# Rapport de développement mobile

# Track Me Inside

## Présentation

L’application Android « Track Me Inside » doit permettre le chargement et l’affichage de données sur le téléphone de l’utilisateur lorsque celui-ci se trouve à un endroit spécifique à l’intérieur d’une salle de classe.

### Contraintes

* L’application doit être développée sur Android ;
* L’application doit permettre de localiser le téléphone de l’utilisateur à l’aide de plusieurs balises ;
* La localisation doit être précise dans toute la salle de classe ce qui exclut l’utilisation naturelle du SDK Android de Estimote qui se base sur la proximité de l’utilisateur à une zone définie par une ou plusieurs balises ;
* Ces balises seront représentées des Proximity Beacons de l’entreprise Estimote ;
* L’application doit utiliser au moins deux capteurs différents ;
* L’application doit permettre la persistance de certaines données.

### Déroulement

Premièrement, nous allons présenter les classes du Estimote Android SDK que nous avons implémentées ainsi que celle que nous avons considérée.  
Deuxièmement, nous détaillerons le fonctionnement du protocole Bluetooth.  
Troisièmement, nous expliquerons comment nous avons calculé la position de l’utilisateur à partir de plusieurs balises.  
Finalement, nous parlons des améliorations possibles et ferons une conclusion.

## Estimote SDK

Ce SDK s’adresse à toute la gamme de produit de Estimote. Cela comprend les Proximity Beacons, Location Beacons, Long Range Location Beacons, Stickers, Mirror. Nous nous intéressons uniquement aux Proximity Beacon car ils sont les plus performants.

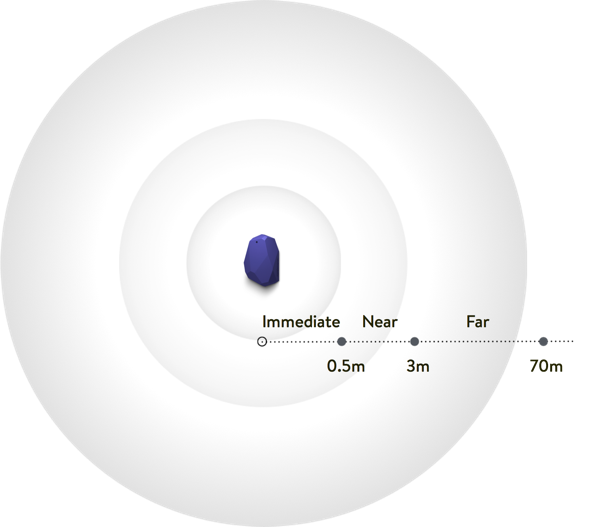
* La classe *Beacon* représente une balise Estimote.
  + Les méthodes permettant de déterminer l’identité d’une balise par le biais de son *UUID*, *minor* et *major* sont *getProximityUUID ()*, *getMinor ()*, *getMajor ()*.
  + La méthode *getRSSI ()* renvoie un Indicateur de puissance du signal reçu. C'est la puissance du signal de la balise au moment où le téléphone reçoit ce signal.
  + La méthode *getMesuredPower ()* renvoie une constante en lecture seule étalonnée en usine qui indique quel est le RSSI attendu à une distance de 1 mètre de la balise. Combiné avec RSSI, il permet d'estimer la distance entre le dispositif et la balise. Nous appellerons cette valeur « **txPower** ».
* La classe *Utils* propose quelques méthodes pratiques :
  + La méthode *computeAccuracy (Beacon beacon)* retourne la distance entre le beacon et le téléphone en mètre à partir du RSSI et de la puissance d’émission mesurée de la balise (txPower).
    - Il n’est pas possible de donner le txPower en argument de la méthode ni te modifier cette valeur dans l’objet *beacon*. Cependant nous pouvons passer par le constructeur de la classe *beacon* pour spécifier le txPower.
* La classe *Utils.Proximity* propose quatre zones pour estimer la distance à une balise. Ces quatre zones sont représentées par un Enum qui est composé de :
  + IMMEDIATE, NEAR, FAR, UNKNOWN
* La classe *BeaconManager* permet de monitorer une zone.
  + Il existe deux classes dérivées qui implémente des listener donnant une indication sur la proximité de l’utilisateur à un beacon ou plusieurs : *MonitoringListener* et *RangingListener*

### Choix entre Ranging et Monitoring

Le SDK détecte et interagit avec les balises de deux façons :

* Monitoring : actions déclenchées à l'entrée et à la sortie de la région. Fonctionne si l'application est en cours d'exécution, suspendue ou tuée (si l'application ne s'exécute pas quand une entrée / sortie arrive, l’OS le lancera en arrière-plan pendant quelques secondes pour gérer l'événement)
* Ranging : actions déclenchées en fonction de la proximité d'une balise ; Ne fonctionne que lorsque l'application est en cours d'exécution (par exemple, elle s'affiche à l'écran ou s'exécute en arrière-plan en réponse à un événement de surveillance, etc.)

Ranging retourne une **liste de balises dans la portée**, avec une proximité estimée de chacun d'eux.



Nous avons choisi la classe *RangingListener* car elle fonctionne même quand l’utilisateur est dans la zone. L’information renvoyée par la classe *RangingListener,* c’est-à-dire une liste de *Utils.Proximity* n’est d’aucune utilité pour notre application. Nous avons choisi d’inférer la distance entre l’utilisateur et les beacons à partir du RSSI.

### Android vs Apple

Le SDK indoor location développer pour iOS emploie une tonne de techniques complexes pour prédire où l'utilisateur est. Estimote est en train d’adapter ces algorithmes pour Android mais c’est un processus qui prend du temps en raison de l'immense variations des appareils Android. Chaque modèle possède une antenne Bluetooth et un hardware différent. (Différents matériels Bluetooth, différentes positions des antennes, différents facteurs de forme et matériaux de fabrication, etc.)

## Technologie d’émission

### Bluetooth LE (BLE)

Le protocole Bluetooth Smart aussi appelé Bluetooth LE ou Bluetooth 4.0 offre des améliorations par rapport à sa technologie Bluetooth prédécesseur. Le protocole fonctionne autour de la fréquence radio 2,4 GHz et est utilisé pour des courtes portées.

Bluetooth Low-Energy vise à fournir des communications RF (Radio Fréquence) à faible consommation d'énergie à court terme, c'est-à-dire dessous de 100 mètres de couverture du signal. Le protocole Bluetooth LE met l'accent sur les économies d'énergie plutôt que sur le débit de données ou sur la modulation.

Selon le cas, le Bluetooth LE peut utiliser jusqu’à 1% de batterie par rapport à la norme Bluetooth. Par contre le débit de donnée pour descendre au tiers de la technologie Bluetooth.

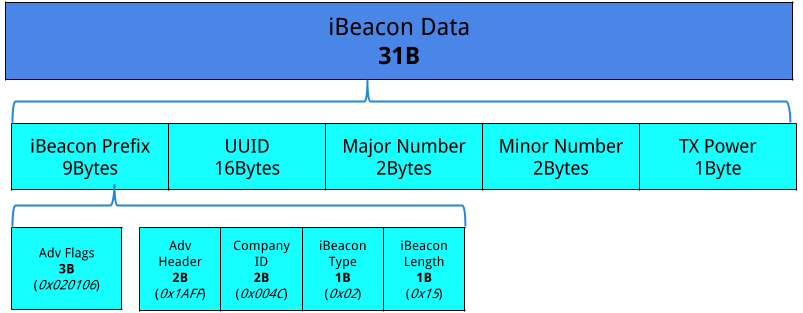
### IBeacon

IBeacon est un système de positionnement en intérieur développé par Apple fonctionnant sur Bluetooth 4.0. Ce protocole est utilisable par IOS, OS X 10.9+ et Android 4.3+. Il diffuse quatre informations :

* Un *UUID*: identifie la balise.
* Un *Major*: valeur identifiant un sous-ensemble de balises dans un grand groupe.
* Un *mineur*: valeur identifiant une balise spécifique.
* Un niveau de puissance TX indiquant la force du signal à un mètre de l'appareil. Cette donnée est étalonnée par Apple pour être précis pour ses propres produits.

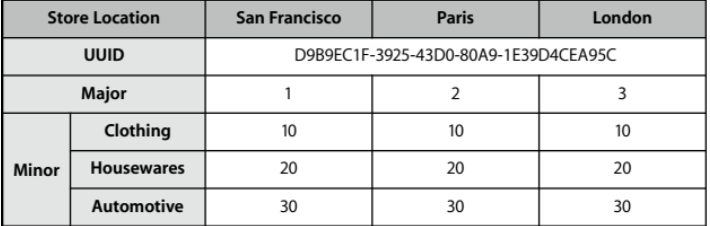
L’application lit l'UUID, le majeur et le mineur et les utilise pour déterminer des informations sur la balise par le biais d’une base de données ; La balise elle-même ne contient aucune information descriptive.

.



#### Exemple de l’utilisation du protocole iBeacon

Voici l’exemple d’un magasin de détail national. L'UUID représente le propriétaire des balises, ici l’entreprise. Le Major définit chaque emplacement du magasin. En utilisant cette valeur, l'application sera en mesure de déterminer facilement dans quel magasin l'utilisateur est actuellement. Le Minor définit une section spécifique dans chaque magasin. Cela permet à l'application de déterminer la section du magasin dans laquelle l'utilisateur se trouve.



L’image ci-dessus montre un exemple de mise en œuvre du protocole iBeacon

## Inférer la position de l’utilisateur

Les estimations de distance aux balises ne seront jamais super fiables. Ils sont basés sur la force du signal qui arrive au téléphone, mais le problème est que ce signal peut parcourir des chemins multiples (ligne droite, ou rebondir sur des murs, etc.), ou être absorbé sur le chemin (par exemple, par d'autres Personne). Même la façon dont le téléphone est tenu par l’utilisateur a un impact sur la précision (passer du mode portrait en le tenant d'une main en mode paysage à deux mains). Le téléphone ne peut pas savoir si la puissance du signal a baissé parce que quelqu’un fait obstacle ou parce que la distance a augmenté.

C'est en fait une relation quadratique inverse. Si une distance à la balise double, la force du signal est divisée par quatre. Cela diminue drastiquement la précision de l’estimation de la distance à mesure que la distance augmente.

Cette puissance du signal reçu est appelée **Received Signal Strength Indication** ou **RSSI.**

### Dériver la distance du signal du balise

Il existe une grande variété de formule qui lie le RSSI à la distance. La plus populaire est :  
RSSI (dBm) = - 10n \* log10(d) + txPower.  
  
n est la constante de propagation ou l'exposant de perte de trajet. Elle varie de 2,7 à 4,3 selon l’humidité du milieu. (Le vide a n = 2 pour référence).

Cette formule ne nous a pas donné de résultat satisfaisant.

Ensuite, nous nous sommes basé sur une formule qui effectue une régression de puissance sur la base de mesures effectuées sur un dispositif spécifique. Cette formule est :  
distance [m] = A \* (RSSI / txPower)B + C avec A, B et C étant des constantes.

La calibration de ces constantes a été faite pour le Nexus 4 dont les valeurs sont :  
A = 0.42093, B = 6.9476 et C = 0.54992.

### Résultat de notre application

À proximité d'environ 1 mètre, vous pouvez vous attendre à voir des estimations de distance entre 0,5-2 mètres. À des distances supplémentaires, vous verrez plus de variation. À 20 mètres ou distance réelle, l'estimation fournie par notre application peut varier de 10-40 mètres. La variation est causée par le bruit sur la mesure du signal, ainsi que les réflexions du signal et les obstructions.

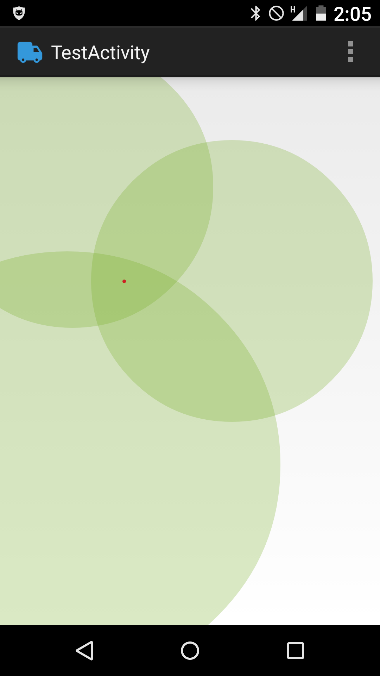
### Trilatération vs triangulation

Afin de déterminer la position de l’utilisateur, l’application scan les balises à proximité. Quand trois balises sont captées et leur puissance d’émission est suffisamment haute, il est temps de mettre ces données ensemble. A cet fin, il existe deux méthodes principales : la triangulation et la trilatération.

La triangulation est le processus de détermination de l'emplacement d'un point en mesurant les angles entre celui-ci et des points connus.

Afin de pouvoir mesurer les angles d’arrivée du signal de réception (angle of arrival 'AOA') il nous faut des antennes spécifiques. Malheureusement, les smartphones actuels ne possèdent pas ces antennes pour la technologie Bluetooth ou Wifi. Nous avons donc opté pour la trilatération.

La trilatération consiste à mesurer la distance le téléphone et trois beacons.



L’algorithme implémenté utilise la méthode d’interpolation linéaire en 2D. Trois cercles sont générés à partir de la position de chaque balise et de sa distance avec l’utilisateur. Le résultat est une donnée par le centre de la zone à équidistance des trois cercles. L’avantage de cette méthode est qu’elle ne nécessite pas que les trois cercles se touchent.

## Utilisation d’un deuxième capteur

Afin de réduire le bruit sur l'estimation, nous récupérons les valeurs du signal sur 10 secondes puis nous ignorons le 10% des plus grandes valeurs et le 10% des plus petites valeurs. Finalement, nous prenons la moyenne des valeurs restantes. Lorsque l’utilisateur se déplace, l'estimation de la distance sera retardée jusqu'à ce qu’il soit stationnaire pendant 10 secondes. À partir de ce moment l'estimation de distance se stabilisera. La variation de temps est réglable.

Afin de détecter les mouvements de l’utilisateur, nous avons utilisé l’accéléromètre. Lorsque la norme du vecteur résultant de l’addition des vecteurs d’accélération sur les trois axes dépasse un seuil donné, un indicateur visuel averti l’utilisateur que l’approximation de la position ne peut plus être effectuée. Quand celui-ci est à nouveau stationnaire, un indicateur visuel s’affiche ainsi que sa nouvelle position estimée.

## Protocole de calibrage

Le téléphone est posé sur une surface à niveau et la balise est placée à un mètre de celui-ci dans la direction du haut du téléphone. Aucun obstacle doit être placé entre les deux appareils. Pendant une période définie par l’utilisateur, le téléphone lit le RSSI de la balise. À la fin du temps imparti, nous récupérons les valeurs du les valeurs extrêmes avant de faire la moyenne.

### Persistance du calibrage

La constante de calibrage est sauvegardée dans le cache de l’application. La première fois que l’utilisateur ouvre l’application, la fenêtre de calibrage sera automatiquement lancée. Une fois cette opération effectuée, cette fenêtre sera accessible par le biais d’un onglet.

## Amélioration

Voici quelques points que nous pourrions investiguer afin d’augmenter la précision du processus du calcul de la position de l’utilisateur :

* Utiliser un modèle de trilatération basé sur R3. Cela ralentirait les calculs mais augmenteraient la précision globale ;
* La distance peut être inférée à partir du moment où le signal est arrivé au téléphone (time of arrival « TOA »). Cependant, cette technique a besoin d'une synchronisation d'horloge précise entre le téléphone et les beacons. Nous n’avons pas pu utiliser cette technique car les beacons ne peuvent pas transmettre le moment d’émission d’un signal.

### Economie de la batterie des beacons

Lors du déploiement d'un réseau de capteurs Bluetooth il est important de garder à l'esprit que chaque nœud a une quantité limitée d'énergie. La batterie des beacons peut tenir jusqu’à deux ans en utilisant les paramètres par défaut et que la gestion intelligente de la batterie est activée.

Afin d’augmenter la précision de l’application nous avons modifié certains paramètres de base de nos balises :

* La puissance d’émission : La valeur par défaut est de -12 dBm et peut varier de -30 dBm à +4 dBm ce qui correspond à une précisant variant d’environ 10 centimètres jusqu’à 40-50 mètres. Nous avons choisi +4 dBm.
* L’intervalle d’émission : La valeur par défaut est de 950ms et peut varier de 100ms à 2000ms. Nous avons choisi 100ms.

Nous avons néanmoins utilisé deux fonctionnalités pour économiser la batterie des balises Estimote :

* « Smart Power Mode » : Le mode intelligent utilise des algorithmes adaptatifs avancés pour optimiser automatiquement la puissance d’émission. S'il n'y a personne à portée, la balise émet moins souvent pour économiser la batterie. Lorsqu'un utilisateur est détecté, la balise émet avec un débit normal.
* « Flip to Sleep » : La balise cesse d’émettre lorsque qu’elle est sur le dos.

## Conclusion

En fin de compte, la question se pose s’il est possible de trouver des valeurs suffisamment précises pour l’évaluation de la distance ? Nous avons eu des résultats mitigés alors que les variables de terrain et les conditions étaient quasi parfaites. Il reste certes des pistes d’améliorations que nous pouvons suivre. Estimote a prouvé que cet objectif était atteignable en utilisant le protocole IBeacon avec des appareils Apple. Les algorithmes qui composent le Estimote Indoor SDK for IOS est le fruit de la mise en commun de plusieurs thèses par une équipe d’expert. Il est donc difficile de juger de l’écart qui sépare nos résultats de ceux du Indoor SDK.

## Références

<https://github.com/lemmingapex/Trilateration>

<https://www.researchgate.net/post/How_to_detect_the_object_as_well_as_its_distance_using_tringulation>

<http://www.slideshare.net/ChrisThomson2/factors-effecting-positional-accuracy-of-ibeacons>

<https://arxiv.org/ftp/arxiv/papers/1312/1312.6829.pdf>

<http://electronics.stackexchange.com/questions/83354/calculate-distance-from-rssi>

<http://web4.cs.ucl.ac.uk/staff/D.Barber/textbook/090310.pdf>

<https://community.estimote.com/hc/en-us/articles/203909356-Is-Indoor-Location-SDK-available-on-Android->

<https://forums.estimote.com/t/android-and-ios-indoor-location/4633>

<http://www.eecs.ucf.edu/seniordesign/fa2014sp2015/g14/docs/sd_1_report.pdf>

<https://developer.mbed.org/blog/entry/BLE-Beacons-URIBeacon-AltBeacons-iBeacon/>

<https://forums.estimote.com/t/determine-accurate-distance-of-signal/2858/4>

<https://altbeacon.github.io/android-beacon-library/distance-calculations.html>

<https://community.estimote.com/hc/en-us/articles/201636913-What-are-Broadcasting-Power-RSSI-and-other-characteristics-of-beacon-s-signal->

<https://en.wikipedia.org/wiki/Trilateration>

<http://stackoverflow.com/questions/30336278/multi-point-trilateration-algorithm-in-java>

<http://estimote.github.io/Android-SDK/JavaDocs/>