CLIPS

Communication & Localization with Indoor Positioning Systems

Università di Padova

Documentazione sullo studio effettuato

leaf.gruppo@gmail.com

Versione	1.00
Data Redazione	14/05/2015
Redazione	Marco Zanella
Verifica	
Approvazione	
Uso	Esterno
Distribuzione	Miriade S.p.a

Indice

1 Introduzione

1.1 Scopo del documento

Questo documento ha lo scopo di raccogliere tutte le scelte effettuate dal gruppo *Leaf* durante lo svolgimento del progetto *CLIPS*. Verranno quindi trattate scelte riguardanti l'hardware e software utilizzato, l'impianto su cui è stato realizzato il software, la soluzione progettata e infine le problematiche riscontrate.

1.2 Scopo del progetto

Lo scopo del progetto è sviluppare un'applicazione mobile per la navigazione indoor, sfruttando la teccnologia BLE e l'utilizzo dei beacon.

1.3 Riferimenti

- Documenti riguardanti la progettazione del codice:
  link repo documenti o luogo dove scaricarli;
- Codice sviluppato:
  link repo del codice;
- Javadoc:
  link javadoc.

2 Scelte hardware

2.1 Dispositivi mobile utilizzati

- lista cellulari

2.2 Beacon

Non sono state rilevate differenze significative di funzionamento tra i due tipi di beacon forniti. Sono stati infatti utilizzati indistintamente beacon da interno e da esterno.

3 Scelte software

Di seguito sono riportate le scelte software fatte.

3.1 Android

È stato scelto di sviluppare un'applicazione per dispositivi mobile Android rispetto ad IOS per la disponibilità prevalente di dispositivi Android.

- Versione minima 4.4
- Versione massima 6.0

3.2 SQLite

Libreria che implementa un DBMS SQL transazionale senza la necessità di un server. Viene utilizzata per salvare e gestire le mappe scaricate e installate nel dispositivo e il relativo contenuto.

- Versione utilizzata 3.9.2

3.3 AltBeacon

Libreria che permette ai sistemi operativi mobile di interfacciarsi ai beacon, offrendo molteplici funzionalità. Viene utilizzata per permettere la comunicazione tra l'applicativo Android e i beacon.

- Versione utilizzata 2.02

3.4 JGraphT

Libreria Java che fornisce funzionalità matematiche per modellare grafi. Viene utilizzata per la rappresentazione delle mappe e per il calcolo dei percorsi.

- Versione utilizzata 0.9.1

3.5 Gson

Libreria Java che fornisce funzionalità per la gestione di oggetti JSON. Tale libreria è utilizzata la gestione del download delle mappe da remoto.

- Versione utilizzata 2.6.2

3.6 Dagger

Libreria Android utilizzata per effettuare la dependency injection. Viene utilizzata per la creazione dei singleton.

- Versione utilizzata 2.0

3.7 Picasso

Libreria per la gestione delle immagini in remoto. Viene utilizzata per scaricare le immagini utilizzate durante la navigazione.

- Versione utilizzata 2.5.2

4 Impianto

4.1 Area indoor

L'area indoor scelta per la sperimentazione del software sviluppato è l'edificio della Torre Archimede. Tale scelta è stata fatta poichè è l'edificio in cui si tengono i corsi del terzo anno della laurea di informatica e quindi il luogo più comodo dove effettuare sperimentazioni.

4.2 Posizionamento dei beacon

Durante il posizionamento e la configurazione dei beacon sono state seguite alcune **best pratices**, di seguito riportate:

- posizionare il più possibile i beacon sopra il gli ostacoli(quali persone o oggetti). Il luogo migliore risulta essere il soffitto;
- posizionare i beacon in modo tale che esista un passaggio rettilineo tra il luogo in cui è posizionato un certo beacon e il luogo in cui invece è posizionato un altro beacon;
- impostare la frequenza di trasmissione in modo tale ci sia meno interferenza possibile tra i segnali di due beacon differenti. Ciò implica di impostare una potenza di trasmissione abbastanza ridotta ma permette di poter fare delle assunzioni sul luogo in cui un dispositivo si trova nel caso in cui venga rilevato il segnale di un certo beacon.

4.3 Utilizzo degli identificativi dei beacon

Ai tre identificativi di un beacon sono stati assegnati tre significati differenti:

- **UUID**: è utilizzato per identificare i beacon che devono essere rilevati dal nostro applicativo, quindi è stato fissato un UUID unico per tutti i beacon utilizzati, filtrando in questo modo eventuali altri beacon;
- **Major**: è utilizzato per identificare tutti i beacon di uno stesso edificio. È stato quindi impostato lo stesso Major pre tutti i beacon utilizzati nella Torre Archimede. Ciò è necessario soprattutto per l'associazione con le mappe degli edifici stessi. Difatti ogni mappa è identificata dal Major dell'edificio che rappresenta e, in questo modo, qualsiasi sia il beacon rilevato in un certo edificio, si può facilmente identificare l'edificio di appartenenza;

- **Minor** è utilizzato per identificare uno specifico beacon in un certo edificio. Tale identificativo è stato diviso in due parti:
  - le prime due cifre sono utilizzate per identificare il piano di appartenenza di uno specifico beacon. È possibile ricavare il piano quindi effettuando la divisione intera tra il numero che rappresenta il Minor e 10000;
  - le ultime tre cifre sono utilizzate per identificare uno specifico beacon. È possibile ricavare il piano quindi calcolando il resto della divisione intera tra il numero che rappresenta il Minor e 10000.

  Tale scelta non permette di mappare edifici con più di 65 piani e 535 beacon per piano.

5 Mappatura edificio

Per **mappatura edificio** si intende la costruzione teorica di un **grafo pesato** che rappresenta l'edificio in cui abilitare la funzionalità di navigazione offerta dall'applicazione.

Per un edificio quindi è necessario individuare i due elementi che compongono un grafo:

- Vertici;
- Archi.

5.1 Vertici: Region of Interest & Point of Interest

I **vertici** rappresentano le aree coperte dal segnale di un singolo beacon, definite Region Of Interest (ROI). Ogni ROI, identificata da un unico beacon, è definita come un insieme di possibili destinazioni da raggiungere all'interno dell'edificio da mappare. Tali destinazioni sono definite Point Of Interest (POI) e rappresentano nella realtà tutti i punti di interesse di un utente.

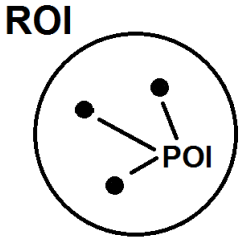

Figura 1: Visione teorica di ROI e POI contenuti in essa

Una ROI e quindi un beacon rappresenta più POI per soddisfare l'obiettivo di **massimizzare l'area da coprire** e **minimizzare il numero di beacon** da utilizzare.

Poiché un POI può essere che abbia più punti d'accesso, si pensi ad una stanza con più di una porta d'accesso, è previsto che un singolo POI possa appartenere a più ROI. Il risultato è una relazione N a N tra ROI e POI. Nella figura 2 si vede come il POI al centro appartiene sia alla ROI_1 sia alla ROI_2.

5.2 Gli archi: Edge

Gli **archi** (Edge) invece rappresentano i collegamenti reali tra una ROI e un'altra ROI, per esempio un corridoio o delle scale. Essi quindi possono

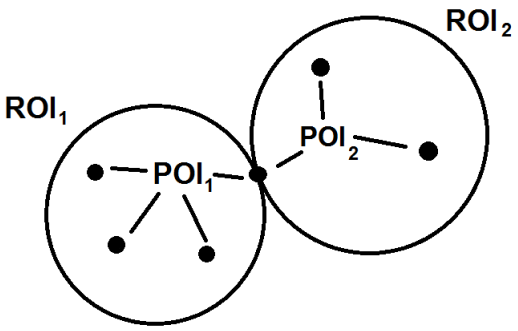

Figura 2: Visione teorica di un POI che appartiene a più ROI

essere di varie tipologie e quindi pesati in modo diverso. Le tipologie fin'ora individuate sono:

- Default Edge;
- Stair Edge;
- Elevator Edge.

Ogni Edge contiene le informazioni necessarie per arrivare correttamente alla ROI di arrivo dalla ROI di partenza.

5.3 Percorso

Un POI è una possibile destinazione dell'utente ossia il punto d'arrivo di un possibile percorso mentre una ROI è l'area in cui l'utente si trova, grazie al rilevamento dei beacon. Gli Edge e i ROI tra queste due entità rappresentano il percorso da percorrere.

Dato quindi un punto di partenza e uno di arrivo abbiamo la necessità di calcolare il percorso più corto per raggiungere il punto di arrivo. Poiché lavoriamo rappresentando gli spazi interni dell'edificio in un grafo possiamo sfruttare tutta la conoscenza matematica su questi. Implementando l'algoritmo di Dijkstra otteniamo il percorso più corto da una ROI ad un'altra.

Poiché il punto di destinazione è un POI e questo può appartenere a diverse ROI nel calcolo del percorso bisogna identificare il percorso migliore tra la ROI di partenza e le ROI di cui il POI fa parte.

5.4 Mappare più piani

Se consideriamo ogni piano di un edificio un grafo possiamo connettere tali grafi con nuovi Edge nei punti reali in cui questi piani si collegano, scale e ascensori. Otteniamo così da due grafi un grafo più grande.

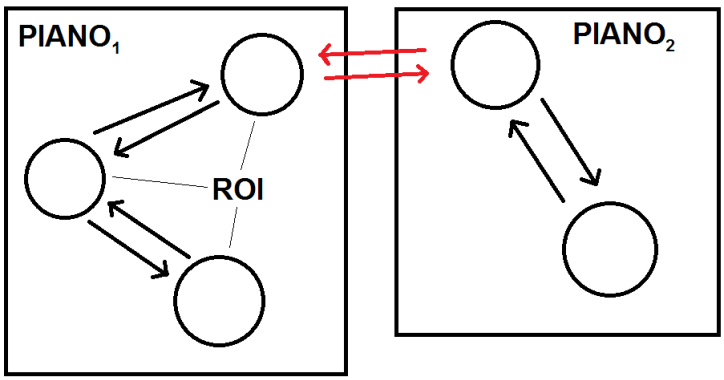

Figura 3: Visione teorica di un collegamento tra due grafi ossia piani di un edificio

5.5 Procedura per la mappatura di un edificio

Nel caso si volesse che l'applicazione CLIPS supporti un altro edificio seguire questi passaggi:

1. procurarsi una planimetria dell'edificio;
2. identificare tutti i POI dell'edificio di ogni piano;
3. raggruppare i POI vicini in ROI garantendo che:
   - tutto l'edificio sia coperto dal segnale dei beacon;
   - non ci sia eccessiva distanza tra una ROI all'altra;
   - una ROI non contenga troppi POI;
4. collegare ogni ROI alle ROI raggiungibili da essa senza passare per altre ROI e identificando la tipologia di Edge (normale, scale, ascensori...). **NOTA:** se da un ROI posso raggiungere un'altra ROI e viceversa dovrò necessariamente documentare due Edge distinti.
5. per ogni Edge identificato rilevare le informazioni necessarie per descriverlo in toto.
   - distanza;

- coordinata magnetica;
- azione che l'utente deve seguire;
- descrizione dettagliata;
- ROI di partenza;
- ROI di fine;

Tali informazioni saranno fondamentali perché l'applicazione offra il maggior numero di informazioni all'utente;

6. assegnare ai beacon da utilizzare un Major univoco che identifica l'edificio che si sta mappando;
7. assegnare ad ogni beacon un Minor in cui le premi due cifre identificano il numero del piano e le ultime tre identificano la ROI in quel piano;
8. inserire nel server delle mappe tali informazioni. Per ulteriori informazioni su quest'ultimo punto si rimanda alla sezione **??**.

6 Soluzione proposta

La soluzione proposta prevede quindi di calcolare un percorso all'interno del grafo che rappresenta l'edificio. Essendo gli archi di tale grafo pesati è possibile calcolare tali pesi dinamicamente sulla base delle preferenze espresse da un utente. Ciò viene fatto semplicemente sommando una costante al peso degli archi che l'utente non vorrebbe incontrare nel suo percorso. Il peso degli archi del grafo dell'edificio dipende fortemente dalla natura dell'arco stesso. Per archi normali è rappresentato dalla sua lunghezza (approssimativa) in metri, mentre per scale e ascensori è regolato da funzioni dipendenti dal numero di piani che l'arco attraversa.
Il calcolo di tali pesi sfrutta due funzioni esponenziali simmetriche rispetto l'asse y. L'idea è che al crescere dei piani il peso delle scale aumenti, mentre quello dell'ascensore diminuisca.

$$fe(x) = e^{x-k}$$
$$fs(x) = e^{-(x-k)}$$

Dove *fe(x)* rappresenta la funzione per gli ascensori, *fs(x)* per le scale, x è la differenza di piani che l'arco attraversa e dove k è una costante tendente al numero di piani per cui prendere le scale e gli ascensori hanno lo stesso peso. Noi abbiamo preso un numero di poco inferiore a 2 cioè k = 1.9999. Ciò vuol dire che per noi il numero di piani da attraversare prendendo scale o ascensori è 2, con preferenza per l'ascensore per arrivare al secondo piano. Sono state scelte tali funzioni per avere archi tutti con peso positivo ed avere la possibilità di utilizzare Dijktra.