République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

ECOLE NATIONALE POLYTECHNIQUE

DÉPARTEMENT D'ÉLECTRONIQUE

Mémoire de fin d'études

En vue de l'obtention du diplôme d'Ingénieur d'Etat en Electronique

Thème :

Conception d'un système de détection de chute.

Encadré par :

Mme L.HAMAMI

Réalisé par :

Mlle Roumaissa KHIRANI

Promotion : Juin 2013

REMERCIEMENTS

En premier lieu je remercie DIEU le Tout Puissant de m'avoir donné le courage et la force pour réaliser ce travail.

Je tiens à remercier vivement, ma promotrice Mme Latifa HAMAMI professeur à l'Ecole Nationale Polytechnique, ainsi que mon Co-promoteur M. Mehdi NEGGAZI doctorant au laboratoire Signal et Communications de l'ENP, pour leurs aides et leurs conseils.

Je tiens également à remercier M. Rabah SADOUN, Maître de Conférences A à l'ENP, d'avoir accepté de présider mon jury de soutenance.

Mes remerciements vont aussi à M. Llies SAADAOUI d'avoir accepté d'examiner mon travail.

Mes remerciements vont spécialement à mon ami Ahmed Belbecir, dit Midou.

Je remercie également Jong Chern Lim de l'équipe Shimmer pour son aide. Merci Jong.

Dédicace

A mes chers parents,

A ma nièce Menouna

A ma chère grand-mère Hadda ,

À mes chères sœurs, à mes chers frères,

A tous mes amis, spécialement mon ami Midou, et ma meilleure amie pendant 20ans Yasmine.

Je dédie mon travail.

Table des Matières

Table des figures

Liste des abréviations

SRS	**S**hock **R**esponse **S**pectrum
MFCC	**M**el-**F**requency **C**epstrum **C**oefficients
BAN	**B**ody **A**rea **N**etwork
SMV	**S**ignal **M**agnitude **V**ector
SMA	**S**ignal **M**agnitude **A**rea
OHA	**O**pen **H**andset **A**lliance
API	**A**pplication **P**rogramming **I**nterface
JRE	**J**ava **R**untime **E**nvironment
JVM	**J**AVA **V**irtual **M**achine
SDK	**S**oftware **D**evelopment **K**it
JDK	**J**ava **D**evelopment **K**it
AIDL	**A**ndroid **I**nterface **d**efinition **l**anguage
ADT	**A**ndroid **D**eveloppement **T**ools Plug-in
LIFO	**L**ast **I**n **F**irst **O**ut

Résumé

ملخص

التطور المتقدم لتكنولوجيات أجهزة الإشعار اللاسلكية والهواتف الذكية سهل إمكانية تصميم نظام رصد الصحة
لقد مكن توفر منصة جديدة (shimmer device) ذات الاستهلاك الضعيف والوزن الخفيف وسهولة الحمل من توفير فرص جديدة للرصد الصحي في الوقت الحقيقي
إن ظاهرة السقوط تعد من الأسباب الأكثر شيوعا في وفاة الأشخاص المسنين واستخدام هده المنصة لكشف السقوط في الوقت الحقيقي يسمح بتدخل طارئ للطبيب والتصرف بسرعة , لقد مكن التطور الهائل الذي تعرفه تكنولوجيات (تقنيات) الاستشعار اللاسلكية و الهواتف الذكية من تسهيل تصميم انظمة التطبيب عن بعد .
ان توفر قاعدة جديدة ذات الاستهلاك الضعيف و التي تتميز بصفة الوزن و سهولة النقل قد مكن من فتح افاق جديدة للمراقبة الصحية في الوقت الحقيقي.
ان استعمال هذه القاعدة لكشف حوادث السقوط التي تعتبر احد الاسباب الاكثر شيوعا لوفات الاشخاص المسنين في الوقت الحقيقي سمح بتدخل السريع للطبيب.

كلمات البحث : الوقت الحقيقي , التطبيب عن بعد, shimmer device , Android

Résumé

Le développement avancé des technologies dans les capteurs sans fil et le Smart phone permettent une conception aisée de systèmes en connected health. La disponibilité d'une nouvelle plateforme (Shimmer Device), de faible consommation, légère et pratique à porter, donne de nouvelles possibilités pour la surveillance de la santé en temps réel.
La chute étant classée parmi les causes les plus fréquentes de décès des personnes âgées, l'utilisation de cette plateforme pour la détection de chute en temps réel, permet l'intervention du médecin de secours d'agir rapidement. Nous avons réalisé un système sans fil de détection de chute par utilisation de Shimmer Device permettant l'acquisition en temps réel et Android pour le traitement des données, la prise de décision et l'envoi du message. Notre algorithme a été simulé d'abord sur Matlab puis le système complet testé en temps réel.

Mot clef : shimmer device, temps réel, télémédecine, Android.

Abstract

Recent technological advances in wireless sensors and mobile communication enabled an easy conception of new types of healthcare systems. The availability of small, lightweight and ultra-low power sensors gives new possibilities for a continuous monitoring of human biomedical data.
The use of this platform for fall detection, which are considered as one of the most common causes of death of old persons, in real time, allows us to reduce the rate of death, health problems.
In our work, we have designed and implemented a wireless fall detection real time system on Android using Shimmer Device which allows the acquisition and the analysis of the data in real time, and then taking decisions and insuring communication between doctor and patient. Our algorithm has been simulated on Matlab, and then the whole system has been tested in real time.

Keywords: shimmer device , Android, connected health, real time

Introduction Générale

Le secteur de la santé occupe une place majeure dans la société, le plus grand problème dans ce secteur est de pouvoir traiter un cas spécifique de la santé le plus tôt possible, c'est pour cela que l'on se penche de plus en plus vers le connected health (ou la télémédecine).

Le connected health est une forme de pratique médicale à distance utilisant les technologies de l'information et de la communication.

Parce que la chute est classée 5ème cause dans le décès des personne âgées, et parce que les problèmes de santé qui en résultent, sont multiples et dangereux, on pourra donc la classer parmi le principal souci de santé qui doit être surveillé à distance, et en temps réel ; c'est pour cette raison qu'une demande accrue d'un système pouvant détecter les chutes se fait sentir. Ce système devra évidemment pouvoir distinguer entre une chute réelle et les activités quotidiennes du patient surveillé à distance.

L'objectif de notre travail est de présenter une solution à ce problème, en utilisant une nouvelle plateforme, de faible consommation, la Shimmer Device, et un Smartphone ayant Android comme système d'exploitation.

Ce mémoire est organisé en cinq chapitres.

Après une introduction générale de notre travail, le premier chapitre présentera la plateforme utilisée, ses parties hardware et software ainsi que leurs avantages.

Le chapitre deux présentera une vue globale des différents algorithmes proposés dans la littérature pour la détection de chute. On citera qu'il existe principalement trois différents systèmes, basés sur la vibration du sol, ou sur la vidéo surveillance ou encore sur le capteur accéléromètre.

Nous présenterons dans le chapitre trois, l'algorithme que nous avons implémenté ainsi que la description détaillée de ses différents modules. Nous exposerons également les différentes simulations faites sur Matlab en essayant d'interpréter les résultats obtenus.

Le chapitre quatre quant à lui va introduire, l'historique d'Android, son architecture, ses avantages, et les composants le constituant.

Dans le chapitre cinq, notre système sera présenté et les différentes étapes suivis lors du développement sous Android seront expliquées ; puis une simulation de notre système sera effectuée.

Nous terminerons notre travail par une conclusion générale et quelques perspectives dans le but d'apporter des améliorations à notre système.

1.1 INTRODUCTION

Plusieurs systèmes ont été proposés pour la surveillance des personnes âgées, mais ce qui est plus important, c'est de les surveiller en temps réel. Dans ce chapitre on présente une plateforme de faible consommation, pour l'acquisition et la transmission des données en temps réel (Wireless Sensor), ce qui peut être très efficace dans le connected health.

1.2 PRESENTATION DE LA SHIMMER DEVICE

La Shimmer Device est une plate-forme sans fils, qui peut transmettre et enregistrer des données physiologiques ou cinématiques en temps réel. Conçu comme un portable pour les capteurs (voir fig.1.1), la Shimmer intègre des capteurs sans fil, ECG, EMG, GSR, Accéléromètre, Gyromètre, GPS.

La Shimmer est une plateforme extrêmement extensible qui permet aux chercheurs d'être à la pointe des technologies de la détection.

Fig. 1.1 : Shimmer Device [1]

1.2.1 Architecture hardware de la shimmer device

On présente dans cette partie l'architecture de la Shimmer et les principales parties de sa structure.

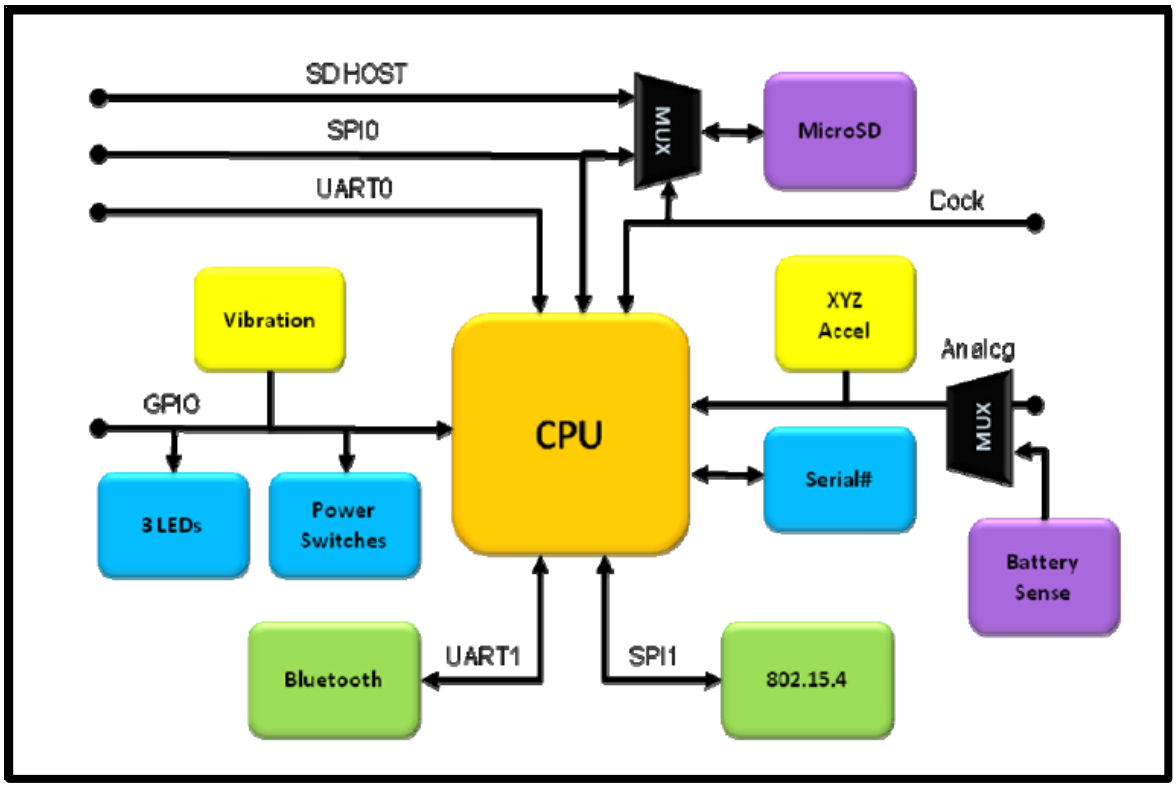

Fig. 1.2 : Architecture hardware [2]

La partie hardware contient principalement :

1. un microprocesseur MSP430F1611 (8MHz, 16bits) représentant le noyau, il contrôle toutes les opérations de Shimmer. Il a comme principale fonction le contrôle et la configuration de tous les périphériques intégrés à travers les différentes connexions (entrées/sorties). Il contient aussi un convertisseur analogique/numérique, 8 canaux 12 bits, qui reçoit les données auprès du capteur (dans notre cas, accéléromètre).
2. MicroSD supportant jusqu'à 2G de mémoire flash, cela permet à la Shimmer de stocker des quantités importantes des données (elle peut stocker en continue jusqu'à 80 jours des données de l'accéléromètre, échantillonnées à 50hz).
3. 3 diodes luminescentes.
4. Bluetooth-RN-42 et le module radio 802.15.4 pour la transmission des données.
5. Batterie rechargeable LI-polymère, 450mAh.
6. Capteur accéléromètre 3D. [2]

1.2.2. Spécifiées techniques

Nous pouvons citer ci-dessous quelques spécificités techniques de la Shimmer:

-un faible gain de sortie qui fournit une pré-amplification bipolaire.

-la source d'excitation de la LED, est contrôlée par le logiciel.

-elle comprend une prise jack 3.5mm de haute qualité, qui peut être utilisée pour brancher la carte shimmer avec le programme fourni pour l'utilisateur. [1]

1.2.3. Caractéristiques et Avantages

- ***Temps réel***

-L'acquisition, le stockage et l'affichage des données se font en temps réel.

- Une capacité de mesurer les forces, les convertir en format numérique et les diffuser directement à un dispositif hôte, sans avoir besoin d'une unité de réception intermédiaire.

- ***Poids et mobilité***

-conçue pour être portée, elle aborde les défis de la mobilité et elle nous fournit une très haute qualité de données fiables, soit comme un émetteur sans fil soit comme un enregistreur de données.

-En nous libérant des contraintes des fils, elle fournit une solution confortable pour les athlètes, les enfants et les personnes âgées.

- Sa petite taille et son poids léger (28g), nous permet de la porter lors de toutes les tâches.

- ***Fiabilité***

-Un préamplificateur peut être localisé pour éviter les pertes en ligne et les interférences.

-Une mesure flottante qui offre, aux utilisateurs, une meilleure sécurité.

-Répond aux éxigences de la sécurité de la santé d'EU, et la sécurité des équipements médicaux.

- ***Adaptabilité***

-Elle offre une plus grande valeur ergonomique que les unités câblées.

-Elle combine facilement les données de la carte avec l'accéléromètre intégré. [1]

1.2.4. Software de la Shimmer

1.2.4.1. Architecture

L'environnement TinyOS est vivement recommandé pour l'implémentation du dispositif, le test, et la validation du software intégré de Shimmer (firmware).

Le TinyOS est un système d'exploitation open-source conçu pour des réseaux de capteur sans fils.

Le TinyOS est un environnement économique grâce à sa plateforme extensive et à son code bibliothèque open-source.

Il est basé sur le langage de programmation nesC, c'est une extension du langage C, utilisé pour incarner les concepts structurants et les modèles d'exécution de TinyOS.

Il utilise le compilateur personnalisé nesC ; le langage de programmation de nesC prend en charge les applications basées en composantes structurées, il prend en charge également les modèles de concurrence TinyOS basés sur les taches et les gestionnaires d'événements matériels. Il est fait pour minimiser l'utilisation de mémoire, et de puissance de calcul pour les capteurs.

Il existe quelques définitions importantes à connaitre

- **Applications**

L'application est constituée d'un ou plusieurs composants reliés entre eux pour former un fichier exécutable. Chaque application nesC est décrite par une configuration de niveau supérieur qui relie les composantes internes.

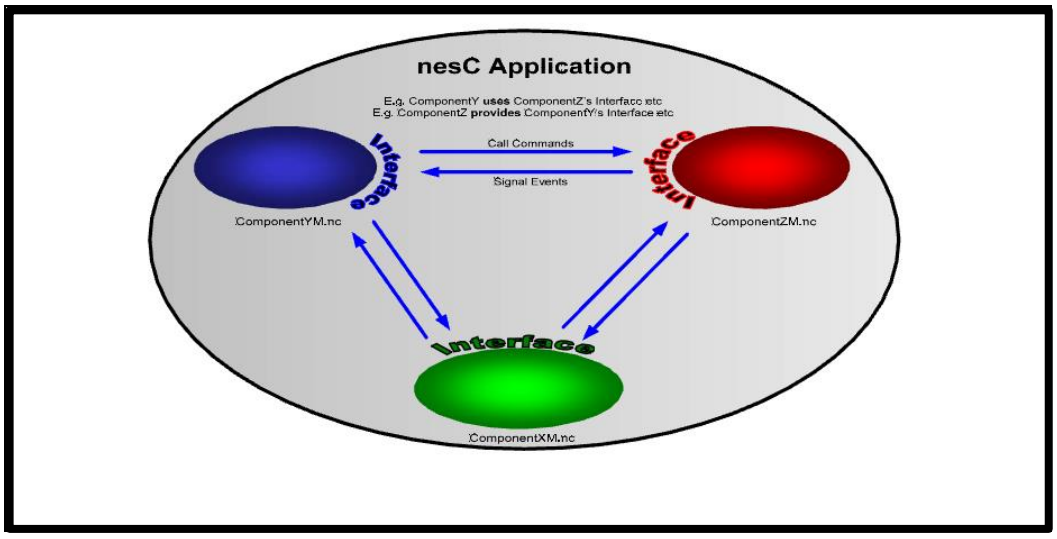

Fig.1.3 : Applications [2]

- **Composants**

Ce sont les blocs constructifs de l'application nesC, le composant fournit et utilise une interface bidirectionnelle bien définie ; d'une certaine manière les composantes nesC sont similaires à des objets de C++ ou java. Cependant contrairement aux objets de C++ ou java, qui se réfèrent à des fonctions et des variables dans un environnement global, les composants nesC utilisent un espace purement local. Cela signifie qu'en plus de la déclaration des fonctions implémentées, un composant doit également déclarer la fonction qu'il appelle. Chaque composant a un cahier de charge, un code bloc qui déclare les fonctions qu'il appelle, ou qu'il fournit.

Le nesC a deux types de composantes, les configurations (le composant fil) et les modules (les composants implémentés). Le rôle des configurations est de connecter les déclarations des différents composants, tandis que les modules définissent les fonctions et allouent les états.

- **Modules**

Les modules fournissent le code d'application, ils implémentent aussi une ou plusieurs interfaces.

- **Configurations**

Elles sont utilisées pour assembler les composants, connecter les interfaces utilisées par les composants aux interfaces fournies par d'autres, c'est ce qu'on appelle « câblage ».

- **Interfaces**

Elles sont bidirectionnelles et elles agissent comme le seul point d'accès au composant. Une interface déclare un ensemble de fonctions appelé commande, que l'interface du fournisseur doit implémenter, et un autre ensemble de fonctions appelé événement, que l'interface de l'utilisateur doit implémenter.

Pour qu'n composant, puisse appeler la commande dans une interface, il doit implémenter les événements dans la même interface. Un seul composant peut fournir ou utiliser des interfaces multiples.

- **Commandes**

Ce sont les fonctions que le fournisseur d'interface doit implémenter, elles sont appelées en utilisant le mot-clef « call ».

- **Evénements**

Les événements sont déclarés par le fournisseur d'interface, mais l'utilisateur de l'interface doit les implémenter en fonction de leur besoin. Les événements sont déclenchés/signalés via le mot-clef « signal ». [2]

1.2.4.2. Modèle d'accès simultané TinyOS

Le système d'exploitation TinyOS exécute un programme, en utilisant deux chemins, l'un contient les tâches, et l'autre contient le gestionnaire des événements du matériel.

Les tâches sont des fonctions dont l'exécution est définie, une fois programmée, elles s'exécutent jusqu'à la fin, et n'interrompent pas une autre tâche.

Le gestionnaire des événements du matériel, est exécuté en réponse à une interruption matérielle, et il s'exécute également jusqu'à la fin, mais il peut interrompre l'exécution d'une tâche ou d'un autre gestionnaire d'événement.

Parce que les capteurs ont un large éventail de capacité matérielle, TinyOS a des limites flexibles de software/hardware.

Dans le système embarqué en temps réel, les opérations du hardware sont plutôt séparées en phase que bloquées. Une caractéristique importante de la séparation des phases, c'est qu'elles sont bidirectionnelles, il y a un appel vers le bas pour lancer l'opération, et un appel vers le haut qui signifie que l'opération est terminée (fig. 1.4).

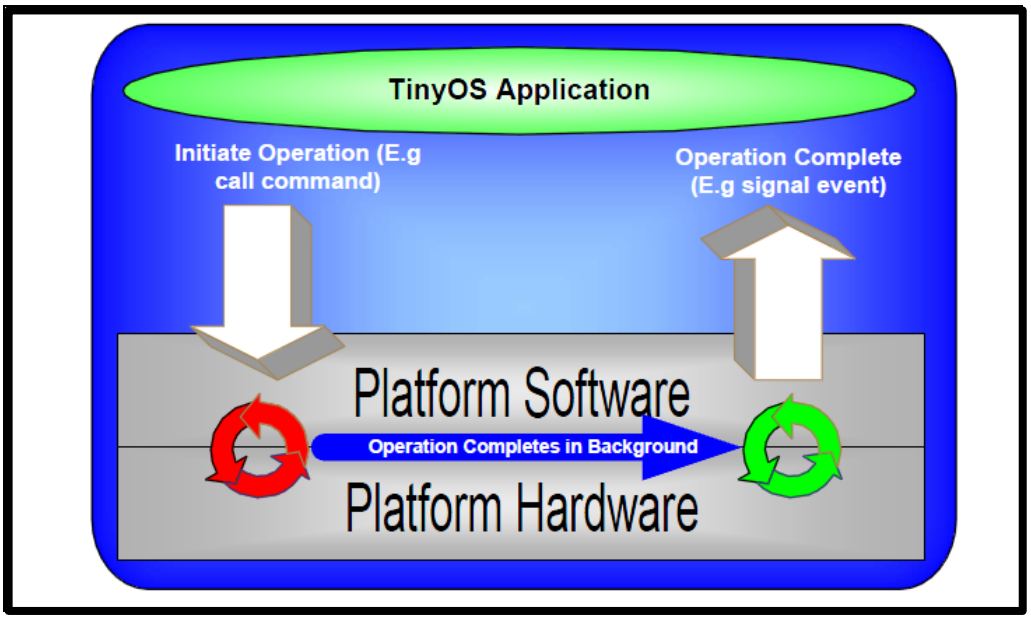

Fig.1.4 : Split-phase [2]

Le blocage en d'autres termes, se réfère à bloquer une application complètement, et attendre jusqu'à ce qu'une opération soit terminée ; par exemple pour obtenir une lecture du capteur avec un convertisseur analogique-numérique, le software écrit dans un certain registre de configuration pour démarrer un échantillon, lorsque la conversion numérique analogique est terminée, le hardware est soumis à une interruption, et le software lit la valeur de sortie du registre de données. En TinyOS, les opérations qui subissent une séparation de phase en hardware, le subissent aussi en software.

C'est-à-dire que plusieurs opérations, comme l'échantillonnage et la transmission des paquets subissent une séparation de phase.

- **Tâches**

Dans certains systèmes d'exploitation en temps réel, les tâches et les chemins peuvent dire la même chose, il y a seulement deux chemins d'exécution, et les tâches constituent l'un de ces

chemins. Les tâches sont non interruptibles, cela veut dire qu'uniquement une seule tâche s'exécute à tout moment, et TinyOS n'interrompe pas une tâche pour exécuter une autre. Quand une tâche est en cours d'exécution, aucune autre tâche ne s'exécute jusqu'à ce qu'elle soit terminée, cela signifie que les tâches s'exécutent indépendamment les unes par rapports aux autres. Ce qui nous donne comme avantage, de ne pas nous inquiéter de l'interférence des tâches entre elles ni de la corruption des données des autres.

Toutefois les tâches doivent être courtes. Si un composant a un long calcul à exécuter, il doit être décomposé en plusieurs tâches.

- **Gestionnaire des événements du matériel**

Les tâches permettent aux composants, d'émuler le comportement de phase-auxiliaire du hardware. Les techniques de deriven de ce genre sont courantes dans le software embraqué, où un temps de réponse rapide et le traitement d'événement est le noyau principal de l'efficacité du système embarqué en temps réel. Cependant ils ont une utilité beaucoup plus importante que les taches de bas niveau, ils fournissent aussi un mécanisme de gestion d'interruption dans le système. Comme les tâches s'exécutent indépendamment les unes par rapport aux autres, les codes qui s'exécutent seulement dans les tâches peuvent être assez simples, car il n'y a pas de risque qu'une autre tâche soit prise en charge, et modifie les données par inadvertance. C'est ce que fait exactement l'interruption, il interrompe de manière asynchrone, l'exécution courante et il commence à exécuter préventivement. Le problème fondamental de l'exécution préventive c'est qu'elle peut modifier l'état du calcul en cours, c'est ce qui peut entrainer que le système entre dans un état incohérent. En nesC, les fonctions qui peuvent être exécutées à titre préventif, à partir d'un contexte de travail extérieur sont étiquetées avec le mot de passe « async »;

Les commandes et les événements sont synchrones par défaut « sync ».

Les définitions d'interface précisent si les commandes et les événements sont synchrones ou asynchrones. Tous les gestionnaires d'interruption sont asynchrones.

Donc ils ne peuvent inclure aucune fonction synchrone dans leur graphe d'appel.

La seule façon pour que le gestionnaire d'interruption puisse exécuter une fonction synchrone est d'afficher une tâche. [2]

1.2.4.3. Configuration et interconnexion

Comme il a été mentionné précédemment, dans la partie de l'architecture de TinyOS, il y a deux types de composants en nesC « module » et « configuration », les modules fournissent le code de l'application et l'implémentation d'une ou plusieurs interfaces.

Cependant la configuration est utilisée pour assembler les composants, connecter les interfaces utilisées par les composants à des interfaces fournies par d'autres.

Chaque application de nesC est décrite par une configuration de niveau supérieur qui relie les composantes d'intérieur.

Pour qu'un module interagisse avec un autre, un ensemble de noms dans un composant, en général une interface, doit être mis en correspondance avec un ensemble de noms dans un autre composant. [2]

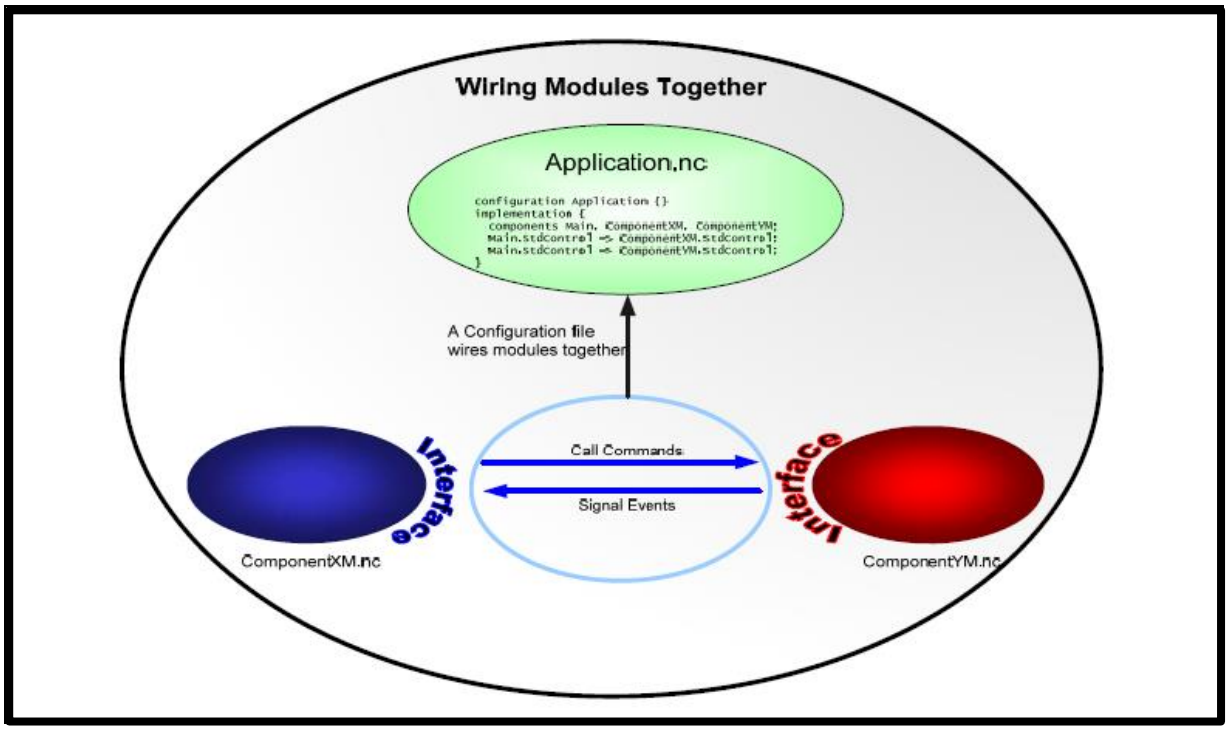

Fig.1.5 : Interconnexion [2]

3.1.CONCLUSION

On a introduit dans ce chapitre, la platefrome de shimmer, et les principales avantages, qui nous ont poussé à travailler avec cette plateforme, faible consommation d'énergie, poids Legé, et traitement en temps réel, un système très pratique et efficace pour le connected health.

2.1. INTRODUCTION

D'après les statistiques, la chute est la cinquième cause des plus fréquentes entrainant le décès des personnes âgées.

De nos jours, la détection de chute a fait l'objet de plusieurs travaux, et plusieurs méthodes ont été élaborées. Nous commençons à présenter dans ce chapitre les exigences de la conception d'un tel système, et quelques algorithmes de détection de chute.

2.2. LES EXIGENCES DE CONCEPTION D'UN SYSTEME DE LA DETECTION DE CHUTE

Pour que le système soit fiable, il doit répondre à certaines exigences que l'on peut résumer comme suit :

1. Il doit détecter tous les impacts, qui sont inférieurs à un seuil prédéfini.
2. Il doit être capable de confirmer une condition d'alarme.
3. Il doit traiter les données mesurées pour confirmer une véritable chute.
4. Il doit activer l'alarme dans un bref délai, dans n'importe qu'elle pièce de la maison.
5. Il doit être robuste et nécessiter peu d'entretien.
6. Il doit être identifié d'une façon unique, c'est-à-dire qu'il doit être perçu.
7. il ne doit pas, porter une injure ou un dérangement lors de l'événement de la chute.
8. il doit être ajustable pour répondre aux besoins de l'utilisateur.
9. il doit être facile à utiliser sans formation préalable.
10. il doit être capable d'être testé facilement. [3]

2.3. OU PLACER LES CAPTEURS ?

En général, quatre emplacements du capteur sur le corps sont choisis comme indiqué sur la figure (2.1).

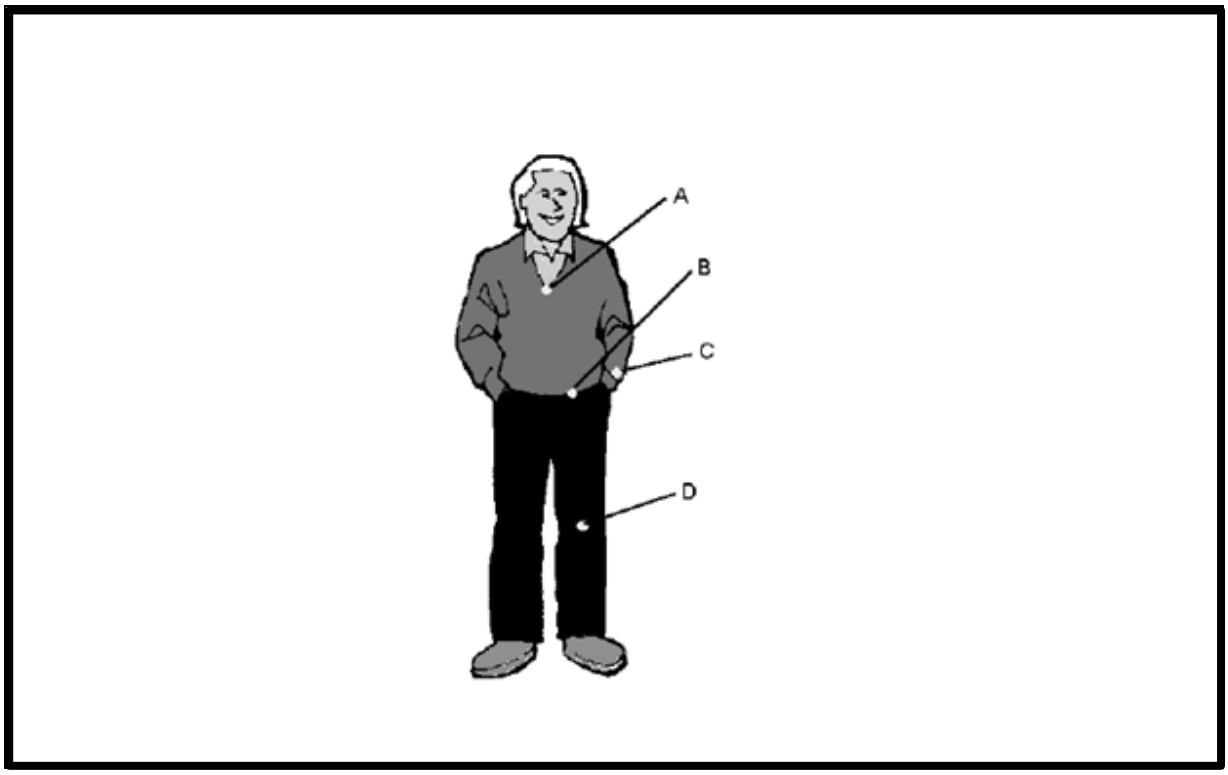

Fig.2.1 : Emplacements du capteur [3]

On teste pour chaque emplacement, cinq types de chute comme indiqué ci-après :

1. chute vers l'avant, les genoux rigides.
2. effondrement vers l'avant, les genoux flexibles.
3. chute en arrière, les genoux rigides.
4. chute sur le côté, les genoux rigides.
5. tomber en avant, les genoux souples

Puis en mesurant le signal maximum, généré pour chaque type de chute, pour les 4 emplacements, on remarque que le signal au niveau du genou est inférieur de 30% par rapport à celui de la poitrine ou de la taille, mais il ya une grande cohérence entre les trois emplacements. Par contre, pour l'emplacement au niveau dun poignet, le signal varie laregemen.

Par conséquent, on peut dire que le capteur peut être porté partout sur le corps sauf, au niveau du bras ; le genou a été éliminé en raison des dommages qui peuvent être causés au corps humain. [3]

2.4. ALGORITHMES DE DETECION DE CHUTE

On distingue essentiellement trois types d'algorithmes de détection de la chute.

2.4.1 Algorithme1, utilisant un système acoustique

En septembre 2009, Yaniv Zigel, Dima Litvak, et Israel Gannot [4], ont developpé un nouveau système pour la détection de la chute, basant sur l'analyse des vibrations du sol, et les ondes acoustiques.

- **Le concept de la détection de la chute par les vibrations et les sons**

Le système proposé est basé sur, la détection des vibrations et des sons des signaux, à partir d'un accéléromètre et un microphone. La partie hardware contient essentiellement

-Un accéléromètre Crossbow CXL02LF1Z

-Microphone MS_3100W

-Les signaux acquises sont transmissent à une carte d'acquisition NI USB-6210(national instruments ,TX,USA),Qui échantillonne le signal a 16khz et il les transmit au PC.

Les signaux numériques provenant d'un capteur, sont collectés par NI LABview software.

Et analysés par MATLAB software,

L'algorithme est basé sur deux étapes, la détection et la classification

L'étape, de la détection utilise la caractéristique à partir de l'énergie des vibrations, et utilise les sons des signaux pour localiser une suspicieuse chute.

L'étape de la classification utilise les classifications, pour distinguer entre une chute et un événement.

L'algorithme est :

Il contient deux phases, phase d'entrainement et phase de test,

Dans les deux phases on utilise les signaux et les vibrations comme entrées

La phase d'entrainement consiste en : détection de l'événement, segmentation, l'extraction des caractéristique, la sélection des caractéristique, et le modèle de l'estimation,

1. Détection et segmentation : la but principale est de détecter une vibration d'événement dans la vibration du signal, et la segmenter, on peut évaluer ça en calculant l'énergie du signal, en calculant l'énergie maximale, on trouve l'indice de l'événement n, ensuite on calcul t_e=L*n, tel que L=20ms, et c'est la largeur de trame, , en trouvant t_e, l'algorithme de la segmentation extrait l'événement.

2. La segmentation d'événement a été évaluée par un algorithme qui identifie, les limites de l'événement :

On calcul d'abord la valeur la plus importante d'énergie e_{hist}. D'âpres l'histogramme du signal de la vibration enregistrée.

La valeur de E_{th} est calculé par l'équation (2.1)

$$E_{th}=e_{hist}(1+C_1) \quad (2.1)$$

tel que $C_1(0.2)$ est une valeur empirique

Apres avoir calculé le E_{th}, on compare chaque valeur échantillonné, et on la compare au seuil, si elle est inferieur donc c'est l'indice de la fin de l'événement, et $T_e=n*L$.

De même pour le début, ainsi on détermine l'intervalle de l'événement.

3. L'extraction de la caractéristique : il existe deux types de caractéristique,

-Caractéristique temporelle : les caractéristique qui sont extraite de la vibration des éléments (énergie et la largeur du signal de vibration, et le signal sonore).

-Caractéristique fréquentielle : (SRS,MFCC) la réponse spectrale de choc, est extraite de la vibration , et les coefficients MFCC sont extraites du signal sonore .

Concernant SRS : on modélise la chute de l'être humain par l'ensemble masse-ressort.SRS est l'accélération maximale de la réponse des différents signaux a degré de liberté différents et fréquence différentes.

Concernant MFCC : Elle approxime, le système auditif humain plus que la FFT, elle est utilisée largement dans la reconnaissance de la parole.

On divise l'événement du signal en sous fenêtre de longueur de 0.03s.

Et on calcule les MFCC coefficients, qui fournit 13 caractéristique pour chaque fenêtre, la première caractéristique est l'énergie.

4. La sélection de la caractéristique :

Elle réduit le prix du modèle de la reconnaissance et elle donne une classification meilleure, parmi les 13 caractéristiques extraites, on choisit les caractéristiques les plus performantes pour la détection.

5. Modèle et l'estimateur de classification :

On utilise le bayes classification, qui a la probabilité la plus importante ?

Apres le calcul de maximum vraisemblance le classifier renvoi si c'est une chute ou pas. [4]

- **Inconvénients:** il ne peut pas étre appliqué, en dehors de la maison.

2.4.2 Détection de la chute basée sur la vidéo surveillance

Il existe plusieurs méthodes qui sont basé sur la vidéo surveillance, qui se basent essentiellement sur l'extraction de la caractéristique dynamique des personnes comme vélocité et l'intensité du gradient. Cette méthode utilise, la caractéristique statique comme l'inclinaison du patient.

Le système proposé, calcul à partir de l'image extraite le rapport de l'aspect qui est donné par l'équation (2.1) :

$$\text{Rapport de l'aspect} = \frac{\text{largeur de la personne}}{\text{hauteur de la personne}} \qquad (2.2)$$

Le rapport de l'aspect : est défini comme le rappeur de la largeur de la personne sur la hauteur de la personne. C'est la caractéristique la plus importante pour distinguer si la personne est debout ou allongé.

-Si le rapport est inférieure à 1, sa hauteur est supérieur à sa largeur donc le patient est debout.

-Si le rapport est supérieur 1, sa hauteur est inférieure à sa largeur : le patient est allongé

Mais justement ce rapport n'est pas assez suffisant pour déterminer si c'est une chute ou pas

Car Si la ration=1, cela veut dire que le patient est allongé mais plié, dans ce cas les paramètres d'inclinaison sont nécessaire pour déterminer la chute, pourtant le rapport peut nous indiquer quand même si le patient est debout ou non.

L'angle de l'inclinaison :

-Si l'angle est inférieur à 45° la personne est debout.

-Si l'angle est supérieur à 45° la personne est allongée.

Donc on peut résumer le processus comme suit :

-Capturer l'image de la vidéo stationnaire de camera.

-Soustraire l'image actuelle de la vidéo à partir de modèle d'arrière-plan.

-Calcul du rapport de l'aspect.

-Calcul de centre de gravité.

-Calcul de l'angle de l'inclinaison.

-Si les conditions sont vérifiées, alors détection de chute.

-Une alarme est activée.

- **Inconvénients** : Ya plusieurs solutions ces dernière années, qui sont basé sur l'analyse et le traitement d'image des mouvements des personnes en temps réel, son inconvénient, est la vie privé, et la difficulté de balayer toute la zone de la maison.

2.4.3 Algorithme3, utilisant le capteur accéléromètre

La principale utilisation de capteur accéléromètre, pour la détection de chute, se résume dans la vérification de changement d'amplitude, mais dans un intervalle qui a été déterminé expérimentalement par une valeur inférieur à 500ms, puis on pourra rajouter un dernier module pour que le système soit assez performant, est la vérification du changement de l'inclinaison du patient.

Pour ce système plusieurs méthodes ont été proposées, on présente ici quelque méthode

- **Algorithme 1**

Dans cet algorithme, Le capteur sera placés au niveau de l'oreille, 3 accélérations seront mesurées selon les 3 axes XYZ, les signaux sont mesurés selon une fréquence de 200 Hz, et seront stockés sur une carte mémoire. Ces signaux sont filtrés par un filtre passe-bas, avec une fréquence de coupure de 80 Hz, lié à la fréquence d'échantillonnage de 200 Hz.

Une chute peut être détectée par le déclenchement de 3 seuils :

Seuil (1) quand la somme des vecteurs d'accélération, sur le plan XY dépasse 2g

Seuil(2) quand la somme des vecteurs de vitesse sera supérieure à 0.7 ms^{-1}

Seuil(3) quand la somme des vecteurs d'accélération sera supérieure à 6g

Pour le premier seuil, la direction Z n'a pas été incluse, car les valeurs d'accélération élevée dans cette direction, ont été mesurées lors de la course, ou au moment de s'assoir.

Pour le 2ème seuil, il a été remarqué qu'après 500 ms, du contact avec le sol, le corps du patient est au repos, c'est à dire que la vitesse est nulle. La valeur 500 ms a été déduite empiriquement, car dans toutes les chutes après 500 ms, il n'y avait aucun changement significatif de la valeur d'accélération.

On calcule la vitesse par l'intégration de l'accélération entre t_0 et t_1 qui représente respectivement 500ms après le contact et le contact initial.

La raison pour laquelle on considère la vitesse et non l'accélération est que, si on prend le cas du patient avec la tête sur son lit, des valeurs de l'accélération peuvent déclencher une fausse alarme, mais la vitesse est nulle. Les valeurs 0.7 ms^{-1} et 2g ont été déterminées empiriquement.

Pour le 3ème seuil, la somme des vecteurs supérieure à 6g, cette valeur a été fixée car elle n'a été détectée dans aucune des activités quotidiennes du patient, par contre elle a été mesurée lors de la simulation des chutes. En utilisant cette approche, toutes les chutes testées ont été identifiées.

L'avantage de fixer le capteur derrière l'oreille, est de pouvoir le porter la nuit. [6]

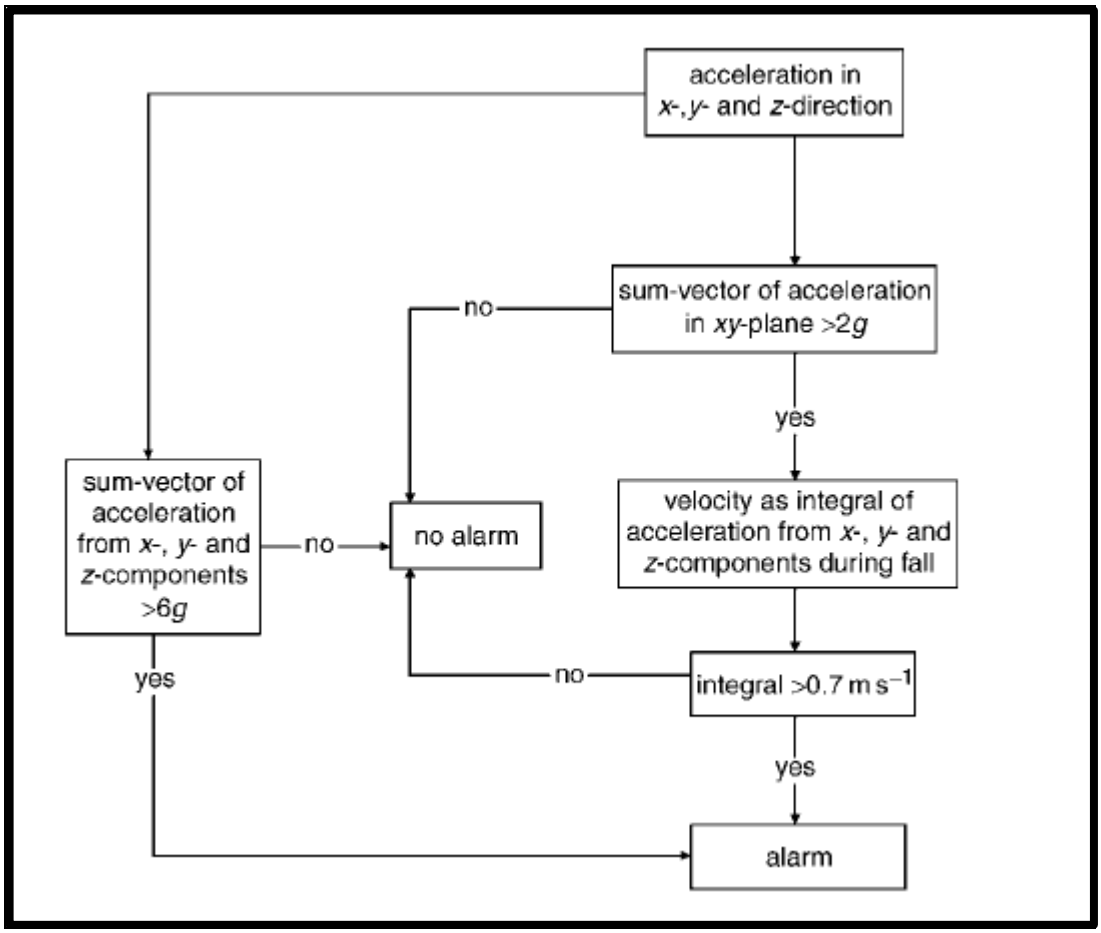

Fig.2.2 : Organigramme de l'algorithme 1 [6]

- **Inconvénients:** Les meilleurs résultats données c 'est quand le capteur est porté sur le centre de gravité, puis de préférence, les dispositifs électroniques doivent être porter loin de cerveau.

- **Algorithme 2 : vêtement intelligent**

En 2008, Gaetano Anania, Alessandro Tognetti, Nicola Carbonaro, Mario Tesconi, Fabrizio Cutolo, Giuseppe Zupone, Danilo De Rossi [7], ont développé des vêtements qui contiennent un réseau de capteurs pour la surveillance de la santé des personnes, en temps réel.

Le système doit être porté, plusieurs heures sans recharger la batterie. Donc, l'utilisation d'une électronique de faible puissance et une mémoire limitée est obligatoire. De plus, il faut réduire la quantité des données en temps réel sur une carte de traitement. Pour cela, la principale raison pour développer un nouvel algorithme peut être résumé en 4 points :

1. mise en ligne de traitement des données.

2. temps de calcul réduit, et la possibilité de l'implémenter, dans une carte de faible consommation, et une carte mémoire limitée.

3. fiabilité élevée, même dans des activités rapides (courir, sauter…)

4. Aucune formation n'est nécessaire, c'est important vue qu'il est recommandé pour les personnes âgées.

Dans ce dispositif, un accéléromètre 3D et une électronique ont été intégrés dans une veste d'un volontaire dans la partie supérieure du torse. Comme le montre la figure 2.3.

Fig.2.3 : Vêtement intelligent [7]

Nouveauté : un filtre de Kalman a été conçu de manière à séparer la composante du signal due à la gravité, de celle due à l'accélération du système. [7]

Description de la partie hardware :

1. Un tri-accéléromètre (AnalogDevice ADXL 330),

2. un microprocesseur de faible puissance pour la conversion A/D, et l'élaboration des signaux en temps réel (Texas instruments MSP 430F149), les données du capteur sont échantillonnées à 100 Hz.

Le capteur est intégré dans la zone BAN (body area network), à travers un bus RS485, et les alarmes de la chute sont élaborées en temps réel, et elles sont envoyées à l'unité centrale .

3. un 2ème module, a été réalisé, ayant les mêmes composantes, en lui rajoutant un système sans fil Bluetooth, il a été utilisé pour tester les performances du système.

L'algorithme est conçu de cette manière : le signal va passer par un filtre de kalman, pour séparer le signal d'accélération du signal de la gravité. La procédure consiste à comparer la valeur de l'inclinaison du patient avec une valeur seuil, puis comparer la vitesse de la chute à une autre valeur seuil et enfin comparer le delta T à un seuil, si les 3 conditions sont vérifiées, une chute est déclarée, ces 3 seuils ont été fixés expérimentalement.

- **Inconvénients:** porter toujours une veste, n'est pas confortable, en plus Elle n'est pas approprié pour le climat chaud.

- **Algorithme 3 : Système de détection de chute en utilisant un baromètre et un accéléromètre**

En 2009, un groupe d'étudiants [8], ont effectué des modifications au système de détection de la chute basé sur l'accéléromètre 3D. Le but de ce travail était d'utiliser un capteur de pression barométrique, pour évaluer les performances de l'algorithme de la détection de chute.

Le signal obtenu à partir du capteur de pression barométrique et de l'accéléromètre, est analysé par un traitement de signal pour distinguer si c'est une chute ou une activité quotidienne du patient.

Le système étudié contient :

-Un accéléromètre (MMA7260, Freescale), avec une fréquence d'échantillonnage de 40Hz.

-Un capteur de pression d'air atmosphérique (SCP1000, VTI Technologies), la fréquence d'échantillonnage est de 1.8 Hz.

-Un Microprocesseur (MSP430F149, Texas Instruments).

-Un module Bluetooth (WML_C30AH, Mitsumi).

- Une batterie rechargeable Li-Pol.

Pour développer un algorithme de détection de chute, on fait ce que l'on appelle une extraction des caractéristiques des capteurs (accéléromètre et capteur de pression). On effectue une série de traitement sur le signal qui a pour but de, premièrement approximer l'accélération de la pesanteur du compassant, deuxièmement soustraire cette dernière du signal original, pour donner une estimation de l'accélération du corps, cette série de traitement se fait par des filtres (filtre passe-bas, filtre médian).

L'algorithme calcule essentiellement trois facteurs :

-Une des plus importantes caractéristiques que l'algorithme calcule est le pic d'accélération : SVM, il donne une estimation sur l'intensité, qui se dérive de la composante de l'accélération du corps, ce pic peut être calculé par la formule (2.3.)

$$\text{SMV}=\sqrt{x_i^2 + y^{2i} + z_i^2} \tag{2.3}$$

-L'algorithme calcule aussi le SMA, c'est un facteur qui nous permet de distinguer, la position de travail de la position de repos.

On utilise pour cela la formule (2.4)

$$\text{SMA}=\text{T}\sum_{i=j}^{i=j+\frac{1}{T}} x_i + y_i + z_i \tag{2.4}$$

On prend un intervalle de 1s.

-L'orientation : elle fournit des informations sur l'inclinaison du patient, elle est définie comme l'angle entre le vecteur de gravitation et l'accélération du patient mesurée le long de l'axe Z.

$$\Theta=\cos^{-1}(Z/G) \tag{2.5}$$

On définit ainsi qu'un angle entre 0° et 20°, signifie que le patient est debout tandis que les angles supérieurs à 20° spécifient que le patient est non-debout.

-La pression différentielle : le signal de sortie du filtre passe-bas, a été filtré de nouveau en utilisant un filtre de butterworth avec une fréquence de coupure de 0.1 Hz, et échantillonné en utilisant une interpolation linéaire. Le signal résultant a été utilisé pour calculer l'écart des paramètres de la pression.

En fusionnant l'algorithme précédent, on est arrivé à l'algorithme suivant :

Une chute est détectée : Si le capteur indique un pic anormal de l'accélération, on mesure la différence de la pression sur un intervalle de 3s, si la différence de la pression est supérieure au seuil, on mesure l'angle, s'il est supérieur à 20°, on mesure de nouveau la différence de pression pendant un intervalle de 60s. Le système est capable de reconnaitre si le patient a réussi à remonter, durant les 60s. [8]

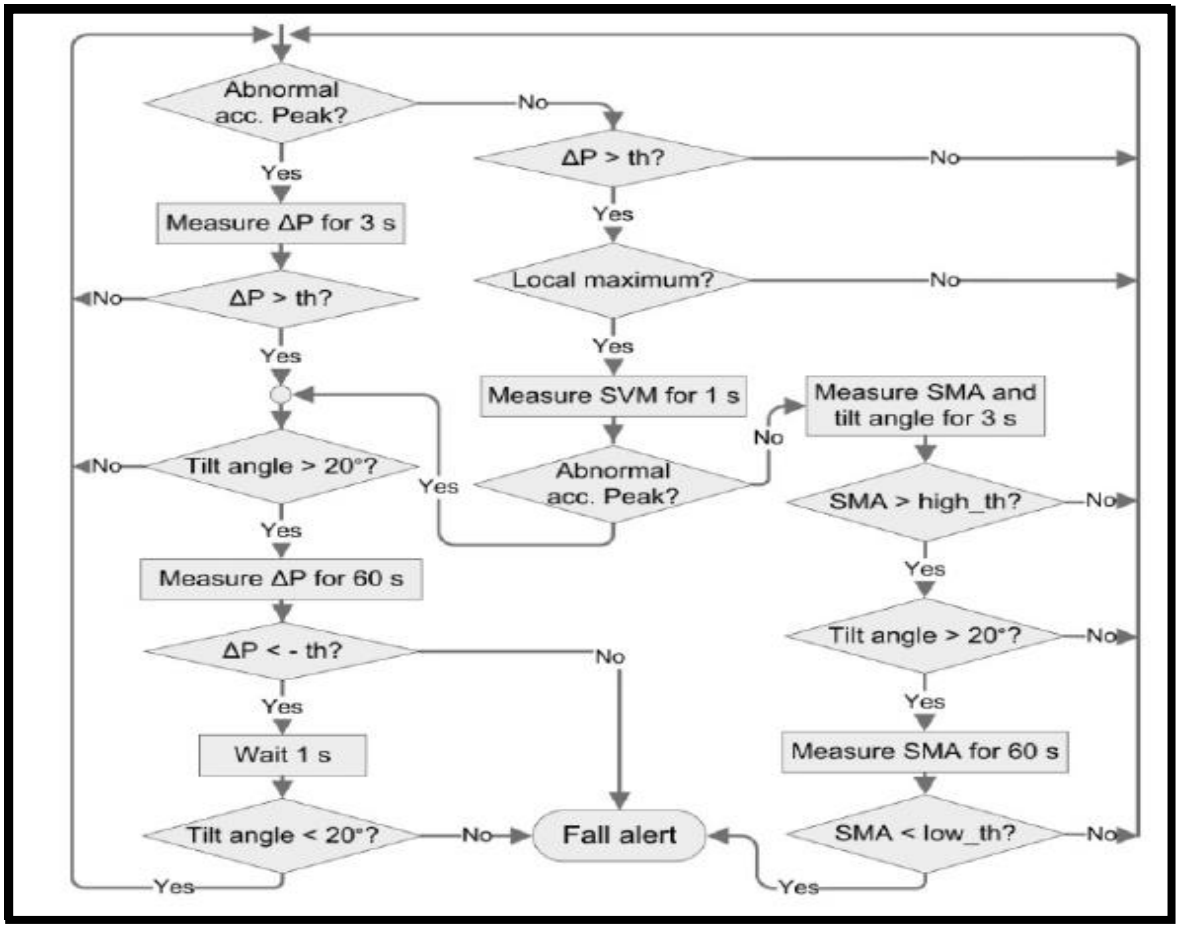

Fig.2.4 : Organigramme d'algorithme 3 [8]

- **Inconvénients:** un algorithme un peu complexe.

2.5. Conclusion

On a vu, dans ce chapitre les différents algorithmes, utilisés pour détecter les chutes, et pour chaque algorithme, les difficultés rencontrées, ou plutôt leurs inconvénients, dans la suite on présentera avec plus des détails, le système de détection de chute par la shimmer.

3.1. INTRODUTCION

On a vu dans le chapitre précédent, les différents algorithmes utilisés pour la détection de chute, ainsi que leurs inconvénients, le coût du système, la fiabilité, le temps réel, ce sont les principales raisons pour lesquelles, on a opté pour l'utilisation de la shimmer device, pour notre système. Dans ce chapitre, on explique les différents modules de notre algorithme, puis on expose les différentes étapes suivies pour simuler notre système sur MATLAB.

3.2. DESCRIPTION DE L'ALGORITHME IMPLEMENTE

Comme il a été mentionné dans le premier chapitre, une des applications importantes de shimmer device est la détection des chutes.

La chute peut être décrite par le changement rapide de la position verticale vers la position horizontale, et le mouvement contrôlé comme allongé doit être distingué et non considéré comme une chute.

L'architecture proposée est constitué de 3 parties :

3.2.1. Module de prétraitement

Il comprend le capteur accéléromètre 3D et un filtre passe bas.

L'accéléromètre 3D nous fournit des informations sur la manière dont la gravité est répartie sur les 3 axes fournissant des informations sur la façon dont le dispositif est orienté.

Dans ce module de prétraitement, on appliquera un traitement sur les trois signaux provenant de shimmer device afin d'extraire la composante continue et supprimer les signaux correspond à l'activité du patient, ceci pourra être réalisé par un filtre passe bas de butterworth du premier ordre de fréquence de coupure 0.25 Hz, et qui est défini par la formule (3.1)

$$Y_n=A*(X_n+X_{n-1})+B*Y_{n-1} \qquad (3.1)$$

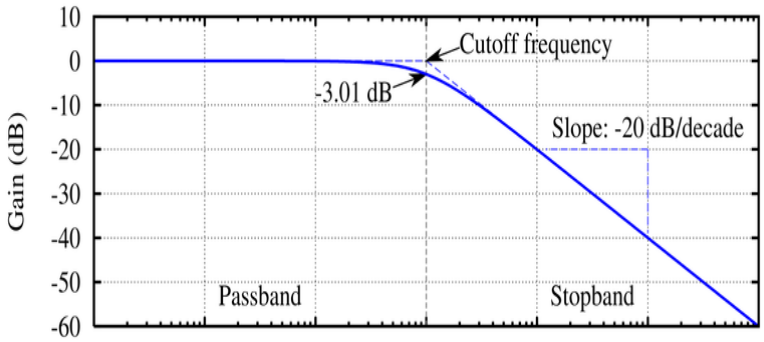

Fig.3.1 : Filtre butterworth premier ordre

Tel que A=0.0155 et B=0.9687

3.2.2. Module de traitement

Ce deuxième Module, qui est le plus important, va calculer deux critères, pour détecter la chute, qui sont l'amplitude et l'orientation.

L'orientation se calculera, à la sortie du filtre Butterworth, par le produit scalaire entre le vecteur échantillonné et le vecteur de référence qui correspondant à la position verticale du patient. La position verticale du patient, peut être définie comme la position dans laquelle le patient aura au minimum sa partie supérieure alignée avec son centre de gravité.

Par un filtrage passe-bas du signal de sortie de l'accéléromètre 3D, la contribution de l'activité du patient au signal d'accéléromètre est largement supprimée de telle sorte que l'amplitude de sortie est proche de 1g.

Le vecteur de référence a toujours une grandeur proche de1g. Donc une formule simplifiée pour calculer l'inclinaison est montrée dans l'équation (3.2)

$$\mathrm{Cos}^2(\theta_{cur}) = \frac{x_0\, x_{ref} + y_0\, y_{ref} + z_0 z_{ref}}{x_0^2 + y_0^2 + z_0^2} \qquad (3.2)$$

Puis on pourra calculer l'angle de la position courante par l'équation (3.3)

$$\Theta_{cur} = \arccos(\theta_{cur}) \qquad (3.3)$$

$(x_{ref}, y_{ref}, z_{ref})$ sont les valeurs de l'accélération sur les trois axes, du vecteur de référence.

(x_0, y_0, z_0) sont les valeurs de l'accélération sur les trois axes, du vecteur échantillonné.

Θ_{cur} c'est l'inclainaison du patient courante par rapport à la verticale.

Les seuils ont été déterminés expérimentalement, on pourra les résumer comme suit :

Le seuil de la position verticale est défini comme l'angle inférieur à 30°.

Le seuil de la position non verticale est défini comme l'angle supérieur à 60°.

Le seuil de la chute est défini approximativement de 45°.

Le seuil de la position inclinée est défini par un angle supérieur à 75°.

L'amplitude de l'accélération se calculera, à la sortie de l'accéléromètre par la formule de la norme euclidienne montré dans l'équation (3.4)

$$V = \sqrt{(X_0^2 + Y_0^2 + Z_0^2)} \qquad (3.4)$$

Si le patient est en position verticale, il aura comme amplitude d'accélération 1g,
Si le patient a subi une chute, il aura une accélération d'amplitude supérieure ou égale à 3g, cette valeur a été déterminée empiriquement.

3.2.3. Module de vérification

Une chute a été définie comme le changement rapide de la position verticale vers la position horizontale, donc en exploitant cette définition, on pourra vérifier si c'est une chute réelle ou pas, pour cela on doit vérifier le changement rapide de la valeur de l'amplitude et de l'orientation du patient.
A chaque 500ms, on calcule la valeur de l'amplitude de l'accélération, et on compare, deux valeurs successives, si l'algorithme détecte, un changement seuil, entre les deux valeurs, la chute sera confirmée.
Remarque : on voit bien que pour détecter une chute, il suffira, de vérifier le dépassement du seuil de l'amplitude de l'accélération, dans un intervalle de temps inférieur ou égal à 500 ms. La vérification de l'inclinaison du patient, peut être rajoutée pour rendre le système plus performant.

3.3. SIMULATION SUR MATLAB

On utilisera pour ce travail le driver de l'équipe de recherche de la shimmer sur MATLAB.

3.3.1. Présentation du driver

Le driver de shimmer sur matlab, contient 3 principales fonctions :

- Plotandwriteexample.m : cette fonction nous trace le graphe des différents capteurs intégrés dans le shimmer, on ne doit y mettre que le numéro de port où le shimmer a été connecté avec le PC ainsi que la durée de l'acquisition des signaux en temps réel.
- ShimmerHandleClass : cette fonction, contient plusieurs sous-fonctions, qui ont pour but essentiel l'ajustement de la fréquence d'échantillonnage et l'acquisition des données.
- Twoshimmerexamplelegacy : cette fonction a pour but de simuler, deux shimmer simultanément.

Pour pouvoir, utiliser ce driver, le terminal realterm doit être installé. Realterm est un terminal spécialement conçu pour la capture, le contrôle et le débogage binaire et des autres flux de données difficiles.

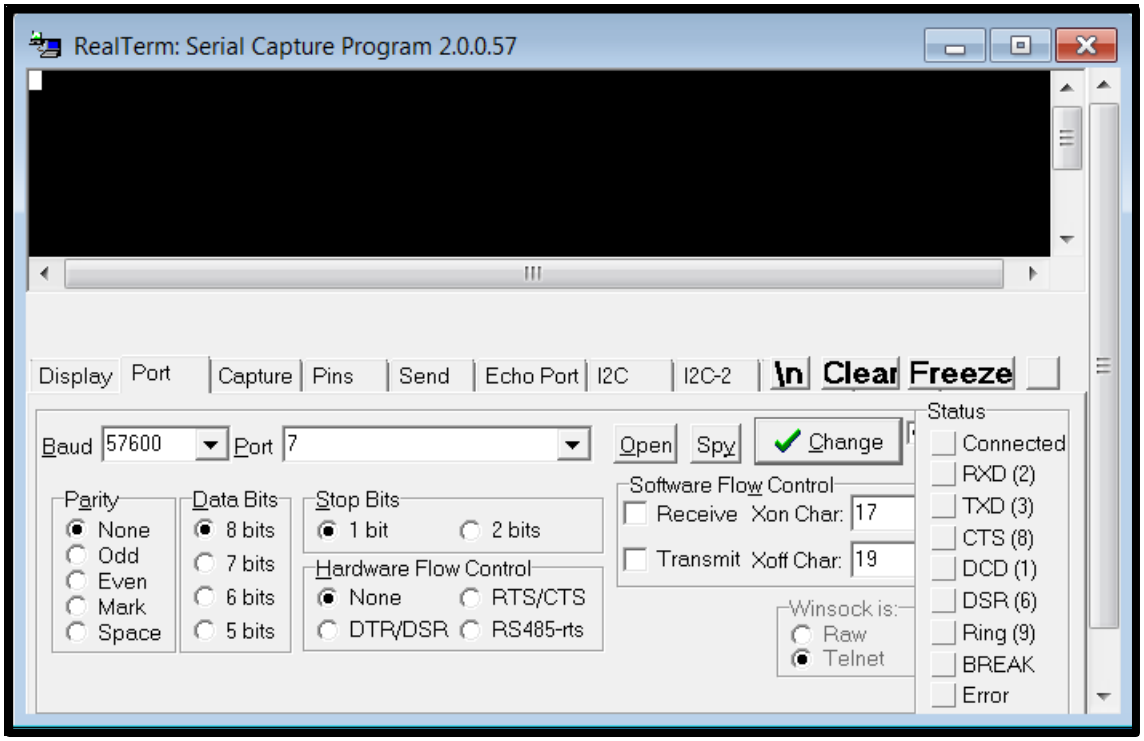

Fig.3.2 : le terminal realterm

Grâce à ce terminal, on peut connecter le shimmer et acquérir les datas, ensuite à son tour, il va envoyer les données à matlab.

3.3.2. Résultats

On a simulé notre algorithme pour deux positions du patient :

- position verticale :

  Les figures (3.2) et (3.3) montrent le tracé des résultats du 2ème module (traitement),

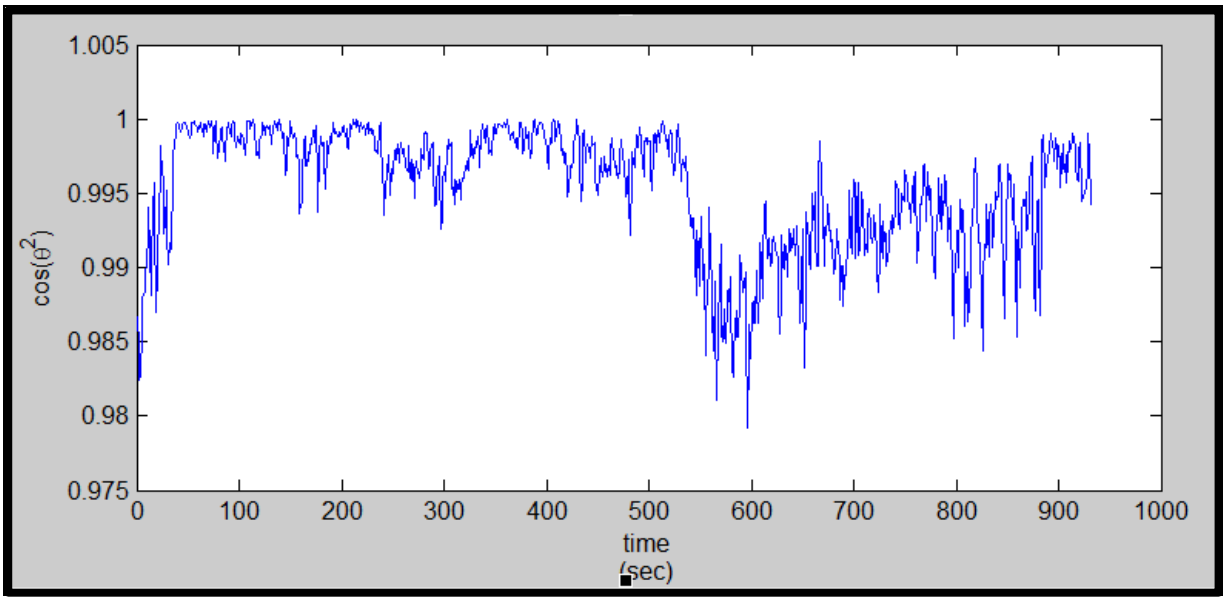

Fig.3.3 : Angle de la position verticale

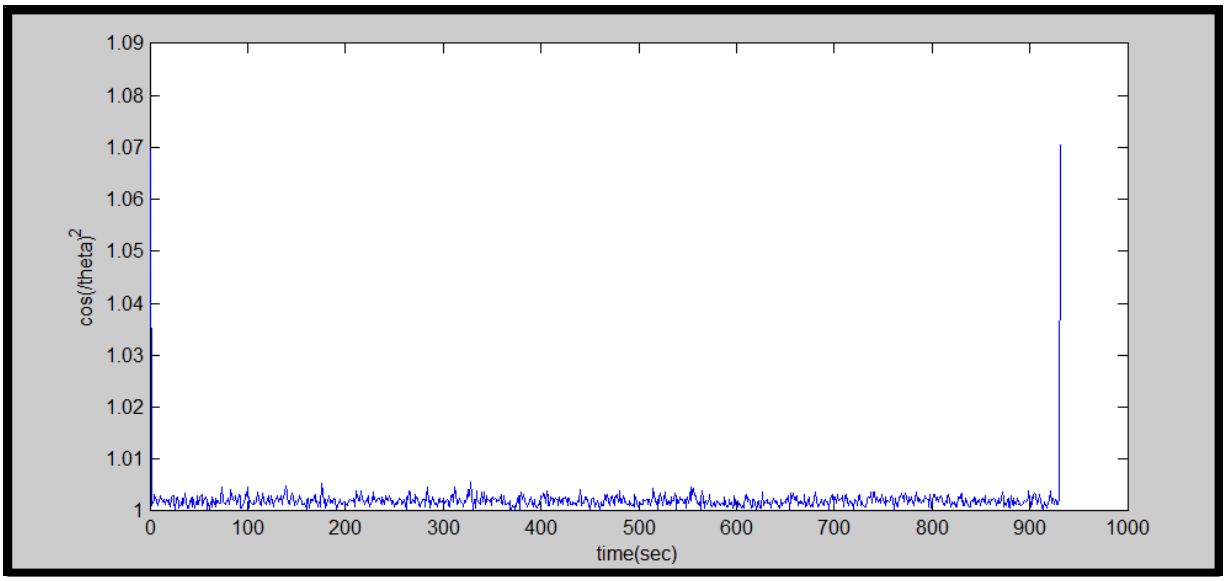

Fig.3.4 : Amplitude de l'accélération de la position verticale

- position durant la chute.

  De la même manière les deux figures (3.4)et (3.5) montrent les résultats de la simulation du module de traitement.

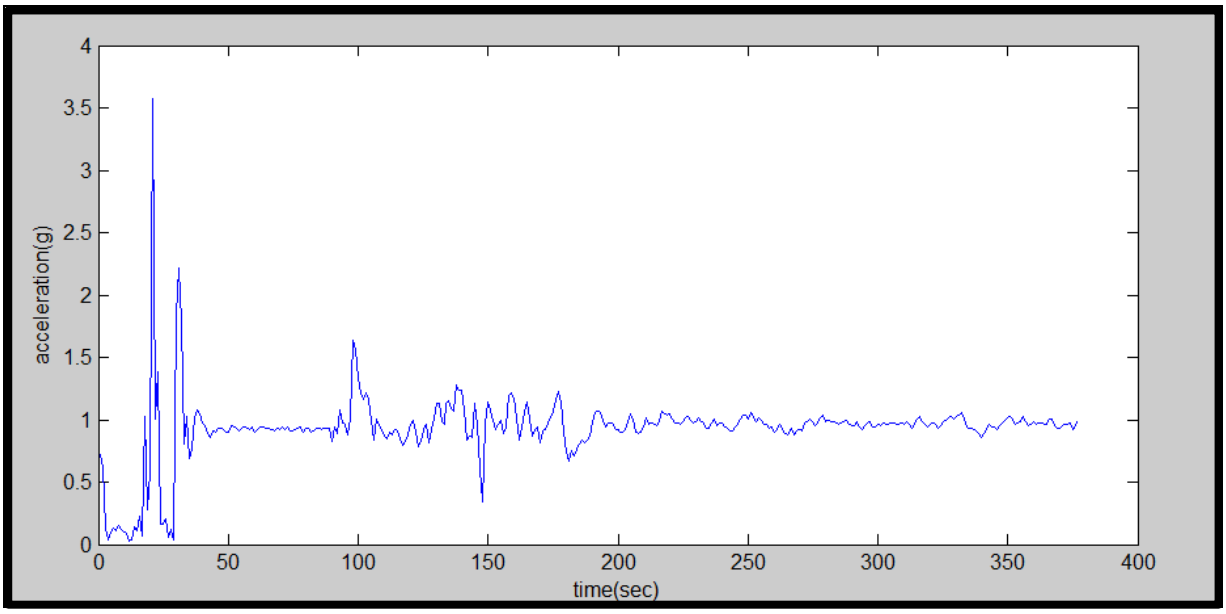

Fig.3.5 : Amplitude de l'accélération pendant la chute

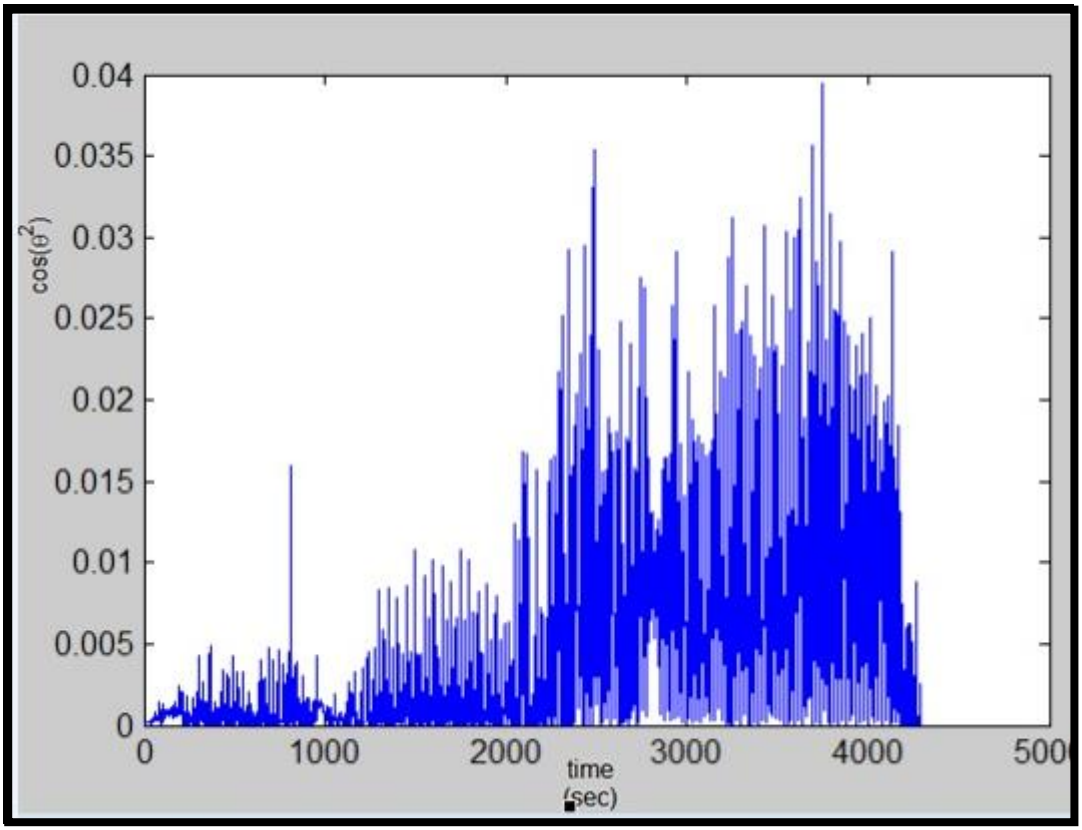

Fig.3.6 : Angle durant la chute

3.3.3. Interprétation des résultats

- On remarque dans la figure (3.3) que le $\cos(\theta)^2$ est approximativement égal à 1, ce qui est évident, car le produit scalaire des deux vecteurs correspondant à la position verticale doit être égal à 1 puisqu'ils sont alignés et dans la même direction.
- Pour la figure (3.4), on remarque que l'amplitude de l'accélération est à peu près 1g, ce qui est logique, car la force de 1 g exercée sur un corps reposant à la surface de la Terre est provoquée verticalement par la réaction du support en empêchant l'objet de tomber en chute libre.
- La figure (3.5) montre la variation de l'amplitude de l'accélération pendant la chute ; on remarque que l'amplitude de l'accélération peut dépasser 3g , cette valeur ayant été trouvée expérimentalement comme il a été mentionné précédemment.
- La figure (3.6) trace l'inclinaison du patient par rapport à la verticale; on remarque que le cosinus a une valeur nulle, ce qui confirme la chute car le produit scalaire entre le vecteur de référence qui est la position verticale et le vecteur de l'accélération échantillonnée doit être égal à zéro.

  Donc en analysant les figures (3.5) et (3.6), on peut conclure qu'il y a une chute parce que les trois critères de décision d'existence d'une chute sont confirmés.

3.4. CONCLUSION

Les résultats obtenus sur MATLAB, montrent bien que l'utilisation de shimmer device, est efficace, pour la détection de chute, pour le nombre tests effectués (10), la simulation donne une efficacité (taux de détection) de 90%.

4.1. INTRODUCTION

Android, est de nos jours, la technologie la plus répandue, plusieurs programmeurs utilisent Android pour développer leurs applications. Les applications les plus téléchargées sont celles développées sous Android, c'est pour cette raison que l'on a choisi d'utiliser un smart phone contenant Android comme système d'exploitation pour réaliser notre système. Nous allons, dans ce chapitre, donner une vue générale sur l'histoire, l'architecture, et les composants d'Android.

4.2. PRESENTATION D'ANDROID

Android : prononcé *Androïd*, est un système d'exploitation open source utilisant le noyau Linux, pour Smartphones, tablettes tactiles, PDA et terminaux mobiles conçu par Android. D'autres types d'appareils possédant ce système d'exploitation existent, par exemple des téléviseurs, des radioréveils, des autoradios et même des voitures.

4.3. CREATION D'ANDROID

Android est le nom d'une PME américaine créée en 2003, puis rachetée par Google en 2005.
L'objectif d'Android était de développer un système d'exploitation mobile plus intelligent qui ne se contente pas uniquement de permettre d'envoyer des SMS et transmettre des appels, mais qui devait permettre à l'utilisateur d'interagir avec son environnement (notamment avec son emplacement géographique).
Il a été annoncé officiellement le 5 novembre 2007 ; c'est le produit d'un consortium " Open Handset Alliance " qui a été créé en novembre 2007 et qui compte aujourd'hui 84 entreprises membres.
En janvier 2007, Apple dévoilait l'iPhone, un téléphone tout simplement révolutionnaire pour l'époque. L'annonce est un désastre pour les autres constructeurs, qui doivent s'aligner sur cette nouvelle concurrence. Le problème étant que pour atteindre le niveau d'iOS (iPhone OS), il aurait fallu des années de recherche et développement à chaque constructeur.
C'est pourquoi en novembre 2007 l'Open Handset Alliance a été créée, et qui comptait à sa création 35 entreprises évoluant dans l'univers du mobile dont Google. Cette alliance a pour but de développer un système *open source* (c'est-à-dire dont les sources sont disponibles librement sur internet) pour l'exploitation sur mobile et ainsi concurrencer les systèmes

propriétaires, par exemple Windows Mobile et iOS. Cette alliance a pour logiciel vedette Android.

- **Open Handset Alliance**

La démarche de Google a été d'ouvrir le développement d'Android en rassemblant autour de lui et au travers de l'Open Handset Alliance (OHA) un maximum de sociétés.
Les membres de ce consortium sont très divers : on y trouve des fabricants de téléphones (Sony Ericsson, Samsung ou Motorola. . .), des opérateurs de téléphonie (Sprint, TMobile ou NTTDoCoMo . . .), des sociétés Internet (Google évidemment, eBay), des constructeurs de puces électroniques (Intel, nVidia. . .), ou encore des acteurs du marché du GPS comme Garmin.
Toutes ces entités se retrouvent donc au sein de ce consortium, pour des raisons qui leur sont propres, afin d'oeuvrer au développement d'Android.

4.4. LES AVANTAGES D'ANDROID

- Open source: Le contrat de licence pour Android respecte les principes de l'*open source*, c'est-à-dire qu'on peut à tout moment télécharger les sources et les modifier selon nos besoins, en plus Android utilise des bibliothèques *open source* puissantes, comme par exemple SQLite pour les bases de données et OpenGL pour la gestion d'images 2D et 3D.
- Gratuit : Android est gratuit, autant pour nous que pour les constructeurs.
- Facile à développer **:**Toutes les API mises à disposition facilitent et accélèrent grandement le travail. Ces APIs sont très complètes et très faciles d'accès. D'une autre manière, on peut dire par exemple qu'on peut envoyer un SMS en seulement deux lignes de code.
- Facile à vendre : Le *Play Store* (anciennement *Android Market*) est une plateforme immense et très visitée ; c'est donc une mine d'opportunités pour quiconque possède une idée originale ou utile.
- Flexible: Le système est portable, il s'adapte à beaucoup de structures différentes. Les Smartphones, les tablettes, la présence ou l'absence de clavier ou de *trackball*, différents processeurs... On trouve même des fours à micro-ondes qui fonctionnent à l'aide d'Android. Non seulement c'est une immense chance d'avoir autant d'opportunités, mais en plus Android est construit de manière à faciliter le développement et la distribution en fonction des composants présents dans le terminal.

- Ingénieux: L'architecture d'Android est inspirée par des applications composites et encourage par ailleurs leur développement. Ces applications se trouvent essentiellement sur internet et leur principe est que l'on peut combiner plusieurs composants totalement différents pour obtenir un résultat surpuissant. Par exemple, si on combine l'appareil photo avec le GPS, on peut poster les coordonnées GPS des photos prises.

4.5. ARCHITECTURE D'ANDROID

L'architecture du systeme Android, se compose de 4 couches, et plusieurs composants, comme le montre la figure (4.1), chaque couche utilise les services fournis par la couche inférieure, on va présenter dans ce qui suit chaque couche, en commeançant par le noyau de Linux.

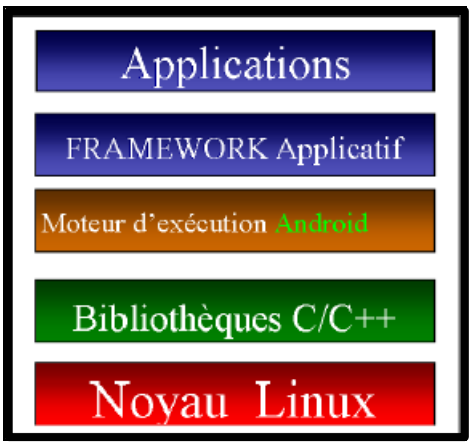

Fig.4.1 : architeture d'Android [11]

4.5.1. le noyau linux

C'est la couche la plus basse de l'architecture, il s'agit de la version 2.6.29 du noyau linux, cette version du noyau utilisée par Android est une version conçue spécialement pour l'environement mobile, avec une gestion avancée de la batterie et une gestion particuliére de la mémoire, c'est cette couche qui fait en sorte qu'Android soit compatible avec tant de supports différents.

Cette couche gére le matériel, la securitè, la gestion de la mèmoire, les drivers, l'alimentation, les processus et les couches réseaux basses. Il inclut, commme indiqué sur la figure 4.2, les différents pilotes nécessaires au fonctionnement de l'appareil, comme les pilotes d'écran, Wifi, IPC...

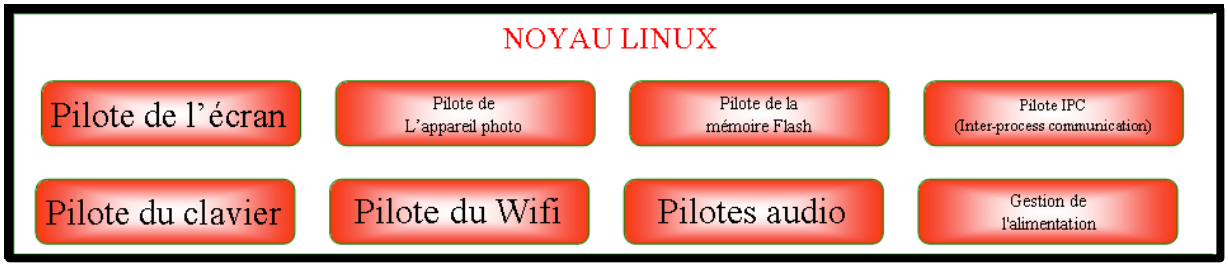

Fig.4.2 : le noyau de linux [12]

Le choix d'un noyau linux a été fait au vue de sa stabilité, sa performance, son modéle de sécurité, ses capacités d'abstraction avec le matériel et enfin pour son aspect open, on remarque que le noyau linux, n'est pas publié sous licence Apache mais sous licence GPL v2.

La figure 4.1 montre que cette couche est la seule qui gére le matériel, en outre le noyau agit comme une couche d'abstraction entre le matériel et la pile logicielle ; c'est à dire que si un constructeur veut ajouter un matériel qui n'est pas pris en compte par défaut par Android , il doit travailler sur les couches supérieures du noyau, qui sont des couches spécifiques à Android, et le matériel peut ètre étendu pour intégrer de nouvelles technologies de pointe, par exemple, de nouveaux pilotes fournis par des developpeurs pour intégrer de nouvelles technologies de communication comme Long Term Evolution.

L'utilisation du noyau de linux comme une couche d'abstraction matérielle, permet à Android d'étre porté à une grande variété de plates-formes (EX: tables, netbooks) dans le futur.

4.5.2. Bibliothèques C/C++

La couche se trouvant au-dessus du noyau comporte un ensemble de bibliothèques C/C++

Comme illustré sur la figure (4.3.)

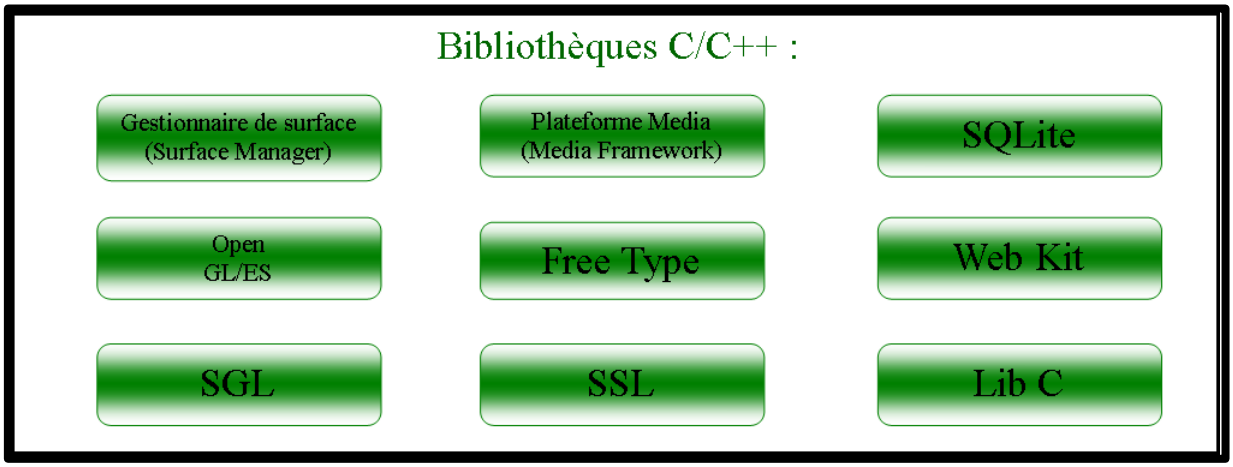

Fig.4.3 : bibliothèques C/C++ [12]

Parmi ces bibliothèques, nous trouvons :

Lib C : La bibliothèque système standard C, dont l'implémentation a été adaptée à un mode embarqué.

Gestionnaire de surface (Surface Manager) : Il s'agit d'un gestionnaire de fenêtres composées (compositing window) .Au lieu d'écrire directement sur la mémoire tampon de l'écran, l'image ira dans des bitmaps (image matricielle) hors écran et sera combinée à d'autres bitmaps pour former l'écran que voit l'utilisateur. Cela permet de réaliser des effets intéressants comme voir à travers une fenêtre.

OpenGL/ES : C'est une version adaptée au système embarqué (ES signifiant Embedded System) de la librairie graphique 3D très é voluée OpenGL. Elle s'appuie sur l'accélération 3D matériel, si l'appareil en est pourvu, dans le cas contraire, elle effectue un rendu purement software.

SGL (Scalable Graphics Library) : Librairie graphique 2D.

Plateforme Media (Media Framework) : Elle est basée sur OpenCore de PacketVideo. Elle permet d'utiliser des standards audio/vidéo/images.

SSL : Il s'agit d'une implémentation de la pile de protocole Secure Sockets Layer. Cette pile est située entre la couche TCP et les protocoles applicatifs. Elle permet de chiffrer l'intégralité

de la connexion entre le client et le serveur.

SQLite : Cette bibliothèque implémente un moteur de bases de données SQL compact, sans serveur et sans configuration.

Web Kit : C'est un moteur de rendu de pages Web (WebKitCore) et d'exécution de code JavaScript (JavaScriptCore). Il est disponible sous licences BSD et LGPL.

4.5.3. Le Moteur d'exécution d'Android

Un moteur d'exécution (« *runtime system* » en anglais) est un programme qui permet l'exécution d'autres programmes. Par exemple pour utiliser des applications développées en Java sur nos ordinateurs on a besoin du JRE (« Java Runtime Environment »).Il s'agit du moteur d'exécution nécessaire pour lancer des applications écrites en Java.

Au dessus des bibliothéques C/C++, se trouve ce moteur d'exécution Android (Android run-time) , comme indiqué sur la figure 4.3. C'est cette couche qui fait qu'Android n'est pas qu'une simple « implémentation de Linux pour portables », elle comporte deux élements.

Fig.4.4 : Moteur d'exécution [12]

4.5.3.1. Bibliothéques internes: on y trouve les bibliothéques qui fournissent les fonctionnalités du language JAVA et des bibliothéques spécifiques à Android

4.5.3.2. Machines virtuelles Dalvik: Android n'utilise pas une JVM classique (JAVA Virtual Machine) mais dispose de sa propre machine virtuelle Dalvik.

Rappelons qu'une machine virtuelle est un ordinateur abstrait simulé à l'intérieur d'un ordinateur réel et employé comme environnement d'exécution d'un langage portable de haut niveau.

4.5.3.3. Comparaison Dalvik avec JVM Classique:

La figure 4.4 est un schéma qui indique les étapes nécessaires à la compilation et à l'exécution d'un programme Java standard.

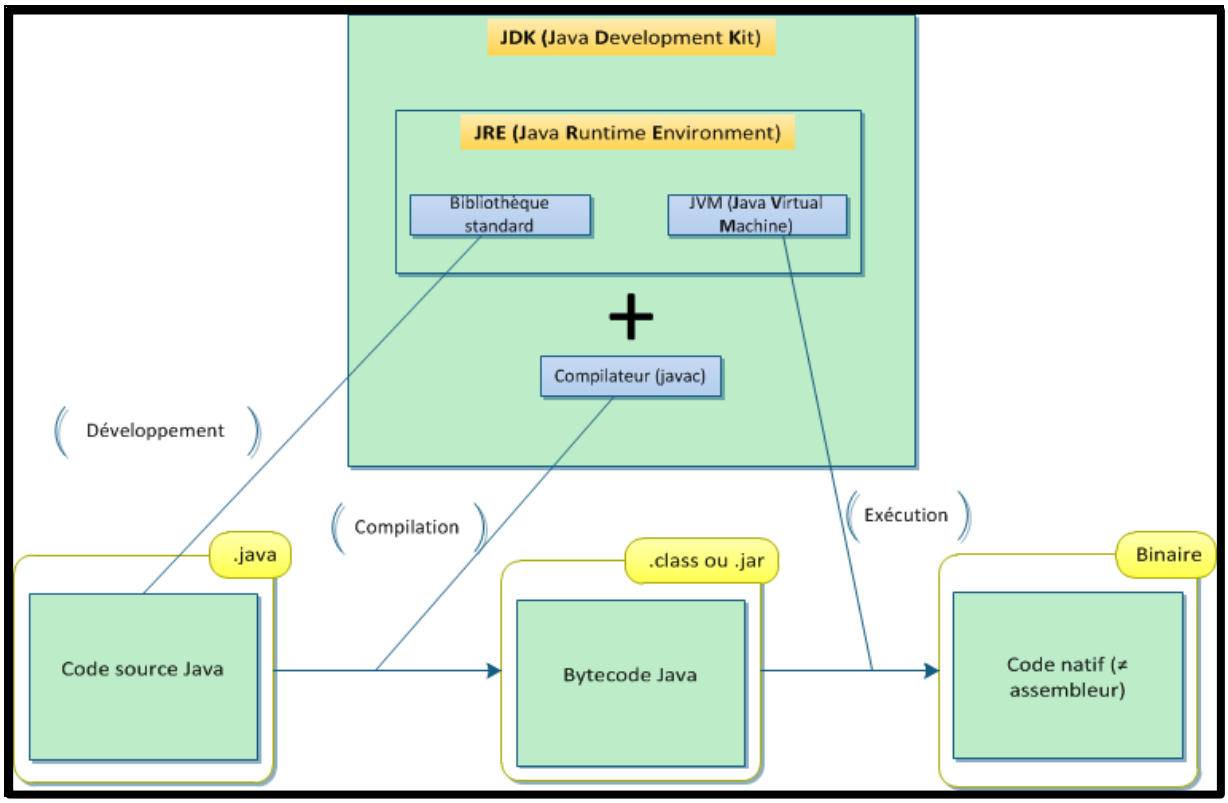

Fig.4.5 : architecture Java [11]

Le code qu'on écrit, est une suite d'instructions que l'on trouve dans un fichier .java qui sera traduit en une autre suite d'instructions dans un autre langage que l'on appelle le « bytecode ». Ce code est contenu dans un fichier .class. Le bytecode est un langage spécial qu'une machine virtuelle Java peut comprendre et interpréter. Les différents fichiers .class sont ensuite regroupés dans un .jar, et c'est ce fichier qui est exécutable. En ce qui concerne Android, la procédure est différente.

À noter que sur le schéma le JDK et le JRE sont réunis, mais il est possible de télécharger le JRE sans télécharger le JDK.

La version de Java qui permet le développement Android est une version réduite amputée de certaines fonctionnalités qui n'ont rien à faire dans un environnement mobile. Par exemple, la bibliothèque graphique Swing n'est pas supportée, on trouve à la place un système beaucoup plus adapté. Android n'utilise pas une machine virtuelle Java, c'est pour cela qu'une machine

virtuelle tout étudiée pour les systèmes embarqués a été développée, la « Dalvik ». Cette machine virtuelle est optimisée pour mieux gérer les ressources physiques du système. Elle permet par exemple de laisser moins d'empreinte mémoire (la quantité de mémoire allouée à une application pendant son exécution) ou d'utiliser moins de batterie qu'une machine virtuelle Java.

La caractéristique la plus importante de Dalvik est qu'elle permet d'instancier (terme technique qui signifie « créer une occurrence de » ; par exemple, quand on crée un objet en java, on instancie une classe puisqu'on crée une occurrence de cette classe) un nombre très important d'occurrences de lui-même : chaque programme a sa propre occurrence de Dalvik et elles peuvent vivre sans se perturber les unes les autres. La figure suivante est un schéma qui indique les étapes nécessaires à la compilation et à l'exécution d'un programme Android standard.

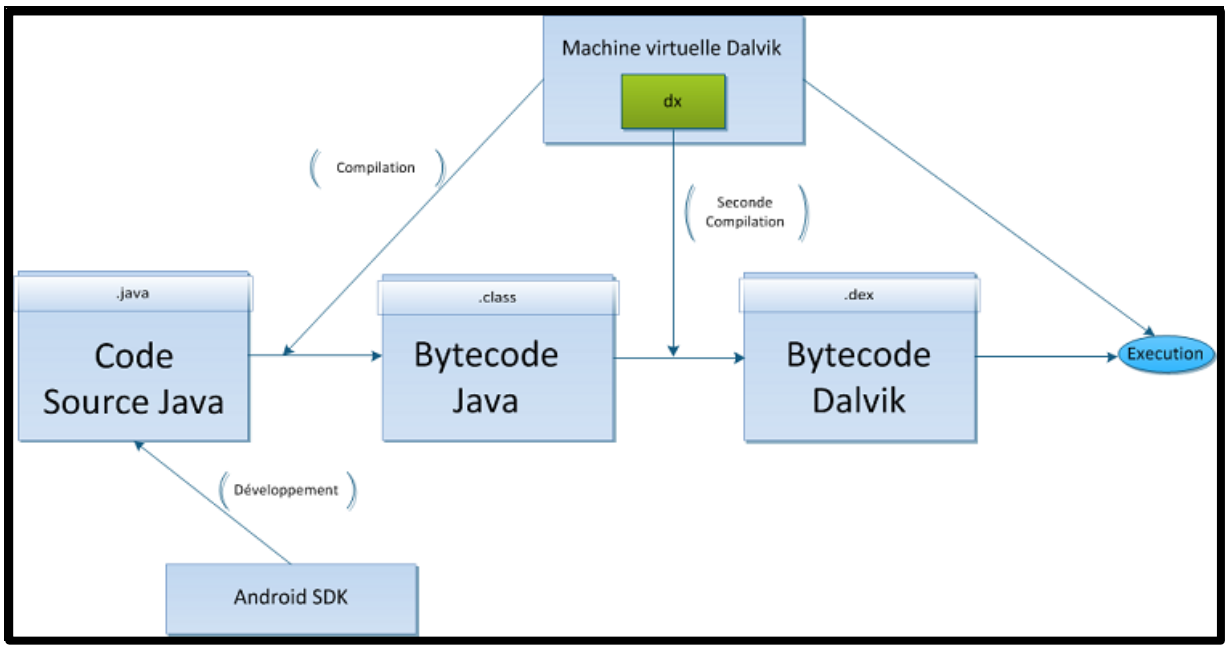

Fig.4.6 : Étapes nécessaires à l'exécution d'un programme sous Android [11]

On voit bien que le code Java est ensuite converti en bytecode Java comme auparavant. Mais, comme il a été déjà mentionné précédemment, que le bytecode Java ne pouvait être lu que par une machine virtuelle Java, et que Dalvik n'était pas une machine virtuelle Java ; il faudra donc procéder à une autre conversion à l'aide d'un programme qui s'appelle « dx » qui s'occupe de traduire les applications de bytecode Java en bytecodeDalvik, qui, lui, est compréhensible par la machine virtuelle.

La puissante machine virtuelle Dalvik est destinée uniquement à Android, mais il est possible de développer une machine virtuelle similaire et qui ne fonctionne pas sous Android. Par exemple, le **Nokia N9** pourra exécuter des applications Android sans utiliser ce système.

Au final, étant optimisé pour utiliser une quantitè de mémoire minimale, la VM Dalvik a quelques caracterstiques spécifiques par rapport à une JVM classique, à savoir :

- Dalvik est basée sur une architecture de registres alors que JVM classique a une architecture de pile.

Ses instructions manipulent un jeu important de registres. Dans le cas où le code réclame plus de données que de registres disponibles, une permutation est effectuèe avec la pile en mémoire.

- Le nombre d'instructions proposées par Dalvik est inférieur de 30% par rapport à une JVM classique et le temps d'exécution proposé par Dalvik est pratiquement deux fois moins important que celui des JVM classiques.

- Les instructions de bas niveau sont trés différentes d'une JVM classique, ce qui implique que le ByteCode java devra d'abord être compilé au format .dex à l'aide de l'outil dx avant son exécution par Dalvik, comme l'illustre la figure 4.6.

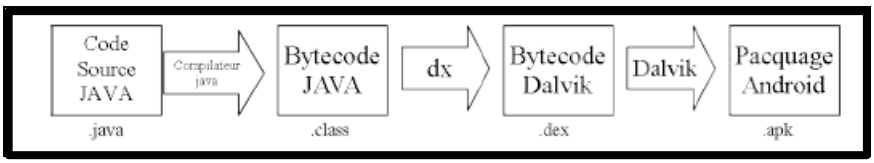

Fig.4.7 : les étapes de compilation [12]

- Le Bytecode Dalvik est beaucoup plus efficace et plus rapide à intitialiser que le Bytecode Java car durant la conversion en format .dex l'édition de liens entre toutes les classes connues est effectuée et tout ce qui peut être mutualisé et anticipé l'est également. L'élimination de cette redondance entre classes permet d'obtenir un seul fichier .dex contenant toutes les classes. Ce fichier est en moyenne deux fois moins volumineux que l'ensemble des fichiers .dex et directement mappé en mémoire.

Remarque: A partir de la version 2.3 Android propose un ramasse-miette qui est un processus de libération automatique des objets qui ne sont plus réferencès par le programme et qui est capable de fonctionner en multitâche pour éviter d'interrompre les applications.

4.5.4. Application framework

La couche au dessus du moteur d'exécution est le framework applicatif.

Framework signifie littéralement cadre d'application, c'est un espace de travail modulaire, il s'agit d'un ensemble de bibliothéques et de conventions permettant, le developpement rapide de l'application, il fournit suffisamment de briques logicielles, impose suffisamment de rigueur pour pouvoir produire une application aboutie et facile à maintenir. Chaque application est en mesure de publier ses capacités, et toutes les autres applications peuvent utiliser ses fonctionnalités.

Cette couche est constituée de plusieurs API (Application Programming Interface), les plus importantes sont mentionnées sur la figure 4.7. Nous pouvons y distinguer :

- **Gestionnaire d'activités (Activity Manager) :** permet de gérer le cycle de vie des applications ainsi que le passage d'une application à une autre et le retour à la précédente quand la dernière application ouverte est fermée.

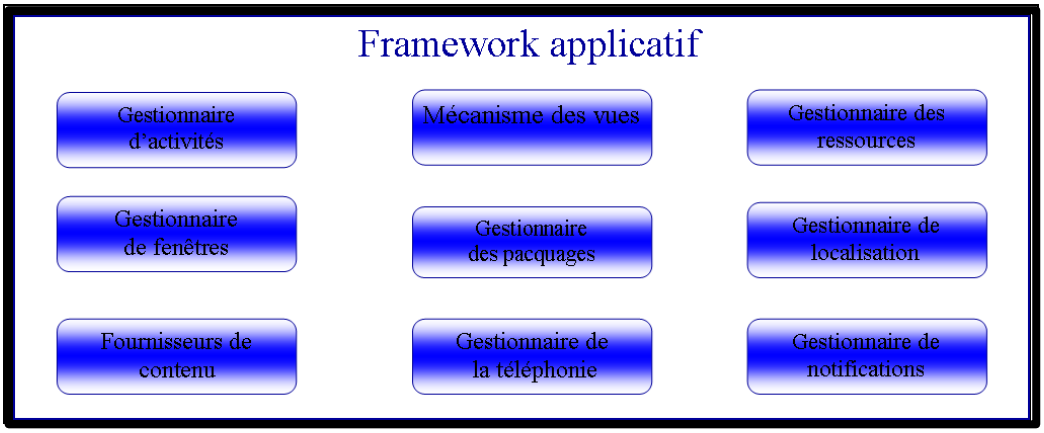

Fig.4.8 : Framework [12]

- **Gestionnaire de paquets (Package Manager) :** utilisé par le gestionnaire d'activités pour charger les informations provenant des fichiers .apk (Android application package file).

- **Gestionnaire de fenêtres (Window Manager) :** il gère les fenêtres des applications. i.e. : qu'elle fenêtre doit être affichée devant une autre à l'écran.

- **Gestionnaire de ressources (Resource Manager)** : gère les images, fichiers audio, etc.

- **Fournisseurs de contenu (Content Provider):** gère le partage de données entre applications. Exemple : La base de données des contacts, peut être consultée par d'autres applications que l'application Contacts.

- **Mécanismes de vue (View System):** fournit tous les composants graphiques : listes, grilles, boutons et même un navigateur web embarqué.

- **Gestionnaire de notification (Notification Manager) :** gère les notifications.

- **Gestionnaire de téléphonie (Telephony Service) :** permet d'accéder aux interfaces téléphoniques (GSM, 3G, etc.).

- **Gestionnaire de localisation (Location Service):** permet d'accéder au GPS.

En plus des gestionnaires de téléphonie et de localisation, le Framework applicatif peut contenir d'autres services matériels fournissant des accès vers les API matérielles de bas niveau , comme par exemple des gestionnaires de Bluetooth, Wifi , USB. . .etc.

4.5.5. Les Applications

Au dessus du Framework applicatif, nous trouvons la dernière couche de l'architecture logicielle, c'est celle qui contient les applications. Android est livré avec un ensemble d'applications de base (fig. 4.8), dont un navigateur web, l'application de gestion des contacts, un programme des SMS, un calendrier. . .

Fig.4.9 : les applications [12]

Toutes les applications sont écrites en utilisant le langage de programmation Java.
La couche application comprend un ensemble d'applications de base préinstallées sur chaque dispositif Android comprenant: Email, Maps, Contacts, Web Browser, Phone Dialer, Calendar, Text message et Android market .
Pour le develepoment des applications, un kit de développement software est fourni avec les outils necessaires et les API.

4.6. LES COMPOSANTS D'ANDROID

Comme il a été mentionné plus haut, une application Android peut utiliser les fonctionalitès des autres applications ; pour que cela fonctionne, le système devrait offrir la possibilitè de démarrer un processus d'une application si une partie de celui-ci est nécessaire et instancier le java pertinent objects.
Par conséquent , les applications d'Android, n'ont pas un seul point d'entrèe.
Au lieu qu'une application soit constituèe de composants que le système peut installer individuellement et exécuter au besoin, Android fournit 4 types essentiels de composants :

4.6.1. Activités

Une activitè présente une interface d'utilisation visuellle, pour une action spécifique que l'utilisateur peut entreprendre.
Un exemple d'une activitèe est l'application du Play Store. On a plusieurs fenêtres à l'intérieur même de cette application : si on effectue une recherche, une liste de résultats s'affichera dans une première fenêtre et si on clique sur un résultat, une nouvelle fenêtre s'ouvre pour nous afficher la page de présentation de l'application sélectionnée. Au final, on remarque qu'une application est un assemblage de fenêtres entre lesquelles il est possible de naviguer.
Ces différentes fenêtres sont appelées des activités. Un moyen efficace de différencier des activités est de comparer leur interface graphique : si elles sont radicalement différentes, c'est qu'il s'agit d'activités différentes. De plus, comme une activité remplit tout l'écran, notre application ne peut en afficher qu'une à la fois. La figure (4.10) illustre ce concept.

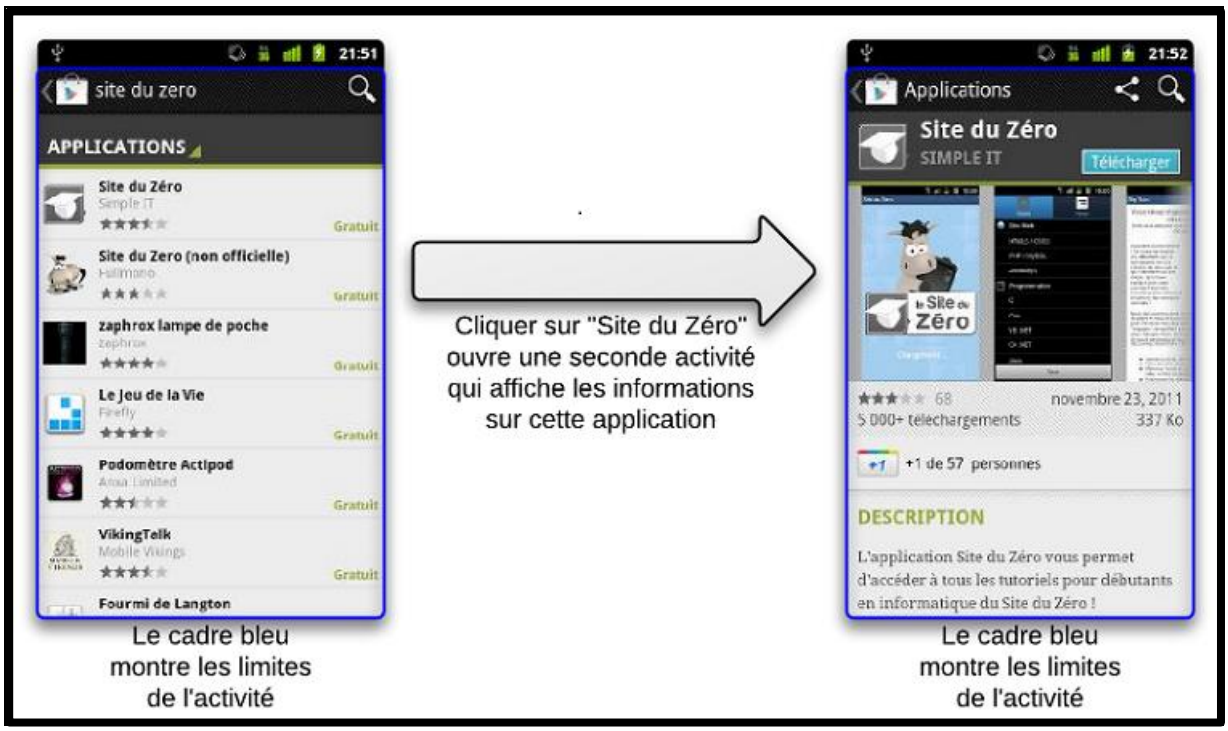

Fig.4.10 : exemple d'une activité [11]

On pourra définir également une activitè comme un support sur lequel on va greffer une interface graphique. Nous rappelons qu'une interface graphique est un ensemble d'éléments visuels avec lesquels peuvent interagir les utilisateurs, ou qui leur fournissent des informations.

Un autre exemple d'une activité est l'application de carnet d'adresses qui peut avoir une activité qui affiche une liste de tous les contacts disponibles, et une deuxième activité qui affiche les détails du contact choisi. En outre, une activité peut consister en plusieurs vues comme des boutons, champs de texte, des barres de défilement, des éléments de menu, cases à cocher et plus encore.

Lorsque l'utilisateur passe sur un bouton à côtè d'un numéro de téléphone, cela declenchera l'action pour commencer une autre acitivté pour écrire un message texte.

Les vues sont là où les interactions de l'activité avec l'utilisateur a eu lieu. L'activitè du détail de contact mentionnée plus haut, peut par exemple consister en plusieurs champs de textes qui affichent les informations des contacts comme: E-mail, numèro de téléphone, adresse et bouton pour effectuer des actions relatives aux informations de contact.

Cependant, ce n'est pas le rôle de l'activité de créer et de disposer les éléments graphiques, elle n'est que l'échafaudage sur lequel vont s'insérer les objets graphiques.

De plus, une activité contient des informations sur l'état actuel de l'application : ces informations s'appellent le contexte. Ce contexte constitue un lien avec le système Android ainsi que les autres activités de l'application, comme le montre la figure (4.11).

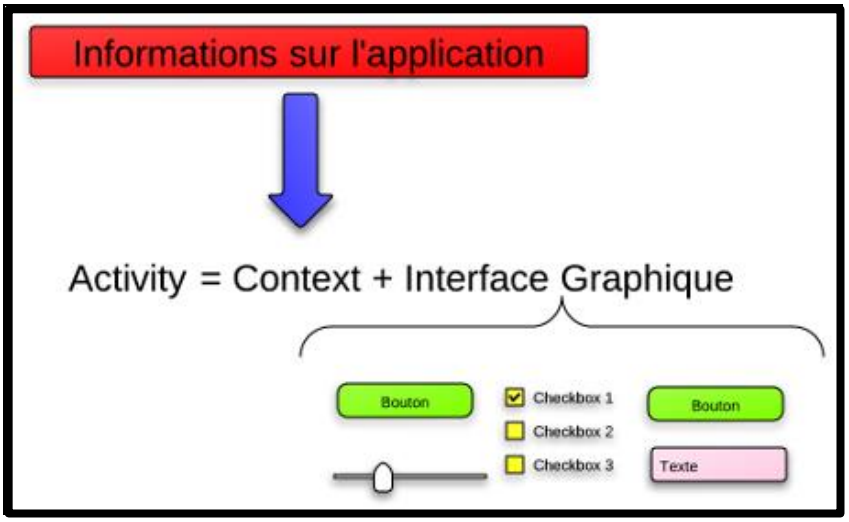

Fig.4.11 : les composants d'une activitè [11]

Pour mieux illusterer ceci, on pourra imaginer que nous naviguons sur facebook avec notre téléphone tout en écoutant de la musique sur ce même téléphone. Il se passe deux choses dans notre système :

- La navigation sur internet, permise par une interface graphique (la barre d'adresse et le contenu de la page web, au moins) ;
- La musique, qui est diffusée en fond sonore mais qui n'affiche pas d'interface graphique à l'heure actuelle puisque l'utilisateur consulte le navigateur.

On a ainsi au moins deux applications lancées en même temps ; cependant, le navigateur affiche une activité alors que le lecteur audio n'en affiche pas.

États d'une activité

Si un utilisateur reçoit un appel, il devient plus important qu'il puisse y répondre que d'émettre la chanson que votre application diffuse. Pour pouvoir toujours répondre à ce besoin, les développeurs d'Android ont eu recours à un système particulier :

- À tout moment une application peut laisser place à une autre application, qui a une priorité plus élevée.
- Votre activité existera dans plusieurs états au cours de sa vie, par exemple un état actif pendant lequel l'utilisateur l'exploite, et un état de pause quand l'utilisateur reçoit un appel.

Pour être plus précis, quand une application se lance, elle se met tout en haut de ce qu'on appelle la pile d'activités.

Rappelons qu'une pile est une structure de données de type « LIFO », c'est-à-dire qu'il n'est possible d'avoir accès qu'à un seul élément de la pile, le tout premier élément, aussi appelé sommet. Quand on ajoute un élément à cette pile, le nouvel élément prendra la première place et deviendra le nouveau sommet. Quand on veut récupérer un élément, ce sera le sommet qui sera récupéré, sorti de la liste et l'objet en deuxième place deviendra le nouveau sommet, comme illustré à la figure (4.12).

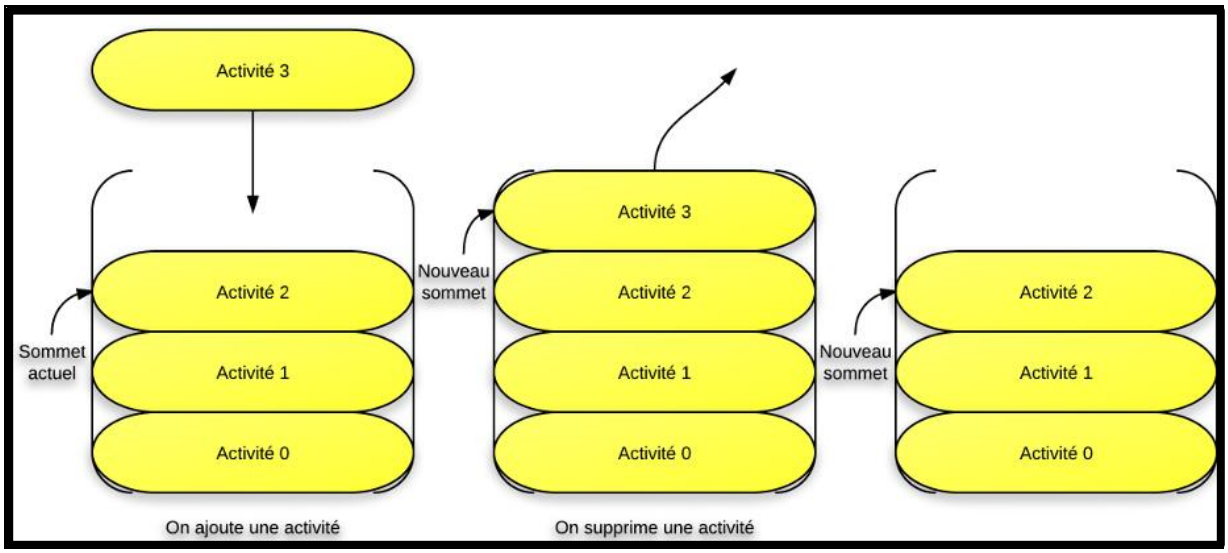

Fig.4.12 : pile d'activité [11]

L'activité que voit l'utilisateur est celle qui se trouve au-dessus dans la pile. Ainsi, lorsqu'un appel arrive, il se place au sommet de la pile et c'est lui qui s'affiche à la place de l'application en cours, qui n'est plus qu'à la deuxième place. L'activité actuelle ne reviendra qu'à partir du moment où toutes les activités qui se trouvent au-dessus d'elle seront arrêtées et sorties de la pile. On retrouve ainsi le principe expliqué précédemment, on ne peut avoir qu'une application visible en même temps sur le terminal, et ce qui est visible est l'interface graphique de l'activité qui se trouve au sommet de la pile.

4.6.2. Les services

Le service est un composant d'application, sans une interface d'utilisateur, qui s'exécute dans l'arriére plan, pour une durée de temps indéfinie. Ils sont parfaits pour effectuer un traitement

continu ou régulier même lorsque l'activitè de l'application n'est pas active, invisible ou fermée.

Android prend une charge deux types de services:

- Service local:

N'est pas pas accessible à partir d'autres applications et donc généralement utilisé pour soutenir l'application qui héberge le service.

- Remote service:

Il est accessible à partir d'autres applications via une interface bien définie, en utilisant l'Android interface definition language (AIDL). Ce genre de service est utilisé pour la communication inter-processus.

Les services sont lancés dans le thread principal des applications, comme toutes les autres composantes de l'application.

Une fois qu'un service est démarré, il va vivre jusqu'à ce qu'il soit arrété explicitement ou résilié par le manager des ressources du moteur d'exécution.

La seule raison pour laquelle Android va arrêter un service qui est en cours d'exécution, est de fournir des ressources supplémentaires pour un composant de premier plan (en général une activité active). Lorsque cela se produit, le service sera redémarré automatiquement dès que les ressources seront disponibles à nouveau.

Donc, parceque les services ont la deuxième prioritè attribuèe par Android, ils sont moins susceptibles d'être résiliés par le gestionnaire de ressources d'un moteur d'exécution.

Un exemple typique d'un service tiré du site Android développeurs, est un média joueur qui joue des chansons à partir d'une liste de lecture. L'application du lecteur peut consister en une ou plusieurs activités qui permettent à un utilisateur de choisir des chansons et de les jouer, bien que la lecture de la musique elle-même ne serait pas traitée par une activité,parce que la musique doit continuer même après que l'utilisateur quitte le lecteur. Pour garder parfaitement la musique continue, l'activité de lecteur multimédia pourrait démarrer un service qui s'exécute en arrière-plan jusqu'à l'arrêt explicite.

Android comprend lui-même plusieurs sevices pouvant ètre utilisès, par exemple, le gestionnaire d'emplacement media controller ou gestionnaire de notification.

4.6.3. Récepteur de diffusion

Le recepteur de radiodiffusion sont des composants destinés à recevoir et réagir aux événements de radiodiffusion.

Dans Android de nombreuses émissions proviennent du code du système telles que les émissions annonçant la disponibilitè du réseau, l'état de la batterie ou les mails entrants.
Mais une application peut également lancer elle même une émission ou informer les autres applications sur un événement en cours.
Les récepteurs de diffusion sont des éléments importants pour permettre la création d'applications événementielles, fondées sur le contrôle interne du système lui même.

4.6.4. Intent

Activitès, services et récepteur de diffusion peuvent être activés à l'aide des intents. Une intent n'est rien d'autre qu'un objet qui contient le contenu du message. Les intents sont des messages asynchrones passés entre les composantes d'une application ou entre les applications et fournissant un moyen de transformer une collection de composants dans un système interconnectè.

4.6.5. Contents providers

Les fournisseurs de contenu sont utilisès pour partager les données de l'application avec d'autres applications sur l'appareil. Pour cela, le fournisseur de contenu étend la classe de *base content provider* et met en oeuvre un ensemble de méthodes pour permettre à d'autres applications de récupérer et stocker les données du type qu'elle contrôle.
Une application va alors utiliser un object *contentresolver* pour communiquer et pour coopérer avec le fournisseur correspondant afin d'extraire et stocker les données. Comme avec le récepteur de diffusion, il y a aussi des fournisseur de contenu natif qui expose les bases des données comme les média store ou bien les informations du contact. Les fournisseurs de contenu sont le seul moyen de partager des données entre les applications.

4.7. OUTILS DE DEVELOPPEMENT

Afin de développer une application Android, il est nécessaire d'installer un ensemble d'outils, que l'on va présenter dans ce qui suit.

4.7.1. JRE (Java Runtime Environment)

Il s'agit de l'environnement d'exécution Java (JRE) qui permet d'exécuter des applications spécifiques à la plate-forme Java.

Il contient la **JVM** (**J**ava **V**irtual **M**achine), les bibliothèques de base du langage ainsi que tous les composants nécessaires au lancement d'applications ou d'applets Java. En gros, c'est l'ensemble d'outils qui nous permettra d'exécuter des applications Java.

Fig.4.13 : le JRE [12]

4.7.2. JDK (Java Development Kit)

Le **JDK** (**J**ava **D**evelopment **K**it) contient le JRE (afin d'exécuter les applications Java) mais aussi un ensemble d'outils pour compiler et déboguer le code.

4.7.3. SDK (Software Development Kit)

Un SDK, ou kit de développement en français, est un ensemble d'outils qui met à disposition un éditeur afin de nous permettre de développer des applications pour un environnement précis. Le SDK Android permet donc de développer des applications pour Android et uniquement pour Android (fig. 4.13).

Il contient plusieurs sous dossiers, dont :

- Le sous dossier Tools qui contient les différents utilitaires Android, comme DDMS un puissant outil de débogage.
- Le sous dossier Platforms qui contient les classes du JDK et celles qui sont propres à Android, le tout regroupé dans la bibliothèque Android.jar.
- Le sous dossier Add-on qui contient les bibliothèques optionnelles, comme la librairie "com.google.Android.maps" qui permet l'affichage de cartes Google.
- Les sous dossiers Docs et Samples , qui contiennent respectivement la documentation d'Android et des exemples d'applications.

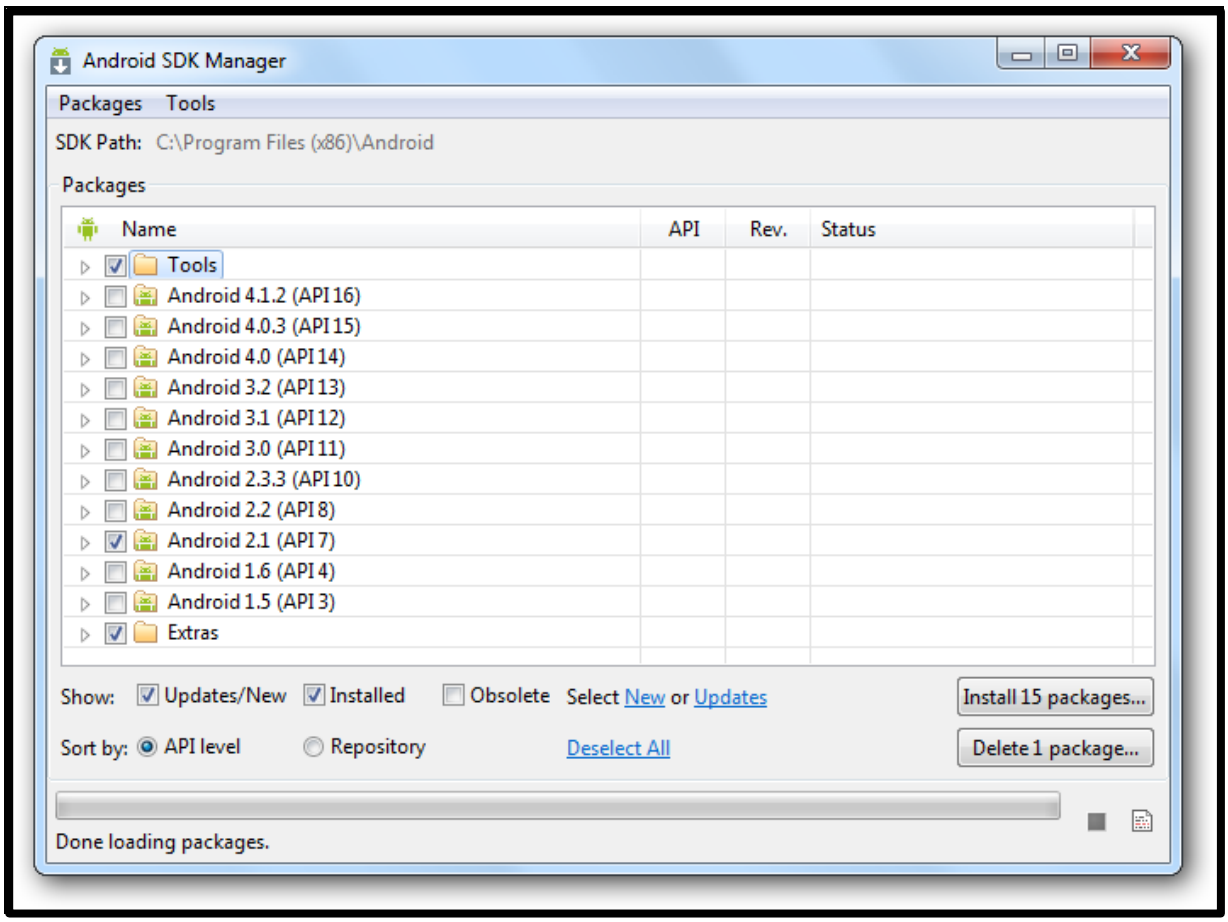

Fig.4.14 : le SDK [11]

4.7.4. Eclipse (IDE) et ADT (Android Developpement Tools Plug-in) :

Eclipse est un environnement de développement gratuit, disponible sous Linux, Windows et Mac OS. Il facilite le développement en Java.

http ://www.eclipse.org/downloads.

Eclipse est bien adapté au développement Android car Google propose un module (plug-in) compatible avec cet IDE qui permet de rendre la transformation des classes java en format dex et la création du package apk transparente pour le développeur.

4.8. CONCLUSION

Flexible, facile à vendre, gratuit et open source, sont les principaux avantages qui nous ont poussé à choisir de développer notre application sous Android.

5.1. INTRODUCTION

Dans le chapitre 3 nous avons montré les différents résultats de simulation de la détection de chute sur Matlab. Nous allons, à travers ce chapitre, utiliser une liaison sans fil entre un circuit d'acquisition (Shimmer Device) et un système d'exploitation (Android), pour montrer la possibilité de la détection de chute à distance et en temps réel constituant un système d'aide à la surveillance de personnes âgées.

5.2. SIMULATION ET IMPLEMENTAION SOUS ANDROID

La figure 5.1 illustre le système que nous avons implémenté et qui est constitué de trois blocs.

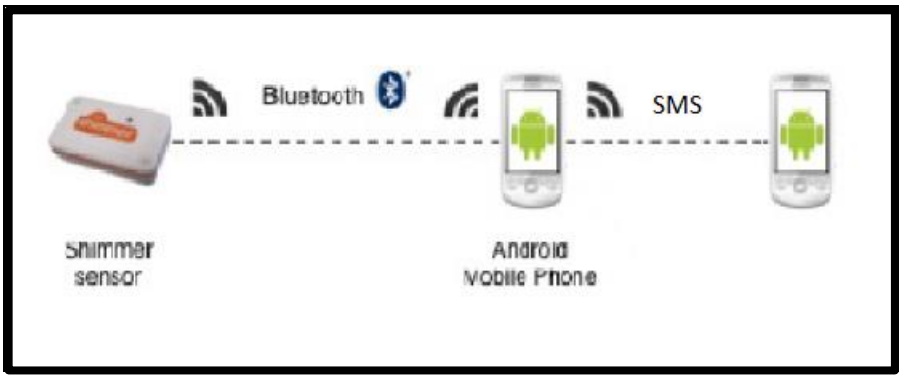

Fig.5.1 : Système proposé

1. Communication et transmission des données entre Shimmer Device et le Smartphone en utilisant le driver de l'équipe de Shimmer.
2. Traitement de données et détection de la chute, cette dernière a été faite par l'implémentation de notre algorithme présenté précédemment.
3. L'envoi d'un SMS au médecin, qui suit le patient à distance, quand une chute est détectée.

   Nous allons expliquer, dans ce qui suit, le fonctionnement et la réalisation de ces trois blocs (ou étapes).

5.2.1. Etapes du bloc 1 :

Avant de présenter le driver de Shimmer, on rappelle que le principe de communication entre Shimmer et Smartphone se fait par le SPP. Le SPP étant un protocole de communication entre deux dispositifs qui communiquent par Bluetooth.

5.2.1.1 Présentation générale du driver de Shimmer sur Android

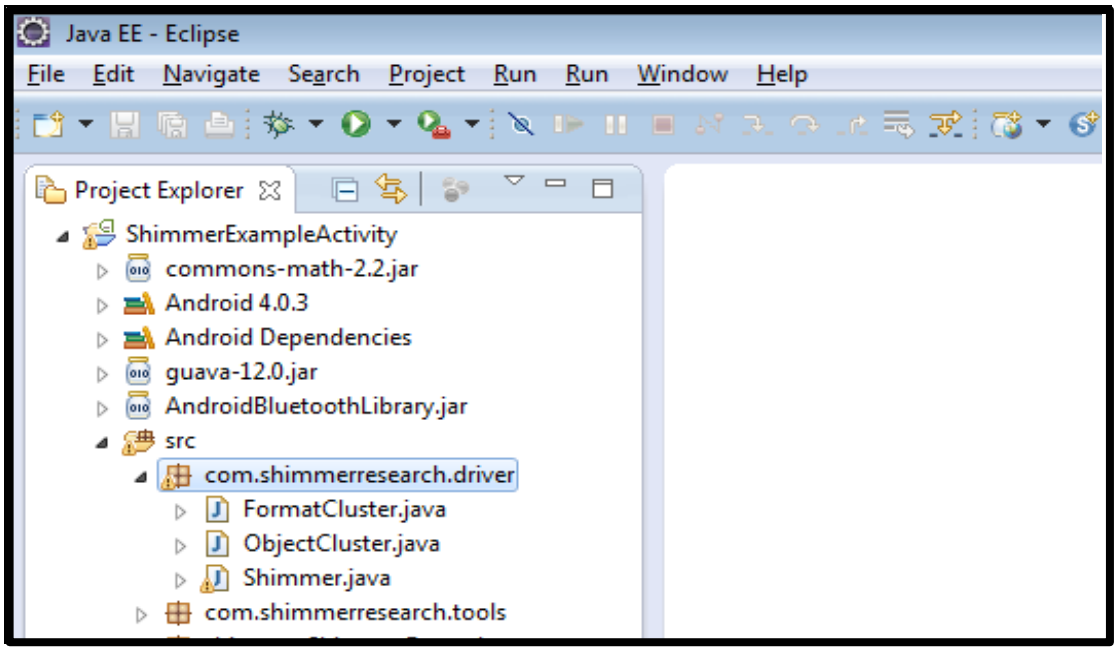

Fig.5.2 : Structure du projet [14]

Le driver de Shimmer sur Android est un ensemble de paquets (packages), fourni dans le but d'accélérer le temps de développement pour les utilisateurs de Shimmer.

Donnons, les principaux packages de la fonction src :

- Le package com.shimmerresearch.driver contient trois fichiers, le premier fichier, shimmer class, c'est le driver principal, qui permet à UI d'Android, de se connecter au ShimmerDevice, il permet aussi l'interaction avec ShimmerDevice comme par exemple, modifier la fréquence d'échantillonnage ou changer le rang de l'accélération. Les 2 autres fichiers (FormatCluster and ObjectCluster), à l'intérieur de package com.shimmerresearch.driver, définissent la structure des données.
- Le package com.shimmerresearch.tools contient une logging classe, qui permet à l'utilisateur de faire entrer (log) aisément les données dans le Smartphone.

5.2.1.2 *La classe Shimmer*

La classe Shimmer se base sur le module Bluetooth, fourni par Android pour communiquer avec ShimmerDevice par l'intermédiaire de serial profile port (SPP). SPP émule une liaison câble série sur la technologie sans fil Bluetooth et chaque ShimmerDevice qui est connecté à Android, est représenté par l'objet Shimmer qui est une instance de la classe Shimmer. Par exemple:

Shimmer mShimmerDevice1 = **new** Shimmer(**this**, mHandler,"RightArm",false);

Shimmer mShimmerDevice2 = **new** Shimmer (**this**, mHandler,"LeftArm",false);

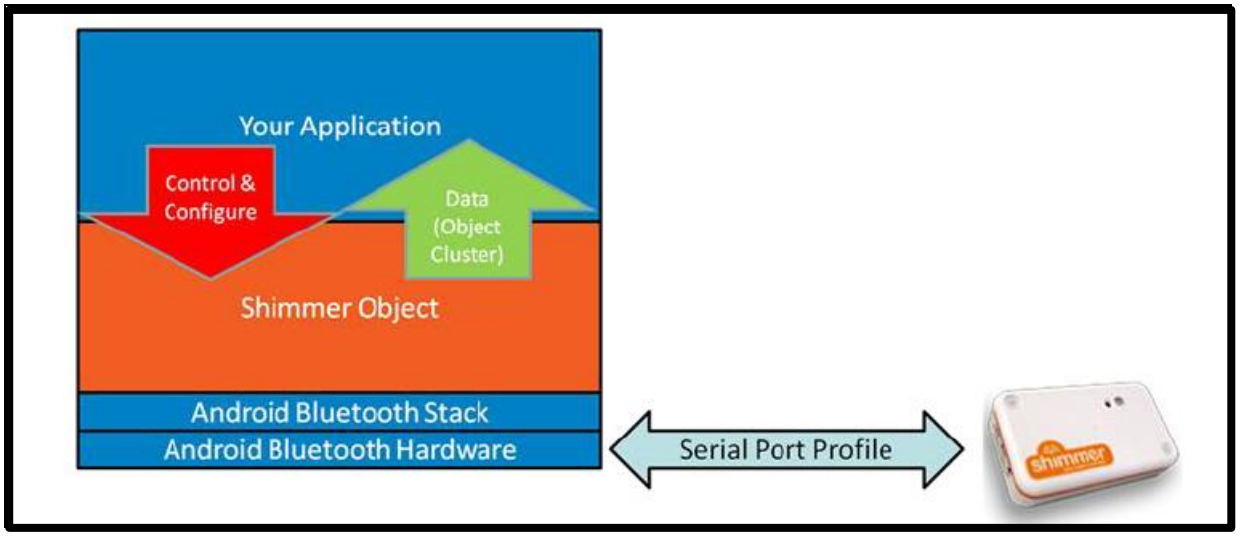

Fig.5.3. : l'objet Shimmer [14]

Les éléments suivants sont parmi les fonctions fournies par la classe Shimmer :

- Connect : exemple, *mShimmerDevice1.connect(bluetoothAddress,"default");*

  Tentative pour connecter Android device à Shimmer Device ; il y a deux bibliothéques séparées fournies, default et gerdavax, et c'est seulement dans le cas où la bibliothéque default ne répond pas à nos appels que l'on doit tenter d'utiliser la bibliothéque gerdavax.
- Startstreaming: commencer l'acquisition des données, provenant de Shimmer Device
- Stopstreaming: arrêter l'acquisition des données provenant de Shimmer
- Inquiry: c'est une commande de renseignement qui permet à l'appereil Android de connaître la configuration actuelle de Shimmer Device ; à la reception d'une demande de renseignement, le Shimmer va transmettre un paquet de réponses à la demande de

renseignement. Le paquet de réponses contient des informations concernant la configuration actuelle du dispositif Shimmer, tels que les capteurs qui ont été activés, la taille des paquets de données et le contenu du paquet de données.

- Writeenabledsensors: cette commande est utilisée pour activer des capteurs spécifiés sur le dispositif, une fois que le paquet de reconnaisance est reçu, une commande est executée afin d'assuer que le format des paquets est mis à jour. Quant à l'activation de plusieurs capteurs sur le dispositif, on utilise le format suivant où l'opérateur a été utilisé.

  Exemple :

  (mShimmerDevice1.writeEnabledSensors(Shimmer.SENSOR_ACCEL|Shimmer.SENSOR_GYRO);)
- Writesamplingrate: on transmet cette commande au Shimmer pour configurer sa fréquence d'échantillonage.
- Writeaccelrange: on envoit cette commande au Shimmer pour régler la plage de variation de l'accélération.

5.2.1.3 Structure des données

La classe objectcluster et propertyCluster, décrit la structure des données utilisées à l'intérieur de la bibliothèque. Cette structure des données est utilisée pour encapsuler les données à partir du driver.

Objectcluster est constitué de multipmap (property cluster), où chaque clé représente une propriété (par exemple l'accéléromètre X) et chaque valeur le Format Cluster de cette propriété. Le Format Cluster est un objet qui détient le format (par exemple calibré), les unités et les données de la propriété.

La figure (5.4) montre la structure

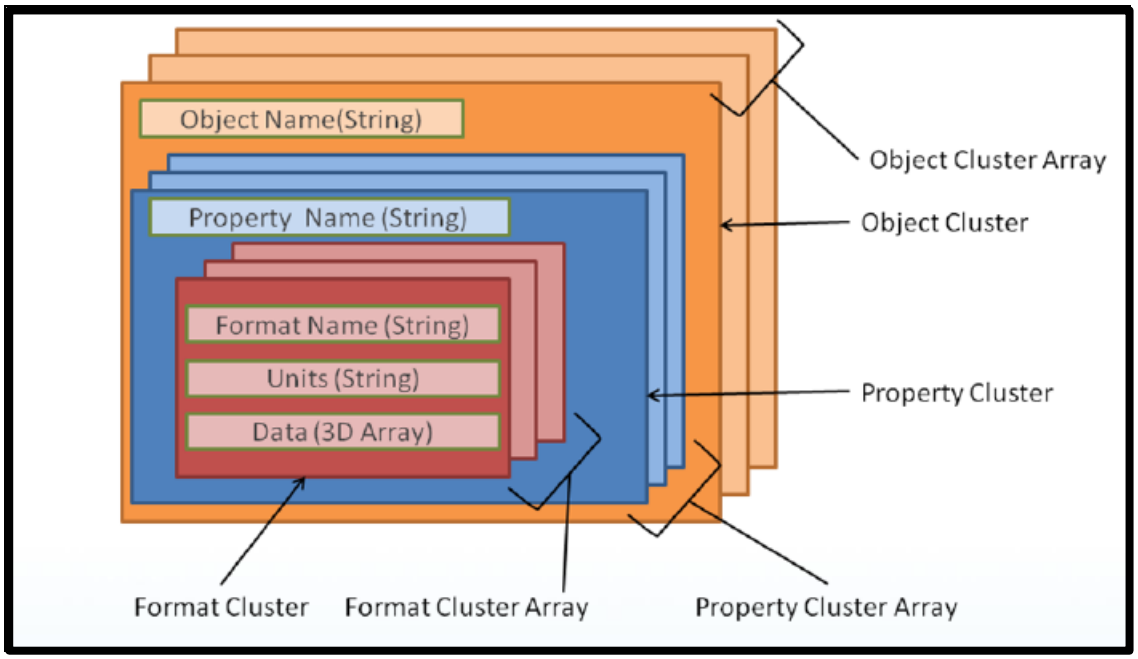

Fig.5.4. la structure des données [14]

Cette conception a été choisie pour permettre de nouvelles propriétés telles que l'accélération et l'orientation linéaire pour être incluses facilement dans l'avenir avec de nouveaux formats tels que ceux filtrés .Grâce à l'utilisation de MultiMaps, les propriétés au sein de la structure de données peuvent être trouvées facilement.

Les 8 points suivants montrent les opérations basiques, de l'objectcluster :

1. ObjectCluster objectCluster=**new** ObjectCluster("RightArm");
2. objectCluster.mPropertyCluster.put("Accelerometer X",**new** FormatCluster("CAL","m/(sec^2)",0.5));
3. objectCluster.mPropertyCluster.put("Accelerometer X",**new** FormatCluster("RAW","no units",50));
4. objectCluster.mPropertyCluster.put("Accelerometer Y",**new** FormatCluster("CAL"," m/(sec^2)",0.5));
5. objectCluster.mPropertyCluster.put("Accelerometer Y",**new** FormatCluster("RAW","no units",50))
6. Collection<FormatCluster> accelXFormats = objectCluster.mPropertyCluster.get("Accelerometer X"); // first retrieve all the possible formats for the current sensor device
7. FormatCluster formatCluster = ((FormatCluster)objectCluster.returnFormatCluster(accelXFormats,"CAL")); // retrieve the calibrated data

8. dataValue = formatCluster.mData;

Dans le premier point un nouvel objet cluster nommé rightarm a été créé, les points 2 à 5 montrent comment les nouvelles propriétés ont été ajoutées à objectcluster. L'objet cluster résultant est montré dans la figure (5.3).

Les points 6 à 8, montrent comment on peut récupérer les données de la structure créée. Initialement dans le point 6, la MultiMAp (propertycluster) est en train de chercher la propriété « accéléromètre X », le résultat de cette recherche est une collection de différents formats, c'est-à-dire calibré et non calibré, puis pour le point 7, la fonction ReturnFormatCluster, est utilisée pour récupérer les données sous le format calibré. Finalement, au point 8, on stockera les données sous la variable datavalue.

La liste des propriétés supportées par la classe Shimmer sont :

Timestamp, Accelerometer X, Accelerometer Y, Accelerometer Z, Gyroscope X, Gyroscope Y, Gyroscope Z,

Magnetometer X, Magnetometer Y, Magnetometer Z, GSR, ECG RA-LL, ECG LA-LL, EMG, Strain Gauge High,

Strain Gauge Low, Heart Rate, ExpBoard A0, ExpBoard A7, VSenseReg and VSenseBatt

La liste des formats supportès par la classe shimmer sont:

CAL, RAW

5.2.2. Etapes du bloc 2 :

Pour que l'on puisse détecter une chute, le changement d'amplitude seulement suffira, à condition que ce changement se passe dans un intervalle inférieur à 500ms.

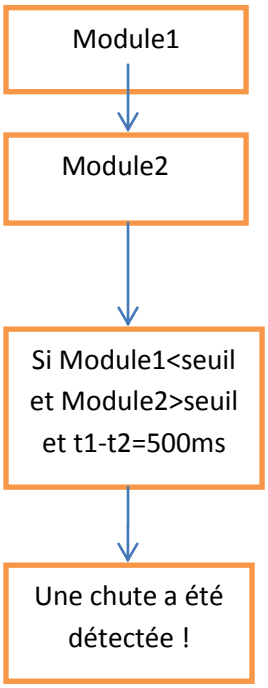

Fig.5.5. : orgranigramme de l'algorithme

Pour simuler, la détection de la chute, on a utilisé la fonction system.out.printf, pour qu'elle nous affiche cette dernière dans un fichier logcat, comme le montre la simulation en figure (5.6)

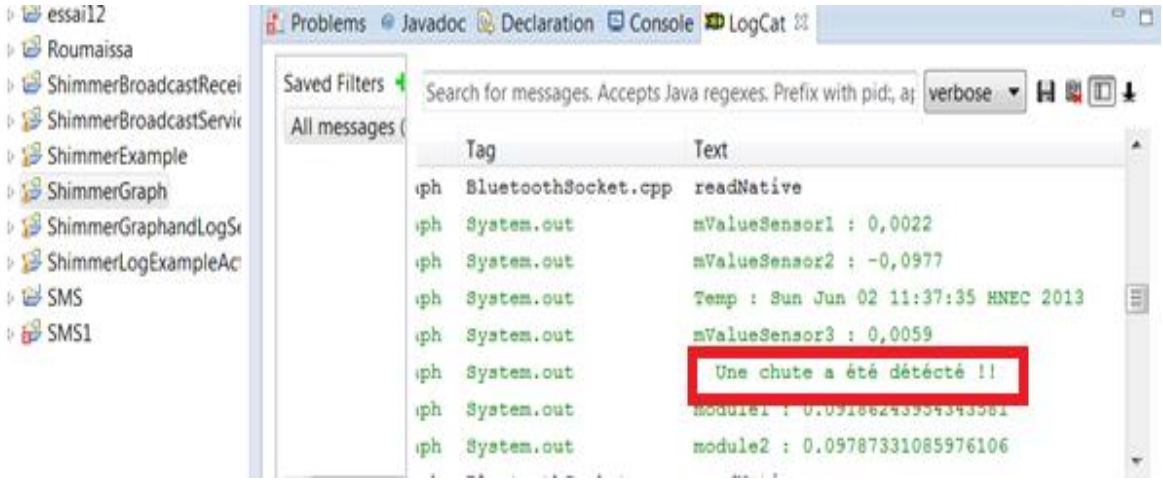

Fig.5.6 : Simulation

5.2.3 Etapes du bloc 3 :

Cette partie consiste en l'envoi d'un SMS au médecin, dans le cas où une chute est détectée.

Pour cela , on doit utiliser la classe, SmsManager, puis on va instancier un objet avec la méthode staticSmsManagerSmsManager.getDefault().

Finalement Pour envoyer un message, il suffit d'utiliser la méthode voidsendTextMessage (String destinationAddress, String scAddress, String text, PendingIntentsentIntent, PendingIntentdeliveryIntent).

5.3. CONCLUSION

Nous avons exposé dans ce chapitre les différentes étapes poursuivies lors de la détection de chute par notre système constitué de *ShimmerDevice* dont le capteur associé (accéléromètre) fournit les données à traiter par *Android du Smartphone* qui doit envoyer un message sur le *Téléphone* du médecin traitant en cas de chute du patient. Ce système a été réalisé puis utilisé pour plusieurs cas (chute réelle détectée, chute lente (cas d'évanouissement) non détectée).

Le nombre de tests n'étant pas suffisant, l'évaluation de notre système ne serait pas probante, c'est pour cette raison que nous ne l'avons pas effectuée.

Conclusions et Perspectives

L'objectif principal de notre Projet de Fin d'Etude est la conception d'un système de détection de chute en temps réel. Nous avons proposé pour la réalisation de notre système, la plateforme Shimmer qui est une nouvelle plateforme, pratique et conçue pour être portée à tout moment et qui a surtout une faible consommation d'énergie.

Les trois signaux provenant de la Shimmer et qui sont l'accélération sur les axes X,Y et Z, ont été acquises par utilisation du logiciel Matlab, pour enfin effectuer une série de traitements consistant à éliminer les activités quotidiennes du patient, puis calculer l'amplitude et l'inclinaison du patient afin de détecter une éventuelle chute.

Nos simulations sur Matlab, ont confirmé l'efficacité de notre algorithme en détectant rapidement les chutes avec la plateforme que nous avons utilisé.

Nous avons conçu également un système performant sans fil, constitué de la plateforme Shimmer et d'un Smart phone. L'acquisition des données provenant des capteurs de la Shimmer Device se fait par l'intermédiaire d'un programme liant Shimmer à Android.

Le traitement des trois signaux se fait sur le Smartphone et dans le cas où une chute est détectée, le Smartphone envoie un SMS d'alarme au médecin qui suit son patient à distance. Notre système a donné des résultats, en temps réel, satisfaisants restant bien sûr à améliorer.

Comme perspectives, Nous pouvons proposer la réalisation d'un système complet pour le Connected Healt détectant non seulement les chutes mais aussi d'autres pathologies comme la surveillance des cardiopathies, les cas d'épilepsies, etc. Et pourquoi ne pas, envisager la possibilité d'implémenter le système sur circuit FPGA.

References Bibliographiques

[1] http://www.shimmer-research.com

[2] Shimmer User Manual Rev2Rd

[3] K Doughty , R Lewis and A McIntosh, The design of a practical and reliable fall detector for community and institutional telecare , Journal of Telemedicine and Telecare Volume 6 Supplement 1 2000

[4] Yaniv Zigel, Member, IEEE, Dima Litvak, and Israel Gannot, A Method for Automatic Fall Detection of Elderly People Using Floor Vibrations and Sound—Proof of Concept on Human Mimicking Doll Falls, IEEE TRANSACTIONS ON BIOMEDICAL ENFINEERING, DECEMBER 2009

[5] V.Vaidehi, Kirupa Ganapathy, K. Mohan, A.Aldrin, K.Nirmal, Video Based Automatic Fall Detection In Indoor Environment, IEEE-International Conference on Recent Trends in Information Technology, ICRTIT 2011

[6] U. Lindemann , A. Hock, M. Stuber , W. Keck , C. Becker ,Evaluation of a fall detector based on accelerometers: a pilot study, Medical & Biological Engineering & Computing 2005, Vol. 43

[7] Gaetano Anania, Alessandro Tognetti, Nicola Carbonaro, Mario Tesconi, Fabrizio Cutolo, Giuseppe Zupone, Danilo De Rossi, Development of a novel algorithm for human fall detection using wearable sensors, Interdepartmental Reserch Center "E.Piaggio" University of Pisa ,Pisa, Italy

[8] Federico Bianchi, Stephen J.Redmond, Michaeal R.Narayanan, Sergio Cerutti, Branko G.Celler, Nigel H.Lovell, Falls Event Detection using Triaxial Accelerometry and Barometric Pressure Measurement , 31st Annual Inetrnatio,al Conference of the IEEE EMBS Minneapolis, Minnesota, USA, September 2-6, 2009

[9] Mehdi Neggazi, Abbes Amira, Latifa Hamami, Signal and communication laboratory, Ecole Nationale Polytechnique A Wireless Reconfigurable System for Falls Detection , 2012

[10] Matlab Instrument Driver User Manual, Rev 1.0a

[11] http://www.siteduzero.com/tutoriels

[12] OUDNI Louiza, Développement d'applications mobiles pour Android, Application au développement d'une application GPS, séminaire 4eme année, 2012, dirigé par Dr.Sadoun

[13] Sacha Gilgen, Basel, Switzerland, Mobile Healthcare on Android Devices, Novembre

2,2010

[14] Shimmer-Android-Instrument-Driver-User-Manual-0.7-Beta