ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI SCIENZE
CORSO DI LAUREA IN SCIENZE E TECNOLOGIE INFORMATICHE

iBeacon
Una nuova tecnologia per la localizzazione in ambienti chiusi

Relazione finale in:
Algoritmi e strutture dati

Relatore
Prof. Luciano Margara

Tesi presentata da
Fantini Enrico

III Sessione - Anno Accademico 2012/2013

Indice

Introduzione

Questa tesi vuole analizzare ed approfondire i concetti chiave alla base di un nuovo ed innovativo sistema di posizionamento indoor lanciato di recente sul mercato: gli iBeacon di casa Apple. Per fare questo, si partirà analizzando il Bluetooth Low Energy, ovvero la tecnologia su cui si basano gli iBeacon, presentandone sia i lati tecnici, sia le finalità applicative.

L'obiettivo principale è quello di fornire una valida raccolta di informazioni a chi si avvicina a questa novità nel campo dell'informatica e del networking, dedicando buona parte della tesi ad esempi pratici e collegamenti con la situazione attuale.

Nel primo capitolo verrà riassunta la storia nell'ambito delle telecomunicazioni che ha portato allo sviluppo di tecnologie sempre più all'avanguardia e moderne: dalla fine della Seconda Guerra Mondiale, con i primi rudimentali meccanismi a radiofrequenze, fino ai giorni nostri con il Bluetooth 4.0.

Il secondo capitolo ha l'obiettivo di fornire tutti i lati tecnici riguardanti Bluetooth Low Energy, mettendolo a confronto con i predecessori e sottolineando i suoi punti di forza, con statistiche alla mano per provarne l'efficenza e le enormi potenzialità offerte. L'architettura alla base di questa tecnologia è analizzata nel dettaglio, descrivendo i protocolli principali e le procedure di comunicazione a basso livello tra i devices.

Nel terzo capitolo l'attenzione si sposterà completamente sui veri e propri protagonisti di questa tesi, gli iBeacons. Dopo la doverosa introduzione e descrizione della tecnologia Low Energy da essi utilizzata, viene offerta un'esauriente panoramica di tutte le peculiarità di questi dispositivi. Nello specifico, è descritta innanzitutto la sfera di necessità che essi vanno a coprire, spiegandone le relazioni con i device attualmente sul mercato, e successivamente le API fornite da Apple, il framework di sviluppo delle applicazioni da presentare ai consumatori. Il lettore verrà inoltre guidato con un mini tutorial allo sviluppo di una rudimentale applicazione per saggiare le potenzialità e la semplicità d'uso di questa tecnologia.

Nel quarto capitolo verranno presi in considerazione due Case Study, ovvero due realtà che già

hanno adottato con risultati ottimi il sistema di posizionamento di iBeacon. Queste due realtà, ovvero il grande magazzino Macy's e la lega di baseball americana, Major League Baseball, sono molto differenti tra loro ma simili nella necessità di offrire un servizio mirato alla personalizzazione ed all'interattività tra il consumatore ed il servizio offerto.

Inutile dire che le possibilità ed i casi in cui questa tecnologia può portare un sostanziale miglioramento dell'esperienza dell'utente sono numerosissimi. Basti pensare a quanto al giorno d'oggi manchi un contatto interattivo col mondo reale che ci circonda in ogni momento: se prima era necessario Internet per poter effettuare una ricerca, con Bluetooth si possono ricavare le informazioni direttamente dall'ambiente circostante, senza bisogno di ulteriori collegamenti.

Va da sé che Bluetooth ed Internet si completano per fornire un'esperienza a 360 gradi all'utente, che può sentirsi davvero collegato ad un mondo digitale e reale allo stesso tempo, in maniera semplice, veloce ed interattiva.

Breve storia delle tecnologie a radiofrequenza

RFID

I modelli moderni di comunicazione wireless basati sulle frequenze radio, come NFC *(Near Field Communication)* e Bluetooth, derivano principalmente da RFID *(Radio Frequency IDentification)*, una tecnologia che traccia le sue origini negli anni della seconda guerra mondiale.

Inizialmente questa tecnologia era rivolta quasi esclusivamente all'ambito militare, ma con il passare del tempo, e già dagli anni '60, i ricercatori del Los Alamos National Laboratory lavorano sotto richiesta degli organi governativi americani ad implementazioni di RFID negli ambiti più svariati. I transponder (abbreviativo di *Trans*mitter res*ponder*), ovvero i primi dispositivi a basse frequenze (30 kHz - 300 kHz) progettati per le comunicazioni con RFID venivano installati su autoveicoli per il riconoscimento ravvicinato, ed ancora oggi sono utilizzati ai caselli autostradali per i pagamenti automatici.

Le compagnie produttrici di questi dispositivi, dopo aver commercializzato sistemi a basse frequenze, si spostarono poi su frequenze più alte che all'epoca erano ancora inutilizzate in gran parte del mondo. La frequenza 13.56 MHz è ancora usata per alcuni sistemi di pagamento, smart card e sistemi di controllo d'accesso. Solo nei primi anni '90 gli ingegneri IBM iniziano a lavorare con frequenze ultra-alte (300 MHz - 3 GHz) che permettevano una comunicazione tra dispositivi più veloce ed a distanze fino a 6 metri.

Subito dopo i primi test sul campo però, a causa di una situazione economica pessima dell'azienda, il progetto venne abbandonato e mai commercializzato. I brevetti vennero venduti ad un'altra compagnia, la Intermec, che tentò di diffondere questa tecnologia, ma a causa della mancanza di standard internazionali e dell'elevato costo di produzione anche questo tentativo di commercializzazione si rivelò fallimentare.

RFID ebbe una ripresa notevole nel 1999, quando due professori del Massachusetts Institute of Technology, David Brock e Sanjay Sarma, svilupparono un'idea per ridurre i costi di produzione dei complessi microchip all'interno dei dispositivi. La loro idea era quella di trasferire le informazioni dell'oggetto associato al chip su Internet, mantenendo all'interno del dispositivo solo un ID univoco chiamato tag. In questo modo si veniva a tagliare drasticamente la quantità di memoria necessaria e le informazioni corrispondenti ad ogni tag venivano rese accessibili tramite un database sulla rete.

Questa idea rivoluzionò il modo di vedere RFID. I tag che in origine non erano altro che database mobili allegati all'oggetto a cui facevano riferimento ora diventavano i “collegamenti” tra l'oggetto fisico ed il database sulla rete Internet che ne conteneva le informazioni associate.

Per le aziende questo fu una grande svolta positiva: tra il 1999 ed il 2003 più di 100 compagnie si dissero interessate a questa tecnologia. L'Auto-ID Center, la sezione del MIT che si occupava del progetto, sviluppò protocolli standard per la comunicazione con RFID, un sistema di numerazione per le esigenze commerciali delle compagnie (*EPC, Electronic Product Code*) ed un'architettura di rete apposta per consultare su Internet i dati associati ad un tag RFID. La tecnologia venne brevettata ed in poco tempo utilizzata a fianco dei codici a barre, tanto che dal 2003 è commercializzata con successo anche tra alcune delle più grandi catene di rivenditori negli Stati Uniti. Nel 2004 viene stabilito un nuovo standard ancora più dettagliato che aprirà la strada ad un utilizzo globale di RFID.

NFC

Nel 2002, Sony e Philips annunciano con un comunicato stampa la loro intenzione di cooperare alla creazione di uno standard di nuova generazione per la comunicazione a basso raggio tramite radiofrequenze: nasce ufficialmente il 5 Settembre di quell'anno la tecnologia denominata Near Field Communication, o NFC. Data la rapida crescita di questo ambito, anche una delle aziende leader nella telefonia, la Nokia, si unisce alla realizzazione di questo progetto e nel 2004 viene fondato l'NFC Forum, un'associazione di aziende che promuove la standardizzazione e l'implementazione di questa tecnologia.

Solo due anni dopo, nel 2006, viene ratificato il primo set di specifiche per i tag NFC. Come in RFID, le tag non sono altro che piccoli allegati, traducendo letteralmente dall'inglese “cartellini” che accompagnano un oggetto. Un dispositivo compatibile con NFC sarà dunque in grado di leggere tali informazioni se messo a contatto con il tag: quest'ultimo potrà, date le ridottissime dimensioni, venir posizionato a fianco di opere d'arte, merce in vendita od altro e fornire

informazioni utili su ciò che l'utente ha di fronte in tempo reale. Sempre nel 2006 inizia la diffusione di NFC anche nei telefoni Nokia come il Nokia 6131, il primo a supportare questa nuova tecnologia.

Sostanzialmente, NFC altro non è che una “specializzazione” standardizzata della tecnologia RFID, in alta frequenza (13,56 MHz), orientata ad uno scambio rapido e sicuro di informazioni a brevissima distanza.

Col passare degli anni sono state definite nuove specifiche che delineano i più svariati e quotidiani casi d'uso, come la visione di video in streaming, la condivisione di link testuali, lo shopping interattivo e alcuni tipi di giochi multiplayer.

Bluetooth e Bluetooth LE

Bluetooth è il nome di una tecnologia ideata dalla multinazionale svedese Ericsson nel 1994, inizialmente progettata per essere un'alternativa wireless ai cablaggi RS-232. L'idea di base era quella di fornire un metodo rapido e semplice per collegare un telefono cellulare ad una serie di accessori come per esempio auricolari o vivavoce per automobili. Quattro anni dopo viene fondato il Bluetooth Special Interest Group, un'associazione non-profit composta da corporazioni del calibro di IBM, Toshiba, Intel ed ovviamente Ericsson, che formalizza sempre nel 1998 le prime specifiche. Al giorno d'oggi, il Bluetooth SIG, che conta più di 19.000 aziende, è proprietario e licenziatario del marchio Bluetooth: per progettare un dispositivo a norma e standardizzato è necessario dunque sottoporlo all'approvazione dello stesso.

La lenta ma inesorabile diffusione della tecnologia Bluetooth va di pari passo con i suoi miglioramenti: in circa una decade vengono rilasciate versioni sino alla 3.0, tutte strettamente retrocompatibili. L'ultima versione, la 4.0 rilasciata nel 2010, introduce significativi miglioramenti.

Già dal 2001 infatti Nokia sviluppa un prototipo di tecnologia con l'obiettivo di minimizzare i costi di produzione ed i consumi energetici dei dispositivi, decidendo di standardizzare tale tecnologia sotto la bandiera Bluetooth. I risultati di questa ricerca, che nel 2006 prendeva il nome di Wibree, spinse il Bluetooth SIG a raggiungere un accordo ed integrando Wibree, sotto il nome di Bluetooth Low Energy Technology, nelle specifiche di Bluetooth 4.0. Quest'ultima versione di Bluetooth è focalizzata esclusivamente sul basso consumo energetico, cambiando completamente direzione al processo di evoluzione intrapreso finora: il punto focale dell'innovazione infatti non sarà più la velocità di trasferimento e la distanza di copertura, ma appunto il dispendio energetico,

insieme alla sicurezza dei trasferimenti ed alla rapidità di setup.

Questo approccio apre la strada ad innumerevoli casi d'uso per questa tecnologia: i dispositivi compatibili, i cosiddetti “Beacon”, sfruttano i vantaggi e il lato low-cost del Low Energy per poter essere utilizzati nei contesti più svariati.

Non tutti i devices che supportano Bluetooth 4.0 sono retrocompatibili, poiché come già detto, rivoluzionano in toto l'architettura alla base della comunicazione: se prima in una comunicazione tra due devices vi erano solo i ruoli di Master e Slave con assegnazione dinamica, il Low Energy definisce altri ruoli a seconda dell'utilizzo e del compito del device stesso, pur mantenendo i due dei predecessori. L'ampiezza del gruppo di dispositivi che possono comunicare contemporaneamente non è più limitata dal Master (il limite precedentemente era 7 Slave per ogni Master) ma è definita durante l'implementazione e variabile a seconda dei casi.

Bluetooth Low Energy: caratteristiche tecniche

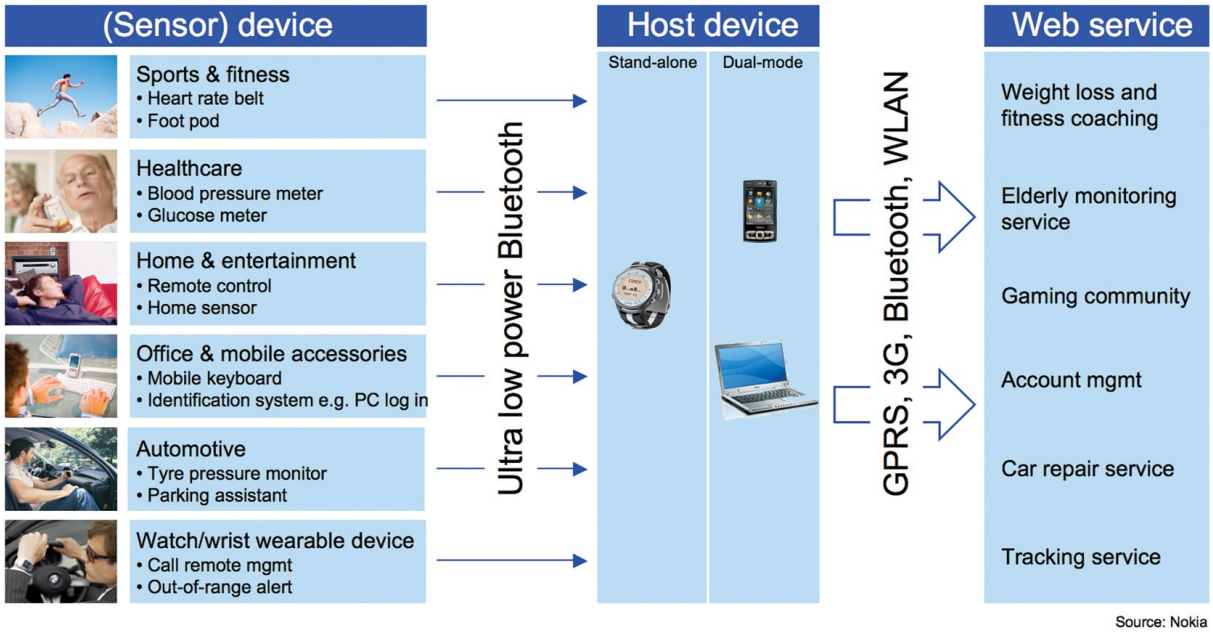
Overview

Le novità introdotte dalla versione 4.0 di Bluetooth portano sul mercato due tipologie di dispositivi che vanno a coprire quell'area di necessità nelle quali i suoi predecessori non riuscivano ad essere ottimali. In breve, un dispositivo che utilizza la tecnologia Low Energy è:

- Fisicamente piccolo: è possibile "applicarlo" praticamente ovunque date le dimensioni ridotte.
- Rapido nel setup: il tempo di discovery è di pochi millisecondi.
- Longevo: il basso consumo di batteria e l'autonomia elevata fanno sì che il dispositivo non abbia alcuna necessità di manutenzione periodica.
- Dotato di una portata flessibile: il range seppur inferiore a quello dei predecessori è comunque buono, ed è possibile misurare la distanza effettiva tra gli interlocutori, permettendo la localizzazione indoor.
- Utile nella quotidianità: è già diffuso in alcune grandi catene di centri commerciali e negozi di elettronica (di cui parleremo più avanti nei Case Study) come ausilio allo shopping.

  Con l'introduzione del Low Energy nel panorama Bluetooth, si è ritenuto necessario suddividere le tipologie di devices essenzialmente in 3 categorie: Bluetooth Classic, Bluetooth Smart e Bluetooth Smart Ready.

I dispositivi Smart Ready, come si può notare dalla figura, sono al centro di questo "ecosistema": essi altro non sono che smartphone, tablet, palmari e laptop, che grazie ad uno stack di protocolli dual-mode possono collegarsi ai devices di vecchia e di nuova generazione. I dispositivi Smart invece sono il vero e proprio cuore dell'innovazione: sensori, strumenti di misurazione, dispositivi di posizionamento ed altro ancora, con consumi bassissimi e capaci di inviare piccole quantità di dati a comando, una volta connessi ad un device Smart Ready. Importante notare che i devices Smart non sono retrocompatibili e possono comunicare solo con devices Smart Ready.



La documentazione ufficiale e gli standard definiti offrono un profilo generico che viene modellato per ogni utilizzo specifico dei devices. Alcuni esempi di archetipi possono essere i profili per il riconoscimento di prossimità od addirittura per la misurazione del battito cardiaco: il dispositivo di misurazione medico utilizza Bluetooth LE per trasmettere dati comprensibili ad un secondo dispositivo di controllo mediante un protocollo di comunicazione, il quale è ben definito nelle specifiche in modo da essere compatibile con ogni device a norma.

I ruoli supportati e predefiniti nelle specifiche di Bluetooth 4.0 sono i seguenti:

- Broadcaster
- Observer
- Peripheral
- Central

Ogni ruolo richiede degli specifici comportamenti da parte del controller sottostante: un controller infatti può venire configurato per assumere più ruoli, ma mai due nello stesso momento.

Il ruolo di Broadcaster è ottimo per gli oggetti che hanno l'unica funzione di trasmettere dati, senza supportare connessioni. Il Broadcaster usa notifiche in advertising per trasmettere le informazioni ai devices in ascolto.

Il ruolo di Observer è complementare al Broadcaster, essendo un ruolo di sola ricezione. La connessione non è supportata, ed il device rimane semplicemente in ascolto degli advert circostanti

Il ruolo di Peripheral è ottimizzato per i devices che supportano una connessione singola e sono meno complessi dei devices centrali. I devices Peripheral necessitano di un Controller che supporti il ruolo di Slave.

Il ruolo Central è il più complesso: è l'unico che supporta connessioni multiple ed è l'unico ad iniziarne di nuove con i device del ruolo Peripheral. Viene richiesto un Controller che supporti il ruolo di Master.

Tech specs di BLE

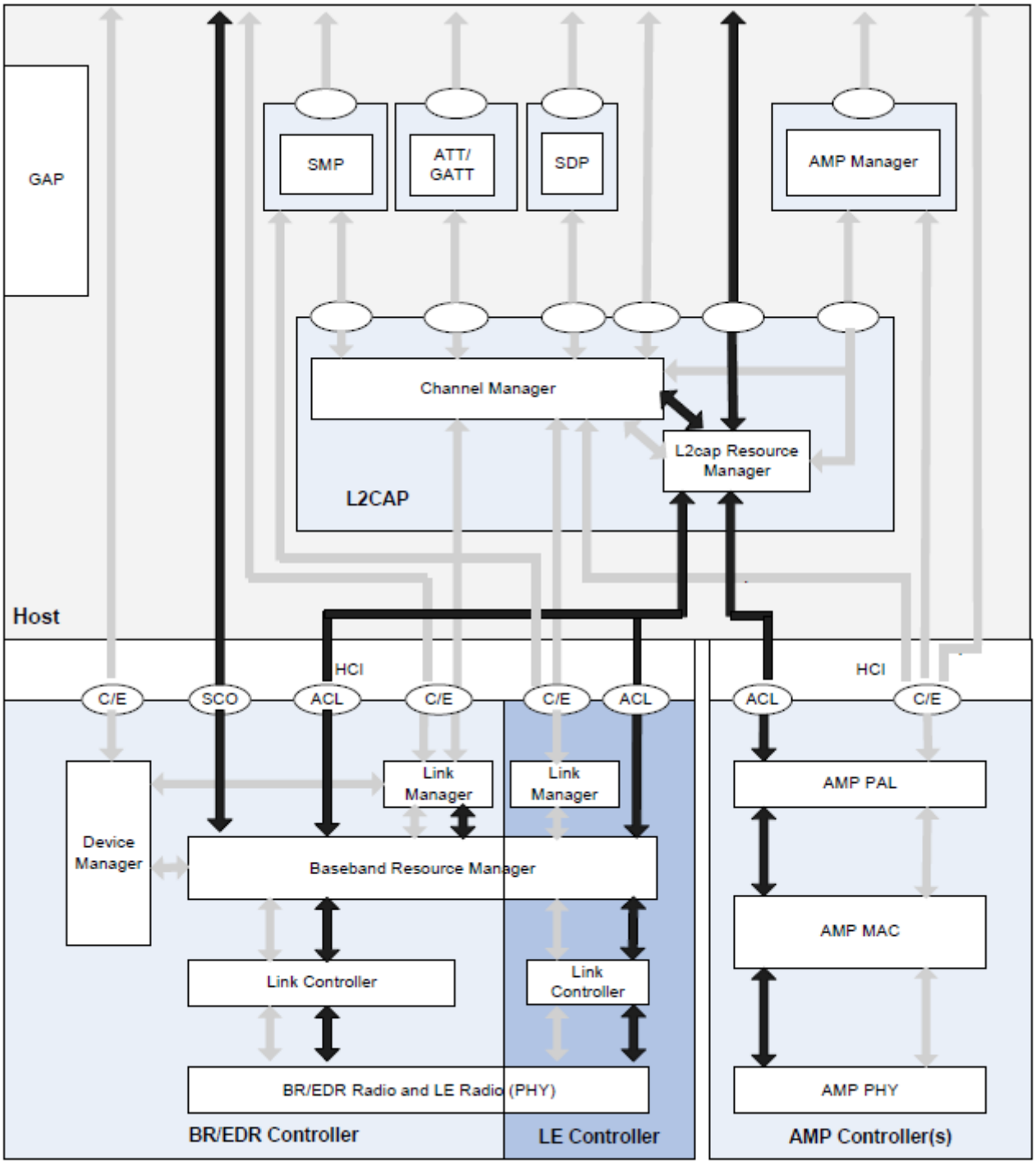
Abbiamo già accennato alle differenze tra Bluetooth Classic e Bluetooth Low Energy, ma per definire meglio le specifiche di quest'ultimo è utile metterli a confronto per poter discernere al meglio anche tutti i punti in comune.

Specifica	Bluetooth Classic	Bluetooth Low Energy
Raggio d'azione	~ 100 m	~ 50 m
Velocità ottimale attraverso l'aria	1–3 Mbit/s	1 Mbit/s
Throughput delle applicazioni	0.7–2.1 Mbit/s	0.27 Mbit/s
Slave attivi	7	Dipende dall'implementazione
Sicurezza	56/128-bit e definita dall'applicazione	128-bit AES con Counter Mode CBC-MAC e definita dall'applicazione
Robustezza	Adaptive fast frequency hopping, FEC, fast ACK	Adaptive frequency hopping, Lazy Acknowledgement, 24-bit CRC, 32-bit Message Integrity Check
Latenza (da uno stato disconnesso)	Tipicalmente 100ms	6ms
Comunicazione vocale	Sì	No
Topologia del network	Scatternet	Star-bus
Consumo	1 (preso come riferimento)	da 0.01 a 0.5 a seconda dell'uso
Consumo di corrente massimo	<30mA	<15mA
Service Discovery	Sì	Sì
Profili	Sì	Sì
Casi d'uso principali	Telefoni cellulari, gaming, cuffie audio, streaming audio, accessori auto, etc.	Telefoni cellulari, gaming, orologi, sport e fitness, sicurezza e rilevazione di prossimità, etc.

Come i predecessori, il segnale radio di BLE opera sulla frequenza 2.4 GHz della banda ISM. Il sistema Low Energy impiega un transceiver a frequency hopping[1] per combattere le interferenze tra devices in prossimità. Le frequenze vengono modulate in maniera pseudocasuale e binaria per minimizzare la complessità del transceiver. Anche la struttura dei comandi in BLE è semplificata: l'Host Control Layer offre infatti circa 50 comandi, molti di meno se confrontati ai 180 comandi/eventi presenti in una completa implementazione di Bluetooth.

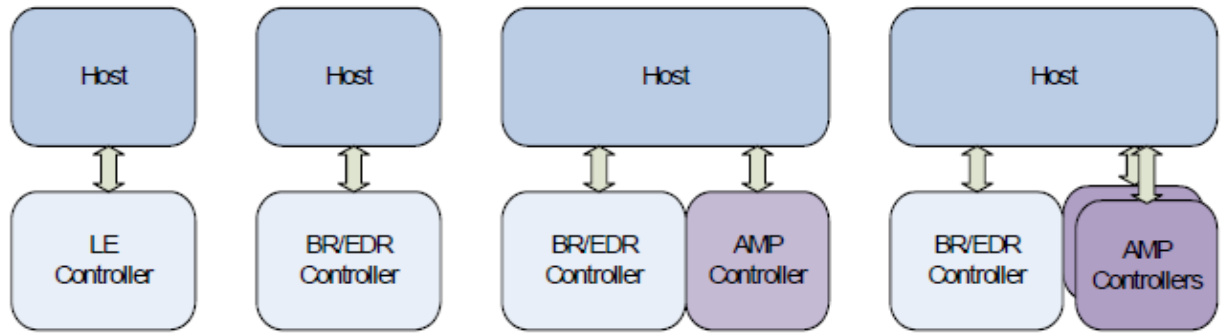
[1] *Frequency Hopping*: tecnica che consiste nel variare la frequenza di trasmissione ad intervalli regolari.

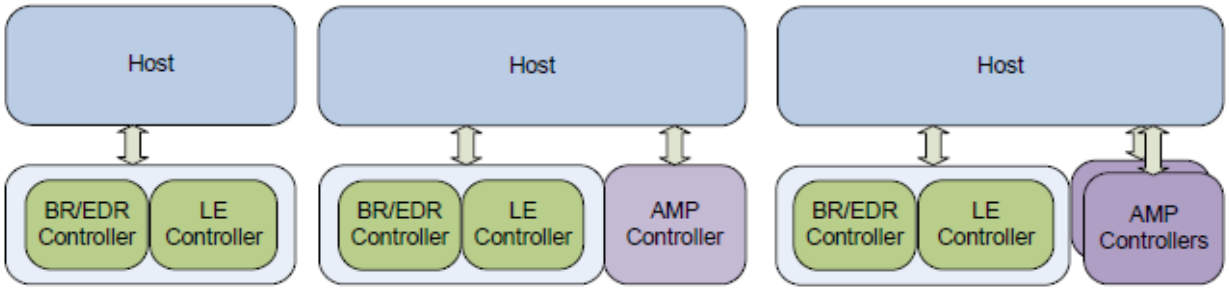
Architettura e protocolli



Il Core System di Bluetooth consiste in un Host, un Controller Primario e zero o più Controller Secondari. Un'implementazione minima di Bluetooth Low Energy copre i quattro layer più bassi, associati ai protocolli a livello di Controller, in associazione con i protocolli di Security Manager (SMP), Attribute (ATT) a livello di Host. Le specifiche richieste per i profili sono delineate nel Generic Attribute Profile (GATT) e nel Generic Access Profile (GAP).

Le implementazioni che uniscono Bluetooth Low Energy a Bluetooth Classic, come nel caso dei device Smart Ready, hanno inoltre bisogno di implementare anche la sezione di protocolli relativa alla versione Basic Rate/Enhanced Data Rate *(BR/EDR, cioé corrispondente alle vecchie generazioni)* .





Andremo ora a descrivere nel dettaglio alcuni dei “blocchi” più importanti:

- **SMP, Secure Manager Protocol**

SMP è il protocollo peer-to-peer usato per generare, gestire ed assegnare le encryption keys e le identity keys. Questo blocco si interfaccia direttamente col Controller per fornire le keys di identificazione e criptazione usate nelle procedure di autenticazione durante l'associazione dei devices. Importante notare che questo blocco si trova solo nei sistemi Low Energy e fa parte dell'Host, mentre la sua controparte nei sistemi BR/EDR è contenuta nel Link Manager del Controller. Questo cambiamento fa sì che i costi di implementazione di Controller esclusivamente LE risultino più bassi.

- **ATT, Attribute Protocol**

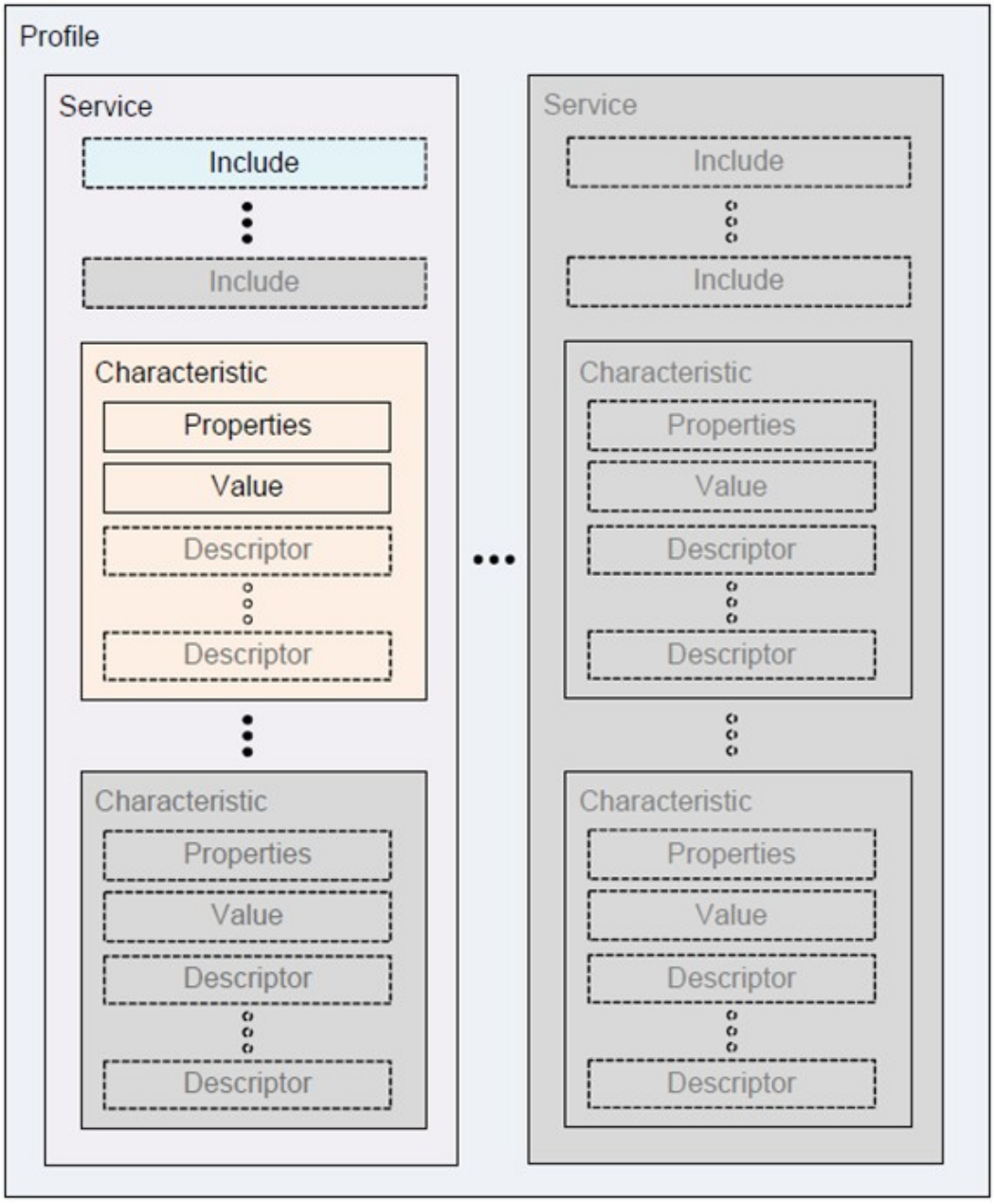
Il blocco ATT implementa il protocollo peer-to-peer tra un attribute server ed un attribute client. Il client apre la comunicazione verso il server su un device remoto attraverso un canale L2CAP, inviando comandi, richieste e conferme. Il server può replicare con risposte, notifiche, indicazioni. Questo servizio permette di utilizzare un device client ATT per leggere e scrivere dati su un device server.

- **AMP Manager Protocol**

Alternative MAC/PHY, o AMP, è una caratteristica introdotta con Bluetooth 3.0, che permette l'utilizzo di canali ad alta velocità (tipicamente il Wi-Fi) per supportare lo scambio ad alta velocità di grosse quantità di dati. Il Manager Protocol è un layer col compito di fare discovery di AMP Manager di dispositivi remoti ed eventualmente collezionare informazioni utili per poter avviare una connessione fisica tra i devices. L'AMP Manager utilizza un canale di segnalazione L2CAP dedicato.

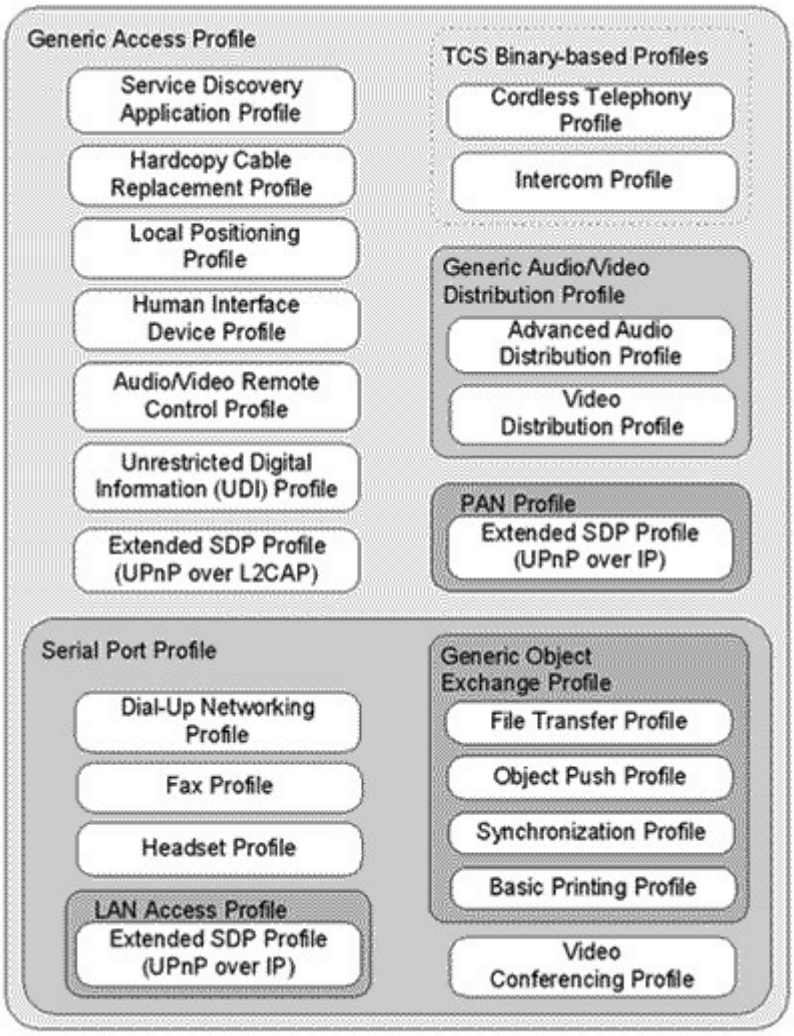
- **GATT, Generic Attribute Profile**

Il Generic Attribute Profile definisce la relazione client-server attraverso la quale il device server comunica i dati di cui dispone (per esempio, le misurazioni effettuate da un sensore), raggruppate in contenitori logici chiamati Services. Alcune caratteristiche sono di sola lettura, altre sono disponibili in scrittura con lo scopo di configurare il server a seconda dell'utilizzo. Ogni caratteristica, oltre al proprio valore, può avere uno o più descrittori, utilizzati per definirne il comportamento e per modulare meglio la configurazione. Il blocco contiene le interfacce per fare discovery, leggere e scrivere le caratteristiche del profilo in uso dal device server. GATT è usato nei dispositivi LE per definire i profili in uso su device remoti.



- **GAP, Generic Access Profile**

Il blocco Generic Access Profile rappresenta le funzionalità di base comuni a tutti i device Bluetooth, come per esempio le modalità e procedure d'accesso usate dai trasporti, dai protocolli e dai profili delle applicazioni. I servizi di GAP includono il discovery di device, discovery di servizi, modalità di connessione, sicurezza, autenticazione e modelli per l'associazione di device.



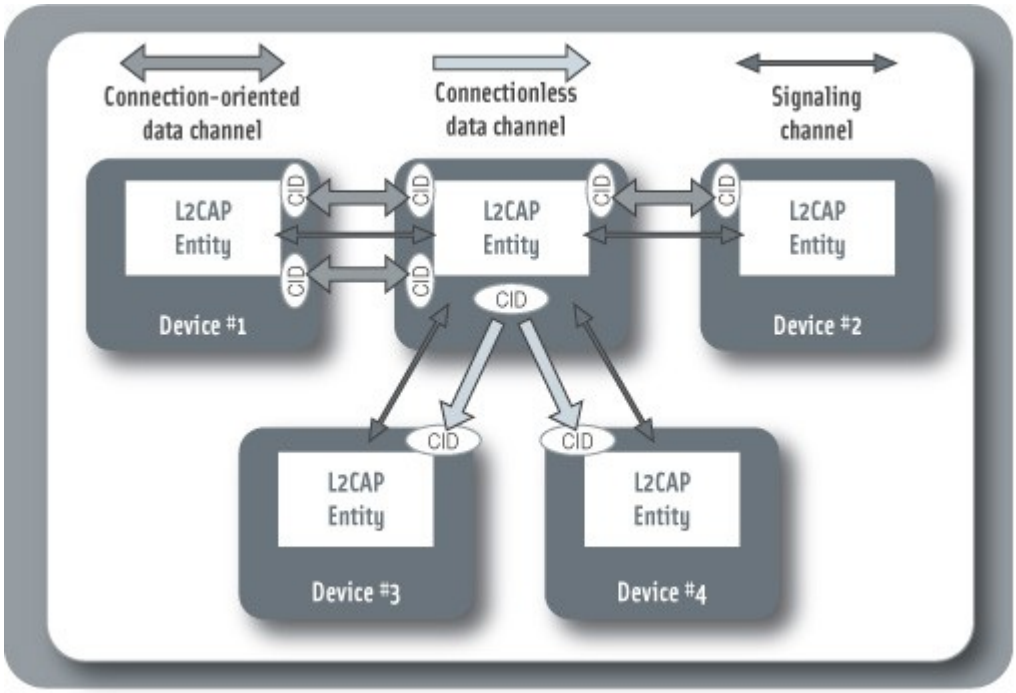
- **SDP, Service Discovery Protocol**

SDP è utilizzato dai dispositivi per richiedere i servizi offerti da un device remoto. Questo protocollo non fornisce l'accesso diretto, ma semplicemente informazioni sui dettagli di tali servizi, descritti tramite UUID.

- **L2CAP, Logical Link Control and Adaptation Protocol**

L2CAP è il protocollo principale a livello di collegamento. Tra le sue funzionalità troviamo:

- Multiplexing dei dati tra protocolli differenti di livello più alto.
- Segmentazione e riassemblamento dei pacchetti.
- Più modalità di comportamento: oltre a quella standard, esso può operare in modalità di controllo di flusso
- Provvede alla gestione di trasmissioni multicast one-way, cioé da un device singolo di partenza a molti.



Questo protocollo lavora mediante canali, identificati dai Channel ID per permettere a protocolli di livello più alto di ricevere pacchetti dai livelli sottostanti. I canali possono essere sia orientati alla connessione, sia “connectionless”, ovvero orientati a gestire una comunicazione unidirezionale.

- **Link Manager**

Il Link Manager è responsabile della creazione, modifica e rilascio dei link logici, comunicando con altri Link Manager su device remoti utilizzando (nell'architettura Low Energy) il Link Layer Protocol (LL).

- **Baseband Resource Manager**

Il Baseband Resource Manager è responsabile di tutti gli accessi al canale radio. Ha due funzioni principali: lo scheduling dei tempi per l'accesso al canale fisico da parte delle entità di livello superiore e la negoziazione di dei permessi all'accesso con queste ultime.

- **PHY**

Il blocco PHY è responsabile della trasmissione e ricezione dei pacchetti sul canale fisico. Essenzialmente è il collegamento più a basso livello, che trasforma gli stream di dati in uscita ed in entrata al canale fisico nei formati corretti.

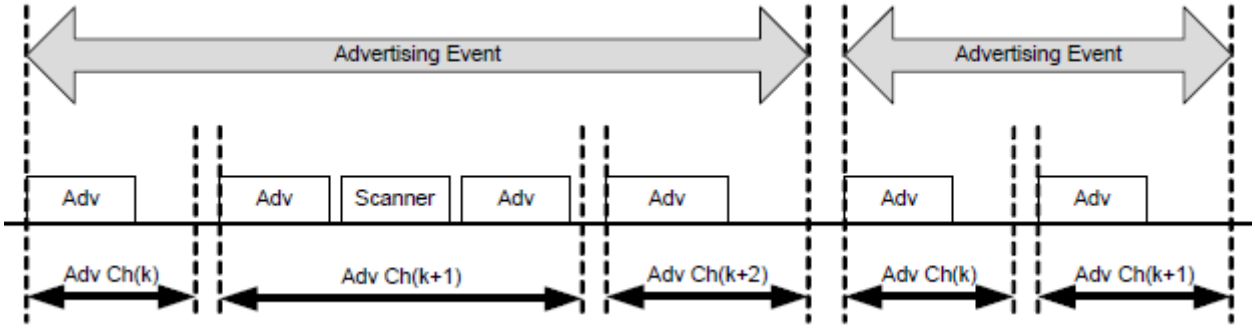
Connessioni tra devices: indirizzi, pacchetti ed operazioni

- **Overview di un'operazione tra devices**

Bluetooth LE impiega due schemi di accesso multiplo per la condivisione del canale di comunicazione: Frequency Division Multiple Access (FDMA) e Time Division Multiple Access (TDMA). Quaranta (40) canali fisici, separati da 2MHz sono utilizzati nell'FDMA, 3 come canali di advertising e 37 come canali di scambio dati. Lo schema TDMA invece è determinato dall'invio di pacchetti tra device ad intervalli di tempo predefiniti.

Il canale fisico è suddiviso in unità di tempo chiamate eventi, che si suddividono in due tipi: Advertising Events e Connection Events. I dati sono trasmessi tramite pacchetti posizionati in questi eventi.

I devices che trasmettono pacchetti di advertising nei canali fisici sono detti advertisers. I devices che invece ricevono questi pacchetti, senza l'intenzione di aprire una connessione sono detti scanners.

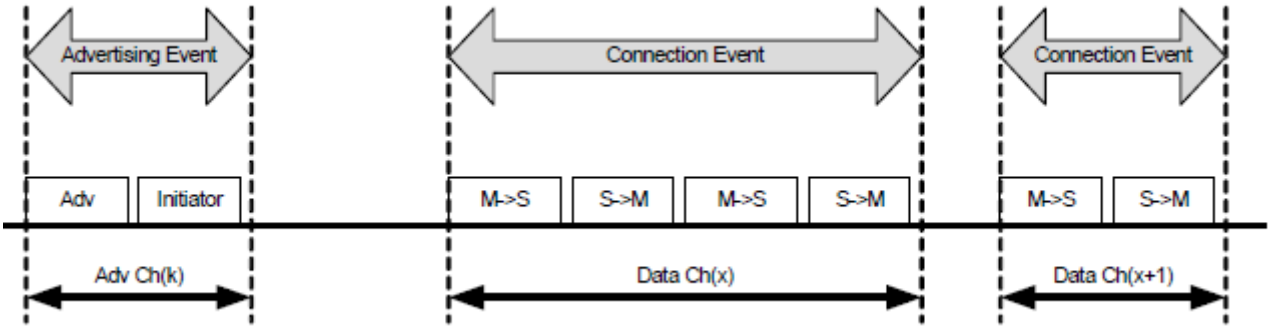


Durante un Advertising Event, il device advertiser invia pacchetti attraverso un canale fisico. Uno scanner può eventualmente, dopo aver ricevuto il pacchetto, inviare una richiesta all'advertiser utilizzando lo stesso canale fisico dove è stato inviato il primo pacchetto di advert. L'advertiser a sua volta può rispondere sempre sullo stesso canale di advert allo scanner che inoltra la richesta. Il seguente pacchetto di advertising destinato al broadcast sarà inviato successivamente su un canale differente.

I devices LE possono agire limitatamente nei canali di Advertising se il loro scopo è semplicemente una connessione unidirezionale od in broadcast.

I dispositivi che hanno invece bisogno di formare una connessione ad un altro device remoto devono rimanere in ascolto sul canale per ricevere dei pacchetti che permettono di avviare un collegamento: questi dispositivi prendono il nome di initiators.

Se l'advertiser sta inviando un pacchetto che permette la connessione, un initiator può inoltrare una richiesta utilizzando lo stesso canale fisico sul quale ha ricevuto il pacchetto. L'Advertising Event termina nel momento in cui l'advertiser riceve ed accetta la richiesta di connessione. Una volta che la connessione è stabilita, si ha che nel seguene Connection Event l'initiator diventa il device master della piconet ed il device advertiser diventa uno slave. I Connection Events sono utilizzati per l'invio di pacchetti tra il master e gli slave, utilizzando il channel hopping all'inizio di ogni periodo di tempo. Il master è colui che inizia e termina tutte le connessioni.

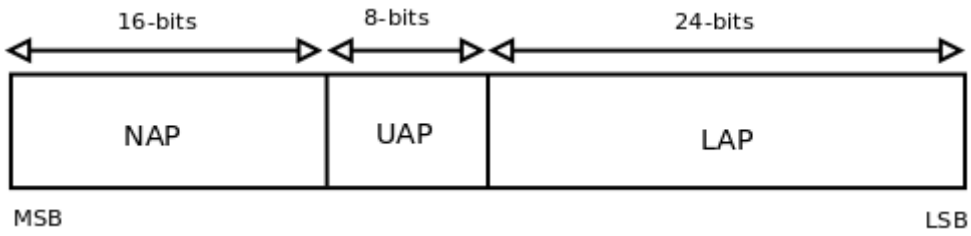


I devices di una stessa piconet utilizzano una sequenza specifica per cambiare frequenza: tale sequenza è determinata algoritmicamente da un campo particolare nella richiesta di connessione inviata dal device initiator, che poi diventerà il master della piconet. Viene provvisto agli slave della piconet un intervallo di cambio frequenza, un ordine pseudocasuale delle 37 frequenze della banda ISM.

Quest'ordine può essere adattato per escludere alcune frequenze che andrebbero a collidere con altri utilizzatori delle stesse frequenze, rendendo Bluetooth LE capace di coesistere con altre tecnologie che comunicano sullo stesso range di frequenze.

- **Indirizzi**

Ogni device dispone di un indirizzo univoco a 48 bit, suddiviso come nella figura sottostante.



  - LAP, Lower Address Part: è la porzione di 24 bit allocata dal costruttore, e forma parte dell'Access Code che precede l'header nei pacchetti trasmessi da Bluetooth.
  - UAP, Upper Address Part: insieme al NAP forma l'OUI, Organizationally Unique Identifier, ovvero il codice identificativo che secondo le registrazioni effettuate presso l'IEEE è assegnato al costruttore del device. Lo UAP è utilizzato anche, tra le altre cose, a generare l'Header Error Correct, un field utilizzato per identificare errori nei pacchetti Bluetooth.
  - NAP, Non-significant Address Part: I rimanenti 16 bit dell'OUI. Il NAP, come dice il nome, non è utilizzato nel networking di Bluetooth.

- **UUID**

Nell'architettura Bluetooth, oltre all'indirizzo proprio del device, per identificare i servizi offerti da un server, i profili di un certo device ed altro si utilizzano degli UUID, o Universally Unique IDentifier.

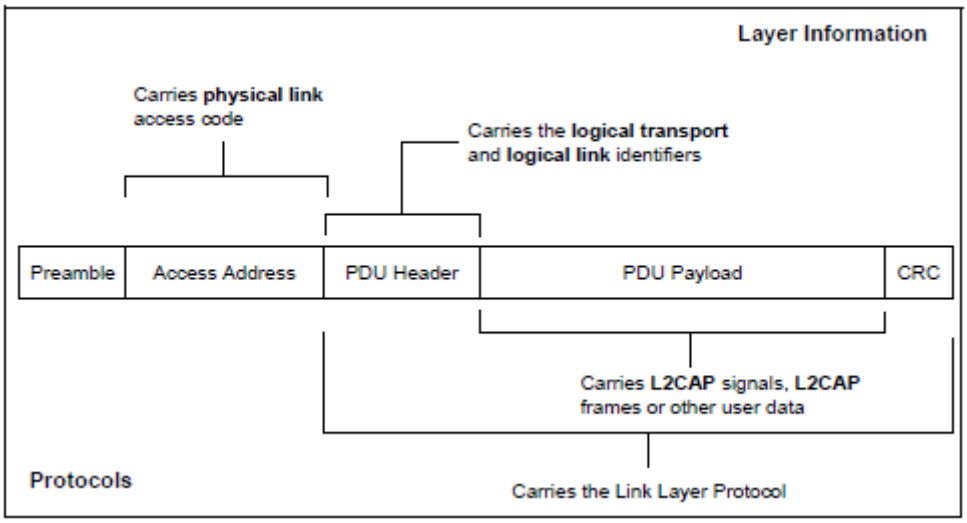
Questi UUID, valori di 128 bit, rimangono unici globalmente e nel tempo, e possono venire creati indipendentemente in maniera distribuita per concetti. Non esiste un registro centrale degli UUID, ma vi è un range pre-allocato per gli UUID assegnati tipicamente a funzionalità comuni. Ogni entità della tecnologia Bluetooth ha un UUID per identificarla univocamente e nella tabella sottostante, oltre a quello base Bluetooth, se ne possono trovare alcuni esempi.

Entità	UUID
Base Bluetooth	**00000000-0000-1000-8000-00805F9B34FB**
Generic Access	0000**1800**-0000-1000-8000-00805F9B34FB
GATT	0000**1801**-0000-1000-8000-00805F9B34FB
Immediate Alert	0000**1802**-0000-1000-8000-00805F9B34FB
Health Thermometer	0000**1809**-0000-1000-8000-00805F9B34FB
Heart Rate	0000**180D**-0000-1000-8000-00805F9B34FB
Battery Service	**0000180F-0000-1000-8000-00805F9B34FB**
Battery Level	**00002A19-0000-1000-8000-00805F9B34FB**
Blood Pressure	0000**1810**-0000-1000-8000-00805F9B34FB
Current Time	0000**1805**-0000-1000-8000-00805F9B34FB
TX Power	0000**1804**-0000-1000-8000-00805F9B34FB

Questi identificatori univoci possono essere anche utilizzati durante i procedimenti di advertising insieme all'indirizzo del device ed un “Friendly Name”, ovvero un titolo facilmente leggibile da un utente per definire le funzionalità dell'entità a cui fa riferimento.

- **Pacchetti**

I pacchetti scambiati a livello di collegamento tra device Bluetooth Low Energy hanno una struttura generica ben definita nelle specifiche.



Si noti come nel pacchetto a livello di link non sono presenti identificatori del canale fisico sottostante, poiché questi sono determinati al momento del setup della connessione. L'Access Address è invece sempre presente, per identificare la comunicazione attiva sul collegamento fisico e per escludere ed ignorare pacchetti che utilizzano lo stesso canale fisico (sono quindi fisicamente in prossimità) ma hanno un differente collegamento. L'Access Address determina inoltre se il pacchetto è destinato al collegamento per l'advertising od ad un device ricevente, in quanto tutti i collegamenti per l'advertising usano un Access Address fisso mentre i collegamenti tra devices LE usano un valore a 32-bit casuale come Access Address. In questo modo vi possono essere un elevato numero di indirizzi, e quindi di devices collegati contemporaneamente, in una singola piconet[1].

Tutti i pacchetti LE presentano inoltre un header PDU. Questo header determina il tipo di broadcast o collegamento logico trasportato sul canale fisico.

Per i canali di advertising, l'header PDU contiene il tipo di payload, la lunghezza del payload e genericamente anche l'indirizzo del device che effettua l'advertising. A seconda dello scopo del device, il payload PDU può contenere informazioni per la richiesta di collegamento ed i dati per il setup della connessione.

[1] *Piconet:* rete di devices che occupano lo stesso spazio fisico. Vi è sempre un device definito come Piconet Master e tutti i restanti sono collegati a quest'ultimo col ruolo di slaves.

- **Piconet e canale fisico**

Nel sistema Low Energy, due devices devono utilizzare un canale fisico condiviso per comunicare tra loro. Per far sì che questo possa verificarsi, i rispettivi transceiver devono essere sintonizzati sulla stessa frequenza ed ovviamente devono trovarsi non oltre una certa distanza l'uno dall'altro.

Siccome il numero di frequenze PHY è limitato, e molti device Bluetooth possono operare indipendentemente nella stessa area di spazio, è estremamente probabile che due di questi devices possano essere sintonizzati sulla stessa frequenza, risultando in una collisione. A differenza del sistema BR/EDR dove viene utilizzato un access code per identificare una piconet, il sistema LE utilizza un Access Address generato casualmente per identificare una piconet fisica.

Vengono definiti due principali canali fisici di comunicazione, ognuno ottimizzato per una specifica funzione: il canale della piconet principale, che viene utilizzato per la comunicazione tra i devices connessi alla piconet stessa ed il canale per gli advertisement in broadcast, utilizzato appunto per comunicazioni destinate a tutti i devices contemporaneamente. Quest'ultimo canale ha grande importanza in quanto è utilizzato in fase di discovery per fare il primo setup della connessione di un device che tenta di connettersi alla rete e per trasmettere informazioni in broadcast tra device non collegati direttamente.

Ogni devices può utilizzare solo uno dei due canali alla volta: le operazioni concorrenti su più canali sono supportate grazie all'utilizzo di un multiplexing basato sulla suddivisione del tempo, cosìcché un device possa supportare la connessione e contemporaneamente inviare messaggi in broadcast.

La topologia di queste piconet definisce un marcato rapporto di master-slave tra i device. Solo un device della piconet è definito Master, mentre tutti gli altri collegati alla stessa sono definiti Slave: virtualmente non vi è limite al numero di Slave per ogni piconet, anche se fisicamente tale limite è imposto dalle limitazioni fisiche del Master. La comunicazione avviene esclusivamente tra il device Master ed uno dei device Slave.

Il canale di advertising al contrario è condiviso da un qualsiasi numero di devices LE, senza distinzioni: ognuno può trasmettere pacchetti in broadcast ed ogni device di tipo scanner nell'area circostante può riceverli.

Beacons: come utilizzarli

Cos'è un Beacon?

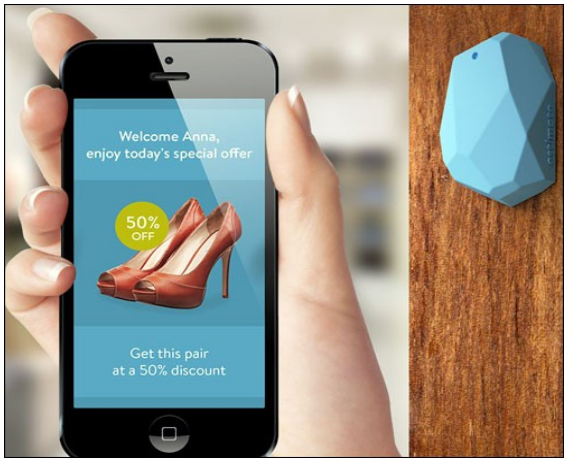
Dopo aver parlato in maniera tecnica della nuova tecnologia Bluetooth LE è naturale chiedersi dove ed in che modo questa entrerà nella vita di tutti i giorni. Se si pensa alla lista dei device abilitati alla tecnologia Bluetooth 4.0, come per esempio ogni iPhone dalla versione 4S in avanti, iPad dalla terza generazione in avanti, gli smartphone della serie Samsung Galaxy, Sony Xperia e Google Nexus 4 e Nexus 5 ed altri ancora, si può constatare che l'innovazione portata dal Low Energy è *già* più che avviata.

I "Beacon", ovvero dispositivi classificati Bluetooth Smart, dotati di tutte le caratteristiche di cui abbiamo discusso, sono già in commercio: è davvero facile pensare alla moltitudine di applicazioni possibili ed al vasto pubblico a cui sono rivolti. Un Beacon può essere programmato per aiutare il cliente di un negozio nello shopping, per guidare uno spettatore al suo posto in un cinema, per effettuare ordinazioni rapide al bar, o semplicemente per raccogliere informazioni sull'ambiente di dispositivi circostanti. Le distanze sono genericamente distribuite in 3 settori, entro i 50 cm, tra i 50cm e i 5 metri, e tra i 5 metri in su. Il range massimo dipende molto anche dall'architettura degli edifici in cui i Beacons vengono usati, in quanto il segnale radio è disturbato dalla presenza di muri e barriere fisiche artificiali.

Un altro concetto importante è quello della privacy e della sicurezza: per evitare un hijacking dei dispositivi, od un utilizzo fraudolento dei Beacons da parte di terzi, è possibile con funzioni avanzate criptare i messaggi scambiati tra devices e prevenire tali scenari. Per quanto riguarda la privacy del consumatore, per quanto possa essere utile per il negoziante tracciare ogni movimento dei clienti all'interno di un edificio, è necessario usufruire delle potenzialità dei Beacons in maniera cosciente. Nessun cliente vorrebbe un'app invasiva, con un continuo susseguirsi di notifiche o pubblicità indesiderate, ed in questo caso sta allo sviluppatore delle App usare una certa dose di buon senso per evitare usi impropri e sgradevoli per il consumatore, magari fornendo l'opzione di poter "sopprimere" un certo numero di notifiche ed essendo chiari a specificare quali informazioni vengono tracciate dalle applicazioni.

I cosiddetti “iBeacon” altro non sono che Beacon brandizzati Apple: essi hanno particolari specifiche richieste e, sorprendentemente non sono un'esclusiva della casa di Cupertino. Alcuni device Android sono progettati per supportare connessioni anche da iBeacons, così da incrementare la compatibilità globale delle applicazioni anche tra device di case differenti. Ogni applicazione mobile che supporta questa tecnologia può sfruttare i trasmettitori anche semplicemente per conoscere la distanza e la posizione precisa nel raggio d'azione indoor, trasmettere notifiche in maniera autonoma al device del consumatore facendolo interagire con l'ambiente.

Importante sottolineare alcuni dettagli pratici: ogni device con una connessione Bluetooth attiva e l'App adeguata reagirà automaticamente ai Beacon circostanti, senza la necessità che l'utente intervenga, per esempio dovendo estrarre lo smartphone dalla tasca. Il negoziante inoltre potrà monitorare le aree di maggior interesse dei propri clienti *ancora prima* che questi acquistino qualcosa, in quanto i Beacon possono tracciare ogni spostamento all'interno del negozio, notificando davanti a quale scaffale i clienti si fermano di più. Inutile sottolineare come questo possa essere un'informazione utilissima in termini di marketing. Tenere costantemente attivo il Bluetooth sul telefono non consumerà più la batteria in maniera eccessiva, proprio grazie alla particolare tecnologia a basso consumo energetico peculiare di BLE. Gli hardware degli iBeacon saranno semplici da installare e sempre più economici: essi potranno inoltre venir programmati non solo per essere degli Advertiser ma anche Scanner, cosìcché un semplice Smartphone, che è a tutti gli effetti un Beacon, possa inviare comandi ai dispositivi installati per esempio in casa, per accendere le luci od elettrodomestici a distanza. Un altro punto fondamentale è che non è mai richiesta una connessione ad Internet: anche se poter accedere a risorse su cloud tramite la rete potrebbe ampliare ancora di più il range di attività possibili con BLE, il network di iBeacons è una struttura autonoma.



Questo innovativo uso della tecnologia fa principalmente leva su due concetti chiave:

- **Micro-location**
- **Interaction/Engagement/Context**

Della Micro-location abbiamo già parlato in breve, ovvero è noto che uno dei punti di forza dei Beacon che sfruttano Bluetooth LE è la possibilità di una localizzazione a corto raggio precisa e rapida, ottima in situazioni indoor dove i devices hanno difficoltà a raccogliere i segnali dei satelliti GPS. I segnali radio hanno difficoltà attraversare mura di mattoni ed acciaio: per questo il sistema di GPS è inadatto per fornire una localizzazione precisa in un raggio di 15-20 metri, specialmente se ci troviamo in un edificio a più piani. Questo tipo di geolocalizzazione ad un alto livello di granularità è appunto definito convenzionalmente con il termine "Micro-location".

La differenza fondamentale tra la Micro-location usata dagli iBeacon e la localizzazione globale del sistema GPS è la scala in cui lavorano: i GPS forniscono una posizione assoluta in termini di latitudine e longitudine, mentre la Micro-location lavora semplicemente con la distanza relativa del device dai punti noti dove sono posizionati i Beacon. Gli iBeacons sono infatti notificati quando un device lascia l'area da loro coperta, diventando così capaci di monitorare le posizioni nel tempo.

Il secondo concetto chiave è alla base di ogni utilizzo di questa tecnologia da parte dell'utente finale. I segnali degli iBeacons possono interagire con le Apps, inviando notifiche e facendo eseguire a queste ultime azioni specifiche in un tempo ed una locazione specifica. Sostanzialmente, il servizio fornito dai Beacons è quello di fornire un contesto esterno all'applicazione, che può ora conoscere cosa realmente circonda l'utilizzatore del device. Per questo motivo viene aperta la possibilità di far interagire le applicazioni mobile con il mondo fisico circostante senza alcuno sforzo, in modo che l'utente finale possa davvero percepire il collegamento tra la realtà ed il device che tiene tra le mani.

Ricapitolando: gli iBeacon utilizzano Bluetooth Low Energy per creare un'infrastruttura smart, orientata alla localizzazione, che i dispositivi Mobile possono utilizzare per ricavare informazioni contestuali basate sull'ambiente stesso in cui si muovono, in tempo reale. Le applicazioni possono ora sapere esattamente dove si trovano e cosa le circonda, aprendo la strada ad un nuovo livello di interazione col mondo, senza bisogno di una connessione ad Internet.

API

Già con iOS 4 Apple ha introdotto nelle API il concetto di regione geografica, che gli sviluppatori potevano utilizzare usando la classe `CLRegion` per seguire i movimenti di un utente all'interno dei confini di una dato spazio. Pur essendo una feature utile, le regioni definite in una specifica locazione sono limitanti ed il processo per definire e tracciare numerose regioni in una vasta area è tedioso e complicato.

Con iOS 7 ed iBeacon, tutto questo cambia: le regioni non devono più essere collegate ad una specifica area geografica fissa ma possono essere relative ai Beacon disposti nella zona. La classe `CLRegion` è stata resa astratta ed ora è suddivisa in due sottoclassi concrete, `CLCircularRegion` e `CLBeaconRegion`, la prima per tracciare locazioni geografiche assolute e specifiche, la seconda per implementare l'utilizzo della locazione relativa al network dei Beacon.

Un'altra modifica importante a queste specifiche è stata l'aggiunta di nuove proprietà alla classe `CLRegion`: `notifyOnEntry` e `notifyOnExit`. Precedentemente infatti, non appena il device entrava in una regione definita dalle App, veniva lanciata una notifica anche con il dispositivo inattivo. Ora è possibile modificare tale comportamento e, per esempio, lanciare la notifica solo se il display del device è attivo con la proprietà `notifyEntryStateOnDisplay`, specifica di `CLBeaconRegion`, in questo modo:

```
// notifica l'utente solo se il display è acceso
region.notifyEntryStateOnDisplay = YES;
region.notifyOnEntry = NO;
region.notifyOnExit = NO;
```

Per istanziare un oggetto di tipo `CLBeaconRegion` è necessario fornire un UUID ed un identifier testuale per identificare univocamente l'applicazione. Per generare un UUID unico è sufficiente utilizzare il comando `uuidgen` da terminale, che ritorna una stringa a 16-bit.

```
// istanziare una nuova regione
NSUUID *myUUID = [[NSUUID alloc] initWithUUIDString:@"10D39AE7-020E-4467-
      9CB2-DD36366F899D"];
CLBeaconRegion *region = [[CLBeaconRegion alloc]
      initWithProximityUUID:myUUIDidentifier:myCompanyIdentifier];
```

`CLBeaconRegion` utilizza inoltre due proprietà opzionali `uint16_t`, `major` e `minor`, che possono essere settate durante l'inizializzazione. Un possibile caso d'uso commerciale per tali valori potrebbe essere utilizzare il primo per definire l'edificio in cui si trova, ed il secondo per indicare la corsia, così da rendere possibile la personalizzazione del contenuto di ogni Beacon in maniera granulare.

Per iniziare a monitorare la regione, è necessario istanziare una classe `CLLocationManager` ed usare la funzione `startMonitoringForRegion:`.

Le notifiche di entrata in una regione verranno lanciante dai metodi di `CLLocationManagerDelegate`; che sono i seguenti: `locationManager:didDetermineState:forRegion:`, `locationManager:didEnterRegion:`, ed `locationManager:didExitRegion:` permettendo così all'app di notificare l'utente quando entra in una nuova regione.

Oltre a provvedere alle notifiche di entrata ed uscita, `CLBeaconRegions` supporta la misurazione della distanza. Quest'ultima viene approssimata basandosi sull'intensità del segnale tra il Beacon ed il device, per poter sfruttare anche tutte le possibili funzionalità che derivando dal conoscere la posizione specifica dell'utente.

```
(void)locationManager:(CLLocationManager *)manager didEnterRegion:
      (CLRegion   *)region {

    // condizione di entrata
    if ([region.identifier isEqualToString:kUniqueRegionIdentifier]
        && !self.didShowEntranceNotifier) {

        // si decide se notificare o meno all'utente l'entrata nella regione
        self.didShowEntranceNotifier = YES;

        // tracking dei beacon
        [manager startRangingBeaconsInRegion:self.targetRegion];
    }
}
```

Controllando la zona nelle vicinanze, si può, a seconda dei casi d'uso, notificare o meno all'utente della presenza di uno o più Beacon. Questa cosa può venire ripetuta od essere visualizzata una tantum, a seconda dei casi.

```
(void)locationManager:(CLLocationManager *)manager
        didRangeBeacons:(NSArray *)beacons
               inRegion:(CLBeaconRegion *)region {

    // si identifica il beacon più vicino
    if ([beacons count] > 0) {
        CLBeacon *closestBeacon = beacons[0];
        if (closestBeacon.proximity == CLProximityImmediate) {
            /**
             Si informa l'utente della sua prossimità ad un Beacon
             Si può fare una tantum oppure ripetutamente, a seconda dei casi
                 d'uso
             Opzionalmente è possibile usare i valori di major e minor per
                 dare una contestualizzazione
             */
            [self fireUpdateNotificationForStatus:@"Sei nel raggio di un
                        Beacon"];

        } else if (closestBeacon.proximity == CLProximityNear) {
            // Si cercano altri beacon nelle vicinanze
            [self fireUpdateNotificationForStatus:@"Ci sono altri Beacon
                        nelle vicinanze"];
        }
    } else {
        // Non ci sono beacon nelle vicinanze
        [self fireUpdateNotificationForStatus:@"Non ci sono Beacon nelle
                        vicinanze"];
    }
}
```

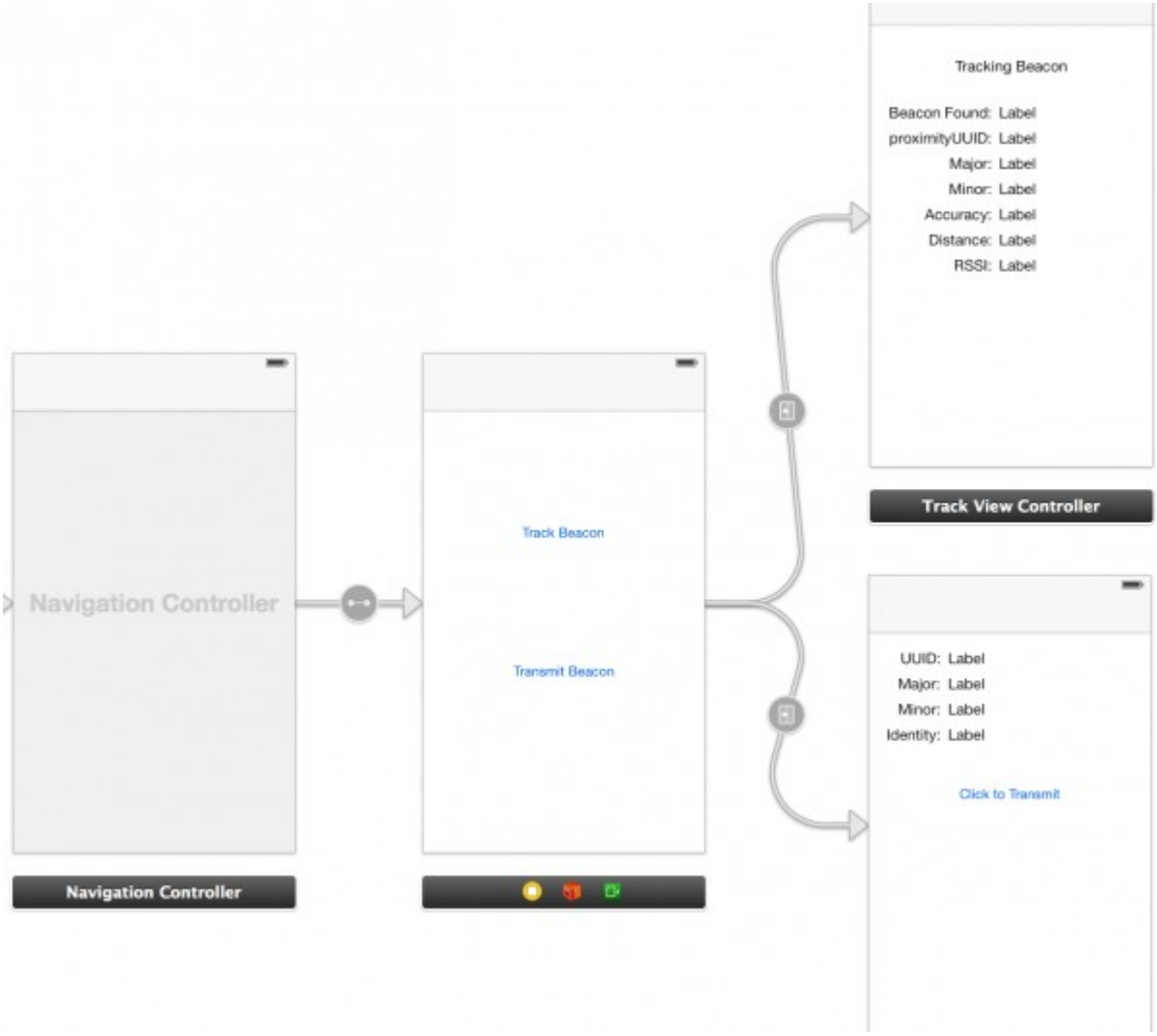
Come specificato in precedenza, ogni device che supporta Bluetooth 4.0 può potenzialmente agire come un Beacon. Per far sì che un dispositivo iOS possa fare advertising del suo segnale Bluetooth, è necessario istanziare una nuova `CLBeaconRegion` utilizzando l'UUID eventualmente creato in precedenza, ed a scelta con valori opzionali o meno. Inoltre è necessario configurare un'istanza di `CBPeripheralManager` per poter iniziare a fare broadcasting col device.

```
// advertising su una regione
CLBeaconRegion *advertisingRegion = [[CLBeaconRegion alloc]
                                    initWithProximityUUID:[CSMAppDelegate
                                                appDelegate].myUUID
                                    major:kMyStoreNumber
                                    minor:kWeeklySpecialItemNumber
                                                identifier:kUniqueRegionIden
                                                     tifier];

NSDictionary *peripheralData = [advertisingRegion
                  peripheralDataWithMeasuredPower:nil];

CBPeripheralManager *peripheralManager = [[CBPeripheralManager alloc]
                                           initWithDelegate:self
                                           queue:dispatch_get_main_queue()];

[peripheralManager startAdvertising:peripheralData];
```

Setup di un iBeacon e prototipo di App

In questo capitolo seguiremo passo-passo una possibile e semplice installazione e configurazione di un Beacon.

Nel nostro prototipo di applicazione avremo una prima view principale che permette di scegliere tra le due modalità, quella di Scanner (per tracciare i Beacon) e quella di Advertiser (cioé per rendere il nostro device un Beacon a tutti gli effetti).



Si creeranno due classi differenti per il Track View Controller ed il Config View Controller assegnandone una a ciascuno ed aggiungendo i dovuti IBOutlets:

```
//nel Track View Controller
@property (weak, nonatomic) IBOutlet UILabel *beaconFoundLabel;
@property (weak, nonatomic) IBOutlet UILabel *proximityUUIDLabel;
@property (weak, nonatomic) IBOutlet UILabel *majorLabel;
@property (weak, nonatomic) IBOutlet UILabel *minorLabel;
@property (weak, nonatomic) IBOutlet UILabel *accuracyLabel;
@property (weak, nonatomic) IBOutlet UILabel *distanceLabel;
@property (weak, nonatomic) IBOutlet UILabel *rssiLabel;
```

```
//nel Config View Controller
@property (weak, nonatomic) IBOutlet UILabel *uuidLabel;
@property (weak, nonatomic) IBOutlet UILabel *majorLabel;
@property (weak, nonatomic) IBOutlet UILabel *minorLabel;
@property (weak, nonatomic) IBOutlet UILabel *identityLabel;
```

Inizieremo con l'obiettivo di configurare un device trasmettitore: per fare questo, è necessario importare alcuni framework Core forniti nelle specifiche Bluetooth.


```
//Un prototipo di header del ConfigViewController
#import <CoreLocation/CoreLocation.h>
#import <CoreBluetooth/CoreBluetooth.h>

@property (strong, nonatomic) CLBeaconRegion *beaconRegion;
@property (strong, nonatomic) NSDictionary *beaconPeripheralData;
@property (strong, nonatomic) CBPeripheralManager *peripheralManager;
```

Inizializzeremo 3 proprietà: `beaconRegion`, che sarà usata per definire i settaggi necessari al beacon advertiser (UUID, major e minor), `beaconPeripheralData` che conterrà i dati dell'area circostante ed il `peripheralManager` dove sono contenuti i metodi per inizializzare le trasmissioni.

Nel nostro esempio, utilizzeremo un metodo `viewDidLoad` per contenere le configurazioni iniziali per inizializzare il Beacon e configurarlo per trasmettere. In questo esempio assegneremo "`1`" per semplicità `major` e `minor`, ed avremo un identifier fittizio.

```
(void)initBeacon {
    NSUUID *uuid = [[NSUUID alloc] initWithUUIDString:@"23542266-18D1-4FE4-
                                        B4A1-23F8195B9D39"];
    self.beaconRegion = [[CLBeaconRegion alloc] initWithProximityUUID:uuid
                                        major:1
                                        minor:1
                                        identifier:@"com.foo.foobarRegion"];
}
```

Nel metodo transmitBeacon configureremo poi tutti i parametri necessari a definire l'area periferica al nostro Beacon. Come abbiamo già visto in precedenza, chiameremo `peripheralDataWithMeasuredPower` della proprietà `self.beaconRegion` e la assegneremo alla proprietà locale `beaconPeripheralData`. Nella riga successiva, dopo aver allocato la proprietà `peripheralManager`, in questo esempio, assegneremo il valore `nil` a `queue` ed `options`, mentre useremo un delegate settato a `self` per implementare una particolare funzionalità che vedremo subito dopo.

```
(IBAction)transmitBeacon:(UIButton *)sender {
            self.beaconPeripheralData = [self.beaconRegion
                                        peripheralDataWithMeasuredPower:nil];
            self.peripheralManager = [[CBPeripheralManager alloc]
                                        initWithDelegate:self
                                        queue:nil
                                        options:nil];
}
```

Per far sì che il set di delegate a `self` funzioni, è inoltre necessario far sì che la classe sia conforme al protocollo `CBPeripheralManagerDelegate`, aggiungendo una riga nell'header del nostro file in questo modo:

```
@interface ConfigViewController : UIViewController
                        <CBPeripheralManagerDelegate>
```

Perché si è deciso di implementare questo accorgimento? La risposta è semplice: per accendere il dispositivo, è necessario chiamare il metodo `startAdvertising`. Se questo viene fatto nel metodo `transmitBeacon` di cui sopra, c'è la possibilità che possa fallire, in quanto i servizi Bluetooth richiesti potrebbero non essere ancora operativi subito dopo l'attivazione della connessione Bluetooth nel device. Per questo, il codice viene messo nel metodo `peripheralManagerDidUpdateState`: appena lo status del Bluetooth cambia (viene acceso o spento dall'utente) il delegate chiama questo metodo, se lo status è cambiato in On allora può partire con il metodo `startAdvertising` dell'oggetto `beaconPeripheralData` di tipo NSDictionary creato in precedenza. In caso il cambio di stato sia uno spegnimento, si può stoppare l'advertising o semplicemente non fare nulla.

```
(void)peripheralManagerDidUpdateState:(CBPeripheralManager *)peripheral {
    if (peripheral.state == CBPeripheralManagerStatePoweredOn) {
        NSLog(@"Power On");
        [self.peripheralManager startAdvertising:self.beaconPeripheralData];
    } else if (peripheral.state == CBPeripheralManagerStatePoweredOff) {
        NSLog(@"Power Off");
        [self.peripheralManager stopAdvertising];
    }
}
```

Infine, possiamo configurare esattamente il contenuto di cosa vogliamo trasmettere. Chiameremo il metodo `setLabels` dal nostro principale `viewDidLoad`.

```
(void)setLabels {
      self.uuidLabel.text = self.beaconRegion.proximityUUID.UUIDString;
      self.majorLabel.text = [NSString stringWithFormat:@"%@",
            self.beaconRegion.major];
      self.minorLabel.text = [NSString stringWithFormat:@"%@",
            self.beaconRegion.minor];
      self.identityLabel.text = self.beaconRegion.identifier;
}
```

In questo metodo stiamo semplicemente settando le proprietà di ogni label, per trasmettere i dettagli del nostro Beacon. Da notare che il `proximityUUID` ha una proprietà chiamata `UUIDString`, che possiamo assegnare così com'è a uuidLabel.text in quanto si tratta di una stringa. I parametri `major` e `minor` avranno al contrario bisogno di una conversione a stringa.

Con questa App siamo ora in grado di utilizzare un device Bluetooth 4.0 come un Beacon.

```
//ConfigViewController Header

#import <UIKit/UIKit.h>
#import <CoreLocation/CoreLocation.h>
#import <CoreBluetooth/CoreBluetooth.h>

@interface ConfigViewController : UIViewController
<CBPeripheralManagerDelegate>

@property (strong, nonatomic) CLBeaconRegion *beaconRegion;
@property (weak, nonatomic) IBOutlet UILabel *uuidLabel;
@property (weak, nonatomic) IBOutlet UILabel *majorLabel;
@property (weak, nonatomic) IBOutlet UILabel *minorLabel;
@property (weak, nonatomic) IBOutlet UILabel *identityLabel;
@property (strong, nonatomic) NSDictionary *beaconPeripheralData;
@property (strong, nonatomic) CBPeripheralManager *peripheralManager;

@end
```

```
//ConfigViewController Implementation

#import "ConfigViewController.h"
@interface ConfigViewController ()
@end
@implementation ConfigViewController
- (void)viewDidLoad
{
    [super viewDidLoad];
        // Eventuali setup addizionali richiesti
    [self initBeacon];
    [self setLabels];
}

(void)initBeacon {
    NSUUID *uuid = [[NSUUID alloc] initWithUUIDString:@"23542266-18D1-4FE4-
                                          B4A1-23F8195B9D39"];
    self.beaconRegion = [[CLBeaconRegion alloc] initWithProximityUUID:uuid
                                        major:1
                                        minor:1
                                        identifier:@"com.foo.foobarRegion"];
}


(void)setLabels {
      self.uuidLabel.text = self.beaconRegion.proximityUUID.UUIDString;
      self.majorLabel.text = [NSString stringWithFormat:@"%@",
            self.beaconRegion.major];
      self.minorLabel.text = [NSString stringWithFormat:@"%@",
            self.beaconRegion.minor];
      self.identityLabel.text = self.beaconRegion.identifier;
}


(IBAction)transmitBeacon:(UIButton *)sender {
            self.beaconPeripheralData = [self.beaconRegion
                                    peripheralDataWithMeasuredPower:nil];
            self.peripheralManager = [[CBPeripheralManager alloc]
                                    initWithDelegate:self
                                    queue:nil
                                    options:nil];
}


(void)peripheralManagerDidUpdateState:(CBPeripheralManager *)peripheral {
    if (peripheral.state == CBPeripheralManagerStatePoweredOn) {
        NSLog(@"Power On");
        [self.peripheralManager startAdvertising:self.beaconPeripheralData];
    } else if (peripheral.state == CBPeripheralManagerStatePoweredOff) {
        NSLog(@"Power Off");
        [self.peripheralManager stopAdvertising];
    }
}

(void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Handling degli errori di memoria
}

@end
```

Per poter sviluppare un'applicazione in grado di tracciare e fare discovery dei Beacon è necessario il CoreLocation Framework: da iOS 7 in avanti, un device può riconoscere quando entra nella regione di un Beacon ed ha la possibilità di misurarne la distanza e scoprire informazioni su di esso.

Per iniziare, va importato nell'header del file TrackViewController il CoreLocation Framework e vanno settate alcune proprietà come nell'esempio:

```
#import <CoreLocation/CoreLocation.h>

@property (strong, nonatomic) CLBeaconRegion *beaconRegion;
@property (strong, nonatomic) CLLocationManager *locationManager;
```

Inizializziamo una `CLBeaconRegion` per definire il tipo di Beacon che stiamo cercando: per esempio, in un museo vorremo solamente cercare tutti i Beacon aventi come UUID quello del museo in questione. Aggiungiamo inoltre un `locationManager` che è utilizzato per cercare e scoprire i servizi dei Beacon circostanti.

Ancora una volta ci servirà un delegate per il `locationManager`, quindi dovrà essere aggiunta una riga all'header per includere il protocollo `CLLocationManagerDelegate`. Una volta incluso, si potrà settare la proprietà `locationManager.delegate` a self nel metodo `viewDidLoad`.

```
// Header
@interface TrackViewController : UIViewController
                        <CLLocationManagerDelegate>


// **** //


// metodo viewDidLoad

self.locationManager = [[CLLocationManager alloc] init];
self.locationManager.delegate = self;
[self initRegion];
```

Utilizzeremo ancora un metodo chiamato `viewDidLoad`, stavolta che richiama un altro metodo denominato `initRegion`, così costruito:

```
(void)initRegion {
    NSUUID *uuid = [[NSUUID alloc] initWithUUIDString:@"23542266-18D1-4FE4-
                   B4A1-23F8195B9D39"];
    self.beaconRegion = [[CLBeaconRegion alloc] initWithProximityUUID:uuid
                   identifier:@"com.foo.foobarRegion"];
    [self.locationManager startMonitoringForRegion:self.beaconRegion];
}
```

Vengono creati ed inizializzati un oggetto NSUUID, con una stringa, e la proprietà `beaconRegion`. Per questo esempio e per far sì che questo device possa rilevare il Beacon che abbiamo configurato poco fa, utilizzeremo lo stesso UUID e lo stesso identifier della regione. Anche se non è buona pratica, per semplicità tali valori vengono "hard-coded" nel codice. Nell'ultima riga, infine andremo ad iniziare lo scanning di questa particolare regione.

Una volta settati questi parametri, è necessario far sì che l'applicazione possa determinare quando il device su cui sta venendo eseguita entra od esce da una particolare area di copertura di un Beacon. Per fare ciò verranno utilizzati due metodi del delegate: `didEnterRegion` e `didExitRegion`. Il primo viene chiamato non appena si entra nell'area del device trasmittente, e successivamente viene chiamato il metodo `startRangingBeaconsInRegion` del `locationManager`, specificando la regione creata precedentemente. Al contrario, uscendo dalla regione, viene chiamato `stopRangingBeaconsInRegion` con lo stesso procedimento.

```
(void)locationManager:(CLLocationManager *)manager didEnterRegion:(CLRegion
                      *)region {
    [self.locationManager startRangingBeaconsInRegion:self.beaconRegion];
}

(void)locationManager:(CLLocationManager *)manager didExitRegion:(CLRegion
                                    *)region {
    [self.locationManager stopRangingBeaconsInRegion:self.beaconRegion];
    self.beaconFoundLabel.text = @"No";
}
```

Il metodo principale, che permetterà all'applicazione di eseguire istruzioni in base alla presenza ed alla prossimità dei Beacon nelle circostanze, ed in base alle informazioni fornite da questi ultimi è il metodo `didRangeBeacons`.

```
(void)locationManager:(CLLocationManager *)manager didRangeBeacons:(NSArray
                      *)beacons inRegion:(CLBeaconRegion *)region {
    CLBeacon *beacon = [[CLBeacon alloc] init];
    beacon = [beacons lastObject];

    self.beaconFoundLabel.text = @"Yes";
    self.proximityUUIDLabel.text = beacon.proximityUUID.UUIDString;
    self.majorLabel.text = [NSString stringWithFormat:@"%@", beacon.major];
    self.minorLabel.text = [NSString stringWithFormat:@"%@", beacon.minor];
    self.accuracyLabel.text = [NSString stringWithFormat:@"%f",
                      beacon.accuracy];
    if (beacon.proximity == CLProximityUnknown) {
        self.distanceLabel.text = @"Distanza sconosciuta";
    } else if (beacon.proximity == CLProximityImmediate) {
        self.distanceLabel.text = @"Entro 50cm";
    } else if (beacon.proximity == CLProximityNear) {
        self.distanceLabel.text = @"Oltre 50cm, entro 5m";
    } else if (beacon.proximity == CLProximityFar) {
        self.distanceLabel.text = @"Oltre 5m";
    }
    self.rssiLabel.text = [NSString stringWithFormat:@"%i", beacon.rssi];
}
```

Una volta creato un oggetto `beacon` di tipo `CLBeacon`, lo inizializziamo dichiarandolo uguale all'ultimo oggetto dall'argomento del metodo (`beacons`). In questo modo avremo uno specifico Beacon su cui lavorare.

Nel corpo del codice vengono semplicemente richieste le informazioni al Beacon selezionato: l'UUID, i valori major e minor, l'accuratezza della localizzazione e l'RSSI. Da notare che questi dati potrebbero variare a seconda dell'implementazione: il Broadcaster potrebbe decidere di non inviare alcuni di questi parametri e nel nostro caso, se vengono modificati i valori nel Transmitter che abbiamo definito in precedenza, anche qui avranno effetto gli stessi cambiamenti.

L'accuratezza, la prossimità e l'RSSI sono le proprietà che determinano la distanza dal Beacon. L'accuratezza varia a seconda delle interferenze nelle circostanze. La prossimità varia in 4 valori predefiniti: Unknown, Immediate, Near e Far, in scala dal più vicino al più lontano. RSSI è semplicemente la forza del segnale in decibel.

Il codice dell'App sul device scanner sarà dunque il seguente:

```
//TrackViewController Header

#import <UIKit/UIKit.h>
#import <CoreLocation/CoreLocation.h>

@interface TrackViewController : UIViewController
<CLLocationManagerDelegate>

@property (weak, nonatomic) IBOutlet UILabel *beaconFoundLabel;
@property (weak, nonatomic) IBOutlet UILabel *proximityUUIDLabel;
@property (weak, nonatomic) IBOutlet UILabel *majorLabel;
@property (weak, nonatomic) IBOutlet UILabel *minorLabel;
@property (weak, nonatomic) IBOutlet UILabel *accuracyLabel;
@property (weak, nonatomic) IBOutlet UILabel *distanceLabel;
@property (weak, nonatomic) IBOutlet UILabel *rssiLabel;

@property (strong, nonatomic) CLBeaconRegion *beaconRegion;
@property (strong, nonatomic) CLLocationManager *locationManager;

@end
```

```
//TrackViewController Implementation

#import "TrackViewController.h"

@interface TrackViewController ()
@end
@implementation TrackViewController

(void)viewDidLoad
{
    [super viewDidLoad];
        //Eventuali setup addizionali richiesti
    self.locationManager = [[CLLocationManager alloc] init];
    self.locationManager.delegate = self;
    [self initRegion];
}

// ***** Continua ****** //
```

```
// ***** Continua ****** //

(void)initRegion {
    NSUUID *uuid = [[NSUUID alloc] initWithUUIDString:@"23542266-18D1-4FE4-
                   B4A1-23F8195B9D39"];
    self.beaconRegion = [[CLBeaconRegion alloc] initWithProximityUUID:uuid
                    identifier:@"com.foo.foobarRegion"];
    [self.locationManager startMonitoringForRegion:self.beaconRegion];
}

(void)locationManager:(CLLocationManager *)manager didEnterRegion:(CLRegion
                                                          *)region {
    NSLog(@"Beacon Trovato");
    [self.locationManager startRangingBeaconsInRegion:self.beaconRegion];
}

(void)locationManager:(CLLocationManager *)manager didExitRegion:(CLRegion
                                                          *)region {
    NSLog(@"Uscita dal range del Beacon");
    [self.locationManager stopRangingBeaconsInRegion:self.beaconRegion];
    self.beaconFoundLabel.text = @"No";
}

(void)locationManager:(CLLocationManager *)manager didRangeBeacons:(NSArray
                        *)beacons inRegion:(CLBeaconRegion *)region {
    CLBeacon *beacon = [[CLBeacon alloc] init];
    beacon = [beacons lastObject];

    self.beaconFoundLabel.text = @"Yes";
    self.proximityUUIDLabel.text = beacon.proximityUUID.UUIDString;
    self.majorLabel.text = [NSString stringWithFormat:@"%@", beacon.major];
    self.minorLabel.text = [NSString stringWithFormat:@"%@", beacon.minor];
    self.accuracyLabel.text = [NSString stringWithFormat:@"%f",
                        beacon.accuracy];
    if (beacon.proximity == CLProximityUnknown) {
        self.distanceLabel.text = @"Distanza sconosciuta";
    } else if (beacon.proximity == CLProximityImmediate) {
        self.distanceLabel.text = @"Entro 50cm";
    } else if (beacon.proximity == CLProximityNear) {
        self.distanceLabel.text = @"Oltre 50cm, entro 5m";
    } else if (beacon.proximity == CLProximityFar) {
        self.distanceLabel.text = @"Oltre 5m";
    }
    self.rssiLabel.text = [NSString stringWithFormat:@"%i", beacon.rssi];
}

(void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Handling degli errori di memoria
}

@end
```

Case Study

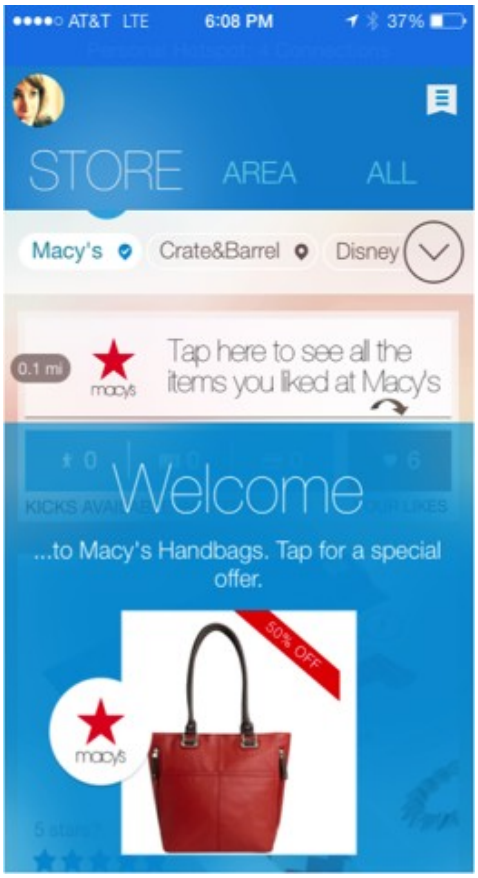
Macy's

Come abbiamo già visto in precedenza, le features di iBeacon sembrano perfette per essere adottate come shop-assistant nei grandi magazzini e negozi. Già dal Novembre 2013, poco tempo dopo l'annuncio del lancio di iBeacon da parte di Apple, una compagnia sviluppatrice, Shopkick, ed il grande retailer americano Macy's hanno trovato un accordo per un iniziale testing di questa tecnologia nel campo commerciale. In due dei principali store di Macy's, a New York nella Herald Square ed a San Francisco nella Union Square, sono infatti già stati installati gli ShopBeacon, un particolare tipo di iBeacon derivato dalle standardizzazioni Apple e personalizzato dalla compagnia sviluppatrice.

L'applicazione Shopkick, omonima della compagnia, è già stata distribuita ai clienti di Macy's che hanno deciso di partecipare a questo beta testing, usufruendo di sconti particolari e suggerimenti calcolati in base al loro storico degli acquisti ed aree visitate. L'applicazione infatti, dopo aver ricevuto il consenso dell'utilizzatore, raccoglie informazioni e le elabora, fornendo coupon di sconto e consigli per gli acquisti personalizzati non appena l'utente munito di device si avvicina al negozio. Gli ShopBeacons possono inoltre guidare fisicamente gli utenti all'interno dei grandi negozi, in modo da permettere al consumatore di giungere rapidamente alle aree di interesse.

Un altro dettaglio interessante è l'integrazione di Shopkeep con la rete Internet: l'App è sviluppata in modo da poter collezionare informazioni sui gusti dell'utente anche dal browsing effettuato da casa, cosìcché il consumatore può sfogliare i cataloghi del retailer comodamente da casa, scegliere i prodotti di suo interesse e l'App farà il lavoro di guidarlo, avvisarlo e consigliarlo non appena si troverà fisicamente nella zona del negozio.

Il retailer inoltre può, a seconda del buon senso e dei consensi degli utenti, raccogliere un'enormità di dati statistici su quali prodotti sono più interessanti per il pubblico ed agire di conseguenza, con pubblicità mirata per ogni singolo consumatore.



Shopkeep può inoltre sostituirsi alle Fidelty Cards, le raccolte punti per acquisti multipli nello stesso store. L'applicazione tiene traccia del numero di visite dell'utente, dei tempi in cui esso rimane all'interno del negozio e lo ricompensa con offerte vantaggiose una volta raggiunto un certo numero di “punti fedeltà”.

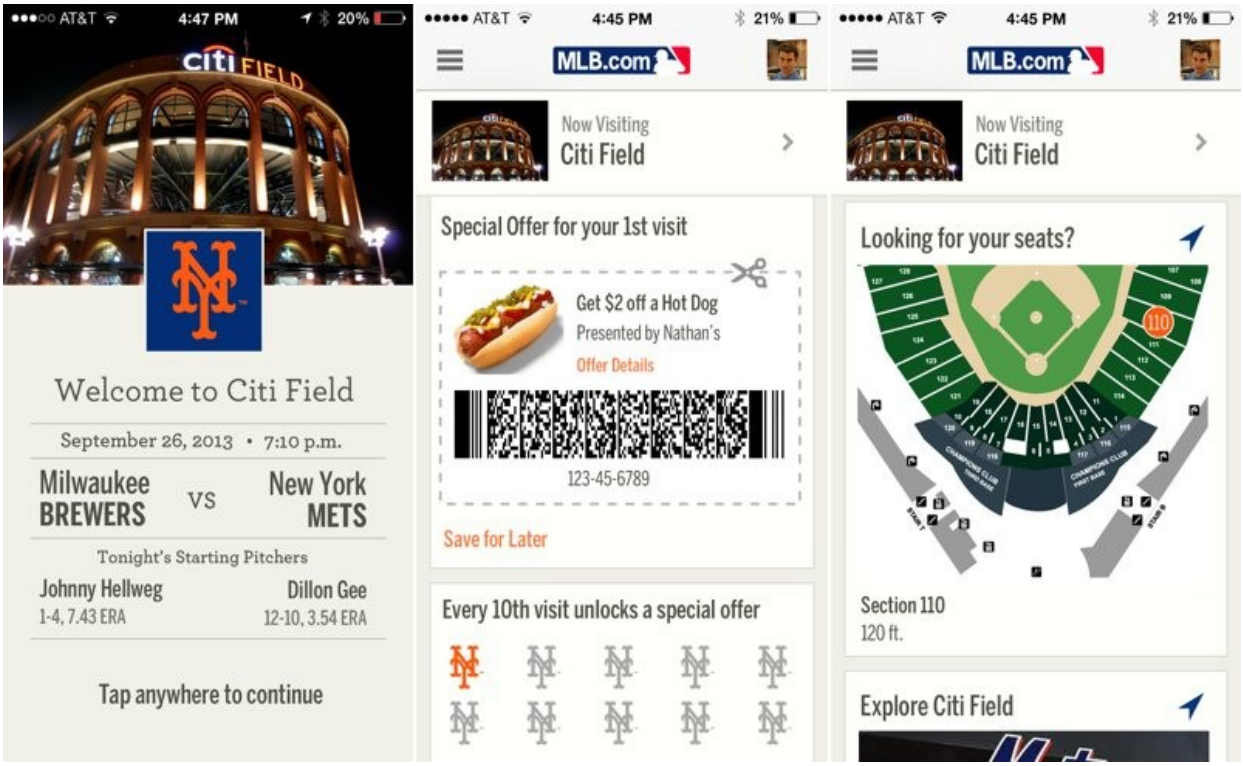
Le tentazioni offerte dagli sconti speciali per gli utilizzatori di Shopkeep sono inoltre aiutate da una tecnologia di pagamento integrata, che con pochi comandi richiesti dall'utente permette di effettuare pagamenti semplificati direttamente dal proprio device. Inutile dire che la possibilità di acquistare e pagare così rapidamente, evitando eventuali file alla cassa e gestendo tutto tramite l'applicazione è un altro enorme incentivo per i consumatori ad acquistare e di conseguenza per i negozianti ad adottare questa tecnologia.

Major League Baseball

Anche la prestigiosa lega di Baseball americana, la MLB, già da Settembre 2013 sta adottando e testando una applicazione basata sulla tecnologia Bluetooth Low Energy per rendere l'esperienza dei fan ancora più interattiva. Citando le parole di Marc Abramson, sviluppatore iOS per la MLB:

"Stiamo cercando di personalizzare le app basate sulla locazione all'interno degli stadi, ma il GPS è non è adatto agli ambienti indoor, specialmente all'interno di enormi strutture d'acciaio. Stiamo infatti implementando le nuove tecnologie Apple di Bluetooth LE e iBeacon per iOS 7 e siamo davvero entusiasti delle loro potenzialità."

La collaborazione della MLB con Apple ha portato allo sviluppo di "At the Ballpark", letteralmente "allo stadio", l'applicazione per devices abilitati a Bluetooth 4.0 che verrà testata dai New York Mets e, secondo le previsioni, dovrà venire aperta al pubblico del Citi Field nel 2014. Il Citi Field è già attrezzato con numerosi iBeacon e l'applicazione è pronta per supportare altri stadi e configurazioni specifiche per ogni team della lega. Tra le features di tale app troviamo la capacità di guidare lo spettatore direttamente dalle fermate della metropolitana adiacenti allo stadio ai cancelli di quest'ultimo, e successivamente ai posti a sedere assegnati.



I fan poi oltre a ricevere coupon per l'acquisto di bevande direttamente all'interno dello stadio potranno usufruire di dettagliate informazioni sul match che si sta svolgendo davanti ai loro occhi.

Un altro dettaglio che sicuramente farà gola al pubblico è quello dell'eliminazione delle code all'ingresso: il codice a barre del proprio biglietto d'ingresso può venire visualizzato dall'app e con un pagamento immediato fornire un accesso privilegiato e rapido all'interno della struttura.

Anche in questo caso può essere integrato un sistema di abbonamenti che offre tutti i vantaggi già discussi: sconti fedeltà, raccolta punti per omaggi come bevande o spuntini gratis, od addirittura merchandise ufficiale della squadra del cuore: magliette, cappellini e tanto altro, tutto questo solo usufruendo gratuitamente dell'applicazione.

Ed ancora, l'applicazione può tenere traccia delle più svariate informazioni utili all'utente: notificare la coda più veloce all'uscita, offrire la possibilità di acquistare una bibita che può venir recapitata direttamente al posto a sedere dell'utente da parte dei camerieri, raccogliere informazioni sulla soddisfazione dei fan in base alla sistemazione e tanto altro.

Il concetto della privacy è ben messo in chiaro dai rappresentati della MLB, che tengono a sottolineare che tutte le informazioni personali raccolte non verranno utilizzate per secondi fini, se non dopo una esplicita richiesta all'utente.

Considerazioni finali

Sviluppi futuri

Considerando la rapida espansione della tecnologia Bluetooth 4.0 e l'ambiente di device abilitati già pronto a riceverla, è facile prevedere una commercializzazione a macchia d'olio in ogni scenario, dai grandi magazzini ed industrie multinazionali alle più piccole attività commerciali.

La facilità d'uso e la semplicità d'implementazione di ogni genere di applicazione rendono il Bluetooth Low Energy alla portata di tutti: il prezzo già relativamente basso di un iBeacon è infatti destinato a diminuire drasticamente una volta che nel mercato si verrà a creare una certa richiesta, che viste le premesse sembra inevitabile.

L'obiettivo di questa tesi è fornire una panoramica completa di questa innovativa tecnologia, destinata anche alle piccole aziende interessate nella zona italiana: essere tra i primi a sviluppare applicazioni per sistemi Bluetooth Low Energy in Italia è sicuramente una prospettiva non da sottovalutare. Le informazioni qui raccolte sono essenzialmente un insieme di nozioni utili a chiunque voglia approcciarsi allo sviluppo di applicazioni, unendo dettagli tecnici, procedurali ed economici per sfruttare al meglio e fin da subito la diffusione nel territorio dei Beacon.

In conclusione, l'auspicio è quello di portare un manuale guida utilizzabile da chiunque, sviluppatori, imprenditori e consumatori, per diffondere anche nel nostro Paese la nuova sorprendente tecnologia, sperando che questa possa segnare un significativo miglioramento della vita di tutti i giorni.

Webgrafia

- https://www.bluetooth.org/
- http://www.nearfieldcommunication.org/history-nfc.html
- http://www.rfidjournal.com/articles/view?1338/2
- http://www.wikipedia.org
- http://www.technologyuk.net/telecommunications/communication_technologies/bluetooth.shtml
- http://www.di.unisa.it/~ads/corso-security/www/CORSO-0203/Bluetooth/testo.htm
- Bluetooth Core Specifications Version 4.0
- http://www.stonestreetone.com/bluetopiaLE.cfm
- http://www.captechconsulting.com/blog/christopher-mann/ios-7-tutorial-series-core-location-beacons
- http://www.devfright.com/ibeacons-tutorial-ios-7-clbeaconregion-clbeacon/
- http://www.engadget.com/2013/11/20/macys-tests-location-specific-store-discounts-using-ibeacon/
- http://www.macrumors.com/2013/11/20/shopkick-and-macys-team-up-for-first-retail-based-ibeacons/
- http://www.fastcompany.com/3021989/tech-forecast/apples-ibeacon-comes-to-macys-is-this-the-future-of-shopping
- http://www.digitaltrends.com/sports/nfl-super-bowl-ibeacon-mlb-opening-day/
- http://mashable.com/2013/09/26/mlb-at-the-ballpark-app/
- http://www.imore.com/apples-ibeacon-hits-home-run-major-league-baseball-while-nfc-strikes-out
- http://www.fanengagement.nl/news/social-media/apple-ruling-location-awareness-with-new-ibeacon/