Rapport sur la localisation d’objet avec le bluetooth low energy



Juin 2015

Auteurs : JG, AP, GM, GF,

Contenu

[Objectifs 3](#_Toc453853146)

[Vue globale 4](#_Toc453853147)

[Les Récepteurs 5](#_Toc453853148)

[Les Emetteurs 6](#_Toc453853149)

[Les Expériences 7](#_Toc453853150)

[Méthodes de localisation 9](#_Toc453853151)

[La trilatération 9](#_Toc453853152)

[Les empreintes 10](#_Toc453853153)

[NN et BLE 11](#_Toc453853154)

[Mode Localisation 16](#_Toc453853155)

[Robustesse et Résilience 17](#_Toc453853156)

[Comparaison entre Fingerprint NN et Trilatération 19](#_Toc453853157)

[Historique des localisations 20](#_Toc453853158)

[Conseils pour le training avec NN 20](#_Toc453853159)

[Amélioration Trilatération (Houcem ?) 22](#_Toc453853160)

[Visualisation des résultats (Alexandre) 23](#_Toc453853161)

[Web Application 23](#_Toc453853162)

[http://192.168.0.246:9000 23](#_Toc453853163)

[Cartes 26](#_Toc453853164)

[Normes 26](#_Toc453853165)

[Sitg,…. 26](#_Toc453853166)

[Project tango 26](#_Toc453853167)

[Annexe : Schéma EXP1 27](#_Toc453853168)

[Annexe : Schéma Exp2 – test hors grille 28](#_Toc453853169)

[Annexe : Schéma Biblio-test1 29](#_Toc453853170)

[Annexe : Schéma Biblio-détail1 30](#_Toc453853171)

[Annexe : Schéma Biblio-détail2 31](#_Toc453853172)

[Annexe : Schéma Biblio-ligne de Gilles 32](#_Toc453853173)

[Annexe : Détail de la précision avec différente configuration (80/20) 33](#_Toc453853174)

[Annexe : Nordic Semiconductor nRF51822 34](#_Toc453853175)

[Annexe : Résultats de Houcem 36](#_Toc453853176)

## Objectifs

Nous rappelons les objectifs que nous devions explorer dans cette étape.

Il est demandé de :

* Localiser un objet: retrouver en temps réel la zone dans laquelle se trouve l’objet et l’identifier dans cette zone
* Suivre des objets: pouvoir reconstruire la trajectoire d’un objet (listes de couples (position,temps))

Contraintes

* Objets mobiles et équipés d’un émetteur
* Capteurs sont fixes avec une localisation connue

Explorer les limites :

Explorer les topologies

distance entre capteurs

les formes du maillage (carré, triangulaire, …)

Limitations dues aux obstacles

Explorer la précision des localisations

Explorer plusieurs types de BLE

Explorer la densité des objets (nbr de BLE au m2)

Explorer les performances: période d’émission, distance, batteries, prix, …

## Vue globale

Le schéma suivant donne une vision globale de l’application.

* Les ibeacons (qui marquent les objets à localiser ou à tracer) émettent régulièrement un signal qui leur est propre.
* Les récepteurs (les raspberry pi 3) écoutent tous les signaux émis. Ils copient ces informations dans la base de données.
* Un processus de calcul des positions x y des BLE maintient la position courante des tous les BLE entendus.
* Un processus de d’affichage des positions et des cartes donne la position courant du BLE recherché par un utilisateur. Le résultat est sous la forme d’une carte annotée.



*Figure : vue globale de la communication du système*

## Les Récepteurs

Nous avons choisi les raspberry pi 3 comme récepteur fixe. Son prix de base est modéré (50 CHF) et il est déjà équipé d’un récepteur Bluetooth et d’un récepteur Wifi.

[](https://cdn-shop.adafruit.com/1200x900/3055-06.jpg)

*Figure : raspberry pi 3*

**Spécifications**

* Quad Core Broadcom BCM2837 64-bit ARMv8 processeur à 1.2GHz
* 1Gbyte de mémoire
* Sortie HDMI, 4 ports USB 2, son,
* Broche pour IoT
* BCM43143 WiFi chip BUILT-IN et un Bluetooth Low Energy (BLE)
* Alimentation 5v, 2.5A

Les raspberry ont été acheté sous forme de Kit pour ajouter le mémoire (carte micro SD 16Gb) sur laquelle le système d’exploitation est écrit, le boitier et l’adaptateur secteur.

Nous avons choisi Raspbian comme système d’exploitation. Les fonctionnalités de ce dernier étaient largement suffisantes pour notre expérimentation.

Les logiciels utilisés sur le raspberry sont

* Python
* Java
* Librairie bluez et cassandra.

Le raspberry est aussi équipé d’une clé USB de 8Gb dans laquelle des paramètres propres au raspberry pi sont copiés (fichier properties, macshortid.txt ). Les données brutes en provenance de la réception BLE sont aussi loguées sur la clé dans un dossier nommé SCAN.

Le raspberry était associé à un point d’accès Wifi. Dans lequel le service DHCP lui attribué une adresse prédéterminée en fonction de sa MAC adresse.

**Fonctionnement de base du raspberry.**

Lors de la mise sous tension :

1. Connection au Wifi
2. Mise à jour de l’heure/date avec le service NTP du routeur Wifi. Ce point est essentiel pour que l’ensemble des raspberry soient synchronisés.
3. Démarage du programme de réception Blutooth BLE. Le programme est associé avec « tee » un service de duplication du flux de données. Une partie du flux est détourné vers un programme java et un autre loguant les informations bruts dans un fichier.
4. Le programme java logue ces entrées dans un fichier et il copie (après avoir filtré et traduit les MAC Adresses en « shortname »  des BLE) dans la base de données Cassandra.

Les activités d’écoute et de copie sur la BD se poursuivent jusqu’à l’arrêt du raspberry.

La console ssh (ou éventuellement vnc) peut être utilisée pour examiner le bon fonctionnement des programmes.

La console ssh est utilisée pour déployer des nouvelles versions des programmes.

## Les Emetteurs

Nous avons testé plusieurs sortes de beacon :

* iBKS,
* Social Retail
* StickNfind

Les beacons de iBKS et Social Retail sont basé sur le circuit de **Nordic Semiconductors nrf51822. Ce circuit est celui qui est utilisé dans un grand nombre de Beacons. Ils sont donc très similaires dans leur performance.**

Nous avons été déçus par les performances des Beacons de StickNfind. L’application « Radar » livrée avec les Beacons est inutilisable. Et la puissance d’émission est plus faible que iBKS. Ces constatations sont confirmées par les « post » que l’on peut trouver sur internet par rapport à ces Beacons.

Nous avons aussi essayé un Beacon avec une antenne. Ce qui permet d’augmenter considérablement la portée des Beacons. L’antenne des autres Beacons est réduite à un composant mesurant 3 mm2. Par ailleurs, Il serait aussi intéressant de voir de vérifier comment les résultats sont modifiés si les récepteurs sont équipés avec une antenne.

Nous avons testé des périodes « advertisement » de 100 ms à 1 seconde.

La densité (maximum de beacon sur une surface) est directement liée à ces périodes.

Un beacon émet successivement dans trois bandes (en ajoutant une valeur de 0 à 20ms à la période). La durée du message « advertisement » est d’environ 1 ms. Donc théoriquement, on a 3000 « slots » possibles. La modélisation des collisions est nécessaire (et une expérimentation aussi). On peut estimer qu’en utilisant 10% des possibilités, cela nous donne 300 slots donc 300 beacons visibles émettant avec une période de 1 seconde.

## Les Expériences

Une expérience est déterminé par :

* Un site
* Les emplacements des raspberry pi
* Des lots de mesures correspondants à des beacons disposés à certains points dans le site.
* Une analyse des résultats.

Initialement, les points de tests devaient servir à évaluer la précision de la méthode de localisation par trilatération. Finalement, ces points sont devenus des points d’apprentissage d’une méthode de « finger print » implémentée avec un réseau de neurones.

Nous avons exécuté les expériences suivantes :

* Exp1 (4 raspis, 4m x 5m)
* Exp2 (9 raspis, 14m x 14m)
* Biblio (11 raspis, 18m x 10m)

Dès la première expérience, il est apparu que les mesures du signal émis par les beacons sont très fluctuantes et très sensibles à l’environnement.

Nous avons choisi la bibliothèque comme lieu d’expérimentation car elle est suffisamment grande pour que les résultats soit significatifs et aussi parce que la moitié est occupée par des étagères métalliques actuellement vides. Ces étagères simulent assez bien un lieu d’entreposage et sont une source de perturbation dans les mesures des BLE.

En examinant les mesures des 4 raspis de l’exp1. La configuration est simple les 4 raspis sont aux coins d’un rectangle et le beacon mesuré est situé sur l’intersection des diagonales du rectangle, donc à égale distance des récepteurs. Dans un monde idéal, on devrait voir 4 lignes superposées de points. Il faut savoir que le récepteur du raspberry écoute toutes transmissions et conservent celles qui sont intègres (pas de collision) et qui respecte le protocole. Les signaux émis par les BLE subissent aussi de nombreuses réflexion dans l’environnement, celles-ci peuvent aussi faire partie des mesures. Le signal « advertisement » émis par le beacon est transmis dans trois canaux successifs. Les beacons n’émettent pas de manière uniforme dans toutes les directions. Les récepteurs n’ont pas une sensibilité uniforme dans toutes les directions.

*Figure : 10 minutes de mesures dans exp1*

L’ensemble de ces points rendent incertain les mesures uniques qui demandent à être regroupées pour effectuer des moyennes. Malgré ces moyennes, les valeurs mesurés sont très dépendantes de l’environnement (obstacles).

Nous avons décidé de ne pas prendre de précaution particulière pour l’installation des récepteurs et de BLE. Car nous partons du principe que nous ne contrôlons pas l’environnement dans lequel ces dispositifs doivent être déployés. Et de plus, si des optimisations peuvent être faites pour améliorer les mesures, ces optimisations profiteront à toutes méthodes de localisation.

Les fichiers bruts de données enregistrés par les raspberry contenaient les information suivantes

les données des fichier log ont le format suivant :

1463733364817,cd:b8:d8:e8:cb:65,fda50693a4e24fb1afcfc6eb07647825,10001,19641,-59,-55

1463733364818,cb:d1:5a:06:b3:d7,0201061107d881c91ab99996abba4086,34688,8204,-18,-47

1463733364835,f3:be:ac:f7:79:0f,0201061107d881c91ab99996abba4086,34688,8204,-18,-50

Avec dans l’ordre:

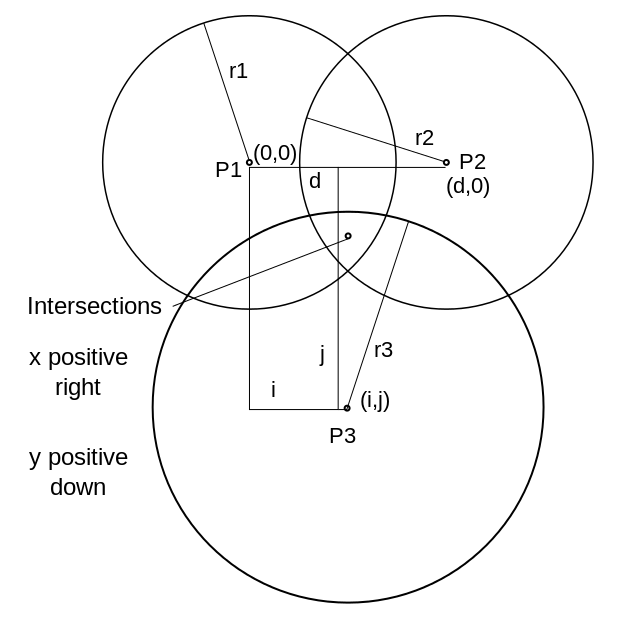
* Timestamp: 1463733364818, le moment en milliseconde de la mesure
* mac du BLE: cd:b8:d8:e8:cb:65, la mac adresse du BLE
* uuid: fda50693a4e24fb1afcfc6eb0764782, universal user id du BLE
* major: 10001, numéro majeur du BLE
* minor : 19641, numéro majeur du BLE
* rssi at 1 meter : -59, valeur du signal à 1 mètre en dbm
* rssi:-55, valeur du signal en dbm

## Méthodes de localisation

Nous nous sommes concentrés sur deux méthodes de localisations.

### La trilatération

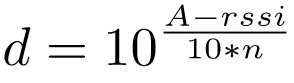
« La **trilatération** est une méthode [mathématique](https://fr.wikipedia.org/wiki/Math%C3%A9matiques) permettant de déterminer la position relative d'un point en utilisant la [géométrie](https://fr.wikipedia.org/wiki/G%C3%A9om%C3%A9trie) des [triangles](https://fr.wikipedia.org/wiki/Triangle) tout comme la [triangulation](https://fr.wikipedia.org/wiki/Triangulation). Mais contrairement à cette dernière, qui utilise les angles et les distances pour positionner un point, la trilatération utilise les distances entre un minimum de deux points de référence. » (citation de Wikipédia)



*Figure : figure montrant l’intersection obtenue par trilatération*

La trilatération peut être étendue à n mesures. Nous avons utilisé le package « lemmingapex » de Scott Wiedemann (Solves a formulation of n-D space trilateration problem using a nonlinear least squares optimizer) qui permet de trouver une solution optimal à ce problème.

Avec les BLE, la difficulté est l’incertitude sur la mesure de la distance. Cette Distance est directement dévirée depuis l’équation suivante :



Où **d** est la distance, **A** la valeur du rssi à 1 mètre, **rssi** la valeur mesurée du signal, n une constante de transmission du signal dans le milieu.

Les fabricants donnent la valeur de A (elle est d’ailleurs transmise lors des « advertisement ».

*Figure : figure montrant donnant la probabilité de l’erreur de la prédiction (moyenne 5.9m). Ces résultats sont issus de l’application directe de la trilatération avec N=11.*

Ces résultats recoupent les données des autres expériences à savoir une précision de l’ordre de 5 mètres. Voir les résultats de Houcem pour plus de détail et qui améliore sensiblement la précsion.

Devant ces résultats, nous avons décidé d’explorer l’approche dite des « empreintes »

### Les empreintes

L’idée des « empreintes » est découper le site en zone est de définir que chaque zone à sa propre signature en terme de signal reçu par les récepteurs. Ceci évite de reposer sur une idéalisation du site. Les obstacles, imperfection des émetteurs et des récepteurs sont intégrés dans l’empreinte qui est associée à chaque zone de référence.

Par analogie, on peut dire que les raspberry pi sont un pixel d’un caméra et que le beacon placé sur une zone est une bougie. L’empreinte est l’image de tous les pixels pour cette bougie à cet emplacement, une tache. On pressent bien que la tache sera différente en fonction de la proximité des capteurs.

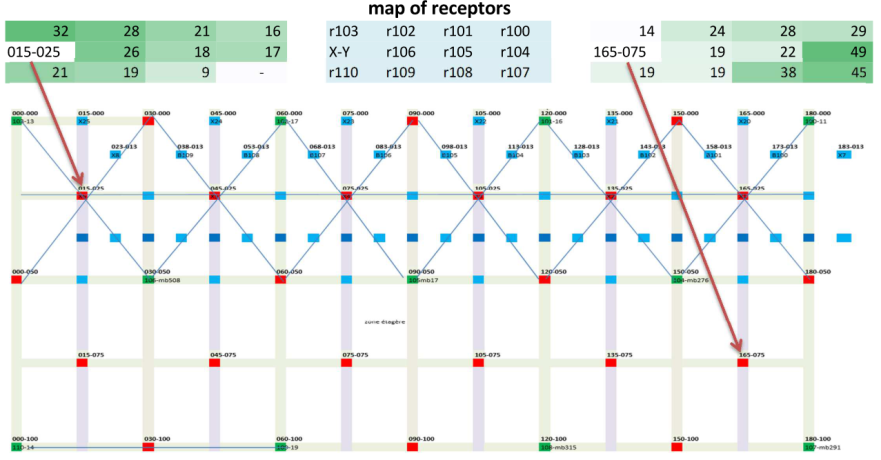


Figure : exemple de deux empreintes pour des positions diamétralement opposées.

Cette méthode est mise en place en deux phases :

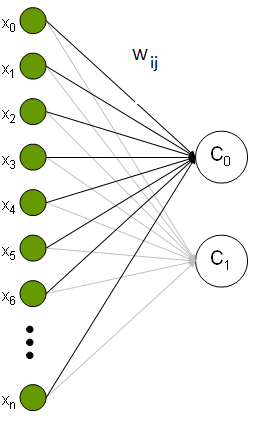
* Une phase d’apprentissage qui permet de mémoriser pour chaque emplacement (le site est recouvert par un maillage qui permet de fixer les points de référence)
* Une phase d’utilisation où une recherche consiste à retrouver l’empreinte (le point de référence) la plus proche de celle mesurée.

Il existe plusieurs façons d’implémenter la recherche de l’empreinte la plus proche (voir littérature). Nous avons choisi d’explorer une approche par réseau de neurones (Neural Network -NN).

Il faut savoir que la méthode des empreintes améliore la précision, mais elle a un coût par rapport à la trilatération car il faut procéder à un entrainement.

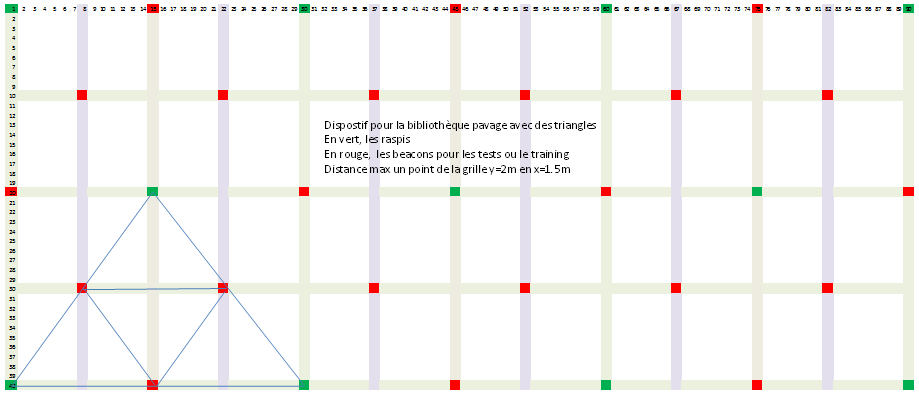
## NN et BLE

Nous avons utilisé l’algorithme de Winnow qui est une technique une technique d’apprentissage pour les exemples étiquetés. Il est similaire au perceptron mais il utilise une pondération multiplicative qui lui permet de converger plus rapidement vers une solution. Sa simplicité lui permet de d’arborder des dimensions élevées (plusieurs millions de caractéristiques en entrée et plusieurs dizaines de milliers de classes en sortie).



*Figure: Les X0 .. Xn sont les caractéristiques, Le C0 .. Cn sont les classes à prédire. Chaque classe est reliée à chaque caractéristique par une « neurone » de poids wij. Le processus d’apprentissage permet de fixer ces poids afin de maximiser la prédiction des classes.*

Pour l’utilisation de cette technique, il faut fixer les classes, les caractéristiques et fournir un jeu d’entrainement.



*Figure: les classes sont représentées par les points de la grilles qui ont été évalués.*

Les classes sont les points de la grille qui ont été évalués. Les caractérisques pour chaque évaluation sont les valeurs du signal reçu par chaque raspberry pi pour ce point (sur une certaine durée). L’ensemble de ces mesures constitue le jeu d’entrainement.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| beacon | X-Y | timestamp | Valeur moyenne mesurées pour chaque raspberry 100..110 pour la période | | | | | | | | | | |
| B14 | 000-100 | 1463733780 | 13 | 16 | 20 | 25 | 15 | 22 | 35 | 17 | 16 | 35 | 85 |
| B16 | 120-000 | 1463733780 | 45 | 84 | 31 | 28 | 36 | 27 | 27 | 25 | 27 | 31 | 15 |
| B17 | 060-000 | 1463733780 | 25 | 36 | 84 | 35 | 31 | 35 | 39 | 20 | 26 | 40 | 21 |
| MB17 | 090-050 | 1463733780 | 19 | 24 | 23 | 23 | 28 | 82 | 27 | 19 | 31 | 26 | 13 |
| MB276 | 150-050 | 1463733780 | 34 | 30 | 24 | 22 | 87 | 30 | 25 | 40 | 33 | 24 | 18 |
| MB291 | 180-100 | 1463733780 | 30 | 16 | 15 | 0 | 35 | 20 | 14 | 85 | 25 | 21 | 12 |

*tableau: Exemple de jeu de training. Les valeurs rssi négatives ont été transformées en valeur positive v=max(0, 100-rssi)*

Les classes sont donc les coordonnées xy (000-100,120-000, …)

Les caractéristiques sont des binaires (oui, non). Il s'agit d'associer une représentation des caractéristiques pour Les valeurs mesurées sur les raspberry :

Par ex : 30 16 15 0 35 20 14 85 25 21 12

Pour la valeur 30 provenant du raspi 100, on peut associer la caractéristiques **r100v30**.

On aimerait aussi conserver l’aspect continue des mesure et donc dire que 30 contient 25, 20, ...

On inclue aussi les caractéristiques r100v30, r100v25, r100v25, ...

Les NN de type Winnow ont des difficultés avec les caractéristiques mutuellement exclusive. Pour contourner cet obstacle, on va aussi inclure des valeurs croisées. Quand le raspi 100 mesurait 30 et que simultanément le raspi 102 mesurait 15, on associe la feature r100v30r103v15. On complète aussi avec les valeurs incluses.

Un programme va donc transformer les valeurs initiales en un fichier d’exemples avec les caractéristiques.

#####B100-150-000-1463733780#####

r100v0r101v0 r100v0r101v5 r100v0r101v10 r100v0r101v15 r100v0r101v20 r100v0r101v25 r100v0r101v30 r100v0r101v35 r100v0r101v40 ........ r109v20r110v5 r109v20r110v10 r109v20r110v15 r109v20r110v20

#####B101-090-000-1463733780#####

... r109v30r110v15 r109v30r110v20 r109v30r110v25

Ce programme va aussi produire le catalogue d’entrainement.

B100-150-000-1463733780 150-000  
B101-090-000-1463733780 090-000  
B102-030-000-1463733780 030-000  
B103-180-050-1463733780 180-050

Une méthode simple d’évaluation et de test consiste à prélever 20% des exemples pour les tests et garder les 80% restant pour l’entrainement.

Pour effectuer les mesures d’entrainement, nous avons effectué plusieurs séries de mesure (voir annexe biblio test1, détail1, détail 2 et lignedegilles) car nous n’avions pas assez de Beacon BLE pour procéder en une seule étape. Les fichiers des expériences ont été rassemblés pour constituer un seule jeu d’entrainement.

Le jeu d’entrainement biblio(test1, détail1 et lignedegilles) : était composé de 73 positions et de 9626 exemples.(dont 1926 pour les tests)

Pour un test 80-20, pour obtenons les résultats suivant : 84% de précision (rappel 100%)

Le détail des erreurs de prédiction montre une image que le NN fait des prédictions avec des erreurs dépassant 5 mètres mais cela ne représente que 2% des cas testés.

*Figure : Evaluation des prédictions du réseau de neurone en terme de distance (moyenne 0.5m)*

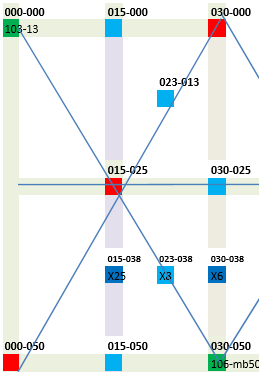
Ces résultats sont prometteurs et améliore la méthode de trilatération. Cependant, il reste un point à vérifier. Que se passe- t’il si les points ne sont pas exactement sur la grille ? Les points hors grille sont-ils proches des points de la grille qui les entourent ?

Nous avions examiné cette question dès l’expérience 2 (voir annexe : Schéma Exp2 – test hors grille) où nous avions noté quel était le point majoritaire qui était prédit pour des points hors de la grille. Les résultats nous avaient montrés qu’il existe une certaine continuité entre les points de la grille.

Nous avons refait ces expériences avec comme training les mesures (test1+détail1+détail) et avec pour test les points de la ligne de Gilles qui n’avaient pas été dans le jeu d’entrainement. Les résultats sont reportés dans le graphique suivant.

*Figure : Evaluation des prédictions de points qui ne sont pas sur la grille d’entrainement en termes de distance (moyenne 2.5 m)*

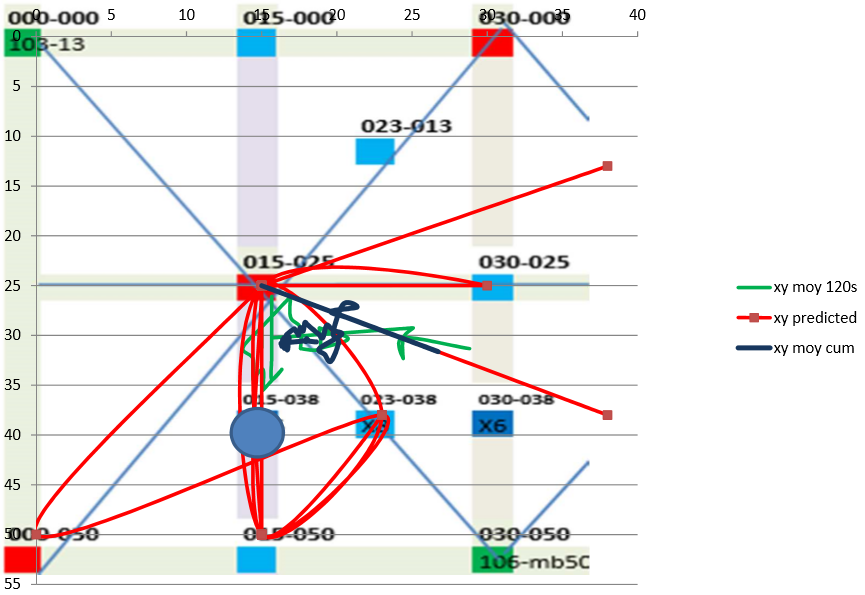
Théoriquement, l’erreur devrait être plus faible étant donné que des points sur la grille sont atteignables au plus à 2 mètres dans toutes les directions.

*Figure : A gauche : le point 015-038 n’est pas sur la grille d’entrainement, mais de nombreux points appartenant à la grille sont à moins de 2 mètres. A droite : la ligne de Gilles*

Il existe une autre possibilité d’améliorer la précision de la prédiction. Les prédictions ne sont pas systématique faites au même endroit, mais ont tendance à se faire autour de la position réelle. En faisant la moyenne des positions prédites, il est possible de se rapprocher de la position réelle.

Nous avons testé de pondération, l’une portant sur toutes les mesures et l’autre sur les 12 dernières mesures. Il est intéressant de voir que ces pondérations se rapprochent du point réel.



*Figure : superposition de la grille et des prédictions. En rouge, les prédictions du NN qui sont obligatoirement situées sur un point de la grille d’entrainement. En bleu foncé, la moyenne de toutes les mesures. En vert, la moyenne mobile sur 12 valeurs.*

Finalement, nous utiliserons la version avec la moyenne mobile. Elle permet en effet sa « mémoire » est déterminée par le nombre de valeurs sur laquelle elle porte. On peut donc déplacer un objet et au plus, après n mesures, on a sa nouvelle position approximée.

*Figure : comparaison des moyennes la position 015-038*

## Mode Localisation

L’élaboration des jeux d’entrainement ou de test peuvent se faire en collectant les données dans des fichiers et agrégeant ces derniers. Par contre, pour la localisation en temps réel, ces informations doivent être immédiatement agrégées pour être traitée par la procédure de localisation.

Nous avons utilisé la base de donnée « No SQL » Cassandra pour rassembler les données. Nous avons choisi Cassandra, car elle possède des propriétés de distribution et pourrait être capable d’enregistrer les informations transmises par des raspberrys très nombreux.

Lors de la localisation, les raspberrys alimentent 2 tables :

CREATE TABLE repoble.rawsearch(

ts10 bigint, -- timestamp à 10 secondes

ts bigint, -- timestamp à la milliseconde

raspid text, -- identité du raspi

shortname text, -- identité du ibeacon

rssi1 int, -- signal à 1 mètre

a int, -- signal mesuré

PRIMARY KEY ((ts10,shortname),raspid,ts,a)

);

CREATE TABLE repoble.raw2search(

ts10 bigint, -- timestamp à 10 secondes

shortname text, -- identité du ibeacon

processed boolean, -- process flag

PRIMARY KEY ((processed),ts10,shortname)

);

La première table enregistre les données brutes. La deuxième indique que dans une certaine tranche de temps, on a enregistré des valeurs pour un beacon (ceci permet de simplifier les programmes).

Les processus de localisation sont alimentés par ces tables.

Nous avons écrit deux processus de localisation qui sont complètement indépendant.

Le premier basé sur les « empreintes » met à jour la table

CREATE TABLE repoble.currentxy(

shortname text, --identité du ibeacon

x double, -- x

y double, -- y

PRIMARY KEY (shortname)

);

Le deuxième basé sur la « trilatération » met à jour la table

CREATE TABLE repoble.currentxyTRI(

shortname text, --identité du ibeacon

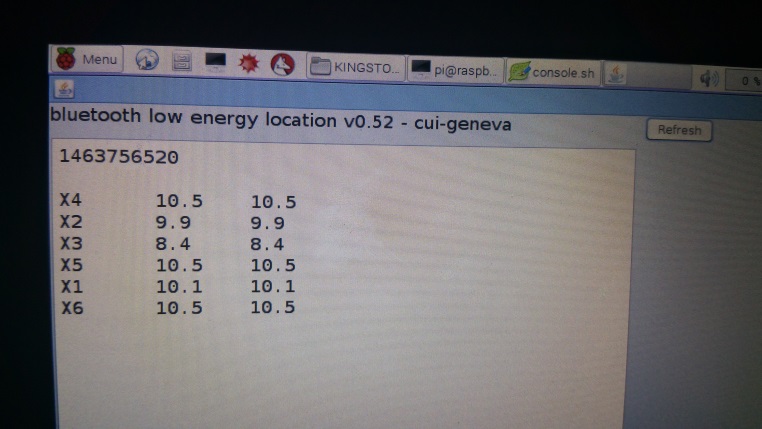
x double, -- x

y double, -- y

PRIMARY KEY (shortname)

);

Ces tables sont celles qui sont utilisées pour afficher la position des beacons sur les cartes ou dans les consoles.

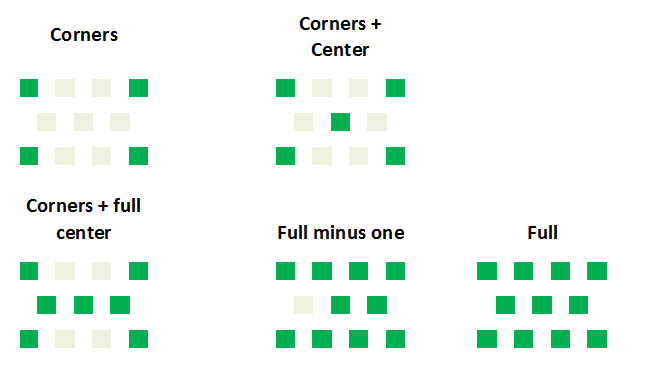


*Figure : Exemple de console (exécutée sur un Raspberry pi équipé avec un écran tactile)*

## Robustesse et Résilience

Un des problèmes majeurs de la méthode par « empreinte » est sa sensibilité aux pannes. En effet que ce passe-t’il si un ou plusieurs raspberrys ne transmettent plus les informations sur les beacons à localiser ? L’empreinte va avoir des trous et elle sera déformée. Nous avons procédé à des simulations sur les données d’entrainement, en éliminant les données en provenance des raspberrys qui n’étaient pas pris en compte.

La figure suivante montre les configurations que nous avons explorées.



*Figure : configurations explorées*

Les tests en 80/20 avec les différentes configurations montrent que si l’on élimine trop de raspberry pi, l’apprentissage devient plus fragile. Par contre, on voit que l’élimination d’un seul raspberry n’est pas un problème en soit (le fait que le score soit supérieur, n’est pas réellement significatif, il faut compter sur une marge de 2 % car les jeux de test sont tirés au hasard)

|  |  |
| --- | --- |
| Configuration | NN precision % |
| Corners | 57.8 |
| Corners + center | 64.2 |
| Corners + full center | 74.3 |
| Full minus one | 85.2 |
| Full | 84.3 |

*Figure : tableau des tests 80-20 avec les différentes configurations*

Ceci permet d’envisager une méthode d’entrainement par sur-échantillonnage et de ne garder lors de l’implantation définitive des points de mesures que les points les plus efficaces. Ceci permet aussi de vérifier qu’il n’y a pas de zone d’ombre (des zones qui ne sont pas visible par un raspberry).

En annexe, on trouve le détail de la précision pour chaque point du jeu d’entrainement. On peut constater que sans surprise la région centrale est moins bien prédite si on conserve seulement les raspberrys pi des coins. On remarquera aussi que certains points sont reconnus à 100% bien qu’ils soient au centre, il s’agit des beacons qui ont été plus entrainés que les autres, les beacons USB des raspberrys. Pour éviter cela, il faut rendre équitable l’apprentissage de chaque emplacement.

*Figure : tableau des tests 80-20 avec uniquement les 4 coins (donc une cellule de 10mx18m)*

*On voit que les empreintes perdent de leur efficacité si les cellules deviennent trop large.*

## Comparaison entre Fingerprint NN et Trilatération

L’avantage de la trilatération est qu’aucun apprentissage préalable n’est nécessaire. L’avantage des finger Print est une amélioration sensible de la précision. Nous avons effectué en parallèle les localisations avec les deux méthodes. Et nous pouvons les confronter sur des situations identiques.

Nous avons choisi deux situations : la pile de beacons et la ligne de beacons .

**La pile : *dix beacons sont empilés sur la même position (14 ;3.5).***

On voit sur le diagramme que premièrement, les dix beacons ont tous des prédictions différentes ce qui démontre une fois encore la grande variabilité des mesures. Pourtant tous les beacons appartenaient au même type (iBKS). Les cercles verts concentriques sont centrés sur le point à chercher. Leurs rayons sont respectivement 0.5, 1 et 2 mètres. On peut constater que les prédictions finger print sont meilleures que celles de la trilatération.

*Figure : dix beacons sont empilés sur la même position (14 ;3.5)*

**La ligne : *dix beacons sont sur la ligne de Gillles (… ;3.8).***

On voit sur le diagramme que les beacons suivent une trajectoire qui est pratiquement monotone (c’est-à-dire qu’il n’y a pas ou peu de points de rebroussement). Comme sur le diagramme précédent, nous avons superposé des bandes (largeur 1, 2et 4 mètres) autour de la ligne à prédire. A nouveau, on peut constater que les prédictions finger print sont plus régulières que celles de la trilatération.

*Figure : dix beacons sur la ligne (… ;3.8)*

## Historique des localisations

Nous avons aussi ajouté dans notre une base données une table HistoXY qui enregistre les positions prédites

CREATE TABLE repoble.histoxy(

ts10 bigint,

shortname text,

x double,

y double,

PRIMARY KEY ((shortname,ts10),x,y)

);

L’illustration suivante montre le suivi d’un beacon sur une période de plus de trois heures. Les emplacement où le beacon «oscille » autour d’une position sont les emplacements où il est restés plus de 5 minutes.

Il est tout à fait possible de trouver les positions où un objet s’est arrêté.

Le suivi d’un objet mobile demandera de revoir les fréquences d’échantillonage pour suivre plus rapidement.

Figure : suivi d’un beacon sur une période de 200 minutes (méthode FingerPrint)

## Evaluation des méthodes

Metric from Journal of Computer Networks and Communications Volume 2013 (Recent Advances in Wireless Indoor Localization Techniques and System Zahid Farid, Rosdiadee Nordin,and Mahamod Ismail) dans github. Cet article distingue bien les méthodes des implémentations matériels…

Performance metric

|  |  |  |
| --- | --- | --- |
| metric | Trilateration | FingerPrint |
| **Accuracy** (or location error) | 3.5 meter @ 90%  Low | 1.5 meter @ 90%  High |
| **Responsiveness** (how quickly the location estimate of a moving target is updated) | Depends on Bluetooth LE freq (1s/3min) -> 100ms / 20s  (limitation in mobility) | |
| Coverage (local🡪global) | Depends on how many raspberry are deployed (scalable) | |
| Adaptiveness (Environmental influence changes) | Easy to update x,y coord if raspberrys are moved | Need to retrain (localy)  Or continues training  No needs of xy coord. |
| Scalability | Depends on Cassandra (high) | |
| Cost and Complexity | Low | Medium |

## Conseils pour le training avec NN

* L’espace à couvrir doit être entièrement accessible pour une période de temps limitée (2-4 heures).
* Une carte de l’espace doit être disponible (précision 10 cm)
* L’emplacement définitif des raspberrys doit être déterminé.
* On peut sur-échantillonner et éliminer des raspberrys à la mise en service.
* La grille de training doit être établie (tous les points à apprendre)
* L’idéal est de faire un seul training, mais on peut découper le training en zone si l’on a pas assez de beacon. L’idéal est que tous les points soit appris sur la même durée.
* La hauteur, la position de détection des beacons doivent être adaptées à la tâche finale de recherche (position entre 60 et 120 cm, vertical, par exemple).
* On peut varier la position des beacons pour un même point d’apprentissage pour diversifier les situations à reconnaître.
* On peut toujours ajouter ou remplacer certain exemple d’apprentissage au cours du temps.
* Il est nécessaire de procéder à des tests périodiques pour vérifier que les conditions d’apprentissage n’ont pas varié.

## Amélioration Trilatération (Houcem ?)

### Distance

Le RSSI (Received Signal Strength Indication) est utilisé pour le calcul de distance séparant l’émetteur du récepteur. On trouve deux formules :

* **Distance = 10txPower-rssi/10\*n [courbe orangée]**avec n est une constante de propagation
* **Distance = 0.89976 \* (rssi/txPower)7.7095+0.111 [courbe bleue]**

Une constante de calibrage (txPower) doit être fixée à l’avance présentant la valeur de RSSI à un mètre du beacon. En général, cette valeur varie entre -58db et -60db selon le milieu de travail. Cette valeur est transmise par le beacon dans le message d’*advertisement*

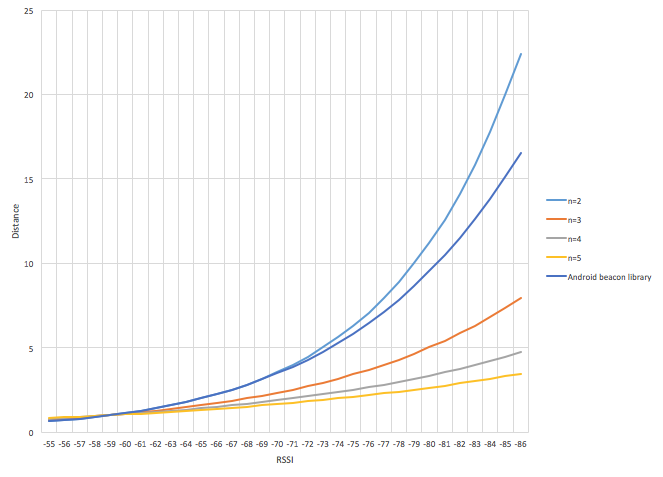


Figure : Courbe de distances

Cette figure présente la différence entre les deux formules présentées ci-dessus. La première formule demande à chercher évaluer la valeur de n. La deuxième ayant éliminer n (les deux courbes sont équivalentes pour n=2.35 pour des distances de moins 5 m). On utilisera par la suite la deuxième formule représentée par la courbe bleue foncé et l’on adaptera le txPower pour les beacons.

Ensuite, on essayera de calculer quelques distances à partir d’une moyenne de 300 valeurs (première expérience du jardin) et estimer le calibrage nécessaire pour les différentes beacons utilisés. (Voir annexe : résultat de trilateration)

On remarque que les résultats sont très sensibles aux calibrages des beacons, ce qui est à son tour sensibles à plusieurs facteurs comme l'environnement, présence d’obstacles, direction des antennes, matériaux, etc.

### Multilateration

On essayera par la suite de trouver le meilleur nombre de distance nécessaire pour la trilateration.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Expérience | tilateration | Ecart  (-58db) | B (-58db) | X(-58db) | MB  (-58db) | Ecart  (-57db) | B (-57db) | X(-57db) | MB  (-57db) |
| Biblio -1 | 3 | 3.14 | 3.32 | 2.13 | 4.28 | 3.07 | 3.28 | 2.13 | 3.96 |
| Biblio -1 | 4 | 2.86 | 3.24 | 1.82 | 3.52 | 2.76 | 3.15 | 1.80 | 3.30 |
| Biblio -1 | 5 | 2.64 | 3.05 | 1.88 | 2.81 | 2.64 | 2.94 | 2.00 | 2.91 |
| Biblio -1 | 6 | 2.79 | 3.23 | 2.19 | 2.62 | 2.88 | 3.38 | 2.24 | 2.64 |
| Biblio -1 | 7 | 2.99 | 3.24 | 2.01 | 3.85 | 3.09 | 3.40 | 2.10 | 3.84 |
| Biblio -1 | 8 | 3.09 | 3.32 | 2.33 | 3.71 | 3.29 | 3.61 | 2.33 | 3.98 |
| Biblio -1 | 9 | 3.32 | 3.74 | 2.46 | 3.60 | 3.68 | 4.16 | 2.61 | 4.10 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Expérience | tilateration | Ecart (-58db) | B (-58db) | X(-58db) | MB(-58db) |
| Biblio -2d | 3 | 3.41 | 4.25 | 2.65 | 2.71 |
| Biblio -2d | 4 | 2.81 | 3.09 | 2.50 | 2.68 |
| Biblio -2d | 5 | 2.49 | 2.84 | 2.42 | 1.71 |
| Biblio -2d | 6 | 2.52 | 2.66 | 2.88 | 1.55 |
| Biblio -2d | 7 | 2.84 | 2.59 | 3.16 | 2.90 |
| Biblio -2d | 8 | 3.27 | 3.01 | 3.61 | 3.30 |
| Biblio -2d | 9 | 3.64 | 3.06 | 4.31 | 3.88 |

En se basant sur plusieurs tests réalisés sur les différentes scénarios fournis (Voir annexe : Résultats de trilateration), l’utilisation de cinq et six distances donne le meilleur résultat. La figure ci dessous montre l’écart d’erreur moyen sur plusieurs tests avec plusieurs distances.

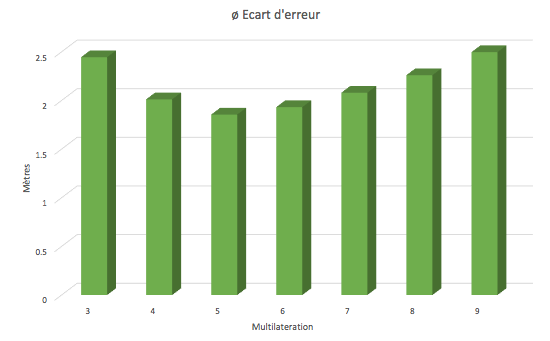


Figure : Multilateration

### Amélioration

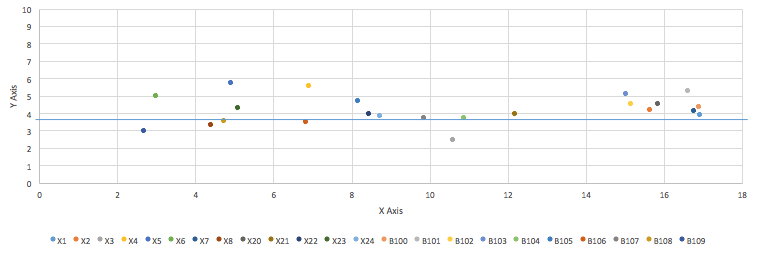
Afin d’améliorer l’écart d’erreur de notre algorithme, on prendra les positions des récepteurs, préconfigurées à l’avance, si le beacon y est à moins de 0.5 mètres.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Expérience | tilateration | Ecart  (-58) | B  (-58) | X  (-58) | MB  (-58) | Ecart  (-57) | B  (-57) | X  (-57) | MB  (-57) | Ecart  (-59) | B  (-59) | X  (-59) | MB  (-59) |
| 3 LG | 3 | 1.71 | 4.11 | 2.39 | 0.00 | 1.65 | 3.98 | 2.25 | 0.00 | 1.69 | 3.84 | 2.48 | 0.00 |
| 3 LG | 4 | 1.21 | 2.20 | 2.21 | 0.00 | 1.23 | 2.21 | 2.27 | 0.00 | 1.20 | 2.21 | 2.19 | 0.00 |
| 3 LG | 5 | 1.11 | 1.50 | 2.46 | 0.00 | 1.09 | 1.61 | 2.30 | 0.00 | 1.16 | 1.50 | 2.61 | 0.00 |
| 3 LG | 6 | 1.13 | 1.62 | 2.42 | 0.00 | 1.05 | 1.57 | 2.19 | 0.00 | 1.22 | 1.72 | 2.63 | 0.00 |
| 3 LG | 7 | 1.17 | 1.64 | 2.54 | 0.00 | 1.08 | 1.43 | 2.40 | 0.00 | 1.31 | 1.88 | 2.81 | 0.00 |
| 3 LG | 8 | 1.27 | 1.75 | 2.77 | 0.00 | 1.23 | 1.66 | 2.72 | 0.00 | 1.41 | 1.95 | 3.05 | 0.00 |
| 3 LG | 9 | 1.37 | 1.84 | 3.01 | 0.00 | 1.55 | 1.82 | 3.61 | 0.00 | 1.42 | 1.99 | 3.08 | 0.00 |

On se retrouve avec un écart d’erreur de :

* ≈ 2.5 mètres pour les beacons de type X.
* ≈ 1.5 mètre pour les beacons de type B.

La figure ci-dessous présente les résultats de notre algorithme pour le scénario « ligne de Gilles » avec un calibrage égale à (-58db) pour tous les beacons.

 Figure : Ligne de gilles

La figure ci-dessous présente le cumul des erreurs. On a donc 90% de faire une prédiction de position à moins de 3.5m.

La figure ci-dessous présente le cumul des erreurs. On a pris les signaux des 4 raspberrys des coins donc la taille de la cellule est de 10m x 18 m. On a donc 90% de faire une prédiction de position à moins de 11 m.

Ceci confirme que la précision de la trilatération est fortement liée à la taille de la cellule.

## Visualisation des résultats (Alexandre)

### Web Application

### <http://192.168.0.246:9000>

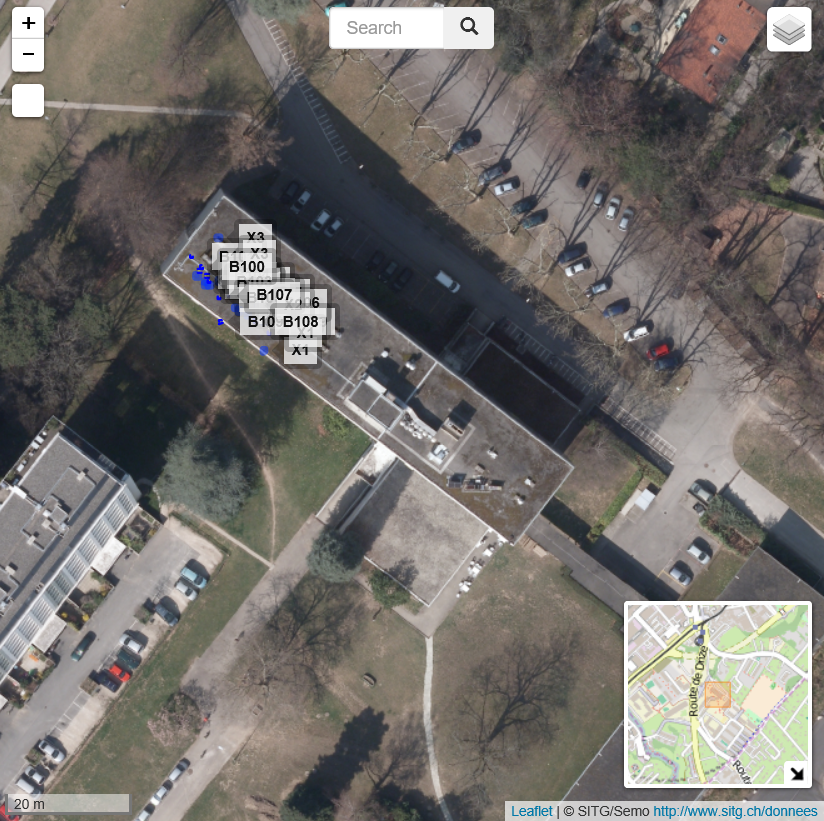


Figure : vision global de l’application avec le plan d’ensemble

Contrôle de l’application

|  |  |  |
| --- | --- | --- |
| boutons de zoom    Vue d’ensemble / déplacement possible | champ de recherche ( exact et flou) | Fonds de carte / plans  Couches |

L’application permet de charger un plan détaillé pour une zone.

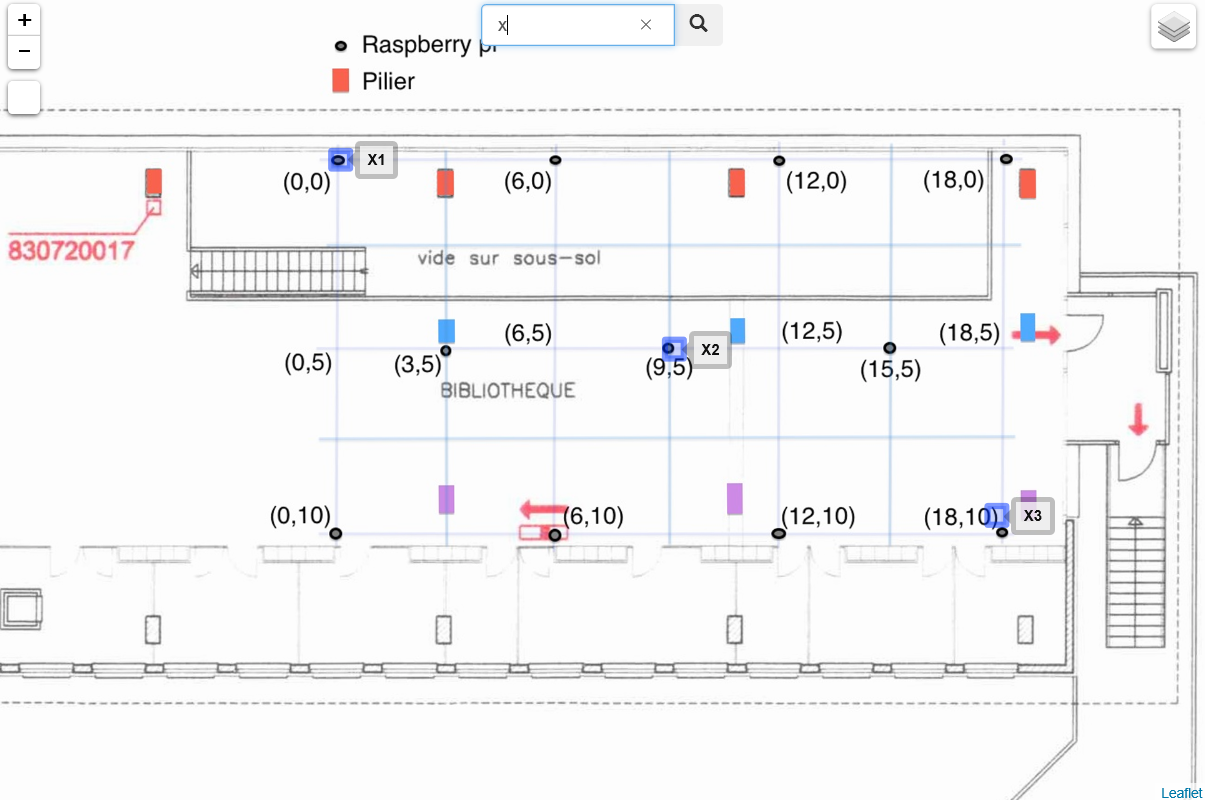


Figure : vision avec la couche plan, en sélectionnant que les beacons contenant X

L’application peut être déployée sur des téléphones mobiles



Figure : Exemples d’écran sur téléphone androïd

### Cartes

### Normes

### Sitg,….

### Project tango

Voir <https://get.google.com/tango/>

Voir aussi http://uk.pcmag.com/smartphones/42448/news/google-future-phones-will-understand-see-the-world

Avec le premier téléphone tango compatible (voir lenovo) il sera possible d’intégrer plus facilement dans un environnement réel une couche virtuelle (réalité augmentée). Ceci permettra donc de visualiser plus facilement les emplacements des objets

## Conclusions et perspectives

### Bilan des expériences

Nous avons montré lors de ce travail qu’il était possible de déployer une couverture de récepteurs Bluetooth pour permettre la localisation de beacon BLE. Deux types de localisation ont été étudiées ; la trilatération et la méthode des empreintes implémentées avec un réseau de Neurones.

La trilatération est rapide à déployée mais à une précision inférieure à la méthode des empreintes qui par contre nécessite un jeu d’entrainement qui doit être mesuré sur le site.

Sans doute qu’un mixte des deux méthodes est envisageable pour les zones nécessisant peu de précision, la trilatération est à employer alors que la méthode des empreintes peut être utilisées pour affiner les prédictions dans les zones où une meilleure précision est nécessaire.

Il est possible de reconstruire l’historique des positions prises par un objet. Le système peut estimer avec une précision de +/- 20 secondes les emplacements et les périodes d’arrêt des objets. Les déplacements sont plus difficiles à estimer. La vitesse des objets doit être inférieure à 2m/s pour les grilles que nous avons explorées.

Le système peut aussi détecter le moment du passage d’un beacon en particulier.

### Pour aller plus loin

**Tester sur un site industriel**

Bien que nous n’ayons pris aucune précaution particulière, l’essai de l’infrastructure actuelle est à expérimenter pour avoir une idée plus exacte des difficultés propres à des sites industriels. Ceci permettra aussi à plus d’acteurs concernés d’avoir une expérience avec la localisation dans leur environnement de travail. Ceci permettra aussi de vérifier l’impact de l’erreur sur les tâches de localisation et de fixer la taille des cellules pour attendre la précision demandée.

**Evaluer l’implémentation avec le WIFI**

Les méthodes (trilatération et fingerprint) sont aussi directement transcriptible avec une technologie WIFI. Il faut cependant savoir que les points d’accès WIFI, même s’ils sont nombreux, sont relativement espacés et donc ceci peut imputer le précision. Et que lors du déploiement, ils ont été positionnnés dans le but pour couvrir l’espace pour permettre un accès wifi et non pour permettre la localisation. De plus, les constructeurs (Cisco) ont des solutions de localisation qui leur sont propres. Cette voie est à examiner dans le cas d’un déploiement opérationnel. Dans le cas d’un test, l’utilisation de l’installation existante risque d’être un frein aux expérimentations possibles.

**Que veux-t’on localiser ?**

Dans le cadre d’une expérimentation sur site, il est important que la localisation/tracking soit associé à une tâche.

* Suivi d’un mini lot ?
* Suivi des cuves Roxanes ?
* Retrouver une personne ayant un malaise ?
* Etc.

**Connecter la localisation avec des contraintes (HASMAT)**

Il est possible avec les données de localisation de la BD d’émettre des messages d’alerte si par exemple :

* Un beacon associé avec un produit A se trouve à proximité d’un beacon associé avec un produit B. et qu’une règle dit que A et B doivent être séparé de 6 m.
* Un beacon associé avec un produit A se trouve dans une zone X. et qu’une règle dit que A ne peut être dans la zone X.

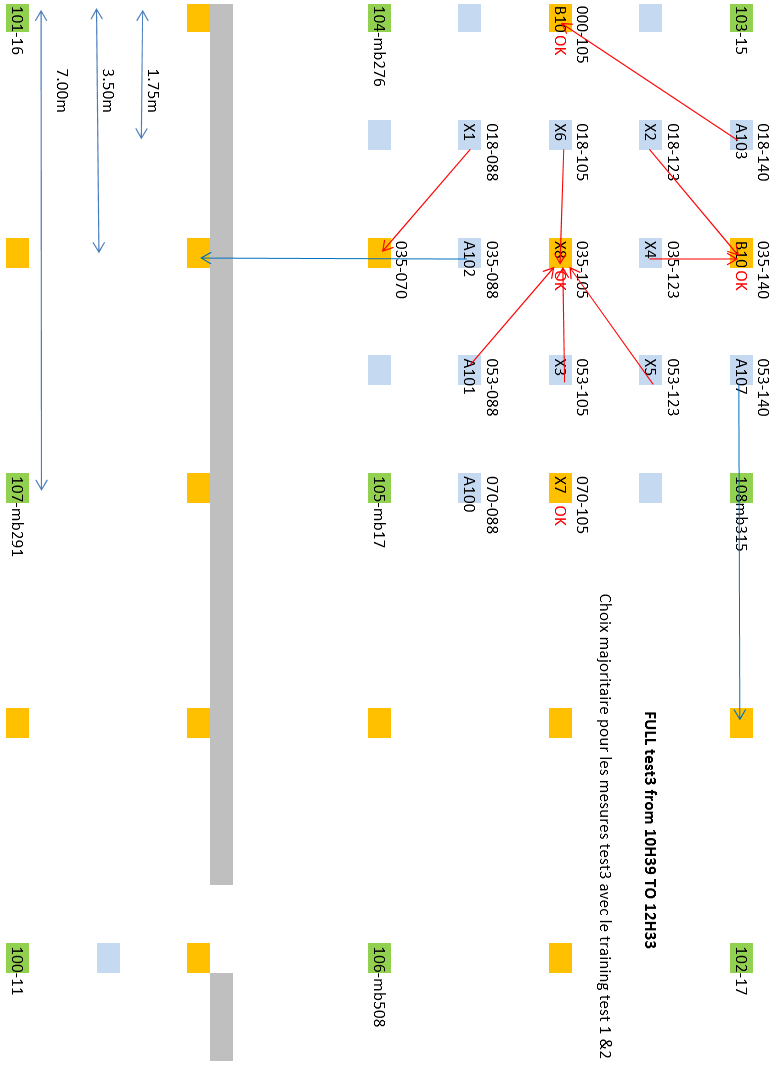
**Améliorer la recherche**

Lors de la démonstration, nous avons vu que l’utilisation du plan peut s’avérer fastidieuse. L’utilisation de la réalité virtuelle et/ou de la réalité augmentée peut faciliter à l’appropriation de l’espace. La réalité augmentée peut être annotée afin de pouvoir repérer des coordonnées (x,y,z). Ceci peut faciliter la tâche lors de la construction du training.

## Annexe : Schéma EXP1



## Annexe : Schéma Exp2 – test hors grille



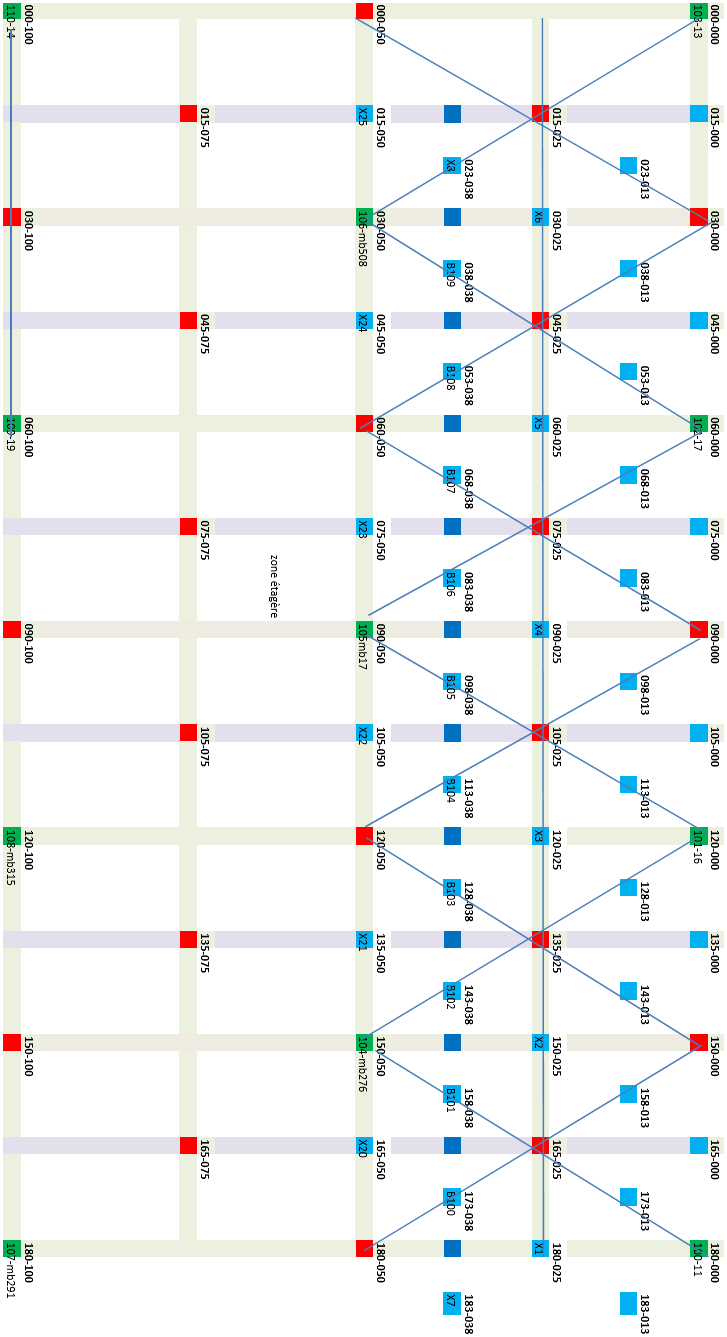
## Annexe : Schéma Biblio-test1

****

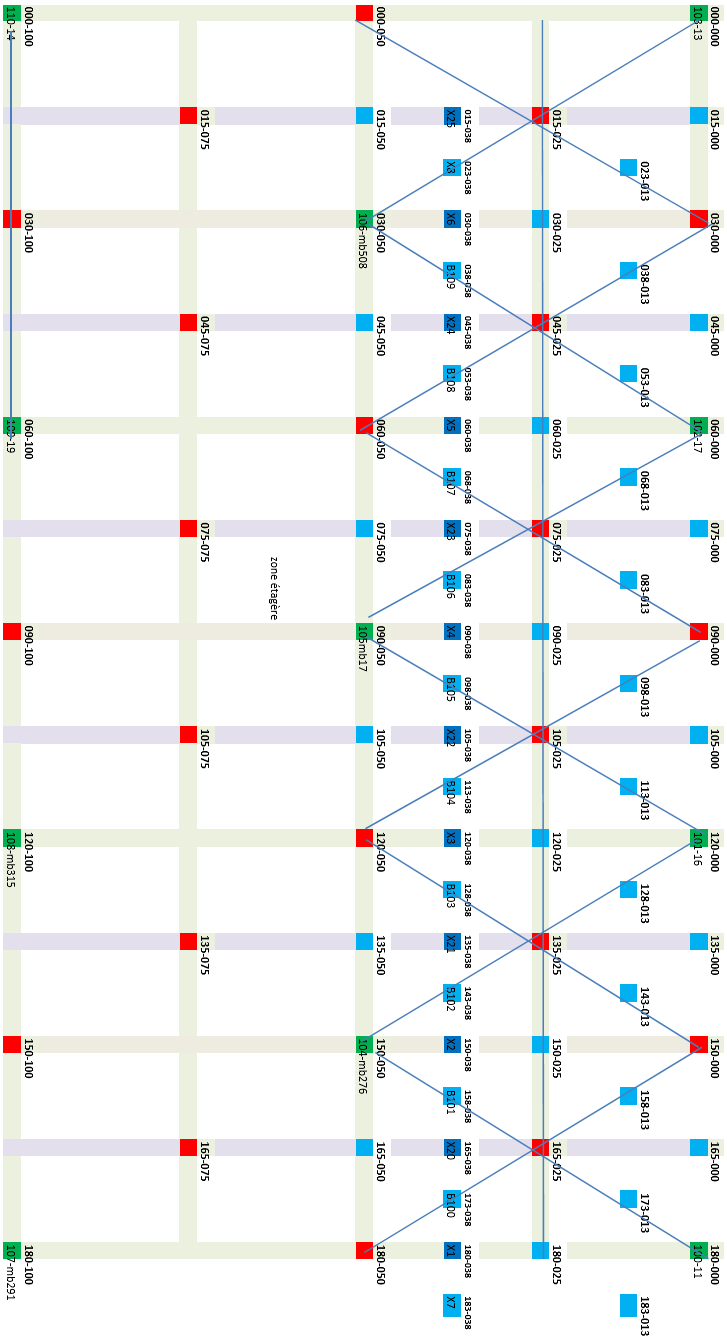
## Annexe : Schéma Biblio-détail1



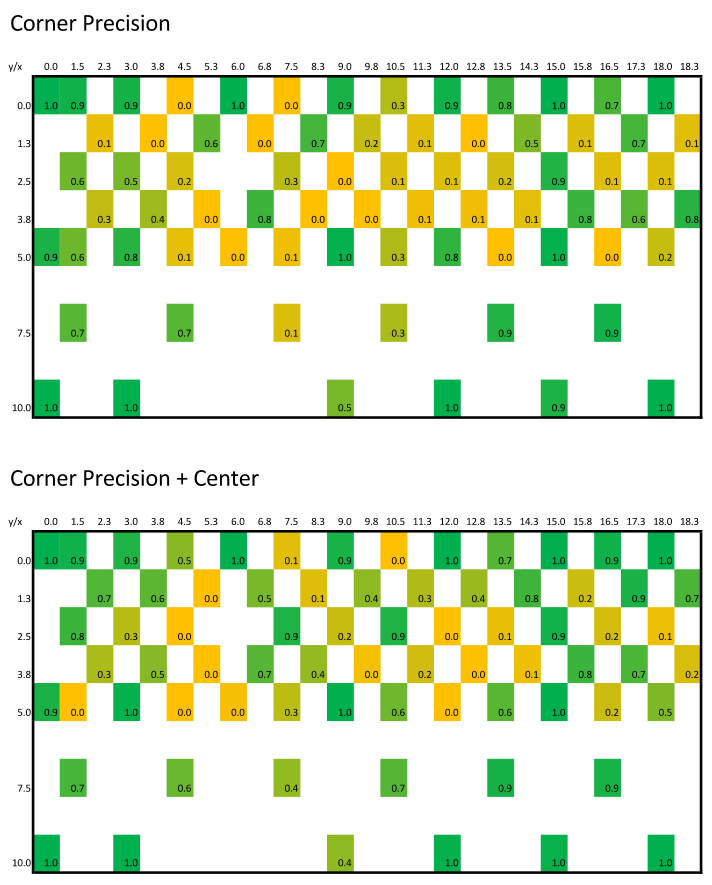
## Annexe : Schéma Biblio-détail2

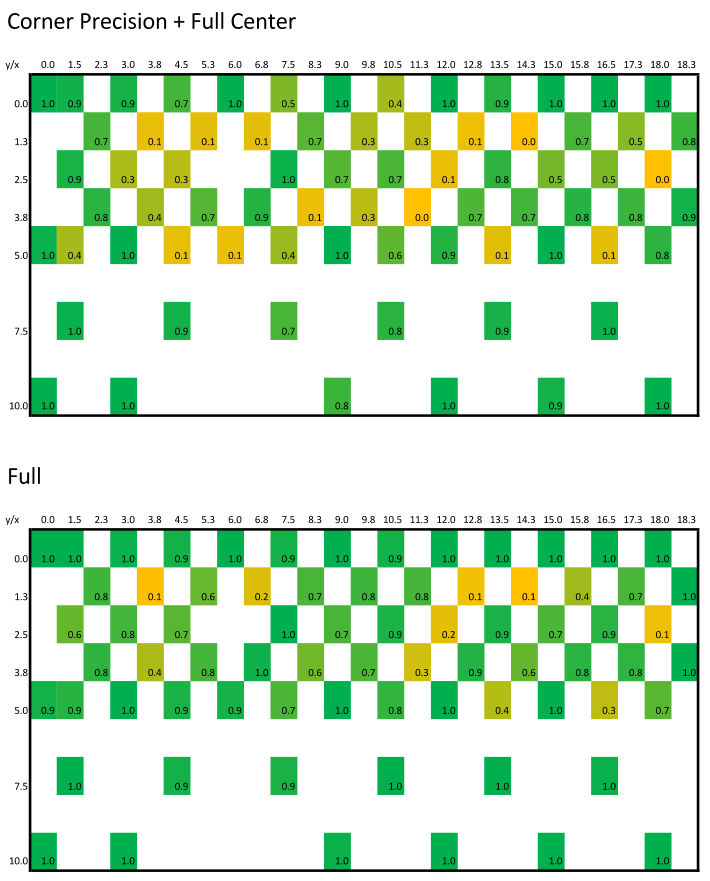


## Annexe : Schéma Biblio-ligne de Gilles



## Annexe : Détail de la précision avec différente configuration (80/20)





## Annexe : Nordic Semiconductor nRF51822

(From A Guide to Selecting a Bluetooth Chipset argenox.com 19-5-2016)

Nordic Semiconductor released the nRF51822 after the nRF8001, and it is one of the most popular BLE devices.

It’s an integrated solution that appeals to developers because of its small size. It integrates a Cortex­M0 CPU, which means there are many compiler options for development. In fact, Nordic explains how to setup a development environment using an ARM GCC compiler, Eclipse and a few other tools. This can enable any developer to use the RF51822 with free tools. For product development it is usually recommended use a vendor toolchain such as IAR or Keil. Some of the distinguishing values of the nRF51822:

• Bluetooth Supported: v4.1 BLE Single Mode

• Complete BLE solution with integrated BLE stack.

• Integrated On­Chip ARM Cortex­M0 with 256kB or 128kB flash and 16kB to 32kB RAM

• Available in 6x6mm QFN and 3.5×3.8mm WLCSP packages

• Supply Voltage: 1.8V­3.6V

• $2.31/1ku

Remember that TI had software defined radio in their chipset? Nordic’s BLE chipset is very much a multi­protocol device. It supports support standard 2.4GHz proprietary radio functionality with three different data rates up to 2Mbps. This can help you future­proof your product if you ever need some proprietary communication. It’s completely compatible with Nordic’s nRF24L radios, which have been used in quite a few products.

Nordic recently announced the nRF51822 running IPv6 which opens the door to control products with the nRF51822 over the internet, as opposed to using Wi­Fi (and at a much lower cost). This can become extremely appealing to product developers.

One advantage of the nRF51822 is that it is a mature device currently in its third revision, so you know that Nordic has worked out many of the earlier issues to make your product more reliable.