Alma Mater Studiorum - Università di Bologna

Campus di Cesena

Scuola di Ingegneria e Architettura
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

Progettazione e realizzazione di un Indoor Positioning System basato su geomagnetismo e sensor fusion

Tesi in
Ingegneria dei Sistemi Software Adattativi Complessi

Relatore

Prof. Mirko Viroli

Presentata da

Federico Torsello

Anno Accademico **2016/2017**
Sessione III

Parole chiave

Indoor Positioning System
Sensor Fusion
IndoorAtlas
Geomagnetismo
Android

Alla mia famiglia

"Misura ciò che è misurabile,
e rendi misurabile ciò che non lo è"
Galileo Galilei

Indice

Elenco delle figure

Sommario

L'oggetto di questa tesi è la realizzazione di un IPS (*Sistema di Posizionamento Indoor*) che permetta la localizzazione degli utenti all'interno di un ambiente chiuso, ovvero laddove la copertura GPS è inutilizzabile.

Il sistema proposto crea un contesto applicativo che amplia le potenzialità di geolocalizzazione offerte da IndoorAtlass determinando e visualizzando la posizione dell'utente in un'area delimitata, come ad esempio un edificio. Le tecnologie utilizzate per la geolocalizzazione indoor sono state il *fingerprint magnetico* in combinazione alla *sensor fusion*, scelte in quanto permettono all'utente di localizzarsi con un'accuratezza di circa 2-3 metri senza che sia necessario l'ausilio di hardware dedicato.

L'IPS progettato si compone di una parte *server* e di una parte *client*. Il lato server comprende la piattaforma cloud IndoorAtlas e il database dei POI (*Punti Di Interesse*) di ogni edificio mappato. In particolare il servizio IndoorAtlas determina ed invia le coordinate (in formato GPS) della posizione occupata dallo smartphone dell'utente e l'immagine della planimetria del piano in cui esso si trova.

La parte client riguarda l'app per smartphone Android che si interfaccia con IndoorAtlas ricevendo la planimetria del piano e gli aggiornamenti continui sulle coordinate dell'utente. Le coordinate ricevute sono convertite in cartesiane al fine di visualizzare la posizione dell'utente nella planimetria dell'edificio in relazione agli spostamenti dell'utente. Oltre a questo l'app permette di selezionare da una lista un POI target, ottenendo a video il cammino minimo costantemente aggiornato utente-POI. Tutto questo ha l'obiettivo di guidare l'utente nell'ambiente considerato evitando gli ostacoli e quindi far risparmiare tempo.

Introduzione

Localizzare[1] cose o persone è di vitale importanza nel mondo attuale. La localizzazione è utile soprattutto se viene eseguita con una precisione di 1-2 metri; in questo senso sfruttare la presenza capillare degli smartphone e i sensori integrati in essi sembra la strada giusta per ottenere una maggiore accuratezza.

La tecnologia di riferimento per la localizzare è il GPS[2]. Tale sistema permette ai dispositivi dedicati e agli smartphone che integrano il ricevitore GPS di determinare la propria posizione sfruttando una rete di satelliti artificiali in orbita intorno alla Terra. A causa di limiti tecnologici noti, questo tipo di localizzazione funziona esclusivamente in ambienti aperti. L'impossibilità di avere informazioni relative alla posizione in contesti indoor ha portato a cercare di ovviare a questo limite, progettando e sviluppando diversi sistemi di localizzazione indoor alternativi in modo da realizzare quello che comunemente viene indicato come *GPS indoor*.

Gli **IPS**[3] (cap. 4) sono dei sistemi che permettono la **localizzazione degli utenti all'interno di un ambiente chiuso**. Il loro obiettivo principale è *geolocalizzare* con precisione (che può variare da 2 a 10 metri) gli utenti all'interno di ambienti chiusi, permettendo la navigazione in edifici (anche sconosciuti), sfruttando dei dispositivi quali sensori e/o emettitori di segnale in combinazione ad app mobile. Data la modularità di cui godono questi sistemi, nulla impedisce di estenderne le funzionalità, integrando anche la localizzazione GPS come un ulteriore fonte di informazioni in modo da realizzare un *sistema unico di localizzazione*.

Il sistema di IndoorAtlas è uno dei sistemi di geolocalizzazione indoor più completi presenti nel mercato. La tecnica di posizionamento che distingue questa soluzione dalle altre è la definizione e il riconoscimento della ***fingerprint*** (impronta digitale) magnetica dell'ambiente, o meglio dell'edificio in cui il sistema deve essere applicato. Tale fingerprint dipende dall'analisi delle alterazioni del campo magnetico terrestre che risultano essere uniche e distinguibili. La precisione di posizionamento con questa tecnica è notevole (circa 2-3 metri nel caso migliore), nonostante non sia impiegato hardware aggiuntivo. L'accuratezza di questa tecnica può essere ulteriormente aumentata se combinata alla tecnica di *sensor fusion* (sez. 3.4).

[1] **Localizzare**: individuare il punto in cui si trova qualcosa, delimitare un'area
[2] **Global Positioning System**: sistema di posizionamento globale
[3] **Indoor Positioning System**: sistema di posizionamento per ambienti chiusi

Oltre al posizionamento indoor, i sistemi IPS possono fornire all'utente **informazioni contestuali** dell'ambiente in cui si trova (anche se sconosciuto) e guidarlo durante la navigazione, emulando il sistema GPS. Sono diversi gli ambiti industriali [1] in cui queste informazioni possono essere sfruttate, ad esempio per accrescere la soddisfazione del cliente nell'ottenere *servizi ad-hoc* o il *marketing tecnologico* basato sulla *proximity*[4]. La possibilità di indicare con precisione il posizionamento degli utenti e dei prodotti in ambiente indoor inoltre apre ad una vasta gamma di possibilità di marketing che possono aumentare la *customer experience*.

Secondo OpusResearch [1] grazie agli IPS in un futuro prossimo i consumatori potranno cercare la posizione dei prodotti e dei servizi presenti all'interno del centro commerciale in cui si trovano semplicemente utilizzando motori di ricerca in app specifiche, direttamente dallo smartphone. Se questo si avverasse, la ricerca da smartphone potrebbe diventare lo *strumento commerciale in-store primario*.

In questa tesi si descriverà la progettazione di un Indoor Positioning System basato su geomagnetismo e *sensor fusion*. Tale progetto si basa sull'IPS IndoorAtlas[5] in combinazione ad una app Android. L'obiettivo del progetto è creare un servizio che permetta di geolocalizzare un utente all'interno di un edificio in modo efficiente ed economico, cioè utilizzando soltanto i sensori integrati nello smartphone e il magnetismo della Terra per il riconoscimento della fingerprint e la sensor fusion, risparmiando il tempo di setup ed il costo di hardware dedicato al posizionamento, senza alterare l'ambiente in cui l'IPS deve essere applicato.

Oltre a questo il sistema proposto offrirà all'utente la possibilità di selezionare dei POI[6] *(punti di interesse)* e visualizzare il cammino minimo[7] per raggiungerli, aggiornando il percorso in base ai movimenti dell'utente.

Il progetto proposto è stato sviluppato interamente nell'azienda **GetConnected** durante un tirocinio di circa due mesi. Nel periodo passato in questa azienda sono state svolte varie interviste per determinare i requisiti e le finalità industriali del progetto, oltre che per comprendere quanto un sistema simile potrebbe avere un interesse commerciale.

La problematica principale affrontata nella fase di progettazione è stata la necessità di creare un **contesto applicativo**, infatti il sistema IndoorAtlas risolve la questione legata alla geolocalizzazione, ma non fornisce un interfacciamento utente. Per creare un contesto grafico relativamente familiare all'utente che visualizzi un cerchio colorato come riferimento della posizione geolocalizzata, si è dovuta creare una **matrice di punti percorribili**. Questa matrice assume un duplice utilizzo: permettere di posizionare l'utente all'interno della planimetria ricevuta e consentire di disegnare il percorso minimo

[4]**Proximity marketing**: il marketing di prossimità è una tecnica di marketing che opera su un'area geografica delimitata e precisa attraverso tecnologie di comunicazione di tipo visuale e mobile con lo scopo di promuovere la vendita di prodotti e servizi

[5]**IndoorAtlas**: piattaforma cloud scalabile integrata con la tecnologia geomagnetica per realizzare IPS

[6]**Point Of Interest**: punti di interesse ritenuti utili dall'utente

[7]**Cammino minimo**: nella teoria dei grafi lo *shortest path* tra due nodi del grafo è quel percorso che collega tali nodi minimizzando la somma dei costi associati all'attraversamento di ciascun lato

utente-POI grazie all'algoritmo di ricerca A* (sez. 6.2) tenendo conto dei vincoli/ostacoli presenti.

La definizione della matrice dei punti percorribili è stata calcolata in modo automatico in base all'immagine della planimetria del piano in cui l'utente si muove mediante l'utilizzo di filtri immagine. Questi prendono in input la planimetria dell'ambiente di cui si ha la fingerprint magnetica[8] [1], analizzano ogni singolo pixel e restituiscono un'immagine "piatta" (in bianco e nero). L'immagine filtrata viene poi rielaborata in una matrice boolena, assegnando valori di verità in base al colore del pixel analizzato.

Il sistema proposto è stato progettato in base alle necessità industriali che un IPS del genere deve risolvere per essere realmente utilizzato, tale approccio si deve ai continui feedback avuti grazie alla collaborazione con GetConnected durante il tirocinio presso la loro sede. Il sistema è stato testato immedesimandosi nell'utente finale, osservando la reattività e l'accuratezza del posizionamento ottenuto. I primi test sono stati svolti nella sede di Borgo Panigale di GetConnected. In un secondo frangente sono stati svolti dei test in un ambiente più vasto, cioè un supermercato a più piani. In entrambi i contesti si è verificato che la localizzazione indoor ha rispettato tutti i requisiti esposti, consentendo di geolocalizzarsi e raggiungere il POI target.

La tesi è stata suddivisa in tre parti. Nella prima parte (capitoli 1,2,3,4,5) si descriveranno le principali tecniche di posizionamento per smartphone in modo da creare un background sulle possibilità adottabili per l'indoor positioning.

Nella seconda parte, che comprende il capitolo 6, si descriverà come si può calcolare un percorso minimo utilizzando l'algoritmo A*. Questa parte è stata particolarmente utile per il progetto, soprattutto per far visualizzare all'utente la strada che col minor numero di passi fa raggiungere il POI target.

L'ultima parte (capitoli 7,8,9,10,11,12) riguarda la progettazione del sistema. In questa parte si analizzano i requisiti e i problemi che un sistema IPS deve risolvere. Si descriveranno le soluzioni trovate prima da un punto di vista generale e poi implementativo.

Di seguito una breve descrizione dei capitoli che compongono questa tesi:

Capitolo 1: in questo capitolo si descriverà il GPS in quanto tecnologia di riferimento per la localizzazione terrestre, analizzando i pregi e i difetti.

Capitolo 2: nel secondo capitolo si parlerà di come si può stimare la distanza che intercorre tra dispositivi wireless descrivendo le tecniche e i metodi matematici applicabili.

Capitolo 3: nel terzo capitolo si approfondiranno le tecniche di posizionamento indoor che permettono di localizzare in uno spazio delimitato un dispositivo wireless o uno smartphone.

[8] **Fingerprint magnetica**: impronta digitale magnetica univoca generata dall'analisi delle alterazioni dei campi magnetici degli edifici al fine di distinguerli

Capitolo 4: in questo capitolo si parlerà degli IPS e della loro utilità in contesti in cui il GPS non può funzionare. Si descriverà come si possono classificare e in che modo è possibile valutarne le performance. In questo capitolo sono inoltre elencati alcuni esempi di possibili utilizzi reali per sistemi di localizzazione indoor con fini commerciali. Nella parte finale sono descritte le tipologie di IPS in base alle tecnologie di cui è composto.

Capitolo 5: in questo capitolo si descriverà la soluzione di IndoorAtlas per la realizzazione di un IPS. Nella prima parte si descriveranno le caratteristiche che contraddistinguono questo sistema, le tecnologie uniche di cui si avvale per consentire il posizionamento (analisi geomagnetica, fingerprint magnetico, sensor fusion). Nella parte finale si descriveranno i vantaggi e le limitazioni di questo sistema.

Capitolo 6: in questo capitolo si descriverà cosa si intente per percorso minimo dal punto di vista formale e i passi per calcolarlo. In particolare si parlerà dell'algoritmo A* in quanto è stato quello scelto per la realizzazione del progetto.

Capitolo 7: in questo capitolo si inizia ad entrare nel vivo della progettazione del sistema, definendo la *vision* e i goal che si vogliono raggiungere, descrivendo i requisiti di sistema senza cui il sistema non può funzionare. Infine si analizzeranno i principali casi d'uso e gli scenari immedesimandosi nell'utente che utilizzerà realmente il sistema per geolocalizzarsi e/o raggiungere un POI.

Capitolo 8: in questo capitolo si interpreteranno i requisiti del sistema, descrivendo delle soluzioni architetturali di alto livello sotto forma di diagramma.

Capitolo 9: in questo capitolo si sposta il focus da una visione più teorica ad una più legata all'effettiva realizzazione del progetto descrivendo le classi e i metodi implementati. Per rendere le spiegazioni più chiare in alcuni punti sono stati inseriti dei frammenti di codice Java.

Capitolo 10: in questo capitolo si descrive la comunicazione tra app e server, descrivendo il protocollo e le librerie impiegate.

Capitolo 11: in questo capitolo si descrivono i test e le valutazioni delle performance del sistema proposto.

Capitolo 12: nel capitolo finale si ripercorrono brevemente i passi che hanno permesso di realizzare il sistema, traendo conclusioni sul lavoro svolto.

Parte I

Background: Posizionamento mediante smartphone

Capitolo 1

Global Positioning System

1.1 Posizionamento utilizzando il GPS

Il GPS è la tecnologia di riferimento per la geolocalizzazione di cose o persone in qualunque punto del suolo terrestre. È utilizzabile esclusivamente negli **ambienti aperti** (*outoor*). Anche in caso di condizioni atmosferiche avverse restituisce la posizione con un'ottima **precisione**.

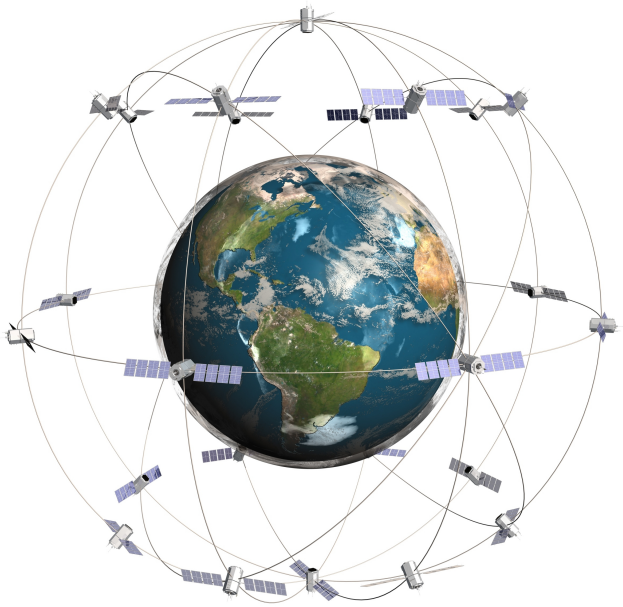


Figura 1.1: GPS
tratta da expeditionportal.com

La localizzazione attraverso questa tecnologia è possibile solo se il dispositivo usato è

in grado di stabilire un contatto con almeno quattro satelliti operativi. I satelliti GPS inviano al ricevitore informazioni relative alla distanza satellite-ricevitore e sul tempo impiegato dal segnale per essere ricevuto.

I livelli di precisione ottenibili dal GPS variano in base:

- al ricevitore utilizzato;
- alla modalità di misura;
- ai tempi di stazionamento (tempo in cui il ricevitore resta nella stessa posizione);
- al numero e alla disposizione dei satelliti al momento della ricezione dei segnali.

Il sistema GPS è suddivisibile in tre componenti:

1. **Segmento spaziale**: costituito dai satelliti responsabili dell'invio dei segnali radio a terra;
2. **Segmento di controllo**: composto da 5 stazioni a terra (Colorado Springs, Hawaii, Kwajalein, Ascension e Diego Garcia), che devono:
   - mantenere sincronizzati gli orologi dei satelliti orbitanti;
   - conoscere la posizione esatta di ognuno di loro, eventualmente correggerla.
3. **Segmento utente**, costituito da:
   (a) un'antenna capace di agganciare i segnali;
   (b) un ricevitore in grado di decodificarli ed elaborarli;
   (c) un software per il trattamento dei dati ricevuti.

1.1.1 Principio di funzionamento del GPS

Il principio di funzionamento del GPS si basa sul presupposto che i satelliti si muovono rispetto al sistema cartesiano geocentrico di riferimento, quindi le loro coordinate sono note in ogni momento. Il calcolo della posizione del ricevitore si basa sul tempo trascorso tra l'emissione del segnale da parte del satellite e la ricezione dello stesso da parte del ricevitore.

Sulla base di queste informazioni è possibile calcolare:

- la distanza satellite-ricevitore;
- le coordinate del satellite per mezzo della trilaterazione.

Per poter determinare il tempo impiegato dal segnale per raggiungere il ricevitore è necessario che i satelliti e il ricevitore siano sincronizzati. I satelliti integrano orologi atomici ad altissima precisione sincronizzati dalle stazioni di controllo, mentre i ricevitori hanno orologi decisamente meno precisi i quali ad ogni nuova accensione utilizzano l'informazione proveniente dal quarto satellite per sincronizzarsi. Il tempo necessario alla sincronizzazione dei ricevitori è variabile e dipende dalla qualità del segnale ricevuto.

1.1.2 A-GPS

Il sistema **A-GPS**[1] consente di abbattere i tempi necessari alla prima localizzazione degli smartphone che integrano il GPS, sfruttando il sistema di localizzazione LBS[2] basato su telefonia cellulare.

Mostra la sua utilità nelle zone metropolitane dove è più complesso acquisire il segnale satellitare e decodificarne il messaggio di navigazione contenente le *effemeridi* (tabelle di grandezze astronomiche) e l'*almanacco* (parametri orbitali approssimati dell'intera costellazione dei satelliti).

1.1.3 Limiti del GPS

Il limite più grande della tecnologia GPS riguarda l'**impossibilità di funzionare adeguatamente in ambienti chiusi**. Questo è dovuto agli effetti negativi della *multi-path propagation*[3] di cui è vittima il segnale satellitare che attraversa le pareti degli edifici. La presenza di errori incontrollabili porta alla perdita di consistenza, rendendo il sistema satellitare inutilizzabile ai dispositivi presenti in **ambienti chiusi** (*indoor*).

Da questo limite nasce la **necessità** di creare un *sistema per la localizzazione indoor*.

[1] **GPS assistito**: sistema che permette di ridurre i tempi di calcolo della posizione GPS

[2] **Servizio basato sulla localizzazione**: servizio che sfrutta le informazioni relative alla posizione geografica dello smartphone per diversi scopi come ad esempio social networking, intrattenimento o sicurezza personale

[3] **Propagazione multi-percorso**: nelle telecomunicazioni wireless, il fenomeno di propagazione multipath si deve ai segnali radio che raggiungono l'antenna ricevente attraverso due o più percorsi. Le cause di multipath includono la rifrazione e riflessione dei corpi idrici e degli oggetti terrestri come le montagne e gli edifici

Capitolo 2

Stimare la distanza di dispositivi wireless

2.1 Metodi per stimare la distanza

Stimare la distanza tra dispositivi wireless è utile perché attraverso questa informazione è possibile determinarne (con un certo errore) la posizione rispetto ad un sistema di riferimento. A tal proposito esistono diverse tecniche (sez. 3) che in base al segnale ricevuto permettono di stimare la distanza emettitore-ricevitore [4]. I metodi che permettono di stimare la distanza emettitore-ricevitore [4, 5]si distinguono in:

- *Range based:*
  - **RSSI** - potenza del segnale radio ricevuto (sez. 2.2.1);
  - **ToA** - tempo d'arrivo: (sez. 2.2.2)
    - *One-way*;
    - *Two-way*;
  - **TDoA** - differenze del tempo di arrivo (sez. 2.2.3);
- *Angle based:*
  - Triangolazione (sez. 2.3.1).

2.1.1 Anchor e Unknown

Per poter determinare le distanze si devono distinguere i **punti di riferimento** (che hanno delle coordinate note) dai **nodi senza posizione nota** a cui assegnare delle coordinate.

Si dicono:

- **Anchor**: i nodi le cui coordinate sono note
- **Unknown**: i nodi di cui non si conosce la posizione.

Ovviamente si può considerare come Unknown un utente che utilizza il proprio smartphone per ottenere informazioni sulla propria posizione. L'**obiettivo del posizionamento** è assegnare le giuste coordinate agli Unknown rispetto ad un sistema di riferimento, come ad esempio quello cartesiano.

2.2 Stime della distanza Range based

Nel posizionamento dei nodi basato sulla distanza la stima della posizione dello Unknown dipende dai seguenti parametri:

- il **tempo trascorso** tra l'emissione e la ricezione del segnale radio;
- la **distanza euclidea** tra ogni emettitore ed il ricevitore;
- la **potenza del segnale** ricevuto.

In alcuni casi sono necessari tre o più Anchor per ottenere le coordinate da assegnare allo Unknown.

2.2.1 Received Signal Strength Indicator - RSSI

La comunicazione tra dispositivi wireless (senza fili) avviene tramite lo scambio di segnali propagati nell'aria. Durante la propagazione i segnali tendono ad attenuarsi in base alla distanza percorsa fino a non essere più percepibili.

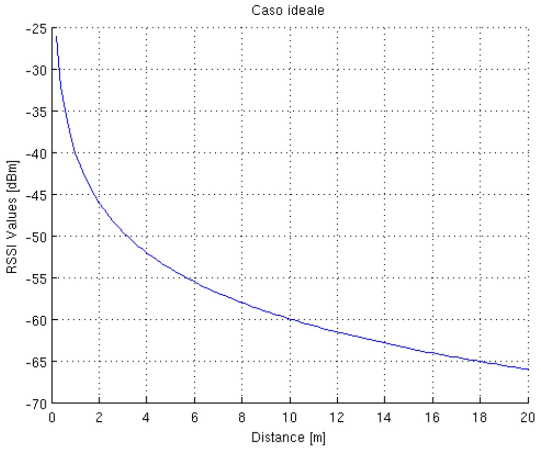


Figura 2.1: RSSI - Andamento della potenza in funzione della distanza
(con $n = 2$ e $A = -40dBm$)

La stima della **potenza del segnale ricevuto** è data dall'indicatore **RSSI**[1] [9].

[1]Received Signal Streght Indicator: nelle telecomunicazioni è la potenza di un segnale radio ricevuto espressa in Watt o dBm

Attraverso RSSI lo Unknown può stimare la distanza emettitore-ricevitore, anche se questo approccio non è particolarmente preciso in quanto soggetto a molti errori di rilevazione.

Equazione di trasmissione di Friis

La distanza emettitore-ricevitore si stima utilizzando l'**equazione di trasmissione di Friis**.

Questa equazione calcola il rapporto tra la potenza del segnale ricevuto e la potenza del segnale trasmesso in condizioni ideali:

$$P_R = P_T \frac{G_T G_R \lambda^2}{(4\pi)^2 \mathbf{d}^n} \tag{2.1}$$

dove:

- P_R : potenza del segnale ricevuto (espressa in Watt);
- P_T : potenza del segnale trasmesso (espressa in Watt);
- G_R : guadagno dell'antenna ricevente;
- G_T : guadagno dell'antenna trasmittente;
- $\lambda = \frac{v}{f}$: lunghezza d'onda, in cui v è la velocità di propagazione e f è la frequenza dell'onda;
- $\mathbf{d}$: distanza espressa in metri;
- n : constante di propagazione del segnale che dipende dall'ambiente.

Stima della distanza con RSSI

La seguente formula permette di convertire la potenza espressa in Watt alla potenza espressa in dBm:

$$P[dBm] = 10 \log_{10}(10^3 P[W]) \tag{2.2}$$

Combinando l'equazione (2.1) con (2.2) e applicando le proprietà dei logaritmi si ottiene:

$$RSSI = -(10\, n \log_{10} \mathbf{d} - A) \tag{2.3}$$

dove A è la **potenza del segnale ricevuto** a distanza fissa di un metro (espressa in dBm), considerando una costante di propagazione n.

La stima della distanza si ha dalla seguente equazione:

$$\mathbf{d} = 10\left(\frac{A - RSSI}{10\, n}\right) \tag{2.4}$$

Con questa formula si può stimare la distanza emettitore-ricevitore conoscendo il valore RSSI e i parametri A ed n.

Problemi della stima con RSSI

- **Riflessione:** il segnale propagandosi sbatte e si riflette su vari ostacoli, seguendo più percorsi, subendo gli effetti del *multi-path fading*.
- **Assorbimento:** alterazione del normale decadimento dell'intensità del segnale che si avrebbe nello spazio libero da ostacoli.

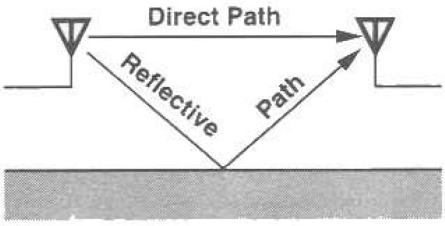


Figura 2.2: RSSI - Problemi di riflessione e assorbimento

2.2.2 Time of Arrival - ToA

Con il metodo **ToA** (tempo di arrivo) si stima la distanza emettitore-ricevitore in relazione al tempo che occorre al segnale emesso per essere ricevuto e alla velocità con cui viaggia nell'aria.

Esistono due modalità per la stima della distanza con ToA:

- *One-way ToA*
- *Two-Way ToA*.

One-way ToA

1. l'emettitore A trasmette il segnale al tempo t_1
2. Il segnale arriva a B al tempo t_2
3. T_f è la differenza degli istanti di invio da A e ricezione in B dovuta alla distanza.

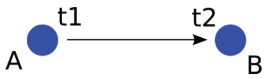


Figura 2.3: Time of Arrival - One-way

Two-way ToA

1. Il trasmettitore A invia il segnale al tempo t_1
2. Il segnale arriva a B al tempo t_2
3. B elabora il messaggio in un tempo T_d e risponde ad A.

$$T_f = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \tag{2.5}$$

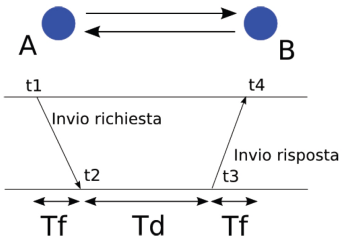


Figura 2.4: Time of Arrival - Two-way

Stima della distanza con ToA

Se del segnale considerato sono dati noti:

- la **velocità di propagazione**[2] c
- il **time of flight**[3] $T_f = \frac{\mathbf{d}_i}{c}$

allora lo spazio percorso, cioè la **distanza** $\mathbf{d}_i$ stimata dal ricevitore i-esimo, sarà:

$$\mathbf{d}_i = c \cdot T_f \tag{2.6}$$

Una volta stimate le distanze da parte di tutti i ricevitori si potranno calcolare le circonferenze[4] C_i aventi:

- per centro le coordinate dell'emettitore;
- per raggio[5] la distanza d_i per ogni i-esimo emettitore.

Si avrà:

$$\sqrt{(x_i - x)^2 - (y_i - y)^2} \tag{2.7}$$

dove:

- (x, y) è la posizione del ricevitore
- (x_i, y_i) è la posizione dell'emettitore i-esimo

Nel caso di 3 Anchor si ottiene il seguente sistema:

$$\begin{cases} \sqrt{(x_1 - x)^2 - (y_1 - y)^2} \\ \sqrt{(x_2 - x)^2 - (y_2 - y)^2} \\ \sqrt{(x_3 - x)^2 - (y_3 - y)^2} \end{cases} \tag{2.8}$$

Nel caso ideale le circonferenze descritte si intersecano in un unico punto che determina la *posizione esatta* del ricevitore. Nella realtà trovare un unico punto è impossibile a causa degli errori di rilevazione[2], quindi si tende a restringere la ricerca in una **zona di intersezione** in cui molto probabilmente il ricevitore si dovrebbe trovare.

[2]Velocità di propagazione della luce nel vuoto $c = 299792458$ m/s

[3]Tempo di volo: tempo impiegato da un segnale per arrivare da un emettitore ad un ricevitore

[4]Circonferenza: in geometria è l'insieme dei punti equidistanti da un punto fisso, detto centro del cerchio

[5]Raggio: in geometria è la distanza dei punti della circonferenza dal centro

2.2.3 Time difference of Arrival - TDoA

Anche in **TDoA**, come per ToA, si considera il tempo di volo. In questo caso però la distanza emettitore-ricevitore si calcola in base alla **differenza dei tempi di volo** dei segnali. Anche in questo caso le coordinate degli emettitori devono essere note.

In questo metodo ogni emettitore trasmette due segnali con velocità diverse e i ricevitori stimano la distanza in base:

- alle differenze degli istanti di invio e ricezione;
- alle velocità di propagazione.

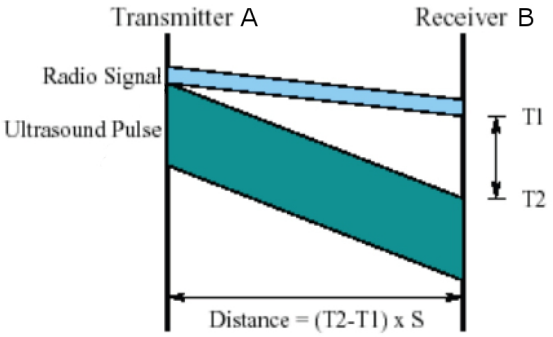


Figura 2.5: Time difference of Arrival - TDoA

2.3 Stime della distanza Based

Le tecniche di posizionamento basate sugli angoli sfruttano le proprietà geometriche e trigonometriche in modo da determinare la *distanza angolare*[6] tra Unknown e gli Anchor.

Viene utilizzata dagli algoritmi di Triangolazione (sez. 2.3.1).

2.3.1 Triangolazione

La **Triangolazione** stima le coordinate dello Unknown utilizzando il **teorema del coseno (o di Carnot)**[3], considerando gli angoli che lo Unknown forma con i tre Anchors.

Questo teorema permette, insieme al **teorema dei seni**, di risolvere un triangolo qualsiasi.

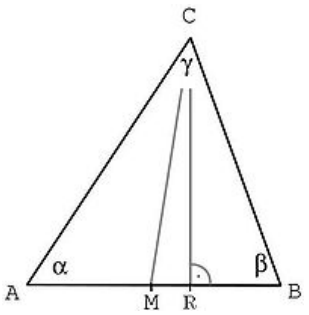


Figura 2.6: Triangolazione

2.3.2 Teorema del coseno

Dato un triangolo, siano a e b la misura di due suoi lati e sia θ l'angolo tra essi compreso, il quadrato del terzo lato è dato da:

$$c^2 = a^2 + b^2 - 2\,ab\cos\theta \tag{2.9}$$

Angolo acuto

Si suppone che siano noti a,b e θ.

Considerando il triangolo ACH, rettangolo in H, si ha:

$$CH = b\cos\theta \tag{2.10}$$

[6] **Distanza angolare**: considerando due punti, è l'angolo da essi sotteso

Se la distanza AB è nota si può facilmente calcolare:

$$AC = \frac{AB \sin \beta}{\sin \theta} \tag{2.11}$$

$$BC = \frac{AB \sin \alpha}{\sin \theta} \tag{2.12}$$

$$RC = BC \sin \alpha = AC \sin \beta \tag{2.13}$$

Da RC si possono ricavare le coordinate della posizione dell'utente.

Applicando il teorema di Pitagora

$$AH^2 = AC^2 - CH^2 = b^2 - b^2 \cos^2 \theta \tag{2.14}$$

Considerando ora il triangolo ABH, rettangolo in H, si ha:

$$BH = BC - CH = a - b \cos \theta \tag{2.15}$$

Applicando di nuovo il teorema di Pitagora

$$\begin{aligned} AB^2 = c^2 = AH^2 + BH^2 = b^2 + b^2 \cos^2 \theta + (a - b \cos \theta)^2 = \\ b^2 - b^2 \cos^2 \theta + a^2 + b^2 \cos^2 \theta - 2ab \cos \theta = \\ a^2 + b^2 - 2ab \cos \theta \end{aligned} \tag{2.16}$$

Angolo ottuso

Si suppone che siano noti a, b e θ (stavolta ottuso).

Considerando il triangolo ACH, rettangolo in H, si osserva che l'angolo (acuto) in C è il supplementare di θ, quindi è pari a $\pi - \theta$ (in radianti).

Per le formule relative agli angoli associati, si ha che:

$$\cos \pi - \theta = - \cos \theta \tag{2.17}$$

Per la misura di CH si ha:

$$CH = -b \cos \theta \tag{2.18}$$

Per il teorema di Pitagora si ottiene (2.14).

Considerando ora il triangolo AHB ed il rettangolo H, si ha l'equazione (2.15).

Applicando ancora il teorema di Pitagora si ottiene di nuovo l'equazione (2.16).

Applicazione del teorema del Coseno

Il teorema del coseno permette, noti due lati e l'angolo tra essi compreso, di determinare il terzo lato.

Estraendo la radice quadrata di entrambi i membri dell'espressione ricavata, si ha:

$$c = \sqrt{a^2 + b^2 - 2ab\cos\theta} \tag{2.19}$$

Le coordinate di un utente sono calcolate dal seguente sistema:

$$\begin{cases} y_p = x_p \tan\theta_1 + (y_1 - x_1 \tan\theta_1) \\ y_p = x_p \tan\theta_2 + (y_2 - x_2 \tan\theta_2) \end{cases} \tag{2.20}$$

dove

- x_p e y_p: sono le coordinate del punto di cui non si conosce la posizione;
- x_1 e y_1: sono le coordinate del primo punto;
- x_2 e y_2: sono le coordinate del secondo punto;
- θ_1: è l'azimut del primo punto verso p;
- θ_2: è l'azimut del secondo punto verso il punto P.

Capitolo 3

Tecniche di posizionamento indoor

Per **posizionamento** si intende l'azione con cui si dispone qualcosa nella giusta posizione, cioè collocare un oggetto in uno spazio secondo un determinato principio rispetto a dei punti di riferimento.

In questo capitolo sono elencate le principali tecniche di posizionamento indoor che permettono di *stimare la posizione dei dispositivi wireless* negli ambienti chiusi come ad esempi gli edifici domestici. Queste tecniche sfruttano i metodi di stima delle distanze dei dispositivi wireless (sez. 2.1) in modo da determinare la posizione assunta da un Unknown in un sistema di riferimento composto da Anchor.

Le tecniche di posizionamento indoor si distinguono in:

- basate sulla distanza stimata:
  - **MIN-MAX** (sez. 3.1.1);
  - **Trilaterazione** (sez. 3.1.2);
  - **MLAT** - multilaterazione (sez. 3.1.3);
  - **Prossimità** (sez. 3.1.4).
- basate sull'angolo stimato:
  - **AoA** - angolo di arrivo (sez. 3.2.1);
  - **PiT** - punto nella triangolazione (sez. 3.2.2);
  - **APiT** - punto approssimato nella triangolazione (sez. 3.2.3).
- basate su fingerprinting:
  - posizionamento magnetico (sez. 3.3.1).
- basate su sensor fusion (sez. 3.4).

3.1 Posizionamento basato sulla distanza stimata

3.1.1 MIN-MAX

Il metodo **MIN-MAX** effettua la stima della distanza emettitore-ricevitore utilizzando il parametro RSSI (sez. 2.2.1). I risultati ottenuti non sono accuratissimi ma si avvicinano a quelli ottenibili della Multilaterazione (sez. 3.1.3).

Per stimare la distanza con MIN-MAX si deve seguire il seguente algoritmo:

1. Stimare la distanza $\mathbf{d}_i$ di ogni nodo i-esimo in base al valore RSSI;
2. Disegnare due linee orizzontali e verticali a distanza $\mathbf{d}_i$ dallo Unknown
3. Disegnare un **quadrato** di lato $2\mathbf{d}_i$ i cui estremi saranno:

$$[\max(x_i - \mathbf{d}_i), \max(y_i - \mathbf{d}_i)] \times [\min(x_i + \mathbf{d}_i), \min(y_i + \mathbf{d}_i))] \tag{3.1}$$

4. Calcolare l'intersezione delle linee del quadrato.

Il centro del quadrato rappresenta la posizione stimata dello Unknown. Più piccola sarà l'area e maggiore sarà l'accuratezza della posizione stimata.

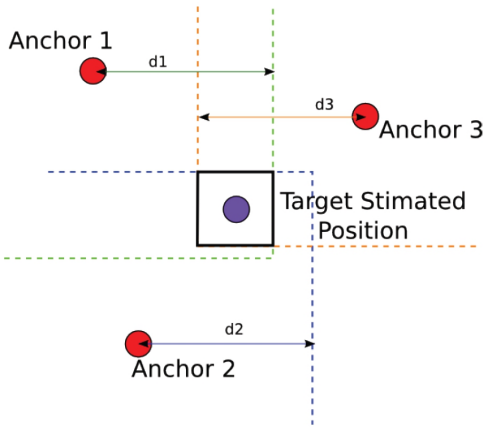


Figura 3.1: MIN-MAX

3.1.2 Trilaterazione

La **Trilaterazione** è un metodo di stima delle distanze più complesso ma anche più efficiente rispetto a MIN-MAX (sez. 3.1.1). Tale metodo stima la distanza dello Unknown dagli Anchor sfruttando le **proprietà dei triangoli**.

Per poter stimare la distanza emettitori-ricevente si considerano 3 Anchor intorno cui disegnare 3 circonferenze aventi:

- per centro le coordinate degli Anchor (le quali sono note ed immutabili);
- per raggio l'RSSI del segnale ricevuto dallo Unknown.

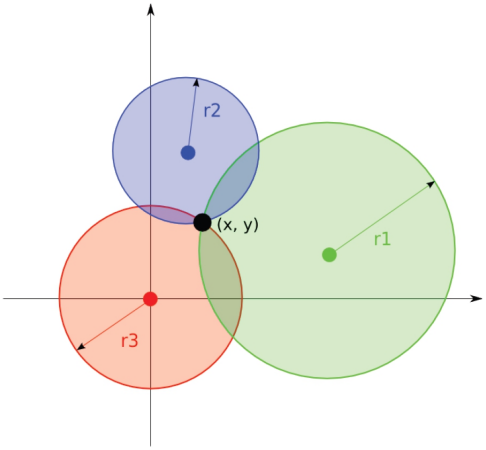


Figura 3.2: Trilaterazione

Per stimare la distanza con la Trilaterazione si deve considerare l'equazione della circonferenza (x_i, y_i) con raggio r_i

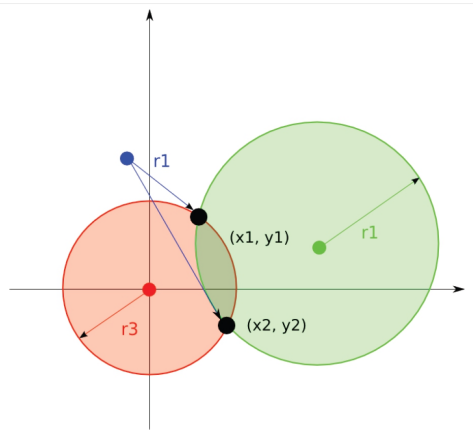
$$(x - x_i)^2 + (y - y_i)^2 = r_i^2 \tag{3.2}$$

in cui la posizione dello Unknown è stimata dal calcolo dell'intersezione delle tre circonferenze:

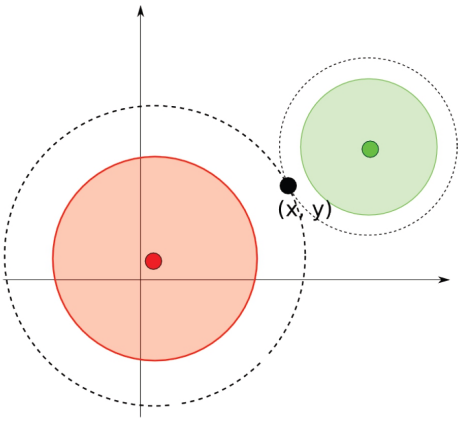
$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 = r_1^2 \\ (x - x_2)^2 + (y - y_2)^2 = r_2^2 \\ (x - x_3)^2 + (y - y_3)^2 = r_3^2 \end{cases} \tag{3.3}$$

In base ai punti di intersezione:

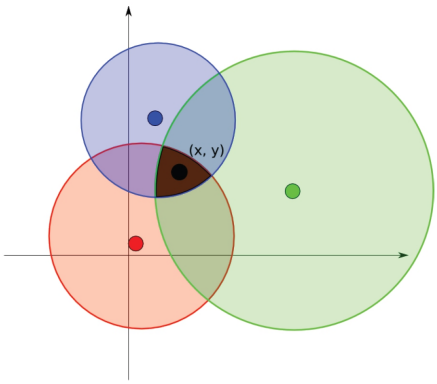
- se si hanno tre cerchi che si sovrappongono, allora il punto di intersezione non è unico.
- se non c'è nessun punto di intersezione, si devono aumentare i raggi di tutte le circonferenze in modo proporzionale fino ad ottenere una intersezione.
- se si hanno solo due cerchi e c'è una sola intersezione, il punto è unico.
- se esistono due punti di intersezione, si deve prendere quello a distanza minore dal terzo emettitore.



(a) Due punti di intersezione



(b) Nessun punto di intersezione



(c) Punto di intersezione non unico

Figura 3.3: Trilaterazione - Altri casi

3.1.3 Multilaterazione - MLAT

La **Multilaterazione** [4] è un caso generale di Trilaterazione (sez. 3.1.2) con cui si riducono gli effetti degli errori della stima della distanza.

Invece di ricevere informazioni esclusivamente da solo tre emettitori, si considerano le distanze rispetto a tutti gli n Anchor. Per poter impiegare questo metodo, lo Unknown deve essere raggiungibile da almeno 3 Anchor.

Il sistema di equazioni con più Anchor è:

$$\begin{cases} (x-x_1)^2 + (y-y_1)^2 = r_1^2 \\ (x-x_2)^2 + (y-y_2)^2 = r_2^2 \\ \vdots \\ (x-x_n)^2 + (y-y_n)^2 = r_n^2 \end{cases} \tag{3.4}$$

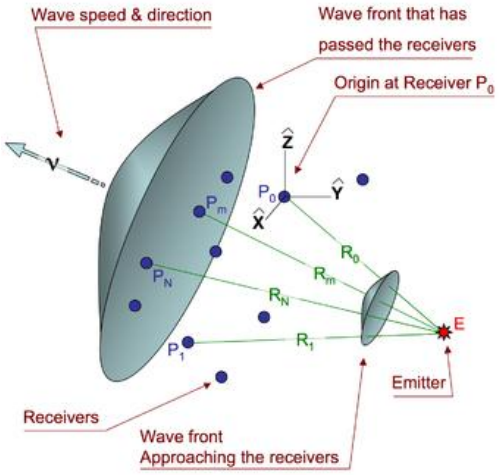


Figura 3.4: Multilaterazione

Multilaterazione iterativa

La **Multilaterazione iterativa** è una tecnica che permette di determinare la posizione di più ricevitori in modo iterativo secondo questo algoritmo:

- il ricevitore determina la propria posizione se nel proprio raggio di copertura ha almeno tre Anchor;
- una volta che ha determinato le proprie coordinate, lo Unknown diventa a sua volta un Anchor;

- a questo punto gli Unknown possono determinare la propria posizione in base ai nuovi e vecchi Anchor.

Il maggiore svantaggio della Multilaterazione iterativa è la propagazione dell'errore causata dalla trasformazione degli Unknown in Anchor.

Ogni volta che ad un Unknown sono assegnate delle coordinate, viene assegnata anche un'**incertezza** che dipende dal numero di Anchors che hanno partecipato alla localizzazione del nodo. Una volta che lo Unknown diventa Anchor trasmette a tutti i nodi della rete la propria posizione ed incertezza, causando una propagazione dell'errore di posizionamento.

Multilaterazione collaborativa

La **Multilaterazione collaborativa** [4] è una estensione della Multilaterazione iterativa. Viene utilizzata quando un Unknown è nel raggio di copertura di uno o due Anchor, in reti in cui la densità di Anchors è minima.

Passi di calcolo:

- Formazione di un **albero di collaborazione**
- Stima della posizione iniziale dello Unknown
- Ottimizzazione del risultato per mezzo di algoritmi come il **Filtro di Kalman**

Filtro di Kalman [10] [12] [11]

Definizione 1 Il filtro di Kalman prende il suo nome da *Rudolf Kalman*. La prima realizzazione pratica del filtro è stata sviluppata da Stanley Schmidt che applicò il costrutto matematico di Kalman ad un sistema di stime di traiettorie al fine di eliminare alcuni disturbi.

Il filtro di Kalman permette di stimare lo stato di un sistema dinamico lineare perturbato da rumore basandosi sulle misure (o osservazioni) linearmente dipendenti.

Lo stimatore è ottimo rispetto a qualunque funzione quadratica dell'errore di stima in quanto si basa su tutte le informazioni disponibili sul sistema considerato.

Definizione 2 Il filtro di Kalman è una tecnica per la risoluzione del problema Gaussiano lineare quadratico, che permette di tradurre un sistema dinamico nella stima dello stato istantaneo a partire dalla sua uscita. Si dice statisticamente ottimale rispetto a qualunque funzione quadratica di stima dell'errore.

Supponendo di conoscere le variabili misurate in funzione delle variabili di interesse, questo filtro si occupa della soluzione del problema inverso, invertendo questa relazione funzionale, stimando le variabili indipendenti come funzione inversa delle variabili dipendenti (misurabili).

Dato il sistema dinamico:

$$\begin{cases} dX = f(x(t), u(t), t) \\ y = h(x(t), u(t), t) \end{cases} \tag{3.5}$$

Il filtro di Kalman risale allo stato $x(t)$ partendo dall'uscita $y(t)$ perturbata da rumore gaussiano. La costruzione del filtraggio dei dati avviene sulla base di una media pesata tra il prossimo valore predetto e il prossimo valore stimato. L'algoritmo è composto da alcune equazioni matematiche che effettuano operazioni ricorsive in grado di dare una soluzione efficace al metodo dei minimi quadrati.

Per la sua costruzione il filtro di Kalman necessita di **tre ingredienti fondamentali** legati in maniera vincolante al sistema che si sta studiando:

1. una serie di misure sul sistema da stimare
2. un modello matematico descrittivo del sistema
3. la conoscenza del modello statistico del sistema

3.1.4 Prossimità

La **Prossimità** [5] determina la posizione dello Unknown come presenza/assenza di Anchor nell'area di rilevamento.

Con la prossimità:

- se uno Unknown si trova nel raggio di copertura di un Anchor, allora allo Unknown si assegnano le coordinate dell'Anchor.
- se all'interno del raggio di copertura dello Unknown vi fossero più Anchor, allo Unknown si assegnano le coordinate pari al valore medio delle coordinate degli Anchor.

La prossimità può essere utilizzata come *trigger* in modo da avviare particolari azioni in app per smartphone. Un esempio classico sono i *proximty system* formati da iBeacon.

3.2 Posizionamento basato sull'angolo stimato

3.2.1 Angle of arrival - AoA

Nella **AoA** [6] si utilizzano antenne direzionali con l'obiettivo di stimare l'angolo di arrivo dei segnali u in base ad un sistema angolare.

I problemi relativi a questa tecnica di stima sono:

- la qualità delle antenne;
- gli errori dovuti alle interferenze;
- gli errori dovuti agli ostacoli.

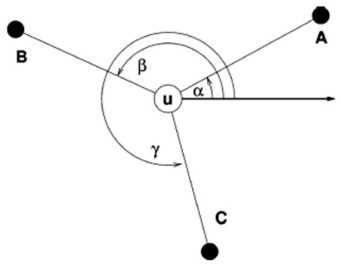


Figura 3.5: AoA - Angle of arrival

3.2.2 Point in Triangulation - PiT

Nella **PiT** allo Unknown sono assegnati tre Anchor tra quelli presenti nel suo raggio di copertura. Durante il movimento lo Unknown verifica se si trova all'interno o all'esterno del triangolo formato dagli Anchor.

- Se durante il movimento lo Unknown si trova ancora all'interno del triangolo e si sposta in qualsiasi direzione, assumerà come sue le coordinate del baricentro del triangolo.
- Se muovendosi si allontana da tutti e tre gli Anchor e si troverà all'esterno del triangolo, allora dovrà scegliere altri tre Anchor più prossimi.

3.2.3 Approximate Point in Triangulation - APiT

La **APiT** si basa sui concetti del PiT, ma in questo caso gli Unknown considerati non sono mobili. Per simulare il loro movimento vengono utilizzate le informazioni RSSI (sez. 2.2.1) degli Anchor vicini.

3.3 Posizionamento basato su riconoscimento del fingerprint

La tecnica del fingerprinting (creazione dell'impronta digitale) permette di identificare in modo univoco un'area geografica.

La sua creazione consiste di due fasi:

1. fase off-line: campionamento dell'ambiente e analisi dei dati raccolti;
2. fase on-line: pattern matching tra i dati rilevati dai sensori e i dati salvati precedentemente.

Durante la prima fase viene redatta una mappa delle misurazioni. Questa mappa dipende dall'analisi delle misure effettuate sul luogo, cioè dalla potenza dei segnale ricevuti all'interno dell'edificio in cui si effettua la fingerprint. I valori misurati vengono inizialmente memorizzati all'interno di un database per poi essere analizzati ed aggregati al fine di essere recuperabili durante la fase online.

Nella fase online si comparano i valori memorizzati nel database con quelli rilevati sempre nello stesso edificio. Questo confronto serve a verificare e stimare la posizione del device.

3.3.1 Posizionamento magnetico

Il **posizionamento magnetico** è stato inventato dal **professor Janne Haverinen** e da Anssi Kemppainen.

Questa tecnica di posizionamento utilizza la tecnica del fingerprinting. Si basa sull'analisi dei campi magnetici della Terra, o meglio sulle **interferenze magnetiche coerenti** causate dalle strutture in acciaio (come quelle degli edifici) a questi campi. In questo senso i disturbi creano una fingerprint magnetica unica e distinguibile che permette di riconoscere ogni singolo edificio e ogni singolo piano di un palazzo.

Nel 2012 il professor Janne Haverinen fonda la società **IndoorAtlas** (sez. 5.1) per commercializzare il posizionamento magnetico.

3.4 Sensor Fusion

La **sensor fusion** è una tecnica che permette di combinare dati sensoriali forniti da fonti diverse al fine di ottenere delle informazioni risultanti migliori in confronto a quanto sarebbe stato possibile ottenere utilizzando le stesse fonti singolarmente. Questo può significare ottenere valori più precisi, più completi e più affidabili.

Nel contesto dell'indoor positioning la sensor fusion permette di **unire ogni fonte di dati legate alla posizione** per ottenere un *posizionamento universale* ancora più preciso e veloce, ottimizzando i tempi di stima e la precisione.

Molti dei dati che possono essere registrati dai sensori integrati negli smartphone sono: le alterazioni dei campi magnetici; i segnali Wi-Fi; i pacchetti Beacon Bluetooth; la pressione atmosferica; le variazioni di posizione mediante accelerometro e/o giroscopio.

Capitolo 4

Indoor Positioning System

4.1 IPS

Un **Indoor Positioning System** (Sistema di Posizionamento[1] per Interni) permette di individuare e monitorare la posizione occupata dai **target** all'interno di uno spazio delimitato. I target localizzabili possono essere utenti o oggetti sia fermi che in movimento.

L'*obiettivo* degli IPS è creare un'esperienza utente simile al GPS in contesti ridotti e delimitati come aeroporti o ospedali, dove il GPS non può funzionare.

Questi sistemi:

- localizzano cose o persone all'interno di un ambiente chiuso delimitato utilizzando i dati dei sensori presenti nei dispositivi come smartphone o tablet;
- raccolgono ed analizzano le varie informazioni sensoriali (come onde radio, campi magnetici, ecc. . .) presenti nell'ambiente;
- sfruttano l'ambiente e le tecnologie presenti per offrire all'utente la possibilità di orientarsi, indicando il **percorso minimo** per raggiungere un obiettivo.

La **precisione di posizionamento** ottenuta dipende da più fattori:

1. le **tecnologie utilizzate**: Wi-Fi, onde radio, Beacons, ecc. . . ;
2. le **attrezzature**: sensori deputati alla raccolta dei dati e dispositivi per la loro elaborazione;
3. le **infrastrutture**: cioè l'insieme dei device disposti con determinati criteri in un'area delimitata al fine di ottenere dati affidabili.

[1] **Posizione**: collocazione di un oggetto nello spazio in relazione a un sistema di riferimento

Nel mercato sono disponibili diversi tipi di IPS commerciali che in base al principio di funzionamento e alle tecniche utilizzate richiedono hardware specifico o la combinazione di più sistemi. Al contrario del GPS per la localizzazione outdoor, tuttora **non esiste uno standard di riferimento** per gli IPS e quindi per la localizzazione indoor.

Gli IPS permettono di creare una vasta gamma di servizi basati sulla localizzazione sfruttabili da app mobile. Alcuni esempi di servizi *location-based* realizzabili sono:

- **Way-Finding**: permettere di navigare in edifici complessi, come ad esempio aeroporti, seguendo il percorso indicato;
- **Ricerca dei POI da un database**: aumentare la *customer experience* facendo trovare all'utente ciò che desidera.
- **Multi-Dot**: visualizzare in una mappa le posizioni degli utenti per creare un grande social network o tracciare i bambini per tenerli al sicuro.
- **Marketing di prossimità**: realizzare marketing altamente mirati *in-app*, inviando annunci sulle ultime offerte, aumentando la soddisfazione e l'affiliazione del cliente.

4.2 Classificazione degli IPS

Gli IPS possono essere classificati in base a diversi fattori:

- **Principi di posizionamento**:
  - ID
  - fingerprint
  - geometrico
- **Tipo di segnale e tecnologia impiegate**:
  - basati su ricezione di segnali radio (Wi-Fi, Bluetooth, infrarossi, RFID[2], ...)
  - magnetismo;
  - ultrasuoni;
  - riconoscimento ottico;
  - ...
- **Metodi di ranging**[3]:
  - RSSI: Received Signal Strength Information (sez. 2.2.1);

[2] **RFID**: identificazione a radio frequenza

[3] **Telemetria**: insieme di metodi e osservazioni aventi lo scopo di fornire la misura della distanza di un oggetto dall'osservatore

  - ToA: Time of Arrival (sez. 2.2.2);
  - TDoA: Time Difference of Arrival; (sez. 2.2.3);
  - AoA: Angle of Arrival (sez. 3.2.1);
  - ...
- **Tecniche di posizionamento**:
  - fingerprinting (sez. 3.3);
  - trilaterazione (sez. 3.1.2);
  - multilaterazione (sez. 3.1.3);
  - prossimità (sez. 3.1.4);
  - triangolazione (sez. 2.3.1).

4.3 Valutazione delle performance

Per poter valutare l'efficienza degli IPS esistono diversi parametri di valutazione:

- **Accuratezza o location error**: è la differenza fra posizione reale e risultato ottenuto.
- **Precisione**: è la caratteristica che garantisce di avere un sistema efficace in grado di operare correttamente nel contesto in cui è inserito. Ad una maggiore precisione corrisponde una più alta probabilità di ottenere risultati soddisfacenti.
- **Complessità**: è un fattore che dipende dall'hardware e dal software utilizzato. Riguarda la difficoltà relativa alle operazioni di installazione e diffusione del sistema e la manutenzione dello stesso.
  - L'analisi della complessità del software si concentra sul costo computazionale, sui tempi di risposta e sull'infrastruttura client-server.
  - Per l'analisi della complessità dell'hardware invece si considerano aspetti come la durata e la qualità dei componenti impiegati.
- **Robustezza e affidabilità**: è la tolleranza del sistema ai guasti di alcuni componenti, la correzione di errori e la capacità di operare anche in situazioni di stress.
- **Scalabilità**: è la possibilità di adattare il sistema in un nuovo ambiente, solitamente più grande, senza intaccare le performance.
  - geografia dell'ambiente: garantire che tutta l'area di interesse sia coperta dal sistema. Questo deve valere anche se quest'area si estendesse;
  - densità dei dati disponibili: è il numero di unità disposte nello spazio che devono essere gestite.

- **Costo**: i costi riguardano:
  - i tempi di progettazione del sistema;
  - l'acquisto e l'aggiornamento dei dispositivi di cui è composto;
  - la configurazione e il testing;
  - la manutenzione ordinaria e straordinaria;
  - il consumo energetico;
  - ecc...

4.4 IPS con tecnologie radio-based

Un primo approccio alla creazione di IPS è stato **sfruttare una singola tecnologia**, ad esempio Access Point Wi-Fi o Beacon Bluetooth, al fine di imporsi sul mercato come *la tecnologia GPS per interni*.

Questo approccio si è rivelato poco efficiente, costoso o addirittura irrealizzabile in quanto si sono ottenuti **risultati poco accurati** rispetto a quelli attesi, nonostante si applicassero dei filtri al fine di minimizzare o compensare gli errori di rilevazione.

Le limitazioni legate alla tecnologia impiegata, la *necessità di acquistare un gran numero di sensori* da disporre nell'ambiente indoor secondo un determinato criterio, i tempi necessari alla disposizione dei sensori nell'ambiente e i costi di manutenzione dei sensori hanno reso questi sistemi poco competitivi. Per tutte le cause elencate *questo tipo di soluzione non ha trovato riscontro nel mercato*.

4.4.1 IPS con Wi-Fi

Utilizzare tecnologie radio-based come il WiFi potrebbe essere costoso e poco scalabile se si dovessero mappare più piani. Inoltre potrebbe essere tecnicamente impossibile a causa di possibili interferenze ambientali.

Per riuscire ad ottenere una precisione costante che si attesta intorno ai 10 metri si dovrebbero installare più AP (Access Point) in ogni edificio e/o piano.

4.4.2 IPS con Beacon Bluetooth

I Beacon Bluetooth (iBeacon[4]) non sono particolarmente indicati per creare IPS, piuttosto sono utili per descrivere dei ***geo-fence***[5] intorno ad essi.

Il protocollo iBeacon [8] è abbastanza elementare:

- gli iBeacon inviano a tutti, con intervallo regolare, sempre lo stesso pacchetto di informazioni;

[4]**iBeacon**: trasmettitori Bluetooth a bassa potenza e a basso costo che possono notificare la propria presenza ai dispositivi Bluetooth vicini

[5]**Geo-fence**: è un recinto/perimetro virtuale che circoscrive un'area geografica nel mondo reale

- tutti i dispositivi Bluetooth che transitano nell'aria di copertura possono ricevere tale messaggio;
- in alcuni casi i ricevitori possono essere programmati per reagire alla ricezione del messaggio.

Gli iBeacon permettono soltanto:

- di delimitare un'area geografica di pochi metri intorno al Bluetooth Beacon (lato emettitore);
- di rendere nota la presenza del Beacon agli smartphone che transitano nell'area di copertura (lato ricevitore).

Viste le limitazioni del protocollo non è possibile implementare un IPS che si basi esclusivamente su Beacon Bluetooth.

4.5 IPS con posizionamento magnetico e sensor fusion

Il **posizionamento magnetico** (sez. 3.3.1) è una soluzione IPS che si basa sui dati ricavati dal magnetometro integrato negli smartphone, cioè sul riconoscimento del *fingerprint magnetico* (sez. 3.3).

Tutte le costruzioni moderne si trovano in un **paesaggio magnetico** dovuto all'interazione tra la **struttura in acciaio** di cui sono composte ed il campo magnetico generato della Terra (che in assenza di strutture sarebbe pressoché costante). Analizzando le alterazioni del campo magnetico terrestre causato dall'edificio l'IPS è in grado di riconoscere la *fingerprint* dell'edificio e del piano in cui ci si trova.

Quindi questo sistema si distingue dai precedenti in quanto non necessita di aggiungere dell'hardware, bensì sfrutta al massimo le capacità computazionali dello smartphone, permettendo di ottenere un'accuratezza della stima di 1-2 metri. Questa tecnica viene utilizzata dalla società IndoorAtlas (sez. 5.1).

Anche se attualmente non esistono degli standard de facto per IPS, *il posizionamento magnetico sembra essere il* ***più completo*** *e conveniente presente sul mercato*, in quanto offre accuratezza di stima senza eccessivi requisiti hardware, mantenendo un costo totale relativamente ridotto. Secondo Opus Research il posizionamento magnetico emergerà come la tecnologia di localizzazione indoor "*fondamentale*".

Utilizzare un IPS che sfrutta la combinazione di più tecnologie mediante la sensor fusion (sez. 3.4) permette di:

- ottenere il massimo risultato,
- aumentare la precisione,
- abbattere i costi,
- sfruttare al meglio le caratteristiche di ogni tecnologia disponibile.

In senso più ampio, permette di **costruire un unico sistema** capace di fornire un servizio di posizionamento accurato sia che ci si trovi in ambiente outdoor che indoor, quindi **fondendo dati GPS e IPS**.

L'IPS di IndoorAtlas è un esempio di IPS che grazie alla sensor fusion permette di ottenere un'ottima stima della posizione indoor. La società IndoorAtlas detiene numerosi brevetti per il posizionamento magnetico e la sensor fusion. Nel 2015 ha vinto il premio SPIFFY del Consiglio Telecom per la tecnologia più innovativa.

Capitolo 5

IndoorAtlas

5.1 IPS IndoorAtlas

L'IPS **IndoorAtlas** permette di realizzare un **GPS per interni** in tempo reale sfruttabile da smartphone. Per farlo si utilizza la tecnica della sensor fusion dei vari sensori integrati e il download di planimetrie al fine di consentire agli utenti di orientarsi anche in spazi sconosciuti.

Si distingue dagli altri IPS in quanto:

- sfrutta una **tecnologia geomagnetica** connessa ad una piattaforma ***cloud***;
- ha una *precisione di rilevamento* della posizione dei target che si attesta intorno a **1-2 metri**.
- non richiede l'acquisto, l'installazione o la manutenzione di grandi infrastrutture costose.

Le descrizioni presenti in questo capitolo sono traduzioni dei contenuti presenti nel sito ufficiale di IndoorAtlas `http://www.indooratlas.com/`.

5.1.1 Caratteristiche

Le seguenti caratteristiche distinguono la soluzione IndoorAtlas dagli altri IPS:

- **scalabilità**: piattaforma cloud-based "infinitamente" scalabile;
- **infrastruttura cloud sicura**
- **posizionamento preciso**: precisione che può arrivare a 1-2 metri;
- **cross-platform SDK**: le SDK (*Software Development Kit - strumento per lo sviluppo software*) disponibili riguardano dispositivi Android e iOS;
- **facilità di implementazione**: flusso di lavoro intuitivo spiegato passo-passo.

- **calibrazione della mappa incrementale**: tutti i dati raccolti servono a migliorare il servizio.
- **costo ridotto**: nessuna necessità di acquistare, installare e mantenere una grande quantità di infrastrutture costose;
- **sistema gratuito fino a 100 utenti al mese**: modello di pricing ***pay as you grow*** (al crescere delle richieste).

5.1.2 Tecnologia geomagnetica e Sensor fusion

La **tecnologia geomagnetica** IndoorAtlas sfrutta il sensore magnetico (magnetometro) integrato all'interno dello smartphone per rilevare le anomalie nel campo magnetico della Terra al fine di individuare e monitorare la posizione di un target in ambiente chiuso.

Questa tecnologia è alla base del posizionamento indoor sviluppato da IndoorAtlas. Per ottenere un'ulteriore ottimizzazione si ricorre alla **sensor fusion** sfruttando anche altre fonti di informazione, unendo ogni fonte di dati legata alla posizione, per ottenere un posizionamento preciso e veloce.

Grazie alla sensor fusion, oltre alle alterazioni dei campi magnetici, vengono rilevati:

- i segnali Wi-Fi;
- i pacchetti Beacon Bluetooth;
- la pressione atmosferica;
- le variazioni di posizione dello smartphone mediante accelerometro e/o giroscopio;

permettendo di ottimizzare i tempi e la precisione necessari per ottenere il posizionamento.

Per implementare IndoorAtlas in app mobile è stata sviluppata una SDK[1] la quale è in grado di analizzare e interpretare:

- le alterazioni del campo magnetico percepito dal magnetometro dello smartphone;
- i dati forniti da altri sensor;
- i dati relativi alle onde Wi-Fi e Bluetooth.

[1] **Software Development Kit**: insieme di strumenti per lo sviluppo e la documentazione di software

5.1.3 Piattaforma cloud-based

La piattaforma cloud-based è l'interfaccia dedicata agli sviluppatori.

Permette di:

- creare le *venues* (sedi);
- gestire i dati relativi all'utilizzo dell'IPS;
- sviluppare app mobile che sfruttino i servizi di localizzazione per interni.

La piattaforma cloud semplifica l'integrazione delle app con il sistema di posizionamento magnetico, permettendo di utilizzare l'SDK e costruire servizi basati sulla localizzazione indoor.

5.1.4 Vantaggi

La piattaforma IndoorAtlas ha i seguenti vantaggi:

- nessun costo per il setup del sistema;
- tempo di setup ridotto;
- non è necessario installare e mantenere hardware dedicato;
- è possibile creare sistemi complessi senza dover installare fisicamente dei sensori;
- piattaforma *cloud-based* per implementare e controllare facilmente i servizi legati al posizionamento per interni;
- riduzione dei costi e dei tempi di gestione.
- la precisione dipende dall'intensità del campo magnetico della costruzione e dagli sforzi fatti nella fase di creazione dell'impronta digitale. Nel migliore dei casi si può ottenere una accuratezza di 1-2 metri.

5.1.5 Limitazioni

La tecnologia IndoorAtlas utilizza le variazioni di campo magnetico per determinare la posizione. Queste variazioni sono tipicamente osservabili in edifici con struttura in acciaio.

- Il tempo di generazione dei fingerprint dipende dalla dimensione della mappa. Può richiedere solo pochi minuti o anche alcune ore se la mappa è composta da centinaia di sentieri.
- Edifici con struttura del telaio in legno possono essere ambienti difficili su cui operare a meno che non siano presenti strutture in acciaio o oggetti che causino anomalie magnetiche.

- Questa tecnologia **può funzionare solo all'interno di spazi chiusi** o nelle immediate vicinanze di edifici in cui sono state registrate le alterazioni magnetiche.
- Potrebbe essere necessario ri-mappare le aree già mappate se l'edificio ha attraversato un grande cambiamento strutturale, ad esempio, le strutture in acciaio sono state modificate o gli oggetti di grandi dimensioni come mensole sono state spostate.
- Con IndoorAtlas è possibile mappare luoghi all'aperto, ma solo se sono presenti strutture in acciaio come ad esempio stadi sportivi o se ci si trovi nell'immediata vicinanza dell'edificio mappato. Tuttavia IndoorAtlas diventa molto meno preciso quando si trova in spazi aperti o con poche strutture in acciaio.

Parte II

Background: Determinare il cammino minimo

Capitolo 6

Calcolo del percorso minimo

Per **percorso minimo** si intende la strada percorribile che con il minor numero di passi permette di partire da un dato punto iniziale e arrivare ad un punto finale scelto. In questa tesi il percorso minimo è stato calcolato utilizzando la **distanza di Manhattan** e l'**algoritmo A***.

6.1 Determinare una distanza

In matematica si definisce **distanza** una qualunque relazione tra due punti A e B che verifichi le condizioni di positività, simmetria e diseguaglianza triangolare:

- $\delta(A, B) \geq 0$
  se $\delta(A, B) = 0$ allora $A = B$
- $\delta(A, B) = \delta(B, A)$
- $\delta(A, B) \leq \delta(A, C) + \delta(C, B)$

6.1.1 Distanza euclidea

Secondo il teorema di Pitagora, la distanza euclidea tra due punti P e Q con coordinate $P = (x_1, y_1)$ e $Q = (x_2, y_2)$ è:

$$d(P, Q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{6.1}$$

Tale distanza non è applicabile per definire i percorsi minimi in contesti in cui sono presenti ostacoli o vincoli.

6.1.2 Distanza di Manhattan

La **geometria del taxi** o **distanza di Manhattan** è un concetto geometrico introdotto da *Hermann Minkowski* secondo il quale la distanza tra due punti è la somma del valore assoluto delle differenze tra le loro coordinate.

Il nome è riferito al sistema stradale tipico del distretto newyorkese di Manhattan in cui gran parte delle vie di scorrimento sono ortogonali tra di loro.

6.1.3 Taxi-distanza

La distanza minima che misura lo spostamento reale che si avrebbe in un ambiente dove sono presenti vincoli è definita come:

$$d_T(P,Q) = |x_1 - x_2| + |y_1 - y_2| \tag{6.2}$$

Per **distanza di Manhattan** d_T [7] si intende la distanza che intercorre tra due punti nello spazio euclideo in cui si considera un sistema di coordinate cartesiane, cioè la somma delle lunghezze delle proiezioni sugli assi cartesiani dei segmenti che congiungono i due punti.

La distanza alternativa d_T prende il nome di *distanza di Minkowski* o **distanza del taxi** perché è la distanza più breve che un'automobile dovrebbe percorrere per raggiungere un qualsiasi punto situato in una città a forma di reticolato.

6.2 Algoritmo A*

L'algoritmo A* (A-start) è un algoritmo di ricerca *best-first*[1] applicabile su grafi. È stato descritto nel 1968 da Peter Hart, Nils Nilsson, e Bertram Raphael.

A* permette di individuare il **percorso minimo** percorribile che si estende a partire da un dato nodo iniziale (detto *start*) e uno finale (*goal*) utilizzando una **stima euristica** che classifica ogni nodo.

Tale algoritmo è applicabile all'interno di un ambiente in cui sono presenti ostacoli.

6.2.1 Caratteristiche degli algoritmi di ricerca

- Si definisce **algoritmo di ricerca ammissibile** quell'algoritmo che garantisce di trovare sempre il percorso più corto verso una meta.
- Si definisce la **migliore euristica possibile** l'effettiva distanza minima verso meta. Un esempio di euristica ammissibile è la distanza in linea d'aria tra start e goal.

6.2.2 La funzione euristica dell'algoritmo A*

Ciò che rende ammissibile A* è la sua ***funzione euristica***, cioè quella funzione che permette di stimare la distanza e il costo necessario per arrivare al nodo goal indicato,

[1] **Best-first search**: strategia di ricerca informata utilizzata per la risoluzione di problemi basati sulla ricerca sfruttando la conoscenza di ulteriori dettagli sugli stati del problema da risolvere

definendo il nodo successivo che l'algoritmo dovrà considerare in base al percorso più probabile e a quello più breve.

Una funzione euristica che rende una ricerca A* ammissibile è detta **euristica ammissibile**.

6.2.3 Funzionamento dell'algoritmo

Per ogni nodo selezionato da A*:

1. si definisce un costo di entrata (di solito zero per il nodo iniziale);
2. si stima la distanza che intercorre tra il nodo corrente e quello goal;
3. si esegue la funzione euristica facendo la somma del costo di entrata e la stima della distanza;
4. infine si assegna al nodo percorso il valore della funzione euristica calcolato.

Ogni nodo analizzato viene aggiunto ad una **lista dei nodi percorribili**, da cui verranno rimossi i nodi con valore della funzione euristica più basso.

Per ciascun nodo attraversato A* calcola la funzione euristica, cioè la somma cumulativa dei pesi di tutti i nodi visitati più il costo dell'operazione per raggiungere il nuovo nodo, e ne salva il risultato nel nodo.

- Se la lista dei nodi percorribili è vuota vuol dire che non esistono percorsi tra start e goal;
- Se il nodo considerato è il nodo goal, A* si arresta restituendo il percorso ottenuto;
- Se il nodo considerato non è il nodo goal allora saranno creati nuovi nodi per tutti i nodi vicini ammissibili, cioè quei nodi che non costituiscono un vincolo o un ostacolo.

Questo tipo di approccio permette di ricostruire il percorso a partire dai nodi più vicini senza dover memorizzare il percorso in ogni nodo.

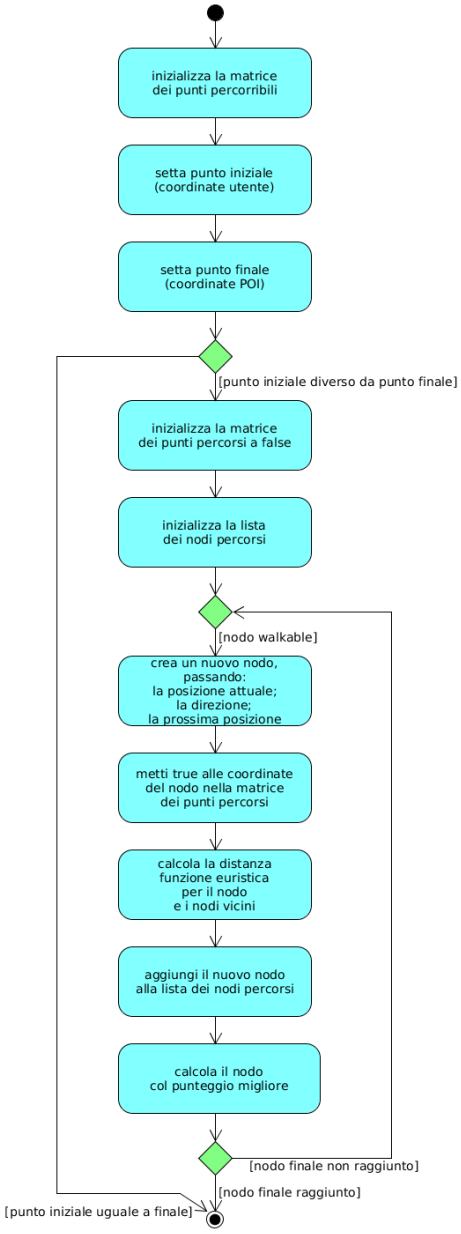


Figura 6.1: Algoritmo A*

Parte III

Progetto

Capitolo 7

Analisi

7.1 Vision

In questa tesi si vuole realizzare un sistema di localizzazione indoor scalabile ed efficiente che geolocalizzi l'utente e lo guidi verso un **POI** selezionato rappresentando in real-time il percorso minimo per raggiungerlo, proprio come se si usasse un GPS.

7.2 Goals

7.2.1 Goals principali

1. Implementare dei **filtri immagine** (sez. 7.5.2) per analizzare la planimetria dell'edificio;
2. Estrarre i **punti navigabili** e i **vincoli** dalle planimetrie;
3. Creare una **matrice di punti percorribili** per poter eseguire l'algoritmo di ricerca del percorso minimo;
4. Implementare l'a**lgoritmo A*** (sez. 6.2) per ottenere il percorso minimo tra due punti dati (posizione dell'utente e POI selezionato);
5. Generare un'**interfaccia utente** che permetta di visualizzare il percorso minimo utente-POI;
6. Creare una corrispondenza tra le coordinate in formato GPS ricevute dall'**IPS IndoorAtlas** (sez. 5.1) e le coordinate cartesiane del punto che rappresenterà l'utente nella planimetria.
7. Ricalcolare dinamicamente in base agli spostamenti dell'utente.

7.2.2 Goals secondari - lato utente

1. Generare un'interfaccia per selezionare i POI;
2. Creare una ricerca sui POI disponibili.
3. Tracciare il percorso aggiornato utente-prodotto.

7.2.3 Goals secondari - lato amministratore

1. Ottimizzare la creazione della matrice dei punti percorribili:
   - estraendo i punti navigabili e i vincoli dalla planimetria una sola volta, salvandoli in locale;
   - ricostruendo la matrice semplicemente leggendo le informazioni salvate in locale.
2. Generare un'interfaccia per la creazione di POI nell'ambiente considerato.
3. Assegnare le coordinate spaziali univoche ad ogni POI.
4. Creare un'interfaccia REST[1] verso il server per salvare e/o leggere i POI di ogni edificio.

7.3 Definizione dei requisiti

Il sistema che si vuole realizzare è un IPS, cioè un sistema modulare che permetta la geolocalizzazione degli utenti in ambienti indoor.

Il sistema considerato è composto da due parti principali:

1. un'app per smartphone che invia/riceve dati al servizio remoto IndoorAtlas;
2. un server privato contenente il database dei POI di ogni edificio/piano.

7.3.1 Requisiti funzionali

I requisti funzionali sono quelle caratteristiche che permettono al sistema di funzionare correttamente.

I requisiti funzionali dipendono dalle interviste svolte durante il periodo di tirocinio presso **GetConnected**.

I requisiti del sistema proposto sono:

[1] **REpresentational State Transfer**: architettura software per sistemi di ipertesto distribuiti come il WWW

- ottenere informazioni sulla posizione dell'utente dal servizio IndoorAtlas;
- scaricare l'immagine della planimetria dell'edificio in cui l'utente si muove;
- tradurre la posizione dell'utente in un riferimento grafico (cerchio colorato);
- ottenere informazioni sui POI presente nell'edificio in cui il sistema viene eseguito;
- caricare, in una lista, i POI del piano in cui l'utente viene geolocalizzato;
- permettere la selezione di un POI target;
- calcolare e tracciare in modo dinamico il percorso minimo POI-utente.

Per il corretto funzionamento del sistema si assume:

- di avere un ambiente chiuso di riferimento, cioè un edificio come ad esempio un'abitazione domestica, realizzata con infrastrutture in acciaio;
- di utilizzare uno smartphone con sistema operativo Android 4.3 (versione 5.0 o superiore consigliata) che integri:
  - connettività Internet (Wi-Fi, 3/4G);
  - magnetometro (bussola);
  - giroscopio;
  - accelerometro;
- che l'ambiente di riferimento sia già stato mappato, cioè che esista una fingerprint magnetica dell'area di utilizzo del sistema;
- che sia stata caricata l'immagine della planimetria degli edifici mappati.

Questi requisiti si possono considerare come dei vincoli per la realizzazione e l'utilizzo del sistema, cioè caratteristiche senza cui il sistema non è in grado di geolocalizzare e visualizzare la posizione dell'utente.

7.3.2 Requisiti funzionali opzionali

I requisiti funzionali opzionali non sono vitali per il corretto funzionamento del sistema, ma sono comunque utili per rendere realizzabili alcune *feature*.

- Lo smartphone utilizzato può integrare:
  - GPS (sez. 1) e/o AGPS (sez. 1.1.2);
  - connettività Bluetooth 4.0 o superiore.
- Nell'ambiente considerato possono essere disposti:
  - AP (Access Point) Wi-Fi;

  - Beacon Bluetooth.

La presenza di AP Wi-Fi anche se opzionale è comunque una caratteristica che può accrescere la *User Experience* in quanto permetterebbe al sistema di essere più accurato nella stima della posizione indoor.

7.3.3 Requisiti non funzionali

I requisiti non funzionali descrivono quelle caratteristiche che sono soggettive per ogni utente.

I requisiti non funzionali sono:

- realizzare un'app che abbia un'interfaccia grafica utente fluida di immediato utilizzo,
- visualizzare l'evoluzione del sistema attraverso feedback in real-time;
- ottenere un sistema che funzioni in modo efficiente (permettendo all'utente di risparmiare tempo);
- creare un sistema quanto più accurato possibile, sfruttando la sensor fusion;
- creare un'affiliazione tra utente e servizio, aumentando la soddisfazione del cliente.

7.4 Analisi dei requisiti

L'analisi dei requisiti del progetto serve a definire le funzionalità che il sistema proposto dovrà avere, cioè quelle caratteristiche che l'utente sfrutterà.

In questo progetto, dal punto di vista dell'utente, sono state evidenziate le seguenti necessità:

- determinare la posizione dell'utente attraverso un sistema di geolocalizzazione;
- far visualizzare all'utente la propria posizione in modo grafico, cioè con un cerchio colorato;
- il cerchio colorato deve muoversi nella direzione in cui si muove l'utente;
- il cerchio deve essere disposto su una planimetria in modo da creare un contesto di utilizzo del progetto, cioè un collegamento tra ciò che l'utente vede e dove esso si trova.

7.4.1 Casi d'uso

Determinare la propria posizione

In questo caso d'uso l'utente esegue l'app nel proprio smartphone Android al fine di determinare la propria posizione in un ambiente il quale potrebbe anche essergli sconosciuto.

Per il corretto funzionamento del progetto il servizio di geolocalizzazione deve essere già attivato nello smartphone in uso. Questo è un vincolo per il funzionamento del sistema. Questo vincolo è stato imposto perché senza geolocalizzazione il servizio di IndoorAtlas non può determinare in nessun modo la posizione dell'utente. Se la geolocalizzazione non è già attiva, si visualizzerà un messaggio che inviterà l'utente ad attivare tale servizio.

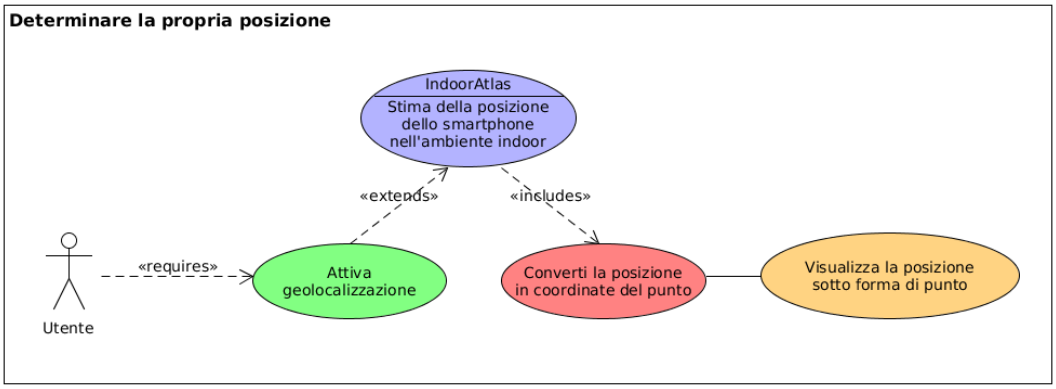


Figura 7.1: Caso d'uso 1: Determinare la propria posizione

Una volta attivata l'app registrerà ed invierà le informazioni captate dal magnetometro al server IndoorAtlas. A sua volta questo servizio remoto analizzerà i dati per riconoscere il fingerprint e restituirà le coordinate della posizione dell'utente in formato GPS.

Ad ogni ricezione le coordinate saranno tradotte dal formato GPS $(latitudine, longitudine)$ in coordinate cartesiane (x, y). Dopo qualche secondo nell'app apparirà l'immagine della planimetria e un cerchio colorato inizialmente posto a $(0, 0)$.

Ottenute le coordinate cartesiane sarà possibile aggiornare le coordinate del punto che determina la posizione dell'utente all'interno della planimetria.

Navigazione verso un POI target

Questo caso d'uso estende il precedente, infatti si da per scontato che l'IPS stia funzionando e che l'utente sia geolocalizzato all'interno dell'edificio.

In questo caso d'uso viene considerata la necessità di un utente nel raggiungere un POI di suo interesse (POI target). Immaginando che l'utente voglia raggiungere il POI nel più breve tempo possibile è utile considerare l'utilizzo di un algoritmo per il calcolo del percorso più breve. Il POI target corrisponderà al punto di arrivo del percorso, mentre la posizione dell'utente sarà il punto di partenza. Questi punti saranno passati all'algoritmo A* 6.2 per il calcolo del percorso minimo che sarà visualizzabile a schermo.

Come condizione a contorno in questo caso d'uso si considera l'esistenza di una lista di POI precedentemente salvata in un server raggiungibile via Internet. Questa lista deve raccogliere tutti i POI geolocalizzati di ogni edificio e quindi di ogni piano.

L'utilità della lista-POI è permettere all'utente di ottenere un elenco di POI che possono essere selezionati o ricercati in modo da ottenere il percorso minimo per arrivarci, risparmiando il tempo che si sarebbe perso in una ricerca random. La lista sarà ricaricata ogni qual volta si cambia piano, in modo da avere riferimenti raggiungibili direttamente, così quando e se l'utente avrà la necessità di raggiungere un determinato POI potrà ricercare e selezionare il POI target dalla lista.

Il percorso minimo utente-POI sarà costantemente aggiornato in base alla posizione occupata dall'utente all'interno della planimetria, dando l'illusione grafica che avvicinandosi al POI target il percorso disegnato nella planimetria diminuisca, viceversa se ci si allontana.

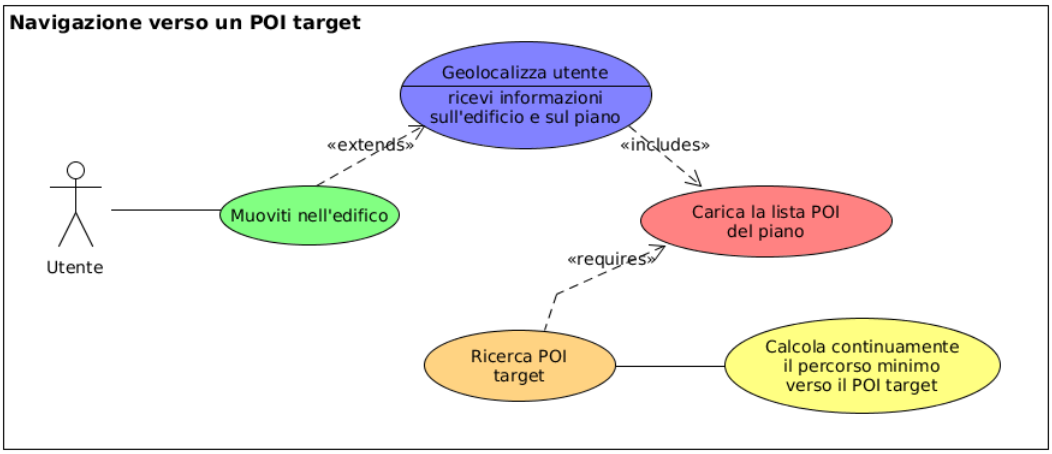


Figura 7.2: Caso d'uso 2: Navigazione verso un POI target

7.4.2 Scenari

Gli scenari rappresentano delle astrazioni di utilizzo del sistema. In questo progetto sono stati considerati principalmente due scenari in quanto rappresentativi.

Scenario 1 - Determinare la propria posizione

Nel primo scenario ci si immedesima nell'utente che vuole determinare la propria posizione all'interno di un edificio.

Anche in questo scenario è presente il vincolo dell'attivazione dei servizi di geolocalizzazione e dell'accesso ad Internet, senza cui il sistema sarebbe inutilizzabile.

Scenario 1	Determinare la propria posizione
ID	S1
Descrizione	L'utente vuole determinare la propria posizione all'interno di un edificio
Attore	Utente
Precondizioni	Geolocalizzazione attiva sullo smartphone in uso e utente all'interno di un edificio di cui è nota la fingerprint
Scenario principale	L'utente definisce la propria posizione in base al punto tracciato sulla planimetria
Scenari alternativi	L'utente che inizialmente non ha la geolocalizzazione attiva viene invitato ad attivarla
Post condizioni	L'utente si muove nel piano e vede muoversi il cerchio Intuendo un rapporto tra questi spostamenti si geolocalizza

Alla base di questo scenario c'è un sistema che funziona (cioè quello che si vuole produrre) e la voglia di soddisfare un'esigenza dell'utente, cioè sapere la posizione.

La riuscita di questo scenario rappresenta un passo in più nella realizzazione di un IPS. La sua realizzazione creerebbe di fatto un GPS per ambienti chiusi, cioè uno degli obiettivi cardine della tesi.

Scenario 2 - Navigazione verso un POI target

Nel secondo scenario ci si immedesima nelle necessità dell'utente che desidera raggiungere un punto di interesse. In base al piano in cui si trova, i POI raggiungibili cambieranno.

Ad esempio si potrebbe immaginare un utente in un aeroporto straniero intento a raggiungere il gate giusto. In questo scenario l'app permetterebbe all'utente di non perdersi e di risparmiare tempo, guidandolo fino al raggiungimento dell'obiettivo.

Scenario 2	Navigazione verso un POI target
ID	S2
Descrizione	L'utente ricerca e seleziona un POI di suo interesse
Attore	Utente
Precondizioni	Geolocalizzazione riuscita (scenario S1 7.4.2) Aggiornare la lista di POI in base al piano
Scenario principale	L'utente seleziona un POI target
Scenari alternativi	L'utente naviga verso il POI target seguendo il percorso minimo tracciato sulla planimetria
Post condizioni	L'utente raggiunge il POI target

La riuscita di questo scenario conferma l'importanza che la IPS avrebbe nell'esperienza utente all'interno di aree delimitate come centri commerciali. La sua realizzazione confermerebbe un altro obiettivo, cioè la possibilità di essere guidati attraverso un edificio fino ad una meta selezionata.

L'importanza di questo scenario si evince ancor di più se si considera la sua applicazione in contesti in cui l'utente non ha informazioni sul luogo in cui si trova. Facile immaginare un utente spiazzato che si trova in un paese straniero di cui non conosce la lingua o casi in cui l'utente vuole sapere con immediatezza, senza perdere tempo in ricerche manuali, dove si trova un prodotto o un servizio.

Rendere la visualizzazione del percorso minimo una questione grafica limita i problemi di traduzione tra lingue diverse. Questo significa che un sistema che implementa una soluzione simile non è costretto ad adattare le descrizioni per ogni lingua/paese in cui si vuole utilizzare.

7.5 Analisi del problema

7.5.1 Architettura del sistema

In base ai requisiti del sistema, si è progettata la seguente architettura:

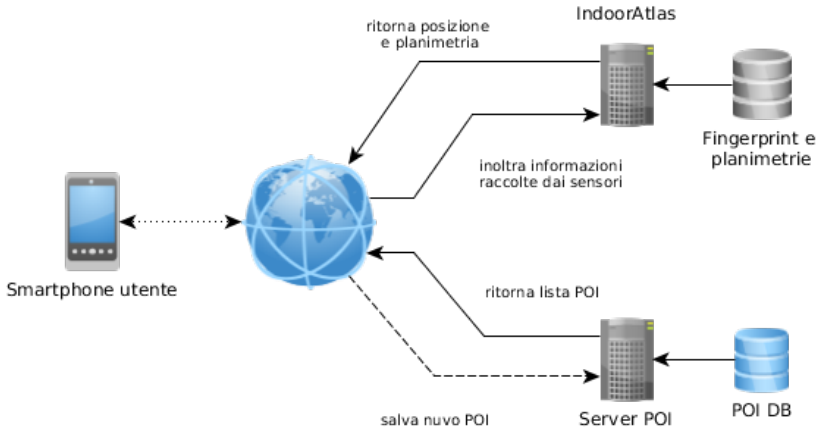


Figura 7.3: Architettura del sistema

L'architettura si suddivide in due parti, la parte client rappresentata dallo smartphone dell'utente in cui è eseguita l'app per la geolocalizzazione e la parte server, suddivisa tra servizio IndoorAtlas e un database della lista POI.

In questo semplice schema le due parti sono connesse attraverso Internet. Le linee tratteggiate rappresentano l'opzionalità delle operazioni di I/O.

7.5.2 Estrarre informazioni da un'immagine

Analizzando i requisiti si evince che trarre informazioni dall'immagine della planimetria è vitale per poter visualizzare la posizione dell'utente e disegnare il percorso minimo.

Essendo l'immagine di tipo PNG, non si hanno direttamente delle informazioni su quali sono i punti percorribili e quali invece i vincoli. Per risolvere questo problema si è progettato un approccio atto ad astrarre, in modo automatico, gli ostacoli e i punti navigabili. Di seguito i passi di analisi che hanno portato alla loro definizione.

Sono stati considerati tre tipi di filtri immagine:

- FastBlur;
- B/W;
- Invert color.

Ogni filtro ha una caratteristica particolare di utilizzo. Il loro scopo è rendere l'immagine PNG più "piatta", eliminando sfumature o informazioni non necessarie all'analisi da eseguire nel passaggio successivo.

Filtro blur

Il compito di questo filtro è sfocare l'immagine della planimetria data in ingresso. Il suo impiego ha un duplice motivo:

- evidenziare i contorni dei perimetri, in particolare i muri;
- limitare o addirittura eliminare alcuni dettagli inutili. In questo contesto ci si riferisce agli spazi bianchi presenti nelle finestre o scale.

Filtro B/W

Il filtro B/W (black/white) dovrà prendere in ingresso l'immagine della planimetria (che potrebbe essere colorata o con aree trasparenti), restituendo in output una nuova "piatta" con solo due colori, cioè bianco e nero, senza ammettere sfumature grigie.

Di seguito un semplice frammento di pseudocodifica.

```
//pseudocosifica del filtro B/W

for (int x = 0; x < width; x++) {
    for (int y = 0; y < height; y++) {

        pixelColor = getPixel(x, y);

        if (pixelColor == WHITE) {
            setPixel(x, y, WHITE);
        } else {
            setPixel(x, y, BLACK);
        }
    }
}
```

Come si evince dal codice precedente si considera l'analisi di ogni pixel. Se il pixel è bianco, allora il suo colore non deve essere cambiato.

Nel caso in cui il colore del pixel in esame fosse diverso da bianco, allora si avrebbe una sostituzione del colore in nero. Questo semplice filtro permette di risolvere molti problemi, perché livellare il tutto in bianco/nero permetterà di definire molto facilmente la matrice dei punti percorribili.

Ovviamente per risparmiare tempo non c'è bisogno di analizzare ogni volta la stessa immagine, basterebbe farlo solo una volta e salvare la matrice dei colori, quindi passare direttamente alla lettura della matrice, risparmiando tempo di computazione. Questa potrebbe essere una ottimizzazione.

Filtro invert color

Un filtro che permette di invertire i colori servirebbe nei casi in cui la planimetria in ingresso fosse composta da un colore di sfondo nero e linee bianche.

Anche per questo filtro si potrebbe applicare una ottimizzazione, calcolando e salvando la matrice dei colori; così se si ricevesse di nuovo la stessa immagine in input non si dovrebbe attendere il calcolo, ma si passerebbe direttamente alla lettura della matrice.

```
// pseudocodifica del filtro invert color
for (int x = 0; x < width; x++) {
    for (int y = 0; y < height; y++) {

        pixelColor = getPixel(x, y);

        R = 255 - red(pixelColor);
        G = 255 - green(pixelColor);
        B = 255 - blue(pixelColor);

        setPixel(x, y, rgb(R, G, B));
    }
}
```

7.5.3 Creazione della matrice dei punti navigabili

I filtri trattati precedentemente avevano lo scopo di "appiattire" l'immagine della planimetria al fine di renderla analizzabile. Superati gli step di filtraggi sarà possibile analizzare l'immagine, creando un contesto su cui immaginare le applicazioni del sistema.

Una volta che l'immagine è stato ridotta ai soli colori bianco e nero, è semplice pensare di associare il tipo di colore ad un valore booleano. Questo permette di creare un contesto in cui, se il valore alla posizione (x, y) è *true* allora vorrà dire che è percorribile, cioè:

- l'utente potrà essere geolocalizzato nel punto (x, y) descritto come percorribile;
- il percorso minimo potrà passare da quel punto, cioè l'algoritmo A* riconoscerà in quel punto la possibilità di aggiungere un nodo.

Viceversa, se alla posizione (x, y) si ha un valore *false*, allora quello sarà:

- un ostacolo: cioè un muro, o un'area in cui non ci si può trovare;
- un vincolo: in quanto il percorso minimo di fronte ad un valore *false* non potrà creare un nuovo nodo, quindi si avrà una deviazione.

Ovviamente un POI non dovrebbe essere piazzato in un'area in cui non è possibile accedere o comunque dove corrisponde un vincolo.

Di seguito la pseudocodifica per il riconoscimento dei colori e quindi la definizione dei punti percorribili da inserire nella matrice.

```
//pseudocodifica della matrice dei punti percorribili

int pixelColor = getPixel(x, y);

if (pixelColor == Color.WHITE) {
    // punti percorribili
    matrixWalkable[x][y] = true;
} else {
    // ostacoli
    matrixWalkable[x][y] = false;
}
```

Capitolo 8

Design

In questo capitolo si interpreteranno i requisiti del sistema descrivendo una **soluzione architetturale di massima**. Per farlo sono stati prodotti dei diagrammi delle attività, in modo da poter avere una descrizione indipendente dai particolari strumenti usati per la costruzione del sistema. Nel primo caso d'uso l'utente può determinare la propria posizione all'interno di un ambiente, potenzialmente a lui sconosciuto.

Una soluzione è far visualizzare la posizione dell'utente come un cerchio all'interno della planimetria dell'edificio in cui l'utente si muove, creando un rapporto tra la posizione dell'utente e quella del cerchio. La soluzione proposta riceve le coordinate dell'utente, in formato GPS, dal servizio IndoorAtlas. Queste devono essere poi tradotte in coordinate cartesiane in modo da posizionare il cerchio nella posizione assunta dall'utente. In questa soluzione per completezza è stata aggiunta la ricezione della lista POI.

Per poter disegnare il cerchio (che rappresenta la posizione dell'utente) all'interno della planimetria nelle posizioni in cui è possibile transitare, bisogna attendere che l'immagine sia analizzata dall'algoritmo di riconoscimento dei punti navigabili e degli ostacoli. Questo processo evita che il cerchio che rappresenta l'utente, in caso di una ricezione di coordinate scorrette, sia disegnato al di fuori della planimetria o su vincoli; quindi evitando o comunque limitando la visualizzazione di errori di posizionamento.

Dal diagramma delle attività (fig. 8.1) si evince che alcune operazioni possono essere svolte in modo parallelo, in quanto indipendenti. La scelta di visualizzare la planimetria immediatamente permette di dare subito un feedback visivo dell'avvenuto download dell'immagine all'utente, in modo da limitare tempi morti.

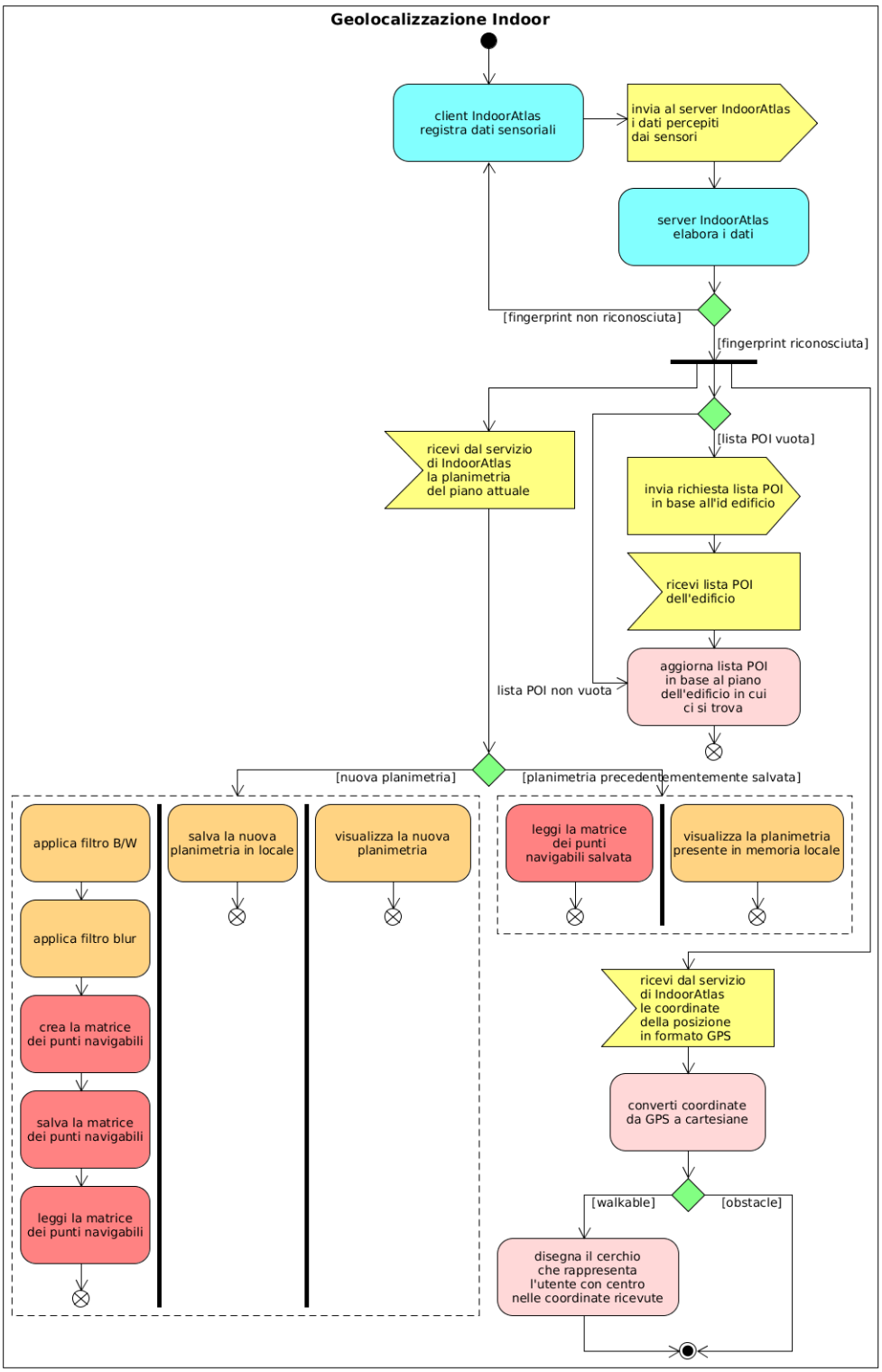


Figura 8.1: Diagramma delle attività - Geolocalizzazione dell'utente

Il secondo diagramma delle attività (fig. 8.2) descrive una soluzione per visualizzare a schermo il percorso minimo utente-POI. Anche in questo caso è imperativo che la fingerprint dell'edificio sia riconosciuta per ottenere le informazioni relative alla geolocalizzazione dell'utente. In particolare in questo diagramma si vede che la lista dei POI viene scaricata una sola volta e ricaricata solo la parte relativa al piano in cui l'utente si muove. Questa scelta non è del tutto vincolante, ma in questo frangente è una soluzione applicabile al problema.

Un'altra possibile soluzione è eseguire il download della lista POI solo del piano in cui ci si trova. Questo però avrebbe significato una dipendenza con la connessione disponibile molto più stringente aprendo ad altri problemi, come per esempio scegliere come reagire nel caso in cui l'utente cambia piano ma la connessione è assente. Nel peggiore dei casi questo scenario impedirebbe di ricevere la lista dei POI e quindi renderebbe impossibile la selezione di POI target.

In questo diagramma non sono state indicate soluzioni di *caching* della lista POI. Sarebbe possibile salvare questa lista sotto forma di file JSON per poi essere riletta, il che potrebbe essere una soluzione in caso in cui l'utente esce e rientra spesso dall'edificio. In questo caso si dovrebbe immaginare un timer dopo cui le informazioni del file JSON sono aggiornate.

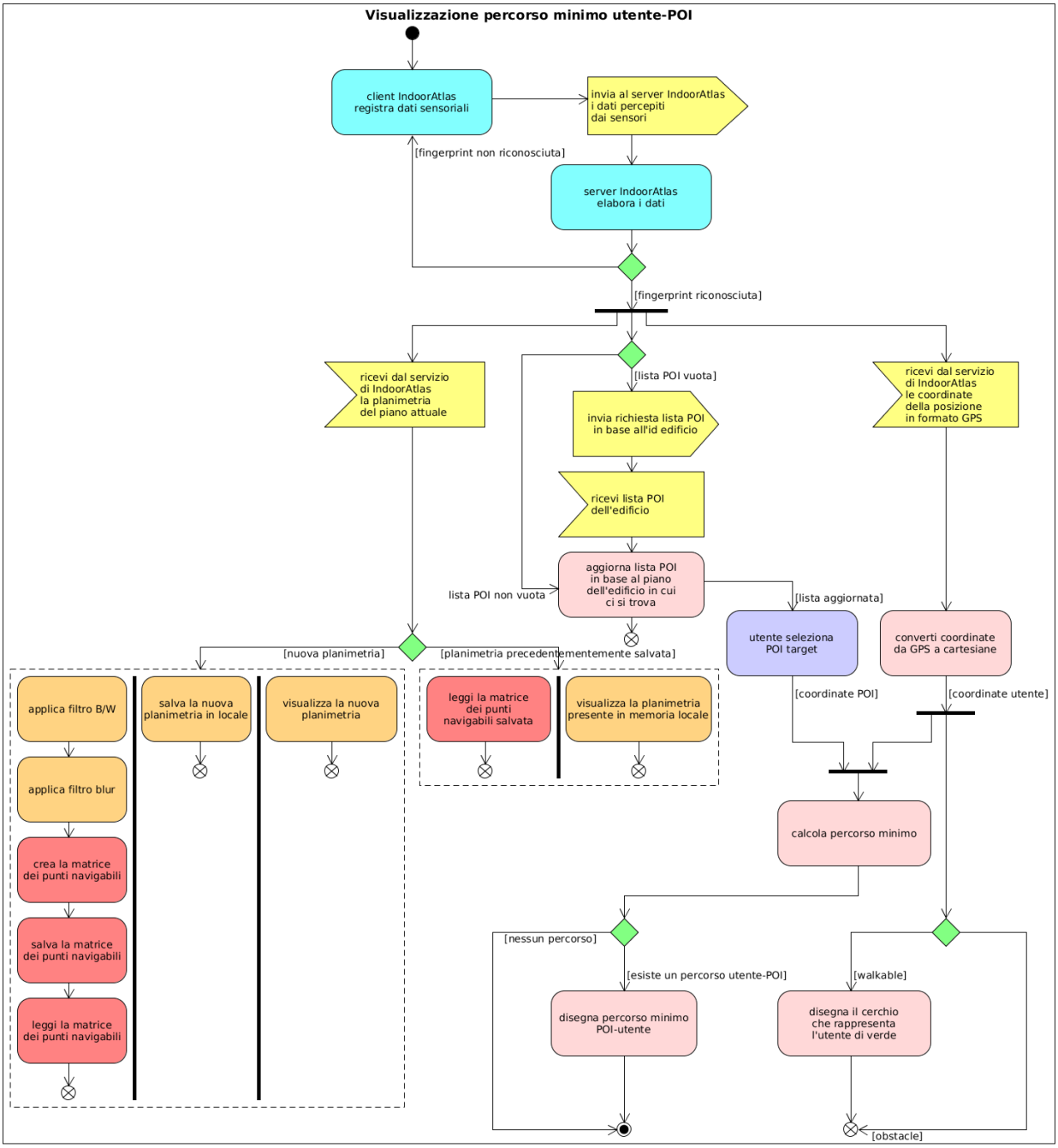


Figura 8.2: Diagramma delle attività - Visualizzazione percorso minimo utente-POI

L'ultimo diagramma delle attività, al contrario dei precedenti, non riguarda direttamente l'utente, bensì l'amministratore del sistema.

In questo caso si considera la creazione di nuovi POI. Ogni POI deve essere compilato e salvato per poi essere inviato al server. Se il POI non è completo in tutte le sue parti, allora deve essere visualizzato un messaggio che invita l'amministratore a inserire tutti i campi.

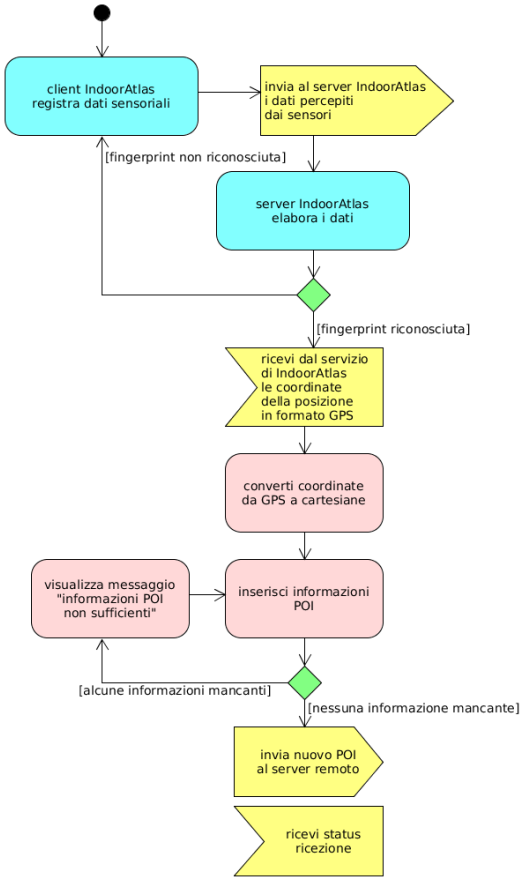


Figura 8.3: Diagramma delle attività - Invio POI al server

Capitolo 9

Implementazione

9.1 Activity

MainActivity

`MainActivity` è la prima classe richiamata nel progetto, cioè l'*Activity*[1] che permette di eseguire l'app in Android.

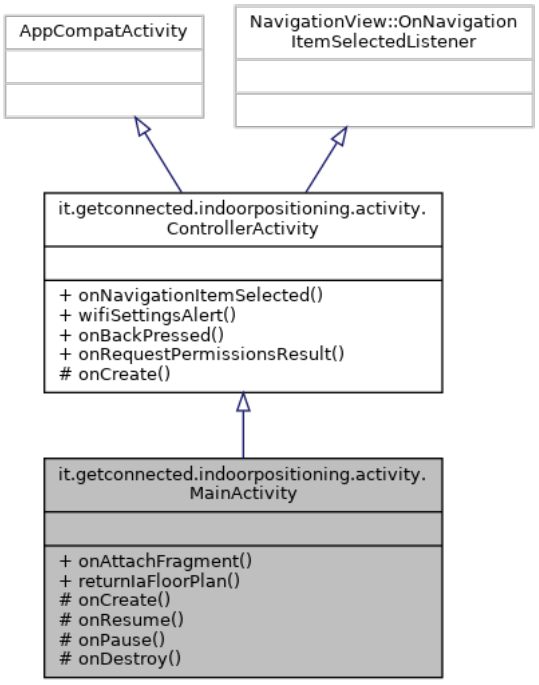


Figura 9.1: MainActivity

[1] **Activity**: classe Android che ha un proprio ciclo di vita espresso da metodi di *callback*. Ogni applicazione Android ha almeno una activity

L'obiettivo di questa classe è creare un canale di comunicazione con il servizio di localizzazione IndoorAtlas al fine di inoltrare i dati ricevuti alle classi interessate a riceverli. Per farlo si istanziano le API della SDK attraverso un servizio Android e si utilizza una classe interfaccia.

`OnFragmentInteractionListener`

In `MainActivity` è presente l'interfaccia `OnFragmentInteractionListener`. Lo scopo di questa interfaccia è ricevere i dati dalla SDK di IndoorAtlas e restituirli ai vari *Fragment*[2] che si alternano nel *content_main*.

```
public interface OnFragmentInteractionListener {

    void onIaRegionVenue(IARegion iaRegionVenue);

    void onIaRegionFloorPlan(IARegion iaRegionFloorPlan);

    void onIaLocation(IALocation iaLocation);

    void onIaLatLng(IALatLng iaLatLng);

    void onIaFloorPlan(IAFloorPlan iaFloorPlan);

    void onIaFloorPlanImage(File floorPlanImage);
}
```

`onCreate(...)`

Nel metodo `onCreate(...)` viene creato il canale con il servizio IndoorAtlas istanziando la classe `LocationService` con un *intent;*

```
...
//Start LocationService
Intent intentService = new Intent(this, LocationService.class);
startService(intentService);
...
```

`onResume()`

Nel metodo `onResume()` viene fatto il binding con la classe `LocationService`. Aver disposto il binding in questo metodo permette di evitare errori nel caso in cui l'app fosse risvegliata dal backgroud.

[2]**Fragment**: componente che rappresenta un comportamento o una porzione di interfaccia utente in un'activity. È una sezione modulare che ha un proprio ciclo di vita, può riceve eventi di input, ecc. . . .

In questo metodo viene anche eseguita la registrazione dell'activity al servizio di download per file. In questo caso servirà per il download delle planimetrie.

```
...
//Resume LocationService
Intent intentService = new Intent(this, LocationService.class);
bindService(intentService, mServiceConnection, BIND_AUTO_CREATE);
registerReceiver(onComplete, new
    IntentFilter(DownloadManager.ACTION_DOWNLOAD_COMPLETE));
...
```

onPause()

Nel metodo `onPause()` viene eseguito l'**unbinding** dal servizio offerto dalla classe `LocationService`. Aver disposto l'unbinding in questo metodo permette di limitare l'utilizzo della batteria quando l'app è in background evitando di essere aggiornati sulla posizione.

```
...
if (mLocationManager != null) {
    mLocationManager.unregisterRegionListener(mRegionListener);
    mLocationManager.removeLocationUpdates(mLocationListener);
}

unregisterReceiver(onComplete);
unbindService(mServiceConnection);
...
```

onDestroy(...)

Il metodo `onDestroy(...)` viene richiamato quando si chiude l'app. In questo metodo viene bloccato il servizio dalla classe `LocationService` per evitare errori e l'utilizzo della batteria in quanto rimarrebbe un servizio attivo senza motivo.

```
...
Intent intentService = new Intent(this, LocationService.class);
stopService(intentService);
...
```

+ onAttachFragment(...)

Il metodo `onAttachFragment(...)` serve ad assegnare al riferimento dell'interfaccia `OnFragment- InteractionListener` il fragment a cui passare i dati. Nel caso in cui un fragment non avesse implementato l'interfaccia si solleverebbe un'eccezione e l'applicazione terminerebbe.

```
...
if (fragment instanceof OnFragmentInteractionListener) {
    mListener = (OnFragmentInteractionListener) fragment;
    returnVenueRegion();
    returnFloorPlanRegion();
    returnImageFile();
    returnIaLocation();
    returnIaLatLng();
    returnIaFloorPlan();
} else {
    throw new RuntimeException(fragment.toString()
    + " must implement OnFragmentInteractionListener");
}
...
```

Come si può notare dal frammento di codice sono presenti molti metodi *return*. Questi servono per memorizzare i dati della posizione e inoltrarli nei fragment senza dover attendere nuovi dati da parte di IndoorAtlas. Si potrebbe dire che svolgono una funzione di caching per ottimizzare i tempi.

ControllerActivity

La classe `ControllerActivity` ha una funzione strutturale.

Istanzia e controlla gli oggetti legati alla GUI[3].

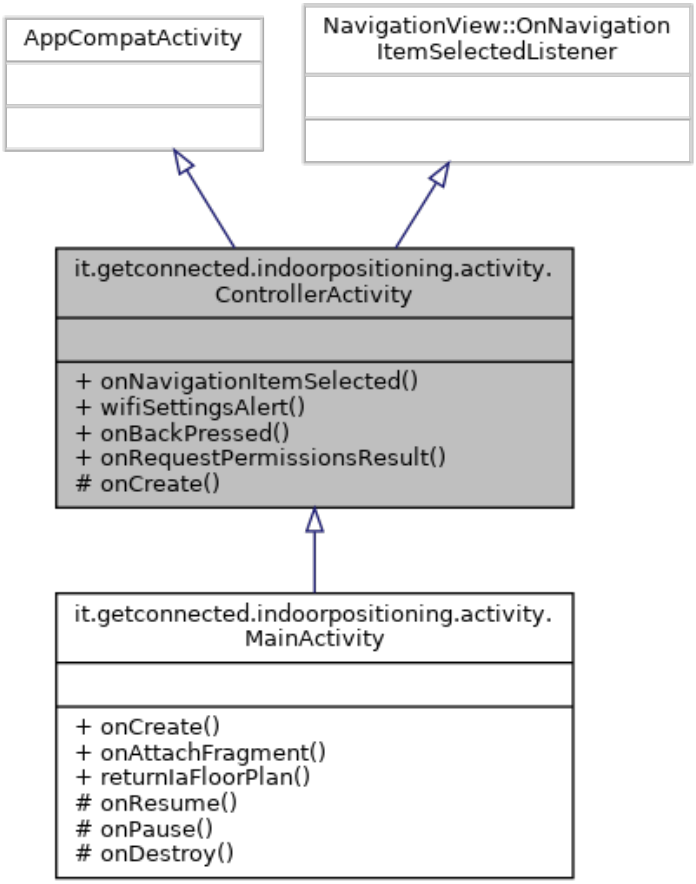


Figura 9.2: ControllerActivity

I metodi più interessanti di questa classe sono:

- `locationSettingsAlert();`
- `wifiSettingsAlert();`

Questi due metodi controllano che i servizi di geolocalizzazione e il Wi-Fi siano attivi; se non lo fossero invitano l'utente ad attivarli visualizzando a schermo un messaggio.

[3]**Graphical User Interface**: interfaccia che consente all'utente di interagire con lo smartphone controllando oggetti grafici

Per ottenere i permessi di scrittura sul *filesystem* e di utilizzo dei servizi di geolocalizzazione in dispositivi con un sistema Android ≥ 6.0 è necessario utilizzare il metodo `onRequestPermissionsResult(...)` che controlla i permessi di cui gode l'app.

Se i permessi richiesti non fossero presenti si avrebbe la terminazione dell'app.

Per ottenere i permessi da parte dell'utente è stato creato il metodo `ensurePermissions()` il quale visualizza dei messaggi, invitando l'utente ad accettare i permessi e, in caso positivo, salvarli.

Constants

La classe `Constants` raccoglie le costanti utilizzate nel progetto.

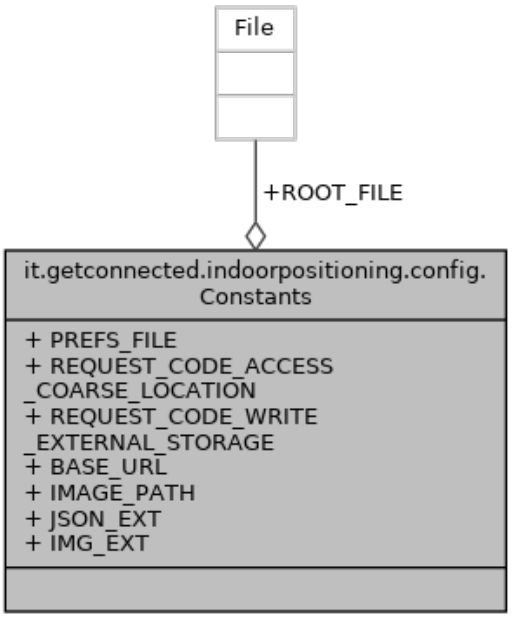


Figura 9.3: Constants

- **`PREFS_FILE`**: è il nome dell'archivio utilizzato dalle `SharedPreferences`. Serve a conservare impostazioni locali dell'app.
- **`REQUEST_CODE_ACCESS_COARSE_LOCATION`**: costante usata per ottenere i permessi nell'utilizzo del servizio di localizzazione.
- **`REQUEST_CODE_WRITE_EXTERNAL_STORAGE`**: costante utile a ottenere i permessi di scrittura sulla memoria dello smartphone. Senza questo permesso non sarebbe stato possibile salvare le immagini delle planimetrie.

9.2 Servizio per la geolocalizzazione

LocationService

La classe `LocationService` estende `Service`, oggetto Android che permette di eseguire lunghi *task* in parallelo senza bloccare la *view* corrente.

In questo progetto `LocationService` è un servizio che si allaccia alla SDK di IndoorAtlas istanziando agli oggetti `IALocationManager` e `IAResourceManager`.

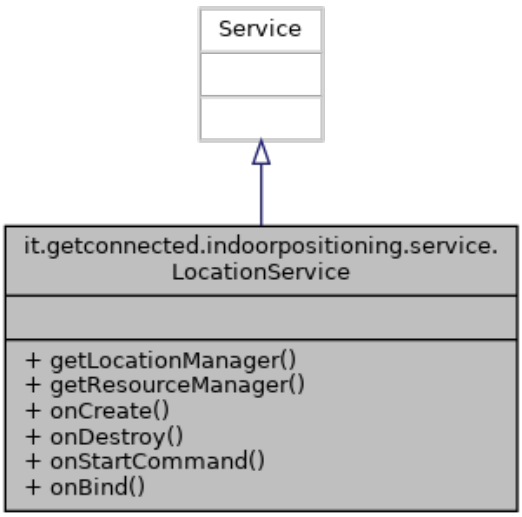


Figura 9.4: LocationService

+ onCreate()

Nel metodo `onCreate()` si istanziano gli oggetti `IALocationManager` e `IAResourceManager`

```
...
mLocationManager = IALocationManager.create(this);
mResourceManager = IAResourceManager.create(this);
...
```

+ onDestroy()

In `onDestroy()` si termina il collegamento al servizio di localizzazione.

```
...
mLocationManager.destroy();
...
```

MyBinder

`MyBinder` estende l'oggetto `Binder`.

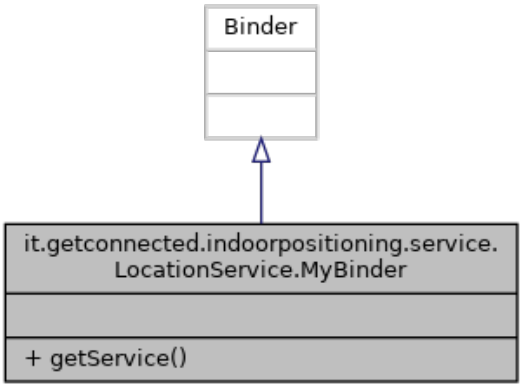


Figura 9.5: MyBinder

`+ getService()`

Il metodo `getService()` ritorna indietro l'oggetto `LocationService`.

Questo permette a `MainActivity` di ricevere i dati sulla localizzazione che il servizio `LocationService` ottiene.

9.3 View della planimetria

IndoorMapImageView

La classe `IndoorMapImageView` permette di visualizzare in un oggetto grafico `FrameLayout` la planimetria del posto in cui l'utente è ubicato.

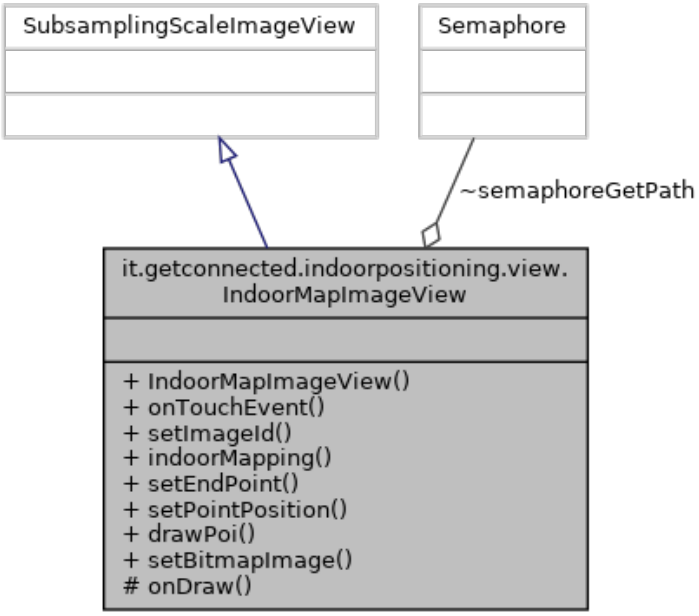


Figura 9.6: IndoorMapImageView

Questa classe estende `SubsamplingScaleImageView`, oggetto che a sua volta estende `View`.
L'oggetto `SubsamplingScaleImageView` non è un oggetto standard Android. Per poterlo usare è stato necessario aggiungere la libreria `subsampling-scale-image-view` nel file `buid.gradle` aggiungendo la stringa:

```
...
compile 'com.davemorrissey.labs:subsampling-scale-image-view:3.6.0'
...
```

9.4 Algoritmo per il calcolo del percorso minimo

PathFinding

`PathFinding` è la classe che esegue l'algoritmo A* (sez. 6.2) per il calcolo del percorso minimo.

Essenzialmente `PathFinding` è un *thread* che resta in attesa fino a quando non riceve una *notify* da parte di `IndoorMapImageView`.

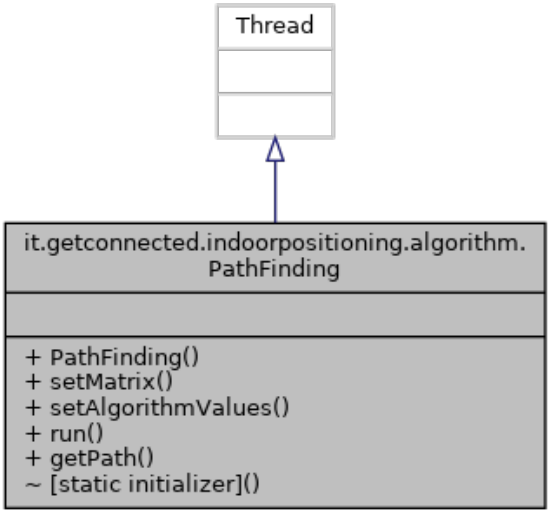


Figura 9.7: PathFinding

In questa classe è presente la classe privata `NodeScoring` che permette di ordinare i nodi del percorso minimo in base al punteggio a loro assegnato. Il punteggio finale dipende dalla direzione che bisogna prendere per raggiungere il nodo successivo.

+ run()

Il metodo `run()` esegue l'algoritmo A* (sez. 6.2) in base ai parametri passati dalla classe `IndoorMapImageView`.

```
while (true) {
    synchronized (this) {
        try {
            wait();
            this.semaphoreDrawPath.acquire();
        } catch (InterruptedException e) {
            Log.d(TAG, e.getMessage());
        }
    }
```

```
    initialize();
    findPath();
    reset();
    setCurrentPath();

    handler.sendEmptyMessage(0);
}
```

Il semaforo `semaphoreDrawPath` evita che ci sia una corsa critica in caso la posizione dell'utente cambi troppo velocemente e il calcolo del percorso minimo non sia ancora terminato.

+ setAlgorithmValues(...)

In metodo `setAlgorithmValues(...)` viene richiamato ogni qual volta si selezioni un nuovo POI o quando viene cambiata posizione. Dalla classe `IndoorMapImageView` riceve i parametri:

- `start`: coordinate del punto in cui si trova l'utente;
- `finish`: coordinate del POI target;
- `mapWidth` e `mapHeight`: larghezza e altezza dell'immagine;

+ setMatrix(...)

In metodo `setMatrix(...)` setta la *matrice dei punti navigabili* ricevuta dalla `IndoorMapImageView`.

+ findPath()

Il metodo `findPath()` esegue un ciclo `while(true)` che si blocca se si raggiunge l'ultimo nodo (punto di arrivo).

```
Point bestDirection = start;
int nextDirectionX, nextDirectionY;
Node nextNode;

while (true) {

    if (lastNode != null) {
        break;
    }

    for (int direction = 0; direction < matrixDirection.length;
        direction++) {

        nextDirectionX = matrixDirection[direction][0] + bestDirection.x;
        nextDirectionY = matrixDirection[direction][1] + bestDirection.y;
```

```
        if (nextDirectionX > 0 && nextDirectionX < mapWidth
            && nextDirectionY > 0 && nextDirectionY < mapHeight
            && matrixPointsWalkable[nextDirectionX][nextDirectionY]
            && !matrixPointsPassed[nextDirectionX][nextDirectionY]) {

            matrixPointsPassed[nextDirectionX][nextDirectionY] = true;

            nextNode = createNode(new Point(nextDirectionX,
                nextDirectionY),
            bestDirection, direction);

            nodePointers[nextDirectionX][nextDirectionY] = nextNode;

            nodePointers[bestDirection.x][bestDirection.y]
                .getChild()[direction] = nextNode;

            bestScoreQueue.add(new NodeScoring(nextNode,
                nextNode.getHeuristicScore()));
        }
    }

    if (lastNode != nodePointers[finish.x][finish.y]) {
        lastNode = nodePointers[finish.x][finish.y];
    }

    bestScoreNode = bestScoreQueue.poll();
    if (bestScoreNode == null) {
        break;
    }
    bestDirection = bestScoreNode;
}
```

Ad ogni ciclo viene creato un nuovo nodo a cui è assegnato un peso attraverso la funzione euristica (sez. 6.2.2).

Al termine del ciclo si ottiene il percorso minimo che va dal nodo di partenza (posizione dell'utente) al nodo finale (nodo goal che in questo caso è la posizione del POI).

9.5 Fragment

Tutte le seguenti classi dal punto di vista strutturale sono simili in quanto:

1. estendono la classe `Fragment`;
2. implementano l'interfaccia `OnFragmentInteractionListener` presente nella classe `MainActivity`;
3. hanno un metodo statico `newInstance(...)` che permette di istanziare la classe passando dei parametri.
4. hanno un metodo `onCreateView(...)` per la definizione del layout (richiamando i vari `FrameLayout`) su cui lavorerà il fragment.

L'interfaccia `OnFragmentInteractionListener` permette ai vari fragment di ricevere (indirettamente) informazioni sulla posizione da parte del servizio `LocationService`.

I fragment realizzati sono:

- `IndoorMapFragment` (9.5): per interagire con l'immagine della planimetria;
- `SearchPoiFragment` (9.5): per poter ricercare i POI del piano;
- `PoiManagerFragment` (9.5): per creare nuovi POI.

IndoorMapFragment

La classe `IndoorMapFragment` istanzia l'oggetto `IndoorMapImageView` in modo da far visualizzare la planimetria al fine di far interagire l'utente col sistema.

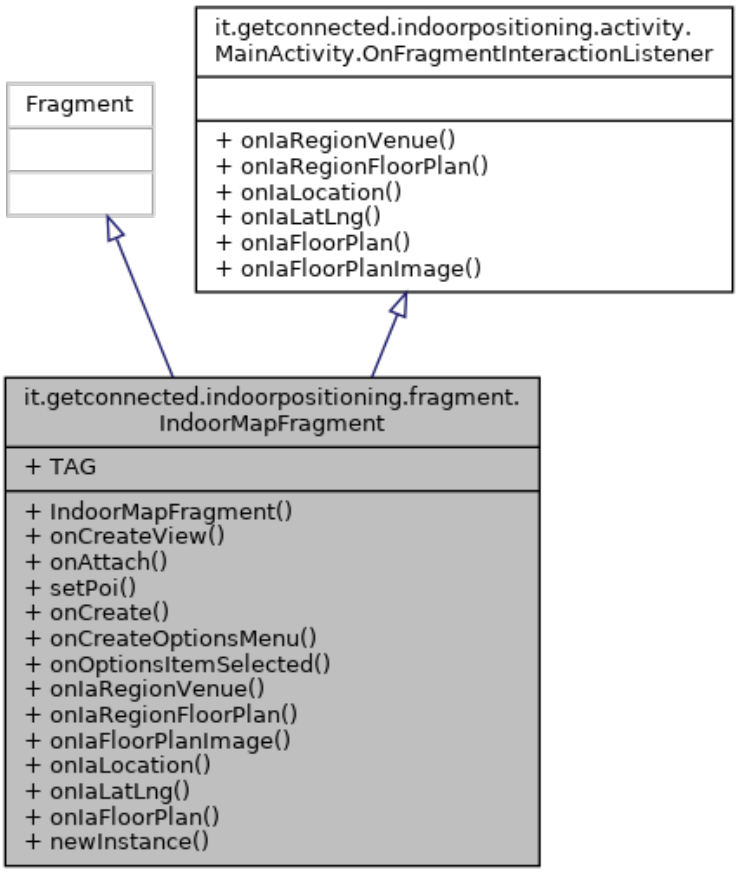


Figura 9.8: IndoorMapFragment

In questa classe si attiva il menù di debug con cui è possibile scegliere di visualizzare i punti navigabili (colorati di verde) o gli ostacoli (di color magenta).

Per consentire la rotazione dell'immagine della planimetria è stato aggiunto un FloatingActionButton che ad ogni click modifica l'angolo della *view* di 90°.

SearchPoiFragment

La classe `SearchPoiFragment` permette all'utente:

- di visualizzare la lista POI del piano in cui si trova;
- di effettuare la ricerca del POI target;
- in generale di effettuare la selezione del POI target cliccando sul relativo bottone presente in ogni riga della lista.

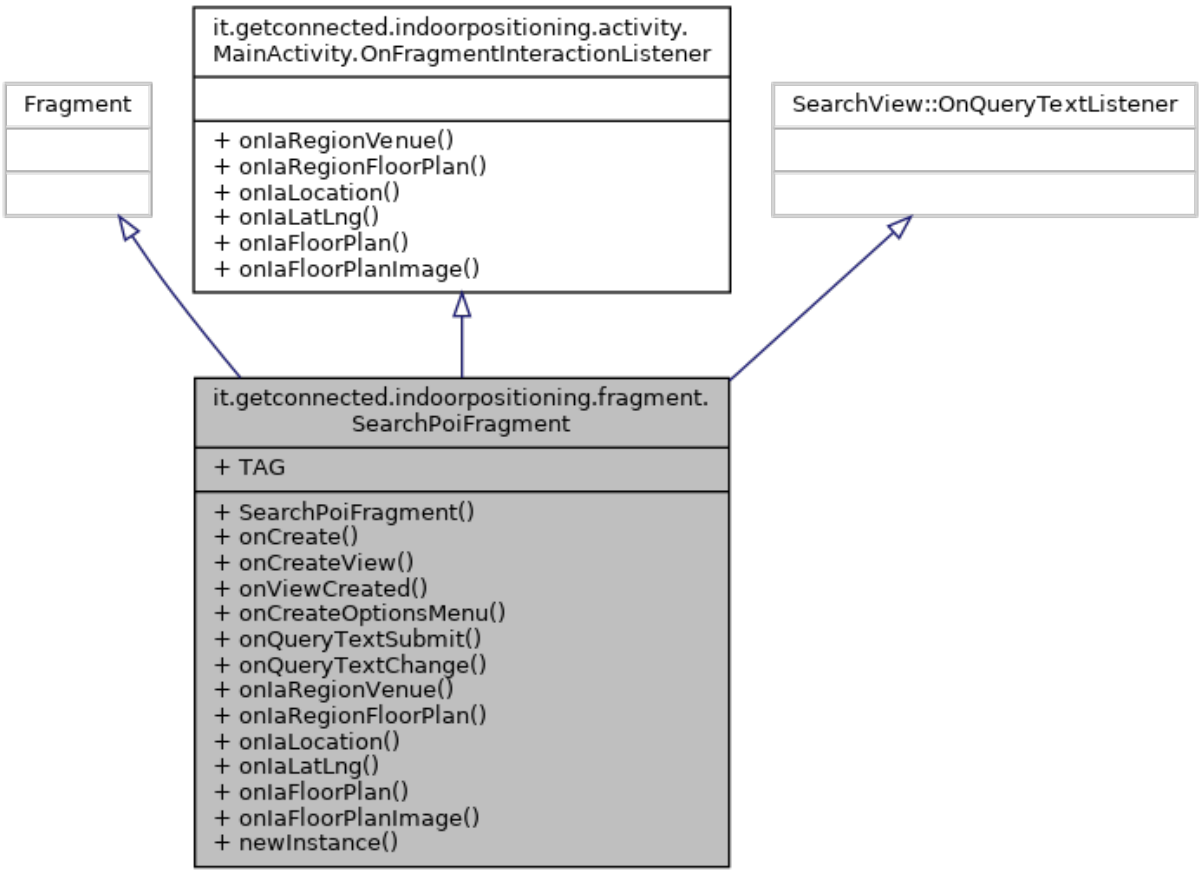


Figura 9.9: SearchPoiFragment

Per permettere la ricerca dei POI la classe `SearchPoiFragment` implementa i metodi `onQueryTextSubmit` e `onQueryTextChange` dell'interfaccia *SearchView*.

L'utilizzo è abbastanza semplice, l'utente inserisce una stringa nella barra di ricerca posta in alto e un filtro visualizza in una lista sottostante tutti i POI che hanno una corrispondenza anche solo parziale. Se non esistono corrispondenze la lista dei POI selezionabili sarà vuota.

SearchPoiAdapter

La classe `SearchPoiAdapter` permette di visualizzare i POI ricevuti dal server all'interno di oggetti `CardView`.

Per ottenere una lista di CardView ordinate e selezionabili si è estesa la classe `RecyclerView.Adapter<SearchPoiAdapter.ItemViewHolder>`.

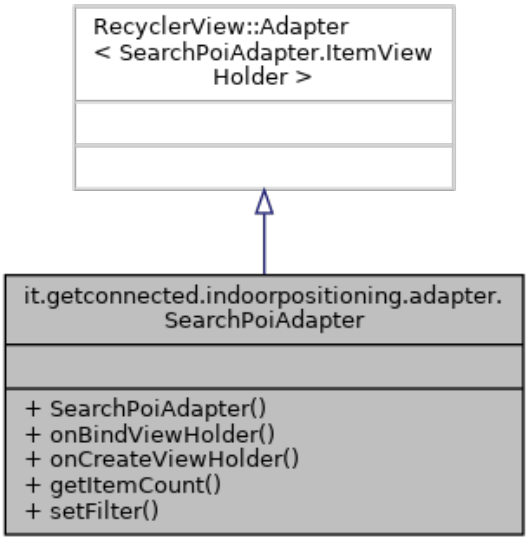


Figura 9.10: SearchPoiAdapter

In particolare della classe si sottolinea l'utilizzo di un oggetto `ImageButton` e del suo metodo `setOnClickListener(...)`. Questo oggetto è presente in ogni riga della lista-POI; se cliccato permette la navigazione dal punto in cui si trova l'utente al punto indicato dalle coordinate del POI selezionato.

Se questo bottone viene selezionato con un click:

- si passa all fragment `IndoorMapFragment` il quale è deputato alla visualizzazione della planimetria;
- si visualizza uno `Snackbar` che indica all'utente che si sta eseguendo la navigazione verso il POI target;
- si visualizza il percorso minimo che porta dall'utente al POI.

```
final ImageButton imageButton = holder.imageButton;
imageButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
```

```
        IndoorMapFragment fragment =
            IndoorMapFragment.newInstance("image_view");
        activity.getSupportFragmentManager().beginTransaction()
                .replace(R.id.contentMain,fragment,IndoorMapFragment.TAG)
                .commit();
        fragment.setPoi(poi);

        Snackbar.make(v, "Navigation to " + poi.getInfo() + " ...",
            Snackbar.LENGTH_SHORT).show();
    }
});
```

PoiManagerFragment

La classe `PoiManagerFragment` è legata alla parte amministrativa dell'app. Permette il salvataggio di POI, cioè la geolocalizzazione di un'area in uno spazio limitato. Il nuovo POI sarà inviato al server in modo da essere visualizzabile a tutti gli utenti che usufruiranno del servizio di IPS.

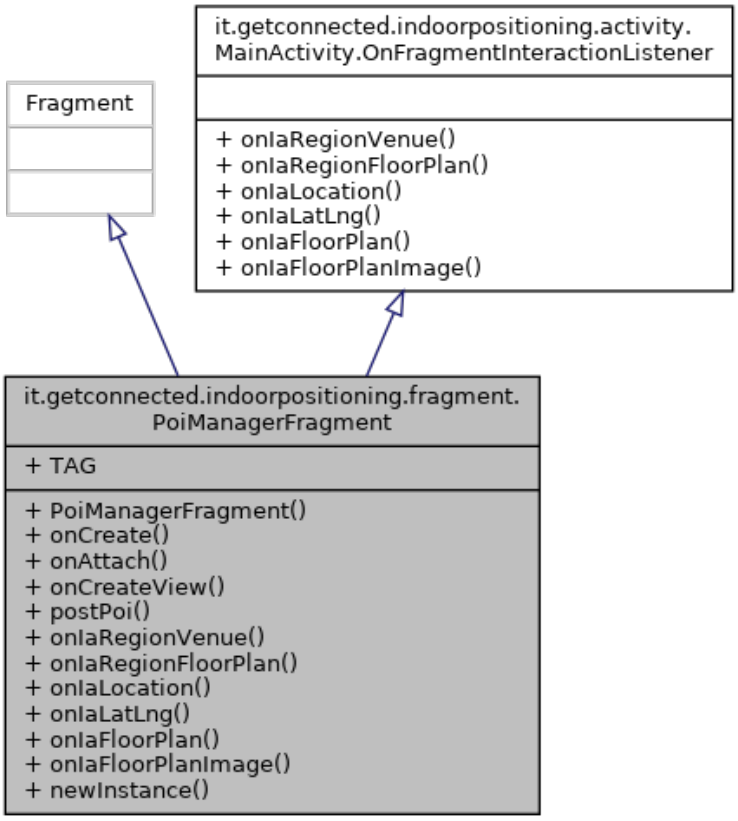


Figura 9.11: PoiManagerFragment

La creazione e poi l'invio del POI verso il server viene eseguita in seguito al click del `FloatingActionButton`, ma solo se sono presenti tutte le relative informazioni. In caso contrario si visualizzerà uno `SnackBar` per informare della mancanza di un dato in ingresso.

9.6 Modellazione del dominio applicativo

Per poter modellare il dominio applicativo sono stati creati degli oggetti come astrazione della realtà. Questi oggetti sono stati funzionali per lo sviluppo del progetto.

Circle

La modellazione dell'oggetto cerchio è stata realizzata con la classe `Circle`.

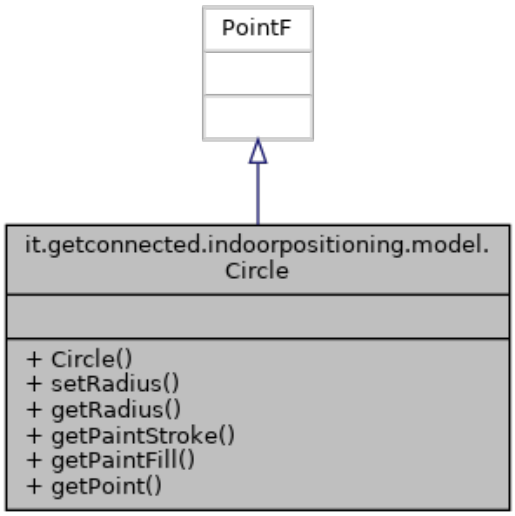


Figura 9.12: Circle

In questo progetto i cerchi sono stati utilizzati:

- In fase di debug, per disegnare:
  - i punti percorribili della planimetria
  - gli ostacoli della planimetria.
- In fase di utilizzo da parte dell'utente, per disegnare:
  - l'ubicazione dell'utente all'interno della planimetria;
  - la posizione del POI target;
  - il percorso minimo utente-POI.

Node

La modellazione dell'oggetto nodo, utile all'algoritmo A* (sez. 6.2) per il calcolo del percorso minimo, è stata realizzata con la classe `Node`.

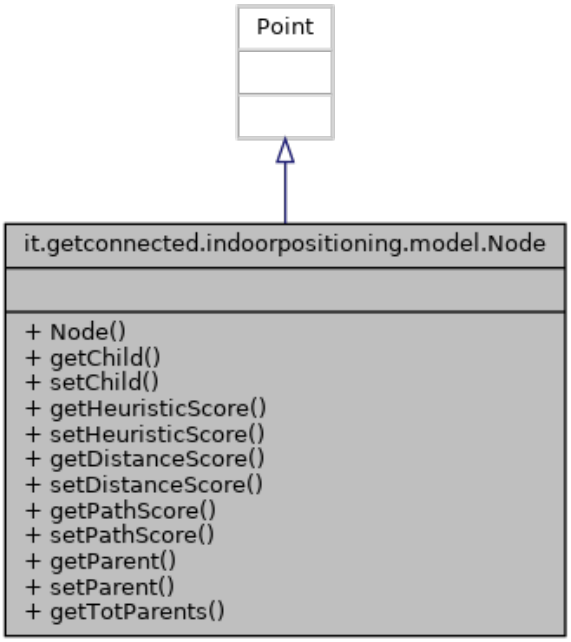


Figura 9.13: Node

In questo progetto un nodo è un oggetto che:

- estende la classe `Point`;
- può contenere al suo interno informazioni sui nodi vicini, detti *nodi child*;
- permette di salvare al proprio interno il proprio valore di score (ottenuto dalla funzione euristica) e quello dei nodi child.

Poi

La modellazione dell'oggetto POI è stata realizzata con la classe `Poi`.

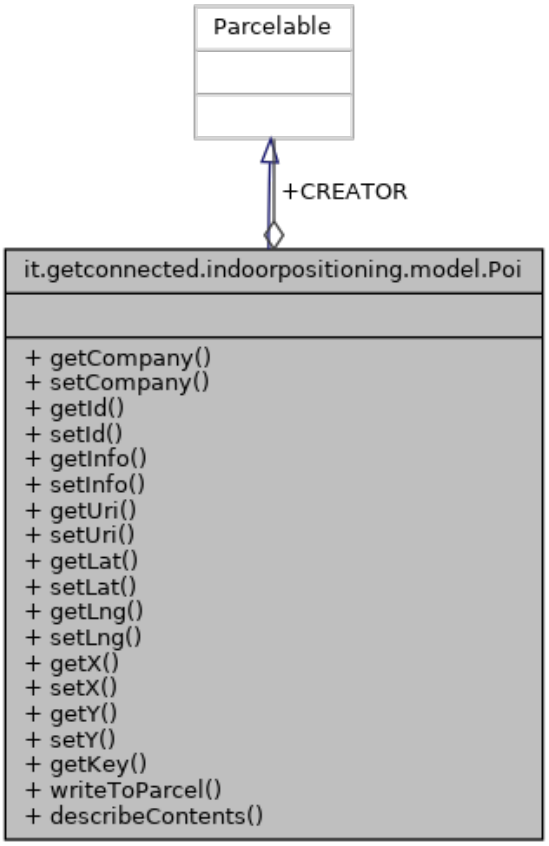


Figura 9.14: Poi

Un oggetto `Poi` è stato pensato come un punto (presente nel mondo reale e nella planimetria visualizzabile nell'app) a cui sono connesse informazioni descrittive utili all'utente.

Di seguito la descrizioni delle informazioni che contraddistinguono un oggetto `Poi`:

- **`company`**: marca di un prodotto o nome di una attività commerciale;
- **`id`**; codice identificativo del POI;
- **`info`**: descrizione sommaria del POI;
- **`uri`**: indirizzo che fa riferimento all'immagine del POI;
- **`lat`**: latitudine reale;
- **`lng`**: longitudine reale;

- **x**: coordinata x nella planimetria;
- **y**: coordinata y nella planimetria;
- **key**: parametro per distinguere due POI simili.

9.7 Gestione della connessione al server

RestManager

La classe `RestManager` permette di utilizzare l'oggetto `Retrofit` della libreria `Retrofit` al fine di effettuare chiamate GET e POST verso il server in cui sono salvati i POI dei vari edifici.

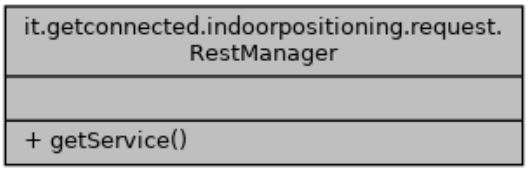


Figura 9.15: RestManager

`RestManager` richiama la classe dell'interfaccia `RestService` al fine di specificare più metodi HTTP richiamabili come delle *callback* all'interno del codice. Questo permette di avere un codice più parametrizzato.

Esempio di definizione di chiamata POST all'interno di `RestService`:

```
@POST("/addpoi")
Call<Poi> addPoi(@Body Poi poi);
```

Esempio di implementazione della callback:

```
RestManager.getService().addPoi(poi).enqueue(new Callback<Poi>() {
    @Override
    public void onResponse(Call<Poi> call, Response<Poi> response) {
        Log.i("onResponse", response.isSuccessful() + "");
    }

    @Override
    public void onFailure(Call<Poi> call, Throwable t) {
        Log.i("onFailure", t.getMessage());
    }
});
```

9.8 Filtri immagine

BinaryFilter

La classe `BinaryFilter` raccoglie dentro di sé i filtri immagine che sono stati impiegati per elaborare l'immagine della planimetria.

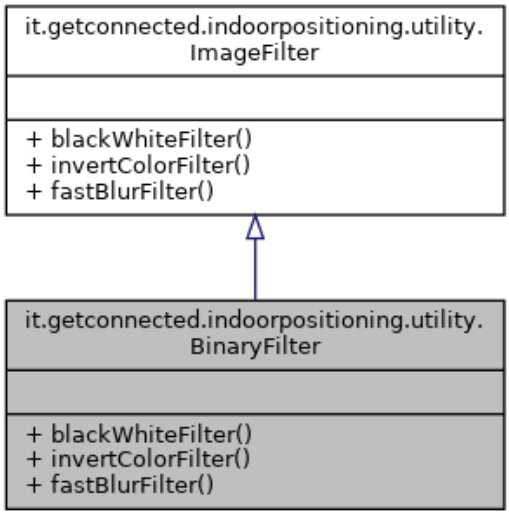


Figura 9.16: BinaryFilter

`BinaryFilter` implementa l'interfaccia ImageFilter definendo i metodi `blackWhiteFilter(...)` `invertColorFilter(...)` `fastBlurFilter(...)`

Capitolo 10

Comunicazione col server

In questo progetto il server essenzialmente svolge la funziona di database remoto accessibile online. La funzionalità a cui deve rispondere è permettere il download della lista dei POI.

Tale download avviene in modo trasparente all'utente ogni qual volta viene identificato all'interno dell'edificio. Come ottimizzazione si può pensare di evitare il download se l'utente in medesimo giorno esce e poi rientra nello stesso edificio. Ovviamente per effettuare il download della lista-POI lo smartphone deve disporre di una connessione Internet. Si può quindi immaginare un centro commerciale in cui è presente una rete Wi-Fi free a cui l'utente può connettersi o anche il semplice utilizzo della connessione dati mobile.

Dal punto di vista software il server si affida ad una WebApp Node.js la quale risponde alle chiamate HTTP inviando o ricevendo dei file in formato JSON.

10.1 Download della lista POI

Per il download della lista si è utilizzata la tecnica AJAX in modo asincrono, cioè utilizzando la libreria Retrofit combinata a degli *AsyncTask*. La lista è stata pacchettizzata come file JSON in modo da rispettare le specifiche AJAX.

JSON

JSON (JavaScript Object Notation) è un formato per lo scambio di dati "semplice":

- Per le persone perché facile da leggere e scrivere.
- Per le macchine perché facile da generare e analizzarne la sintassi.

JSON è basato su due strutture:

- Un insieme di coppie nome/valore. In Java questo è realizzato come un oggetto POJO.
- Un elenco ordinato di valori. In java questo si realizza con un array di oggetti.

Esempio di file JSON

```
{
    "company": "nomeCompany",
    "id": "000001",
    "info": "descrizione info",
    "lat": 44.14258480730086,
    "lng": 12.256917465085975,
    "uri": "/img/000001.png",
    "x": 213,
    "y": 89
}
```

POJO

Nell'ingegneria del software, **POJO** è un acronimo di **Plain Old Java Object**. Il nome è usato per accentuare che un oggetto dato è un oggetto ordinario Java. Il termine fu coniato da Martin Fowler, Rebecca Parsons e Josh MacKenzie nel settembre 2000.

Un POJO è un oggetto Java non legato ad alcuna restrizione diversa da quelle costrette dalla specifica del linguaggio Java (Java Language Specification). In altre parole, è imperativo che un POJO:

- non estenda delle classi pre-specificate;
- non implementi delle interfacce prespecificate;
- non contenga delle annotazioni prespecificate.

10.1.1 Convertire JSON in POJO

Per poter convertire i dati JSON restituiti dal server in un oggetto Java è necessario:

- creare un modello POJO dell'oggetto da restituire;
- utilizzare un JSON parser (ad esempio Gson).

I nomi dei campi del POJO devono essere gli stessi del JSON a meno che non si usi l'annotazione `@SerializedName` che permette di specificare un nome particolare per il campo desiderato. Per la serializzazione/deserializzazione si possono usare due tipi di classi: `Serializable` and `Parcelable`.

Mettendo a confronto le due soluzioni:

- `Parcelable`: garantisce prestazioni migliori avendo una *reflection* anche 10 volte più rapida di `Serializable` (http://www.developerphil.com/parcelable-vs-serializable/). Di contro implica la scrittura di una notevole quantità di *boilerplate code* (codice standard).

- `Serializable`: un'interfaccia standard Java più facile da implementare. La reflection risulta più lenta in quanto vengono creati molti oggetti temporanei che devo poi essere gestiti dal *garbage collection*

Jsonschema2pojo

Grazie a ***Jsonschema2pojo*** http://www.jsonschema2pojo.org/ è possibile generare automaticamente POJO partendo da *JSON-Schema* o JSON, rendendo la conversione JSON-POJO molto più semplice.

10.1.2 Librerie JSON per Android

Android integra nativamente la possibilità di utilizzare oggetti JSON, ma il loro utilizzo è piuttosto complesso. Per questo sono nate diverse soluzioni disponibili come librerie che si integrano perfettamente nelle app.

In questo progetto le librerie utilizzato sono Gson e Jackson. La prima è più adatta alle conversione POJO-Json, la seconda è più adatta al salvataggio di dati locali.

Gson

Gson è una libreria Java sviluppata da Google che permette:

- di cenvertire oggetti Java nella rappresentazione JSON;
- di effettuare il parsing di una stringa JSON in un oggetto Java equivalente.

Gson può lavorare con oggetti Java arbitrari, compresi quelli di cui non si ha il codice sorgente.

Per aggiungere la libreria Gson al progetto si è inserita la seguente dipendenza nel file *gradle*:

```
dependencies {
    ...
    compile 'com.google.code.gson:gson:2.8.0'
    ...
}
```

Jackson

Jackson (https://github.com/FasterXML/jackson) è una suite di strumenti *multi-purpose* per l'elaborazione di dati in formato Java e JSON. Si propone come la migliore combinazione di velocità, correttezza, leggerezza ed ergonomia per gli sviluppatori.

Comprende:

- una libreria di streaming parsing e generazione di JSON;
- una libreria per il data-binding (da POJO a JSON e viceversa);
- moduli per supportare formati di dati aggiuntivi ampiamente utilizzati quali Guava, Joda, PCollections e molti altri.

Per aggiungere Jackson al progetto è bastato inserire le seguenti dipendenze nel file *gradle*:

```
dependencies {
    ...
    compile 'com.fasterxml.jackson.core:jackson-core:2.8.6'
    compile 'com.fasterxml.jackson.core:jackson-annotations:2.8.6'
    compile 'com.fasterxml.jackson.core:jackson-databind:2.8.6'
    ...
}
```

10.2 Retrofit

Retrofit è una libreria *type-safe* creata da **Square Open Source** (`http://square.github.io/`) che permette di realizzare client REST in Android e Java in grado di interfacciarsi con le API REST dei vari webservices.

In questo progetto Retrofit è stato utilizzato per l'invio e la ricezione di dati in formato JSON da e verso il server. In particolare è stato utile per l'invio al server di nuovi POI e per il download della lista POI dell'edificio.

Per aggiungere Retrofit al progetto si sono aggiunte le seguenti dipendenze nel file *gradle*:

```
dependencies {
    ...
    compile 'com.squareup.retrofit2:retrofit:2.1.0'
    compile 'com.squareup.retrofit2:converter-gson:2.1.0'
    ...
}
```

Retrofit è stata scelta come libreria da aggiungere al progetto perché permette di inviare e ricevere dati via internet utilizzando delle callback. Queste callback sono in grado di eseguire conversioni POJO-JSON e viceversa a patto che il modello POJO esista.

Per impostazione predefinita Retrofit può solo eseguire la deserializzazione di *HTTP bodies* nel tipo `ResponseBody` di *OkHttp* ed accettare tipi `RequestBody` con la clausola `@Body`.

Per utilizzare altri convertitori esistono sei librerie facilmente integrabili:

- **Gson**: com.squareup.retrofit2:converter-gson;
- **Jackson**: com.squareup.retrofit2:converter-jackson;
- **Moshi**: com.squareup.retrofit2:converter-moshi;
- **Protobuf**: com.squareup.retrofit2:converter-protobuf;
- **Wire**: com.squareup.retrofit2:converter-wire;
- **Simple** XML: com.squareup.retrofit2:converter-simplexml;
- **Scalars** (primitives, boxed, e String): com.squareup.retrofit2:converter-scalars;

Al fine di poter utilizzare Retrofit in questo progetto si è dovuto impostato l'URL dell'*endpoint* di base in modo che fosse disponibile a tutte le chiamate API. Questo URL è l'indirizzo internet del server.

Il settaggio iniziale di Retrofit è stato fatto nella classe `RestManager` (sez. 9.7), dove si è anche impostata la conversione Gson come illustrato nel seguente frammento:

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl(Constants.BASE_URL)
    .addConverterFactory(GsonConverterFactory.create())
    .build();
```

Per effettuare una chiamata API con Retrofit invece si sono creati dei metodi nell'interfaccia `RestService` (sez. 9.7) in modo da ottenerne una sostituzione in grado di ritornare un *Java model object* in caso di ricezione di dati JSON.

La scelta architetturale di utilizzare interfacce statiche permette di sostituire l'endpoint delle path al momento della chiamata API utilizzando le variabili POST/GET, ecc. . . , rendendo tutto più pulito e dinamico.

Esempio di chiamata API

```
RestManager.getService()
    .getPoi(iaRegionVenue.getId(), iaRegionFloorPlan.getId() + JSON_EXT)
    .enqueue(new Callback<List<Poi>>() {
        @Override
        public void onResponse(Call<List<Poi>> call, Response<List<Poi>>
            response) {
            Log.i(TAG, "onResponse " + response.isSuccessful());
```

```
            if (response.isSuccessful()) {
                List<Poi> poiList = response.body();
                if (!poiList.isEmpty()) {
                    indoorMapImageView.drawPoi(poiList);
                }
            }
        }

        @Override
        public void onFailure(Call<List<Poi>> call, Throwable t) {
            Log.i(TAG, "onFailure " + t.getMessage());
        }
    });
```

In questo frammento di codice si crea una callback che in base all'id dell'edificio e del piano richiede la lista dei POI.

Se la richiesta avrà esito positivo si riceverà un file JSON che sarà analizzato e convertito da JSON in POJO dalla libreria Gson. In questo caso la conversione permette di popolare un oggetto `List` composto da oggetti `Poi` (sez. 9.6).

Capitolo 11

Testing

11.1 Testing dell'IPS realizzato

La fase di testing serve a verificare che gli obiettivi di progetto siano stati raggiunti e a definire le performance del sistema.

I test sono stati eseguiti:

- sempre nello stesso edificio;
- con lo stesso smartphone;
- con una connessione Wi-Fi attiva;
- attivando precedentemente all'avvio dell'app i servizi Wi-Fi e di geolocalizzazione;
- con la medesima versione dell'app.

Questo per evitare di falsare i risultati ottenuti.

I test dell'IPS proposto sono serviti a riconoscere alcuni bug del sistema (prontamente risolti), immedesimandosi nell'utente.

11.1.1 Visualizzazione della posizione dell'utente

In questo primo test si è verificato che l'app visualizzi correttamente la posizione dell'utente all'interno dell'edificio.

Passi di test:

1. avvio dell'app;
2. attesa del riconoscimento dell'edificio e del piano (tempo variabile 2-6 sec);
3. download, analisi e visualizzazione della planimetria (< 1 sec);

4. determinazione delle coordinate dell'utente e spostamento del punto rosso *indicativo dell'utente* (circa 2 sec).

L'obiettivo di questo test era verificare che l'app ricevesse correttamente i dati dal servizio e muovesse il punto (astrazione della posizione occupata dall'utente) nel caso in cui l'utente fosse in movimento.

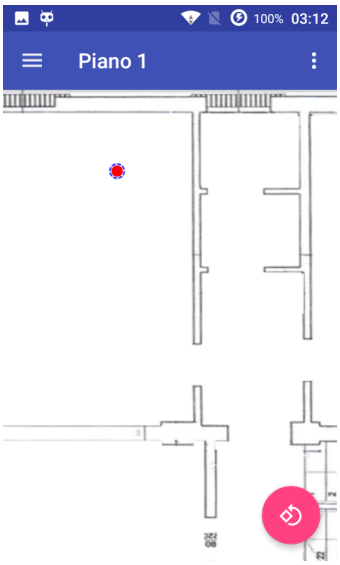


Figura 11.1: Posizione dell'utente nell'edificio

Da come si può vedere nell'immagine 11.1 il test è terminato positivamente, cioè ha visualizzato l'immagine correttamente e disposto il cerchio nella prossimità della posizione realmente occupata dal tester.

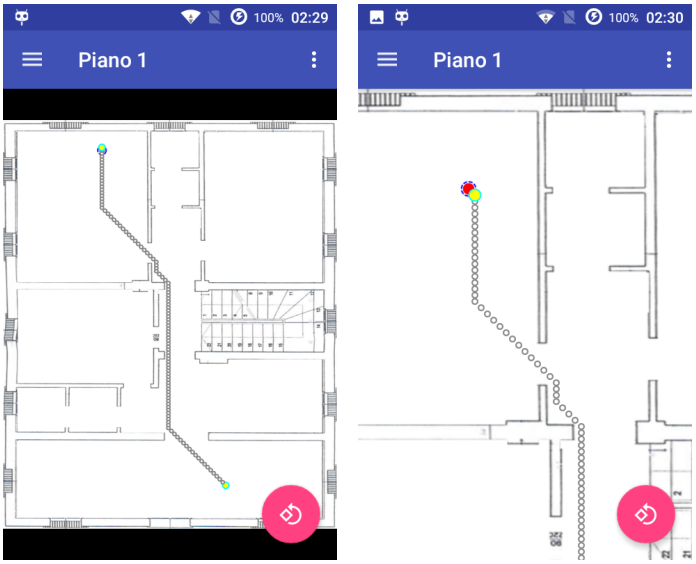
Questo test ha quindi dimostrato che il servizio IPS progettato è realmente in grado di geolocalizzare gli utenti in ambienti chiusi utilizzando la tecnica del fingerprint magnetico unito alle sensor fusion.

11.1.2 Visualizzazione del percorso minimo

Il secondo testing ha verificato la possibilità di ottenere e visualizzare il percorso minimo che permetta all'utente di raggiungere un POI target.

In questo test si è:

1. atteso il download e il caricamento della lista dei POI disponibili per il piano (< 1 sec);
2. selezionato un POI;
3. verificato che il percorso minino fosse visibile;
4. verificato che la posizione del POI corrispondesse con quella precedentemente salvata sul server;
5. navigato verso il POI per vedere il percorso diminuire durante il tragitto (in tempo reale);
6. cambiato strada volutamente per vedere il ricalcolo del percorso minimo (in tempo reale).



(a) Visualizzazione del percorso minimo

(b) Zoom del percorso minimo

Figura 11.2: Navigazione nell'ambiente indoor

Anche questo test è stato superato in quanto tutti i punti precedenti hanno avuto esito positivo. La cosa più importante di questo test ovviamente è il calcolo del percorso minimo in tempo reale.

La scelta del ricalcolo ad ogni movimento dell'utente a lungo andare potrebbe essere penalizzante per il consumo della batteria o comunque usurare il processore. Durante il test non si sono evidenziati dei cali eccessivi di batteria.

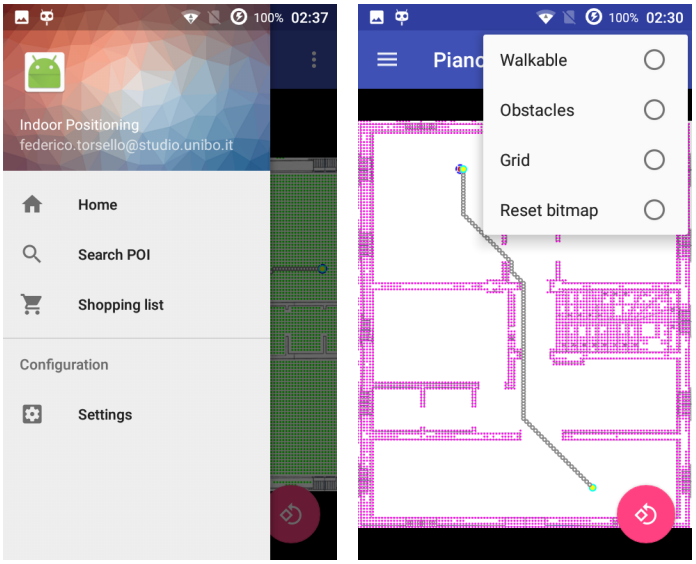
Questo potrebbe essere un punto da approfondire nel caso in cui il progetto dovesse avere una evoluzione commerciale.

11.1.3 Menù laterale e di debug

In questo semplice test si è verificato che i vari menù funzionassero senza problemi, reagendo al click dell'utente, rimpiazzando la view senza avere crash o ritardi di visualizzazione.

Nel dettaglio si è verificato che il passaggio da un fragment all'altro non incidesse sulla qualità della accuratezza del posizionamento. Anche se un fragment viene ricaricato la posizione dell'utente viene visualizzata in 1-2 sec.

Questo test è propedeutico al successivo.



(a) Visualizzazione del menù laterale

(b) Visualizzazione del menù di debug

Figura 11.3: Menù laterale e di debug

11.1.4 Matrice dei punti navigabili

In questo test si è verificato la visualizzazione dei punti navigabili e degli ostacoli "creati" dall'analisi dell'immagine della planimetria scaricata.

Questo test riguarda l'area di debug dell'app che serve a far visualizzare al programmatore il risultato del filtro immagine.

(a) Visualizzazione dei punti navigabili

(b) Visualizzazione degli ostacoli

Figura 11.4: Definizione dei punti navigabili e degli ostacoli

In questo test si è analizzato il tempo necessario per visualizzare i vari punti. Nell'immagine considerata il disegno dei punti ha impiegato frazioni di secondo, senza evidenziare errori di riconoscimento. Come si può vedere dalle immagini 11.4a e 11.4b la visualizzazione dei punti navigabili e degli ostacoli ha superato il test.

Un secondo test è stato navigare verso un POI selezionato in modalità debug. Anche in questo caso il test è stato superato senza difficoltà.

11.1.5 Rotazione della planimetria

In questo test si è verificato se la rotazione dell'immagine inficiasse in qualche modo l'usabilità dell'app.

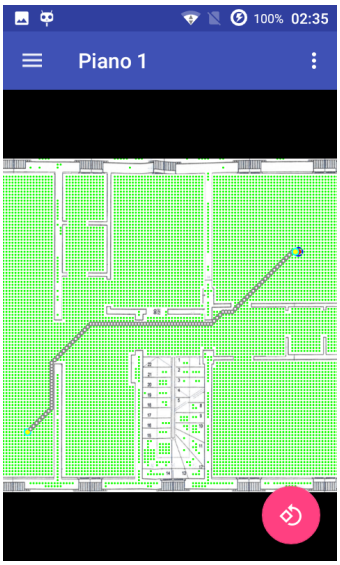


Figura 11.5: Rotazione della planimetria

Da vari test non sono emersi problemi durante la rotazione e a rotazione avvenuta.

I test condotti hanno compreso tutti i precedenti, quindi posizionamento dell'utente, cammino verso un POI e modalità di debug. Anche questo test è stato superato.

11.2 Valutazione delle performance del sistema proposto

La valutazione delle prestazioni dell'IPS proposto si basa sui parametri descritti nella sezione 4.3.

- **Accuratezza**: dalle rilevazioni effettuate non si è riscontrata una grossa discrepanza fra posizione reale e risultato ottenuto. Nei test condotti si è visto che il cerchio rappresentante la posizione dell'utente all'interno della planimetria è abbastanza fedele alla posizione occupata nella realtà.

  Si è però notato che nel caso in cui il segnale Wi-Fi fosse limitato, l'aggiornamento della posizione potrebbe diventare meno istantaneo e si potrebbero verificare dei "salti" del cerchio in posizioni fra loro distanti.

- **Precisione**: il sistema proposto è efficace, in quanto è in grado di operare correttamente nel contesto in cui è inserito. Naturalmente questa efficacia dipende dalla qualità della fingerprint prodotta; cioè dal numero di rilevazioni effettuate e dalla perizia con cui sono state prodotte. Senza una buona fingerprint il sistema funziona, ma non al massimo delle sue potenzialità.

- **Complessità**: il sistema proposto ha una complessità di utilizzo simile a zero, in quanto l'utente non deve far altro che attivare i servizi di geolocalizzazione, il Wi-Fi e una connessione internet per rendere il sistema funzionante.

  Anche dal punto di vista implementativo il sistema non ha particolari problemi. La fase di ricognizione dipende dalla vastità dell'area da mappare, ma il fatto che questa registrazione possa essere fatta in vari *step* e che le registrazioni sono incrementali (senza perdita di dato, bensì con aumento della precisione), rende l'operazione fattibile anche per aree che possono estendersi per molti metri quadrati e su più piani. La particolarità del fingerprint magnetico si non necessitare di dispositivi ad-hoc da comprare ed installare si riflette in una complessità di messa in opera del sistema veramente ridotta se confrontato ad IPS costituiti esclusivamente da ripetitori Wi-Fi.

  Oltre a questo si deve tenere presente che il sistema è in grado di sfruttare la sensor fusion, attingendo alle informazioni captabili da tutti i sensori dello smartphone dell'utente.

  Questo si traduce in una migliore precisione che sfrutta in pieno l'ambiente così com'è. Ad esempio, se nel palazzo sono presenti ripetitori Wi-Fi, allora saranno considerati dal sistema come Anchor, abbattendo i tempi necessari alla prima localizzazione indoor.

  Tutte queste caratteristiche rendono il sistema facilmente replicabile in diversi contesti applicativi, che vanno dall'utenza domestica fino ai centri commerciali o aeroporti, ecc. . .

  – L'analisi della complessità del software si concentra sul costo computazionale, sui tempi di risposta e sull'infrastruttura client-server.

  - Per l'analisi della complessità dell'hardware invece si considerano aspetti come la durata e la qualità dei componenti impiegati.

- **Robustezza e affidabilità**: il sistema proposto sicuramente ha un'alta tolleranza agli errori. Dagli stress test eseguiti non si sono avuti casi in cui il sistema non funzionasse.

  Da questo punto di vista bisogna considerare che il sistema dipende da un servizio remoto, quindi se questo non funzionasse più, automaticamente anche quello proposto non andrebbe. Questo però, entro un certo limite, è lo stesso problema che si potrebbe verificare con il sistema di riferimento GPS.

- **Scalabilità**: il sistema proposto è scalabile, cioè può essere applicato a contesti dinamici. Come limite non può funzionare in ambienti aperti, ma questo dovrebbe essere un falso problema visto che il sistema nasce per geolocalizzare gli utenti esclusivamente in ambienti indoor.

  - Il sistema è scalabile in quanto tutta l'area di interesse, se ha un fingerprint valido, è coperta dal sistema;
  - Il sistema è scalabile anche dal punto di vista dell'utenza e delle richieste di geolocalizzazione. Tutta la parte di calcolo della posizione viene demandata al servizio cloud, il quale per definizione è scalabile.

- **Costo**: il costo di un sistema, a prescindere dalla sua precisione, può determinarne l'applicazione su larga scala. Un costo limitato unito ad un sistema efficace ed efficiente è il massimo che un compratore dovrebbe desiderare, facendo la differenza tra una tecnologia funzionante ma economicamente sconveniente ed una che potrebbe diventare nel tempo la tecnologia de facto per la localizzazione indoor.

  - dal punto di vista dei tempi di progettazione, il sistema progettato è immediatamente replicabile così com'è, quindi questo costo sarebbe più assimilabile a quello che si ha per l'uso di una licenza che per la creazione di un progetto da zero;
  - il sistema non necessita di hardware specifico, quindi non si presenta la necessità di comprare dispositivi, pagare per l'adeguata sistemazione, aggiornamento e manutenzione;
  - la configurazione e il *testing* del progetto potrebbero essere fatte anche solo una volta, al limite con aggiornamenti a distanza di anni per incrementare la precisione della fingerprint;
  - anche il consumo energetico è demandato al cloud, quindi non è direttamente assimilabile alla spesa da affrontare.
  - l'unico costo reale del progetto è l'utilizzo del servizio IndoorAtlas in base all'utenza gestita.

Capitolo 12

Conclusioni

L'obiettivo di questa tesi è stato proporre un sistema di posizionamento indoor applicabile realmente in contesti quotidiani, progettato e realizzato in un'ottica industriale. Per far questo è stato svolto uno studio preliminare delle varie soluzioni commerciali adottabili ed in generale delle tecnologie di indoor positioning disponibili; inoltre sono state effettuate varie interviste presso GetConnected per definire requisiti e prestazioni necessari ad un sistema IPS commerciale.

L'analisi di soluzioni come il GPS o gli IPS basati esclusivamente su segnali radio ha permesso di comprendere meglio ciò che era necessario ottenere da un IPS commerciale, come ad esempio una buona accuratezza di stima della posizione (di circa 2 metri), un'altrettanto buona reattività (nell'ordine dei secondi), da unire ad un interfacciamento utente che ne valorizzi le caratteristiche.

Per determinare la posizione dell'utente si è scelto di utilizzare le tecniche di riconoscimento del fingerprint magnetico in combinazione alla sensor fusion. Il fingerprint magnetico permette di abbattere costi e tempi di realizzazione, permettendo la realizzazione di sistemi scalabili, in condizioni che per altri approcci sarebbero proibitive, in quanto avrebbero bisogno di molti dispositivi e sensori disposti fisicamente in ogni edificio/piano. La sensor fusion invece ha permesso di sfruttare al massimo la sensoristica integrata nello smartphone, rendendolo uno strumento di geolocalizzazione indoor in grado di integrare le informazioni sensoriali percepite da ogni fonte possibile, migliorando la precisione del posizionamento magnetico.

L'utilizzo di queste tecnologie è stato possibile grazie al sistema di geolocalizzazione IndoorAtlas il quale, attraverso una libreria software ed un servizio cloud dedicato, rende lo smartphone dell'utente utile alla localizzazione per interni. La sola geolocalizzazione non è però sufficiente a rendere il sistema "completo", in quanto si è dovuto definire un contesto applicativo che lo renda utile ed utilizzabile all'utente oltre che commercializzabile. Quindi è stato necessario adattare la visualizzazione della posizione dell'utente alla planimetria del piano dell'edificio in cui esso si muove, analizzando i pixel per definire la matrice di punti navigabili in modo automatico, definendo implicitamente gli ostacoli, come ad esempio i muri. A tal fine è stato ideato un piccolo sistema di riconoscimento delle immagini che può discernere i punti navigabili dagli

ostacoli in base al colore del pixel selezionato.

La matrice dei punti navigabili ottenuti oltre che permettere la visualizzazione della posizione dell'utente all'interno della planimetria, rende operativo l'algoritmo A* per il calcolo del percorso minimo. Tale algoritmo consente di convertire in nodi del grafo i punti navigabili utili a raggiungere un obiettivo dato. Grazie a A*, note le coordinate dell'utente e le coordinate di un POI selezionabile, si è potuto tracciare il percorso minimo che permette ad un utente di raggiungere un POI target da lui selezionato.

La visualizzazione di questo percorso, che può mutare in base ad ostacoli e al cambiamento della posizione dell'utente, ha lo scopo di assisterlo durante la navigazione, evitandogli ricerche inutili, soprattutto se l'ambiente in cui naviga gli è sconosciuto. Questo goal è sicuramente uno dei più interessanti del sistema, in quanto potrebbe avere anche una ricaduta economica se il sistema fosse applicato realmente, ad esempio in un grande centro commerciale.

In definitiva il sistema proposto, in base ai test che ne hanno accompagnato lo sviluppo e ai test realizzati in un centro commerciale disposto su più piani, si può dire funzionante ed efficiente.

Ringraziamenti

Vorrei ringraziare la mia famiglia per avermi dato la possibilità di raggiungere anche questo traguardo.

Ringrazio i miei amici e compagni di corso. Grazie per tutte le belle serate passate insieme e per l'aiuto dato, sia che fosse di natura tecnica che altro.

Un pensiero speciale ad Agnese che durante questi mesi mi ha sempre sostenuto, soprattutto durante le difficoltà.

Un ringraziamento anche a GetConnected e in particolare ad Alessandro Rizzoli per avermi accolto per due mesi di tirocinio, facendomi conoscere la tecnologia geomagnetica che è alla basa della tesi.

Infine vorrei ringraziare il prof. Viroli per la professionalità dimostrata e per i consigli dati.

Bibliografia

[1] *"Magnetic Positioning - The Arrival of 'Indoor GPS'"*, OpusResearch, 2014, https://www.indooratlas.com/wp-content/uploads/2016/03/magnetic_positioning_opus_jun2014.pdf

[2] L. Leh, *"ZigBee-based intelligent indoor positioning system soft computing"*, 2008, Springer, http://link.springer.com/article/10.1007/s00500-013-1067-x

[3] Paolo Caramanica *"Il teorema del coseno (o di Carnot)"*, 2010, http://www.trigonometria.org/trigonometria/pdf/teorema-coseno.pdf

[4] Luca Pappalardo, *"Localizzazione - Problema, Tecniche, Algoritmi - Reti mobili: Ad Hoc e di sensori"*, 2011, http://didawiki.di.unipi.it/lib/exe/fetch.php/rhs/localizzazione.pdf

[5] Cuccado, De Franceschi, Fauri, Sartor, *"Analisi di algoritmi di autolocalizzazione per reti di sensori wireless"*, 2007, https://art.torvergata.it/retrieve/handle/2108/773/6945/Paolo-Sperandio-Tesi-PhD.pdf

[6] Paolo Sperandio, *"Algoritmi di localizzazione per reti di sensori wireless"*, Tesi di laurea, 2007, https://art.torvergata.it/retrieve/handle/2108/773/6945/Paolo-Sperandio-Tesi-PhD.pdf

[7] Joan Gómez Urgellés, *"Quando le rette diventano curve - Le geometrie non euclidee"*, 2012, RBA

[8] Matthew S. Gast, *"Building Applications with iBeacon"*, 2015, O'Reilly,

[9] Ugur Bekcibasi, *"Increasing RSSI Localization Accuracy with Distance Reference Anchor in Wireless Sensor Networks"*, 2014,
http://www.uni-obuda.hu/journal/Bekcibasi_Tenruh_54.pdf

[10] Xuchen Yao, *"An Introduction to the Kalman Filter"*, 2006,
http://unc.live/1dE2kRF

[11] Angelo Nunzio La Bruna, *"Applicazioni del filtro di Kalman su accelerometri"*, 2013,
http://inglabruna.altervista.org/Applicazioni%20del%20filtro%20di%20Kalman%20su%20accelerometri.pdf

[12] Ilenia Tinnirello, *"Un Esempio di Applicazione del Filtro di Kalman alle reti WiFi"*, 2014,
http://www1.unipa.it/~laura.giarre/kalman_app.pdf