Date : samedi 8 février 2014

Documentation du projet « SensorTag »

DMIT – ESIR 3 INGENIERIE BIOMEDICALE

Rédacteurs :

- Ana Fernandez Marquez
- Alexis Piel
- Clotaire Delanchy

Contenu

1 Contexte et objectifs du projet 3

2 Problèmes à résoudre 3

3 Solutions testées 4

3.1 Dongle fourni [échec] 4

3.2 Utilisation d'un MacBook Air [succès] 4

3.3 Nouveau dongle USB BLE [succès] 4

4 Architecture 5

4.1 Architecture globale 5

4.2 Architecture BLE/SensorTag/Serveur 6

4.3 Entités 6

4.3.1 SensorTag 6

4.3.2 SensorTagDaemon 6

4.3.3 Server 7

4.3.4 Base de données 7

4.3.5 Client 8

4.4 Communication 8

5 Environnement de développement et installation du projet 8

5.1 Environnement de développement 8

5.2 Installation du projet 9

6 Lancement du projet 9

7 Etat actuel du projet et améliorations possibles 9

7.1 Etat actuel 9

7.2 Améliorations possibles 10

7.2.1 Traitement de signal plus précis 10

7.2.2 Communication du Daemon vers le Serveur en HTTP 10

7.2.3 Centralisation et unification des technologies 10

1 Contexte et objectifs du projet

Le projet consiste à réaliser une chaîne de télémonitoring complet : de la réception des informations de multiples capteurs jusqu'à leur affichage, en passant par leur traitement et stockage via un serveur. L'objectif final de ce projet est d'avoir une station domotique qui réagit en fonction des différentes informations provenant des capteurs et de lancer des alertes en cas de besoin.

Ce projet veut surveiller la fréquence cardiaque via une ceinture portée par le patient, sa température et aussi son équilibre pour détecter ses mouvements (des chutes en particulier). Les données seront analysées, affichées, puis envoyées vers un serveur domotique qui actionnera (ou non) des modules dans la maison.

2 Problèmes à résoudre

La partie traitée par notre groupe concerne la communication entre un module contenant des capteurs et un daemon lisant et analysant ses données. Cette communication est assurée en Bluetooth 4.0 Low Energy (BLE). Comme cette technologie est récente, l'objectif principal de notre partie du projet est de réaliser un proof of concept (PoC) sur les éléments suivants :

1. communiquer en BLE avec le capteur depuis un ordinateur ;
2. récupérer les données des différents capteurs ;
3. moduler certains paramètres comme la fréquence d'échantillonnage.

Si ce PoC est fonctionnel, les objectifs secondaires sont les suivants :

- analyser les données en provenance des capteurs ;
- détecter des chutes éventuelles ;
- lancer des alertes vers un serveur (en cas de chute par exemple) ;
- transmettre nos données au groupe d'affichage, utilisant Processing via un fichier texte.

Notre priorité est donc la communication en BLE entre le capteur, appelé SensorTag, et un ordinateur. Le SensorTag est produit par la société Texas Instruments et fournit différents capteurs au sein d'un même module. Voici une liste des capteurs qui nous intéresse :

- boutons poussoirs ;
- 1 capteur de température infra-rouge ;
- 1 capteur de température ambiante ;
- 1 gyroscope ;
- 1 accéléromètre.

3 Solutions testées

Nous avons testé différentes solutions pour établir la communication avec le SensorTag, et essayer que cette communication soit facile à mettre en place. Voici la liste de ce que nous avons essayé, et dans le cas d'un échec, l'explication du pourquoi la solution n'en est pas une.

3.1 Dongle fourni [échec]

Le SensorTag nous a été fourni avec 2 autres modules en provenance de Texas Instruments : le KeyFob (télécommande à 2 boutons) et un dongle USB pour communiquer avec. Le KeyFob fonctionne également en BLE.

Après avoir installé le logiciel proposé par Texas Instruments, sous Windows, pour communiquer avec le KeyFob via le dongle USB fourni ; nous avons essayé de communiquer avec le SensorTag par l'intermédiaire de ce même dongle et de ce même logiciel.

La communication fonctionne bien, mais nous devons forcément passer par le logiciel de Texas Instruments, ce qui va nous poser problème dans le développement futur où nous avons besoin de communiquer directement avec le SensorTag, sans passer par le logiciel fourni.

Nous avons essayé en utilisant la librairie bluez sous Linux, qui permet de gérer le BLE, mais après maintes tentatives et beaucoup de recherche, la conclusion s'est imposée : le dongle USB que nous avons ne respecte pas les standards HCI et par conséquent, il n'est pas possible de l'utiliser sans logiciel propriétaire, ou sans flasher le programme du dongle pour y mettre le nôtre - ce qui aurait été extrêmement fastidieux.

Le dongle n'est donc pas bon. Nous avons demandé l'achat d'un nouveau dongle USB BLE respectant les standards.

3.2 Utilisation d'un MacBook Air [succès]

L'ESIR n'était pas sûre de pouvoir nous fournir un nouveau dongle USB BLE à temps pour la fin du projet. Nous avons donc récupéré un MacBook Air qui possède déjà un module BLE intégré. Cependant, nous n'avons pas trouvé de librairies vraiment utilisables sur Mac OS X. Nous avons donc utilisé la librairie créée par M. Alfredo Hernandez et son laboratoire.

Cette librairie répond à nos besoins, mais elle est en Objective-C et n'est donc pas multiplateforme (spécifique Mac ici). Le code est aussi assez compliqué. C'est néanmoins une solution fonctionnelle pour notre projet.

3.3 Nouveau dongle USB BLE [succès]

Ne pouvant emporter le MacBook Air avec nous pour travailler sur le projet durant notre temps libre, un membre a pris la liberté d'acheter pour lui un dongle USB BLE respectant les standards. Nous avons donc trouvé une librairie en Node.js (JavaScript) permettant de communiquer avec le SensorTag et qui est très facile d'utilisation. De plus, cette librairie est compatible Mac et Linux (donc les systèmes Unix en général). Le code est simple à comprendre et à utiliser.

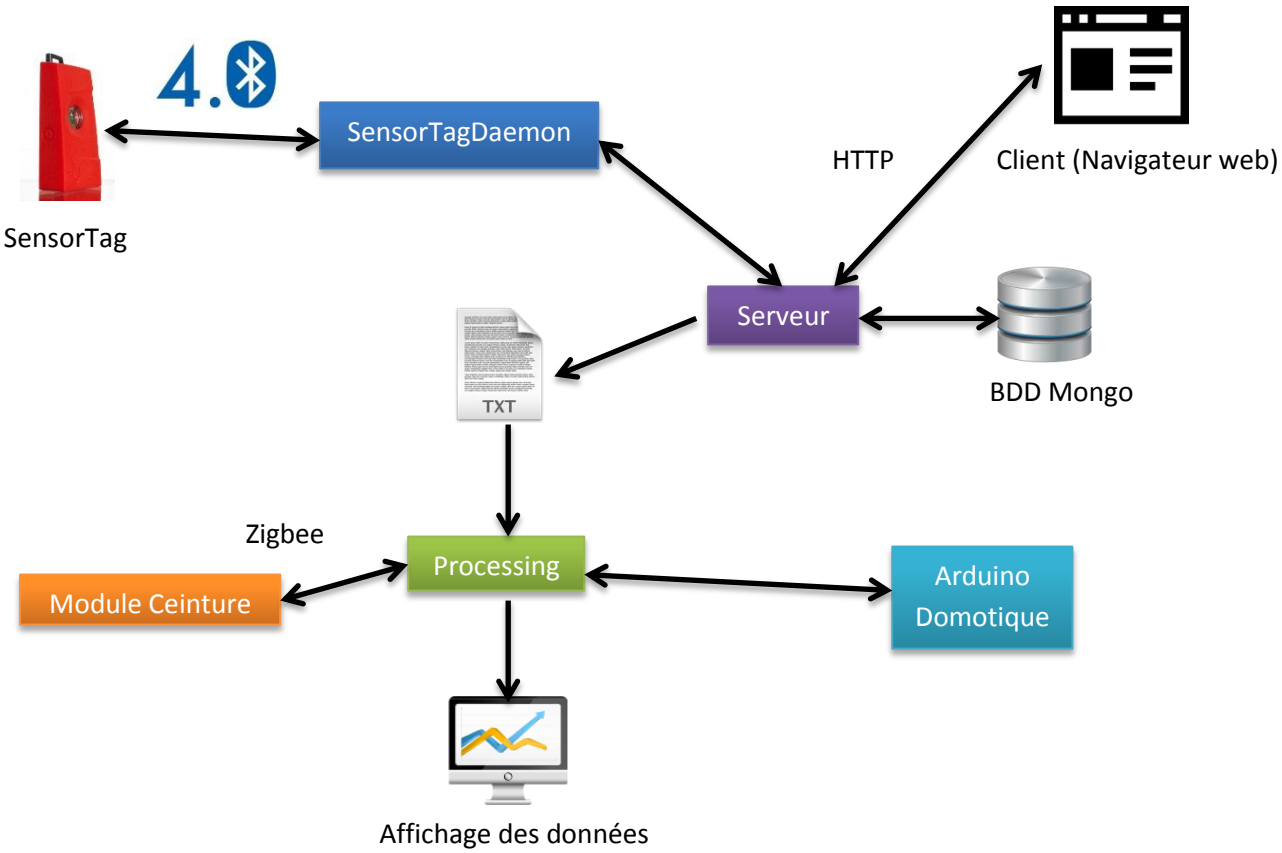
Nous l'avons donc installé à la fois sous Linux, et à la fois sur le MacBook Air. Cette librairie répond mieux à nos besoins que la librairie fournie par le laboratoire de M. Hernandez. C'est cette solution qui a finalement été retenue.

La librairie s'appelle "noble" et permet d'utiliser le BLE en Node.js. A cela, nous avons ajouté la librairie "sensortag", du même auteur et toujours en Node.js, permettant une communication très simple avec le SensorTag.

4 Architecture

Nous allons dans cette partie décrire l'architecture du projet et les différents éléments qui composent notre partie.

4.1 Architecture globale

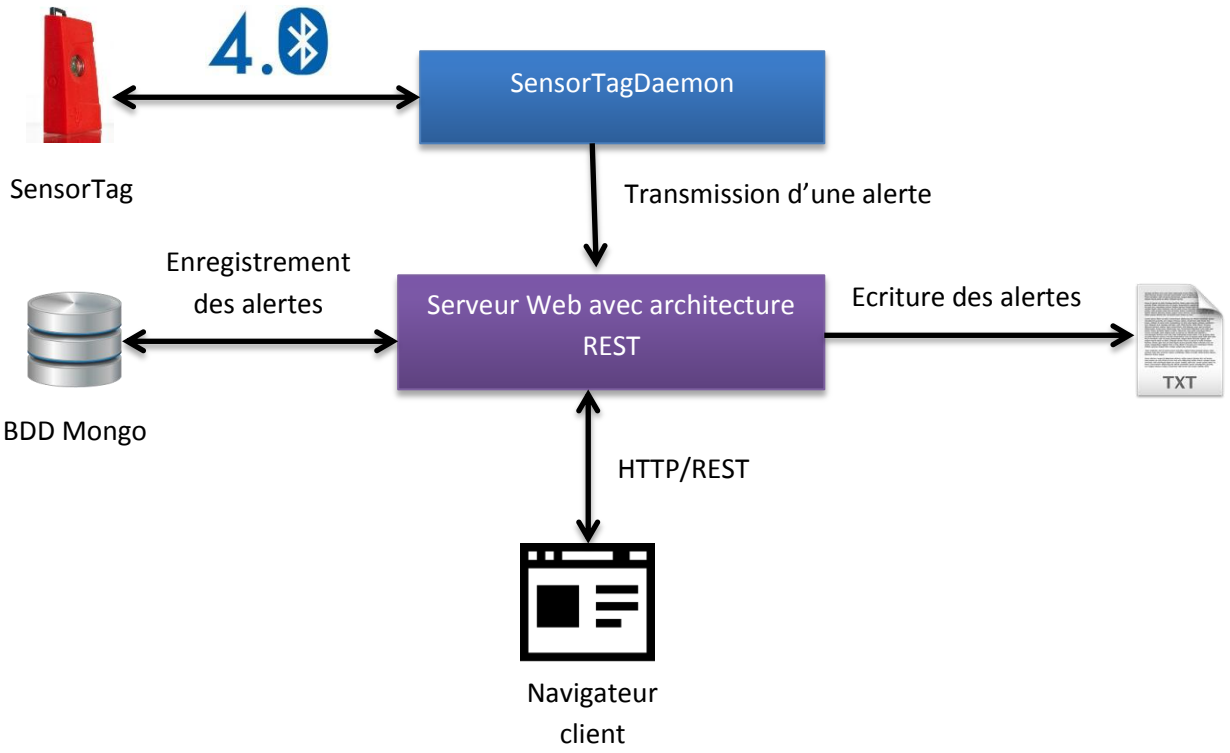


Le projet est donc divisé en 4 groupes. Un premier groupe s'occupe de la récupération des informations de la ceinture (fréquence cardiaque) et de l'envoi de ces données vers le groupe "Processing". Le groupe "Processing" récupère les données du groupe "Fréquence Cardiaque" et du groupe "SensorTag" et affiche ces données, et lancent des alertes au groupe "Domotique". Ce dernier doit écouter les alertes en provenance du groupe "Processing" et doit lancer des actions domotiques via Arduino en fonction des informations reçues.

Enfin, notre groupe “SensorTag” récupère les informations du SensorTag et les envoie au groupe “Processing”. La communication entre les groupes s’effectuent principalement à l’aide de fichiers textes qui sont lus/écrits par les différents groupes.

Nous allons détailler un peu plus notre architecture, et les actions supplémentaires que nous avons effectuées.

4.2 Architecture BLE/SensorTag/Serveur



Comme nous pouvons le voir, plusieurs entités sont à l’œuvre dans notre architecture.

4.3 Entités

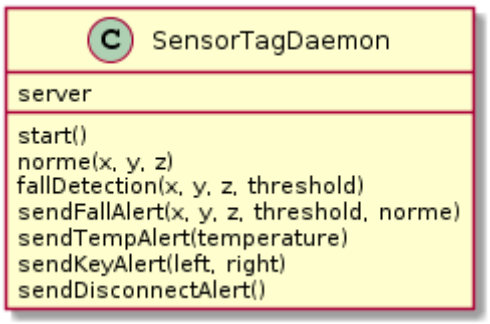
4.3.1 SensorTag

Le SensorTag est donc le capteur de chez Texas Instruments permettant de surveiller différents éléments comme la température, l’appui sur des boutons etc. Il communique uniquement via BLE.

4.3.2 SensorTagDaemon

Ce petit programme est un démon qui tourne donc en continue et va communiquer avec le SensorTag dès qu’il est assez proche. Une fois connecté avec le SensorTag, le Daemon va récupérer les données du SensorTag (température, valeurs du gyroscope, etc.) et va les analyser.

Cette analyse va permettre de détecter les chutes, et de lever des alertes. Dès qu'une alerte est détectée (chute ou appui sur un bouton par exemple), elle est transmise par le Daemon au Server.

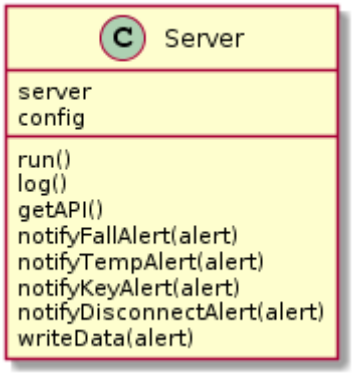


4.3.3 Server

Le serveur est un serveur web, qui est lancé en continue et qui possède une architecture REST. Il va recevoir les alertes levées par le SensorTagDaemon et va stocker ses alertes dans une base de données associée : ici MongoDB.

Il va aussi écrire dans un fichier texte les alertes qui sont enregistrées dans la base de données, afin que le groupe "Processing" puisse les afficher.

Enfin, nous avons implémenté ce serveur web pour que, dans une évolution future du projet, on puisse accéder à toutes les alertes enregistrées depuis une simple page web, à distance. Un exemple de page web est déjà existant.



4.3.4 Base de données

Cette base de données enregistre toutes les alertes reçues par le serveur web via le SensorTagDaemon. Le système de gestion de base de données utilisé dans ce projet est MongoDB.

Voici comment les données sont stockées actuellement :

```
{
	_id: ObjectId('6f35cb7179396e0d610000d9'),
	date: '2013-03-20 13:10:02',
	type: 'KeyAlert',
		data: {
		left: 'true',
		right: 'false'
	}
}
```

4.3.5 Client

La partie cliente correspond aux pages web qui peuvent être consultées depuis n'importe quel navigateur. Ces pages peuvent être visionnées par exemple, par des médecins ou la famille du patient qui est suivi.

4.4 Communication

Ces différents modules communiquent entre eux par les moyens suivants :

- **SensorTag/Daemon :** BLE
- **Daemon/Serveur :** local (ils font partis du même programme) et il serait bien que le Daemon envoie les alertes au Serveur via HTTP dans le futur. Cela rendrait le Daemon indépendant du serveur, et il pourrait alors être lancé sur n'importe quelle machine. Nous pourrions alors avoir, par exemple, le Daemon dans la maison du patient, et le serveur web à l'hôpital.
- **Client/Serveur :** HTTP + REST
- **Serveur/Processing :** fichier texte

Voici le format du fichier texte écrit :

AAAAMMJJHHMM ; B ; F ; T

- AAAAMMJJHHMM décrit l'année, le mois, le jour, l'heure et les minutes à laquelle l'alerte a été levée.
- B signifie les boutons enfoncés et prend la forme d'un int : 3 pour les deux boutons, 1 pour le bouton droit, 2 pour le bouton gauche, et 0 pour aucun des deux.
- F code la chute : 0 pour aucune, 1 pour une chute signalée.
- T code la température en décimal avec un chiffre après la virgule.

5 Environnement de développement et installation du projet

5.1 Environnement de développement

L'environnement de développement possède les éléments suivants :

- OS : Mac OS X ou distributions GNU/Linux ;
- MongoDB ;
- Node.js ;
- un SensorTag ;
- dongle USB BLE pour communiquer avec le SensorTag ;
- Git (pour plus de praticité. Si la ligne de commande vous fait peur, jetez un coup d'œil à GitHub for Mac)

Les procédures d'installations peuvent être trouvées dans le README.md à la racine du projet, sur le GitHub correspondant. Elles sont aussi disponibles sur les sites Internet des projets (MongoDB, node.js).

5.2 Installation du projet

Nous partons ici du principe que l'environnement de travail (Node.js, MongoDB) est correctement installé.

Le projet est disponible sur GitHub à cette adresse : https://github.com/Eriatolc/dmit-esir-project

Vous pouvez le télécharger en tant qu'archive .zip ou bien le cloner à l'aide de la commande appropriée :

```
git clone https://github.com/Eriatolc/dmit-esir-project.git
```

L'architecture du projet est décrite sur le README du GitHub (qu'il est fortement conseillé de lire).

Le projet utilise de nombreux modules Node.js. Pour les installer, allez dans le dossier racine du projet (dmit-esir-project), et lancer la commande suivante (en administrateur/root) :

```
sudo npm install
```

Les dépendances du projet s'installeront d'elles-mêmes. Les deux plus importantes sont "noble" et "xpc-connection" (si vous êtes sur Mac).

6 Lancement du projet

Pour lancer le projet, il faut lancer le script bin/dmit-esir-project.js. Ce script lance à la fois le serveur web et le daemon du SensorTag. Voici la commande pour le lancer :

```
sudo node bin/dmit-esir-project.js
```

Vous pouvez aussi rendre le fichier exécutable et le lancer directement comme tel :

```
chmod u+x bin/dmit-esir-project.js
./bin/dmit-esir-project.js
```

7 Etat actuel du projet et améliorations possibles

7.1 Etat actuel

Le projet remplit actuellement tous les objectifs du PoC listés en haut de ce document, ainsi que les objectifs secondaires cités juste après.

Il va même plus loin en stockant les alertes dans une base de données et en permettant leur affichage via une page web.

7.2 Améliorations possibles

De multiples améliorations sont possibles. Voici plusieurs pistes à explorer.

7.2.1 Traitement de signal plus précis

Il doit être possible de rendre le traitement du signal permettant la détection de chute plus robuste, en utilisant à la fois les données du gyroscope et de l'accéléromètre afin d'éviter les faux-positifs.

7.2.2 Communication du Daemon vers le Serveur en HTTP

Pour l'instant, le Daemon et le Serveur fonctionnent ensemble. Même si le code est séparé, le Daemon possède une référence directe vers le serveur. Il serait bon que cette référence disparaisse dans le futur et que le Daemon envoie les alertes directement par HTTP au Serveur, en utilisant l'architecture REST de ce dernier. Ainsi, les Daemon seraient totalement indépendant du Serveur web et ils ne seraient alors plus obligés d'être installés sur la même machine physique. Nous pourrions alors avoir le serveur à l'hôpital par exemple, et le Daemon dans la maison du patient.

7.2.3 Centralisation et unification des technologies

Comme on peut le constater sur l'architecture générale du projet, beaucoup de technologies différentes sont utilisées et de nombreux moyens de communication sont nécessaires entre les différents modules. Il serait bon de simplifier l'architecture en la centralisant un peu plus et en éliminant certaines technologies, qui au final, se révéleront peu pratiques.

Ainsi, le serveur jouerait un rôle central dans le projet. Il s'occuperait du stockage de l'ensemble des données (et pas seulement celles du SensorTag) et la partie web s'occuperait de l'affichage de ces informations. Ainsi, on améliore le serveur actuel avec la technologie web (Node.js/JavaScript) et on élimine la partie "Processing" du projet. De cette manière, il est possible d'afficher les données de n'importe où dans le monde via le Web, et non seulement là où Processing est installé. De plus, cela évite d'écrire/lire de nombreux fichiers textes différents, ce qui ralentit le programme et peut causer des erreurs (écritures simultanées).

En résumé, créer un Daemon qui communiquerait avec la ceinture et enverrait les données au Serveur doit être possible. Idem avec un Daemon qui sert d'intermédiaire entre l'Arduino domotique et le Serveur. Et l'affichage des données se ferait en accédant à une page web (la partie Client). Tous les Daemon peuvent reprendre l'architecture déjà réalisée en Node.js. Nous aurions donc uniquement de l'Arduino/Xbee/Node.js en technologie, et aucun fichier texte écrit ou lu.