POLYTECHNIQUE MONTRÉAL

TECHNOLOGICAL UNIVERSITY

INF8405 - Informatique Mobile

Hiver 2020

Polytechnique Montréal

TP #2

Application de détection d'appareils Bluetooth – Bluetooth Analyzer

Soumis à **Bilal Itani** le 3 avril 2020

Par :

Sami Tamer Arar, 1828776

Aurélien Bouis, 1787523

Julien Legault, 1847125

Table des Matières

Introduction

Le but de ce laboratoire était de développer une application mobile permettant à son utilisateur de géolocaliser les potentiels appareils à proximité pouvant être pairés grâce à une connexion Bluetooth. L'objectif académique de ce laboratoire était d'approfondir les compétences en developpement Android et utiliser les fonctionnalités matérielles offerts par les appareils Android, tel que le GPS, le Bluetooth et le WIFI. Ce travail avait aussi pour but de nous faire développer notre compétence en conception et en communication. En premier lieu, nous ferons la présensation des travaux. En deuxième lieu, nous communiquerons nos diffcultés rencontrés. Enfin, nous terminerons en disant notre avis sur le travail et finirons avec une conclusion.

Présentation des travaux

Tout d'abord le projet ne comporte que 2 activités. La première s'appelle *SplashScreenActivity*. La deuxième s'appelle *MainActivity*. Après 3 secondes sur *SlashScreenActivity*, l'utilisateur atterit sur l'interface de *MainAcitvity* qui est séparée en deux morceaux verticalement. Pour que l'application ne plante pas, il faut avoir autorisé les permissions GPS et Storage de l'application.(À partir des *settings* de Android ou *app info*.(*Permissions manager*))

MainActivity

Premier morceau : RecyclerViews

Premièrement, pour afficher de façon dynamique la liste des appareils récoltés par la recherche Bluetooth, il faut créer un *RecyclerView*. Ainsi, le projet inclut 3 *RecyclerViews*, soit 1 pour les résultats de recherche, 1 pour l'historique de recherche et 1 pour les favoris. Ainsi, on peut alors trouver 3 adapteurs personnalisés reliés à ces *RecyclerViews*: *AdapterScanFavorites*, *AdapterScanHistory* et *AdapterScanResults*. Il faut savoir que cette liste est mise dans un *Hashmap* afin d'éviter des appareils dupliqués et dont la clé est l'adresse *MAC* jugée unique à chaque appareil Bluetooth.

Deuxième morceau : Google Maps Android API v2

Deuxièmement, le deuxième morceau qui contient le service API de google maps est instancié et affiche seulement les éléments de la liste des résultats et non ceux de l'historique, ni des favoris. On peut interagir avec les éléments sur la carte. De plus, les éléments trouvés sont positionnés sur la carte de façon à établir un rayon de distance avec l'appareil Bluetooth trouvé. C'est-à-dire, que l'appareil qui s'affiche n'est pas nécessairement à la position géolocalisée, mais peut être partout sur la circonférence du cercle qui passe par l'élément en question.

Logique de l'application

Scan Bluetooth

Puis, sachant qu'il existe 2 types de Bluetooth : les *Low Energy* (*BLE*) et les Classiques, nous avons décidé de faire un scan classique, car, en même temps, il effectue également la découverte des *BLE*. Pour le scan, nous avons utilisé la libraire *AndroidBluetoothLibrary* : https://github.com/douglasjunior/AndroidBluetoothLibrary
De cette manière, on n'a pas eu à créer de nouvelles fonctions ou de classe pour gérer cela avec la librairie de Google. Aussi, on obtient facilement l'information de chaque appareil Bluetooth détecté par son objet *BluetoothDevice*. De plus, la libraire nous retourne également le RSSI de l'objet trouvé, c'est-à-dire l'intensité du signal Bluetooth.

Premièrement, grâce au BluetoothDevice retourné, on arrive à obtenir de l'appareil détecté, son nom, son adresse Mac, son uuids. Pour récolter le Class of Device, le statut de l'appareil (s'il est déjà jumelé ou pas), et le type de Bluetooth détecté (Classique ou LE), nous avons créé un interpreteur de résultats qui se trouve dans le dossier *utils*. La classe *BTInfoAnalyser* transforme alors les bits retourné par *BluetoothDevice* en un *string* lisible. Cette classe a été peuplée à l'aide de la documentation Google.

Deuxièmement, grâce au RSSI retourné, on peut estimer la distance de l'appareil Bluetooth. Pour cela, 2 fonctions ont été réalisées dans le fichier *DistanceCalculator.kt.* La première fonction *CalculateDistance*, calcule la distance en mètre à partir de l'intensité reçue. Pour cela il faut un point de référence. L'intensité moyenne à une distance de 1 mètre est alors estimée à 59dB. Ainsi, avec cette donnée, on peut estimer les autres intensités. Il faut garder à l'esprit que cela n'est aucunement précis et plusieurs facteurs rentre en jeu tels que les murs. L'algorithme est basé sur cette étude : https://www.radiusnetworks.com/2018-11-19-fundamentals-of-beacon-ranging

La deuxième fonction *newLocationOnMap* prend la position actuelle de l'utilisateur en longitude et en latitude, une distance en mètre, et une direction (ex : ouest) une nouvelle localisation. Ainsi, cette fonction permet alors de localiser, sur la carte, un appareil Bluetooth trouvé. L'algorithme est inspiré par cette étude : http://www.movable-type.co.uk/scripts/latlong.html

Il faut savoir que la direction, étant non connue, est alors aléatoirement assignée. Puisque l'élément Bluetooth apparait sur un cercle sur la carte, donc son rayon est connu. L'élément peut se trouver à n'importe quel endroit sur ce cercle, soit s'il apparait au nord, il peut se trouver réellement au sud, etc.

Géolocalisation GPS

Pour localiser l'utilisateur en récoltant ses coordonnées GPS, on a utilisé la librairie *Android-SimpleLocation* disponible sur : https://github.com/delight-im/Android-SimpleLocation

De cette manière, on peut avoir la localisation en temps réel, c'est-à-dire que si elle change, nous pouvons être notifiés à l'aide d'une fonction *listener*.

Base de données

Pour mettre en favoris l'appareil sélectionné ou pour mettre en historique les appareils découverts, nous envoyons un *broadcast* local à notre *MainActivity*. En effet, ce broadcast local est effectué à partir des adapter des recyclerviews. Ainsi la classe *DBManage*r se chargera de mettre en mémoire les appareils scannés (historique de scan) et les appareils mis en favoris.

Pour entreposer les appareils Bluetooth, on utilise la méthode *getSharedPreferences()* qui nous permet de créer une collection privée dans la mémoire de l'application. À l'intérieur de cette collection, nous pouvons appeler *edit()* sur la collection pour ensuite entreposer l'appareil bluetooth avec *putStringSet()* et de les associer à une clé qui serait l'adresse mac. Pour récupérer les données, on utilise *getStringSet()* en passant la clé des valeurs en paramètre.

Partage par email

Par ailleurs, pour pouvoir partager les données récoltées d'un appareil sélectionné, nous avons utilisé la libraire *Email Intent Builder*. Celle-ci facilite grandement l'écriture de code. En effet, en quelques lignes on peut créer la fonctionnalité de partage. Elle est disponible ici : https://github.com/cketti/EmailIntentBuilder

Difficultés rencontrées

Lors du développement de l'application, nous avons rencontré certains défis plus ou moins difficiles à surpasser. Premièrement, il a fallu que l'on comprenne bien la technologie du Bluetooth. Il fallait aussi étudier les différentes librairies à notre disposition, c'est-à-dire, ce que chacune retournait comme valeurs et comment interpréter ces valeurs. Il a fallu mettre beaucoup de temps sur cela. Deuxièmement, il a fallu afficher de faconde dynamique les appareils récoltés. Pour cela il a fallu comprendre l'importance de l'utilisation d'un *RecyclerView* qui requiert un *Adapteur*. Troisièmement, il a fallu mettre en place un algorithme qui estime les coordonnées GPS de l'appareil Bluetooth grâce à son intensité. Ainsi, plusieurs recherches ont été effectuées pour arriver à la meilleure étude possible sur le sujet, selon nous, comme décrite dans la section ci-haut.

Critiques et améliorations

Nous avons apprécié la formulation de l'énoncé qui est concis et assez «droit au but». C'est-à-dire qu'il laisse un bon degré de liberté quant à la conception de l'application. En revanche, l'énoncé n'était pas très clair au sujet de fonctionnalité « Comment y arriver » et la fonctionnalité « Partager ». On croit que l'énoncé pourrait être plus détaillé sur ces points.

Conclusion

En définitive, ce laboratoire nous a permis d'approfondir nos connaissances Android sur les librairies matérielles dont Android a à nous offrir, c'est-à-dire l'utilisation du Bluetooth et du GPS par exemple. C'est quelque chose que nous avons jamais effectué par le passé. Nous sommes assez contents de notre produit final. Ce travail pratique nous a fait découvrir un volet assez important de la programmation mobile. Finalement, cette application de localisateur Bluetooth nous est apparue comme un choix très pertinent pour une deuxième prise de contact avec les technologies du développement mobile. Par ailleurs, il serait intéressant d'accommoder l'application avec la technologie du Bluetooth 5.1 dont certains nouveaux appareils possèdent. En effet, avec le Bluetooth 5.1, nous aurons la capacité de beaucoup mieux estimer la localisation des appareils sur la carte, entre autres grâce à la capacité de recevoir l'angle du signal reçu.