Dossier Station Météo

Leonardo Saba

Avril-Mai 2021

1 Cahier des charges

1.1 Présentation générale du projet

L'objectif de ce projet alliant informatique logicielle et informatique matérielle est de réaliser une station météorologique capable de donner des données telles que la température, l'humidité, la luminosité ou encore la qualité de l'air, tout cela visible depuis un terminal de réception des données, qu'il soit sur un poste fixe ou depuis un mobile, impliquant la création d'une application graphique sur mobile aidant à un meilleur confort utilisateur.

1.2 Besoins et contraites liés au projet

1.2.1 Besoins

Comme dit en présentation, il est nécessaire de pouvoir assurer un suivi et une excellente fiabilité des données à travers l'interface utilisateur mobile, en soit une application mobile compatible Android. La communication entre ces différents appareils doit se faire par l'intermédiaire d'un protocole réseau nouveau assurant et optimisant les données qui y sont acheminées.

1.2.2 Contraintes

Le langage disponible sur les cartes Arduino est le langage Arduino, *Arduino language*, basé sur un framework C++. Pour ce qui est du terminal mobile, l'application sera faite sur *Android Studio*. Cette environnement de développement intégré (IDE) ne supporte que le langage *Java*, le nouveau et populaire *Kotlin* ou encore du code natif en *C++*. Le plus simple et le plus connu et surtout le plus documenté étant *Java*, notre choix se porte donc sur ce langage. L'infrastructure réseau sera faite avec ce même langage pour assurer une meilleure rapidité avec la sérialisation des packets à partir de *Socket*. Le choix du langage *Python3* aurait rendu l'infrastructure réseau plus fragile car étant basée sur du flux textuel, et plus lente car entraînant une conversion vers un type de donnée "universel" comme *JSON*.

2 Démarche collaborative

Dans le cadre de ce projet, nous avons pu distinguer trois parties distinctes. La première étant l'informatique matérielle et logicielle requise pour le travail sur l'Arduino, la deuxième étant le travail sur l'infrastructure réseau assurant la communication entre différents appareils et enfin la troisième étant l'applicatif Android assurant le suivi de la station.

Les deux premières parties seront faites par Leonardo et la dernière par Esther.

De plus, le code source du projet sera publié sur la platforme de travail collaboratif *Github* afin de faciliter les étapes de mise en commun. Le code source sera publié gratuitement avec une licence MIT.

3 Avancement du projet ainsi que les difficultés rencontrées

3.1 Partie Arduino

3.1.1 Partie informatique matérielle

L'utilisation des capteurs *Grove* de *Seeed Studio Electronics* est pratique au vu de leur simplicité d'utilisation ainsi que d'une riche documentation sur Internet. L'appareil est contrôlable à travers un branchement au port série menant à un port USB sur l'ordinateur. Il a été rajouté par la suite un afficheur LCD 16x2 ainsi que d'un bouton pour créer une interface d'affichage affichant les différentes données des capteurs directement depuis l'appareil, le bouton servant à jouer une animation faisant défiler les différentes données.

Cependant, il y eu quelques problèmes par rapport au capteur d'humidité, celui-ci ne fonctionnant absolument pas à conditions réelles d'humidité quasi-totale. Enfin, il y eu besoin de changer de carte Arduino, nous pensions que rajouter un module Bluetooth de contrôle serait faisable dans les temps, nous y avons renoncé tout en gardant la nouvelle carte.

3.1.2 Partie informatique logicielle

Comme dit précédemment, le choix des capteurs c'est faite aussi sur la qualité de la documentation fournie, le code étant par la suite bien plus simple et intelligible. Certains capteurs ont nécessité des bibliothèques annexes pour pouvoir fonctionner correctement, c'est notamment le cas des capteurs de qualité de l'air ainsi que d'humidité. L'afficheur LCD a eu lui aussi besoin de quelques bibliothèques.

3.2 Partie réseau

3.2.1 Serveur

Le serveur est un *ServerSocket* provenant donc de *Java*, celui-ci est basé sur le protocole *TCP*. Le serveur démarre sur deux phases d'initialisation. La première est la connexion au port série grâce à au *Character Device* présent à l'adresse */dev/ttyACM0*, ce fichier pouvant être lu tel un fichier normal, sauf que la lecture ce fait directement sur le port série. La seconde étape est de démarrer le *SocketServer* et d'initialiser les composant de gestion du serveur tel *ObjectInputStream* et *ObjectOutputStream*. Une infrastructure démarre aussi au même moment, la création de *callbacks* servant à utiliser le fichier serveur comme une bibliothèque, la rendant utilisable sur d'autres projets et utile notamment sur l'applicatif Android. Une fois le serveur initialisé, il va créer des objets contenant en attribut toutes les données météorologiques et l'envoyer à tout les clients dans un délais défini. À noter que le protocole défini par le serveur n'implémente pas de chiffrement, les données circulant étant en clair sur le réseau.

Une difficulté certaine est d'appréhender la concurrence des donnée, un exemple serait la liste contenant les clients connectés au serveur. Si un client se connecte au serveur et qu'au même moment un autre se déconnecte (ce qui est probable, le risque augmentant avec le nombre de clients connectés), la liste *ArrayList* ne pourra pas supporter ces opérations et va entraîner une *ConcurrentModificationException*. Une solution simple pour régler ce problème est d'utiliser une *CopyOnWriteArrayList*, un type de liste supportant la concurrence.

3.2.2 Client

Le client reçoit les paquets et récupère les données météorologiques pour ensuite envoyer dans un système de *callbacks* ces mêmes données. L'utilisateur de la bibliothèque peut alors simplement les récupérer de manière asynchrone.

Un problème constaté est en lien avec l'applicatif Android, celui ayant une politique rapide de déconnexion des *Socket* en cas de saturation de mémoire vive ; il a donc fallut implémenter un système de reconnexion automatique et de vérification des clients à partir de données de test, il a donc été implémenté un système de réponse au ping sur le serveur. Ainsi il est possible de connaître l'état et le ping de n'importe quel client grâce à cela et en temps réel.

3.3 Partie Android

Esther a réalisé avec succès cette partie, mais n'a pas encore les étapes et les difficultés de son travail.

4 Vidéo sur le rendu final

En cours de production.

5 Sources

Toutes les sources sont disponible sur la page Github : `https://github.com/esthheeeeer/station_meteo`