PAO *smartcontrol*

-

Document de conception

BOURGEOIS Alex

8 mai 2016

Chapitre 1

Introduction

1.1 Présentation

Le présent document entre dans le cadre des Projets d'Approfondissement et d'Ouverture (PAO) du semestre 4.2 et est encadré par M. Alexandre Pauchet. L'objectif du projet est de développer un système permettant de contrôler une application Desktop par l'intermédiaire d'intéractions avec un smartphone Android.
Le smartphone est un serveur auquel se connecte un PC jouant le rôle de client et récupérant les états des différents capteurs présents sur le téléphone. Dans ce sens le smartphone sera utilisable comme contrôleur pour diverses applications grâce aux données de l'accéléromètre, du gyroscope, de l'écran tactile, etc. La preuve de fonctionnement du système sera le contrôle d'un objet 3D dont les mouvements seront associés à ceux du téléphone.

1.2 Choix techniques

1.2.1 Possibilités d'architecture

Les possibilités de fonctionnement étudiées sont les suivantes :

Le serveur émet en continu les valeurs des capteurs dont il dispose.
Avantages :

- Tous les capteurs disponibles sont émis par le serveur, pas d'erreur avec une demande par le client sur un capteur spécifique ;

Inconvénients :

- Le serveur va envoyer beaucoup d'informations dont certaines qui ne seront pas utilisées, ceci est coûteux en énergie pour un téléphone dont l'autonomie est limitée ;
- Le client peut manquer des informations et devoir attendre que le serveur émette à nouveau les valeurs du capteur. Ceci induirait une perte de temps et réduirait la réactivité du système.

Le client demande au serveur d'émettre les données d'un unique capteur, puis ce dernier envoie les données en continu. **Avantages :**

- Le client ne perdra plus de temps à attendre que le bon capteur soit émis ;

Inconvénients :
- Le serveur va potentiellement envoyer des données qui ne seront pas traitées par le client car elles pourraient être émises trop rapidement. On a toujours un gaspillage d'énergie ;
- Le client peut demander un capteur inexistant sur l'appareil, entrainant un dysfonctionnement du programme.

Afin de concilier le meilleur de ces solutions, le fonctionnement est le suivant :
Le serveur attend la connexion d'un client, ce-dernier spécifie le capteur dont il souhaite récupérer les données. Pour obtenir ces données, le client envoie une requête à laquelle le serveur répond.

Avantages :
- La consommation d'énergie côté serveur sera gérée efficacement, aucune information ne sera envoyée si le client n'en fait pas la demande explicite ;
- Le client ne manquera aucune donnée.

Inconvénients :
- Le serveur devra envoyer un message d'erreur si une requête sur un capteur inexistant est faite.

1.3 Spécifications

1.3.1 Généralités

- La connexion entre le client et le serveur se fait en Bluetooth.
  Pourquoi Bluetooth ?
  Etant donné que le serveur se situera sur un appareil android, la notion d'économie d'energie est à prendre en compte car ces appareils fonctionnement sur batterie. Dans ce sens le Bluetooth est une technologie rapide et plus économe en énergie que le Wifi. De plus la liaison sera directe et ne sera pas encombrée comme pourrait l'être une connexion Wifi (de nombreuses trames circulant sur le réseau) ;
- Le système est développé entièrement grâce à la bibliothèque Qt.
  Pourquoi Qt ?
  Qt est multi-plateforme (windows, linux, OSX, android) ce qui facilite grandement le développement du serveur et du client par l'utilisation d'un langage unique : le C++. De plus Qt utilise le paradigme de la programmation évenementielle qui permet une meilleure lisibilité du code par l'utilisation de signaux et de slots. Le projet étant de développer une API, les classes seront facilement utilisables dans un autre projet en connectant simplement leurs slots et signaux ;

- La société Qt s'oriente vers la 3D avec la création d'un moteur de rendu 3D donc ce projet permettra d'en explorer quelques possibilités comme la manipulation de caméra via l'utilisation des données capteur issues du téléphone, ceci servira comme preuve de fonctionnement de l'application.

1.3.2 Protocole de communication

Il existe 4 types de requêtes entre le client et le serveur :
— Établir la connexion ;
— Spécifier le capteur souhaité ;
— Demander la valeur du capteur ;
— Fermer la connexion.

Établissement de la connexion

— Le client fait une requête de connexion ;

— Le serveur y répond ;

— Le client est connecté ou non selon la réponse du serveur.

Spécification du capteur

— Le client doit choisir entre les capteurs disponibles via l'IHM. Dans la version actuelle du code il existe 3 capteurs : Rotation (numéro 1), Gyroscope (numéro 2) et Accéléromètre (numéro 3). Le client choisit ensuite l'interval de mise à jour du capteur (pour son timer interne). La requête envoyée est de la forme "x y" avec x le numéro du capteur et y l'interval de temps (ce-dernier est envoyé à titre informatif au serveur, il ne l'utilisera pas) ;

— Le serveur répond "ready" si le capteur a été correctement créé et un message d'erreur si le capteur n'existe pas.

Demande de valeur

— Ayant précedemment choisi le capteur qu'il désirait, le client envoie simplement "value" au serveur. La fréquence d'envoie de cette requête est déterminée par la valeur d'interval saisie lors du choix du capteur ;
— Le serveur va répondre à cette requête par une réponse du type : "c : x y z" avec c : capteur choisi, x : valeur sur l'axe X, y : valeur sur l'axe Y et z : valeur sur l'axe Z.

Fermeture de la connexion

— Le client fait une requête de fermeture ;

— Le serveur y répond et clot la connexion.

Chapitre 2

Architecture

2.1 Diagrammes

— Diagramme de séquence

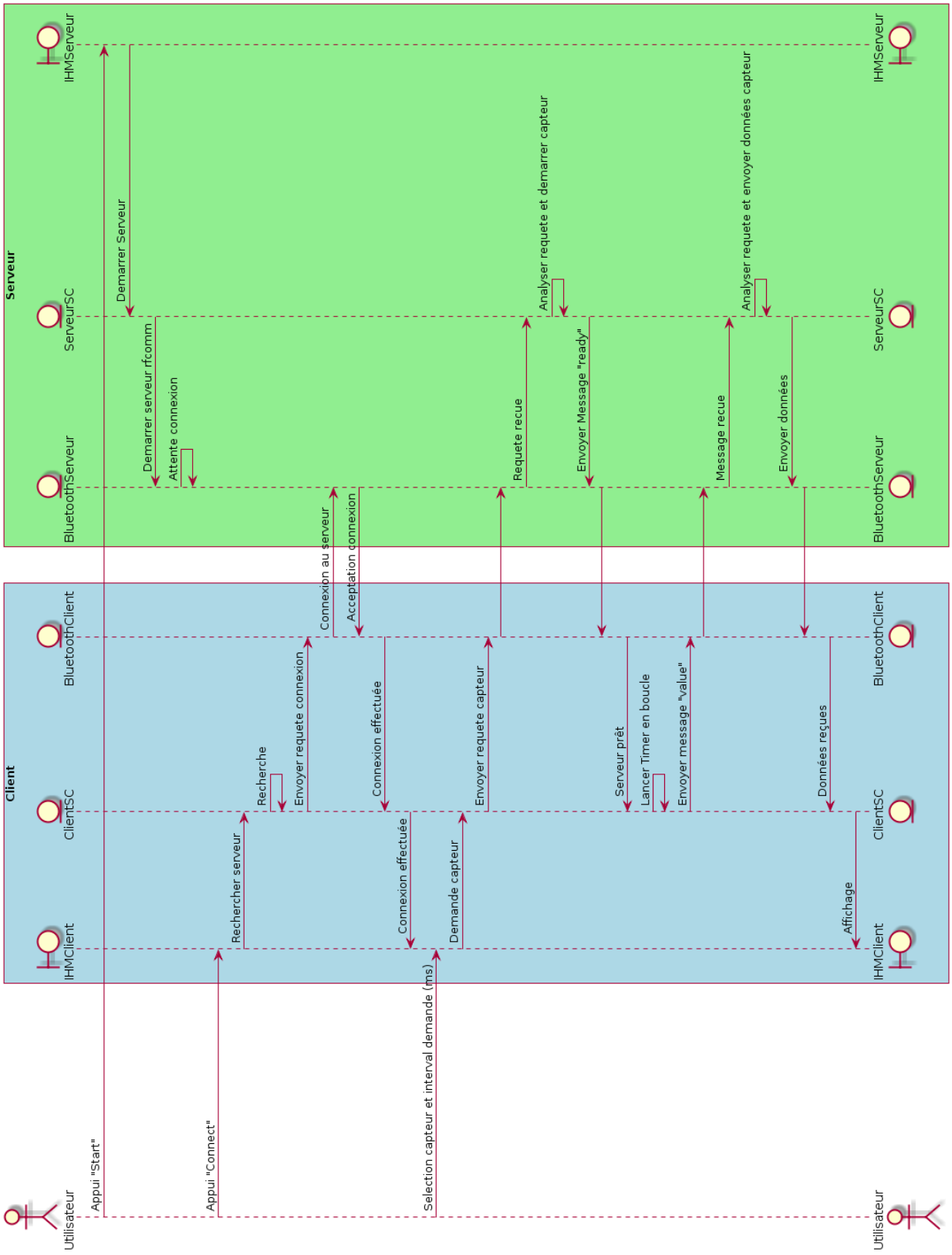

FIGURE 2.1 – Diagramme de séquence

— Diagramme de classe

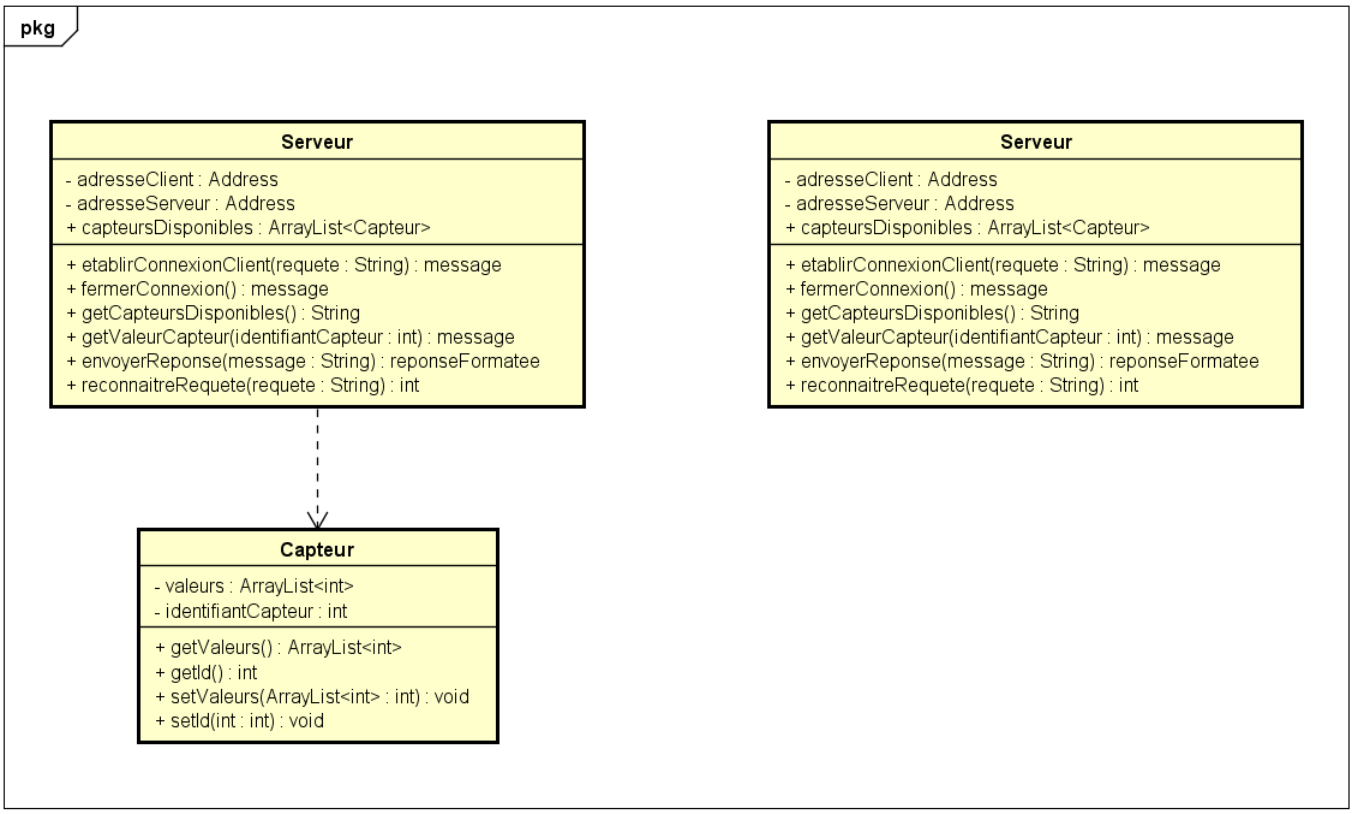

FIGURE 2.2 – Diagramme de classe

2.2 Client

Analyse descendante du Client

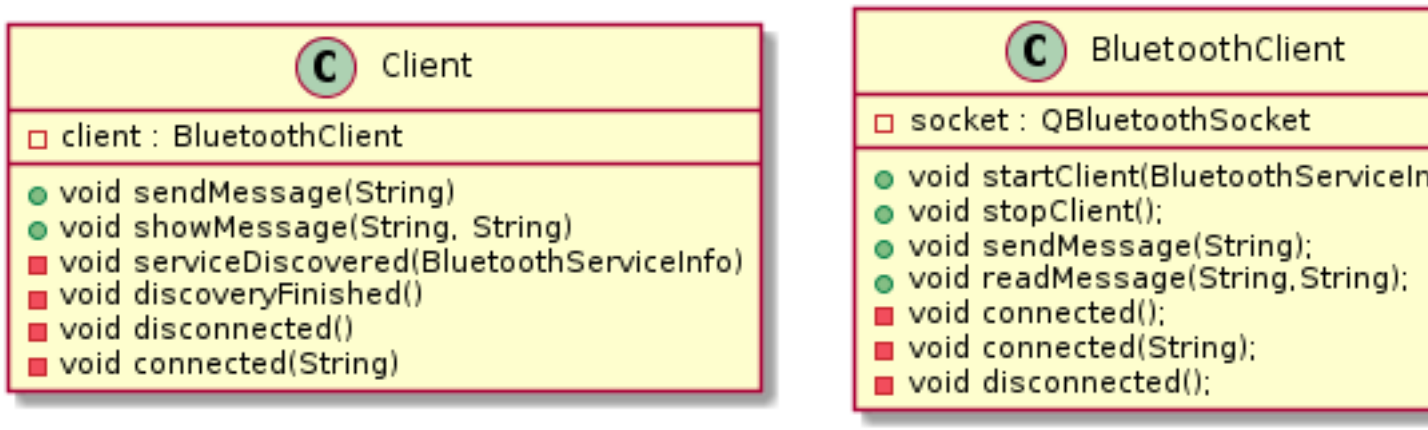

FIGURE 2.3 – Diagramme de classe Client

2.3 Serveur

Analyse descendante du Serveur

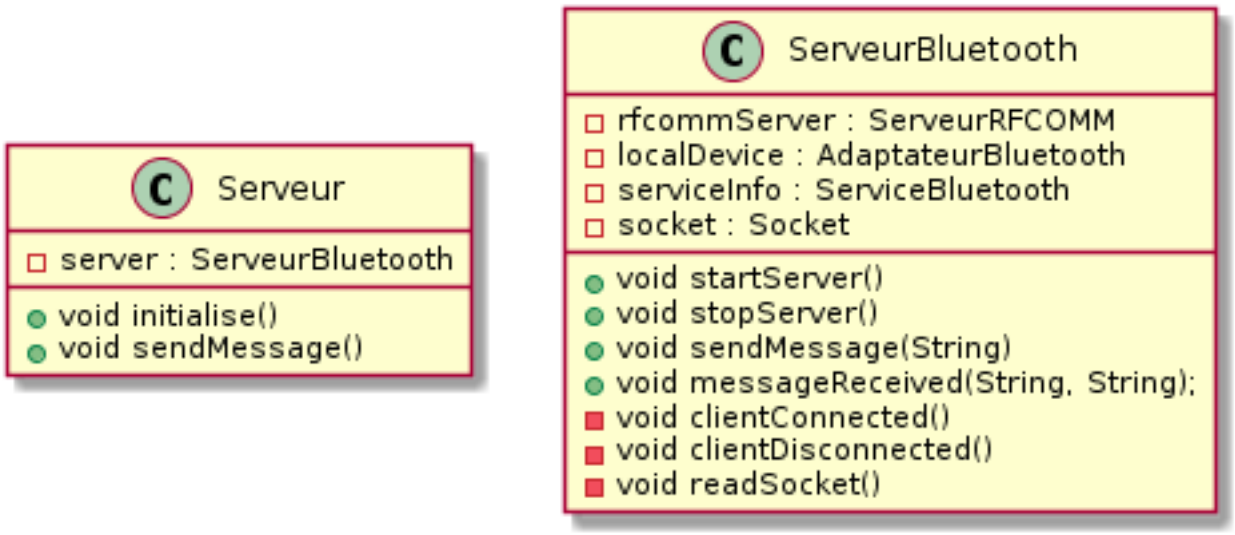

FIGURE 2.4 – Diagramme de classe Serveur

2.4 Conclusion

Le système est fonctionnel en l'état. La manière dont il est conçut permet de l'intégrer facilement dans un projet. Toutes les intéractions se font avec une unique classe que ce soit côté client ou côté serveur. L'ajout de nouveau capteur est relativement aisée, il suffit de rajouter dans le code les capteurs dans le switch du traitement des messages. Les différents tests que j'ai effectué avec plusieurs smartphones (bas de gamme et haut de gamme) montrent des différences de performances. Certains téléphones ne semblent pas pouvoir accéder à la même vitesse à leur capteur : sur certains bas de gamme il est possible de lire plus rapidement les données que sur d'autres plus haut de gamme. Cela dépend vraiment du téléphone et je n'ai pas réussi à expliquer ces différences. Il pourait être intéressant d'installer des versions différentes d'Android sur un même téléphone pour savoir si ce problème vient de la ROM utlisée ou des capteurs internes.