Sciences Numériques
2A filière réseaux

Rapport Projet de Réseaux Sans Fils

Yahya Younes
Mohamed El Yessefi
Nabil Ait Alaya

Supervisé par :
Mr Kateu Demlabin Firminleroy

Preface

Ce projet vise à nous familiariser avec la manipulation des données des capteurs, leur envoi sur un réseau sans fil et leur traitement sur une plateforme cloud. il s'agit de développer une application qui permet de compter le nombre de pas effectués par l'utilisateur. Le dispositif de collecte de données sera le smartphone, la connectivité sera assurée par un simple point d'accès wifi, et l'essentiel du traitement incluant le calcul du nombre de pas sera fait par une application web implémentée sur un serveur (un PC connecté au point d'accès).

Table des matières

1 Introduction **4**

2 Rappel algorithme : **4**

3 Partie client **5**

4 Partie serveur **5**

5 Difficultés rencontrées et horizons d'amélioration **6**

5.1 Difficultés rencontrées et solutions proposées : 6

5.2 Tests et horizons d'amélioration : 7

6 Notre interface pour l'application **8**

7 Conclusion **9**

1 Introduction

Le projet a pour objectif de vous initier à la manipulation des données des capteurs, à leur transmission via un réseau sans fil et à leur traitement sur une plateforme cloud. Dans ce contexte, nous allons développer un podomètre, une application qui permet de compter le nombre de pas effectués par l'utilisateur. Pour collecter les données, nous utiliserons un smartphone, assurant ainsi la connectivité via un point d'accès wifi. Le traitement principal, comprenant le calcul du nombre de pas, sera effectué par une application web hébergée sur un serveur, à savoir un PC connecté au point d'accès.

2 Rappel algorithme :

La plupart des smartphones modernes ont un accéléromètre intégré , c'est un composant matériel qui mesure l'accélération dans les directions x , y et z ; ces directions étant relatifs au téléphone portable. Au niveau logiciel , le framework Android 'sensor' fournit des classes qui permet de gérer les capteurs de l'appareil mobile.

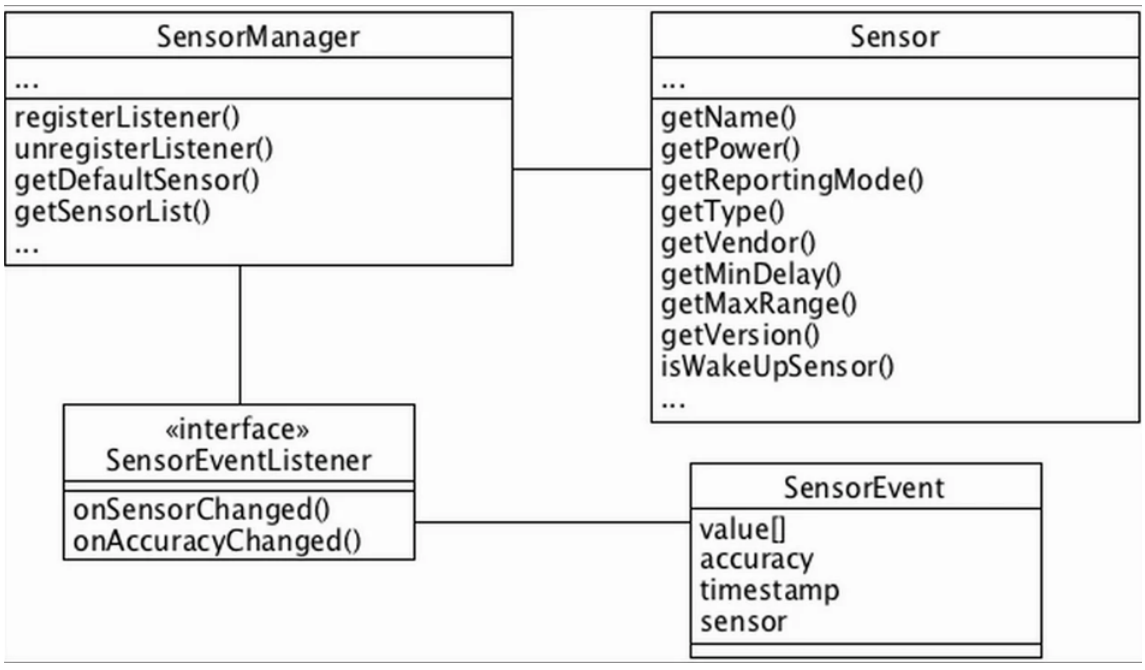

FIGURE 1 – les composants du framework sensor

On avait choisi de calculer la valeur de la fréquence d'échantillonnage par défaut au cours du projet en déclenchant un chronomètre , puis de laisser l'accéléromètre fournir des valeurs de a_z , puis d'arrêter le chronomètre. en divisant le nombre de valeurs échantillonnées par la durée

d'échantillonnage , on a obtenu *la fréquence d'échantillonnage* de l'accéléromètre.

Lorsqu'une personne marche, elle rebondit légèrement à chaque pas. On l'observe quand on voit le sommet de la tête d'une personne qui s'éloigne de nous. nous allons donc nous concentrer sur l'accélération suivant l'axe z , a_z . On va effectuer une transformée de fourrier sur le tableau des accélérations , puis on va sélectionner la fréquence fondamentale associée à l'amplitude maximale de la transformée , celle-ci n'est autre que la cadence de mouvement de l'utilisateur.

Le nombre de pas est donné donc par la relation :

$$\text{stepCount} = \text{cadance} \times \text{SAMPLING\_PERIOD} \tag{1}$$

où sampling period est la durée d'échantillonage.

3 Partie client

Nous avons créé une interface utilisateur conviviale permettant aux utilisateurs de visualiser les données du podomètre, telles que le nombre de pas effectués et le temps écoulé. L'interface utilisateur était intuitive et facile à utiliser. Cette partie client a été implémentée dans une classe appelée 'MainActivity'. Laissons de côté l'aspect de communication avec le serveur pour le moment , et voyons le fonctionnement générale de cette classe : Principalement , on exploite le framework 'sensor' pour extraire l'accélération a_z , puis on la stocke dans une liste appelée data.

Nous avons mis en place un mécanisme pour créer une requête HTTP contenant les valeurs de l'axe z capturées. Cette requête était préparée à l'aide de Volley. Une fois la requête HTTP créée, nous avons utilisé la bibliothèque Volley pour envoyer les données vers le serveur distant. Nous avons configuré les paramètres de la requête, tels que l'URL du serveur et les en-têtes appropriés pour garantir une communication réussie avec le serveur. On a créé une RequestQueue , et on lui a ajouté une 'Request' qui contient les paramètres de la requete HTTP POST , et une méthode qui précise ce qui doit être fait en cas de réponse par le serveur, et dans notre cas , c'était de visualiser à l'écran du mobile le nombre de pas calculé par le serveur

4 Partie serveur

Dans cette partie serveur, les données sont envoyées depuis le client vers une servlet dans l'API Apcahe TomCat. On développe la partie servlet de manière à ce qu'elle reçoit périodique-

ment les données a_z depuis le téléphone, c-à-d recevoir la requête "POST" envoyée par le client, et d'en extraire a_z. Cela était fait par la méthode doPost, donc à chaque fois que le client envoie une donnée, a_z le code dans la servlet commence par la stocker dans une liste data qui sera ensuite taité pour pouvoir calculer le nombre de pas.

On a défini un attribut entier appelé $SlidingWindow$, la période d'échantillonnage (Sampling Period) est initialisée avec une valeur cohérente avec la fréquence d'échantillonnage qu'on a déjà calculé , et la taille de la fenêtre Sliding Window, ensuite à chaque fois qu'on accumule " *Sliding Window* " données a_z , on calcule le nombre de pas pendant cette fenêtre.

On continue à stocker les données a_z dans la liste data jusqu'à ce qu'on accumule un nombre égale à *SlidingWindow modulo*(c-à-d égale à une, deux, trois,.. fois *Sliding Window*) données et puis on calcule les pas. Pour le calcul des pas, on commence par appliquer la transformée de Fourrier sur la liste data et on stocke le résultat dans une liste magnitudes. On fait un clear pour la liste data pour la vider et pouvoir recevoir de nouveau les données du client.

Une fois on a notre liste magnitudes on cherche le maximum de la liste qui correspond à la cadence des pas du client. Ensuite, on multiplie cette cadence par le *Sliding Window* pour avoir le nombre de pas parcouru pendant cette fenêtre.

$$\text{stepCount} = \text{cadance} \times \text{SAMPLING\_PERIOD} \tag{2}$$

On ajoute ce nombre de pas calculé dans cette fenêtre à une liste appelée iStep , dont le ième élément est le nombre de pas effectué pendant la ième sliding window. Et

Naturellement, le nombre de pas qu'il faut envoyer en retour au client en utilisant la librairie Volley , n'est autre que la somme des éléments de iStep.

5 Difficultés rencontrées et horizons d'amélioration

5.1 Difficultés rencontrées et solutions proposées :

1) Au début, on n'a pas su comment implémenter la transformée de fourrier car on avait un problème dans l'importation de FastFourrierTransformer, on avait du mal à la faire fonctionner : La solution qu'on a proposée c'est d'implémenter de zéro la fonction qui fait la transformée de Fourrier et de l'appliquer sur nos données jusqu'à ce qu'on a pu surmonter le problème d'importation de bibliothèque et on a pu utiliser la fonction déjà built-in

2) L'utilisation de deux téléphones aux même temps en communication avec le serveur : Pour

surmonter cette difficulté on a pensé à utiliser la notion des Handler, c-à-d que dans la servlet la calcul de pas de chaque client se fait séparément de l'écoute de l'autre client qui arrive. Malheureusement on a pas pu aboutir cette idée à cause des contraintes de temps.

5.2 Tests et horizons d'amélioration :

Dans le cadre de ce projet, nous avons réussi à accomplir avec succès la récupération des données des capteurs, leur envoi en temps réel vers un serveur distant et l'obtention des résultats en temps réel, les résultats étaient très proches des applications optimisées notamment Pedometer de AppStore avec laquelle on a comparé nos résultats en mettant les deux téléphones chacun dans une poche, la marge d'erreur était d'une moyenne de $\pm$ 5 pas, résultat qui était très satisfaisant pour nous, car la méthode utilisée n'était pas optimale comparée aux autres méthodes de calculs implémentées dans les applications déjà en service. Mais cela n'empêchait pas qu'on pouvait améliorer le calcul de pas pour être plus précis.

6 Notre interface pour l'application

FIGURE 2 – Interface de notre application

Dès qu'on active l'application en mode ON le calcule des pas commence et le timer qui s'affiche à côté était pour nous servir principalement pour nos calculs, mais on s'est dit pourquoi pas le garder pour développer l'application et calculer le temps de sport, de marche ou autres.

Cette capture d'écran est prise depuis le simulateur de machine virtuel d'android studio, on n'a pas pu mettre des vidéos ou d'autres photos de l'application, car on n'a pas de téléphone Android pour pouvoir refaire les tests, mais durant la présentation, on a réussi à atteindre l'objectif qui

était le calcul de pas en utilisant un serveur en temps réel.

7 Conclusion

Ce projet nous a permis de nous familiariser avec la manipulation des données des capteurs, la transmission sans fil et le traitement sur une plateforme cloud. En développant un podomètre, nous avons pu compter le nombre de pas effectués par l'utilisateur en utilisant un smartphone comme dispositif de collecte de données. La connectivité a été assurée par un point d'accès wifi, et l'application web hébergée sur un serveur a effectué le traitement nécessaire, y compris le calcul du nombre de pas. Cette expérience nous a permis de mieux comprendre les différentes étapes impliquées dans la mise en place d'une solution de collecte, de transmission et de traitement des données des capteurs.

La prochaine fois qu'on ferra du sport, on a une idée des énormes calculs qui se font dans nos poches pour calculer notre cadence !