Realizzato dall'Università degli Studi di Cagliari

simple

StrumentI e Modelli Per La mobilità sostenibilE

Beep4Me
Repository del modulo mobile per la validazione automatica

UNIONE EUROPEA
Fondo europeo di sviluppo regionale

REPUBBLICA ITALIANA

REGIONE AUTÒNOMA DE SARDIGNA
REGIONE AUTONOMA DELLA SARDEGNA

POR FESR
SARDEGNA 2014-2020

SARDEGNA RICERCHE

Progetto finanziato con fondi *POR FESR 2014/2020 - ASSE PRIORITARIO I "RICERCA SCIENTIFICA, SVILUPPO TECNOLOGICO E INNOVAZIONE".*

Informazioni sul progetto

Numero del progetto	N/A	**Acronimo**	SIMPLE
Titolo completo	StrumentI e Modelli Per La mobilità sostenibilE		
Soggetto	Progetto CLUSTER ICT		
Data inizio	01/02/2018		
Durata in mesi	30		
Coordinatore	UniCA – Università degli Studi di Cagliari		
URL del progetto	http://www.simple-cluster.it		

Informazioni sul documento

Numero del Deliverable	2.3.1	**Titolo**	Beep4Me Repository del modulo mobile per la validazione automatica
Numero del Workpackage	WP3	**Titolo**	Sperimentazione
Data di inizio redazione del documento	24/08/2020		
Autore/i responsabile/i	Marco Garau		
Livello di diffusione	Non applicabile		

Modifiche del documento

Data	Autore	Modifiche	Versione
24/08/2020	Marco Garau	Introduzione e Struttura	0.1
31/08/2020	Marco Garau	Sintesi sezioni 2 e 3	1.0

Tavola dei contenuti

Introduzione **4**

Descrizione ad alto livello di Use Case con eventi **4**

Descrizione Use Case “Bus con Beacon” 4

Descrizione Use Case “Bus senza Beacon” 10

Definizioni **11**

Concetti Fondamentali 11

Eventi 15

Costanti usate nell’app 16

1 Introduzione

In questo documento viene descritto il funzionamento di Beep4Me, il modulo mobile per la realizzazione della validazione automatica sviluppato dal team SIMPLE e attualmente implementato per i sistemi iOS.
Il sistema ha lo **scopo** di **rilevare la presenza** dell'utente all'interno di un bus per l'intera durata del suo viaggio, al fine di **validare il biglietto in modo automatico,** senza o con minima interazione da parte dell'utente.
Per farlo, il sistema effettua una rilevazione della **salita/discesa** dell'utente su/da un mezzo attrezzato con uno o più beacon[1] per mezzo della **tecnologia bluetooth low energy** del telefono. Il software utilizza anche la **localizzazione** e la rilevazione della motion activity sui bus con e senza i beacon (per la realizzazione del check-out), affiancate alla tecnologia bluetooth.

Nella prossima sezione, descriviamo il funzionamento ad alto livello del modulo per i due use case principali: il caso di **bus senza beacon** e il caso di **bus con i beacon**.
Nella sezione 3 vengono messi in evidenza alcune definizioni e Concetti Fondamentali.

2 Descrizione ad alto livello di Use Case con eventi

Per il sensing che ci permette di capire quando un utente è salito o sceso da un bus attrezzato con i beacon, vengono utilizzate principalmente tre tecnologie/sensori dello smartphone:

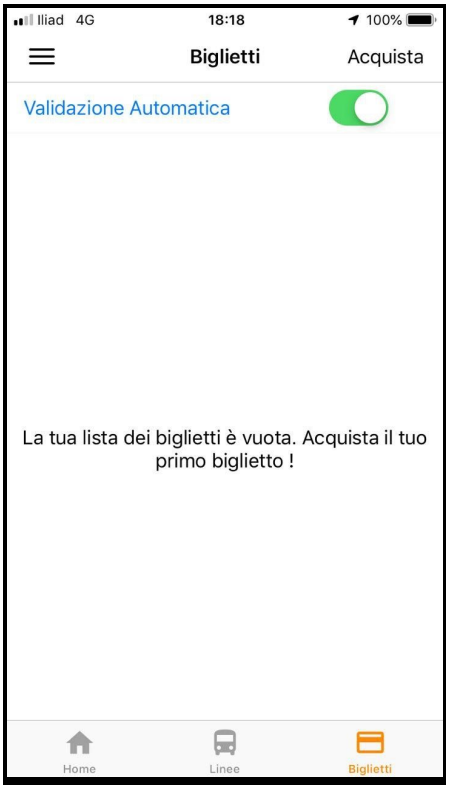
- la rilevazione della motion activity (accelerometro, giroscopio ...),
- la rilevazione della posizione e la tecnologia delle geofence[2], (GPS) e
- il Bluetooth che rileva la "vicinanza" con i beacon BLE installati su alcuni bus.

Con il sensing dell'attività di movimento rileviamo che tipo di attività di moto sta svolgendo l'utente[3], e la associamo al verificarsi di eventi come l'**avvicinamento/allontanamento da una geofence**, l'**ingresso/uscita in/da una fermata** o l'**avvicinamento/allontanamento a/da un beacon** installato su un bus.

2.1 Descrizione Use Case "Bus con Beacon"

Una volta attivata la Validazione Automatica, lato server, il modulo mobile avrà accesso all'API Beep4Me. Questa è integrata nel sistema server per la gestione dei biglietti, validazioni ed altre entità relative alla convalida dei biglietti, dell'azienda di trasporto.

[1] I beacon rilevabili sono **solo** quelli registrati nel sistema e appartenenti a quel gruppo di aziende. Il beacon è un dispositivo Bluetooth che rappresenta la vettura e serve per capire se l'utente è entrato in quella vettura.
[2] Geofence = Area circolare attorno a una posizione fissata. È possibile rilevare l'ingresso e l'uscita da questa zona.
[3] Distinguiamo tra Automotive, Walking, Running, Cycling, Stationary.



Screenshot del bottone attivato nella schermata biglietti

Per attivare la Validazione Automatica (lato app) viene utilizzato un toggle switch Validazione Automatica all'interno della Schermata Biglietti Acquistati. Se questo è acceso, la funzionalità Beep4Me è attiva.

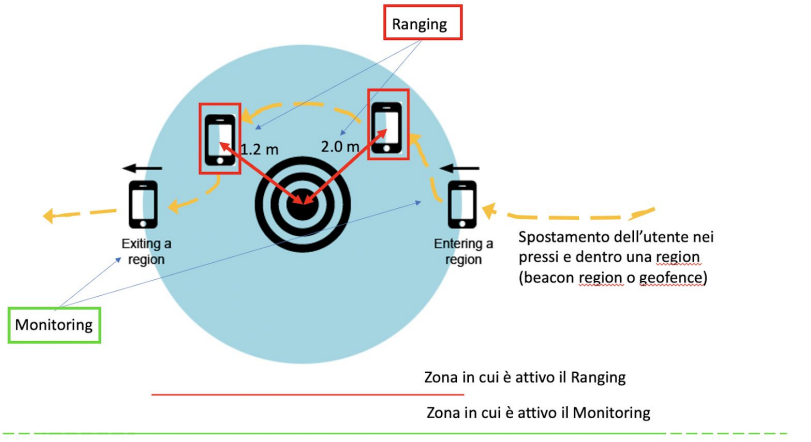
All'avvio dell'app, viene scaricato l'elenco di tutte le fermate (aggiornate una volta al giorno) e viene attivato il monitoring[4] sulle fermate più vicine con la tecnologia delle geofence. Inoltre vengono attivati il monitoring della beacon region[5] per i beacon associati alle aziende di trasporto pubblico e la rilevazione della motion activity[6] dell'utente, che rimane sempre attiva.

In questo modo il SO può rilevare eventi di ingresso e uscita da queste regioni (le geofence e la regione dei beacon), **indipendentemente dallo stato dell'app** e incidendo poco sul livello della batteria. Per caratterizzare meglio ciascuno di essi, associamo a questi eventi una motion activity tra quelle rilevabili.

La rilevazione delle fermate e dei beacon nella beacon region viene effettuata con due step, usando le tecnologia di **monitoring** e di **ranging**[7]. Il monitoring può rimanere sempre attivo e serve in sostanza per attivare/disattivare il ranging, quando avviene un evento di ingresso e uscita da una regione. Il ranging serve per determinare la distanza in modo "più preciso" da una fermata o da un beacon, ma incidendo sensibilmente sul livello della batteria. Quindi deve essere attivato solo quando è necessario.

[4] Monitoraggio su un'area circolare attorno a un punto geografico. È possibile rilevare l'ingresso e l'uscita da essa.
[5] Regione di copertura sferica attorno a un beacon registrato. È possibile rilevare ingresso e uscita da tale zona.
[6] Attività di moto categorizzata da iOS in: Running, Walking, Cycling (queste incluse in Human), Stationary, e Automotive.
[7] Stima della distanza da un beacon: è possibile stimare accuracy e proximity (3 livelli: immediate, near, far).

Nota: Con la tecnologia iBeacon, è necessario uno step in più rispetto ad altre tecnologie. Dato che esiste un limite di 20 regioni monitorabili dal SO[8], usando le librerie iBeacon, è necessario aggiornare le fermate monitorate più vicine con cadenza regolare e abbastanza frequentemente, perché non è possibile monitorare un numero qualsiasi di regioni. Tale step permette di ottenere che sulle fermate vicine all'utente sia sempre attivato il monitoring, man mano che l'utente si sposta durante il viaggio. L'idea è che l'utente abbia sempre attorno a sé una specie di "nuvola" con un certo numero n di fermate monitorate (n = N_FERMATE_MONITORATE).

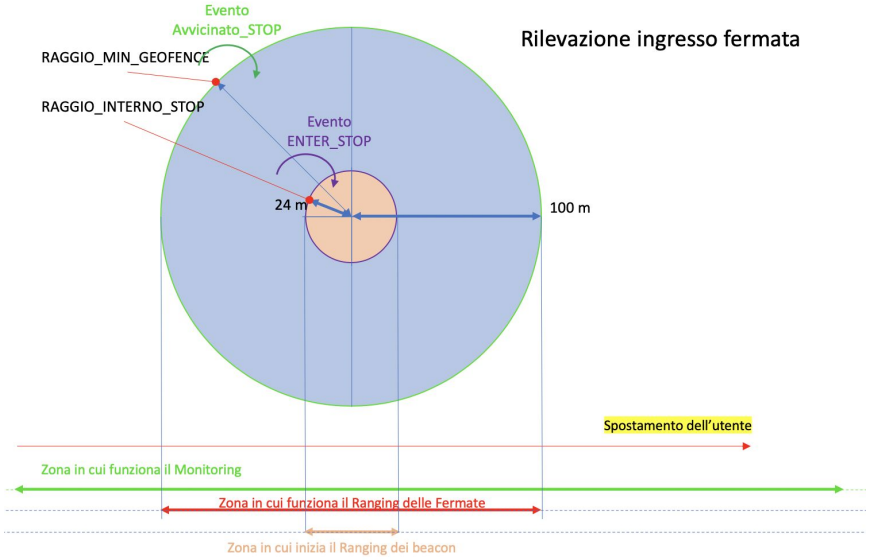


Monitoring / Ranging in generale: l'area celeste rappresenta una geofence oppure l'area di copertura di un beacon BLE.

Per la rilevazione dell'**ingresso in una fermata** vengono usato due raggi (nell'ordine):

- RAGGIO_MIN_GEOFENCE: È la distanza dal CENTER che identifica il minimo raggio della geofence per cui è possibile rilevare un ingresso nella stessa, con i dispositivi utilizzati per i test.. Quando l'utente entra in una geofence, quindi si trova ad una distanza minore di RAGGIO_MIN_GEOFENCE dal center, si verifica l'evento AVVICINATO_STOP, generato dal SO (monitoring) e corrispondente a un evento di enter nella region.
- RAGGIO_INTERNO_STOP: è il raggio che usiamo per rilevare l'ingresso nell'area centrale della fermata. (RAGGIO_INTERNO_STOP < RAGGIO_MIN_GEOFENCE). Se l'utente si trova ad una distanza dalla palina minore di RAGGIO_INTERNO_STOP, viene generato dall'app (GPS) l'evento ENTER_STOP.

[8] Sono incluse nell'insieme sia le beacon region che le geofence.



Mentre l'utente si sposta camminando (o correndo, o in bicicletta o in auto ...) se attraversa il confine di una geofence, cioè se si avvicina a meno di RAGGIO_MIN_GEOFENCE dal centro della geofence, il SO rileva un evento di enter region (vedi geofence), qualunque sia lo stato dell'app[9]. Definiamo questo evento col nome AVVICINATO_STOP.
L'evento AVVICINATO_STOP scatena l'attivazione del ranging di quella fermata[10], per determinare se l'utente entra nella parte centrale della geofence corrispondente al cerchio con raggio RAGGIO_INTERNO_STOP (evidenziata in color salmone in figura Rilevazione ingresso fermata). Se la distanza diventa inferiore a RAGGIO_INTERNO_STOP, si afferma che l'utente è *entrato nella fermata*, definendo l'evento un ENTER_STOP.

> **Nota**: Il fatto di sapere se l'utente è entrato nella fermata è utile perché all'interno della fermata potrebbe iniziare un nuovo viaggio. L'ENTER_STOP è molto importante quindi e, quando si verifica, le informazioni relative a tale evento vengono inviate al server.

Quando avviene l'evento ENTER_STOP viene attivato il ranging dei beacon. Se arriva un bus con i beacon installati e l'utente sale a bordo, si vuole rilevare il suo ingresso e l'inizio del viaggio. L'app, che sta eseguendo il ranging dei beacon, immediatamente inizia a registrare i pacchetti di advertising ricevuti dai beacon. Questi hanno associate diverse informazioni, tra cui una potenza ricevuta attraverso il segnale BLE (RSSI), una proximity e una accuracy. Attraverso queste informazioni è possibile **stimare la vicinanza con ciascuno dei beacon del gruppo di aziende**[11], e attraverso i codici impostati su ogni beacon installato sul bus è possibile **associare il beacon alla vettura**. Quindi è possibile determinare la presenza dell'utente nei pressi dei beacon di quel determinato bus e quindi sul bus stesso.

[9] Ad esempio anche se l'app risulta Killed, o in Background. È necessario che l'app sia stata installata e avviata la prima volta, in modo da aver scaricato almeno una volta le fermate del gruppo di aziende e avviato il processo di monitoring delle geofence.
[10] Calcolo della distanza della posizione GPS del dispositivo utente dal centro della geofence, calcolata ogni n secondi.
[11] Solo dei Beacon che si trovano a pochi metri dal dispositivo utente.

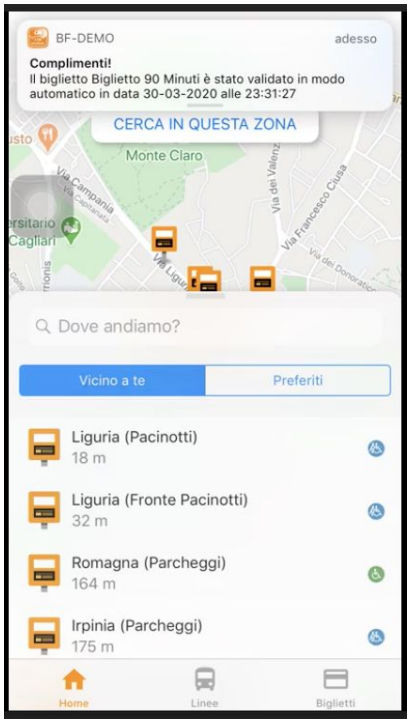
L'instabilità del segnale bluetooth rende però un singolo campione di potenza ricevuta inadatto a determinare con affidabilità l'ingresso dell'utente sul bus. Allora si è reso necessario l'uso di una coda FIFO per fare una sorta di rilevazione di continuità della presenza dell'utente sul bus.
Il buffer FIFO trattiene solo gli ultimi campioni di potenza ricevuta[12] e permette di capire se l'utente si trova stabilmente sul bus (o se ne è uscito stabilmente), oppure, ad es., se si è avvicinato ad esso solo per qualche istante per chiedere una informazione all'autista. Lo facciamo contando i campioni con una RSSI superiore a una certa soglia, MIN_RSSI_TO_ENTER_BEACON. Se questi, in numero, superano una certa percentuale PERC_HIGH_RSSI_TO_ENTER_BUS rispetto al totale dei campioni del buffer, allora *si potrebbe* trattare di un evento di salita sul bus.
Esempio:

```
N_LAST_RSSI_RECEIVED = 10  // lunghezza buffer FIFO
N_CAMPIONI_SOPRASOGLIA = 8
PERC_HIGH_RSSI_TO_ENTER_BUS = 80%

if (PERC_CAMPIONI_SOPRASOGLIA >= PERC_HIGH_RSSI_TO_ENTER_BUS) {
   // evento candidato ENTER_BUS
   if (detected_activity == Automotive) {
      ENTER_BUS
   }
}
```

In questo momento entra in gioco l'activity detection: se l'attività rilevata è ***Automotive***, allora la salita sul bus viene etichettata come un evento ENTER_BUS. L'evento ENTER_BUS è la condizione utile per poter eseguire una richiesta di check-in al server. Quando avviene un evento ENTER_BUS, le informazioni riguardanti l'evento stesso e le informazioni sul bus ricevute dai beacon vengono inviate all'API per la validazione automatica. Il server, dopo aver verificato alcune condizioni sui biglietti dell'utente, decide quale biglietto validare e risponde con un messaggio che significa "check-in eseguito!" oppure risponde con informazioni che rimandano la scelta del biglietto da validare all'utente[13] oppure ancora con una risposta che servirà per proporgli l'acquisto di un biglietto, se non dovesse averne disponibili. Lato app, l'utente riceverà delle notifiche sul display del suo dispositivo simili a quella nella figura sotto.

[12] La dimensione del buffer è definita come N_LAST_RSSI_RECEIVED.
[13] L'utente lo deve solo selezionare, completando così la validazione.



Per la rilevazione dell'**uscita dalla fermata**, usiamo ancora due raggi:

- RAGGIO_ESTERNO_STOP: l'area interna della fermata per rilevare l'uscita ha questo raggio. Uscendo dal confine di quest'area viene generato l'evento EXIT_STOP.
- RAGGIO_MIN_GEOFENCE: l'uscita dalla geofence viene rilevata dal SO con un evento exit region dalla geofence. Il raggio monitorato per l'exit (RAGGIO_MIN_GEOFENCE) è uguale a quello definito per l'ingresso nella geofence. L'uscita dall'area di raggio RAGGIO_MIN_GEOFENCE dovrebbe triggerare tale evento. L'evento di exit dalla region invece può essere rilevato con un errore di anche 100 metri, indipendentemente dalla dimensione della geofence (almeno con il dispositivo utilizzato per i test). Questo errore dipende da fattori esterni (ricezione segnale GPS, disposizione delle celle e degli hospot wifi) e dalle **caratteristiche dello smartphone**.

Dopo avere effettuato l'ENTER_BUS (l'utente si trova sul bus), il ranging dei beacon rimane attivo fintantoché i beacon rimarranno "visibili" dallo smartphone. Quindi avvengono i seguenti passaggi:

1. il bus parte e viene rilevata motion activity = Automotive.
2. Il ranging della fermata che è attivo, perché siamo all'interno della geofence rileverà un evento EXIT_STOP, nel momento in cui ci si allontana a più di RAGGIO_ESTERNO_STOP metri[14] dal centro.
3. L'evento EXIT_STOP triggera l'invio di informazioni riguardanti l'evento stesso al server.
4. Allontanandosi ancora di più dal centro si incrocerà il RAGGIO_MIN_GEOFENCE in uscita dalla geofence.
5. Il SO rileva l'evento di exit region, ma questo evento può essere segnalato con un certo ritardo dipendente da vari fattori[15]. In questo momento (dopo il ritardo) viene "notificato" dal SO un evento di exit region, che noi definiamo ALLONTANATO_STOP.
6. L'evento ALLONTANATO_STOP ci è utile per triggerare il blocco del monitoring di quella fermata.

[14] Impostato a 26 m nell'implementazione iOS.
[15] Portando il confine in uscita a circa 200-220 metri dal centro della geofence, nell' ambiente e con il dispositivo utilizzato per i nostri test preliminari.

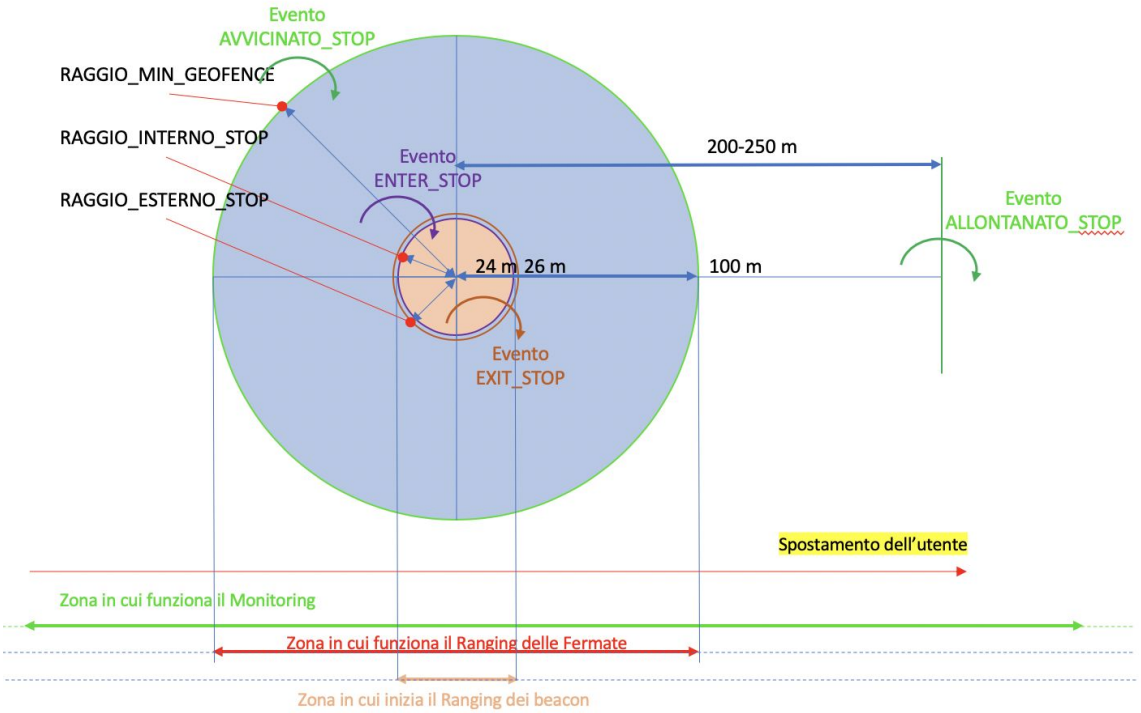
7. Il ranging invece viene bloccato volutamente un po' dopo il momento in cui avviene l'evento ALLONTANATO_STOP, ad una distanza definita DISTANZA_MAX_STOP_RANGING[16] dalla palina. Abbiamo ritenuto necessario dare una certa distanza di sicurezza tra il punto in cui avviene lo stop del monitoring e questo punto in cui avviene lo stop del ranging. Questo per essere sicuri che il ranging di quella fermata non sia più utile, ferma restando la necessità di evitare che troppi processi di ranging di fermate rimangano attivi nell'app.
8. Mentre il viaggio procede, il bus attraversa diverse geofence delle fermate monitorare generando una successione di eventi AVVICINATO_STOP / ALLONTANATO_STOP (attivando e disattivando il ranging di volta in volta). Potenzialmente, entra ed esce da una fermata, generando eventi ENTER_STOP ed EXIT_STOP, se la distanza misurata dal ranging scende sotto i RAGGIO_INTERNO_STOP metri e poi sale sopra i RAGGIO_ESTERNO_STOP metri. Questi ultimi due eventi vengono inviati comunque al server, quando avvengono, perché sono utili per l'esecuzione del checkout con la posizione. I beacon continuano a essere rilevati ("ranged") dall'app. Dato che è avvenuto l'ENTER_BUS, l'app continua ad aspettare che si verifichi un evento EXIT_BEACON[17], per iniziare il processo di rilevazione dell'uscita dal bus.
9. All'arrivo alla fermata di fine viaggio, l'utente scende dal bus e si allontana da esso (o il bus si allontana ripartendo).
10. Avviene l'EXIT_BEACON e per rilevare l'effettiva discesa dal bus viene utilizzato di nuovo il buffer FIFO. Vengono contati i campioni di RSSI con valore sotto una certa soglia pari a MAX_RSSI_TO_EXIT_BEACON). Se la percentuale di campioni sotto-soglia supera la percentuale sul totale di PERC_LOW_RSSI_TO_EXIT_BUS, si *potrebbe trattare di un evento di discesa dal bus*. Perché diventi un evento EXIT_BUS, questo deve essere associato ad una activity rilevata uguale a una attività compiuta da un essere umano, che può essere walking o running o cycling o stationary.
11. Quando si verifica, le informazioni sull'evento stesso vengono inviate al server, per richiedere un check-out. Il server, fatte le verifiche del caso, risponde positivamente e l'app notifica il check-out all'utente, oppure negativamente con una risposta di *evento non significativo*. L'app quindi non mostra alcuna notifica all'utente. Quindi se l'utente si allontana dalla fermata, prima uscirà dalla stessa (EXIT_STOP con post al server), poi uscirà dalla geofence (ALLONTANATO_STOP con stop del monitoring di quella geofence) e poi il ranging rileverà una distanza dell'utente dalla palina superiore a DISTANZA_STOP_RANGING metri, disattivando anche il ranging di quella fermata.
12. Il ciclo continua entrando e uscendo dalle geofence delle fermate più vicine, fino all'inizio di un nuovo viaggio su un bus, quando vengono rilevati dei beacon in salita sul bus.

2.2 Descrizione Use Case "Bus senza Beacon"

Nel caso in cui il bus su cui sale l'utente non sia provvisto di beacon. Non possiamo quindi identificare il bus attraverso il segnale che arriva da questi allo smartphone dell'utente. Vogliamo almeno eseguire il "check-out con la posizione", sfruttando il sensing delle fermate. Si tratta di utilizzare i 4 eventi riferiti alla geofence/fermata di cui abbiamo già parlato. Dopo i due eventi AVVICINATO_STOP e ENTER_STOP, l'ingresso nel bus non può essere rilevato con i beacon, perché questi non sono presenti. Quindi non è possibile effettuare la validazione automatica (check-in). Allora l'utente è tenuto ad eseguire la validazione col QR-code (trasbordo o validazione). Anche in questo caso, le informazioni[18] riguardanti i due eventi

[16] Nell'implementazione iOS è stata impostata a 300m.
[17] Che avviene quando la RSSI scende sotto la soglia MAX_RSSI_EXIT_BEACON.
[18] Tra queste informazioni è compresa anche l'activity associata all'evento.

ENTER_STOP ed EXIT_STOP, rilevati dal ranging della prima fermata, vengono inviate al server, perché sono utili allo scopo di eseguire il check-out automatico con posizione. La prima fermata è per noi la fermata in cui rileviamo (lato server) un ENTER_STOP con **activity Human**, e un EXIT_STOP con activity Automotive, dopo aver rilevato altri eventi di enter ed exit con diverse combinazioni di activity per i due eventi.



Durante il viaggio anche i successivi eventi ENTER_STOP ed EXIT_STOP nelle fermate attraversate vengono inviati al server, con activity associata uguale a *Automotive*. All'arrivo alla fermata di destinazione, l'app rileverà come sempre gli eventi AVVICINATO_STOP ed ENTER_STOP (con activity Automotive). L'utente scende dal bus e inizia ad allontanarsi dalla fermata con activity Human. All'attraversamento del raggio RAGGIO_ESTERNO_STOP con activity Human, viene generato l'evento EXIT_STOP che viene inviato al server, come tutti gli altri. Il server adesso valuta questo evento e lo identifica come un possibile "check-out". Se nelle N ore precedenti[19] a tale evento è stato validato un biglietto dell'utente su un certo bus, allora sullo stesso biglietto verrà effettuato un checkout su quel bus. Quando l'app riceve una risposta positiva di checkout, notifica il checkout eseguito all'utente. Se la risposta è negativa (*evento non significativo*) non mostra nulla all'utente.

3 Definizioni

3.1 Concetti Fondamentali

Beacon: dispositivo Bluetooth Low Energy normalmente alimentato a batteria che emette con una certa cadenza periodica un segnale/pacchetto che può essere rilevato da dispositivi come smartphone e tablet. Nel caso della tecnologia iBeacon, il pacchetto comprende tre codici identificativi UUID, major e minor e

[19] Con N = 2, nell'implementazione attuale. In questo caso, si ipotizza che il viaggio dell'utente possa durare massimo 2 ore. In questo modo, le validazioni precedenti vengono scartate.

altre informazioni (come la potenza ricevuta RSSI). In base a queste informazioni lo smartphone è in grado di rilevare il singolo beacon e stimare la vicinanza con esso, triggerando utili azioni.

Geofence: definiamo geofence una regione geografica circolare di un certo raggio, attorno a una certa posizione geografica fissata (center). Su questa regione, si può attivare il monitoring, funzione con la quale il SO[20], "sente" l'ingresso e l'uscita dalla geofence, generando eventi di **enter** ed **exit** dalla regione che *"svegliano"* l'app portandola in background, indipendentemente dal suo stato (anche dallo stato "killed"), per permetterle la gestione dell'evento.

Area Centrale della Fermata: chiamiamo Area centrale delle Fermata o Fermata, per brevità, un'area circolare con centro nella palina e di raggio = RAGGIO_INTERNO_STOP (per rilevare l'ingresso) oppure RAGGIO_ESTERNO_STOP (per rilevare l'uscita). Nell'implementazione iOS i due raggi della fermata sono nell'intorno di 25 metri.

Center: è il centro di una geofence definita su una fermata, espresso come coordinate geografiche della palina.
Raggio della Geofence: è il raggio della geofence usata per il monitoring di una fermata. Se si attraversa questo raggio in ingresso/uscita si ottengono eventi di enter/exit dalla geofence, che possono triggerare azioni dell'app. Il raggio minimo impostabile per una geofence è RAGGIO_MIN_GEOFENCE = 100m (sperimentale).
Regione dei beacon: una beacon region non è definita da alcuna proprietà geografica. La **rappresentazione fisica di una beacon region in generale è il range (area di copertura del segnale) di tutti i beacon che fanno parte della regione.** Una beacon region (vedi iBeacon) è caratterizzata dai tre attributi UUID, major e minor (dettagli sui codici dei beacon).
Con la tecnologia iBeacon, è possibile definire una regione in 3 modi:

- **Usando solo l'UUID**: in questo caso la beacon region consiste di tutti i beacon con un dato UUID. Per esempio: una region con UUID ACFD065E-C3C0-11E3-9BBE-1A514932AC01 (che è quello di default di iBeacon) consiste di tutti i beacon con quell'UUID particolare impostato. Ad esempio, in uno scenario in cui operano Azienda1, Azienda2 e Azienda3, la beacon region con questo UUID rappresenta i beacon appartenenti a questo gruppo di aziende. Opzionalmente si può anche dare un **Identifier** (una stringa alfanumerica human readable) a quella region, che corrisponde a quel particolare UUID all'interno dell'app.
- **Usando UUID e Major**: in questo caso la beacon region consiste di tutti i beacon che utilizzano una specifica combinazione di UUID e Major. Per esempio, tutti i beacon con UUID di default e Major impostato su 11334.
- **Usando UUID, Major e Minor**: in questo caso la beacon region consiste di un singolo beacon. Per esempio, un beacon con UUID di default, Major impostato su 112 e Minor impostato su 27026.

In base alla potenza di trasmissione dei beacon, la copertura (e quindi la dimensione fisica della beacon region) può essere aumentata/ridotta. Le aree intorno al beacon possono essere sentite anche in termini di proximity, oltre che in termini di enter ed exit.

[20] Per esempio iOS.

Pacchetto di advertising: è un piccolo blocco di informazioni che l'app riceve dai beacon tramite il segnale BLE. Esso contiene varie informazioni tra cui RSSI, major, minor, proximity, accuracy.

UUID: codice a 16 byte, rappresentato con una stringa (es. "B9407F30-F5F8-466E-AFF9-25556B57FE6D"). Ad es. in uno scenario in cui operano Azienda1, Azienda2, Azienda3, l'UUID rappresenta tutti i beacon di questo gruppo di aziende, e quindi i beacon che il sistema può rilevare.

Identifier: stringa di testo human-readable che può essere usata nell'app, come alias di quella region (beacon region o geofence). In questo documento l'UUID rappresenta il gruppo di aziende di trasporto che riguardano il caso d'uso di interesse.

> **Nota**: nel mondo fisico esistono i beacon con le loro potenze di trasmissione e i loro codici (UUID, major e minor) settati. Le region sono create in app, e quindi l'Identifier è un parametro che esiste solo alla creazione di una certa region, ed è usato come alias per essa.

Major: è un numero di 2 byte (o un "unsigned short") tra 0 e 65535. In questo documento, il *major* rappresenta un identificativo riconducibile al numero della vettura su cui è installato quel beacon.

Minor: è un numero di 2 byte (o un "unsigned short") tra 0 e 65535. Nel sistema, il *minor* rappresenta l'id del beacon all'interno del bus (ipotizzando di installare da 1 a N beacon per ogni mezzo).

Proximity: la proximity è definita come una *enumerazione* di tre possibili livelli di prossimità:

- far = assimilabile a qualche metro - 10 metri
- near = assimilabile a più di 1 metro
- immediate = assimilabile a circa 1 centimetro - 50 centimetri[21]

La stima migliora con la vicinanza. L'errore di questa stima è rappresentato dal valore dell'Accuracy. È possibile associare a una proximity (passaggio da una zona all'altra) un trigger per determinate azioni. Per avere proximity e accuracy più accurate, si dovrebbe tarare il valore di RSSI rilevato a 1 metro per ciascun beacon.

Accuracy: è un valore in metri che stima l'errore di rilevazione della proximity, assimilabile alla distanza dal beacon per distanze molto ravvicinate (proximity = immediate).

RSSI: acronimo di Received Signal Strength Indicator. È un valore che sta nel range -100 dBm (valore più basso) e 0 dBm (valore più alto). Indica il valore di potenza ricevuta attraverso il segnale Bluetooth col pacchetto a cui è associata.

Gruppo di aziende: in questo documento parliamo di gruppo di aziende per indicare le aziende che fanno parte del servizio di trasporto integrato, che hanno interesse alla funzione di convalida automatica del biglietto.

Sensing: definiamo sensing la capacità dell'app di rilevare **fermate** e **beacon** relativi al gruppo di aziende

[21] Valori basati su una corretta calibrazione dei beacon. In base all'ambiente di installazione, alle impostazioni dei beacon, ma anche alle normali fluttuazioni del segnale bluetooth la distanza reale a cui queste proximity vengono rilevate, può variare

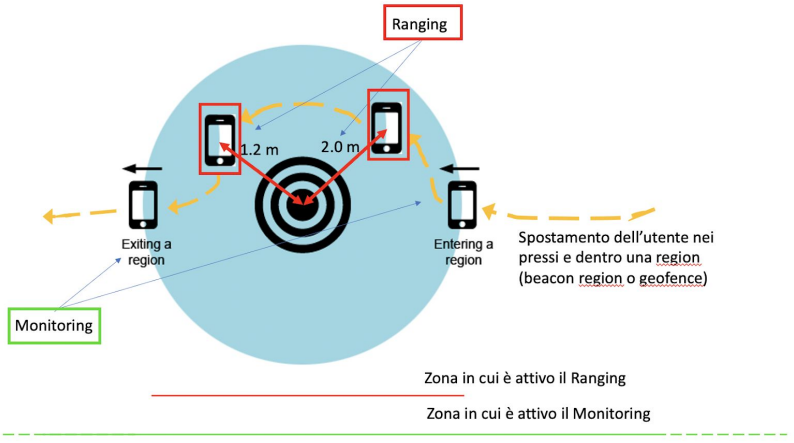
L'app interagisce con i beacon e le fermate in due modi:

- il **Monitoring (dei beacon o delle geofence)**: è una funzionalità offerta dal sistema operativo che permette il trigger di azioni basate su eventi di ingresso e uscita nel range della **regione del beacon** o di una **geofence**. Funziona indipendentemente dallo stato in cui si trova l'app (running, suspended o killed).
- **Ranging (dei beacon o delle fermate)**: è una funzionalità dell'app, che permette di stimare la "distanza" da un **beacon** oppure dal **centro di una fermata**. In base alla distanza si può triggerare un'azione basata sulla proximity (vicinanza) a un beacon (utilizzando il BLE) o a una fermata (utilizzando il GPS). **Funziona solo se l'app è running** (in foreground o in background).

  Il ranging dei beacon interagisce con il singolo beacon appartenente a una regione. Questo nella tecnologia iBeacon permette di rilevare 3 livelli di proximity (coi quali è possibile triggerare diversi tipi di comportamenti):

  - far, che teoricamente significa "a diversi metri, fino a decine di metri dal beacon"
  - near, che si potrebbe ottenere "a circa un metro dal beacon"
  - immediate, che dovrebbe essere rilevato ad "alcuni centimetri dal beacon"
  - unknown, che sta per "proximity non rilevata"[22].

  insieme con l'accuracy, l'RSSI, i codici del beacon e altre informazioni.



Monitoring / Ranging in generale: l'area celeste rappresenta una geofence oppure l'area di copertura di un beacon BLE.

Motion activity: attività di movimento rilevata dal servizio di motion detection del SO. Il SO iOS permette di rilevare una delle seguenti activity:

- walking (human)
- running (human)
- cycling (human)
- automotive
- stationary
- unknown

[22] Che di solito si ottiene quando ci sono problemi di ricezione, o anche in avvicinamento/allontanamento dal beacon.

Buffer delle ultime RSSI ricevute: questo buffer mantiene solo le ultime **N_LAST_RSSI_SAMPLES** RSSI ricevute. per dedurre se l'utente si trova effettivamente sul bus.

Potenza di trasmissione del beacon: è la potenza con cui i beacon trasmettono i loro pacchetti di advertising. Ad es. nei beacon in uso al team SIMPLE, può essere impostata con i valori: +4dBm, 0dBm, -4dBm, -8dBm, -12dBm, -16dBm, -20dBm, -30dBm, -40dBm. Al crescere della potenza trasmessa il beacon avrà un'area di copertura (ma anche un consumo di batteria) maggiore.

Intervallo di Advertising del Beacon: è l'intervallo di tempo tra due trasmissioni successive in broadcast di pacchetti di advertising da parte di un beacon. Nei beacon a disposizione del team SIMPLE, si possono impostare valori da 100 ms fino a 10 s.

Controllo sulla Stabilità dell'Ingresso e dell'uscita: Definiamo controllo sulla stabilità dell'ingresso (o dell'uscita) nel bus, un check eseguito sugli ultimi valori di potenza RSSI ricevuta dai beacon, memorizzati in un buffer.
Biglietto Attivo: è un biglietto validato almeno una volta e non scaduto.

3.2 Eventi

AVVICINATO_STOP (alias enter in geofence o enter region): definisce l'ingresso in una geofence di raggio RAGGIO_MIN_GEOFENCE e centro (CENTER) nella palina della fermata. Triggera il ranging di quella fermata.

ALLONTANATO_STOP (alias per exit geofence o exit region): definisce l'uscita da una geofence. Triggera lo stop del monitoring di quella fermata (geofence). L'uscita dalla geofence può essere rilevata dal SO iOS con un errore di circa 100-120 metri (RAGGIO_MIN_GEOFENCE + 100 metri dal center). Tale errore sembrerebbe legato alle caratteristiche del dispositivo e alla disposizione delle celle telefoniche e hot spot wifi nella città, oltre che ovviamente alla qualità del segnale GPS ricevuto.

ENTER_STOP: si tratta dell'ingresso nella fermata. L'evento ENTER_STOP si verifica quando l'utente si trova a meno di RAGGIO_INTERNO_STOP dal centro della fermata (= 24 m nell'implementazione iOS). Tale evento scatena task come l'invio dell'evento stesso al server e lo **start** del ranging dei beacon.

> **Nota**: al verificarsi dell'evento ENTER_STOP avviene solo l'invio della post al server, e l'app alla ricezione della risposta non deve fare niente.

EXIT_STOP: si tratta dell'uscita dalla fermata, ad una distanza dal centro (palina) superiore a RAGGIO_ESTERNO_STOP (= 26 m nell'implementazione iOS,). L'evento EXIT_STOP avvia dei task, come l'invio dello evento al server e lo **stop** del ranging dei beacon nel caso non siano stati rilevati beacon dentro la fermata. Quando avviene un EXIT_STOP, con activity = Walking, se il corrispondente ENTER_STOP ha activity = Automotive e l'utente ha un biglietto attivo nel database Acquisti, si fa un checkout su quel biglietto. La risposta l'app contiene le informazioni per "notificare il check-out" all'utente.

> **Nota**: se vengono rilevati dei beacon nell'intervallo di tempo compreso tra ENTER_STOP ed EXIT_STOP, è corretto lasciare il ranging attivo per rilevare la distanza del dispositivo utente dai beacon del bus[23] per tutta la durata del viaggio.

[23] E quindi la permanenza del dispositivo utente sul bus.

ENTER_BEACON_REGION: ingresso nella regione dei beacon rilevato dal SO (monitoring).

EXIT_BEACON_REGION: uscita dalla regione dei beacon che viene rilevata dal SO (monitoring). In genere ho un ritardo di 30 secondi dopo l'effettivo allontanamento da quella regione.

ENTER_BEACON: quando si entra in una beacon region, i beacon possono essere "ranged" (dall'app). Evento rilevato quando ci si avvicina a meno di una certa distanza dal beacon e la potenza ricevuta da questo beacon sale sopra una certa soglia RSSI_ENTER_BEACON. L'evento di per sé non consente di affermare che l'utente sia effettivamente entrato nel bus, a causa delle fluttuazioni del segnale Bluetooth che arriva dai beacon.

EXIT_BEACON: sempre all'interno della beacon region, quando ci si allontana più di una certa distanza dal beacon e la potenza scende sotto una certa soglia MAX_RSSI_TO_EXIT_BEACON, abbiamo un evento EXIT_BEACON (rilevato dall'app). Questo evento di per sé non basta per determinare che l'utente è effettivamente/stabilmente uscito dal bus, perché un allontanamento di breve durata o una potenza rilevata bassa per un breve periodo/ istante potrebbe essere dovuto alla fluttuazione dei segnali Bluetooth rilevati dall'app. Le due soglie di potenza RSSI_ENTER_BEACON ed RSSI_EXIT_BEACON sono abbastanza diverse, per cercare di filtrare gli effetti della fluttuazione del segnale bluetooth low energy.

ENTER_BUS: utile a causa della fluttuazione dei segnali BLE. Si utilizza un buffer FIFO per memorizzare gli ultimi N campioni di potenza ricevuta, distinguendo l'evento di ingresso nell'area del beacon (ENTER_BEACON) dall'evento di ingresso nel bus (ENTER_BUS), che accade quando il numero di campioni di potenza soprasoglia[24] nel buffer supera una certa percentuale rispetto al totale nel buffer FIFO.

EXIT_BUS: L'impossibilità di utilizzare l'evento exit region della funzionalità di monitoring delle regioni dei beacon offerta dal SO, determinata dal fatto che l'uscita da un bus e l'ingresso immediato su un altro bus nella stessa fermata non permettono di ottenere un exit beacon region (l'exit avviene quando non vengono rilevati pacchetti di advertising e quindi beacon per circa 30 secondi) perché i beacon sui bus fanno parte dalla stessa regione dei beacon[25], ha spinto a rilevare anche l'uscita dal bus attraverso il ranging. Abbiamo quindi definito ed utilizzato anche un evento EXIT_BUS. L'evento EXIT_BUS avviene quando la coda FIFO dei campioni di potenza RSSI memorizzati è piena e l'X % di questi campioni ha valore sotto la soglia MAX_RSSI_EXIT_BEACON. Triggera l'invio di una post al server, per la richiesta di check-out.

3.3 Costanti usate nell'app

N_FERMATE_MONITORATE: numero di fermate monitorate nel sistema iOS, che deve essere inferiore a 20 (N_BEACON_REGION_MONITORATE).
N_BEACON_REGION_MONITORATE: data l'impossibilità di definire una beacon region per ogni bus (dovuta al limite massimo delle 20 regioni monitorabili) e la conseguente scelta di definire una sola beacon region con il solo codice UUID, comprendente tutti i beacon del gruppo di aziende, **il numero delle regioni dei beacon monitorate è 1**.
RAGGIO_MIN_GEOFENCE: definisce il valore minimo per il raggio della geofence, entro il quale iniziamo ad utilizzare il GPS per rilevare il suo ingresso nell'area centrale della fermata (vedi figura), 100 metri nel nostro caso di studio. Il valore minimo per questo raggio, misurato con il dispositivo a disposizione del

[24] RSSI > MIN_RSSI_TO_ENTER_BEACON.
[25] Due bus posizionati alla fermata, uno vicino all'altro, risultano per l'applicazione (per il SO) la stessa beacon region.

team, è stato di 100 metri. Quando si cerca di impostare un raggio inferiore, enter ed exit[26] da quella geofence vengono comunque rilevati a 100 metri dal suo centro.
RAGGIO_INTERNO_STOP: questo raggio serve per determinare l'ingresso dell'utente nella fermata generando un evento ENTER_STOP, con la sua activity rilevata. Nell'implementazione iOS, è stato impostato a 24 metri.
RAGGIO_ESTERNO_STOP: tale raggio serve per determinare l'uscita dell'utente dalla fermata generando un evento EXIT_STOP, con la sua activity rilevata. Nell'implementazione iOS, è stato impostato a 26 m.
LUNGHEZZA_BUFFER = N_LAST_RSSI: è il numero di campioni che può contenere il buffer. Rappresenta il massimo tra i campioni di potenza che vengono ricevuti da tutti i beacon secondo per secondo. Tutti i beacon di un certo bus (major) concorrono a riempire lo stesso buffer.
MIN_RSSI_TO_ENTER_BEACON: rappresenta la potenza ricevuta (RSSI) alla quale posso affermare di essere abbastanza vicino a un beacon.

MAX_RSSI_TO_EXIT_BEACON: rappresenta la potenza ricevuta alla quale posso affermare di essere lontano dal beacon.

> **Nota**: Le due soglie di potenza sono abbastanza diverse, per cercare di limitare gli effetti di fluttuazione del segnale bluetooth low energy.

TIME_TO_ENTER_BUS: è l'intervallo di tempo in cui il buffer FIFO dove sono memorizzati gli ultimi LUNGHEZZA_BUFFER valori di RSSI, ci mette per riempirsi con un numero di campioni sopra la soglia (MIN_RSSI_ENTER_BEACON) maggiore del 80 % del totale.

TIMESTAMP_LAST_DOWLOAD_STOPS: è l'orario a cui sono state scaricate le fermate l'ultima volta.

PERC_HIGH_RSSI_TO_ENTER_BUS: percentuale dei campioni di potenza RSSI sul totale dei campioni contenuto nel buffer che devono superare la soglia MIN_RSSI_TO_ENTER_BEACON, da superare per poter affermare che l'utente è entrato nel bus.

PERC_LOW_RSSI_TO_EXIT_BUS: percentuale dei campioni di potenza RSSI sul totale dei campioni contenuto nel buffer che devono essere sotto la soglia MAX_RSSI_TO_EXIT_BEACON, da superare per poter affermare che l'utente è uscito dal bus.

PERC_HIGH_RSSI_TO_EXIT_BUS: percentuale dei campioni di potenza RSSI sul totale dei campioni contenuto nel buffer che risultano ancora sopra-soglia (RSSI > MIN_RSSI_TO_ENTER_BEACON), da non superare per poter affermare (insieme alla condizione sui campioni sotto-soglia) che l'utente è uscito dal bus.

DISTANZA_STOP_RANGING: triggera lo stop del ranging di quella fermata.
FREQUENZA_UPDATE_DISTANZA : frequenza con cui calcoliamo la distanza dell'utente dal centro della fermata. Impostiamo una frequenza di 1 valore di distanza al secondo nell'implementazione iOS.

UUID_AGENCY_GROUP: è l'UUID utilizzato per definire la beacon region associata al gruppo di aziende di nostro interesse. Il valore impostato nell'implementazione iOS è ACFD065E-C3C0-11E3-9BBE-1A514932AC02.

[26] La rilevazione dell'exit può essere un po' imprecisa e notificata dal SO anche con errori di 100-120 metri.

IDENTIFIER_AGENCY_GROUP: la stringa usata come alias per l'UUID_AGENCY_GROUP.

INTERVALLO_TEMPO_INVIO_POST_STOP: L'intervallo di tempo di guardia, definito per bloccare l'invio delle post per eventi di ENTER_STOP ed EXIT_STOP già avvenuti. È stato impostato con un valore di 5 minuti nell'implementazione iOS attuale.