TELLIER Olivier
MODOUX Josua
CHABOUD Thomas

Dossier Objets connectés
Where is Bertrand ?

Année 2017-2018 LP IEM

Sommaire

1. Le projet

Notre projet se nomme “Where is Bertrand ?”. Il s’agit d’une application permettant de retrouver un objet perdu. Pour cela, on dispose d’une interface avec un signal devenant de plus en plus fort en fonction de notre proximité avec l’objet. La recherche se fait par Bluetooth. On dispose également d’un bouton sur l’application qui permet à l’objet perdu d'émettre un son pour mieux le localiser.

Pour ce projet, nous utilisons Android pour l’application mobile, développée sur Smartphone. Les cartes utilisées pour retrouver l’objet sont une RedBear RBLink ainsi qu’une RedBear Duo pour le Bluetooth.

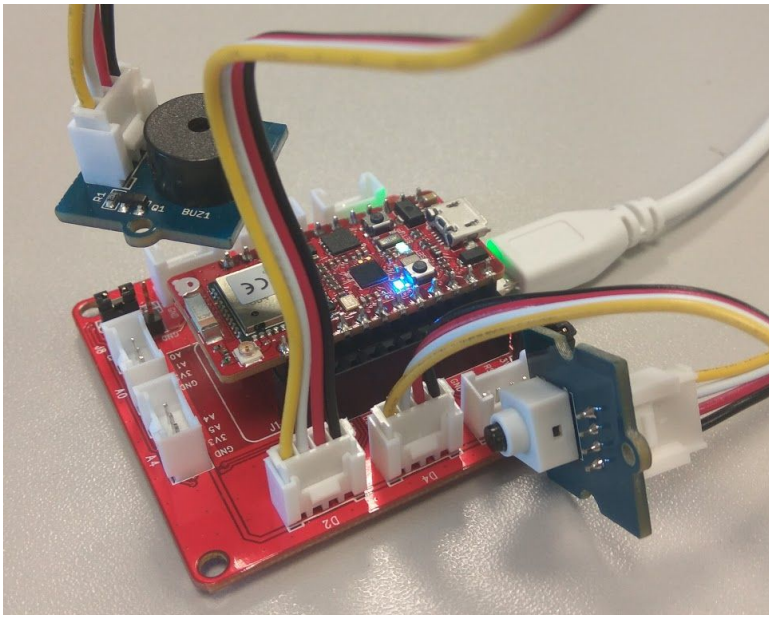
2. Installation

2.1. Matériel requis

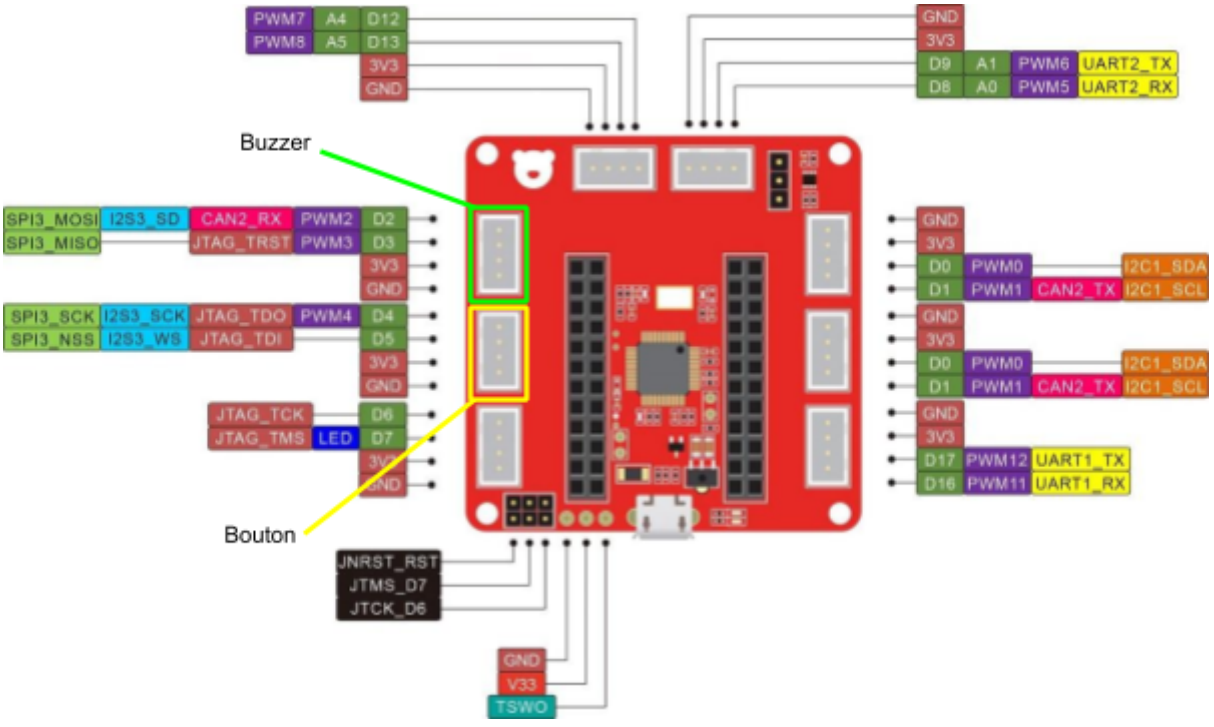
Pour ce projet, nous avons besoin :

- D’un smartphone sous Android (api > 21)
- D’une RedBear RBLink
- D’une RedBear Duo
- Un bouton poussoir
- Un module de son

2.2. Plan de montage



1 - Tout d'abord, il faut connecter le Redbear Duo au Redbear RBLink (les 2 ports micro USB doivent être du même côté). Ensuit, connectez le buzzer sur le PIN D2 et le bouton poussoir sur le PIN D4 de la Redbear RBLink (voir schéma ci-dessous).



2 - Brancher par la suite le micro-contrôleur à l'ordinateur via le port micro USB du Redbear RBLink.
3 - Sur Windows uniquement, veuillez installer les drivers USB en suivant ce guide : Windows Driver Installation Guide.
4 - Installer l'IDE Arduino 1.6.7 (ou plus).
5 - Lancez Arduino et allez dans "File > Preferences" et ajouter l'URL : https://redbearlab.github.io/arduino/package_redbear_index.json dans le champs "Additionnal Boards Manager URLs".
6 - Allez dans "Tools > Board > Boards Manager". Cherchez le package pour RedBear Duo board. Sélectionnez une version et cliquez sur "Install".
7 - Allez dans "Tools > Boards" sélectionnez "RedBear Duo (RBLink USB Port)".
8 - Allez dans "Tools > Programmer" sélectionnez "Duo FW Uploader".
9 - Pour finir, ouvrez le code fournit sur le Github (RedBearDUO.zip) et uploadez le.

2.3. Code source

Voici le lien du repo git de notre projet. Vous y trouverez le code source de l'application Android, le programme pour la RedBear ainsi que les maquettes du projet et une vidéo de présentation de l'application : https://github.com/otellier/WhereIsBertrand

3. Communication avec la RedBear

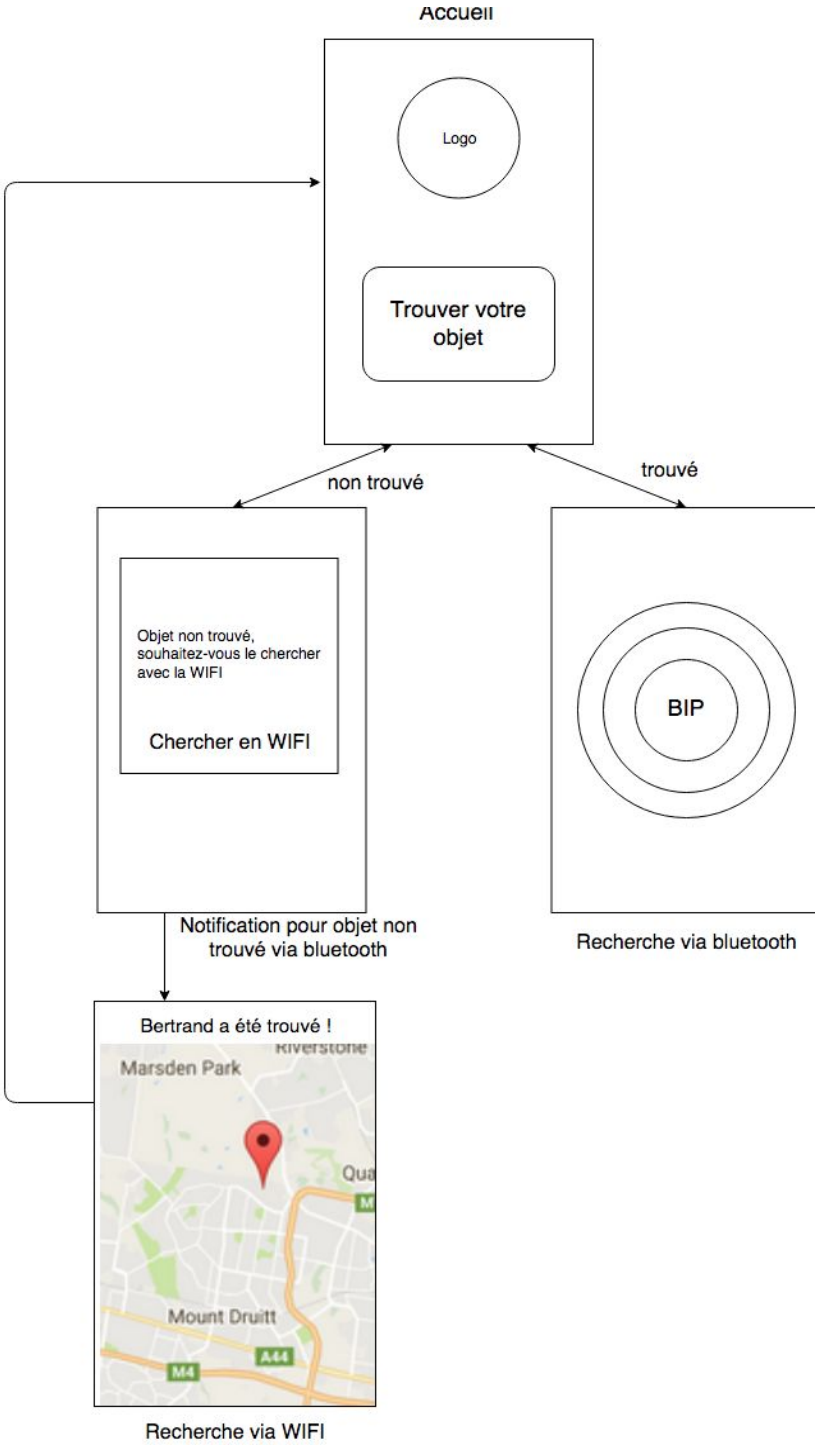
La RedBear active d'abord le bluetooth BLE et est ensuite en "attente" d'une connexion. Lorsqu'un device se connecte à elle, elle déclenche le buzzer et s'arrêtera lorsque l'utilisateur appuiera sur le bouton. Pour la partie Android, l'application possède l'adresse MAC de la RedBear ce qui lui permet de vérifier son RSSI toute les 7 secondes. Ainsi que de se connecter directement à elle lorsqu'on appuie sur le bouton "Bip" de l'interface, ce qui déclenche donc le buzzer.

4. Code Métier

Pour notre application Mobile liée à notre objet connecté "Bertrand", nous avions tout d'abords souhaité avoir deux parties distinctes :

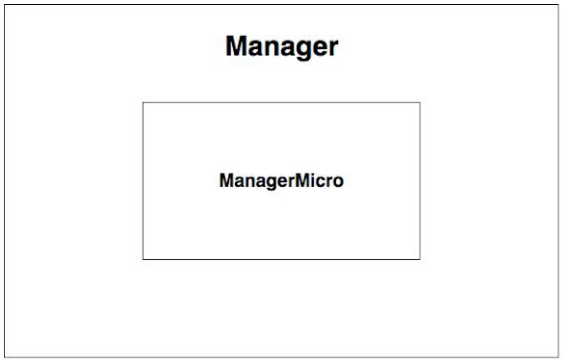
En effet, depuis l'écran d'accueil, le bouton "Trouver votre objet", devait lancer un scan Bluetooth dans le but de détecter Bertrand ce qui indiquerait une relative proximité avec lui, le cas échéant une recherche Wi Fi aurait été lancée avec l'API Google Maps avec un itinéraire jusqu'à lui.

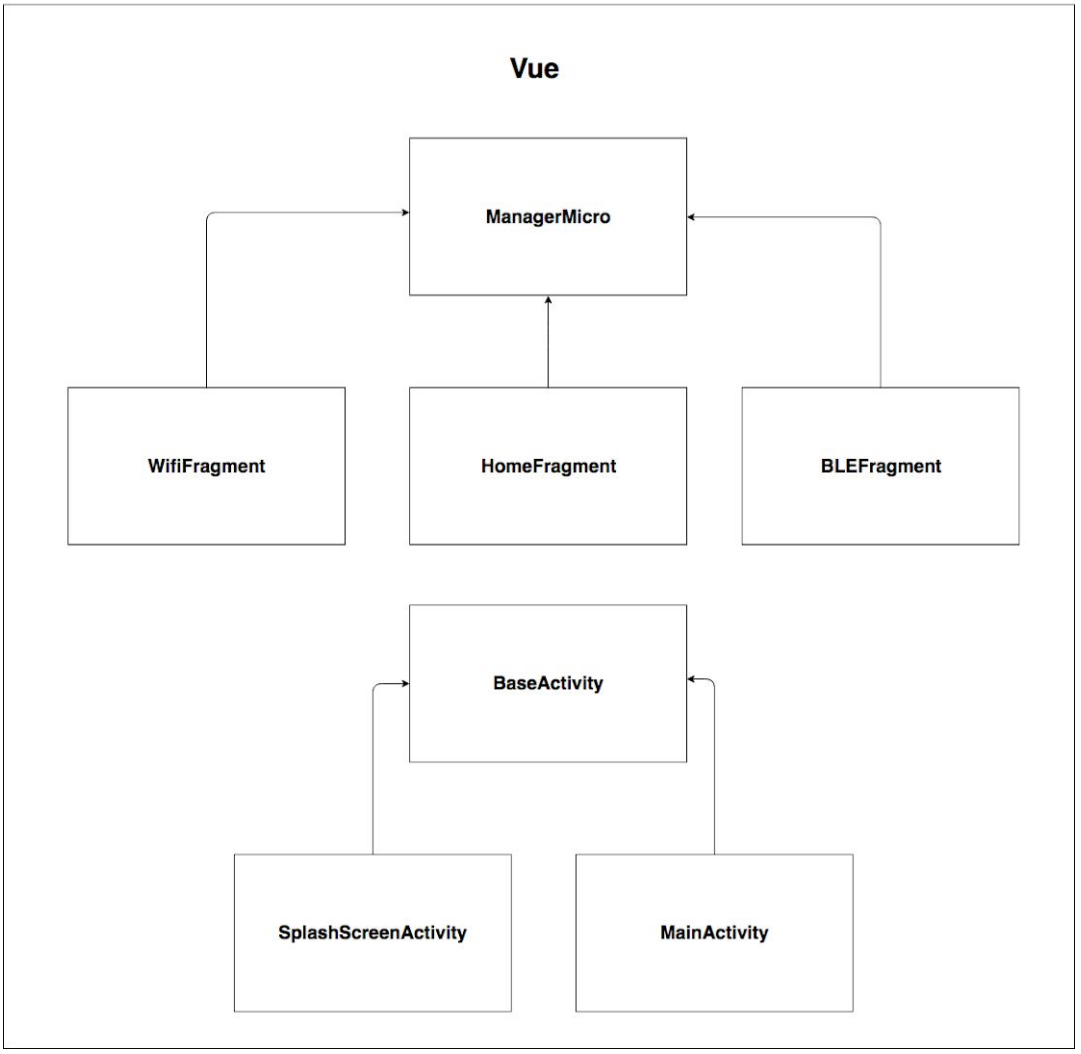
Dans le cas où le signal Bluetooth est trouvé, la page “radar” s’ouvre alors, indiquant grâce à la vitesse des ondulations du radar si l’objet est plus ou moins proche. Le bouton central, lui, déclenche un signal sonor jusqu’à ce que Bertrand soit retrouvé et que l’utilisateur appuie sur le bouton mécanique présent sur celui-ci.



Malheureusement, la recherche WI FI n’a pas pu être implémenté, voir la parties des difficultées rencontrées.

Architecture





Vues

Les Activités

Au nombre de deux, elles héritent de notre BaseActivity pour leurs permettre de toutes posséder les méthodes dont nous avons besoin pour la gestion des fragments :

- SplashScreen : Classe accessoire pour afficher des loading Screen au lancement de l'application ou lors des requêtes asynchrones (pour combler l'attente de l'utilisateur, pour son confort et pour limiter son impatience)

- MainActivity : Une fois le splashScreen terminé et l'application chargée, l'utilisateur est redirigé sur cette activité, contenant uniquement un fragment vide que nous allons ensuite remplir

Les Fragments

Nos fragments sont donc l'ensemble des fonctionnalités de l'application qui sont affichés depuis la MainActivity, héritant tous, cette fois de BaseFragment pour la gestion et le switch entre ceux-ci :

- HomeFragment : Il s'agit de notre écran d'accueil, composé du Logo de l'application et du bouton central pour lancer la méthode isDeviceInRange(), pour détecter si l'objet se trouve à une distance proche (actuellement, n'ayant pas fait la partie Wi Fi, nous partons du principe que l'objet est toujours détecté). Si tel n'était pas le cas, un dialogue s'ouvre alors pour proposer la recherche Wi fi.

- BluetoothResearchFragment : Voici la page permettant à l'utilisateur de retrouver Bertrand. Elle est composée d'un texte indiquant la valeur RSSI indiquant la qualité de réception du signal, d'un bouton central pour lancer un bip sur l'objet ainsi que d'un GIF sous forme de radar.

Nous utilisons les librairies présentent dans android.bluetooth. Ici, la possibilité d'utiliser le bluetooth de l'appareil est initialisée, détectant si le téléphone a le Bluetooth activé. S'il ne l'est pas, une pop up vient demander les droits à l'utilisateur pour celui-ci. Une fois cela fait, un scan Bluetooth BLE est lancé pour rechercher Bertrand avec son adresse MAC indiquée. Lorsque l'objet est trouvé, son RSSI actuel est indiqué dans le textView et le scan est stop pour une durée de 7s (voir les difficultés) avant d'être relancé pour avoir les valeurs actualisées et un refresh du GIF avec la vitesse adéquate. Lorsque le bouton est pressé, l'application se connecte à Bertrand en utilisant GattClientCallBack, une simple connection suffit, le reste étant dans la partie code Arduino.

Manager :

ManagerMicroController : Dans le manager, nous avons le display de l'animation radar grâce à la classe AnimationDrawable. Les images du radars sont ajoutées avec un temps entre les images définis par la valeur RSSI actuel.

5. Difficultés rencontrées

5.1. La RedBear

Plusieurs difficultés ont été rencontrées. Concernant la RedBear, nous avons eu des problèmes au niveau des plugins. En effet, nous ne parvenions pas à lancer les projets exemples, ce qui était plutôt contraignant. Nous avons finalement trouver une documentation nous permettant de régler ces problèmes.

5.2. L'Application Android

2 problèmes nous ont retardé dans le développement de l'application mobile :

- Le premier est un problème lié au Bluetooth. En effet, nous ne parvenions pas à utiliser le bluetooth avec des téléphones récents. Au départ, nous pensions qu'il s'agissait de la version d'android ou encore de la version du bluetooth utilisée sur les anciens modèles de smartphones mais le problème ne venait pas de là. Après de nombreuses recherches, nous avons pu déterminer qu'il s'agissait d'un problème de droits à ajouter dans le manifest pour les versions plus récentes d'Android.

- Nous avons ensuite essuyé une difficulté inattendue qui nous a bloqué un certain temps par rapport au refresh de l'animation radar. En effet nous avions remarqué qu'avec un scan toutes les secondes (notre objectif initial, pour une mise à jour rapide et constante), au bout de 5 scans et toujours 5 scans, un sixième se lançait mais ne trouvait plus l'objet, même si celui-ci était tout proche. Avec des recherches, nous avons constaté que par sécurité, Android bloque le processus au-delà de cinq scans toutes les trente secondes. Pour résoudre ce soucis, nous avons, au final, retardé les scans à un toutes les sept secondes, ce qui est plus embêtant d'un point de vue expérience utilisateur, mais il s'agissait de la seule solution que nous avons trouvées.