `Tech=333ms`. (cf Figure 5). Les deux compteurs présents dans la fonction serviront dans le bloc

de corrélation. Pour la mesure de la saturation, le principe est le même sauf que les valeurs de références sont 94% et 99% inclus. Entre les deux, la mesure est normale. Dans le cas contraire, on incrémente un autre compteur et on retourne le code 1, synonyme de problème sur la mesure.

En revanche, pour la valeur d'airflow, un autre principe a été utilisé. En effet, il ne suffit pas de regarder la valeur du capteur pour détecter une anomalie, il faut regarder la période respiratoire. Pour cela, une première idée a été de détecter les expirations (les fronts montants des valeurs du capteur d'Airflow) grâce au comparateur analogique présent sur l'ATMEGA328, le microcontrôleur de l'Arduino. Il suffisait alors de comparer la valeur sortante du capteur respiratoire et arrivant sur la Pin A1 de l'Arduino à une valeur de référence présente sur la Pin A0 de l'Arduino. La valeur de référence pouvait être soit une tension appliquée sur la Pin A0, soit une tension interne de 1,1V équivalente à une valeur analogique de 225. En utilisant cette dernière option, il faut alors configurer le comparateur pour qu'il déclenche une interruption à chaque fois que la valeur d'Airflow dépasse 225. Pour cela, j'ai utilisé la bibliothèque *Arduino AnalogComp*. Celle-ci joue sur les registres *ADCSRB*, *ACSR*, *DIDR1* et sur le vecteur d'interruption relatif au comparateur : *ANALOG_COMP_VECT*. Ainsi, il suffisait de compter le temps entre deux interruptions. Ceci a été fait grâce au *TIMER1*, par interruption. Cette méthode marchait bien, même si il fallait souffler assez fort afin que la valeur d'Airflow dépasse la référence de 225. Cependant, pour une raison qui m'est inconnue, en utilisant le comparateur, je ne pouvais plus lire la valeur sur la Pin A1 et donc la valeur d'Airflow. Il m'a fallu alors trouver une autre solution qui est décrite ci-dessous.

```
// On regarde le temps d'inspiration ou le temps d'apnée
if (airflow < 100{          // inspiration ou non expiration (=apnee)
    if (cpt2 == 0{          // si il n'y a pas de pb détecté (alors cpt2 = 0)
        if (airflowCpt1 >= 10{      // si 10s d'apnée
            apneeFlag = true;      // on lève le flag apnee
        }
        else {
            airflowCpt1 ++;
            airflowCpt2 = 3*airflowCpt1;
        }
    }
    else {
        if (airflowCpt2 >= 30){
            apneeFlag = true;
        }
        else{
            airflowCpt2 ++;
            airflowCpt1 = airflowCpt2 / 3;
        }
    }
}
else {
    if (airflowCpt1 >= 10 || airflowCpt2 >=30){      // si 10s d'apnée
        correlFlag = true;
        apneeFlag = true;
    }

    airflowCpt1 = 0;
    airflowCpt2 = 0;
}
```

J'ai donc décidé de relever, comme les valeurs de pouls et de saturation la valeur d'Airflow à intervalles réguliers. Si cette valeur est en-dessous de 100, alors cela veut dire que l'utilisateur inspirait (ou ne respirait pas). Dans ce cas on incrémente un compteur qui correspond au temps d'apnée. Ici, *airflowCpt1* correspond au cas où Tech=1s et s'incrémente toutes les secondes alors que *airflowCpt2* correspond au cas où Tech=333ms et s'incrémente donc 3 fois plus vite. Au moment où une nouvelle expiration est détectée, on regarde la valeur des compteurs. Si leur valeur est supérieure à 10 secondes pour le premier ou 30 secondes pour le deuxième, alors plus de 10 secondes d'apnée ont été mesurés, ce qui correspond à une apnée. On lève alors un flag afin de montrer qu'une apnée a été détectée.

3. Bloc de corrélation

D'après la figure 5, on a vu que dès qu'une mauvaise mesure était détectée, on augmentait la fréquence d'acquisition et on attendait 10 secondes de mauvaises mesures avant de déclencher la corrélation. Ceci permettait d'éviter de corréler pour des mauvaises mesures isolées ou des artefacts. Cependant, au bout de 10 secondes de mesures anormales, on déclenche la corrélation. La fonction de corrélation repose sur la valeur des compteurs précédents et du flag d'apnée (cf *Annexe 5*). Les compteurs sont comparés à une valeur limite *ValCritique* (25). Si un compteur est supérieur à *ValCritique*, alors cela veut dire que 25 mesures sur 30 secondes sont anormales et qu'il y a donc un vrai problème. Ainsi, suivant la combinaison des compteurs qui respectent cette condition, on établit les corrélations trouvées plus haut et on retourne le « code alerte » correspondant au problème. Par exemple, si le flag d'apnée est à « true » et que la saturation est supérieure à *Valcritique* alors on établit une apnée avec désaturation et on retourne le code « 8 ».

4. Bloc Bluetooth

Le bloc Bluetooth permet d'envoyer, après chaque acquisition de nouvelles données, les trois mesures ainsi que le code alerte lorsqu'une corrélation a été trouvée. Ceci est permis grâce au module Bluetooth HC-05 (cf *Annexe 1*) branché comme suivant :

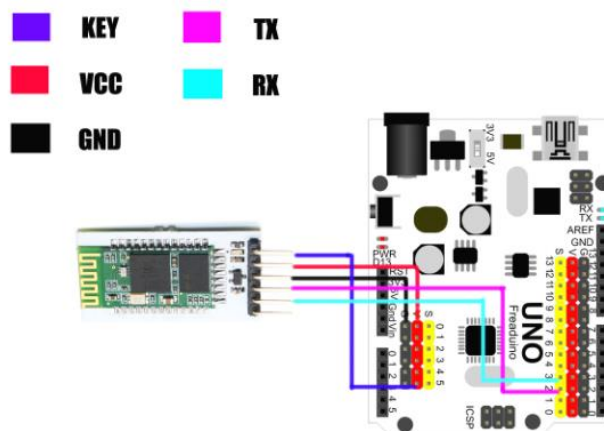


Figure 8: Branchement du module Bluetooth HC-05

Pour cela, la librairie *SoftwareSerial* permet d'émuler une liaison série et permet ainsi d'utiliser les Pins de l'Arduino comme liaison série. Ainsi nous ne sommes pas pas obligés d'utiliser les Pins de la liaison UART (Pins D0 et D1) utilisés pour le débogage. Ici, les Pins 2 pour la réception et 3 pour la transmission ont été utilisées. Cette librairie permet également de configurer la liaison (baud rate, pin à utiliser comme récepteur et transmetteur...) et d'envoyer et recevoir des octets. Cependant, on ne peut envoyer qu'octet par octet. Pour envoyer un int Arduino (16 bits), il a donc fallu séparer le mot sur 16 bits en deux mots 8 bits afin d'envoyer d'abord le msb, puis le lsb comme montre le code suivant :

```
void sendInt(int val)
{
    if (val > 255)
    {
        byte msb = val >> 8;
        byte lsb = val;
        mySerial.write(msb);
        mySerial.write(lsb);
        mySerial.write('\n');
    }
    else
    {
        mySerial.write(val);
        mySerial.write('\n');
    }
}
```

Par ailleurs, les données ont été envoyées sous forme de trame comme suivant :

	0xFFFF : début de la trame
	« \n » : caractère séparateur de données dans la trame
	16 bits (int Arduino) : Code Alerte représentant le problème détecté
	« \n »
	16 bits (int Arduino) : correspondant à la valeur du pouls
	« \n »
	16 bits (int Arduino) correspondant à la valeur du SPO2
	« \n »
	16 bits (int Arduino) correspondant à la valeur du airflow.
	« \n »
	16 bits correspondant à la valeur de la période d'échantillonnage.
	« \n »

Chaque donnée est séparée par le caractère « \n », ce qui permettra de pouvoir décoder la trame plus facilement du côté de l'application.

b. Application

1. La connexion Bluetooth

L'IDE Qt donne accès à plusieurs classes C++ qui permettent d'établir une connexion Bluetooth avec un appareil distant, de recevoir et d'envoyer des données à travers la liaison. Pour se connecter, il faut tout d'abord chercher les services Bluetooth environnant afin de savoir si le service prodigué par l'Arduino est visible. En l'occurrence le service associé à l'Arduino est un service de « Serial Port Profile » puisque, comme vu précédemment, une liaison série est émulée sur les pins 2 et 3 de l'Arduino. La recherche de service se fait dans la classe *RemoteSelector* grâce à la classe *QBluetoothServiceDiscoveryAgent*. Lorsque le bon service a été trouvé, il suffit de vérifier que l'adresse (*QBluetoothAddress*) de l'appareil est bien celle de l'Arduino. Si oui, la classe *Client* permet de se connecter à ce dernier en initialisant un socket Bluetooth (*QBluetoothSocket* est la classe associée). Etant donné que c'est l'application qui se connecte à l'Arduino, celle-ci joue le rôle d'un client et l'Arduino d'une sorte de serveur. Cet enchaînement est effectué à chaque lancement de l'application. Il est résumé dans le schéma suivant qui met en évidence le système de Signal associé à un Slot, souvent utilisé dans le projet.

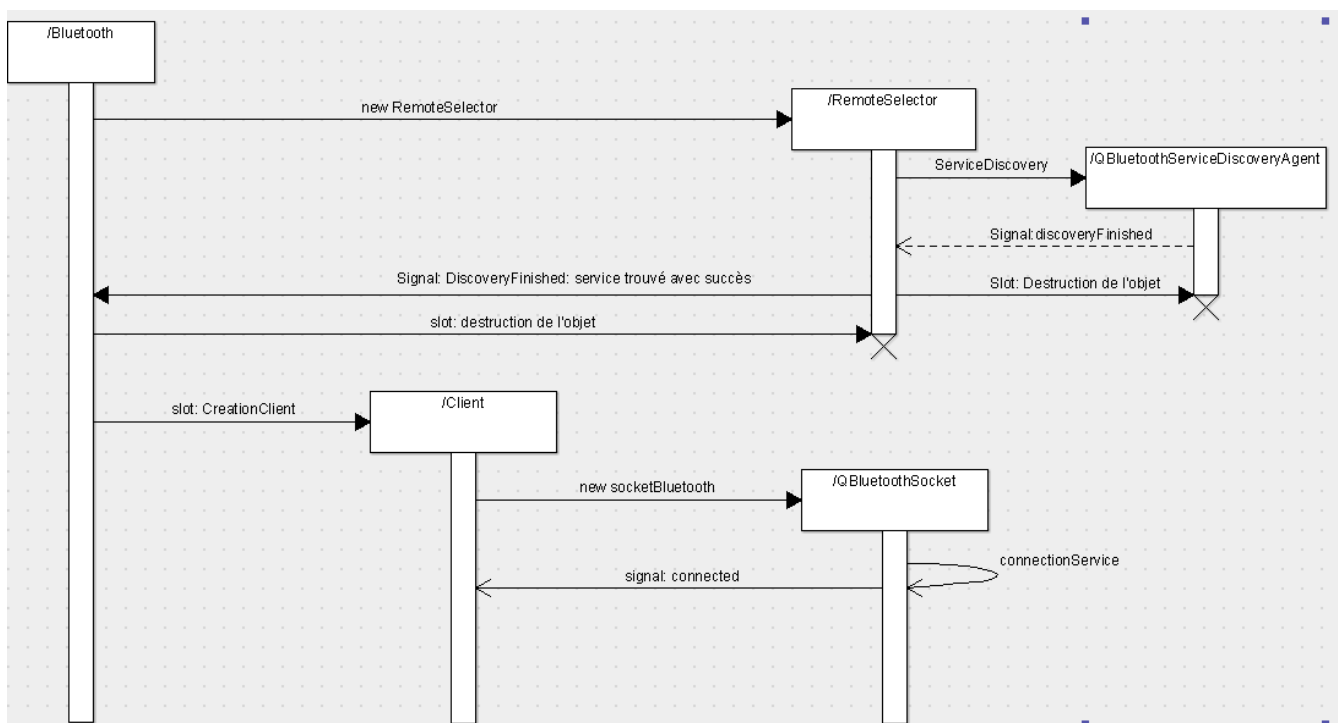


Figure 9: Diagramme de séquence lors de la connexion à l'Arduino

2. Réception des données

Lorsque la connexion est établie, nous sommes à même de recevoir les données envoyées par l'Arduino. Tout d'abord, elles sont reçues par le socket Bluetooth comme le montre le diagramme suivant :

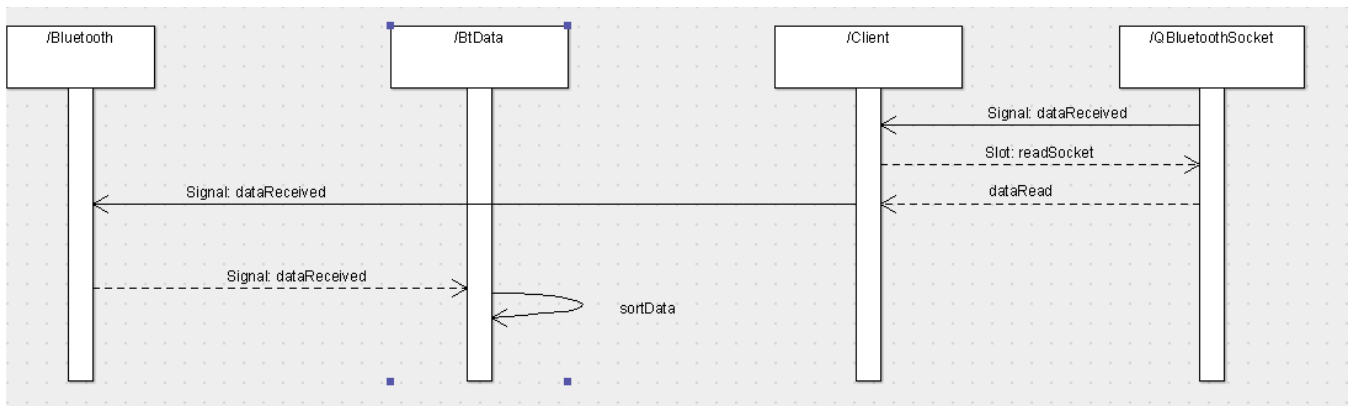


Figure 10: Diagramme de séquence de la réception des données

Le signal de réception ordonne à la classe *Client* de lire les données. Lorsqu'elles sont lues, un autre signal contenant les données lues est envoyé à la classe Bluetooth qui renvoie ce signal à la classe *Btdata* pour lui ordonner de trier la nouvelle trame.

3. Extraction des données

Tout ce qui concerne les données reçues se trouve dans la classe *BtData*. Tout d'abord les données qui se trouvent dans un tableau de *byte* sont triées. Pour cela, il faut déjà extraire chaque valeur. Le principe repose sur la recherche des différents séparateurs «\n» afin d'extraire la valeur entre deux séparateurs. Un premier curseur se trouve à la position du premier séparateur et un deuxième se trouve à la place du second. Il est alors facile d'accéder à la valeur qui se trouve entre les deux curseurs et de la reconstituer (on rappelle qu'une donnée est envoyée en deux fois : son msb et son lsb). On avance ensuite les deux curseurs : le premier prend la place du deuxième et le deuxième prend celle du prochain « \n ». Ensuite, dès lors que l'on a extrait une valeur, il faut savoir à quoi elle correspond (pouls, SPO2, airflow, code alerte...). Pour se faire, un compteur permet de savoir combien de valeurs ont été extraites dans la trame qui doit en contenir normalement 6. Il est alors facile d'associer chaque grandeur à sa valeur. Ces dernières sont ensuite enregistrées dans trois buffers contenant les dernières minutes de données. Il est à noter que les données peuvent être envoyées toutes les secondes ou toutes les 333ms. Cependant, notamment pour l'affichage des courbes temps réel, cela posait problème. En effet, comme les données arrivaient trois fois plus vite, l'échelle de temps de la courbe était faussée. Pour rétablir cela, dans le cas où Tech était égal à 333ms, une moyenne sur trois points a été faite, pour n'avoir en fait qu'un point toutes les secondes. A noter : les fonctions ne sont pas écrites dans ce rapport car elles sont trop longues.

4. Gestion des alertes

A la fin du parcours de la trame, il faut regarder le code alerte. En effet, s'il est égal à 1, c'est qu'il n'y a pas de problème. Cependant, si un code alerte est différent de 1, ce code va être traduit en une alerte textuelle, compréhensible par un homme. On va également lui associer l'heure à laquelle elle a été détectée, son numéro d'alerte (alerte 1, 2, 3...) ainsi que trois buffers contenant les valeurs de pouls, SPO2 et Airflow durant les 5 minutes autour de l'alerte (c'est-à-dire 2min30 avant et après l'alerte). Il est à noter qu'au moment de la détection de l'alerte, les deux minutes après l'alerte ne sont pas accessibles. Il faut donc attendre ce laps de temps avant d'avoir toutes les données accessibles. Toutes les caractéristiques vont être enregistrées dans un nouveau *QmlData* qui est la classe contenant la description d'une alerte puis enregistré en mémoire de la tablette afin d'y avoir accès en permanence, même à la fermeture de l'application.

5. Partie Qml

Comme expliqué précédemment, l'affichage est effectué par le Qml et constitue donc l'interface graphique de l'application. Au niveau de cette dernière, l'utilisateur peut choisir entre deux modes : un mode « normal » et un mode dédié au personnel médical. Dans le premier, l'utilisateur a d'abord accès à ses constantes ainsi qu'aux courbes temps réel.

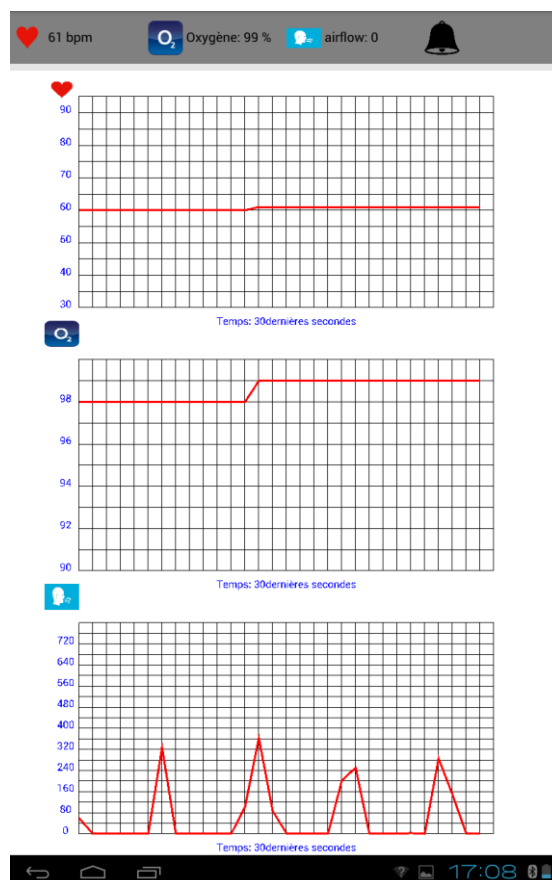


Figure 11: Première fenêtre de la partie utilisateur

On voit sur cette capture d'écran, un premier bandeau dans lequel sont affichées les valeurs actuelles des capteurs et en dessous les 30 dernières secondes de mesures. Les courbes ont été tracées en JavaScript et rafraîchies à chaque nouvelle mesure reçue. Dans ce mode, l'utilisateur a également accès à l'historique de ses alertes ainsi qu'à leur détail associé (c'est-à-dire la fenêtre de cinq minutes autour de l'alerte montrant l'évolution des trois constantes) en cliquant sur la cloche dans le bandeau tout en haut. Le rendu visuel de l'historique est donné ci-dessous :



Figure 12: deuxième fenêtre du mode utilisateur

Le détail des alertes est affiché en cliquant sur celle que l'on veut étudier. Les courbes des trois grandeurs s'affichent alors ainsi que l'endroit où la corrélation a été trouvée comme le montre les images ci-dessous.

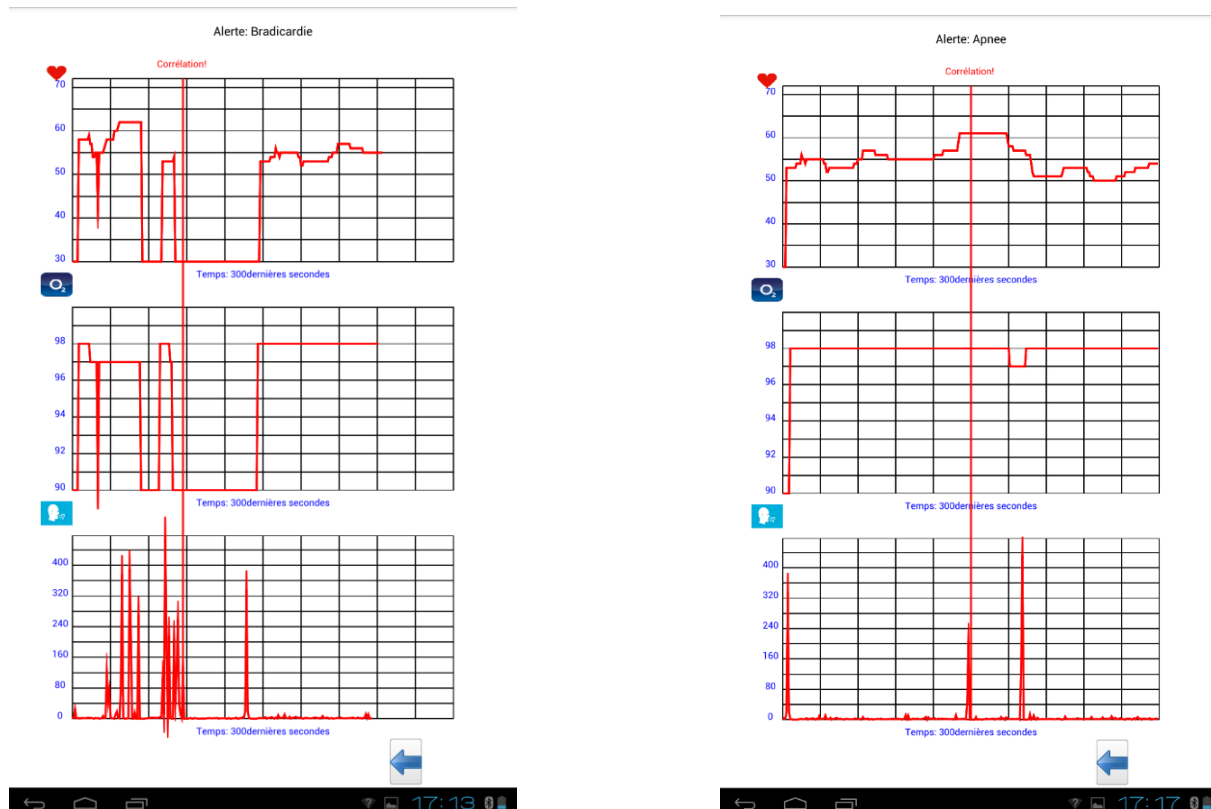


Figure 13: Deux exemples de détail d'alerte

Ici, sur l'image de gauche, une bradycardie a été détectée au niveau du curseur de corrélation (ligne rouge) et en effet, on observe que le pouls est descendu en-dessous de 40 bpm. Sur l'image de gauche, une apnée a été détectée sans désaturation. On observe bien sur la courbe d'Airflow un large temps sans respirer. Deux choses sont tout de même à faire remarquer. Premièrement, ce sont des simulations effectuées sans la fonction de contrôle des capteurs, sinon le système n'aurait pas corrélié et aurait détecté un capteur débranché. Deuxièmement, pour les apnées, la corrélation s'effectue au moment de l'expiration suivant l'apnée afin d'avoir accès à la période de temps sans respirer.

Dans le second mode, l'utilisateur arrive directement sur l'historique et le détail des alertes. Ceci permet au personnel médical de gagner du temps et d'aller directement à l'essentiel.

E. Tests pratiques

a. Test du système

Tout d'abord, la partie Arduino a été testée de plusieurs façons. Tout d'abord pendant la phase de conception de cette partie, la liaison UART a été amplement utilisée afin d'afficher les informations que l'on souhaite contrôler, notamment les codes alertes, la valeur des capteurs... De plus, pour la vérification de la liaison Bluetooth, l'UART ainsi que la liaison Debug de la tablette a permis de regarder ce qui était envoyé et reçu afin de vérifier la

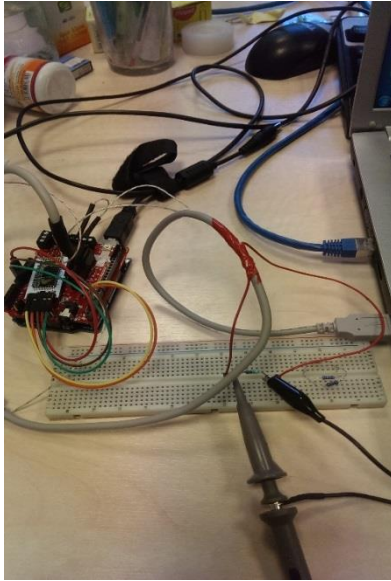


Figure 14: Système de mesure de la consommation de l'Arduino

correspondance. Par ailleurs, le système final est facile à tester, puisque l'on a accès aux valeurs temps réel. Ainsi, lorsque par exemple on expire dans le capteur d'Airflow, si un pic dans la courbe d'Airflow ne s'affiche pas, c'est qu'il y a un problème. De plus, pour les deux autres grandeurs, la sonde oxymétrique donne elle-même les valeurs. Il est donc facile de vérifier la correspondance avec ce qui est affiché sur l'application. Quant aux corrélations, il est également facile de les vérifier.

b. Consommation

Pour mesurer la consommation de la carte Arduino, il a fallu trouver un système afin de mesurer le courant débité. Pour se faire, comme la carte est alimentée par USB, un câble USB a été dégainé, puis le fil VCC a été coupé. A chaque extrémité du fil coupé, deux fils ont été soudés afin d'insérer une résistance entre, le but étant de mesurer le courant débité. L'image de gauche montre le système utilisé.

Grâce à un oscilloscope, nous avons pu alors mesurer la tension aux bornes de la résistance et ainsi mesurer l'intensité aux bornes de celle-ci. Plusieurs mesures ont été effectuées et sont rassemblées dans le tableau suivant.

Cas	SPO2	Air	Bt	U_{Rmoy}	R	I_{moy}	P
Plateforme seule				800 mV	12 Ω	67 mA	335mW
Plateforme + fonction sleep				728 mV	12 Ω	61 mA	305 mW
Avec le programme Arduino				846 mV	12 Ω	70.5 mA	352.5 mW
Programme Arduino + fonction sleep				820 mV	12 Ω	68 mA	340 mW
idem	Branché non allumé	Branché		829 mV	12 Ω	69mA	345 mW
idem	Branché allumé	Branché		1.13 V	12 Ω	94 mA	470 mW
Idem			Allumé	1.2 V	12 Ω	100 mA	500 mW
Idem	Branché allumé	Branché	Allumé	1.8 V	12 Ω	150 mA	750mW
idem	Branché allumé	Branché	Allumé connecté	1.1 V	12 Ω	92 mA	460 mW

On remarque tout d'abord que présence d'une fonction sleep diminue d'un facteur non négligeable la consommation. Le sleep mode utilisé est le mode « POWER SAVE » qui ne garde qu'une horloge et dont la source de réveil est le TIMER2, donc idéal pour l'application puisque l'on utilise celui-ci pour l'acquisition des nouvelles mesures. Par ailleurs, on remarque que le fait de brancher la sonde oxymétrique et de l'allumer augmente considérablement la consommation. En effet, la sonde présente un afficheur sept segments qui affiche les valeurs de la saturation. Il serait donc préférable, pour diminuer la consommation d'éteindre l'affichage. De plus, le fait d'allumer le module Bluetooth sans être connecté l'augmente encore. Cela est sûrement dû au fait que le module présente une diode qui ne clignote pas de la même façon selon que le module soit connecté ou non. Enfin, si on devait alimenter la carte avec une pile 6V à 19 Ah par exemple, on aurait une autonomie de 206h soit 8 jours et demi, ce qui n'est pas beaucoup pour un système qui doit fonctionner en permanence.

Pour améliorer la consommation, il serait donc important de couper toutes les petites LED présentes sur la carte, le module Bluetooth, la sonde oxymétrique... Une autre solution serait également d'améliorer la communication Bluetooth en envoyant peut-être moins de données ou moins souvent par exemple.

F. Validation du système

La plupart des points du cahier des charges sont respecté. En effet, tout d'abord, au niveau de l'Arduino, le système surveille bien en continu les trois constantes choisies. De plus, il détecte bien les mauvaises mesures et établit une corrélation avec les apnées et un problème de cœur. Ces mesures et corrélations sont bien envoyées directement sur la tablette grâce au Bluetooth. Ces affirmations sont vérifiables sur l'application qui affiche en temps réels les valeurs des capteurs mais trace également la courbe de ces grandeurs comme le montre les dernières photos de l'application.

Par ailleurs, les différentes alertes sont bien gardées en mémoire sur la tablette, tout comme leur détail associé. Cependant, le système n'est pas vraiment modulaire, il ne peut pas s'adapter à un deuxième utilisateur, d'autres capteurs. De plus, l'utilisateur n'est pas alerté, dans le sens où je ne suis pas arrivé à afficher des notifications dans le style Android. Dans le cas où il n'a pas l'application sous les yeux, il ne peut donc pas être alerté. Enfin, la consommation peut être encore améliorée.

V. Conclusion



Tout d'abord, pour reprendre la partie précédente, la plupart des objectifs ont été atteints. En effet, le système est fonctionnel, il détecte les corrélations envisagées : problèmes de cœur (tachycardie, bradycardie), les apnées avec et sans désaturation grâce à deux capteurs essentiels : une sonde oxymétrique et un capteur d'Airflow. De plus, grâce à l'application Android, l'utilisateur est alerté en temps réel lors de la détection d'un problème. Cette dernière permet également de visualiser la valeur actuelle des capteurs et de dresser un historique des alertes avec, pour chacune d'elles, le détail des 3 grandeurs avant et après l'alerte. On peut donc dire que le projet respecte le cahier des charges.




Cependant, outre la consommation qui doit bien évidemment être améliorée, beaucoup d'autres aspects restent encore à développer. Tout d'abord, à la première utilisation, l'application devrait demander à l'utilisateur de remplir un questionnaire afin d'en savoir un peu plus de lui : son âge, ses habitudes, ses problèmes de santé afin de définir la valeur normale de ses constantes. Il serait alors plus intéressant de comparer l'évolution de ses constantes par rapport à ses vraies valeurs de référence, chose que l'actuelle application ne fait pas (les valeurs de référence sont identiques quel que soit l'utilisateur). D'autre part, l'aspect de généricité et de modularité reste à implémenter. Ce point est très important car il permettrait de modifier légèrement l'architecture sans avoir à tout modifier ou d'adapter facilement le projet à d'autres capteurs afin, par exemple de détecter de nouvelles corrélations. Ces capteurs pourraient être de plusieurs sortes : des capteurs d'activité (qui mesurent les efforts fournis pendant la journée), des capteurs environnementaux (humidité, température, pression...) ou tout autre capteur qui mesure une grandeur agissant sur le corps et la santé d'une personne. On peut également envisager une mini caméra afin de pouvoir examiner la personne à distance. Cela permettrait également d'intégrer facilement plusieurs utilisateurs à la même plateforme. Le système serait donc capable de changer les valeurs de références avec lesquelles les valeurs des capteurs sont comparées en fonction de l'utilisateur. De plus, on peut imaginer que l'application puisse dialoguer avec l'extérieur en cas de problème pour alerter les proches ou du personnel médical ou tout simplement pour leur envoyer historique mensuel des alertes. Enfin, pour que ce projet puisse un jour exister et être commercialisé, il faudrait que les capteurs soient moins invasifs car, à l'heure actuelle, avec les capteurs présents sur la plateforme, il est impossible de vivre en permanence connecté. Mais ce projet reste tout de même une première brique afin de concevoir ses différents points.




Par ailleurs, je pense avoir acquis une nouvelle expérience dans le domaine de la bioélectronique. Ce stage a consolidé mon idée de travailler dans ce domaine. J'espère que cette expérience me servira cette année pour trouver un stage dans ce dernier. Je me suis également rendu-compte des difficultés à tenir les objectifs fixés. En effet, au début du stage, je pensais pouvoir finir le projet et implémenter de nouvelles fonctionnalités assez rapidement. Cependant, même si le projet est quasiment abouti, des problèmes sur mon matériel personnel (Bluetooth, pilotes, programmes et extensions manquantes...) m'ont fait perdre beaucoup de temps et empêcher le développement d'autres fonctionnalités.




Pour conclure, le bilan retenu est globalement positif, que ce soit sur le projet où les fonctionnalités implémentées fonctionnent ou sur le côté humain où j'ai découvert le monde de la recherche basé sur l'autonomie et des conditions de travail assez souples.


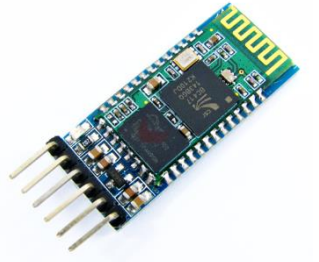
Annexe 1 : Description des capteurs

Element	Commentaires
Plateforme de capteurs	<ul style="list-style-type: none"> - Shield Arduino sur lequel se branchent les capteurs médicaux qui vont suivre.  <p>The image shows a red printed circuit board (PCB) labeled 'e-Health Sensor Platform V2.0'. It is an Arduino shield with various electronic components including a microcontroller, resistors, capacitors, and connectors. It has a USB Type-B port on the left, a DC power jack on the right, and a multi-pin header at the bottom for connecting to an Arduino board.</p>
Sonde oxymétrique	<ul style="list-style-type: none"> - Se branche sur la plateforme de capteurs. - Permet de mesurer la saturation en oxygène en %, c'est-à-dire la quantité d'oxygène dissout dans le sang. Cette grandeur est utile pour les personnes ayant une oxygénation instable par exemple. - Permet également de mesurer le pouls en bpm, qui est une grandeur essentielle dans la détection de nombreux problèmes de santé.  <p>The image shows a pulse oximeter probe, which is a small black device with a white cable. The cable has a connector that fits into the multi-pin header of the e-Health Sensor Platform. The probe itself has a small display screen and a button.</p>
Capteurs pour électrocardiogramme	<ul style="list-style-type: none"> - Se branche sur la plateforme de capteurs. - Permet de mesurer l'activité électrique du cœur grâce à 3 électrodes placées au niveau du cœur : une au niveau du cœur (+), une de l'autre côté du cœur (-) et une sous le cœur (N). - L'un des plus courants des tests médicaux. - Utile dans la détection de multiples problèmes cardiaques comme l'ischémie cardiaque (diminution de l'apport sanguin artériel), les attaques cardiaques...

	
Capteur d'airflow nasal.	<ul style="list-style-type: none"> - Se branche sur la plateforme de capteurs. -  <ul style="list-style-type: none"> - Retourne une valeur analogique entre 0 et 1023. La valeur est positive lors d'une expiration. Lors d'une inspiration, la valeur est égale à 0. - Permet de détecter les apnées, les apnées obstructives ou une possible hypoxémie (diminution de la quantité d'Oxygène dans le sang).
Thermomètre	<ul style="list-style-type: none"> - Se branche sur la plateforme de capteurs. - Permet de mesurer la température du corps humain. - Il est possible d'améliorer la précision du thermomètre en recalibrant le capteur avec un voltmètre. 
Tensiomètre automatique	<ul style="list-style-type: none"> - Utile pour les personnes souffrant de maladies cardiovasculaires.

	<ul style="list-style-type: none"> - Possibilité de récupérer les mesures par USB. 
Capteur de position	<ul style="list-style-type: none"> - Se branche sur la plateforme de capteurs. - Capteur contrôle 5 positions : Debout/Assis, couché sur le dos, couché sur le ventre, couché latéralement sur la gauche ou la droite. 
Capteur d'activité galvanique	<ul style="list-style-type: none"> - Se branche sur la plateforme. - Permet de mesurer la conductance électrique de la peau qui varie en fonction de son niveau d'humidité (en fait la sueur). - La mesure est effectuée au bout des doigts. - Possibilité d'améliorer la précision du capteur en le recalibrant avec un voltmètre. 

<p>Glucomètre</p>	<ul style="list-style-type: none"> - Permet de mesurer le taux de glucose dans le sang - Stocke les différentes valeurs dans la mémoire du glucomètre. - Possibilité de récupérer les mesures en connectant le capteur à la plateforme. - Calcule le taux de glucose en lui donnant une goutte de sang. 
<p>Capteurs pour électromyogramme</p>	<ul style="list-style-type: none"> - Se branche sur la plateforme. - Mesure l'activité électrique des muscles grâce à des électrodes placés sur le muscle. 
<p>Capteur d'activité physique</p>	<ul style="list-style-type: none"> - Montre Bluetooth permettant d'envoyer les données recueillies - Mesure les déplacements quotidiens. - Mesure l'activité pendant la nuit. - Application mobile Android et IOS 

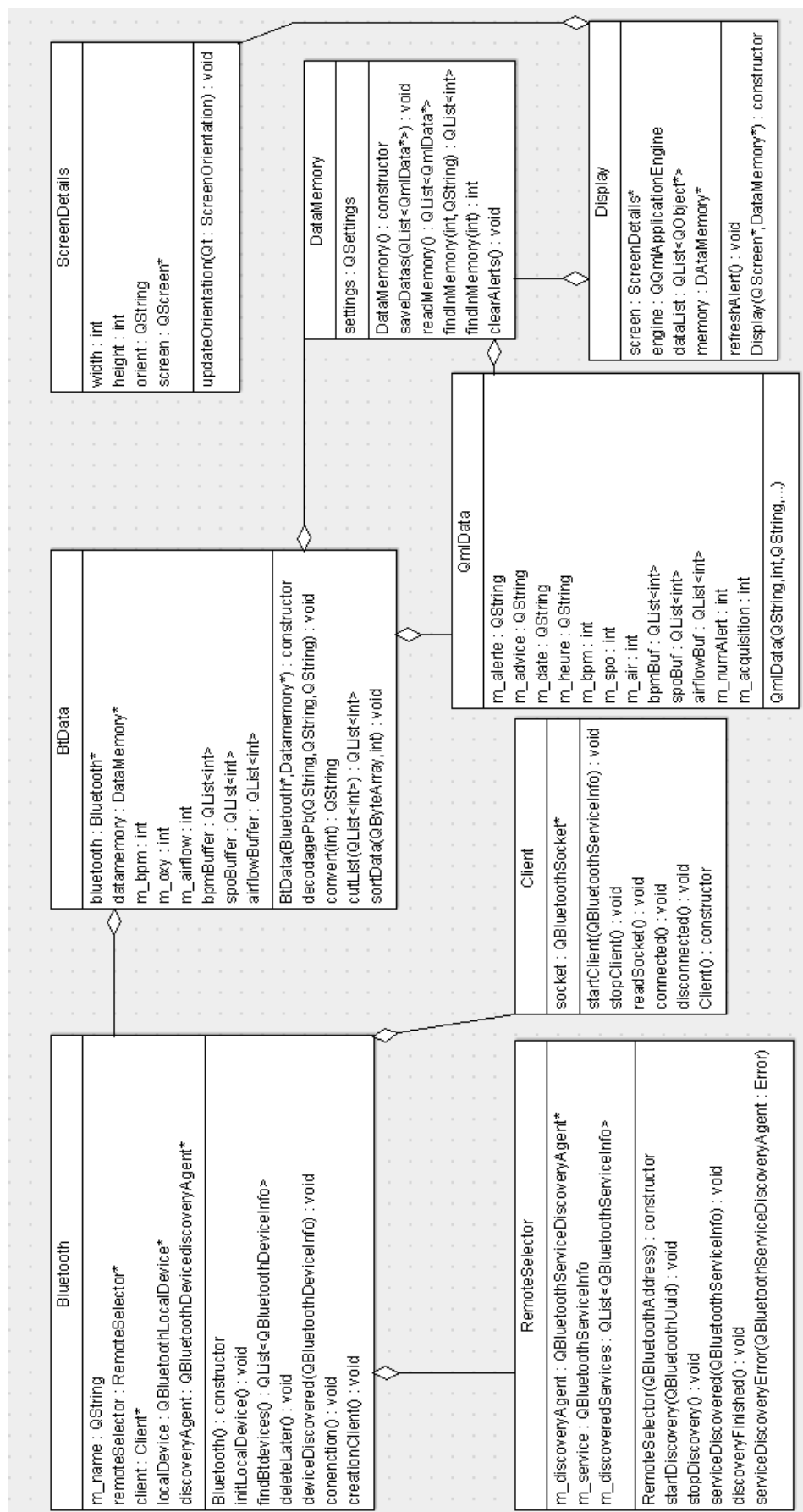
<p>Arduino Uno</p>	<ul style="list-style-type: none"> - Carte électronique sur laquelle se fixe son shield : la plateforme de capteurs. - Microcontrôleur ATmega328. - Horloge : 16 MHZ. - Tension d'alimentation : 5 V - 14 pins numériques - 6 pins analogiques - Mémoire Flash : 32 kB - Datasheet en Annexe X.  <p>The image shows an Arduino Uno board, a popular microcontroller board based on the ATmega328P. It features a blue PCB with a USB Type-B port, a DC power jack, and a reset button. The board is populated with various components including the ATmega328P microcontroller, a 16MHz crystal oscillator, and a USB-to-UART bridge. The text 'MADE IN ITALY' and 'ARDUINO UNO' are visible on the board.</p>
<p>Module Bluetooth pour Arduino</p>	<ul style="list-style-type: none"> - Module HC-05 - Alimentation : 3.3 à 6V. - Pins : 3.3 à 6V tolérantes - Fonctionne comme une liaison Série - AT Commandes permettent de changer ses caractéristiques : Baud Rate, fonctionnement en maître ou esclave... (cf Annexe X).  <p>The image shows an HC-05 Bluetooth module, a small, compact module used for wireless communication between two devices. It features a green PCB with a gold-plated USB Type-A connector on one end and a 5-pin header on the other. The module is populated with various components including a Bluetooth chip, a crystal oscillator, and a small antenna.</p>

Tablette Archos	<ul style="list-style-type: none">- Tablette Archos utilisée pour le développement de l'application mobile. 
-----------------	--

Annexe 2 : Références

- [1] Emil Jovanov, Aleksandar Milenkovic, Chris Otto and Piet C de Groen: A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation, 2005
- [2] Ahmed Lounis, Abdelkrim Hadjidj, Abdelmadjid Bouabdallah and Yacine Challal: Secure and Scalable Cloud-based Architecture for e-Health Wireless sensor networks, 2012
- [3] Min, Cheol-Hong ; Tewfik, A.H. : Integration of Wearable Wireless Sensors and Non-Intrusive Wireless in-Home Monitoring System to Collect and Label the Data from Activities of Daily Living, 2006
- [4] Alfredo Burgos, Alfredo Goñi, Arantza Illarramendi, and Jesús Bermúdez: Real-Time Detection of Apneas on a PDA, 2010
- [5] Site internet de l'entreprise Capsule: <http://www.capsuletech.fr/>
- [6] Site Internet de l'entreprise MIR : <http://www.spirometry.com/>
- [7] Site internet de l'entreprise TWITOO : <http://www.twitoo.org/>
- [8] Source apnée du sommeil :
<http://sommeil.univ-lyon1.fr/articles/onen/apnee/print.php>
- [9] Source sur le pouls :
 - http://umvf.univ-nantes.fr/cardiologie-et-maladies-vasculaires/enseignement/cardio_309/site/html/cours.pdf
 - <http://www.bodyscience.fr/?Qu-est-ce-qu-un-rythme-cardiaque>
- [10] Source sur la respiration :
http://www.uvp5.univ-paris5.fr/WIKINU/docvideos/Grenoble_1011/launois_rollinat_sandrine/launois_rollinat_sandrine_P03/launois_rollinat_sandrine_P03.pdf

Annexe 3 : Diagramme de classe de l'application



Annexe 4: Fonctions importantes de la classe e-Health

```
/*!*****
    //!      Name:      initPulsioximeter()

    //!      Description: Initializes the pulsioximeter sensor
                                *

    //!      Param: void

    //!      Returns: void
                                *

    //!      Example: eHealth.initPulsioximeter();
                                *
    /*!*****
*****

void eHealthClass::initPulsioximeter(void)
{
    // Configuring digital pins like INPUTS
    pinMode(13, INPUT);           pinMode(12, INPUT);
    pinMode(11, INPUT);           pinMode(10, INPUT);
    pinMode( 9, INPUT);           pinMode( 8, INPUT);
    pinMode( 7, INPUT);           pinMode( 6, INPUT);
    // attach a PinChange Interrupt to our pin on the rising edge
}
```

```
/*!*****
    //!      Name:      readPulsioximeter()

    //!      Description: It reads a value from pulsioximeter sensor

    //!      Param: void
                                *

    //!      Returns: void
                                *

    //!      Example: readPulsioximeter();
                                *
    /*!*****
*****

void eHealthClass::readPulsioximeter(void)
{
    uint8_t digito[6];

    uint8_t A = 0;
    uint8_t B = 0;
    uint8_t C = 0;
    uint8_t D = 0;
    uint8_t E = 0;
    uint8_t F = 0;
    uint8_t G = 0;

    for (int i = 0; i<6 ; i++) { // read all the led's of the module
        A = !digitalRead(13);
        B = !digitalRead(12);
        C = !digitalRead(11);
        D = !digitalRead(10);
        E = !digitalRead(9);
        F = !digitalRead(8);
        G = !digitalRead(7);

        digito[i] = segToNumber(A, B, C ,D ,E, F,G);
    }
}
```

```

                                delayMicroseconds(3100); //2800
microseconds
                                }

                                SPO2 = 10 * digito[3] + digito[2];
                                BPM  = 100 * digito[5] + 10 * digito[1] + digito[0];
                                }

```

```

//!*****
//!      Name:      getAirFlow()
//!                                     *
//!      Description: Returns an analogic value to represent the airflow
//!      *
//!      Param : void
//!                                     *
//!      Returns: int with the airFlow value (0-1023)
//!      *
//!      Example: int airFlow = eHealth.getAirflow()
//!      *
//!*****
*****

int eHealthClass::getAirFlow(void)
{
    int airFlow = analogRead(A1);

    return airFlow;
}

```

Annexe 5 : Fonctions importantes de la classe Monitor

```
byte Monitor::correlation(boolean _apneeFlag)
{
    if (nbSPO2 < valInter && _apneeFlag == false && nbTachycardie <
valInter &&
        nbBradycardie < valInter && nbHyperventil < 3)
        return 1;
    else if (nbSPO2 > valCritique && _apneeFlag == true) //nbAirflow >
valCritique)
    {
        position = eHealthClass::getBodyPosition();
        nbSPO2 = 0;
        //nbAirflow = 0;
        return 8;          // 8 = apnée
    }
    else if (nbTachycardie > valCritique && nbHyperventil > 3)
    {
        nbTachycardie = 0;
        nbHyperventil = 0;
        return 2;          //2 = Stress/emotion
    }
    else if (nbBradycardie > valCritique)
    {
        nbBradycardie = 0;
        return 3;          //3 = Bradycardie
    }
    else if (nbTachycardie > valCritique)
    {
        nbTachycardie = 0;
        return 4;          //4 = tachycardie
    }
    else if (_apneeFlag == true)
    {
        return 5;          // 5 = apnee sans SPO2;
    }
    else if (nbSPO2 > valCritique)
    {
        nbSPO2 = 0;
        return 6;          //6 = probleme d'oxygénation
    }
    else if (nbHyperventil > 3)
    {
        nbHyperventil = 0;
        return 7;          //7 = hyperventilation
    }
    else
        return 10;
}
```