Ultrateam - Rapport de projet

Anthony Geourjon - Clément Rouquier

Polytech Grenoble - RICM4

Résumé Ce rapport s'attache à synthétiser les objectifs, ainsi qu'expliquer le travail réalisé lors du projet UltraTeam dans le cadre du projet de S8 de RICM4. Le projet a été réalisé par deux binômes. Le premier binôme (nous) est Clément Rouquier et Anthony Geourjon et le second Alexandre Ferrera et Romane Gallier. Nous avons travaillé ensemble sur des parties différentes du projet. Le projet est disponible ici : `https://github.com/ultratrail/mobapp2`

Table des matières

Introduction

Le projet de semestre 8 de la formation RICM est une réelle opportunité de mettre en pratique les connaissances acquises et la méthodologie d'auto-apprentissage sur de nouvelles technologies. Disposant d'un volume horaire important et de créneaux dédiés, il est le plus gros projet que nous ayons jamais mené.

La situation de notre groupe de projet a été relativement floue car nous sommes deux binômes à travailler sur le même projet et parfois nous faisons des choses en commun (soutenance préliminaire) et parfois séparément (ce rapport et la soutenance en commun). L'encadrant et tuteur du projet est Didier Donsez. Veuillez noter que ce flou artistique autourde nos conditions de travail et d'évaluation a été un réel problème.

Notre projet s'appelle Ultrateam, c'est une application dédiée aux amateurs de sports en pleine nature. Le projet sera décrit de façon détaillée dans la partie suivante. Ensuite, nous verrons le déroulement du projet puis les résultats obtenus. Enfin nous expliquerons succinctement son fonctionnement.

1 Description du projet

1.1 Principe du projet

Le projet s'inscrit dans une problématique d'absence de couverture cellulaire dans les espaces reculés, tels que les montagnes ou les forêts. Les humains évoluant souvent en groupe dans de tels environnements, et doivent donc trouver des solutions afin de s'adapter à la potentielle absence de réseau classique pour continuer à communiquer entre eux. Nous pouvons citer les trailers, les chasseurs ou encore les bûcherons qui évoluent dans de tels milieux.

Un des outils apprécié est par exemple le talkie-walkie, qui permet, en affectant un terminal à chacun des membres du groupe, de créer un réseau ad hoc, permettant aux membres de pouvoirs continuer à communiquer entre eux.

Le projet Ultrateam se base sur le même principe que le talkie-walkie. Nous partons du principe que chacun peut aujourd'hui disposer d'un smartphone disposant de capacité acceptable. De fait, il peut potentiellement être utilisé pour communiquer dans les zones disposant d'une couverture réseau suffisante. Le problème des zones hors-couverture sera résolu par l'utilisation de modem LoRa connectés aux smartphones. Les modems LoRa sont des boîtiers électroniques faiblement encombrants et de faible consommation disposant d'une connexion USB et LoRa. L'utilisation de tels boîtiers nécessite que le smartphone soit USB OTG.

La difficulté du projet réside dans la foultitude et la très grande diversité des choses à inclure dans le projet. Citons évidement le LoRa, les cartes, l'abonnement à un serveur web, la connexion avec une ceinture ou une montre Bluetooth, etc.

Passons maintenant au fonctionnement de l'application. L'idée est qu'un groupe de personne puisse se réunir en groupe sur l'application, chaque groupe possédant un chef. Chacun des utilisateurs pourraient communiquer de façon unilatérale ou bilatérale avec les autres (en fonction de leur équipement, que nous détaillerons plus tard) Les informations qui pourront être échangées seraient, au départ, d'information de localisation, de notifications d'urgence en cas de problèmes ainsi que d'informations supplémentaires acquises à l'aide d'équipements optionnels connectés en Bluetooth Low Energy avec le smartphone de l'utilisateur.

Le groupe est créé de façon virtuelle grâce à une des 2 technologies :

— En cas de couverture cellulaire disponible, un groupe Mqtt est créé, et les utilisateurs de ce groupe peuvent chacun s'échanger des données à l'aide des réseaux cellulaires
— Au contraire en l'absence de couverture cellulaire, les données sont échangées grâce au réseau LoRa, qui a les avantages combinés de très peu consommer tout en ayant une portée d'une dizaine de kilomètres, et ce même en environnement accidenté, comme en montagne. Un groupe LoRa est simplement créé par écoute d'une certaine fréquence

Les possibilités d'évolution sont nombreuses, notamment d'un point de vue du catalogue des données échangées. On peut en effet penser à un chat textuel, vocal, média..

Le public intéressé pour notre application peut être multiple. On peut en effet penser à un groupe de chasseur, désirant connaître de façon fiable et en permanence la position des autres membres d'une battue par exemple. De même, des organismes comme l'UCPA (l'Union nationale des Centres sportifs de Plein Air) peuvent être intéressés, car cela permettrait à leurs encadrants de gérer un groupe facilement en milieu extérieur. A l'aide de systèmes comme celui-ci, les groupes d'enfants pourraient avoir plus de libertés lors des randonnées par exemple, et on peut notamment imaginer de grandes parties de cache-cache, améliorant l'aspect ludique que peut avoir la montagne pour les jeunes-ados, qui n'ont pas forcément envie d'avoir toujours un moniteur derrière eux.

La fiabilité de notre application revet pour nos utilisateurs, une importance vitale. En effet, si un encadrant de randonnée compte dessus pour gérer une sortie, elle ne doit en aucun cas être défaillante. Cette contrainte amène plusieurs problématiques :

— La communication du réseau doit être assurée en permanence, quel que soit l'environnement, tout en restant à une distance raisonnable et définie dans les conditions générales. On peut par exemple imaginer une alerte pour prévenir lorsqu'un membre s'éloigne trop et commence à perdre le signal.
— La consommation de batterie doit être la plus réduite possible, étant donnée que les smartphones sont déjà des appareils embarqués avec de grosses contraintes énergétiques, étant donné que le poids de ceux-ci est un enjeu de taille. De ce fait, toute les technologies utilisées doivent prendre en compte cette contrainte. D'un point de vue de l'affichage, on peut par exemple penser à utiliser un thème majoritairement noir, afin de limiter la consommation d'énergie sur les écrans amoled (Un pixel noir étant un pixel éteint avec cette technologie). De même l'utilisation du réseau ou du Bluetooth ne doivent entrainer de surconsommation excessive. Il est donc essentiel minimiser les communication et d'utiliser des technologies tels que le Bluetooth Low Energy, implantée dans l'API 18 d'Android.
— Le volume des données échangés doit être le plus limité possible afin de ne pas dévorer le quota de données permit par le forfait de l'utilisateur.

L'application sera dans un premier temps uniquement déployée pour les terminaux Android, avec une API au minimum de 22. Cela correspond à une version égale ou supérieur à Android 5.1.1 et cela représente plus de 75% des smartphones au jour du 6 mars 2017. Notre application n'effectuant pas de traitements lourd, n'importe quel smartphone sera en mesure d'exécuter notre application si celui-ci dispose evidement d'un port USB OTG. Un port USB OTG est simplement un port capable d'alimenter l'appareil qui lui est connecté, dans notre cas le modem LoRa.

Voici les différents niveaux d'équipements nécessaires pour les utilisateurs :

— Chef de groupe : devra obligatoirement posséder un smartphone et un boîtier LoRa et peut posséder des équipements supplémentaires communiquant en Bluetooth.
— Membre de groupe (équipement minimal) : une Loramote pour la position GPS, impossibilité d'utiliser des équipements supplémentaires
— Membre de groupe "amélioré" : un smartphone ainsi qu'un boîtier LoRa. Il peut également, de façon optionnelle, disposer d'un équipement Bluetooth, afin d'étoffer le catalogue de données qu'il envoit.

Il est obligatoire que les smartphones possèdent une connexion aux données mobiles au moins au lancement de l'application afin de télécharger la carte. Il est possible de penser à une évolution afin d'éviter cette contrainte, en prétéléchargeant les cartes. Dans ce cas la, seul un boîtier LoRa serait nécessaire.

Dans l'autre sens, et même ci cela dénature totalement le principe de l'application, les utilisateurs peuvent très bien se passer de boîtier de LoRa. Dans ce cas la, il est nécessaire que tous possèdent un smartphone, et la fiabilité ne pourra plus être garantie hors des zones de couvertures réseaux classique.

1.2 Objectifs à réaliser

Nous avons identifié plusieurs priorités dans ce projet. La priorité est l'affichage d'une carte permettant de se géolocaliser ainsi que visualiser les autres membres du groupe. Cela sous-entend donc qu'une gestion d'un groupe (nous définirons la notion de groupe plus tard) soit faites et que nous puissions communiquer avec les autres membres du groupe soit par l'intermédiaire d'un serveur grâce au réseau classique, soit en LoRa. Le LoRa implique également, qu'une connexion série USB puisse être utilisé pour communiquer avec le modem et qu'un protocole de communication soit établie..

De façon moins prioritaires, nous devrons réussir à interagir avec des équipements Bluetooth comme les ceintures et montres cardios et inclure les données récoltées à la carte. Nous disposons également de LoRamote. Il s'agit de boîtiers disposant de leurs modems LoRa ainsi que de leurs propres batteries. Ces boîtiers sont autonomes et peuvent donc être donnés à des personnes ne disposant pas de smartphone.

A cela, nous pouvons rajouter, la gestion des SOS (une personne envoi un SOS, les personnes sont prévenues et relais elle aussi pour que chacun soit prévenu), une gestion plus avancé de la carte et des données récoltées. Par exemple,

le chef du groupe pourrait disposer d'un tableau de bord indiquant la distance parcourus par chacun des membres, leurs éloignements maximales, etc.

2 Réalisation du projet

2.1 Conception et prise en main du sujet

Tout d'abord nous avons commencé à isolé les besoins des utilisateurs finaux de l'application afin de voir les fonctionnalités à développer. Ensuite nous avons regardé les applications existantes, et même si plusieurs s'en rapprochaient, nous n'en avons trouvé aucune qui correspondaient réellement à notre sujet.

A partir de la, nous avons du faire des choix techniques pour savoir ce que nous allions développer. Nous avions 3 choix face à nous :

— Partir de rien et développé une application Apache Cordova (en Javascript/HTML donc)
— Partir de l'application native OSMAND et la modifier pour qu'elle colle à notre sujet.
— Développer une application native *from scratch*

Une application Cordova n'étant pas adapté à nos besoins, nous nous sommes tournés vers le natif. Nous avons essayé de prendre en main OSMAND mais celle-ci s'est révélé beaucoup trop complexe et usine à gaz pour nos besoins. C'est donc naturellement que nous avons décidé de partir d'un application vierge.

2.2 Travail réalisé

Bluetooth

Le service Bluetooth est divisé en 4 classes :

— BluetoothActivity : Il s'agit de l'activité de base pour le Bluetooth. Elle est créée dès qu'un utilisateur clique sur le bouton "bluetooth" présent dans l'activité principale. Cette classe s'assure de la compatibilité de l'appareil avec le Bluetooth Low Energy, et active le Bluetooth si ce n'est pas déjà fait. Ensuite, elle s'assure du scan des dispositifs BLE environnants, et affiche la liste, référencés par le nom si celui ci est fourni ainsi que l'adresse MAC.

— DeviceControlActivity : Activité lancée à partir d'un clic sur le nom d'un dispositif présent dans la liste générée par BluetoothActivity. L'activité essaye automatiquement de se connecter au dispositif trouvable à l'adresse MAC passée par la précédente activité. Si il y arrive, il montre toutes les données disponibles. Actuellement, on ne peut écouter qu'une seule donnée en même temps. Il serait aisé de pouvoir à la place s'inscrire à toutes les données qu'on veut écouter, afin de pouvoir envoyer plus d'informations à partir d'un même dispositif.

— Défini en dur les correspondance entre les identifiants de services (norme Bluetooth) envoyé par le dispositif auquel on est connecté en string. Pour le moment, tous les services sont affichés, que la correspondance id -> nom soit connue ou non. Avec une construction de classe telle, il serait extrêmement facile de n'afficher que les services gérés par l'application.

— BluetoothLeService : Une fois la connexion établie, cette classe gère le service afin de continuer à recevoir les informations à chaque fois qu'une mise à jour est disponible, et ce même en changeant d'activité, ce qui du coup permet de récupérer les données lorsqu'on regarde la carte (et donc de pouvoir les envoyer)

Cette construction se base sur le code disponible ici : `https://github.com/googlesamples/android-BluetoothLeGatt/`. Ce code est sous la licence Apache 2.0, et notre code doit donc suivre les contraintes définies par celle-ci, bien qu'elle ne restreigne pas la distribution.

Toutes les Activités désirant utiliser les informations reçues par les dispositifs Bluetooth connectés doivent lancer registerReceiver(...) , en faisant bien attention que le service soit effectivement lancé et connecté. Pour se faire, deux variables sont définies et mise à jour en temps voulu :

```
public static boolean BLUETOOTH_SERVICE_ACTIVE;
```

→ Variable globale mise à jour lors d'un changement d'état de la connexion.

```
private boolean BLUETOOTH_SERVICE_REGISTER;
```

→ Variable locale définie dans chaque Activité désirant souscrire au service, et mise à jour à chaque register/unregister.

Le problème majeur lors de l'élaboration de ce service a été d'arriver à implanter le code du GoogleSample (Dont le Bluetooth est l'activité principale) dans notre application dont le squelette était déjà fixé, et le Bluetooth en était donc une activité secondaire. De fait, le sample utilisait une ListActivity, et dans notre cas il fallait se servir d'une AppCompatActivity pour que ça reste compatible avec le reste de l'application. Sauf que n'ayant que de faibles connaissances en développement Android, il a été très dur d'identifier le problème (étant donné que sa résolution nécessitait d'autres modifications dans le code), et nous avons errer entre toutes les solutions proposées sur internet, sans forcément trouver la bonne.

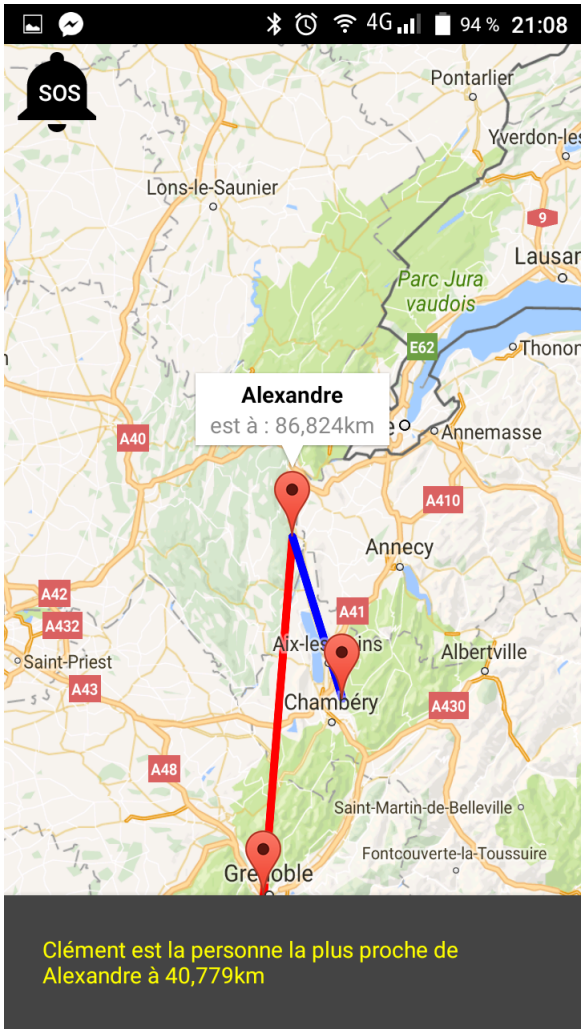
Ceci nous a fait perdre un temps considérable, et ayant à titre personnel découvert Android avec ce projet, je trouve que les problèmes de ce type et leur difficulté de résolution est l'une des distances sémantiques les plus importantes quant à l'accès à cet écosystème.

Maps et localisation

Android étant un système d'exploitation encore assez jeune (sortie en 2007) qui évolue encore beaucoup et donc son API change encore assez régulièrement.

Ainsi de nombreux exemple de code fonctionne très bien avec certaines versions alors que des SDK plus récents les font crasher ou ne compile simplement même plus.

Nous avons toutefois réussi à afficher une map ainsi que des markers (les fameux points rouge de Google) afin de pointer certains points de la carte. Nous pouvons interargir avec eux, et cliquer sur eux permet d'obtenir des informations supplémentaires telles que la distance entre ce point et le téléphone ou la personne la plus proche de nous.

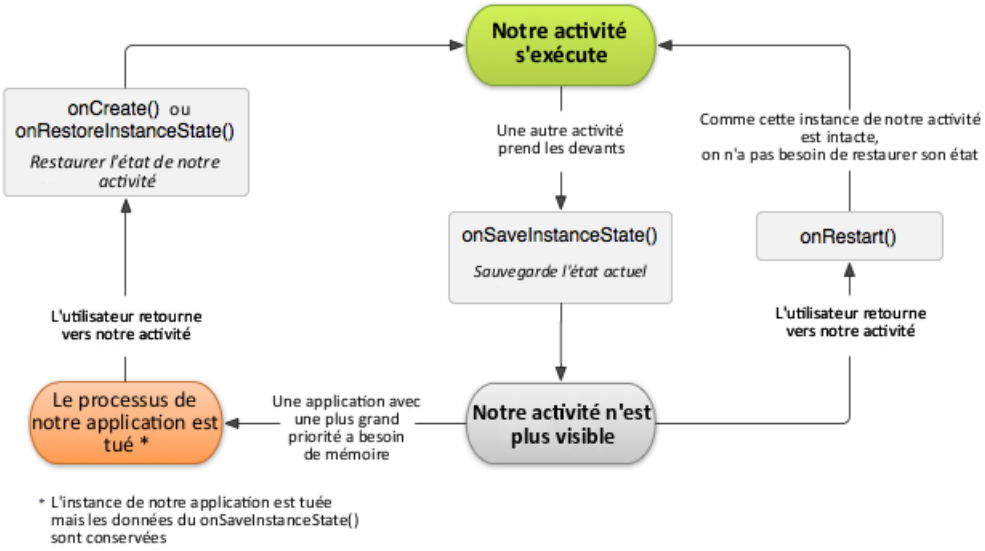


Grâce à la carte nous voyons qu'Alexandre est à 86 km du propriétaire du téléphone et qu'en cas de problème Clément est la personne la plus proche de lui à 40 km. Le bouton SOS en haut à gauche n'est pas encore fonctionnel, le serveur mqqt ou les boîtiers LoRa n'étant pas encore utilisable.

Toute la gestion des maps est réalisé dans l'activité MapsActivity. Une carte de type Google Map est relativement dur à gérer car elle doit être chargé dynamiquement dans un fragment. De plus, une activité (et plus généralement une application) dispose d'un cycle de vie complexe. Par exemple, si on utilise l'application, que l'on bascule sur une autre application et que l'on revienne il faudra recréer la carte seulement si c'est nécessaire (onRestart).

Du fait de l'absence de serveur et de communications LoRa fonctionnelles, les personnes (autre que le possesseur du téléphone) sont disposé aléatoirement sur la carte. Cela nous a permis d'outrepasser le problème des communications et de développer comme si de rien n'était. Grâce à ce subterfuge, nous pouvons montrer que la carte fonctionne ainsi que ses utilisations.

Comme pour le Bluetooth, Google Sample aura encore été notre meilleur allié : `https://github.com/googlemaps/android-samples`.



USB

L'interaction entre les modems LoRa et le téléphone se fait au moins d'un lien série USB. Pour communiquer avec le modem nous devons envoyer des commandes au format texte. Par exemple pour configurer l'antenne en réception, nous utilisons la commande *radio rx 0*.

Nous avons également implémenté des listeners afin d'être notifié en cas de communication asynchrone du modem. Les structures de données sont gérées pour attendre un message en USB et utiliser les données qu'il transporte dans l'affichage des marqueurs sur la carte.

Cette fois-ci c'est GitHub qui sera venu à notre secours avec ce dépôt très bien fait : `https://github.com/RomSand/https-github.com-felHR85-UsbSerial`.

Fusion des parties

La fusion des parties a posé de gros soucis, qui seront expliqués lors de la présentation. De même, un coup d'oeil sur le GitHub permet de s'en rendre compte

3 Utilisations et fonctionnements

Au lancement de l'application : Il faut au préalable sélectionner un groupe. Pour se faire, aller dans l'onglet config groupe. Ici, un groupe peut être défini en local en mettant le nom de chaque membre et appuyer sur "Add" ainsi que récupéré depuis un serveur (dans l'onglet /master/groupes/) du github, en mettant l'ID à récupérer et en appuyant sur "import". Pour le moment, seul le groupe "test" existe sur ce serveur

Une fois ceci fait, il est possible de sélectionner un service Bluetooth. pour le moment, seul un service de Health Equipement est implémenté. Allez dans l'onglet config bluetooth, puis "scan" sur certains téléphones, il faut réitérer le scan un grand nombre de fois afin de détecter les dispositifs disponibles. Une fois trouvé, cliquez dessus, puis sélectionnez Health Equipement puis Health Service. Vous verrez que les data sont bien reçues dans le champ "data". Une fois ceci fait, retourner sur main.

On peut maintenant lancer la carte. Il y aura un nombre de marqueurs égal à la taille du groupe défini, avec comme label le nom des membres du groupe. Chacun à pour le moment une position random, qui sera par la suite récupérée par mqtt/usb(LoRa). Lors d'un clic sur un marqueur, des informations sont données à l'aide de lignes et de distance quand à la distance entre le marqueur et le chef, et la distance entre ce marqueur et l'autre marqueur le plus proche. Si une connexion bluetooth a été définie, le marqueur chef va bouger sur la carte de façon aléatoire. C'était simplement afin de donner un exemple de broadcastReceiver, étant donné que le mqtt/LoRa n'est pas fonctionnel.

Conclusion

Dans le cadre de notre projet de fin d'année nous devions développer une application pour des personnes évoluant en pleine nature et souhaitant communiquer en toute circonstance. Après une phase de réflexion et de conception, nous nous sommes lancés dans son développement. Même si les communications ne fonctionnent pas avec le serveur, l'application dispose de toutes les primitives nécessaires à cela et il ne resterais que peu de travail à faire si nous disposions d'un serveur fonctionnel. Nous retiendrons plutôt la bonne gestion du Bluetooth, de l'USB, de la carte ainsi qu'un design graphique de l'application plutôt réussi.

Durant ce projet, notre binôme a bien fonctionné et nous avons réussi à travailler en parallèle quasiment sans soucis grâce à GitHub. Même si tout ne fonctionne pas, ce projet a été une bonne expérience de développement. Il nous a permit de découvrir l'écosystème Android et la technologie Lora ainsi que la gestion d'un gros projet à plusieurs et les problèmes que cela impliquent.