Ranging beacons

Ing. Giuseppe Sottile

Nella parte 2 abbiamo visto ed imparato come monitorare i beacons in una applicazione in modo da emettere delle notifiche quando si entra o si esce dalla regione dei dispositivi. Il monitoraggio è certamente una tecnica molto potente, tuttavia risulta essere molto grossolana. Fortunatamente il monitoraggio è completato da un'altra tecnica: Il **ranging beacon**.

Cos'è il ranging

Mentre il monitoraggio crea una sorta di recinto virtuale per rilevare quando ci si sposta all'interno e/o all'esterno, il ranging esegue attivamente la scansione di eventuali beacons nelle vicinanze e fornisce risultati ogni secondo.
Supponiamo di aver definito una regione “Z”. Se si adotta il monitoraggio l'utente verra avvisato ogni qualvolta si entra e/o si esce dalla regione associata all'area (“Z”), se viceversa usiamo il ranging, sulla medesima regione, avremo una lista di tutti i beacon rilevati con tutte le informazioni associate(UUID, major, minor).

Esiste anche un'ulteriore caratteristica che contraddistingue il ranging che merita una descrizione di seguito.

La stima di prossimità

Ogni beacon trasmette il segnale bluetooth con una certa forza, una forza che diminuisce perche il segnale attraversa l'aria. Ciò consente al dispositivo di fare una certa stima della distanza alla quale si trova il trasmettitore beacon e evidente che se il segnale è forte il beacon è più vicino di quanto non lo sarebbe se il il segnale fosse più debole*.

*In realtà si tratta di una legge dell'inverso dei quadrati della distanza; vale a dire che se la distanza dal beacon aumenta di due, l'intensità del segnale diminuisce di quattro. Questo fa si che l'accuratezza delle stime di approssimazione della distanza descrescono drasticamente all'aumentare di essa. (L'intensita a 20 metri e circa 100 volte minore rispetto a 2 metri).

Il ranging usa delle differenze in merito alla percezione dell'intensità del segnale:

(a) Determina quali beacons sono probabilmente più vicino, quali più lontano dal dispositivo.

(b) categorizza i beacons in quattro classi:

- IMMEDIATO (segnale forte, distanza di pochi cemtimetri).
- VICINO (segnale di media intensità, di solito uin paio di metri).
- LONTANO (segnale debole, più di un paio di metri).
- SCONOSCIUTO (difficile da dire, quando il segnale e molto debole).

Importante! : Mentre i valori ricevuti in zona di prossimità possono essere usati teoricamente per valutare una stima della distanza, in pratica tutto questo è tutt'altro che banale e richiede complessi modelli matematici per spiegare le fluttuazioni del segnale in aria: per farla breve non aspettatevi stime precise sulla distanza dai beacons.

Il trade-off

Il ranging fornisce dati piu precisi rispetto al monitoraggio, questo a discapito di scaricare piu velocemente la batteria, questo significa che di solito non è buona idea eseguire il ranging per lunghi periodi di tempo per esempio ore. Certamente non sarebbe praticabile per farlo funzionare in ogni momento.

Start ranging

Implementare un ranging è molto simile ad implementare un monitoraggio; abbiamo solo bisogno di definire una regione per ivi associare quali beacon possono essere scansionati. Diciamo che siamo interessati a tutti i beacon presenti, per fare questo possiamo definire una regione ampia con l'uso del solo UUID. Idealmente si potrebbe associare una UUID unica per tutti i beacon dell'area interessata, ma per gli scopi di questa esercitazione utilizziamo la UUID di default Estimote: `B9407F30-F5F8-466E-AFF9-25556B57FE6D` .(se avete modificato la UUID dei vostri beacon, dovrete necessariamente modificare il codice in modo appropriato).

Dal momento che le caratteristiche dei nostri luoghi food nelle vicinanze sono legate a schermate, posizioneremo un nuovo beacon manager in corrispondenza della Activity legata allo schermo. Questo ci permetterà di manipolare la vista direttamente dal Listener del ranging beacon (un modo più idiomatico sarebbe quello di avere il beacon manager che abbiamo gia nella classe Application per manipolare un modello dati ed una Activity per la vista dei cambiamenti del modello dati, invece vogliamo usare un approccio più semplificato nell'esempio).

Andiamo nel file `MainActivity` e impostiamo un secondo BeaconManager. Inoltre questa volta creeremo Regione che verra poi utilizzata successivamente all'inizio ed alla fine del ranging
Il codice va nella MainActivity:

```
private BeaconManager beaconManager;
private Region region;


@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    beaconManager = new BeaconManager(this);
    region = new Region("ranged region",
            UUID.fromString("B9407F30-F5F8-466E-AFF9-25556B57FE6D"), null, null);
}
```

Questo è il codice per avviare (quando compare l'activity sullo schermo) e stoppare (quando scompare l'activity dallo schermo) il ranging. Il codice va incluso nella MainActivity

```
@Override
protected void onResume() {
    super.onResume();

    SystemRequirementsChecker.checkWithDefaultDialogs(this);

    beaconManager.connect(new BeaconManager.ServiceReadyCallback() {
        @Override
        public void onServiceReady() {
            beaconManager.startRanging(region);
        }
    });
}

@Override
protected void onPause() {
    beaconManager.stopRanging(region);

    super.onPause();
}
```

Legare i beacon ai dati nell'app

Il nostro caso d'uso del monitoraggio era banale: mostrare una notifica all'entrata ed all'uscita dalla regione. Con il ranging il nostro obiettivo è ora quello di mostrare diverse opzioni nelle vicinanze di un terminal; il tutto comprende di legare due entità di dati insieme: identificatori di beacon e informazioni inerenti al terminale(In generale: dal momento che i beacon inviano i loro identificatori, questo approccio verra usato spesso nelle applicazioni che fanno uso di beacons).

In primo luogo abbiamo bisogno di una struttura dati per contenere informazioni e gli id dei beacon.

In secondo luogo dovremo codificare un semplice algoritmo

- Individuare il beacon più vicino.
- Cercare tutti i posti di cibo piu vicini a quel beacon.

Anche in questo caso per mantenere semplice questo tutorial sui beacon taglieremo corto.

- Useremo una lista statica di beacon id e opzioni sulle info. Idealmente si potrebbe impiegare una WEB Application.
- Ci fermeremo al calcolo della sola lista delle info, lasciando al lettore il compito di abbellire l'interfaccia grafica.

Cominciamo con l'aggiunta dei dati. Anche questo va all'interno della MainActivity

```
private static final Map<String, List<String>> PLACES_BY_BEACONS;

// TODO: replace "<major>:<minor>" strings to match your own beacons.
static {
    Map<String, List<String>> placesByBeacons = new HashMap<>();
    placesByBeacons.put("22504:48827", new ArrayList<String>() {{
        add("value1");

        add("value2");

        add("value3");

    }});
    placesByBeacons.put("648:12", new ArrayList<String>() {{
        add("value3");
        add("value2");
        add("value1");
```

```
    }});
    PLACES_BY_BEACONS = Collections.unmodifiableMap(placesByBeacons);
}
```

Abbiamo optato per una lista annidata all'interno di una mappa. La mappa "mappa" il <major:minor> associato al beacon (incapsulato in una stringa) alle opzioni memorizzate nell'ArrayList delle info associate(dal piu vicino(primo elemento) al piu lontano(ultimo elemento)). Tutto sommato niente di piu sofisticato e poco elegante, ma per questo essempio introduttivo puo andare.

Ricordate di sostituire le major e minor con quelle dei vostri beacons.

Info: Abbiamo volutamente omesso l'UUID dei beacons dal momento che l'abbiamo definita precedentemente della Region e l'abbiamo passata come parametro al metodo di startranging.

Implementeremo ora un metodo che prende come argomento un oggetto che rappresenta il beacon piu vicino e restituisce i luighi associati al beacon in ordine di vicinanza(ordinati in base alla distanza dal faro).

```
private List<String> placesNearBeacon(Beacon beacon) {
    String beaconKey = String.format("%d:%d", beacon.getMajor(), beacon.getMinor());
    if (PLACES_BY_BEACONS.containsKey(beaconKey)) {
        return PLACES_BY_BEACONS.get(beaconKey);
    }
    return Collections.emptyList();
}
```

Ranging Listener

Ricordiamo che l'elenco dei beacon è gia ordinato dal piu vicino al piu lontano in base all'intensita del segnale ricevuto.

Info: Si! Siamo molto persistensi sulla parola chiave probabile, questo perche tutte le stime fatte sulla distanza fanno riferimento alla forza del segnale che è suscettibile dalle oscillazioni, quindi di tanto in tanto potra accadere che beacon piu lontani possono essere interpretati piu vicini e viceversa. Temporaneamente queste anomalie non danno problemi per la nostra applicazione, cosa invece piu grave se si sta implemntando una applicazione in un museo o in altri contesti.

L'avere la lista gia ordinata dall'Estimote SDK e da tutto il lavoro che abbiamo fatto fin qui, il listener risulta molto semplice.

```
// find this line inside the `onCreate` method:
beaconManager = new BeaconManager(this);

// add this below:
beaconManager.setRangingListener(new BeaconManager.RangingListener() {
    @Override
    public void onBeaconsDiscovered(Region region, List<Beacon> list) {
        if (!list.isEmpty()) {
            Beacon nearestBeacon = list.get(0);
            List<String> places = placesNearBeacon(nearestBeacon);
            // TODO: update the UI here
            Log.d("str1", "str2: " + places);
        }
    }
});
```

Questo è tutto (A parte l'aggiunta dell'interfaccia utente, naturalmente). Esegui l'applicazione sul tuo dispositivo e sposta il telefono dall'uno all'altro. L'ordine dei posti stampati nella dovrebbe continuare a cambiare a seconda di qual è il beacon più vicino.

Punti Salienti

- Informazione più dettagliata rispetto al monitoraggio. I dati comprendono UUID, major, minor e stima di prossimità.
- Le stime sulla distanza non sono attendibili al 100%.
- Utilizzare startRanging e stopRanging per attivare/disattivare il ranging i risultati vengono consegnati ogni secondo. Il ranging dei risultati è un array di Beacon ordinati da quello piu vicino a quello piu lontano dal dispositivo.