Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Corso di Dottorato in Ingegneria dell'Informazione

Studio ed implementazione di WSN per applicazioni IoT basate su Bluetooth mesh

Tesi di Dottorato di:
Andrea Gentili

Tutor:
Prof.ssa Paola Pierleoni

Coordinatore del corso:
Prof. Franco Chiaraluce

XXXIII ciclo - anno accademico 2019/2020

Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Corso di Dottorato in Ingegneria dell'Informazione

Studio ed implementazione di WSN per applicazioni IoT basate su Bluetooth mesh

Tesi di Dottorato di:
Andrea Gentili

Tutor:
Prof.ssa Paola Pierleoni

Coordinatore del corso:
Prof. Franco Chiaraluce

XXXIII ciclo - anno accademico 2019/2020

Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Facoltà di Ingegneria
Via Brecce Bianche – 60131 Ancona (AN), Italy

"Lo scopo della scienza non è quello di aprire la porta a una saggezza infinita, ma di fissare un limite all'errore infinito."
Bertolt Brecht

Sommario

Il presente lavoro di tesi si inserisce nell'ambito dello sviluppo di innovative tecnologie di comunicazione wireless rivolte alle più evolute applicazioni per reti di sensori wireless e dell'Internet of Things. Nello specifico, ci si è focalizzati sullo studio del recente *standard* Bluetooth mesh in riferimento a due suoi ambiti applicativi. Riguardo al primo ambito, l'attività intrapresa ha condotto allo studio, test e ottimizzazione del protocollo Bluetooth mesh su dispositivi reali in uno scenario di applicazioni di interesse nel settore *smart lighting*. In particolare è stata condotta un'analisi dettagliata sui limiti della sua applicabilità in riferimento all'affidabilità della ricezione dei messaggi di conferma in una rete Bluetooth mesh, essendo questa una delle maggiori sfide al momento nel campo del *lighting*. Pertanto, è stata proposta una configurazione ottimizzata della rete Bluetooth mesh che raggiunge una significativa riduzione della perdita di pacchetti. Per valutare la bontà della tecnica proposta è stato effettuato un confronto delle prestazioni con la configurazione standard utilizzando setup sperimentali con dispositivi reali. I risultati ottenuti nei test hanno mostrato un significativo miglioramento dell'affidabilità della ricezione di tali messaggi negli scenari ottimizzati proposti con buon compromesso tra la riduzione della perdita di pacchetti e il carico computazionale. E' stata inoltre realizzata la coesistenza dei servizi *beacon* basati sulla tecnologia Bluetooth Low Energy. Nello specifico, è stata implementata l'integrazione tra un applicativo con funzionalità *beacon* basato sul protocollo Bluetooth Low Energy con un applicativo fondato sul protocollo Bluetooth mesh per il *lighting*, precedentemente sviluppato per la creazione della rete. Infine, è stata implementata una procedura efficiente per l'aggiornamento *over the air* del *firmware* nei dispositivi.

Il secondo ambito affrontato può trarre vantaggio dalla trasmissione Bluetooth dell'informazione, rientrante nel campo delle *Wireless Body Sensor Networks* e si inquadra nell'ambito del monitoraggio di segnali biomedici in dispositivi indossabili h24. In particolare è stato realizzato un algoritmo per ridurre gli artefatti da movimento su segnali fotopletismografici durante intensa attività fisica sfruttando tecniche ed elaborazioni di *signal processing* nel dominio tempo-frequenza. L'algoritmo proposto, potenzialmente implementabile in dispositivi indossabili, permette un'accurata stima dell'*heart rate* e la ricostruzione del segnale fotopletismografico in presenza di significativi artefatti da movimento su soggetti valutati durante la camminata e la corsa.

Indice

Elenco delle figure

Elenco delle tabelle

Listings

Acronimi

ACK acknowledgment. 48

BLE Bluetooth Low Energy. 6

CCCD Client Characteristic Configuration Descriptor. 28

CRC Cyclic Redundancy Check. 24

DFU Device Firmware Update. 15

FHSS Frequency Hopping Spread Spectrum. 21

FN friend node. 42

HR heart rate. 7

IEEE Institute of Electrical and Electronics Engineers. 19

IoE Internet of Everythings. 9

IoT Internet of Things. 5

ISM Industrial, Scientific and Medical. 21

LPN low power node. 42

LSB least significant bit. 56

MAC Mediaum Access Control. 19

MIT Massachussets Institute of Technology. 9

MSB most significant bit. 56

OTA over-the-air. 69

PLR Packet Loss Rate. 46

Acronyms

POI point-of-interest. 15

PPG fotopletismografico. 5

PSR Packet Success Rate. 46

SIG Bluetooth Special Interest Group. 14

SO Smart objects. 10

TTL Time To Live. 43

UUID universally unique identifier. 25

VFCDM variable frequency complex demodulation. 109

WBSN Wireless Body Sensor Networks. 5

WSN Wireless Sensor Networks. 7

Capitolo 1

Introduzione

Con il presente lavoro di tesi è stata condotta un'importante attività di studio, test e ottimizzazione del protocollo Bluetooth per sistemi di comunicazione *wireless* per *Internet of Things* (IoT). In particolare, l'attenzione è stata rivolta al recente e innovativo *standard* Bluetooth mesh che consente di estendere il range di copertura della rete senza *failure points* e di ottenere alta scalabilità. Nel seguito verranno riportati i due ambiti trattati nel corso di questi tre anni di dottorato, in cui ci si è dedicati principalmente a due applicazioni: la prima, focalizzata su sistemi per il *lighting* e gestione intelligente di sistemi illuminotecnici basati su topologia mesh; la seconda, in ambito *Wireless Body Sensor Networks* (WBSN), riguardante l'implementazione di un efficiente algoritmo per l'elaborazione di segnali fotopletismografico PPG affetti da forti artefatti da movimento. Il lavoro si inquadra nell'ambito del monitoraggio di segnali biomedici in dispositivi indossabili h24 le cui informazioni potranno essere veicolate tramite la trasmissione Bluetooth.

La prima attività intrapresa nell'ambito dei sistemi di comunicazione *wireless* per IoT per applicazioni *smart lighting* si è focalizzata sullo studio, test ed ottimizzazione del protocollo Bluetooth mesh in uno scenario di applicazioni di interesse nel settore *smart lighting* che ha condotto alla realizzazione di una rete Bluetooth mesh. Comprendere le prestazioni di tale rete consente di metterne in luce alcuni sui limiti di applicabilità. Molte sono infatti le simulazioni effettuate, ma pochi i test reali, compiuti su set up di moltitudini di dispositivi Bluetooth disseminati nell'ambiente, che siano in grado di farci comprendere meglio il suo funzionamentoe darci maggiori informazioni circa le prestazioni, limiti, eventuali criticità. L'analisi delle criticità è alla base di proposte e implementazioni in grado di ottimizzare le prestazioni e identificare i limiti di applicabilità. Sulla base delle precedenti considerazioni, si è scelto di affrontare una delle maggiori sfide al momento nel campo del *lighting*: la corretta ricezione dei messaggi di conferma, denominati Status. I messaggi di Status rappresentano un aspetto critico che può generare problemi rilevanti nel caso in cui più dispositivi rispondano nello stesso tempo, comportando l'insorgere di collisioni e riducendo la probabilità di una loro corretta ricezione. Test

condotti su dispositivi reali hanno consentito di migliorare l'affidabilità della ricezione di tali messaggi nella rete Bluetooth mesh. In particolare sono stati condotti due test in differenti contesti: il primo in uno scenario controllato con elementi interferenti limitati, mentre il secondo in uno scenario reale che pone maggiori sfide. Verrà quindi descritta nel seguito una tecnica di *spreading* degli Status nel tempo partendo da una ottimizzazione del firmware *standard* della Nordic conforme alle specifiche *core* dello *standard* Bluetooth mesh. Al fine di valutare le performance della tecnica proposta è stata eseguita una comparazione proprio con la configurazione *standard* della Nordic, usando lo stesso setup sperimentale e gli stessi parametri di configurazione della rete per entrambi i test, come verrà descritto nel capitolo 4. La soluzione proposta per il Bluetooth mesh consiste quindi nell'implementazione di un *firmware* per il controllo di gruppi di luci in dispositivi a basso consumo energetico, basati su *chipset* Bluetooth. I test sul firmware implementato hanno portato ad ottenere un miglioramento complessivo rispetto al comportamento generale della rete, sia in termini di quantità totale di messaggi di conferma ricevuti dal cliente sia di un'uniformità generale nel numero di questi messaggi ricevuti da tutti i nodi, rispetto all'implementazione standard. Quest'ultima, ad oggi, non sembra affrontare una particolare soluzione volta a risolvere o , almeno limitare, il problema della congestione dei messaggi. Infine, è stata condotta un'analisi approfondita basata sui risultati ottenuti per migliorare ulteriormente la conoscenza della percentuale di comandi realmente eseguiti.

Nella sezione 5 dello stesso capitolo verrà descritta una procedura automatizzata per l'aggiornamento del *firmware over-the-air* per il *lighting* realizzato. Tale procedura rende possibile l'aggiornamento dei nodi, contemporaneamente e in *background*, sfruttando le caratteritiche del protocollo Bluetooth mesh, mentre la rete continua a svolgere il suo lavoro correttamente. Inoltre è stata realizzata l'integrazione tra un applicativo con funzionalità *beacon* basato sul protocollo Bluetooth Low Energy (BLE) e l'applicativo sviluppato per il *lighting* basato sul protocollo Bluetooth mesh. Nel presente lavoro sono stati presi in considerazione due formati *standard* di *beacon*: l'iBeacon di Apple e l'Eddystone di Google. Nel capitolo 6 verrà descritta quindi una possibile implementazione sulla coesistenza dei due protocolli in un unico *firmware*, in cui ogni dispositivo mesh può trasmettere un proprio messaggio *beacon*, secondo il proprio formato e, nello stesso tempo, partecipare attivamente nella rete creata. Tale implementazione ha avuto l'obiettivo di definire un punto da cui partire per aggiungere nuove funzionalità ai dispositivi della rete Bluetooth mesh per l'erogazione di nuovi servizi per IoT.

Nell'ambito delle *Wireless Body Sensor Networks* per applicazioni basate su sistemi di monitoraggio h24 di parametri biomedici è stato sviluppato un algoritmo per la stima dell'*heart rate* e la ricostruzione della curva fotopleti-

smografica basato su tecniche di *signal processing* nel dominio tempo-frequenza e *data fusion* di segnali sia fotopletismografici che accelerometrici. I rispettivi segnali grezzi sono stati acquisiti da un sensore indossabile da polso in presenza di severe condizioni di artefatti da movimento. La rimozione efficace degli artefatti da movimento su segnali acquisiti da dispositivi indossabili è ad oggi una delle sfide più difficili da affrontare in questo settore. I risultati ottenuti, riportati nel capitolo 6, mostrano che sia possibile ricostruire la forma d'onda del segnale PPG nel tempo e un'accurata stima dell'*heart rate* (HR). La trasmissione di questi segnali biomedici, elaborati con algoritmi innovativi potrà essere veicolata da nuove soluzioni di *Wireless Sensor Networks* (WSN) per IoT, quali ad esempio l'impiego di Beacon Bluetooth o l'utilizzo di topologie mesh per ambienti, come ad esempio quelli ospedalieri o delle residenze assistite, in cui una pluralità di utilizzatori potrebbere essere monitorati h24.

Capitolo 2

Wireless Sensor Network per IoT: un paradigma visionario

L'Internet of Things (IoT) è un neologismo utilizzato nelle telecomunicazioni, un nuovo paradigma di comunicazione ad un livello più astratto. L'IoT rappresenta un modo in cui gli oggetti, le "cose", possono automaticamente comunicare tra di loro fornendo servizi a beneficio delle persone. Facendo un esempio, IoT è un frigorifero che ordina il latte quando "si accorge" che è finito; IoT è una casa che accende i riscaldamenti appena ti sente arrivare. Questi sono esempi di IoT, ovvero di oggetti che, collegati alla rete, permettono di mettere in comunicazione mondo reale e virtuale.

La prima definizione di IoT può essere attribuita nel lontano 1999 a Kevin Ashton, ricercatore presso il MIT, Massachussets Institute of Technology, che la usò durante una presentazione presso Procter & Gamble, che la usò per indicare tutti gli oggetti del mondo reale connessi a Internet [5]. Sebbene il termine IoT sia di relativa recente concezione, si parla di questi concetti già da lungo tempo, in pratica dalla nascita di internet e del web semantico (un web fatto di "cose", non di righe di codice: *"things, not strings"*). Tra le molte definizioni che sono state fornite nel corso degli anni, forse, la più significativa è quella citata nell'Internet report dell'ITU del 2005 [6]:

"The next logical step in the technological revolution connecting people anytime, anywhere is to connect inanimate objects. This is the vision underlying the Internet of things: anytime, anywhere, by anyone and anything"

Il termine *Internet of Things*, quindi "Internet delle cose", si riferisce proprio ad oggetti univocamente identificabili, definiti appunto "cose", che sono tra loro collegati e in comunicazione. Questi oggetti possono essere letti, riconosciuti, localizzati, indirizzati e controllati tramite Internet. Il 99% delle cose nel mondo fisico non sono ancora connesse, ecco perchè l'*Internet of Things* può essere definito un sottoinsieme dell'*Internet of Everythings*[1] (IoE).

[1] L'*Internet of Everythings* estende il concetto ad altre tre componenti: persone, processi e dati

L'IoT nasce quindi per rendere possibile l'interazione tra dispositivi, attraverso l'uso di oggetti avanzati, denominati *Smart objects* (SO). Gli SO sono oggetti in grado di ascoltare, interpretare, interagire con l'ambiente circostante ed auto-configurarsi, raccogliendo e condividendo i dati raccolti tra di loro con la possibilità di offrire nuovi servizi ed applicazioni sempre più avanzate. Tali oggetti sono inoltre, molto spesso, dotati di una bassa potenza computazionale, i cui ingredienti sono un' unità di processamento (CPU), un'interfaccia wireless, un'unità di sensing e attuatori. Infine, alcuni oggetti possono essere incorporati in altri dispositivi (per renderli smart), come ad esempio termometri, motori delle auto, stazioni meteo, dispositivi indossabili e interruttori di luci.

L'*Internet of Things*, può anche essere definita una "rete di reti" che promette di consentire a miliardi di esseri umani e macchine di interagire tra loro. A tal proposito, sono le WSN che permettono di concretizzare il significato dell'IoT, creando una rete diffusa di dispositivi (nodi) nell'ambiente circostante, con l'obiettivo, attraverso l'utilizzo di sensori ed attuatori, di raccogliere, trasmettere e processare informazioni. Ad oggi, le tecnologie WSN sono sempre più diffuse, ma ancora troppo spesso basate su protocolli proprietari che non permettono l'interconnessione tra diverse strutture. L'evoluzione tecnologica di questi ultimi anni sta portando alla definizione di *standard* comuni per garantire l'interoperabilità tra dispositivi eterogenei, al fine di far convergere i dati in un'infrastruttura di rete e di servizio unificata, per diverse attività e applicazioni, permettendo inoltre l'interconnessione tra strutture, sia tra di loro, che con il *World Wild Web*.

2.1 Scenari applicativi

La tecnologia IoT trova sempre più consenso nel mercato e rappresenta sempre più una occasione di sviluppo. Si parla di IoT dalle più comuni applicazioni di sensori e attuatori in ambito domenstico *smart home* a quelle per dispositivi indossabili in ambito *health-care*. Per fare un esempio concreto, in ambito urbano, un dispositivo collocato in una strada può controllare l'accensione e lo spegnimento dei lampioni e segnalarne eventuali malfunzionamenti, ma, se appositamente progettato, può raccogliere anche informazioni sulla qualità dell'aria o sulla presenza di persone. Ecco quindi che gli oggetti che ci circondano, dotati di una propria intelligenza, sono in grado raccogliere, fondere insieme, aggregare e immagazzinare dati autonomamente. Sono quindi in grado di monitorare, controllare e trasferire informazioni tra rete internet e mondo reale per poi svolgere azioni conseguenti, come attuare certi comandi o segnalare allarmi.

In generale un sensore rileva i dati della "cosa" che "parla" della sua temperatura, del suo movimento, della qualità della sua aria e "mette in rete" questi

dati.

Possiamo riassumere e classificare i principali ruoli di questi dispositivi in due macro categorie:

- Dispositivi connessi in rete, in grado di rilevare dati e in grado di comunicare i dati;
- Dispositivi connessi in rete, in grado di rilevare dati, di selezionarli, di trasmettere solo quelli necessari ai progetti nel quale sono coinvolti, di effettuare azioni sulla base delle indicazioni ricevute e di effettuare azioni in funzione di una capacità elaborativa locale.

Il mercato globale dell'IoT presenta un trend in continua crescita (Figura 2.1). I principali ambiti di applicazione dell'IoT sono rappresentati da quei contesti nei quali ci sono "cose" che possono "parlare" e generare nuove informazioni come ad esempio:

- Smart cities, smart mobility [7, 8],
- Smart cars [9], [10],
- Monitoraggio ambientale: controllo della qualità dell'aria, della temperatura, del livello dei fiumi, rilevamento di alluvioni o incendi boschivi [11]
- Smart home, domotica [12, 13],
- Healt-care, sanità, mondo biomedicale [14], [15] [16],
- Agricoltura, precision farming, sensori di fields [[17] [18], [19],
- Industria 4.0 [20],
- Early warning system [21, 22],
- Industrial lighting [23],
- ecc..

Chiaramente questi sono solo alcuni tra gli scenari applicativi dell'IoT, alcuni già relativamente consolidati, altri ancora più acerbi e in via di sviluppo, ma sicuramente tutti in continua evoluzione. Infine, la scelta dei più appropriati protocolli e tecnologie di comunicazione sulla base delle singole esigenze realizzative, congiuntamente alla variazione e/o aggiunta delle unità di sensing più appropriate, può consentire l'utilizzo della stessa rete per essere estesa a più contesti applicativi.

Infine, ricerche di settore evidenziano una continua crescita del mercato IoT. L'azienda Cisco[2] prevede che si arriverà a circa 29 miliardi di apparati IoT entro il 2023.

[2]Cisco è una azienda multinazionale specializzata nella fornitura di apparati di networking

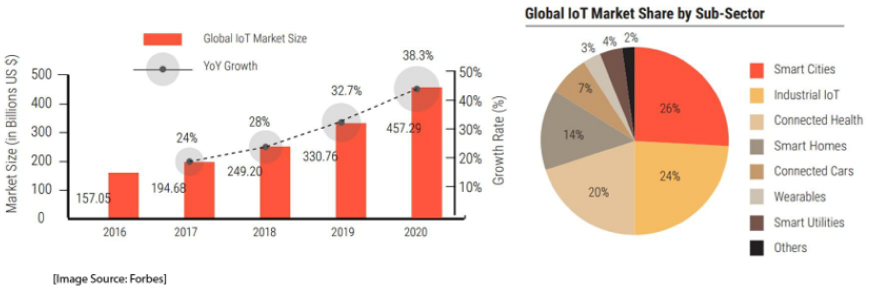


Figura 2.1: Global IoT market size. **Fonte:Forbes**

2.2 Reti di Sensori Wireless per sistemi IoT

Nelle sezione precedente (Sezione 2.1) si è discusso sul significato della parola IoT e su quali siano i ruoli dei dispositivi che realizzino questo concetto. Alla base del paradigma IoT si hanno le Wireless Sensor Network (WSN), che costituiscono la spina dorsale della sua realizzazione.

È possibile definire una WSN, nella sua forma più semplice, come una rete di dispositivi di piccole dimensioni e poco complessi in grado di percepire l'ambiente e comunicare le informazioni raccolte dal campo monitorato tramite collegamenti wireless. I dati raccolti vengono inoltrati in modo cooperativo attraverso la rete in un punto particolare o a un *sink*[3] dove questi possono essere osservati e analizzati.

Ciascun nodo di una WNS è costituito da una alimentazione, un'unità di *sensing*, un processore con il quale effettuare semplici elaborazioni sui dati, come ad esempio *signal processing*, e un trasmettitore wireless. I singoli nodi in una WSN presentano quindi risorse intrinsecamente limitate in riferimento a:

- Velocità di elaborazione.
- Capacità di archiviazione.
- Larghezza di banda di comunicazione.

Le altre caratteristiche principali di una WSN da tenere in considerazioni nella scelta sono:

- Affidabilità.
- Versatilità.

[3] *Sink*, è nodo speciale della rete, detti nodi *sink*, i quali hanno lo scopo di raccogliere i dati e trasmetterli tipicamente ad un *server*

- Numerosità dei nodi.
- Basso consumo energetico.
- Basso costo, dimensioni e peso contenuti per ogni nodo;
- Scalabilità della rete.
- Protocolli con bassa complessità e leggeri.
- Gestione del traffico rispettando i criteri di priorità (QoS).

In fase progettuale di una WSN gli aspetti da considerare sono principalmente: consumo energetico, dimensionamento dei nodi e costo realizzativo. Le dimensioni di una WSN possono essere importanti, dove la mole dei dati e la velocità di trasmissione influenzano le prestazioni della rete oltre che la vita della stessa. Quest'ultimo aspetto è particolarmente importante nel caso di alimentazione a batteria necessaria per molti scenari applicativi d'uso. Inoltre la miniaturizzazione dei nodi rende possibile il contenimento dei costi nella progettazione, in particolare nel caso di una rete molto estesa, oltre che consentire ad un nodo di essere meno invasivo possibile se incorporato in elementi applicativi a seconda dello scenario d'uso. Infine, il giusto trade-off deve essere considerato tra costi ed esigenze progettuali: in generale ogni nodo deve essere *low-cost*, sia nei componenti *hardware* che *software* in una considerevole distribuzione della rete. Per questo motivo è necessario effettuare in prima istanza ponderate scelte progettuali per ottenere il giusto *trade-off* tra gli aspetti descritti precedentemente e la soluzione ottimale per ogni specifica applicazione.

Come affermato all'inizio di questa sezione, le WSN permettono quindi la realizzazione del paradigma IoT, nato per rendere possibile l'interazione dei dispositivi con l'ambiente circostante, mediante l'utilizzo di trasduttori ed attuatori. I trasduttori vengono utilizzati per convertire grandezze fisiche (e.g. segnali biologici, temperatura, umidità, luminosità, ecc) in un segnale elettrico proporzionale che poi può essere interpretato dal dispositivo, mentre gli attuatori convertono il segnale elettrico in una azione (e.g. motori elettrici, relè, interruttori di luci, ecc.).

Un particolare tipo di WSN è quella di tipo mesh[4]. A differenza di una WSN tradizionale che si basa sul rapporto *client-server*, la rete mesh è formata dal sistema *pear-to-pear*, basata su un sistema a maglie. La rete è caratterizzata da una maglia di nodi che si occupano di diffondere e archiviare i dati. L'interconnessione di questi nodi, privi di gerarchia, permette alla rete, configurata in *multi-hop* di funzionare costantemente senza alcuna interruzione, in modo fluido. Nel caso in cui un nodo non funziona più, il sistema non si blocca, ma trova

[4]La rete mesh, in telecomunicazioni, è una topologia di *network* in cui ciascun nodo funge da ripetitore estendendo la copertura del segnale.

una strada alternativa, utilizzando altri nodi attivi. La ridondanza dei percorsi possibili per un dato contenuto informativo inviato da un nodo ad un altro consente di raggiungere più punti, ad esempio, di un appartamento qualora il segnale generato dal nodo che invia è troppo debole per raggiungere il nodo di destinazione. Questo sistema permette di estendere il range di copertura della rete per coprire aree più o meno estese, sfruttando i più appropriati protocolli e tecnologie di comunicazione sulla base delle singole esigenze realizzative.

In conclusione, le WSN di tipo mesh non sono più strutture chiuse, ossia un insieme di trasduttori o attuatori che monitorano determinate grandezze fische o inviano dei comandi proveninienti da e verso un nodo collettore (Sink), il quale elabora e invia le informazioni alla rete internet. Le WSN di tipo mesh sono vere e proprie strutture interconnesse sia tra loro che al *World Wide Web* [24], capaci anche di fare *edge computing*, elaborando i dati il più vicino possibile a dove i dati vengono richiesti. Questo comportamento consente considerevoli vantaggi in termini di latenza di elaborazione, riduzione del traffico dati e maggior resilienza in caso di interruzione nella connessione dati.

2.3 La tecnologia Bluetooth per applicazioni smart lighting

Il settore *smart lighting* rappresenta uno di quegli ambiti applicativi, citati nella sezione 2.1, in cui le tecnologie hanno registrato negli ultimi anni importanti progressi. Rientrano nel concetto di *smart lighting* tutti i sistemi "intelligenti", ovvero quelli dotati di dispositivi che consentono una gestione più potente, raffinata e flessibile rispetto a quella del tradizionale (sistema basato su cablaggi, comandi localizzati e manuali). Questi sistemi sono inoltre in grado di scambiare segnali e dati con altri dispositivi, oggetti e in genere "cose" che, pur essendo a essi estranee, sono però presenti nel contesto ambientale, oppure remote in altri luoghi, ma pur sempre connesse in rete.

Proprio in ambito *lighting* la tecnologia Bluetooth sta avendo negli ultimi anni un forte impatto nel mercato, in particolare la sua versione BLE [25], rivolta ad applicazione IoT per dispositivi *low power*.

Uno dei maggiori beneficiari della tecnologia Bluetooth è il settore del *lighting*. Il settore del *lighting* è il candidato più ovvio per formare la spina dorsale di una rete Bluetooth: è sempre presente ed è collegato alla rete elettrica, quindi non ci sono problemi di disponibilità o di alimentazione. Tanto più che nel 2017 il *Bluetooth Special Interest Group* (SIG)[5] ha rilasciato il protocollo *standard* Bluetooth mesh [26], fondato sul protocollo BLE, che consente di creare reti

[5]Il Bluetooth Special Interest Group è un'organizzazione che sovrintende lo sviluppo degli *standard* Bluetooth e la concessione in licenza delle tecnologie e dei marchi Bluetooth ai produttori.

mesh per soluzioni *smart* e IoT e, fin dall'inizio, concepito prevalentemente per il settore dell'illuminazione intelligente. Una rete Bluetooth mesh può essere intesa come un *framework* per lo sviluppo in larga scala di reti, utilizzando una tecnologia già sicura, affidabile e supportata da tutti gli smarthphone. Infatti, grazie alla semplicità della sua struttura, bassi costi di implementazione, l'assenza di tabelle di routing [27], di connettività di tipo IP [28] e l'interoperabilità con la diffusa tecnologia BLE, la rete Bluetooth mesh è una attraente soluzione per un ampio spettro di applicazioni IoT [29]. La caratteristica principale di questa topologia di rete consiste nell'assenza di un *hub* centrale, dove ogni dispositivo comunica con tutti quelli intorno rientranti nel suo range, fungendo da ripetitore intermedio e facendosi carico di ritrasmettere l'informazione a quelli adiacenti verso il nodo di destinazione. Infine, i corpi illuminanti possono essere scambiati di posto o sostituiti rendendoli facilmente di nuovo operativi tramite la procedura di *Device Firmware Update* (DFU) e indirizzabili in quanto non funzionanti sulla base di tabelle di *routing*. Infine, tutte le applicazioni mesh garantiscono l'interoperabilità tra i prodotti da diversi fornitori e i comportamenti vengono implementati utilizzando modelli *standard*, che verranno descritti in maniera più approfondita nel capitolo 3, definiti da Bluetooth SIG per soddisfare i requisiti specifici di ogni prodotto [30].

Allo stesso modo, i dispositivi che comunicano con il protocollo Bluetooth mesh possono veicolare informazioni aggiuntive, relative a servizi basati sulla tecnologia Bluetooth, rientranti in ambito IoT. Ad esempio, la tecnologia Beacon, relegata fin dai primi anni del suo avvento ad un utilizzo per applicazioni più basilari, solo negli ultimi tempi sta diventando, soprattutto a causa della recente emergenza pandemica[6], tra le principali protagoniste sulla scena del mercato [31]. I *beacon* non vengono più utilizzati solo per ricevere informazioni del tipo *point-of-interest* (POI)[7], ma anche per il monitoraggio di persone in ambienti al chiuso (*indoor navigation asset*), tracciamento di oggetti (*item traking*), piuttosto che per analisi statistiche sull'utilizzo di spazi (*space utilization*). Quindi, se da un lato i *beacon* continuano a fornire utili informazioni ad un utente nell'ambiente in cui si trova (e.g. coupon, informazioni sui voli, tour guidati), dall'altro possono essere utilizzati, se distribuiti in modo capillare in una area più o meno estesa, per raccogliere informazioni utili sui comportamenti dell'utente stesso nonchè per il monitoraggio di parametri di interesse biomedico. Non è difficile immaginare quindi come luci connesse che comunicano tra di loro consentano di prelevare dall'ambiente circostante utili informazioni tramite la comunicazione diretta di uno smartphone o di un dispositivo indossabile; queste informazioni possono essere elaborate internamente,

[6] La tecnologia beacon è stata utilizzata nell'App Immuni, introdotta dal governo italiano per aiutare a combattere l'epidemia di COVID-19

[7] Il compito di una raccomandazione *point-of-interest* (POI) è fornire consigli personalizzati sui luoghi di interesse

facendo *edge computing* ed eventualmente trasmesse ad un *server* locale o ad una piattaforma web integrata con servizi Cloud.

Concludendo, la rete Bluetooth mesh ha dimostrato di essere una soluzione flessibile ed economica adatta per la gestione intelligente di sistemi illuminotecnici, includendo funzionalità specifiche per i sistemi *lighting* e *smart home*, garantendo inoltre l'interoperabilità tra prodotti di diversi produttori [32].

Capitolo 3

Tecnologia Bluetooth e Bluetooth mesh

Il Bluetooth è una tecnologia radio wireless, a corto raggio, definita *open standard*, nata nel 1994 in sostituzione all'utilizzo dei cavi a breve distanza. Il suo campo di applicazione spazia dal più classico utilizzo nei dispositivi wireless, ad esempio per lo streaming audio e video, all'uso in ambito IoT in piccoli dispositivi wireless, quali sensori, dispositivi indossabili, dispositivi medici, ecc. Guidata a partire dal 1998 dal gruppo SIG, negli ultimi due decenni la tecnologia Bluetooth ha rivoluzionato l'IoT e continua la sua espansione ed innovazione sia a livello commerciale che industriale, anche grazie al duro lavoro e la collaborazione tra le aziende associate (ad oggi circa 36 mila).

Fin dalla sua introduzione nel 1998, le spedizioni di dispositivi Bluetooth continuano ad aumentare senza alcun segno di rallentamento: entro il 2021 saranno 48 miliardi i dispositivi connessi a Internet e si prevede che il 30% di essi includerà la tecnologia Bluetooth. Inoltre, entro il 2024 si stima che supereranno i 6 miliardi, come mostrato in Figura 3.1.

Come introdotto nel precedente capitolo, dalla sua nascita, nel luglio 2017, lo *standard* Bluetooth mesh ha rivestito un ruolo di prim'ordine nello sviluppo dei mercati emergenti, dallo *smart building* alla *smart industry*, dalla *smart home* alla *smart city*. In modo particolare, la rete Bluetooth mesh trova nel settore dell'*industrial* e *commercial lighting* la sua principale fonte di ispirazione, rivolgendo attualmente a questi ultimi gran parte della sua attenzione.

Gli ultimi dati di mercato, pubblicati nel 2018 dal gruppo Bluetooth SIG, mostrano che la velocità di adozione della tecnologia Bluetooth mesh ha superato tutte le aspettative: ogni mese vengono introdotti nuovi prodotti che supportano una vasta gamma di servizi smart per soluzioni IoT, dal controllo dell'illuminazione ai sistemi di automazione per edifici [33].

Nel 2019 sono stati 65 i prodotti Bluetooth qualificati con funzionalità mesh, e il numero di nuovi prodotti continua ad aumentare a un ritmo vertiginoso (Figura 3.2). In confronto simili tecnologie concorrenti, rilasciate anni prima

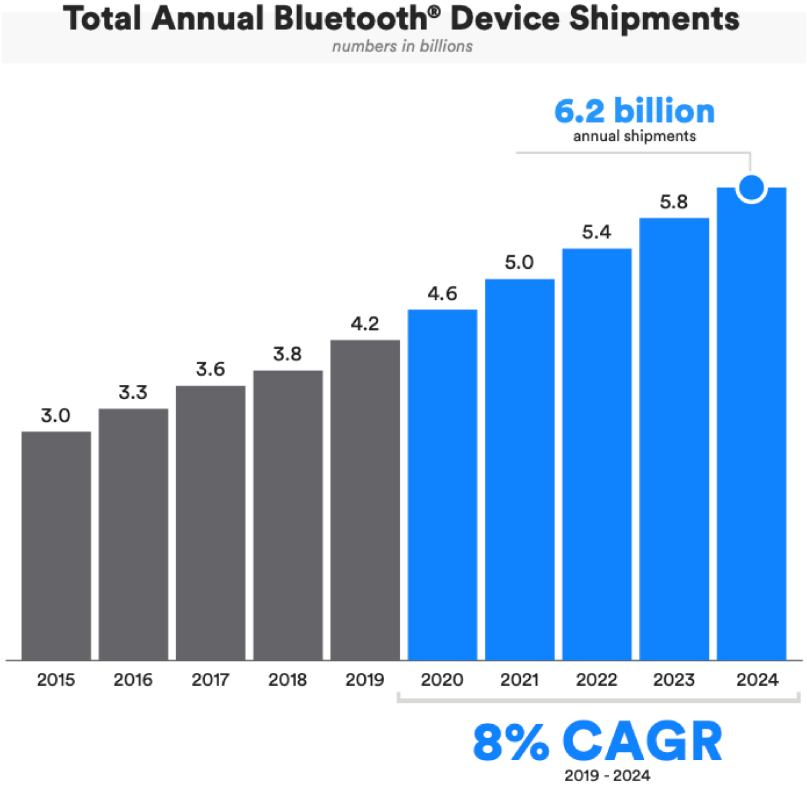


Figura 3.1: Totale dei dispostivi Bluetooth spediti annualmente. **Fonte: Bluetooth SIG**

nel mercato, possono vantare solo una frazione del numero di prodotti qualificati disponibili di cui dispone la tecnologia Bluetooth mesh [33].

In questo capitolo verranno descritte le principali caratteristiche protocolli e i relativi concetti della rete realizzata, con particolare riferimento ad alcuni aspetti appicativi e di ricerca a cui ci si è dedicati in questo lavoro di tesi. Pertanto si partirà con la descrizione dei livelli protocollari più bassi dello *stack* ISO/OSI[1] fino ad arrivare al livello applicazione. A tal fine verrà fornita la descrizione generale dello *stack* protocollare Bluetooth Low Energy, in particolare del livello fisico e del livello di collegamento, proseguendo poi con la descrizione di tutti i livelli del protocollo Bluetooth mesh che poggiano proprio sopra essi.

[1] Il modello Open Systems Interconnection (ISO/OSI) è stato progettato dall'International Organization for Standardization (ISO) come modello di riferimento per consentire una comunicazione aperta tra diversi sistemi tecnici.

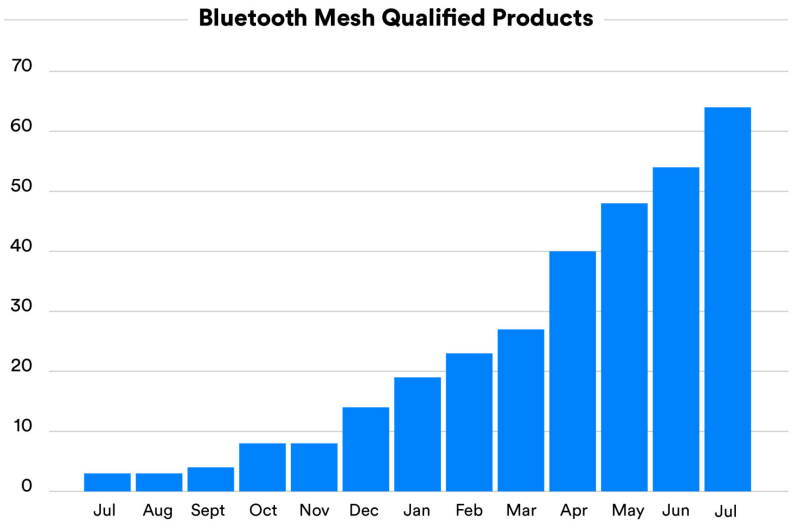


Figura 3.2: Prodotti qualificati Bluetooth mesh. **Fonte: Bluetooth SIG**

3.1 Bluetooth Low Energy

Il Bluetooth è stato rilasciato nella sua prima versione ufficiale da Ericsson nel 1994 ed è attualmente gestito dal gruppo SIG, quest'ultimo fondato nel 1998, dalla stessa Ericsson, IBM, Nokia, Intel e Toshiba (ad oggi comprendente più di 1900 aziende). Nel 2002 il Bluetooth diviene per la prima volta uno *standard*, IEEE 802.15.1, ad opera del *IEEE working group of the Institute of Electrical and Electronics Engineers* (IEEE). Lo *standard* IEEE 802.15.1 è basato sulla versione v1.1 della tecnologia Bluetooth realizzata che ne definisce il livello fisico (PHY) ed il sottolivello MAC[2] della pila ISO/OSI [34].

Ad oggi la tecnologia Bluetooth si è evoluta al punto tale da poterla scindere in due vere e proprie distinte tecnologie:

- *Bluetooth Classic* (BR/EDR): utilizzato in altoparlanti wireless, sistemi di infotaiment per auto e auricolari, in flussi relativamente elevati di streaming audio e video.

- *Bluetooth Low Energy* (BLE): introdotto nella versione Bluetooth 4.0 [25], utilizzata soprattutto nelle applicazioni in cui il consumo di energia è cruciale (come i dispositivi alimentati a batteria) e dove piccole quantità di dati vengono trasferite raramente, come ad esempio nell'uso della sensoristica.

[2] *Mediaum Access Control*

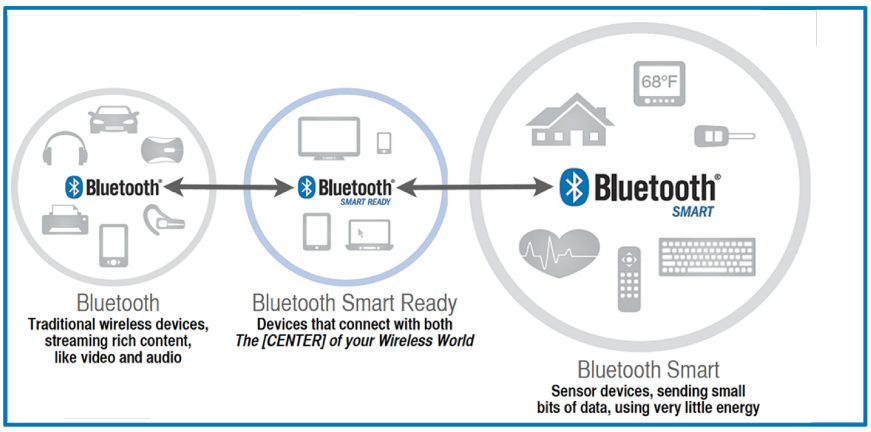


Figura 3.3: Tipi di dispositivi Bluetooth. **Fonte: Aislelabs**

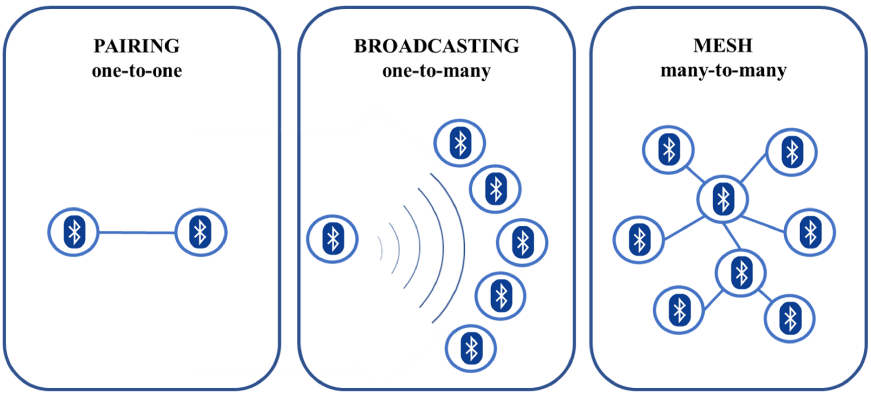


Figura 3.4: Topologie di rete Bluetooth

Queste due tecnologie sono incompatibili tra loro anche se condividono lo stesso *brand* e le stesse specifiche *core*. Per questo, la maggior parte dei dispositivi come gli smartphone, tablet e PC scelgono di implementare entrambi i tipi di tecnologie, e sono denominati comunemente *dual-mode*: ciò consente loro di comunicare con entrambi i tipi di dispositivi, (Figura 3.3). BLE utilizza le stesse frequenze radio a 2.4 GHz del BR/EDR. I dispositivi *dual-mode* condividono una singola antenna radio, ma utilizzano un sistema di modulazione più semplice. Da evidenziare che recenti sondaggi di mercato prevedono che entro il 2024 il 100% dei dispositivi Bluetooth supporterà sia il BR/EDR che il BLE [33].

Inoltre, il BLE ha subito alcune importanti revisioni e modifiche dal suo rilascio ufficiale nel 2010 [25, 35].

Figura 3.5: Stack protocollare BLE

3.2 Stack protocollare e concetti generali

Partendo dal più basso, tre principali blocchi compongono l'architettura del BLE, secondo le specifiche Bluetooth core [25]: *controller*, *host* e *applicazione.*

3.2.1 Controller

Il controller include i seguenti livelli, di seguito brevemente descritti: Host Controller Interface (HCI) lato controller, livello di collegamento (LL) e livello fisico (PHY).

Livello fisico (PHY)

Il livello fisico (PHY) si riferisce ai canali radio, quindi al livello hardware nel dispositivo, usati per comunicare e modulare/demodulare i dati binari (0 e 1) in ricezione/trasmissione. Il BLE opera nella banda ISM[3] (spettro 2.4 GHz), divisa in 40 canali RF, ognuno separato da 2 MHz (*center-to-center*).

In base al tipo di comunicazione che si vuole instaurare tra due o più dispositivi della rete Bleutooth, i canali possono essere così suddivisi:

- *Advertising* primari, 3 canali adibiti alla comunicazione tra due o più dispositivi tramite messaggi in broadcasting, senza l'instaurarsi di alcun tipo di connessione tra di loro.
- *Advertising* secondari, 37 canali usati per trasferire i dati durante la connessione tra due dispositivi.

Ne consegue che la comunicazione tra dispositivi Bluetooth può avvenire essenzialmente tramite due meccanismi: *advertising* e *connecting*, i cui concetti chiave verranno meglio descritti nelle sezioni successive di questo capitolo.

Altri aspetti tecnici importanti relativi al livello fisico sono:

- Uso del *Frequency Hopping Spread Spectrum*(FHSS)[4].
- Potenze massime di trasmissione, ovvero 100 mW (+20 dBm) per il Bluetooth 5 e successive versioni, 10 mW (+10 dBm) per il Bluetooth 4.2 e

[3] ISM, *Industrial, Scientific and Medical*

[4] *Frequency Hopping Spread Spectrum*(FHSS), è una tecnica di trasmissione radio usata per aumentare la larghezza di banda di un segnale; consiste nel variare la frequenza di trasmissione durante la comunicazione tra due dispositivi, a intervalli regolari in maniera pseudocasuale attraverso un codice prestabilito. Ciò migliora notevolmente l'affidabilità e consente ai dispositivi di evitare canali di frequenza che possano essere congestionati e utilizzati da altri dispositivi nell'ambiente circostante.

precedenti versioni ≤ 4.2, come anche le potenze minime di trasmissione, ovvero 0.01 mW (-20 dBm).

- *Data rate*[5] che nel BLE è fisso a 1 Mbps, definito nel livello fisico *1M PHY*, obbligatorio anche nelle versioni successive del Bluetooth 5.

Infine, dal Bluetooth 5 sono stati introdotti anche i nuovi livelli fisici *2Mbps PHY*, usati per raggiungere il doppio della velocità rispetto alle precedenti versioni del Bluetooth, e il *Coded PHY*, usato per la comunicazione a lungo raggio.

Livello di collegamento (LL)

Il livello di collegamento (LL) si interfaccia con il livello fisico e fornisce agli strati di livello superiore un modo per interagire con i canali radio attraverso un livello intermedio, chiamato *Host Controller Interface* (HCI). L'HCI standardizza la comunicazione tra *controller* e *host*: può essere interno o esterno al *chipset*. Il LL è inoltre responsabile della gestione dello stato[6] dei canali radio, come vedremo meglio in seguito, e dei requisiti di temporizzazione necessari per soddisfare le specifiche BLE e della gestione delle operazioni con accelerazione hardware.

Sintetizzando, il livello di collegamento gestisce gli stati principali in cui operano i dispositivi BLE (canali radio) *advertising*, *scanning* e *connected*. In generale, quando un dispositivo è in *advertising*, consente ad altri dispositivi, che stanno scansionando la rete in *scanning*, di trovare il dispositivo e, se possibile, connettersi ad esso. Se il dispositivo che fa *advertising* non consente la connessione, il dispositivo in *scanning* riceve solamente i dati; altrimenti se consente la connessione, il dispositivo in *scanning* decide o no di connettersi ad esso, dopodichè entambi entrano nello stato *connected*.

Direct Test Mode (DTM)

Il livello *Direct test mode* (DTM) consente di eseguire i test sui parametri RF a livello PHY, utilizzata sia durante la produzione che per i test di certificazione sul prodotto finale.

Host Controller Interface (HCI)

Il livello *Host controller interface* (HCI) è un protocollo *standard* definito dalle specifiche Bluetooth che consente all'*host* di comunicare con il *controller*. Tali livelli possono essere separati o coesistere nello stesso *chipset*. In tal senso,

[5] Il *data rate* è la velocità dei dati è la velocità con cui i dati vengono trasferiti attraverso alcune telecomunicazioni o mezzi di calcolo

[6] Modalità operativa in cui si trova un dispostivo.

consente anche l'interoperabilità tra i *chipset*, quindi uno sviluppatore può scegliere due dispositivi certificati Bluetooth, un *controller* e un *host*, ed essere sicuro al 100% che essi siano compatibili tra loro[7].

Il compito del livello HCI è trasmettere i comandi dall'host al controller e inviare di nuovo gli eventi dal controller all'host. Esempi di messaggi includono: pacchetti di comando, configurazione del controller, richiesta di azioni, controllo della connessione e dei parametri di connessione, pacchetti di eventi, completamento del comando ed eventi di stato.

3.2.2 Host

L'host è composto dai seguenti livelli:

- Controllo del collegamento logico e protocollo di adattamento (L2CAP).
- Protocollo gestore della sicurezza (SMP).
- Profilo di accesso generico (GAP).
- Protocollo attributi (ATT).
- Profilo attributo generico (GATT).
- Host Controller Interface (HCI) lato *host*

Controllo del collegamento logico e protocollo di adattamento (L2CAP)

Il livello controllo del collegamento logico e protocollo di adattamento (L2CAP) funge da livello di *multiplexing* del protocollo. È preso in prestito dallo *standard* Bluetooth *Classic* ed esegue le seguenti attività nel caso del BLE:

- Prende più protocolli dai livelli superiori e li inserisce in pacchetti BLE *standard* che vengono passati agli strati inferiori sottostanti.
- Gestisce la frammentazione e la ricombinazione. Sul lato trasmettitore, prende i pacchetti più grandi dagli strati superiori e li divide in blocchi che si adattano alla dimensione massima del carico utile BLE supportata per la trasmissione. Sul lato ricevente, prende più pacchetti e li combina in un pacchetto che può essere gestito dagli strati superiori.

Per il BLE, il livello L2CAP gestisce due protocolli principali: Attribute Protocol (ATT) e Security Manager Protocol (SMP) trattati nel seguito di questa sezione.

[7] Nel caso in cui l'*host* e il *controller* si trovino in *chipset* separati, il livello HCI verrà implementato su un'interfaccia di comunicazione fisica. Le tre interfacce hardware ufficialmente supportate dalle specifiche sono: UART, USB e SDIO (Secure Digital Input Output). Nel caso in cui i due livelli (host e controller) risiedano sullo stesso *chipset*, il livello HCI sarà invece un'interfaccia logica.

Protocollo gestore della sicurezza (SMP)

Il protocollo gestore della sicurezza (SMP) è utilizzato per generare chiavi di crittografia; esso opera su un canale L2CAP dedicato e gestisce anche l'archiviazione delle chiavi di crittografia e di identità. Si interfaccia direttamente con il *controller* per fornire chiavi memorizzate durante le procedure di crittografia o associazione.

Profilo di accesso generico (GAP)

Il livello profilo di accesso generico (GAP) è il profilo che fornisce un *framework* per definire come i dispositivi BLE possono interagire l'uno con l'altro, cioè ne definisce i loro ruoli. In particolare, vengono presi in considerazione i seguenti aspetti:

- Ruoli dei dispositivi: trasmissione, ricezione o entrambi i ruoli.
- Operazioni di *advertising* o *scanning*, ed i relativi parametri, come anche la trasmissione dei dati del *payload*.
- Operazioni per la connessione: inizializzazione, accettazione e parametri di connessione e connection events.
- Aspetti riguardanti la sicurezza e l'affidabilità dei dati trasmessi: CRC[8], *acknowledgment*[9]., ritrasmissioni, operazioni di criptazione e decriptazione dei dati.

In definitiva, questa parte del protocollo che consente ai dispositivi di comunicare tra loro è obbligatoria e parte delle specifiche Bluetooth. In pratica il livello GAP si occupa principalmente di definire il pacchetto dati da inviare con tutte le informazioni necessarie alla propria identificazione e alla trasmissione del suo contenuto informativo, come:

- Nome del dispositivo Bluetooth LE remoto.
- Indirizzo MAC del dispositivo remoto.
- Dati di servizio che sono i dati associati a un servizio.
- Dati specifici del produttore.

Nelle prossime sezioni verrà descritto più nel dettaglio, tramite il caso d'uso dei *beacon*, come avvenga la comunicazione tra dispositivi nei due stati *advertising* e *connecting*, come anche i ruoli che essi assumono nella rete.

[8] CRC *Cyclic Redundancy Check*, generazione di numeri casuali e crittografia.
[9] Acknowledgment (ACK), in ambito delle telecomunicazioni e informatico, identifica un segnale di *acknowledge* emesso in risposta alla ricezione di un'informazione

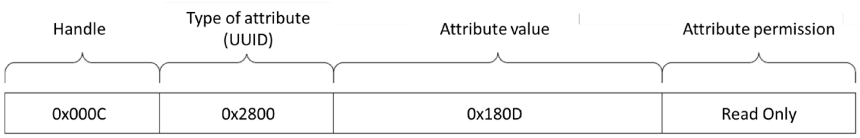


Figura 3.6: Rappresentazione logica di un attributo

Protocollo attributi (ATT)

Il livello protocollo attributi (ATT) è il protocollo utilizzato per trasportare i dati sotto forma di comandi, richieste, risposte, indicazioni notifiche e conferme tra dispositivi. Il protocollo ATT definisce due ruoli: *server* e *client*. Il *server* è il dispositivo che accetta comandi e richieste, in arrivo dal *client* e a cui invia risposte, indicazioni e notifiche. Il *client* è il dispositivo che genera comandi e richieste verso il *server* e può ricevere risposte, indicazioni e notifiche inviate dal *server*. L'ATT definisce il modo in cui un *server* espone i suoi dati al *client* e come questi dati sono strutturati. Un *server* espone i suoi dati sotto forma di attributi che possono essere notificati, scritti o letti da un *client* specifico. Inoltre ogni attributo è composto da quattro parti: ***handle***, ***tipo***, ***valore*** e ***permessi***. La Figura 3.6 mostra una rappresentazione logica di un attributo, in riferimento ad una sua tipica applicazione, chiamata *Heart Rate Profile*.

L'*handle* è un indice corrispondente ad un attributo specifico. Il *tipo* dell'attributo è definito *universally unique identifier* (UUID) che specifica ciò che esso rappresenta. Il *valore* è il dato descritto dal tipo dell'attributo e indicizzato dall'*handle*.

Profilo attributo generico (GATT)

Il livello profilo attributo generico (GATT) è un profilo implementato sopra all'ATT e definisce le operazioni comuni da compiere per entrambi i dispositivi e il *framework* per i dati trasportati ed immagazzinati dall'ATT (servizi e caratteristiche). È fondamentale tenere presente che il profilo GATT entra in gioco solamente dopo che due dispositivi hanno stabilito una connessione tra di loro. Le autorizzazioni vengono utilizzate dal *server* per determinare se è consentito l'accesso in lettura o in scrittura per un determinato attributo: queste ultime sono stabilite proprio dal profilo GATT. Il profilo GATT definisce quindi una gerarchia ben precisa, come mostrata in Figura 3.7.

Il profilo GATT fa uso di concetti come **profili**, **servizi**, **caratteristiche** e **attributi**, necessari per poter incapsulare correttamente e trasferire le informazioni esposte dal *server* in una struttura gerarchica. Il profilo GATT definisce quindi il formato (struttura) dei dati esposti da un dispositivo BLE, le procedure necessarie per accedere ai dati esposti da un dispositivo. Risponde

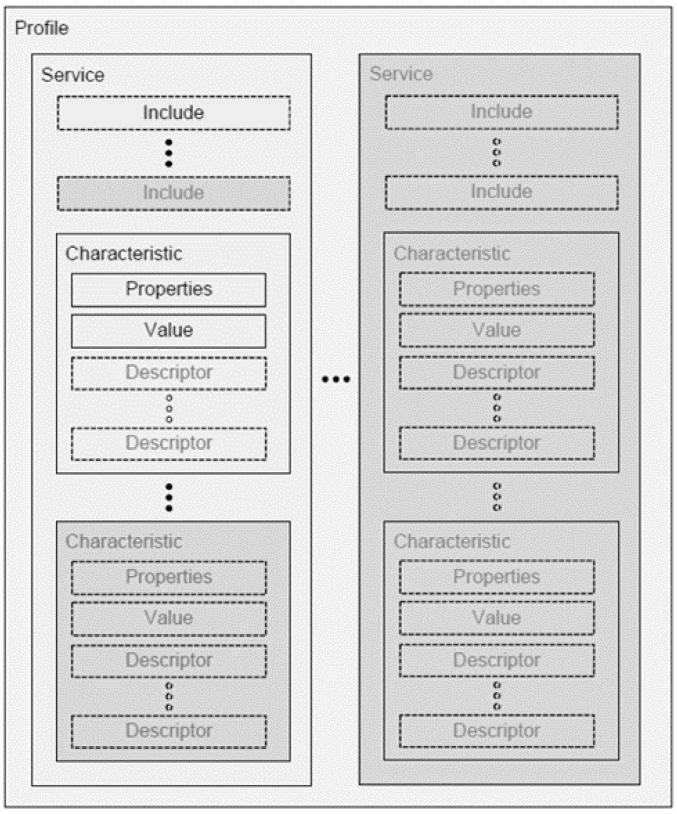


Figura 3.7: Gerarchia del profilo GATT

inoltre alle richieste di lettura e scrittura ed invia notifiche ai *client* sottoscritti. Un servizio è una raccolta di dati e comportamenti associati per realizzare una particolare funzione o caratteristica. Nel profilo GATT, una definizione di servizio può contenere servizi di riferimento, caratteristiche obbligatorie e caratteristiche opzionali. Esistono quindi due tipi di servizi: primari e secondari. Un servizio primario descrive la funzionalità principale di un dispositivo. Un servizio secondario è un servizio all'interno di un altro servizio primario, un altro servizio secondario o altra specifica di livello superiore, anche se il suo utilizzo nella pratica è raro. Una dichiarazione di servizio è un attributo con il tipo impostato sull'UUID per «Servizio primario» (0x2800) o «Servizio secondario» (0x2801). Il campo valore può essere un UUID a 16 o a 128 bit. Le autorizzazioni di attributo devono essere di sola lettura e non richiedono autenticazione (Figura 3.8).

Una caratteristica è un attributo utilizzato in un servizio con proprietà e informazioni su come viene visualizzato o rappresentato il valore e su come accedervi. Una dichiarazione di una caratteristica è un attributo con il tipo

Attribute Handle	Attribute Type	Attribute Value	Attribute Permission
0xNNNN	0x2800 – UUID for «Primary Service» OR 0x2801 for «Secondary Service»]	16-bit Bluetooth UUID or 128-bit UUID for Service	Read Only, No Authentication, No Authorization

Figura 3.8: Dichiarazione di un servizio

Attribute Handle	Attribute Type s	Attribute Value			Attribute Permissions
0xNNNN	0x2803–UUID for «Characteristic»	Characteristic Properties	Characteristic Value Attribute Handle	Characteristic UUID	Read Only, No Authentication, No Authorization

Figura 3.9: Dichiarazione di una caratteristica

Attribute Value	Size	Description
Characteristic Properties	1 octets	Bit field of characteristic properties
Characteristic Value Handle	2 octets	Handle of the Attribute containing the value of this characteristic
Characteristic UUID	2 or 16 octets	16-bit Bluetooth UUID or 128-bit UUID for Characteristic Value

Figura 3.10: Campo value nella dichiarazione della caratteristica

impostato sull'UUID «Caratteristica» (0x2803) e il valore dell'attributo diviso in *proprietà*, *handle* e *UUID*. Le autorizzazioni devono essere leggibili e non richiedono autenticazione o autorizzazione (Figura 3.9)

Il campo *valore* di una caratteristica è disponibile solamente in lettura, come mostrato in Figura 3.10.

- Il campo *proprietà* determina come è possibile utilizzare il valore o come accedere ai descrittori delle caratteristiche.
- Il campo *handle* contiene l'handle dell'attributo in cui è presente il valore della caratteristica.
- Il campo *UUID* è un UUID Bluetooth a 16 bit o UUID generico a 128 bit che descrive il tipo di valore della caratteristica.

I descrittori delle caratteristiche contengono informazioni correlate al valore della caratteristica. Il profilo GATT definisce un set *standard* di descrittori che possono essere utilizzati da **profili** di livello superiore. Ogni descrittore di

Attribute Handle	Attribute Type	Attribute Value	Attribute Permissions
0xNNNN	02902 – UUID for «Client Characteristic Configuration»	Characteristic Configuration Bits	Readable with no authentication or authorization. Writable with authentication and authorization defined by a higher layer specification or is implementation specific.

Figura 3.11: Dichiarazione Client Characteristic Configuration

Configurazione	Valore	Descrizione
Notifiche	0x0001	Il campo value sarà notificato
Indicazioni	0x0002	Il campo value sarà indicato

Tabella 3.1: Definizione dei Characteristic Configuration Bits

caratteristiche è identificato dall'UUID. Il *Client Characteristic Configuration Descriptor* (CCCD) è un descrittore di caratteristiche facoltativo che definisce come la **caratteristica** possa essere configurata da uno specifico *client*. Un *client* può scrivere nel CCCD per controllare la configurazione della **caratteristica**. Possono essere richieste autenticazione e autorizzazione dal *server* per scrivere nel descrittore. Il descrittore è contenuto in un attributo il cui tipo deve essere impostato sull'UUID come «Client Characteristic Configuration» (0x2902). Il CCCD agisce come un interruttore, abilitando o disabilitando gli aggiornamenti del campo **value** del *characteristic value* della caratteristica in cui si trova. Il suo valore è un bitfield a due bit, uno corrispondente alle notifiche e l'altro alle indicazioni (Figura 3.11).

I Characteristic Configuration Bits possibili sono elencati in Tabella 3.1.

3.2.3 Applicazione

Il livello applicazione dipende dal caso d'uso e si riferisce all'implementazione sopra al livello GAP e Generic GATT, che si occupano di gestire rispettivamente la logica con cui due dispositivi possono comunicare tra di loro e come l'applicazione dello sviluppatore può gestisce i dati ricevuti e inviati ad altri dispositivi.

3.2.4 Advertising e scanning

Ogni dispositivo BLE inizia sempre a trasmettere nello stato di *advertising*. Questo è anche il caso in cui si voglia operare nello stato *connecting* la maggior parte del tempo. Affinché due dispositivi BLE si individuino l'un l'altro, uno di loro deve entrare in modalità *advertising* sui tre canali 37, 38 e 39. L'altro in

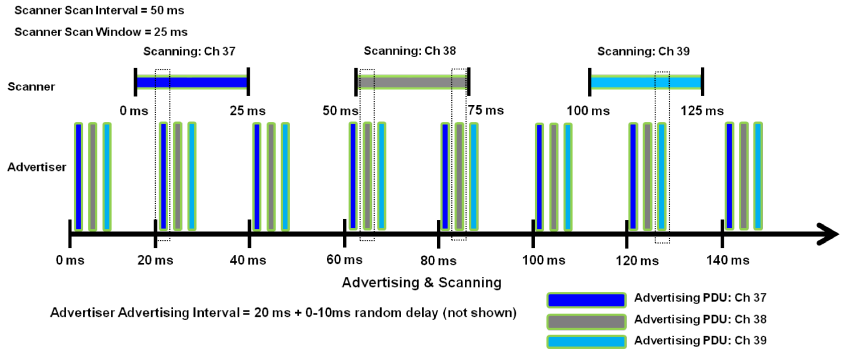


Figura 3.12: *Advertising* e *scanning* BLE. **Fonte: microchipdeveloper.com, Bluetooth Low Energy Discovery Process**

Figura 3.13: Esempio di comunicazione tra due dispositivi in modalità *advertising*

modalità *scanning* deve scansionare sui gli stessi canali omologhi alla ricerca di pacchetti di *advertising*. Dopodichè quest'ultimo dispositivo, chiamato centrale, potrà scegliere se connettersi o no al primo dispositivo, chiamato periferico (Figura 3.13).

Nello stato di *advertising* quindi un dispostivo invia un pacchetto contenente dati ad intervalli regolari, definiti come *advertising interval*[10], consecutivamente in ognuno dei 3 canali primari di *advertising*. Dall'altro lato il dispositvo centrale, alla ricerca di altri dispositivi, ascolterà quei canali per i pacchetti di *advertising*, utilizzando un intervallo di scansione per ognuno dei canali, definito *scan window*. Questa procedura, di per sé auto-esplicativa, è mostrata in Figura 3.12. Per massimizzare la possibilità che ciò accada e per farlo accadere rapidamente, è possibile regolare i parametri di *advertising* e *scanning*.

I dati di *advertising* principali sono limitati a 31 byte. Il protocollo BLE prevede inoltre l'adozione di pacchetti dati, denominati di *advertising* secondari, fino a 254 byte, che consentono ad un dispositivo periferico di inviare ulteriori

[10] *advertising interval*, secondo specifiche Bluetooth da 20 ms a 10.48 s

dati di *advertising* che non rientrerebbero nel pacchetto *advertising* iniziale, qualora il dispositivo centrale ne faccia esplicita richiesta.

Se da un lato questo approccio permette a un dispositivo di trasmettere il suo contenuto informativo a molti dispositivi in ascolto, senza bisogno di connessione, dall'altro soffre della mancanza di una adeguata sicurezza nella trasmissione dei dati, oltrechè l'impossibilità di ricevere dati da un dispositivo centrale: il trasferimento dei dati è infatti unidirezionale.

Differenti tipi di pacchetti inviati, sono inseriti nel pacchetto Bluetooth nel campo *Packet Data Unit* (PDU) types all'interno del PDU header. Esistono diversi tipi di PDU types per l'advertising di un dispostivo periferico, ma i più comuni sono:

- ADV_IND: non diretto ad un dispostivo centrale specifico ed accetta la connessione.
- ADV_NONCONN_IND: usato quando un dispositivo periferico non vuole accettare una connessione, tipicamente il caso di un *beacon*.

Ai fini della trattazione che seguirà nei prossimi capitoli, verrà qui di seguito descritta la composizione di un pacchetto BLE di 31 byte. La Figura 3.14 mostra la tipica composizione di questo tipo di pacchetto, composto da:

- *Preambolo*: è un valore di 1 byte utilizzato per la sincronizzazione e la stima dei tempi sul ricevitore ed è sempre 0xAA per i pacchetti trasmessi
- Access address: un valore fisso ed è impostato su 0x8E89BED6
- *Packet data unit* (PDU): composto a sua volta da:
  - *PDU header*: contiene informazioni sul tipo di pacchetto, in particolare il PDU type definisce la modalità di funzionamento del dispostivo. Il bit TxAdd indica se l'indirizzo dell'advertiser, contenuto nel payload, è pubblico (TxAdd = 0) o random (TxAdd = 1). RxAdd è riservato ad altri tipi di pacchetti non trattati in questa sezione, poiché non si applicano ai beacon.
  - *Data payload*: contiene informazioni sui dati del frame.
- Cyclic Redundancy Check (CRC): è un codice di rilevamento degli errori utilizzato per convalidare il pacchetto e garantire l'integrità dei dati per tutti i pacchetti trasmessi.

La modalità *advertising* utilizza il livello GAP per trasmettere dati a chiunque sia in ascolto, in modalità *one-to-many*. Alcuni dispositivi come ad esempio la maggior parte dei *beacon* sfruttano proprio questo tipo di comunicazione per inviare i propri dati in *broadcasting* senza accettare richieste di connessione, utilizzando pacchetti periodici appositamente formattati, come vedremo nella prossima sottosezione.

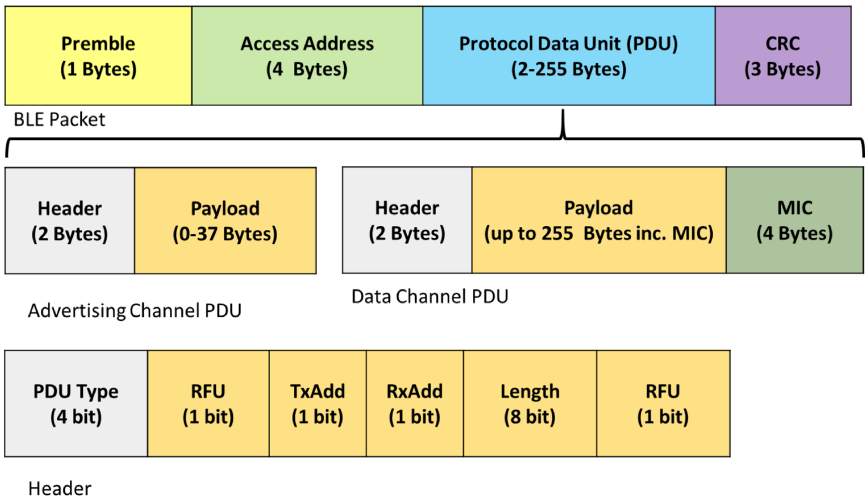


Figura 3.14: Pacchetto BLE

3.2.5 Connecting

Un dispositivo centrale che riceve inizialmente un pacchetto di *advertising* può avviare una connessione inviando un pacchetto del tipo *CONNECT_IND*, se un dispositivo periferico lo consente. Esso è dunque in grado di leggere tutte le informazioni associate al *server* al fine di stabilire la connessione. Nella Figura 3.15 sono mostrati tutti gli step associati all'istaurarsi della connessione tra due dispostivi. Dopodichè la connessione viene considerata creata, ma non ancora stabilita. Una connessione è considerata stabilita una volta che il dispositivo riceve un pacchetto dal suo dispositivo *peer*. Una volta stabilita la connessione, il dispositivo centrale diventa nota come *master* e la quello periferico diventa lo *slave*. Il master è responsabile della gestione della connessione, del controllo dei suoi parametri e della temporizzazione dei diversi eventi al suo interno [25].

3.3 I beacon

Un *beacon* è un piccolo trasmettitore che emette un segnale a radio frequenza continuo e periodico (ad esempio la sua identificazione o posizione). Questo dispositivo è basato sulla tecnologia BLE Beacon [36], basata sulla trasmissione di piccole quantità di informazioni che possono contenere dati ambientali (temperatura, pressione dell'aria, umidità ecc.), dati di micro-localizzazione (tracciamento degli oggetti, vendita al dettaglio, ecc.) o altri tipi di dati (accelerazione, rotazione, parametri di interesse biomedico ecc.). I *beacon* sono

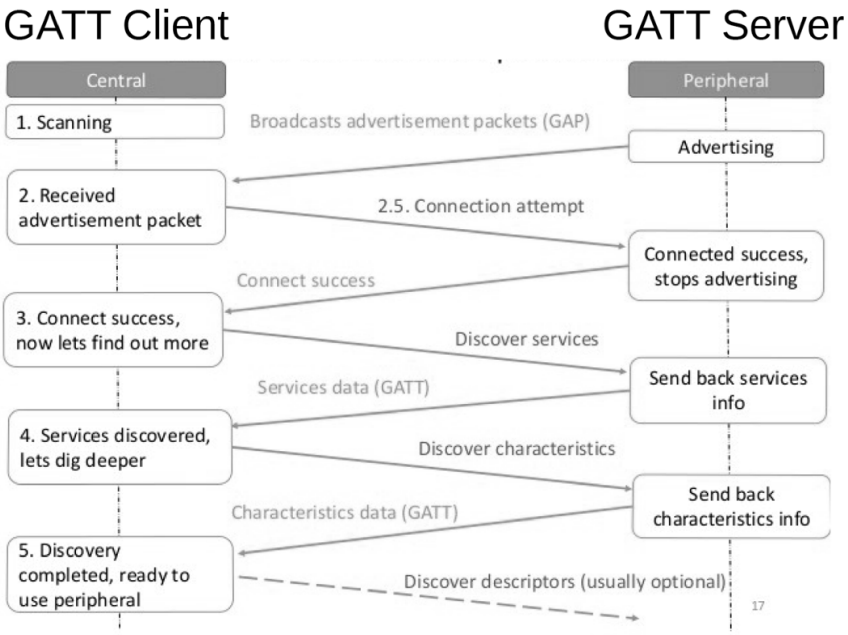


Figura 3.15: Connecting BLE

basati sulla tecnologia BLE e per comunicare inviano un segnale contenente un ID univoco ad un dispositivo ricevitore dotato della stessa tecnologia Bluetooth (smartphone) capace di ricevere ed interpretare l'informazione in esso contenuta. Nella maggior parte dei casi il dispositivo ricevente nel range del *beacon* sottoscrive a questo ID e triggera un'azione in risposta alla sua ricezione. Queste informazioni possono essere definite statiche o dinamiche, ovvero possono essere modificate in qualunque momento dall'utente.

Come precedentemente descritto, il BLE ha la capacità di scambiare dati in due stati: *connecting* e *advertising*. La modalità *connecting* utilizza il livello GATT per trasferire i dati in una configurazione 1:1 tra due dispositivi, con possibilità quindi di ricevere oltre che trasmettere, consentendo ad un dispositivo centrale (ad esempio uno smartphone) di connettersi e interagire con i servizi implementati sul dispositivo periferico, *beacon*, per modificarne i dati al suo interno. La modalità *advertising* utilizza il livello GAP per trasmettere dati in *broadcasting* a chiunque sia in ascolto, in modalità 1:M, con minor consumo energetico, dato che dispositivo si sveglia da uno stato di *sleep*, trasmette dati e torna in *sleep*, ma senza la possibilità di ricevere i dati e quindi modificarne i propri.

3.3.1 iBeacon

L'iBeacon è uno *standard* proprietario, ideato da Apple nel 2013 [37], ed anche la prima tecnologia BLE Beacon ad essere stata sviluppata. Ad oggi la maggior parte dei *beacon* si ispira ancora al formato dati iBeacon, sebbene questo formato presenti diverse limitazioni nel suo utilizzo, come si vedrà in seguito. In particolare, gli iBeacon trasmettono quattro tipi di informazioni:

- *UUID*, un valore che identifica il *beacon*.
- *Major*, un valore che identifica un sottoinsieme di *beacon* all'interno di un gruppo più grande.
- *Minor*, un valore che identifica uno specifico *beacon*.
- *TX power*[11] è un numero che deve essere calibrato dall'utente o dal produttore del dispositivo.

Un'applicazione che fa la scansione di *beacon* legge l'*UUID*, il *Major* e il *Minor* facendo riferimento in genere ad un suo un database interno per ottenere informazioni sul *beacon* poiché lo stesso non porta informazioni descrittive. Il campo di *TX power* viene utilizzato per determinare la distanza del *beacon* dallo smartphone, ovvero l'RSSI[12]. Il formato *data payload* iBeacon (31 bytes) è mostrato in Figura 3.16.

L'*iBeacon Prefix* contiene i dati esadecimali 0x0201061AFF004C0215 e si scompone come segue:

- 0x020106 definisce il pacchetto come BLE *General Discoverable*, cioè si tratta di un pacchetto di sola trasmissione e non di connessione.
- 0x1AFF specifica che il dato seguente è lungo 26 bytes e che sono di tipo *Manufacturer Specific*.
- 0x004C è l'identificativo Apple del Bluetooth SIG
- 0x02 è un ID secondario che denota un *beacon* di prossimità.
- 0x15 definisce che la lunghezza rimanente del dato, cioè 21 bytes.

I restanti campi sono piuttosto auto-esplicativi:

- *UUID* è un identificativo *standard* a 16 bytes ed è in genere univoco per un'azienda

[11] TX power corrisponde al segnale RSSI (Received Signal Strength Indicator) misurato ad 1 metro di distanza.

[12] Nelle telecomunicazioni, l'indicatore di potenza del segnale ricevuto (RSSI) è una misura della potenza presente in un segnale radio ricevuto

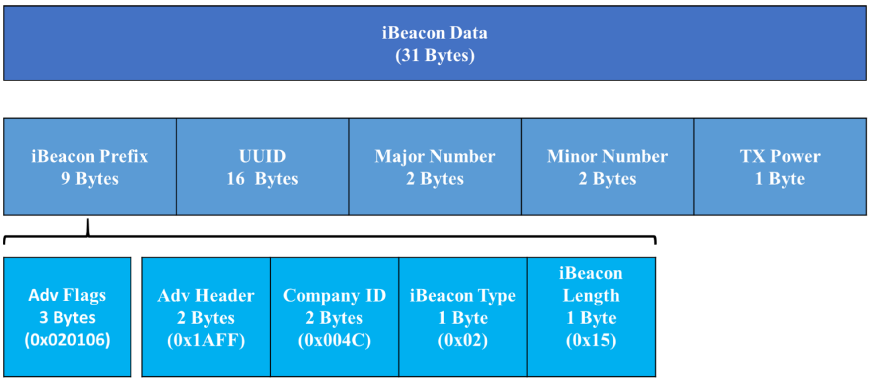


Figura 3.16: Dati iBeacon

- *Major* e *Minor* sono dei valori utilizzati per identificare, rispettivamente, ad esempio un negozio, quindi 65.536 negozi possibili, e dei singoli *tag* all'interno dei negozi, quindi 65.536 *tag* possibili per ogni negozio.

3.3.2 Eddystone

Eddystone, sviluppato da Google nel 2015, è progettato pensando alla trasparenza e alla flessibilità nel suo utilizzo. Infatti è un formato *open standard* con diversi tipi di *data payload* [38] che possono essere inclusi nel formato del *frame*, come mostrato in Figura 3.17, tra cui:

- Eddystone-UID: un ID statico univoco composto da un Namespace di 10 byte e un componente Instance di 6 byte
- Eddystone-URL: un URL compresso che, una volta analizzato e decompresso, è direttamente utilizzabile dal *client*
- Eddystone-TLM: dati sullo stato del *beacon* utili per la manutenzione e che alimentano l'endpoint diagnostico dell'API *beacon* di Google Proximity. Deve essere integrato con un frame UID o EID (per cui la versione crittografata di eTLM preserva la sicurezza)
- Eddystone-EID: un frame *beacon* variabile nel tempo che può essere un identificatore stabile ricevuto da un resolver collegato, come l'API *Proximity Beacon*

3.4 Bluetooth mesh

Molti produttori di illuminazione connessa negli anni hanno dovuto sviluppare i propri *standard* proprietari per realizzare la propria rete mesh [39]. Ciò

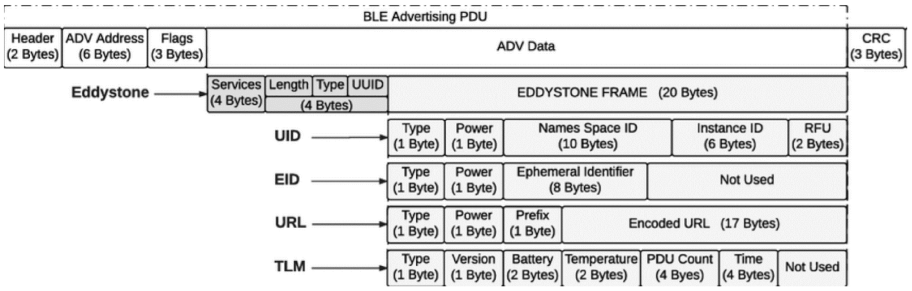


Figura 3.17: Eddystone Data. **Fonte:** [1]

ha consentito implementazioni su larga scala, ma a scapito dell'interoperabilità. Tutto è cambiato nel luglio 2017 quando Bluetooth SIG ha rilasciato lo *standard* Bluetooth mesh [26]. Piuttosto che avere ogni dispositivo nella rete che comunica con un *hub* centrale, in una rete Bluetooth mesh, come in altre tipologie di reti mesh, come ad esempio quelle basate su ZigBee [40], Thread [41] o soluzioni proprietarie [42], ogni dispositivo può comunicare con uno qualsiasi degli altri attorno ad esso purché sia nel suo raggio d'azione. Ciò rende le dimensioni e l'area della rete praticamente illimitate, motivo per cui è così utile per le applicazioni *industrial* IoT come le grandi reti di sensori connesse.

Bluetooth Mesh offre quindi due vantaggi principali:

- Aumento della portata della rete, in cui ogni dispositivo non deve trovarsi entro 20-30 metri da un *hub*, ma solo entro 20-30 metri da un altro dispositivo.
- Aumento della resilienza della rete, in cui non esiste alcun *single failure point*: se un nodo smette di funzionare, i dati possono aggirarlo e passare al nodo successivo nel raggio di azione.

Ci sono inoltre altre differenze fondamentali tra il protocollo Bluetooh mesh ed altri protocolli mesh (incluso Thread, Zigbee). Tra le più importanti:

- Bluetooth mesh non utilizza il protocollo internet (IP) per funzionare, dato che è basato interamente sui livelli sottostanti dello stack protocollare BLE;
- Bluetooth mesh utilizza la tecnica del *managed floodig*, descritta in seguito, in comparazione con tecniche di *routing* adottate da altri protocolli (Thread, Zigbee).

In una rete Bluetooth mesh la comunicazione avviene in modalità *many-to-many* (M:M) 3.4, attraverso la trasmissione di messaggi in *broadcasting*, seguendo i meccanismi di *advertising* e *scanning* propri della comunicazione

BLE tra un dispositivo ed un altro della rete. La comunicazione risulta essere quindi di tipo *message-oriented* e gestita tramite il rispetto di alcune regole di comunicazione tra i dispositivi, con elevati livelli di sicurezza. Queste regole saranno spiegate nel seguito di questa sezione.

Bluetooth mesh richiede uno *stack* BLE completo per essere in esecuzione sul dispositivo. Supporta infine anche lo stato *connected* e la GATT, che consente a dispositivi speciali, chiamati nodi *proxy*, di comunicare ad esempio con uno smartphone,

Tuttavia, il Bluetooth mesh specifica un livello *host* completamente nuovo, e anche se alcuni concetti sono condivisi, Bluetooth mesh è incompatibile con il livello *host* del BLE. Bluetooth mesh è un nuovo protocollo di comunicazione, ma nello stesso tempo poggia sui livelli più bassi della pila ISO/OSI, ereditando alcune tra le caratteristiche basilari del funzionamento del BLE. Ne deriva quindi la sua retrocompatibilità con i dispositivi basati su tecnologie Bluetooth precedenti (4.0,4.1,4.2,5), tramite solamente un aggiornamento del *firmware*.

3.4.1 Stack protocollare

In questa sezione verrà esaminato lo stack protocollare Bluetooth mesh. In Figura 3.18, nella parte inferiore, è mostrato lo *stack* BLE (livello fisico e livello di collegamento) necessario per fornire funzionalità di comunicazione fondamentali sfruttate dall'architettura mesh che si trova al di sopra di esso. Lo stack mesh dipende quindi dalla disponibilità di uno stack BLE. Nel seguito verranno descritti brevemente tutti i livelli che compongono lo stack mesh, partendo da quello più basso.

Livello bearer

Il livello *bearer* definisce il modo in cui vengono gestiti i diversi PDU mesh da un determinato sistema di comunicazione. Vengono definiti due tipi di livelli *bearer*:

- *Advertising bearer*: sfrutta gli stati *advertising* e *scanning* propri del GAP (BLE) per trasmettere e ricevere PDU di tipo mesh.

- GATT *bearer*: utilizza il protocollo *proxy* per consentire a un dispositivo che non supporta l'*advertising Bearer* di comunicare indirettamente con nodi di una rete mesh. Un nodo *proxy* ha in sè le caratteristiche GATT: supporta il GATT *bearer* e l'*advertising bearer* in modo che possa operare da traduttore per l'inoltro dei messaggi tra i due tipi di *bearer*.

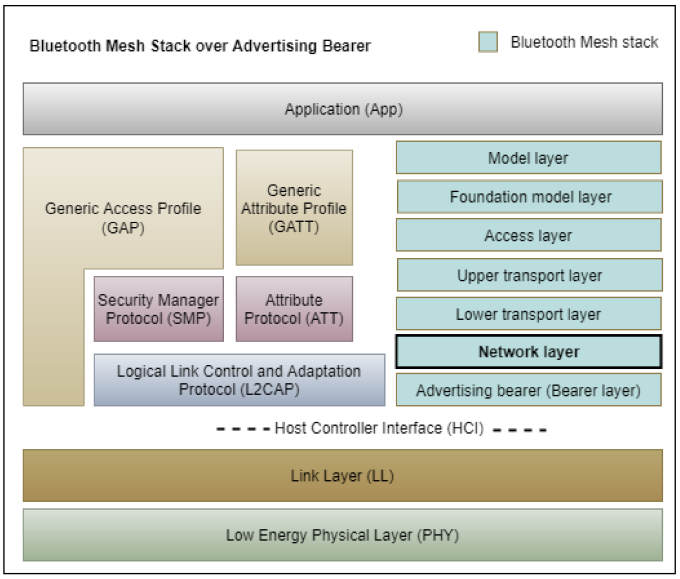


Figura 3.18: Stack protocollare Bluetooth LE and Bluetooth mesh. **Fonte: Mathworks, Energy Profiling of Bluetooth Mesh Nodes in Wireless Sensor Networks**

Livello di rete

Il livello di rete definisce i tipi di indirizzo e il formato dei messaggi di rete che consente ai PDU del livello di trasporto di essere iniviati dal livello *bearer*. Può essere impostato un filtro per i messaggi in entrata dal *bearer* per determinare se devono essere recapitati o meno al livello di rete per ulteriori elaborazioni. Inoltre sia la funzionalità *relay* che *proxy* dei dispositivi può essere implementate in questo livello, come verrà descritto nel prossimo paragrafo.

Livello di trasporto inferiore

Il livello di trasporto inferiore si occupa di inviare le PDU dal livello di trasporto superiore a quello inferiore. Se necessario, esegue anche la segmentazione e il riassemblaggio dei PDU per pacchetti più lunghi, che non rientrano in una singola PDU di trasporto: il livello di trasporto inferiore eseguirà la segmentazione, suddividendo la PDU in più PDU di trasporto. Il livello di trasporto inferiore del dispositivo ricevente riassemblerà i segmenti in una singola PDU per il livello di trasporto superiore.

Livello di trasporto superiore

Livello di trasporto superiore è responsabile di alcune operazioni sia sui dati in arrivo che su quelli destinati al livello di accesso:

- Crittografia.
- Decrittazione.
- Autenticazione.
- Messaggi di controllo, generati e processati a livello di trasporto superiore, in relazione a operazioni della rete mesh (*Heartbeat*[13], *friendship*[14], ecc.)

Livello di accesso

Il livello di accesso definisce come le applicazioni possono utilizzare il livello di trasporto superiore, gestendo le seguenti attività:

- Definizione del formato dei dati dell'applicazione.
- Definizione e controllo del processo di crittografia e decrittografia che viene eseguito nel livello di trasporto superiore.
- Verifica che i dati ricevuti dal livello di trasporto superiore siano destinate alla rete e all'applicazione corretta, prima di inoltrare questi nei livelli superiori dello stack.

Livello fondazione del modello

Il Foundation model layer si occupa della configurazione della rete e i modelli di gestione della rete.

Model layer

Il livello fondazione del modello si occupa dell'implementazione vera e propria dei modelli, descritti nella precedente sezione: comportamenti, messaggi, stati e associazioni di stati.

[13] Lo scopo del messaggio *Heartbeat* è quello di indicare ad altri nodi che il nodo che invia il messaggio *Heartbeat* è ancora attivo e di consentirne la determinazione della distanza dal destinatario, in termini di numero di *hop* necessari per recapitare il messaggio *Heartbeat*.

[14] Una richiesta di amicizia, *friendship*, è un tipo di messaggio di controllo che viene inviato da un nodo a basso consumo per avviare la ricerca di un amico, *Friend*.

3.4.2 Concetti e termini generali

In questa sezione verranno introdotti alcuni concetti e terminologie di base che caratterizzano una rete mesh Bluetooth [26], ma estranei al BLE: in particolare, la descrizione che seguirà, sarà propedeutica per inquadrare nello specifico l'attività di ricerca svolta descritta nel prossimo capitolo.

Nodi ed elementi

Possiamo inizialmente fare una distinzione tra i dispositivi che fanno parte di una rete Bluetooth mesh, denominati ***nodi***, da quelli che presentano tutte le caratteristiche per essere implementati secondo lo *standard* ma non sono stati ancora associati alla rete Bluetooth mesh, denominati ***nodi non provisionati***. Un nodo non provisionato diventa un nodo attraverso un particolare procedura denominata ***provisioning*** ad opera di un particolare nodo o di un dispositivo esterno.

Con il termine nodo si indica in genere un dispositivo che fa parte della mesh, senza entrare nel dettaglio delle sue caratteristiche e funzionalità interne. Ogni nodo è costituito da una o più entità indipendenti, chiamate elementi. Un esempio è costituito da una lampada (nodo) con diverse lampadine (**elementi**) che possono essere controllati separatamente.

Stati

È facile intuire che gli elementi necessitano di un indicatore della loro condizione, ad esempio accensione/spegnimento della lampadina, denominati **stati** nello *standard*. Uno stato è un valore contenuto in un elemento. Ad esempio, considerando una luce che può essere accesa o spenta, il Bluetooth mesh definisce uno stato chiamato *Generic On/Off*: un valore di stato On riflette il comportamento dell'accensione di una luce, mentre un valore di stato Off il suo spegnimento.

Ogni elemento è indirizzabile attraverso indirizzi ***unicast*** per identificare, in modo univoco, l'elemento che invia e quello che riceve un messaggio all'interno di un nodo. Gli indirizzi *unicast* vengono assegnati ai dispositivi durante il processo di *provisioning*. Lo *standard* definisce altri due tipi di indirizzi: di ***gruppo*** e ***virtuali***; entrambi sono indirizzi *multicast*, che etichettano uno o più elementi all'interno di uno o più nodi.

Indirizzi

Un indirizzo di gruppo è un indirizzo *multicast* che rappresenta uno o più elementi. La differenza tra i due tipi risiede nel numero di indirizzi che possono essere assegnati. Nei gruppi ci sono 256 indirizzi fissi definiti dal Bluetooth SIG

e sono conosciuti come SIG *Fixed Group Addresses.* e dinamici stabiliti dall'utente tramite un'applicazione di configurazione. Questi ultimi rifletteranno la configurazione fisica di una rete: sono 16128 gli indirizzi di gruppo assegnati dinamicamente. Un indirizzo virtuale è un indirizzo che può essere assegnato a uno o più elementi che si estende su uno o più nodi. Un indirizzo virtuale è un UUID a 128 bit a cui è possibile associare qualsiasi elemento ed è molto simile a un'etichetta. Gli indirizzi virtuali sono preconfigurati dal produttore del dispositivo. Lo *standard* consente di assegnare dinamicamente indirizzi sia di gruppo che virtuali per adattare la rete alla struttura fisica in cui è installato.

Pubblicazione e Sottoscrizione

Gli elementi interagiscono tra loro per scambiarsi informazioni. Questa interazione è resa possibile grazie all'uso degli indirizzi, sopra descritti, unitamente all'introduzione di un paradigma di ***publishing-subscribing***. In questo schema, il mittente (***publisher***) pubblica semplicemente il suo messaggio nella rete senza essere a conoscenza dell'identità del destinatario (***subscriber***). Al *subscriber* è permesso interpretare le informazioni solo se è un nodo della rete. Quindi, il *subscriber*, attraverso il meccanismo di *subscribing*, può specificare con la massima precisione possibile a quali messaggi iscriversi. Nonostante questo, è difficile conoscere l'origine del *publisher* o il numero di nodi verso i quali esso stesso pubblica, quindi se sono indirizzi di gruppo o virtuali.

Messaggi

Quando un nodo deve interrogare lo stato di altri nodi o deve controllare in qualche modo il loro stato, invia un messaggio: la struttura dei dati utilizzata per invocare queste operazioni tra i nodi sono chiamati ***messaggi***. I messaggi sono una delle caratteristiche principali di questa rete, definita appunto ad inizio di questo capitolo *message-oriented*. All'interno dello *standard*, possiamo identificare tre tipi di messaggi:

- *Get*: per richiedere lo stato a uno o più nodi.
- *Set*: per modificare il valore di stato di un nodo o di un gruppo di nodi. I messaggi di *Set* possono essere con *acknowledgment* o *uncknowledged*.
- *Status*: per rispondere a una richiesta di tipo *Get*, in risposta a un messaggio di *Set* con *acknowledgment* o viene utilizzato da qualsiasi elemento per indicarne lo stato in modo indipendente; un esempio è un sensore che comunica periodicamente il suo valore di temperatura.

Modelli

I ***modelli*** riuniscono in sè tutti i concetti della mesh precedentemente descritti. I modelli definiscono ed implementano solo alcune o tutte le funzionalità di un elemento [43]. Ogni elemento all'interno di un nodo deve supportare un o più modelli che possono essere classificati in tre macro-categorie:

- Modello *server*: definisce i messaggi che il modello può trasmettere e ricevere, gli stati in cui un elemento può trovarsi e i comportamenti (cambiamenti di stato) che sono attivati dalla ricezione di messaggi appropriati.
- Modello *client*: non definisce alcuno stato. Al contrario, definisce i messaggi che può inviare o ricevere per fare la *Get*, *Set* o acquisire lo *Status* degli stati definiti nel modello *server* corrispondente.
- Modello di controllo: contengono sia un modello *server*, che consente la comunicazione con altri modelli *client*, sia un modello *client* che consente la comunicazione con i modelli *server*. I modelli possono essere creati estendendo altri modelli [26].

Per caratterizzare ciascuna delle tre macro-categorie, Bluetooth SIG ha fornito un tipo di modello ben definito: *Fundamental*, *Generic*, *Sensor*, *Lighting* e *Vendor-Specific* Model; questi modelli sono ulteriormente suddivisi in base alla destinazione d'uso. Qualsiasi dispositivo alimentato presenta uno stato di On/Off, (e.g. un ventilatore, un condizionatore d'aria, una luce o un presa di corrente ecc). Lo *standard* per evitare di caratterizzare ciascuno dispositivo con il suo modello On/Off, ha generalizzato il concetto creando un modello Generic On/Off con stati e messaggi che possono adattarsi a qualsiasi esigenza. All'interno della categoria Generic ci sono anche altri modelli basati sullo stesso principio di Level, Transition, Location, Power, ecc.. [44] e il termine Generic accanto a questi termini indica appunto la generalità d'uso.

Generic On/Off

Nel seguito verrà descritta la struttura del pacchetto del modello *Generic* On/Off, in particolare quella dello Status. Tale descrizione servirà per introdurre alcuni termini e concetti a cui si farà spesso riferimento nel prossimo capitolo, focalizzato sui test effettuati proprio su questo tipo di pacchetti.

In tutta la rete a più livelli dell'architettura, inclusa quella di Bluetooth Mesh *standard*, dati aggiuntivi vengono incapsulati e decapsulati in ogni strato. Per il *Generic* On/Off Status si può dedurre quanto segue dai campi del pacchetto nella PDU a livello di trasporto superiore [43]): *Present* On/Off: stato in cui si trovava il LED prima di ricevere il comando *Set*; *Target* On/Off: stato in

cui era il LED dopo l'esecuzione del messaggio Generic On/Off Set; Tempo rimanente: il tempo richiesto dal momento in cui viene inviato il messaggio al termine della transizione verso lo stato *Target*.

Caratteristiche dei nodi

In una rete Bluetooth mesh tutti i nodi possono ricevere e inviare messaggi: tuttavia, alcuni nodi hanno delle caratteristiche particolari che gli permettono di avere capacità speciali.

- *Relay*: i messaggi vengono inviati da un *publisher* agli altri nodi che rientrano nel suo range di copertura. Se i nodi da raggiungere sono fuori dalla portata del *publisher*, un "ripetitore" di messaggi è necessario per consentire loro di arrivare a destinazione. Un nodo *relay* può assumere questo ruolo: può ricevere e ritrasmettere un messaggio trasmesso in *broadcasting* da altri nodi all'interno della rete [45]. Questo tipo di nodi sono essenziali per estendere il range di copertura della rete permettendo la comunicazione *multi-hop*.

- *Proxy*: i nodi della rete che supportano le funzionalità *proxy*, noti come nodi *proxy*, sono in grado di inviare e ricevere messaggi tra (GATT) e *advertising bearer*. In questo modo qualsiasi dispositivo con tecnologia Bluetooth (superiore a 4.0) che non supporta *advertising bearer*, come ad esempio smartphone o tablet, può interfacciarsi con la rete e comunicare con essa.

- *Low Power and Friend*: queste due funzionalità sono specifiche di due tipi di nodi, *low power node* (LPN) e *friend node* (FN), che operano nella rete ad un duty cycle ridotto. La relazione tra LPN e FN si chiama *friendship* [26].

- *Provisioner*: questa funzionalità è prerogativa di tutte le periferiche che, attraverso il processo di *provisioning*, possono aggiungere un dispositivo alla rete mesh. Durante il *provisioning*, il nodo *provisioner* fornisce al nodo *unprovisioned* tutte le informazioni e i parametri necessari per interagire con la rete. Il ruolo di *provisioner* può essere svolto da un dispositivo all'interno della rete Bluetooth Mesh che implementa l'intero stack di protocollare dello *standard* [46]. Lo stesso ruolo può essere assolto anche da un dispositivo esterno alla rete, come ad esempio uno smartphone o un tablet: attraverso l'uso della funzione *proxy*. In quest'ultimo caso, l'uso di un dispositivo mobile rende sicuramente più agevole e versatile la configurazione della rete.

Tecnica del flooding

Come abbiamo visto nella sottosezione 3.2.4, un nodo riceve e ritrasmette messagi del tipo mesh attraverso l'*advertising bearer*, utilizzando dei nodi con funzionalità *relay*, per estendere la rete. Questo tipo di processo è chiamato *managed flooding*. In questa configurazione della rete, un nodo *relay* trasmette un messaggio a tutti i nodi nel suo range, ma con un paio di accorgimenti, basati sull'utilizzo:

- Time To Live (TTL): un valore assegnato che viene decrementato ad ogni ritrasmissione. In questo modo un messaggio viene ritrasmesso solo quando il suo TTL è maggiore di 1.
- Cache: si riferisce ad un messaggio ricevuto e quindi già presente nella memoria di un dispositivo: lo stesso messaggio, se nuovamente ricevuto, sarà automaticamente scartato.

Capitolo 4

Analisi e ottimizzazione della ricezione dei messaggi di Status

4.1 Introduzione

Nel presente capitolo verrà proposta una tecnica per migliorare l'affidabilità nella ricezione dei messaggi di Status in una rete Bluetooth mesh, [47]. Il significato e la novità del lavoro è quello di proporre una configurazione ottimizzata che raggiunga una significativa riduzione della perdita di pacchetti. La corretta ricezione dei messaggi di Status, intesi come la conferma della ricezione dei messaggi scambiati nella rete, in linea di principio è utile in diverse applicazioni, fornendo informazioni utili sull'operatività della rete. La raccolta e l'analisi statistica dei messaggi ricevuti fornisce quindi una stima sull'affidabilità complessiva della rete e della corretta consegna dei messaggi nel tempo. D'altra parte, il loro utilizzo, come riportato nelle specifiche Bluetooth mesh [26], potrebbe generare problemi imprevisti quando più nodi rispondono contemporaneamente, riducendo così la probabilità della loro consegna a causa del verificarsi di collisioni sia in una particolare configurazione della topologia di rete sia quando il numero di nodi nella rete aumenta.

Ad oggi, una delle sfide più grandi proviene dal settore dell'illuminazione, sia commerciale che industriale. A tal riguardo, in questo lavoro sono stati eseguiti dei test su un gruppo di dispositivi che riceve contemporaneamente un comando di *Set* On/Off. Questi dispositivi risponderanno con un messaggio di conferma (Status) indicando che il comando è stato ricevuto. Come descritto nella sottosezione 3.4.2, qualsiasi dispositivo alimentato presenta uno stato di On/Off. Motivo in più per cui il tipo di analisi e i test condotti nel seguito possono essere estesi facilmente anche ad altri dispositivi. I messaggi di Status riferiti allo stato di attivazione/disattivazione (On/Off), sono il fulcro del seguente studio. Essendo la tecnologia Bluetooth mesh relativamente recente, non molti studi specifici sono stati ancora affrontati per valutare le prestazioni in uno scenario applicativo reale, che mostra limiti e va oltre le considerazioni generali attualmente fornite dalla letteratura sui comportamenti della rete

mesh. Ad oggi, la maggior parte degli studi disponibili sono basati sull'analisi delle prestazioni in termini di *delay* della rete e di *Packet Loss Rate* (PLR) (o *Packet Success Rate* (PSR)), al variare dei parametri e delle caratteristiche proprie della rete in ambienti simulati [45]. Altri propongono alcune scelte da adottare per migliorare lo *standard* a fronte di limiti implementativi a livello di *chipset* di alcuni dispositivi, valutati durante l'implementazione e l'esecuzione in dispositivi reali [43].

Nel seguente capitolo verrà quindi condotta una approfondita analisi sulla ricezione degli Status nella rete Bluetooth mesh implementata, basata su un'implementazione *firmware* standard su dispositivi reali. In particolare, verranno condotti due esperimenti in due scenari diversi: il primo è stato realizzato in uno scenario controllato con limitati elementi di interferenza. Il secondo invece è stata realizzato in un ambiente più sfidante, come un tipico scenario reale. Per entrambi gli scenari verrà proposta un'ottimizzazione *firmware*, con una significativa riduzione dei pacchetti persi. Infine, verrà proposta un'analisi generale delle performance della rete, migliorata da un'analisi offline basata sui risultati ottenuti nel setup sperimentale proposto.

4.2 Il problema delle collisioni

Bluetooth mesh funziona tramite il meccanismo del *managed flooding*, tecnica descritta nel paragrafo 3.4.2, in cui ogni dispositivo trasmette sia i propri dati che quelli appartenenti ad altri dispositivi, utilizzando la modalità *advertising*. La gestione e l'invio dei relativi pacchetti di *advertising* segue le regole di temporizzazione imposte dal protocollo BLE, con i meccanismi già descritti nel capitolo 3.2.4.

La stessa PDU (39 bytes) viene trasmessa in sequenza sui tre canali di *advertising* primari (ADV), in un time slot definito *Advertising Event*. La PDU viene quindi ricevuta da ciascun dispositivo nel range radio sul canale ADV corrispondente durante lo *Scanning Event* corrispondente che ha origine dopo un *time slot* definito *ScanInterval*. Inoltre, il parametro *scan window* identifica la durata di ogni scansione su ciascun canale ADV. Da notare che la scansione passiva dovrebbe in questo caso essere impostata con un *duty cycle* il più vicino possibile al 100%, per evitare di perdere alcune PDU mesh in arrivo. Quindi, l'inizio di due *Advertising Event* sono definiti da *AdvInterval* ($\geq$20 ms) più il valore aggiunto del ritardo casuale (*advDelay*) compreso tra 0 e 10 ms. Questo comportamento rispetta le specifiche BLE ed è controllato dal livello di collegamento. Inoltre, per abbassare la probabilità di collisioni tra i pacchetti su tutti i canali di *advertising*, lo stesso *standard* raccomanda di inserire un gap tra l'inizio di due PDU consecutive all'interno dello stesso *Advertising Event* (T_{ChPDU}), che deve essere random e inferiore a 10 ms (Figura 4.1).

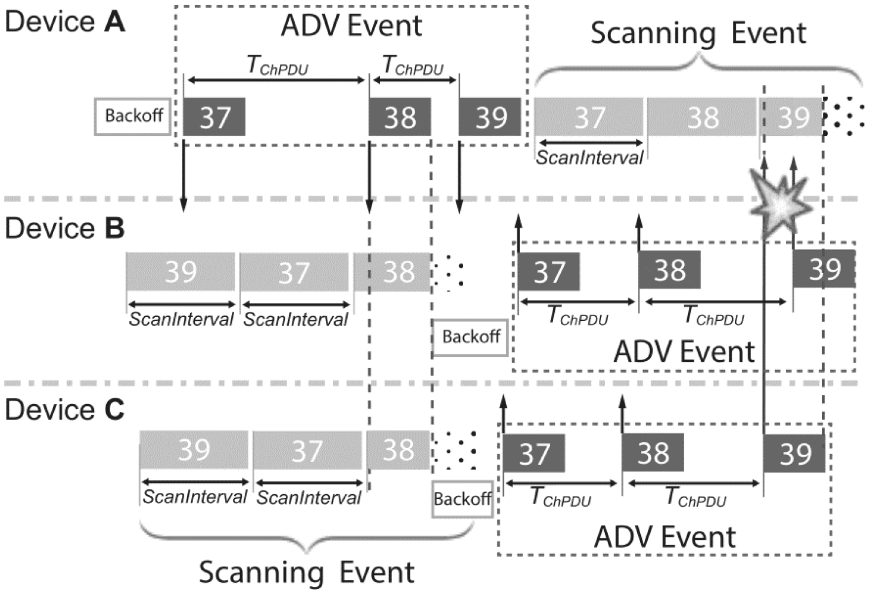


Figura 4.1: Esempio del problema delle collisioni durante il flusso di comunicazione tra tre dispositivi in una rete Bluetooth mesh.

4.2.1 Parametri della rete

L'efficienza e l'affidabilità di una rete mesh dipendono da diversi fattori: la randomizzazione su e all'interno degli *Advertising Event*, discussi nella sezione precedente; il controllo delle ripetizioni dei messaggi ridondanti, per assicurarne la ricezione, attraverso l'aggiustamento di più parametri di configurazione. Una errata scelta di questi valori può contribuire ad accrescere le collisioni tra pacchetti durante le fasi del flusso di comunicazione.

La rete Bluetooth mesh utilizza due tipi principali di pacchetti di *advertising*: *network* e *relay*. *Network transmit count* e *Network transmit interval steps* (*Nris*) a livello di rete si riferiscono, rispettivamente, al numero delle ripetizioni di una PDU generata da un nodo e lo spazio tra queste ripetizioni, con un intervallo di trasmissione uguale a (*Nris* + 1) 10 ms. Come per *Network transmit*, *Relay transmit count* e i *Relay transmit interval steps* (*Rris*) sono parametri a livello di rete, legati alle funzionalità *relay*, utilizzati quando il messaggio proviene da un altro nodo e deve essere ritrasmesso. L'intervallo di trasmissione è pari a (*Rris* + 1) 10 ms.

Come accennato nella sezione precedente, il livello di collegamento impone l'introduzione di un valore di ritardo casuale da 0 a 10 ms tra ogni trasmissione, anche per quelle gestite da Nris e Rris. Da un lato, quando attivata, la funzionalità *relay* è particolarmente utile nel momento in cui il numero di nodi che in contemporanea ritrasmettono lo Status aumentano nella rete. La

funzionalità *relay* aumenta i possibili percorsi per la PDU e la possibilità che il pacchetto corrispondente venga trasmesso attraverso la rete mesh, aumentando le prestazioni complessive di affidabilità. D'altra parte, le ritrasmissioni consecutive potrebbero aumentare le collisioni, quindi generare la congestione della rete. In quest'ultimo caso, i messaggi di Status inoltrati, che si trovano nel mezzo, possono essere persi prima di arrivare a destinazione o non essere ricevuti correttamente, a causa delle collisioni o l'indisponibilità del destinatario nello scansionare i canali ADV. Quest'ultimo caso è associato al fenomeno denominato *blind time* [43], relativo alla decodifica o alla funzione *switch* della frequenza, che potrebbe causare una scansione non continua, portando a un tasso di perdita di PDU più elevato del previsto. A questo proposito, le specifiche mesh suggeriscono di includere un valore casuale, detto *backoff*, prima dell'inizio di ciascun evento di *advertising*. In particolare, per tutte quelle PDU di rete ricevute contemporaneamente dai nodi con funzionalità *relay* è necessario applicare un piccolo ritardo casuale prima di reinoltrarle, per evitare le collisioni.

4.2.2 Randomizzazione dei messaggi di Status

A volte l'intervallo di tempo tra le diverse PDU non è sufficiente a garantire la corretta ricezione di tutti messaggi. In particolare, ciò si verifica quando vengono inviati messaggi quasi simultaneamente, come mostrato in Figura 4.1. La Figura mostra un esempio semplificato e generale del caso di cui sopra. I dispositivi B e C inviano due messaggi, che vengono ricevuti quasi contemporaneamente dal dispositivo A sul canale 39 con il verificarsi della collisione dei messaggi inviati. A questo proposito, le specifiche mesh suggeriscono che è necessario applicare un ulteriore ritardo per un messaggio di Status inizialmente inviato in risposta a un messaggio di *acknowledgment* (ACK) ricevuto con un indirizzo unicast: il nodo dovrebbe trasmettere lo Status con un ritardo casuale ($Rand_{ACK}$) tra 20 e 50 millisecondi. Inoltre, nel caso in cui lo Status venga inviato in risposta a un messaggio ricevuto, quest'ultimo inviato con un indirizzo di gruppo o un indirizzo virtuale, il nodo dovrebbe trasmettere lo Status con un ritardo casuale ($Rand_{ACK}$) tra 20 e 500 millisecondi. Ciò riduce la probabilità di avere più nodi che rispondono a questo messaggio contemporaneamente e quindi aumenta la probabilità di recapito del messaggio piuttosto che avere collisioni di messaggi [26].

4.3 Setup sperimentale

Nel setup proposto, alcune schede Nordic nRF52832 sono state adottate per implementare la rete di nodi BLE mesh, che supportano lo *standard* Bluetooth

mesh [48]. Partendo dal Kit di Sviluppo Software della Nordic (SDK), è stato implementato nelle schede nRF52832 DK il *firmware Light Switch Server* per 10 dispositivi, e *Light Switch Client* per 1 dispositivo. Nel seguito, ogni tipo di nodo sarà semplicemente indicato come i termini *client* e *server*. Il *server* consiste in una scheda con ruolo di semplice *server* implementante una istanza denominata *Generic OnOff Server Model*. Questa istanza in generale identifica il comportamento di una lampada che deve effettuare lo *switch* On/off. A tal proposito, nel setup preso in esame, è stato assimilata una lampada ad un LED integrato nelle schede nRF52 DK. Il *client* è una scheda con ruolo di un semplice *client* che implementa una istanza denominata *Generic OnOff Client Model* ad identificare uno *switch* che controlla tutte le lampade. Il *client* è stato connesso a un PC e, tramite l'utilizzo del *software J-link Real-Time Transfer* (RTT) *Logger*[1] sono stati acquisiti i dati dalla scheda nRF52832 archiviandoli in un file di log.

Per creare la rete Bluetooth mesh è stato utilizzato uno smartphone e l'App Android nRF Mesh. L'App ha permesso di gestire il processo di *provisioning* per configurare tutti i nodi, sia il *server* che il *client*, consentendo loro di operare e scambiare messaggi, utilizzando le funzionalità *proxy*.

Una volta eseguito il *provisioning*, ogni nodo può essere configurato, a seconda del comportamento che deve avere all'interno della rete mesh: oltre a fornire le informazioni essenziali per associare i dispositivi alla rete è possibile assegnare indirizzi di pubblicazione e sottoscrizione tramite l'App. Per ottenere una comunicazione 1:M tra *client* e *server*, è stato assegnato un indirizzo di pubblicazione di gruppo al *Generic OnOff Client model*. Lo stesso indirizzo è stato quindi impostato come indirizzo di sottoscrizione nel *Generic OnOff Server model* implementato in ogni *server*. In questo modo ogni volta che viene inviato un messaggio di *Set* con questo indirizzo tutti i *server* ricevono contemporaneamente, o quasi, la stessa richiesta. Per quanto riguarda i messaggi di Status diretti dai *server* al *client*, nonostante la trasmissione si possa schematizzare con una struttura M:1, la risposta di ogni *server* avviene sempre tramite l'utilizzo di un indirizzo Unicast.

La topologia di configurazione e i parametri di temporizzazione scelti per la rete mesh creata nel setup proposto sono riassunti nella Tabella 4.1. La definizione di alcuni di questi parametri è stata già discussa nella sottosezione 4.2.1, riguardanti la topologia della rete e la dimensione della PDU. E' stata quindi impostata la potenza di trasmissione a 0 dBm, per tutti i nodi, più che sufficienti per il tipo di test effettuato e impostato il TTL, già discusso nella sottosezione 3.4.2, uguale a 6. Tenendo presente alcune fondamentali

[1] Il software J-link RTT Logger fa parte di pacchetto software di base per J-Link. Sviluppato da SEGGER Microcontroller, consente la comunicazione bidirezionale tra la scheda di destinazione e il PC. La tecnologia si basa sulla capacità di J-Link di leggere i dati nella memoria del microcontrollore durante l'esecuzione del codice.

Tabella 4.1: Parametri di configurazione

Paramenter	**Exp 1**	**Exp 2**
Number of devices	11	
Servers SΔ (m)	0.40	2-6
Client-servers CSΔ (m)	3-3.6	-
Tx Power	0 dBm	
PDU size	39 bytes	
TTL	6	
Network retransmit interval steps	5	
Network retransmit count	3	
Relay retransmit interval steps	5	
Relay retransmit count	3	
AdvInterval (ms)	20	

considerazioni emerse in [45], abbiamo adottato sia per il *client* che i *server* *Network retransmit interval steps* = 3, e *Network retransmit count* = 5: questo per garantire la ricezione del comando di On/Off, quindi, aumentare il PSR in generale. Inoltre, in questo caso specifico, il ritardo massimo durante l'uso delle repliche delle ritrasmissioni rappresenta un compromesso accettabile per il miglioramento dell'affidabilità. La funzionalità *relay* è stata lasciata attiva sia nei *client* che nei *server*, con *Relay retransmit interval steps* = 3, e *Relay retransmit count* = 5, consapevoli della possibilità di aumento dei pacchetti persi e quindi delle collisioni. La scelta si è rivelata necessaria per compensare la perdita dei messaggi di Status reinoltrati quasi contemporaneamente dai *server*. Si noti che la scelta dei valori dei parametri fin qui descritti sono il risultato di decisioni prese sui test preliminari effettuati, per ottimizzare le prestazioni della rete. Infine, è cruciale considerare la relazione tra la durata di un *Advertising Event* ed uno *Scanning Event* per garantire il successo nella ricezione dei messaggi ed la corretta corrispondenza tra il canale attraverso il quale viene trasmessa la PDU e il suo omologo durante la scansione in ricezione. Questo elemento è altamente rilevante dal momento che ci sono solo tre canali di *advertising*.

Nel setup sperimentale proposto, il comportamento dei dispositivi può essere sintetizzato come segue:

1. Il *client* invia un comando *Generic OnOff Set* con *acknowledgment*: ad esempio, richiede che i LEDs di ciascuno *server* vengano accesi. Un tale comando può assumere solo due valori: 1 per On e 0 per Off.

2. I *server* ricevono il comando, lo elaborano modificando lo stato dei LEDs, e infine inviano il loro messaggio *Generic OnOff Status* al *client*. Allo

stesso tempo, i nodi con funzionalità *relay* attiva in ogni nodo del *server*, si occupano anche di ritrasmettere i messaggi ricevuti dagli altri nodi e dal *client* stesso.

3. I molteplici messaggi di Status raggiungono il *client*.

4. Il software *J-link RTT Logger* preleva i dati dal *client* e li memorizza nel file di log.

La sequenza di comandi generata dal *client* presenta un'alternanza di comandi di *Set* On/Off, intervallati da 10 s. Quest'ultima alternanza è gestita da un timer appositamente creato per i nostri test in modo che i comandi vengano eseguiti in modo automatico nel tempo. Il comportamento del *client* consente di ripetere ciclicamente quanto descritto dal punto 1 al punto 4 per l'intera durata, circa 15 ore, di ciascuna prova per un totale di 5470 comandi di tipo *acknowledged* dal *client* ai 10 nodi *server*.

Alla fine di ogni sequenza di comandi, il file di log è stato sottoposto ad una prima ed attenta revisione per trovare possibili errori che potessero invalidare le prove. Infine, comparando questa prova ad uno scenario d'uso quotidiano, ad esempio all'interno di un locale commerciale ipotizzando una media di 6 comandi al giorno tra On e Off, il valore simulato di 5470 comandi corrisponderebbe a circa 911 giorni o equivalenti a 2.5 anni. Durante un trial il consumo corrente medio misurato sul client è di circa 5.52 mAh, mentre ogni server è di circa 2.08 mAh. La sequenza di comandi generati dal *client* e i relativi messaggi di conferma dei *server* richiedono un consumo energetico di rete di circa 86.86 mWh. Tali misurazioni del consumo energetico sono state eseguite utilizzando Keysight N6705B DC Power Analyzer. Nel trial proposto ogni messaggio di conferma perso nella rete richiede ritrasmissioni di comandi e risposte che comportano un fabbisogno energetico di almeno 0.24 mW. Quindi anche solo una riduzione dell'1% del tasso complessivo di successo dei pacchetti dei messaggi di conferma porta ad un aumento del consumo energetico di circa 13.13 mW. Data l'importanza di analizzare i pacchetti persi nella rete Bluetooth mesh, vengono valutate le prestazioni di rete in base ai risultati ottenuti nei test costituiti da 10 ripetizioni del *trial*. Nella configurazione sperimentale proposta, il *trial* è stato ripetuto 10 volte per avere una quantità statisticamente sufficiente di dati per analizzare i risultati ottenuti. I test sono stati condotti in due differenti ambienti e condizioni in cui opera la rete Bluetooth mesh creata. In particolare, abbiamo condotto i test in uno scenario controllato e in uno scenario reale. Lo scenario controllato è stato disposto all'interno di un laboratorio con elementi interferenti limitati, come in Figura 4.2.

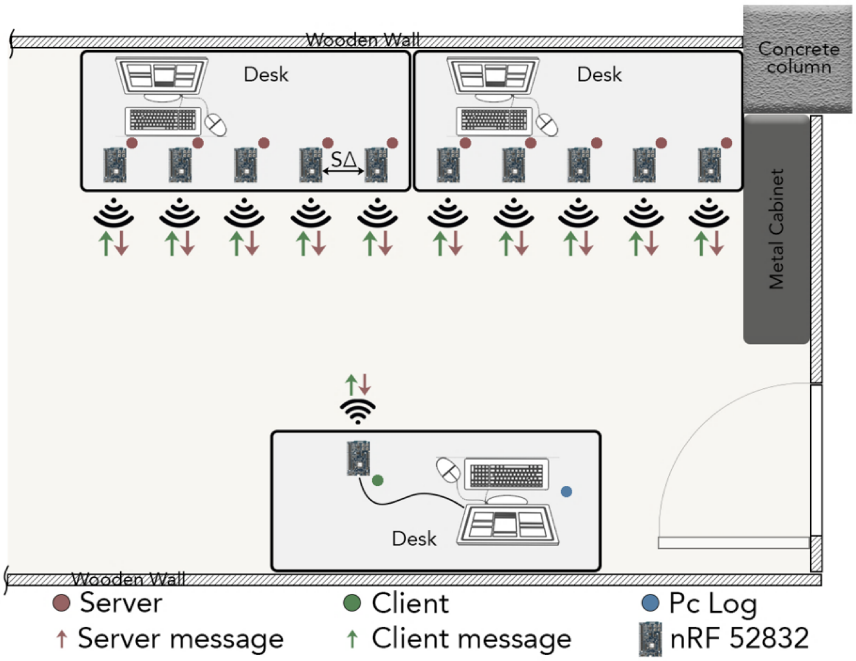


Figura 4.2: Implementazione della rete Bluetooth mesh nello scenario controllato.

4.3.1 Laboratorio

Per quanto riguarda il primo *trial*, una volta verificato che tutti i nodi comunicassero tra loro, sono stati disposti all'interno della stanza, come mostrato in Figura 4.2. Il *client* e PC sono stati posizionati su un banco del laboratorio e, su un altro banco, ad una distanza tra 3 - 3.6 m, sono stati disposti i 10 *server* distanziati di circa 40 cm l'uno dall'altro. Per come è stata settata la potenza di trasmissione, tutti i nodi della rete sono in completa visibilità. Una tale configurazione non è sicuramente la più favorevole per lo scambio dei messaggi all'interno della rete mesh: infatti, grazie alla funzionalità *relay* attiva in ogni nodo, il traffico dei messaggi che la rete deve gestire è molto alto, generando collisioni e perdite di pacchetti. D'altra parte è anche vero che non è raro trovare un simile scenario in ambito commerciale, piuttosto che in ambienti come case, industrie, sale espositive, ecc. Oltretutto, questo tipo di configurazione della rete rappresenta uno degli scenari peggiori dello *smart lighting* in termini di comunicazione tra i dispositivi: il *client* si trova a gestire più Status ricevuti nei suoi canali di *scanning*, temporalmente distanziati solo da piccoli ritardi, nella maggior parte dei casi insufficienti.

Il *trial* nello scenario controllato è stato condotto durante l'orario di chiusura

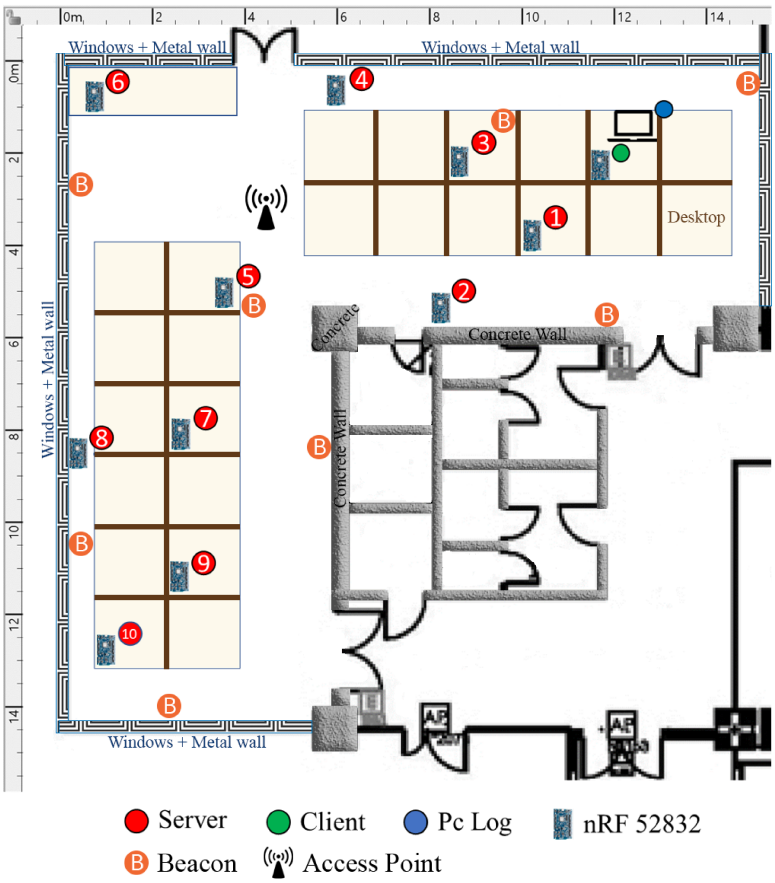


Figura 4.3: Implementazione della rete Bluetooth mesh nello scenario reale.

del laboratorio, in un ambiente senza ostacoli e altri dispositivi, per ridurre al minimo le interferenze imprevedibili e garantire condizioni sperimentali simili per ogni test. Ciò garantisce la riproducibilità del test e impedisce che le misurazioni PLR siano influenzate dalle interferenze co-canale dovute a ostacoli e dispositivi nell'ambiente.

4.3.2 Ufficio

Il trial nello scenario reale è stato condotto in un grande ufficio Figura 4.3 con due blocchi, ognuno composto da 12 scrivanie all'interno. Ogni postazione di lavoro è separata da pareti in legno che si innalzano a circa 70 cm sopra il piano della scrivania. I nodi *server* sono stati disposti in modo tale da avere una distribuzione omogenea in tutta la stanza e ridondanza di percorsi per lo scambio di messaggi. Le distanze tra un nodo e l'altro variano tra 2m e 6m.

Il processo è stato eseguito durante l'orario di ufficio, considerando un arco di tempo di 15 ore. A differenza del primo ambiente, questo presenta numerose sorgenti interferenti e attenuante per i segnali trasmessi dalla rete mesh Bluetooth. Oltre all'attenuazione e ai riflessi dovuti agli elementi strutturali e d'arredo, ci sono anche molteplici disturbi RF. Come mostrato in Figura 4.3 troviamo un punto di accesso WIFI posizionato sul soffitto in posizione centrale e una rete di otto fari Bluetooth sparsi su tutta la stanza. Questi ultimi trasmettono le loro informazioni sugli stessi canali pubblicitari (Ch 37, 38, 39) sfruttati dalla rete mesh Bluetooth. A questi disturbi che possiamo considerare fissi all'interno della stanza, aggiungiamo tutti quelli casuali dovuti alle persone e ai loro dispositivi personali come smartphone, tablet, PC che accedono a questo posto.

4.4 Ottimizzazione della ricezione degli Status

Per valutare le prestazioni della ricezione dei messaggi di conferma in una rete Bluetooth mesh, in questo studio sono stato condotti due differenti tests in ogni scenario sulla base del setup sperimentale illustrato nella sezione precedente. L'unica differenza tra i test consiste in una sostanziale novità nella gestione dei messaggi di Status, introdotti a livello di modello nel *server*.

In particolare, è stato condotto un test con la configurazione *standard* (SC) della rete Bluetooth mesh e un'altra con una configurazione (OC) ottimizzata in termini di tempistica nell'invio dei messaggi di Status. Nel test OC, la modifica apportata nel livello di modello nel *server* consiste principalmente nell'introdurre un ritardo compreso tra 20 e 500 ms prima di inoltrare i messaggi di Status attraverso i livelli sottostanti. Diversamente, nel test SC i *server* non implementano nessun ritardo. Tutti gli altri parametri di configurazione della rete Bluetooth mesh sono rimasti invariati per entrambi i tipi di test. Questa scelta ha reso possibile confrontare direttamente le prestazioni del test OC con quello SC, a partire dalle stesse condizioni iniziali per tutte le altre variabili di configurazione. Pertanto, OC è diverso da SC nell'implementazione di una tecnica di spreading[2] degli Status nel tempo, ma il setup e il comportamento generale dell'intera rete sono identici, come descritto nella sezione precedente.

La tecnica implementata consiste nello *spreading* dei messaggi di Status nel tempo per aumentare la possibilità della loro ricezione e evitare il problema delle collisioni. Il *firmware* OC è stato implementato in ogni server e le principali fasi coinvolte, dalla ricezione di un messaggio di Set alla risposta con l'*advertising* di uno Status, sono mostrate in Figura 4.4.

All'inizio, per ogni comando On/Off inviato dal *client* e ricevuto da ogni *server*, il livello applicazione riceve un nuovo comando con valore di Set On/Off,

[2]Diffusione dei messaggi nel tempo

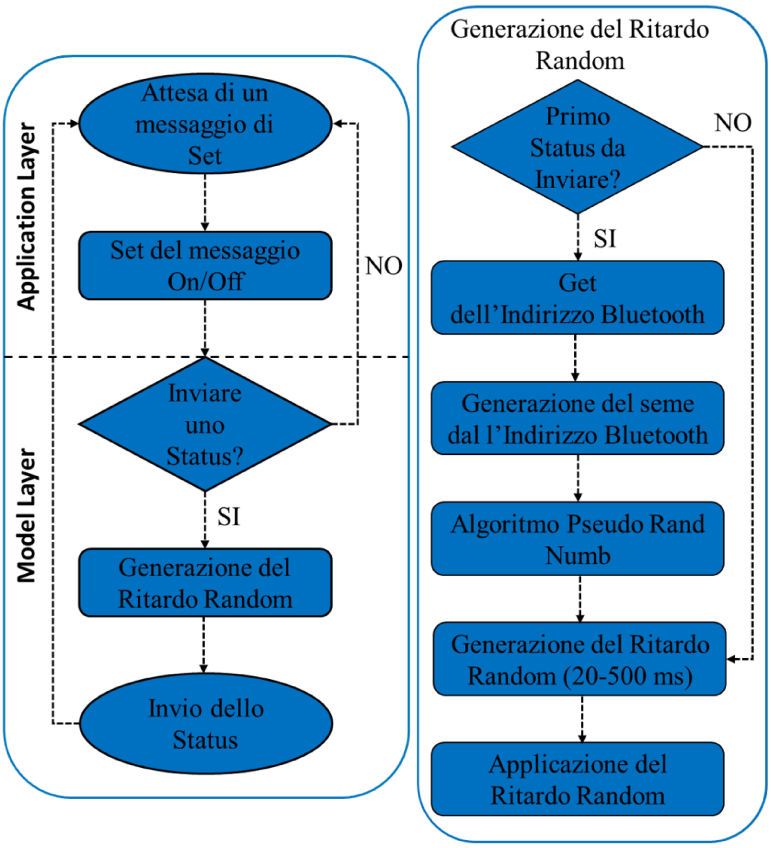


Figura 4.4: Schema a blocchi dell'algoritmo di randomizzazione degli Status.

inserito nella PDU proveniente dal livello di accesso, a sua volta trasmesso dai livelli mesh sottostanti. Quindi, tramite una funzione di *callback* a livello di modello, si è verificato se un messaggio di Status dovesse essere reinoltrato al *client* e pubblicato verso i livelli sottostanti. Il reinoltro avviene dopo aver imposto un tempo di ritardo casuale. Ancora una volta, il processo si ripete per ogni nuovo comando di *Set* ricevuto.

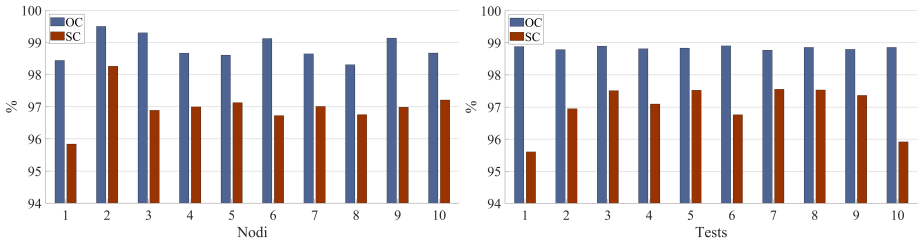
La prima volta che viene ricevuto un messaggio di Status nel livello di modello, si deve essere certi di poter generare un valore di ritardo casuale per tutti i successivi Status che verrano inviati. A tal fine è stato utilizzato un algoritmo di generazione di numeri pseudocasuali, basato sull'indirizzo Bluetooth del dispositivo, scelto come seme. L'indirizzo Bluetooh è un identificatore univoco a 48 bit e assegnato a ciascuno dispositivo Bluetooth dal produttore. L'indirizzo Bluetooh di solito viene visualizzato come sei bytes scritti in esadecimale. Quindi, dopo aver ottenuto l'indirizzo del dispositivo Bluetooth, viene generato un seme, rappresentato come un numero intero, da usare nell'algoritmo per la

generazione di numeri pseudo-casuali. Come seme, sono state scelte le prime quattro cifre dell'indirizzo Bluetooth del dispositivo. Cioè, sono state concatenate le cifre, da destra a sinistra, dal byte meno significativo (LSB) al byte più significativo (MSB). L'intera stringa di quattro bytes generata è stata poi convertita in un seme con valore decimale. Da notare che i passaggi descritti finora sono eseguiti solo la prima volta che ogni nodo riceve uno Status, come una sorta di inizializzazione. Quindi, ogni volta viene generato un nuovo valore di ritardo casuale, tra 20 e 500 ms, per ogni nuovo Status da inoltrare, seguendo le specifiche Bluetooth mesh.

4.5 Risultati

Sia per lo scenario controllato che per quello reale, le prestazioni della rete sono state valutate in termini di Status PSR (e.g. SC e OC test). Dall'analisi delle prestazioni nell'esecuzione dei comandi di On/Off, sono stati considerati un totale di 54700 comandi inviati dai *client* per ogni test, rispettivamente 5470 per ciascuno dei 10 *server* nella rete. Quindi, è stato calcolato il numero totale degli Status non ricevuti per ogni test e il numero di Status ricevuti. Dopodichè è stata calcolata la media dei risultati ottenuti in ogni test per tutti i 10 *trial* ottenuti in ogni scenario. Infine, rispettivamente, per capire se ci siano differenze nelle medie calcolate sui 10 test SC e OC e quantificare quanto sia l'impatto sulla grandezza di questa differenza, sono stati adottati il test t di Student [49] e il *Cohen's d test*.

Per quanto riguarda il *trial* condotto nello scenario controllato effettuato in laboratorio, i risultati dei test SC e OC sono mostrati rispettivamente in Tabella 4.2 e 4.3. Considerando i risultati complessivi per ogni nodo (Tabella 4.2) e quindi mediati per tutti i 10 test, è stato ottenuta, nel peggiore dei casi del test SC, una media del 95.84% dei messaggi di Status ricevuti dal *client* dal nodo uno, mentre il miglior risultato è stato ottenuto dal nodo due, con un 98.26% dei messaggi di Status recapitati correttamente. D'altra parte invece, nel test OC è stata ottenuta una media nel peggiore caso del 98.44% per i messaggi di Status ricevuti dal *client* dal nodo uno, ottenendo il miglior risultato dal nodo due, con il 99.50% dei messaggi di Status recapitati correttamente. Considerando i risultati complessivi ottenuti per ciascuna prova, mediati tra tutti i nodi, il test SC mostra, le prestazioni peggiori, con una media del 95.60% per i messaggi di Status ricevuti dal *client* nel test uno e le migliori ottenute nel test sette, con il 97.55% dei messaggi di Status recapitati correttamente. Inoltre, per il test OC, come previsto, nella peggiore situazione, è stata ottenuta una media del 98.77% per i messaggi di Status ricevuti dal *client* nel test sette, con il miglior risultato ottenuto nel test sei, dove il 98.91% dei messaggi di Status è stato recapitato correttamente. In generale, le prestazioni di ogni nodo sono



(a) (b)

Figura 4.5: **Test nello scenario controllato.** Confronto del packet success rate basato sugli Status, per ogni: (a) nodo, mediato su tutti e 10 i tests, tra il caso in configurazione ottimizzata (OC) e quello in configurazione *standard* (SC); (b) test, mediato su tutti e 10 i nodi, tra il caso in configurazione ottimizzata (OC) e quello in configurazione *standard* (SC).

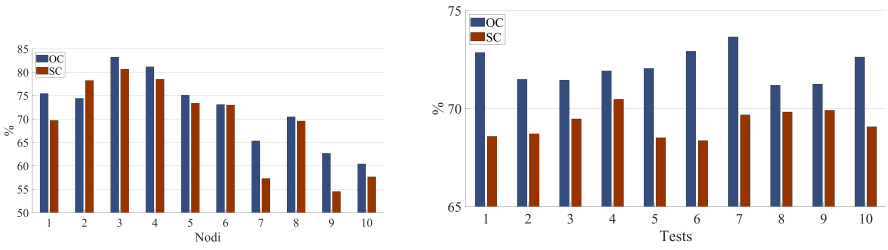
Tabella 4.2: Risultati della percentuale di successo complessiva dei pacchetti per il test di configurazione *standard* (SC) dello scenario controllato.

	NODO										
ID Test	**n1**	**n2**	**n3**	**n4**	**n5**	**n6**	**n7**	**n8**	**n9**	**n10**	**Media Singolo Test**
SC_1	93.03	97.51	96.33	95.90	96.25	94.75	94.73	95.37	95.98	96.16	**95.60**
SC_2	96.07	97.88	96.63	97.68	97.07	96.73	96.42	96.36	96.45	98.17	**96.95**
SC_3	96.56	98.45	97.11	97.48	97.92	97.04	98.01	97.44	97.53	97.57	**97.51**
SC_4	95.70	97.90	97.22	97.48	97.61	96.65	97.61	96.89	96.78	97.09	**97.09**
SC_5	96.01	98.76	96.87	97.75	97.95	97.53	97.86	97.77	97.33	97.39	**97.52**
SC_6	95.78	98.68	96.91	95.72	96.86	96.98	96.91	96.18	97.39	96.20	**96.76**
SC_7	96.75	98.61	96.98	97.31	97.26	97.53	97.44	97.68	97.90	98.08	**97.55**
SC_8	97.35	98.57	97.62	97.15	96.76	97.57	97.57	96.97	97.71	98.03	**97.53**
SC_9	96.67	98.48	97.09	97.42	96.78	97.46	97.70	97.15	97.17	97.70	**97.36**
SC_10	94.46	97.77	96.12	96.07	96.80	95.01	95.85	95.72	95.61	95.72	**95.91**
Media Nodo	**95.84**	**98.26**	**96.89**	**97.00**	**97.13**	**96.73**	**97.01**	**96.75**	**96.99**	**97.21**	

risultate sempre maggiori per il test OC rispetto alla sua controparte SC per tutti i 10 test mediati (Figura 4.5a e Figura 4.5b)). Si possono ottenere risultati simili considerando la media tra tutti i nodi per ogni test.

Come mostrato nella Tabella 4.4, per quanto riguarda OC, il 98.84% degli Status sono stati ricevuti, mentre per il test SC il 96.98%, con una media di 1.86% punti di differenza. Infine, la relativa varianza media calcolata su entrambi i test SC e OC risulta anch'essa un dato particolarmente significativo, rispettivamente dello 0.44% per SC e 0.002% per OC, come presentato in Tabella 4.4. Questo risultato si ottiene sia per quanto riguarda la varianza tra le medie sui singoli nodi sia sui singoli test effettuati.

Per quanto riguarda il *trial* nello scenario reale svolto in ufficio, i risultati



(a) (b)

Figura 4.6: **Test nello scenario reale.** Confronto del packet success rate basato sugli Status, per ogni: (a) nodo, mediato su tutti e 10 i tests, tra il caso in configurazione ottimizzata (OC) e quello in configurazione *standard* (SC); (b) test, mediato su tutti e 10 i nodi, tra il caso in configurazione ottimizzata (OC) e quello in configurazione *standard* (SC).

Tabella 4.3: Risultati della percentuale di successo complessiva dei pacchetti per il test di configurazione ottimizzato (OC) dello scenario controllato.

	NODE										
ID Test	**n1**	**n2**	**n3**	**n4**	**n5**	**n6**	**n7**	**n8**	**n9**	**n10**	**Avg. Single Test**
OC_1	98.56	99.45	99.07	98.85	98.52	99.40	98.81	98.48	99.21	98.50	**98.88**
OC_2	98.32	99.47	98.96	98.56	98.48	99.20	98.78	98.39	99.16	98.57	**98.79**
OC_3	98.45	99.52	99.38	98.81	98.59	99.18	98.68	98.19	99.32	98.83	**98.90**
OC_4	98.63	99.31	99.36	98.63	98.70	99.10	98.48	98.03	99.12	98.78	**98.81**
OC_5	98.45	99.52	99.36	98.63	98.56	99.07	98.59	98.12	99.31	98.76	**98.84**
OC_6	98.23	99.63	99.52	98.76	98.78	98.98	98.63	98.46	99.29	98.79	**98.91**
OC_7	98.28	99.65	99.34	98.39	98.81	99.16	98.37	98.19	98.88	98.61	**98.77**
OC_8	98.56	99.65	99.51	98.72	98.68	98.90	98.74	98.34	98.94	98.52	**98.86**
OC_9	98.46	99.31	99.29	98.68	98.50	99.03	98.67	98.37	98.87	98.79	**98.80**
OC_10	98.50	99.49	99.21	98.63	98.45	99.21	98.72	98.52	99.25	98.57	**98.86**
Node Avg.	**98.44**	**99.50**	**99.30**	**98.67**	**98.61**	**99.12**	**98.65**	**98.31**	**99.14**	**98.67**	

Tabella 4.4: Confronto tra il test in configurazione ottimizzata (OC) e quello in configurazione *standard* (SC) dello scenario controllato.

	Overall Average	**Variance**
SC	96.98	0.445
OC	98.84	0.002

Tabella 4.5: Risultati della percentuale di successo complessiva dei pacchetti per il test di configurazione standard (SC) dello scenario reale.

	NODE										
ID Test	**n1**	**n2**	**n3**	**n4**	**n5**	**n6**	**n7**	**n8**	**n9**	**n10**	**Avg. Single Test**
SC_1	67.41	77.59	80.65	76.94	72.96	72.41	58.52	69.54	54.17	55.65	**68.58**
SC_2	69.00	79.23	80.95	79.53	73,86	70.62	56.94	69.60	51.57	55.83	**68.71**
SC_3	68,98	77.04	79.54	78.33	73.70	70.37	55.93	67.31	60.74	62.78	**69.47**
SC_4	64.17	77.31	80.56	78.61	77.04	70.00	59.07	70.19	63.33	64.44	**70.47**
SC_5	70.37	77.50	78.61	78.15	69.17	74.26	57.69	70.56	52.69	56.11	**68.51**
SC_6	70.37	78.52	81.30	77.22	69.91	74.17	56.48	70.46	50.28	54.91	**68.36**
SC_7	70.19	81.67	83.52	79.35	72.69	74.35	58.80	6815	54.17	59.89	**69.68**
SC_8	74.17	77.50	80.37	80.19	72.69	73.70	57.69	73.15	52.59	56.20	**69.82**
SC_9	71.02	77.59	81.39	77.96	76.48	76.11	57.31	70.93	52.22	58.06	**69.91**
SC_1	71.51	78.45	79.83	78.91	75.39	73.91	54.58	66.05	53.47	58.65	**69.07**
Node Avg.	**69.72**	**78.24**	**80.67**	**78.52**	**73.39**	**72.99**	**57.30**	**69.59**	**54.52**	**57.65**	

Tabella 4.6: Risultati della percentuale di successo complessiva dei pacchetti per il test di configurazione ottimizzato (OC) dello scenario reale.

	NODE										
ID Test	**n1**	**n2**	**n3**	**n4**	**n5**	**n6**	**n7**	**n8**	**n9**	**n10**	**Avg. Single Test**
OC_1	74.17	83.15	79.07	81.76	77.04	74.00	69.35	70.74	56.57	62.78	**72.86**
OC_2	74.26	81.57	78.89	80.19	76.48	72.04	65.37	66.67	56.30	63.15	**71.49**
OC_3	71.57	79.63	82.78	77.31	75.37	75.46	58.33	72.69	63.24	58.15	**71.45**
OC_4	73.70	77.22	83.61	81.39	76.20	74.17	60.74	68.98	64.17	58.98	**71.92**
OC_5	77.96	70.65	82.96	81.20	75.00	71.02	67.31	72.41	61.85	60.09	**72.05**
OC_6	77.69	72.41	84.35	81.67	75.09	70.56	70.09	73.70	63.24	60.37	**72.92**
OC_7	77.22	73.43	84.72	81.20	77.13	73.70	70.00	74.26	65.37	59.44	**73.65**
OC_8	73.80	67.96	84.44	82.31	72.78	73.33	64.07	68.43	64.54	60.28	**71.19**
OC_9	75.83	67.41	84.07	80.56	72.78	74.35	63.33	67.96	64.91	61.30	**71.25**
OC_10	78.33	70.74	87.50	83.98	73.15	72.50	64.81	68.89	66.67	59.54	**72.61**
Node Avg.	**75.45**	**74.42**	**83.25**	**81.16**	**75.11**	**73.11**	**65.34**	**70.47**	**62.69**	**60.41**	

Tabella 4.7: Confronto tra il test in configurazione ottimizzata (OC) e quello in configurazione *standard* (SC) dello scenario reale.

	Overall Average (%)	**Variance (%)**
SC	69.26	0.46
OC	72.14	0.62

dei test SC e OC sono riportati nelle tabelle 4.5 e 4.6. Nel test SC che abbiamo ottenuto, considerando i risultati complessivi per ogni nodo mediando tutti i 10 test, il risultato peggiore e migliore si ottiene, rispettivamente, dal nodo nove con 54.52% e dal nodo tre con il 80.67% dei messaggi di Status correttamente recapitati. Nel test OC i risultati peggiori e migliori sono stati raggiunti, rispettivamente, dal nodo dieci e tre, rispettivamente, con il 60.41% e l'83.25% dei messaggi di Status recapitati correttamente. Considerando i risultati complessivi ottenuti per il test SC, mediando tutti e 10 i nodi, abbiamo ottenuto, rispettivamente, i risultati peggiori e migliori, per il test sei con il 68.36% e il test quattro con il 70.47% dei messaggi di Status correttamente consegnati. D'altra parte, nel test OC, il test otto e il test sette hanno ottenuto prestazioni peggiori e migliori, rispettivamente, con il 71.19% e il 73.65% dei messaggi di Status consegnati correttamente. Nello scenario reale, le prestazioni di ogni nodo sono quasi sempre più elevate per il test OC rispetto alla sua controparte SC (vedi Figura 4.6a), sebbene solamente il nodo due nel test OC abbia ottenuto un risultato peggiore del nodo due nel test SC. D'altra parte, nello scenario controllato, le prestazioni di ogni test sono sempre più elevate per il test OC rispetto alla sua controparte SC per tutte le medie dei 10 test (Figura 4.6b)). Come illustrato nella Tabella 4.7, il PSR degli Status risultante nel test OC è stato significativamente diverso rispetto a quello SC, con una media di 2.88% di differenza. Per quanto riguarda l'OC, sono stati ricevuti il 72.14% degli Status, mentre per SC il 69.26%. Inoltre, la relativa varianza media calcolata sui test SC e OC è stata, rispettivamente, dello 0.62% per l'OC e dello 0.46% per lo SC, come illustrato nella Tabella 4.7.

In conclusione, complessivamente per lo scenario controllato e lo scenario reale, sono stati effettuati test di circa 15 ore ed effettuato il test t per campioni indipendenti per determinare se vi fossero differenze statisticamente significative tra le medie osservate per il test SC e OC. Sia per lo scenario controllato che per lo scenario reale le analisi preliminari non hanno mostrato la presenza di valori anomali né per SC né per OC e i dati relativi alle medie dei singoli test (Avg Single Test) sono risultati distribuiti secondo una distribuzione normale standard in entrambi i test, come verificato dal test Shapiro-Wilk ($p > 0.05$). Per lo scenario controllato, assumendo ipotesi nulla (la differenza tra la media dei gruppi è zero), cioè ipotizzando che le medie dei due test siano uguali tra di loro, risulta che la differenza fra le medie osservate per il test SC e OC è p = 1.48E-07 ($p < 0.5$), quindi statisticamente significativa, Inoltre l'*effect size* è $d = 3.69$, con deviazione standard, rispettivamente, $\sigma = 0.59$ e $\sigma = 0.40$. Per lo scenario reale, è risultato che la differenza tra le medie osservate per il test SC e OC è stata p = 1.46E-07 ($p < 0.5$), quindi statisticamente significativa, con deviazione standard, rispettivamente, $\sigma = 0.71$ e $\sigma = 0.83$. Infine l'*effect size* è $d = 3.73$.

4.6 Analisi delle performance generali della rete basata sugli Status

Per fornire un'analisi approfondita dei risultati ottenuti nei test eseguiti, è stato sviluppato un metodo di valutazione delle prestazioni complessive della rete elaborando i file di log ottenuti da ciascuna prova durante i test svolti e analizzando gli Status persi provenienti da ogni *server*. Questo tipo di analisi è specifico per i nostri scopi e tutti coloro che utilizzano valori binari per il campo *present* (P) e *Target* (T) nei rispettivi modelli. L'utilizzo di modelli con *Present* e *Target* con più di due valori richiede una analisi differente.

I file di log ottenuti da ogni prova per i test SC e OC sono stati elaborati nell'ambiente Matlab per valutare le prestazioni complessive della rete implementata. In particolare, è stato implementato un semplice *script* per contare la quantità totale di Status ricevuti da ciascun nodo *server*, su un totale di 5470 comandi inviati per ogni test. Inoltre, è stata introdotta la capacità di identificare gli Status non ricevuti dal *client*, denominati da questo punto in poi della trattazione con il termine anomalie, differenziando il caso in cui questi vengono persi solo una volta, chiamati isolati e quelli che si perdono più di una volta, chiamati consecutivi. Infatti, le anomalie possono essere identificate dalle analisi effettuate sui campi pacchetto del *Present* On/Off (P) e *Target* On/Off (T) dopo uno Status perso, utilizzando solo le informazioni di cui sopra lato *client*. L'idea generale è stata quella di osservare lo Status ricevuto prima e dopo l'anomalia per capire quello che è successo al suo interno con un certo grado di incertezza. In questo caso, ragionando in questo modo, il termine incertezza indica l'incapacità di determinare univocamente cosa accade all'interno di un sequenza di n-Status persi consecutivamente e sempre riferita allo stesso nodo.

Per dare una corretta contestualizzazione del concetto di incertezza e spiegare più chiaramente la logica utilizzata per ottenere i risultati proposti, verranno mostrate in Figura 4.7 alcuni casi di anomalie che possono verificarsi in un test e che ci hanno infine permesso di formulare delle condizioni generali, come descritte alla fine di questa sezione, per caratterizzare l'incertezza per ogni tipo di anomalia presente all'interno della sequenza P/T.

Come mostrato nel caso (a), la ricezione di uno Status garantisce la ricezione del comando On/Off per ogni singolo elemento del *server*, controllando il valore dei campi del pacchetto P e T nell'istante successivo in cui lo Status è stato ricevuto. Perciò, possiamo formulare la seguente condizione che deve sempre verificarsi all'interno di una sequenza P/T di un messaggio (k-Status) ricevuto:

$$Set_k = P_{k+1} = T_k \tag{4.1}$$

Inoltre, come si può vedere nel caso (b) per la caratterizzazione deterministica

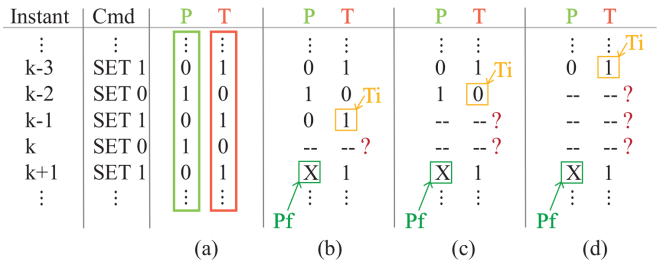


Figura 4.7: Analisi degli Status persi. Caso (a), sequenze P/T degli Status che si susseguono per ogni comando di Set inviato; caso (b), 1-Status perso; caso (c), 2-Status persi consecutivi; caso (d), 3-Status persi consecutivamente.

della sequenza di *Set* e di conseguenza della sequenza P/T, è possibile in modo univoco determinare cosa è successo nell'istante k, in assenza di uno Status ricevuto, analizzando il valore di *Set*, il campo P nell'istante k+1 (*Final Present (Pf)*, e il campo T nell'istante precedente l'anomalia *Initial Target (Ti)*. Questo secondo caso è anche il più semplice, poichè rappresenta uno Status isolato in un certo istante k. Come mostrato, assumendo come in Figura $Pf = x$ dove possiamo avere $x = 0$ o 1, se $Pf \neq Ti$ e $Pf = Set_k$ allora possiamo concludere che all'istante k, anche se il *client* non ha ricevuto lo Status, il comando Set_k è stato ricevuto ed eseguito dal *server*. Al contrario, se $Pf = Ti$, significa che $Pf \neq Set_k \neq Tk$, allora si può dire con certezza che all'istante k il comando $Set = 0$ non è stato eseguito. Anche se la perdita di uno Status riguarda un comando Set_{k-1}, quindi uguale a 1, il ragionamento rimane lo stesso valido: si può concludere che l'incertezza è zero anche in questo caso. In conclusione, nel caso di una perdita di un solo Status, possiamo recuperare sempre le informazioni ad esso associate.

Il caso (c) riguarda la perdita di due Status consecutivi. Se $x = 1$, allora $Pf \neq Ti$, che porta ad avere $Pf \neq Set_k$. In questo modo possiamo dire con certezza che il comando Set_k non è stato eseguito. Con uguale certezza, possiamo anche dire che Set_{k-1} è stato eseguito al contrario di Set_k; infatti, l'unica ipotesi per cui si verifichino entrambe le condizioni iniziali sono che $Pf = Set_{k-1}$. Inoltre, l'incertezza è zero. Se $x = 0$, allora $Pf = Ti$. Non si può dire univocamente che tutti i comandi sono stati eseguiti o il Set_{k-1} non lo sia stato, come anche il Set_k se sia stato eseguito oppure meno durante lo Status perso. Qualsiasi conclusione è valida e non ci sono considerazioni che ci consentano di escludere un caso piuttosto che un altro. In questa situazione l'incertezza è massima e uguale al numero di Status persi.

Anche nell'ultimo caso (d) riportato nel nostro esempio, in cui si ha una perdita di tre Status consecutivi, possiamo distinguere due casi e fare le dovute

Tabella 4.8: Confronto tra il test in configurazione ottimizzata (OC) e in configurazione *standard* (SC) dello scenario controllato, in termini di media e varianza, per gli Status persi (consecutivi e isolati) e Status ricevuti.

	Test	Status Missed			Status Received
		Consecutive	**Isolated**	**Consecutive Respect to the Isolated Ones**	
OC	**Average (%)**	0.01	1.14	**1.29**	98.84
SC		0.25	2.77	**7.80**	96.98
OC	**Variance (%)**	2.93E-04	1.02E-03	**2.07**	1.29E-03
SC		0.03	0.48	**13.32**	0.73

considerazioni. Se $Pf \neq Ti$, ciò significa che qualcosa tra lo Status nell'istante $k-3$ e $k+1$ è avvenuto: sicuramente è stato eseguito uno dei Set=0, ma senza ulteriori informazioni, non possiamo fare considerazioni per ridurre ulteriormente l'incertezza. In questo caso concluderemo che l'incertezza è uguale a n-1, dove n è uguale a numero di Status consecutivi persi. Per il secondo caso, se $Pf = Ti \neq Set_k$, possiamo dire con certezza che il *server* non ha eseguito Set_k; altrimenti, avremmo la seguente uguaglianza P/T, $P_{k+1} = T_k = Set_k$.

I contenuti della Figura 4.7 sono solo alcune possibili combinazioni a titolo di esempio. Potrebbero verificarsi più di tre Status persi consecutivamente, sebbene le possibilità che ciò accada siano molto poche. Per generalizzare la valutazione dell'incertezza I(n) per qualsiasi anomalia presente all'interno della sequenza P/T, abbiamo formulato le seguenti condizioni (4.2):

$$I(n) = \begin{cases} n-1, & \text{se } n \text{ is } dispari \\ n-2|Pf-Ti|, & \text{se } n \text{ is } pari \end{cases} \tag{4.2}$$

ricordando che n indica il numero dei messaggi *Generic OnOff Status* persi consecutivamente, *Ti* il valore del *target* dell'ultimo Status ricevuto prima delle sequenze di Status persi consecutivamente e *Pf* indica l'attuale valore del primo Status ricevuto dopo la sequenza di Status censecutivi persi. L'incertezza *I(n)* si riferisce al calcolo dell'incertezza per ogni anomalia. Per calcolare l'incertezza totale all'interno dell'intero test, sarà quindi necessario sommare tutte le incertezze individuali e metterle in relazione con il numero complessivo di *Set* inviati.

Riguardo lo scenario controllato, per il *trial* condotto in laboratorio, i risultati del suddetto metodo di valutazione delle prestazioni sul confronto tra SC e OC, in termini di media e varianza, sono riportati in Tabella 4.8.

Per quanto riguarda OC, sono stati ricevuti il 98.84% degli Status, quindi i

Tabella 4.9: Confronto dei test in configurazione ottimizzata (OC) e in configurazione *standard* (SC) dello scenario controllato, in termini di media, nel calcolo dell'esecuzione minima dei comandi On/Off e dell'incertezza.

Test		Status Missed						Status Received	Minimum On/Off Commands Executed	Uncertainty
		Consecutive				Isolated				
		Even (n = 2)		Odd (n = 3)						
		I(n) = n	*I(n) = n-2*	*I(n) = n-1*		*I(n) = 0*				
		Pf=Ti	*Pf≠Ti*	*Pf=Ti*	*Pf≠Ti*	*Pf≠Ti*	*Pf=Ti*			
OC	**Average (%)**	0.01	0	0	1.05E-03	1.14	5.48E-04	98.84	99.99	0.01
SC		0.23	0	0	2.40E-02	2.77	0	96.98	99.75	0.25

comandi On/Off sono stati eseguiti con certezza. Inoltre, più dell'1.14% sono risultati isolati, mentre più dello 0.01% degli Status consecutivi sono risultati persi. D'altra parte, è stato ottenuto il 96.98% degli Status ricevuti per lo SC, il che significa che i comandi di On/Off sono stati eseguiti con certezza. Inoltre, per quanto riguarda gli Status persi, più del 2.77% sono risutati isolati, laddove più dello 0.25% sono risultati consecutivi. In particolare, la percentuale di Status consecutivi rispetto agli isolati è stato il 7.80% nel caso di SC e 1.29% per OC. Come spiegato nella sezione precedente, per gli Status isolati siamo sempre in grado di recuperare le informazioni ad essi associate, quindi, se il comando è stato eseguito o meno ($I(n) = 0$). In questo caso, come mostrato in Tabella 4.9, nella analisi proposta, si è in grado di determinare con certezza che la maggior parte dei comandi sono stati eseguiti anche in assenza di ricezione dello Status, lato *client*, quando $Pf \neq Ti$. Solo pochissimi comandi non sono stati realmente eseguiti, quando $Pf = Ti$.

A differenza dell'analisi sugli Status persi consecutivamente, gli Status isolati non aggiungono alcun contributo all'aumento o alla diminuzione globale dell' incertezza. Tuttavia, è possibile per gli Status consecutivi giungere alle stesse conclusioni inequivocabili come per gli quelli isolati. Tuttavia, è necessaria un'analisi più approfondita per cercare di ridurre, ove possibile, l'incertezza associata con questo tipo di anomalie. Cioè, cercare di recuperare qualunque informazione all'interno dell'anomalia, per la quale si può certamente concludere che i comandi, relativi a uno o più Status consecutivi persi, siano stati ricevuti o meno.

L'obiettivo di quanto descritto sopra è aumentare ulteriormente il grado di accuratezza nel riconoscere correttamente i comandi totali eseguiti, e la componente relativa alla totalità dei comandi per i quali possiamo dire che il comando è stato ricevuto o no. In particolare, come mostrato nella Tabella 4.9, sono state sperimentate *pari* anomalie con $n = 2$ e *dispari* con $n = 3$, dove n è uguale al numero di Status consecutivi persi.

Di conseguenza, per quanto riguarda il *trial* effettuato in laboratorio e coe-

Tabella 4.10: Confronto tra il test in configurazione ottimizzata (OC) e in configurazione *standard* (SC) dello scenario reale, in termini di media e varianza, per gli Status persi (consecutivi e isolati) e Status ricevuti.

	Test	Status Missed			Status Received
		Consecutive	Isolated	Consecutive Respect to the Isolated Ones	
OC	**Average (%)**	13.98	13.88	**100.72**	72.14
SC		16.10	14.64	**109.97**	69.26
OC	**Variance (%)**	0.62	0.13	**482.86**	0.62
SC		0.20	0.44	**45.03**	0.46

Tabella 4.11: Confronto dei test in configurazione ottimizzata (OC) e in configurazione *standard* (SC) dello scenario reale, in termini di media, nel calcolo dell'esecuzione minima dei comandi On/Off e dell'incertezza.

Test		Status Missed						Status Received	Minimum On/Off Commands Executed	Uncertainty
		Consecutive				Isolated				
		Even		Odd						
		I(n) = n	*I(n) = n-2*	*I(n) = n-1*		*I(n) = 0*				
		Pf=Ti	***Pf≠Ti***	***Pf=Ti***	***Pf≠Ti***	***Pf≠Ti***	***Pf=Ti***			
OC	**Average (%)**	9.21	0.06	0.04	4.68	13.78	0.10	72.14	87.43	12.47
SC		10.69	0.06	0.04	5.31	14.53	0.11	69.26	85.27	14.44

rentemente con le considerazioni fatte nella sezione precedente, abbiamo trovato che tutte le anomalie consecutive si sono verificate per $n = 2$ con incertezza massima pari a n, quando $Pf = Ti$. Solo in pochi casi si è verificata l'incertezza uguale a $n - 1$ per $n = 3$ e $Pf \neq Ti$. Solo per quest'ultimo caso, come spiegato nella sezione precedente, la riduzione dell'incertezza viene considerata come un contributo nel conteggio dell'aumento dei comandi per cui possiamo certamente dire siano stati eseguiti. In generale, dall'analisi proposta risulta che, il 99.99% dei comandi sono stati eseguiti con certezza, con un'incertezza per i restanti comandi dello 0.01%. Mentre, nell'analisi SC, è stato raggiunto il livello di conoscenza del 99.75% dei comandi certamente eseguiti, con incertezza dello 0.25%.

Considerando lo scenario reale, abbiamo condotto lo stesso metodo di valutazione delle prestazioni sul confronto tra il test SC e OC come per lo scenario controllato precedentemente descritto. Come mostrato nella Tabella4.10, sia per il test OC che SC abbiamo ottenuto una percentuale di consegna degli Status notevolmente inferiore, rispettivamente, 70.81% e 69.09%. Ciò significa che i comandi vengono eseguiti con certezza. Inoltre, per quanto riguarda gli Status

persi, abbiamo sperimentato per l'OC il 100.72% di Status consecutivi rispetto agli isolati (13.98% consecutivi e 13.88% isolati) e per lo SC il 109.97% (16.10% consecutivi e 14.64% isolati). Come illustrato nella Tabella4.10, nello scenario reale gli Status consecutivi sono maggiori in percentuale rispetti agli isolati. In particolare, come mostrato in Tabella4.11, abbiamo ottenuto l'87.43% dei comandi certamente eseguiti, con un'incertezza per i comandi rimanenti del 12.47% nella nostra analisi OC. Il restante 0.10% rappresenta gli Status isolati persi e certamente non eseguiti. Mentre, nell'analisi SC abbiamo raggiunto l'85.27% dei comandi certamente eseguiti, con un'incertezza del 14.44% e dello 0.11% degli Status persi certamente non eseguiti. Inoltre, a differenza dello scenario controllato in cui sono stati trovati gli unici casi consecutivi persi di Status con n = 2 e n = 3, nello scenario reale il numero di casi è molto più ampio, raggiungendo valori con n = 12. Come risulta dall'equazione (4.2), è possibile raggruppare i valori delle incertezze genericamente in casi pari e dispari risultanti dalla somma delle incertezze calcolate per ciascun caso.

4.7 Conclusioni

Questo lavoro ha avuto come obiettivo quello di valutare le prestazioni della ricezione dei messaggi di conferma (Status) nella rete Bluetooth mesh. Il principale punto debole nell'utilizzo di questo tipo di messaggi, in risposta a un comando On/Off utilizzando indirizzi di gruppo, riguarda la presenza di problemi imprevisti quando più nodi rispondono contemporaneamente riducendo la probabilità di recapito dei messaggi stessi a causa delle collisioni tra di loro. In particolare, l'implementazione firmware *standard* in un nodo Bluetooth mesh, ad oggi, non sembra adottare una particolare soluzione volta a risolvere o, almeno limitare, il problema della congestione dei messaggi di Status.

In questo lavoro si propone una tecnica di *spreading* degli Status per superare i problemi nella loro ricezione in una rete mesh Bluetooth. Per valutare le prestazioni della tecnica proposta in termini di Status PSR, si è utilizzato un setup sperimentale composto da una rete Bluetooh mesh operante in due diversi scenari: lo scenario controllato con elementi interferenti limitati e lo scenario reale con numerose fonti interferenti e attenuanti. Per mostrare un confronto tra la procedura *standard* implementata della rete Bluetooth mesh con la tecnica proposta, è stato utilizzato lo stesso setup sperimentale per eseguire due specifici test (SC o OC). Nel primo test si è implementata una rete con un test SC, mentre nel secondo test si è implementata la tecnica proposta con la configurazione ottimizzata (test OC).

Nello scenario controllato i risultati ottenuti in termini di successo complessivo nell'esecuzione dei comandi On/Off dimostrano che il nostro contributo adottato nello OC, influenza positivamente il PSR degli Status, con un miglio-

ramento medio dell'1.86% rispetto a quello SC. A conferma di ciò, abbiamo osservato una differenza nel calcolo varianza per entrambi i test eseguiti da diversi ordini di grandezza, con una varianza media calcolata dello 0.44% per lo SC e 0.002% per OC. Inoltre, è stato proposto un metodo di valutazione approfondito delle prestazioni della rete utilizzando solo le informazioni associate agli Status. Dall'analisi effettuata nell'esecuzione dei comandi di On/Off abbiamo riscontrato che la percentuale di Status consecutivi persi nello SC è molto più alto di quella nello OC. Stesso discorso per gli Status isolati, che si sono rivelati essere più del doppio. Tutto ciò dimostra che il nostro contributo ha portato a una diminuzione significativa del numero di Status persi consecutivi. Quindi, si è ottenuta una riduzione dell'incertezza totale e quindi un aumento nella determinazione inequivocabile dell'esecuzione o meno del comando di On/Off, ottenendo una conoscenza dell'esecuzione del comando di On/Off in media del 99,99% sul totale dei comandi inviati, per ciascuna prova eseguita. Pertanto, nel test OC, abbiamo ottenuto un miglioramento complessivo riguardante il comportamento generale della rete, per la quantità totale di messaggi di Status ricevuti dal *client* e un'uniformità generale nel numero di Status ricevuti da tutti i nodi, rispetto al SC. Inoltre, l'OC proposto porta a una significativa diminuzione del numero di Status consecutivi persi e dell'incertezza nella determinazione inequivocabile dell'esecuzione del comando On/Off.

Nello scenario reale, abbiamo ottenuto un miglioramento medio dell'2.88% nel test OC rispetto a quello SC. In questo scenario così come in quello reale, il miglioramento delle prestazioni ottenuto nel test OC rispetto a quello SC è comunque significativo come dimostra il test t condotto e l'*effect size*. In generale, il livello complessivo di successo dei pacchetti di test SC e OC nello scenario reale sono inferiori agli stessi test eseguiti nello scenario controllato. Ciò può essere dovuto ai numerosi elementi interferenti dello scenario reale come: l'interferenza del co-canale da un punto di accesso WIFI e altri dispositivi Bluetooth, l'attenuazione, i riflessi da elementi strutturali e di arredo e la presenza variabile di persone che occupano l'ambiente durante le prove. Queste variabili chiaramente non sono controllabili, configurandosi come fonte di interferenze casuali.

Questo lavoro propone una tecnica per migliorare la ricezione dei messaggi di conferma basata su un algoritmo generatore di numeri pseudo-casuali con *seed* ottenuto dai primi quattro *byte* dell'indirizzo Bluetooth del dispositivo. Questa tecnica ha rappresentato un buon compromesso tra la riduzione dei pacchetti persi e il basso carico computazionale, ottenendo un miglioramento delle prestazioni nei test di valutazione adottati. Gran parte della qualità del lavoro è quella di aver ottenuto un miglioramento complessivo rispetto al comportamento generale della rete, sia in termini di quantità totale di messaggi di conferma ricevuti dal *client*, sia di un'uniformità generale nel numero di questi

messaggi ricevuti da tutti i nodi rispetto all'implementazione *standard*, che, ad oggi, non sembra affrontare una particolare soluzione volta a risolvere o, almeno limitare, il problema della congestione dei messaggi. Tuttavia, considerando l'aumento del numero di dispositivi nella rete, potrebbero essere adottati algoritmi ad alte prestazioni per evitare collisioni, ma avrebbero lo svantaggio di aumentare i carichi computazionali. Questi aspetti verranno esaminati in lavori futuri, incluso un maggior numero di test eseguiti in diversi ambienti maggiormente sfidanti. Questi test purtroppo, per via delle restrizioni generali imposte a causa del periodo pandemico dovuto all'emergenza Covid-19 per il momento non si sono potuti eseguire.

Capitolo 5

Implementazione di una procedura OTA-DFU

Spesso nelle reti *wireless* molto estese si rende necessaria una procedura di aggiornamento del *firmware* per ogni suo nodo che generalmente non avviene in tempi brevi e risulta semplice da eseguire: spesso si ha a che fare con una moltitudine di nodi da aggiornare e non sempre questi sono accessibili fisicamente una volta installati.

Una rete mesh ad esempio può contenere centinaia di dispositivi e il loro singolo aggiornamento può richiedere molto tempo. Per aggirare questa problematica è necessario consentire ai dispositivi mesh di trasmettere ai loro vicini i dati che stanno ricevendo, assicurandosi così che il trasferimento raggiunga tutti i dispositivi nella rete. Oltretutto, una procedura di aggiornamento del *firmware over-the-air* (OTA) è preferibile o, quantomeno necessaria, quando non è possibile accedere fisicamente ai singoli nodi.

In questo capitolo, al fine di rendere più pratico e rapido l'aggiornamento necessario del *firmware Light Switch* adottato nei test descritti nel capitolo precedente 4, è stata implementata una procedura per l'aggiornamento, OTA-DFU, con DFU è inteso *Device Firmware Update*.

Per eseguire il DFU su mesh, è stato implementato il protocollo proprietario sviluppato dall'azienda Nordic nei singoli nodi, denominato DFU mesh. Inoltre, lo sviluppo della soluzione OTA-DFU implementata è nata anche come caso d'uso dall'esigenza, in considerazione del periodo di emergenza pandemica, di poter effettuare test anche da remoto. Ad ogni modo OTA-DFU resta un'esigenza implementativa imprescindibile, ad esempio, in ambito industriale.

Questa procedura, sulla base del protocollo Nordic, consente di aggiornare i nodi contemporaneamente in backgroud, mentre la rete continua a funzionare correttamente. In particolare, ogni nodo riceve contemporaneamente, in *backgroud* DFU, il nuovo aggiornamento *firmware* senza perdere l'operabilità della rete. Durante il trasferimento del *firmware* quindi il nodo continua a funzionare; solamente quando il trasferimento è finito l'applicazione carica il nuovo *firmware*. Inoltre, per gestire in maniera più rapida ed automatica la

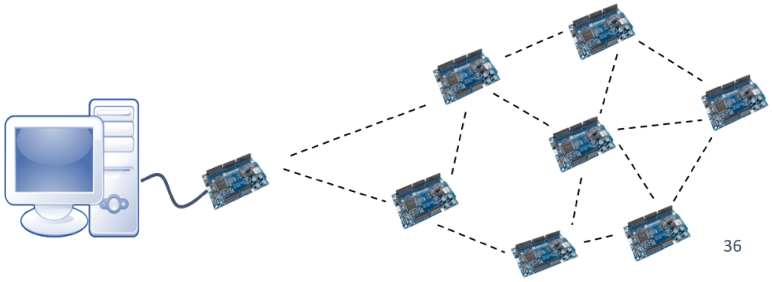


Figura 5.1: Esempio di procedura DFU mesh.

procedura OTA-DFU sui singoli nodi, è stato creato un apposito *tool*. Il *tool* contiene tutte le istruzioni da eseguire, oltre alla possibilità di aggiornare ed inviare alcuni file, necessari per portare a termine con successo ogni nuovo aggiornamento del firmware. Tramite un menù iniziale è possibile gestire l'intera procedura. Inoltre tale procedura consente di aggiornare tutti i nodi oppure solo un sottogruppo.

5.1 Il protocollo DFU mesh

Il protocollo DFU mesh consente ai dispositivi di ritrasmettere i pacchetti di dati che ricevono, assicurandosi così che il trasferimento DFU raggiunga tutti i dispositivi nella rete. Questo metodo è molto più veloce rispetto al passaggio dell'intero trasferimento DFU a ciascun dispositivo individualmente. La figura 5.1 mostra un esempio di aggiornamento del firmware in cui una board connessa al PC tramite seriale riceve i pacchetti di aggiornamento e nello stesso tempo li ritrasmette a tutte le board nel suo range di visibilità. Il protocollo DFU mesh è basato sull'*advertising* a differenza del normale funzionamento Bluetooth mesh non supporta l'*addressing*, i messaggi *acknowledged* o la criptazione dei dati.

Senza entrare nel dettaglio del codice DFU implementato nel *firmware Light Switch* [50], un dispositivo che riceve un trasferimento DFU eseguirà i seguenti passaggi per ogni messaggio di dati che riceve:

- Verifica che il pacchetto dati non sia già stato ricevuto.
- Memorizzazione del pacchetto di dati nella memoria flash all'offset appropriato per dati trasferiti.
- Contrassegno del pacchetto di dati come ricevuto.
- Ritrasmissione del pacchetto di dati, utilizzando la funzionalità *relay* un numero di volte predefinito, a un intervallo regolare.

I dispositivi che non sono direttamente interessati al trasferimento dei dati al loro interno eseguiranno solo i passaggi 1, 3 e 4 per garantire che i dispositivi di destinazione più lontani nella rete possano ricevere i pacchetti.

Inoltre, l'intera procedura DFU mesh prevede il trasferimento di alcuni pacchetti[1], che andranno ad occupare una area ben precisa dello spazio nella memoria flash, come mostrato in Figura 5.2, e di seguito brevemente descritti:

- *Applicazione*, la quale durante il trasferimento occuperà uno spazio nella memoria flash del dispositivo. Questo spazio dovrà essere inutilizzato, per evitare di sovrascrivere i dati dell'applicazione in esecuzione.

- *Bootloader*, il quale, una volta finito il trasferimento della nuova applicazione, si occuperà di copiarla nell'area dell'applicazione in sostituzione alla precedente[2].

- *Softdevice*, è la parte del firmware che implementa il protocollo Bluetooth nel microcontrollore.

- *Device page*, definisce la configurazione del dispositivo e contiene i parametri operativi per il *bootloader*. *Device page* deve essere generata su un computer host e visualizzata su ogni dispositivo prima della distribuzione.

Questa procedura rende possibile l'aggiornamento dei nodi contemporaneamente in backgroud, mentre la rete continua a funzionare correttamente sfruttando le caratteritiche del protocollo Bluetooth mesh. In particolare, ogni nodo riceve contemporaneamente, in *backgroud* DFU, il nuovo aggiornamento *firmware* in modalità di trasferimento *bootloader* DFU, senza perdere l'operabilità della rete. Durante il trasferimento del *firmware* il nodo continua a funzionare; solamente quando il trasferimento è finito l'applicazione carica il nuovo *firmware*.

5.2 Processo di configurazione OTA-DFU

I seguenti step implementativi [51] necessitano dell'installazione iniziale e l'utilizzo del *toolnrfutil*[3] oltre che del tool *nrfjprog*[4] e altri file presenti nelle varie sotto-directory dell'SDK mesh v3.1.0 (ultima versione rilasciata al momento

[1] Ognuno può essere trasferito separatamente ed avere una propria area di memoria.
[2] Da tenere presente che la l'applicazione deve essere posizionata in un'area flash sufficientemente grande da contenere l'intera applicazione in entrata durante il trasferimento e che non si sovrapponga alla nuova o alla vecchia applicazione
[3] nrfutil è un pacchetto Python che include l'utility da riga di comando nrfutil e la libreria nordicsemi, quest'ultima scritta per Python 2.7
[4] Disponibile nel sito nordicsemi.com

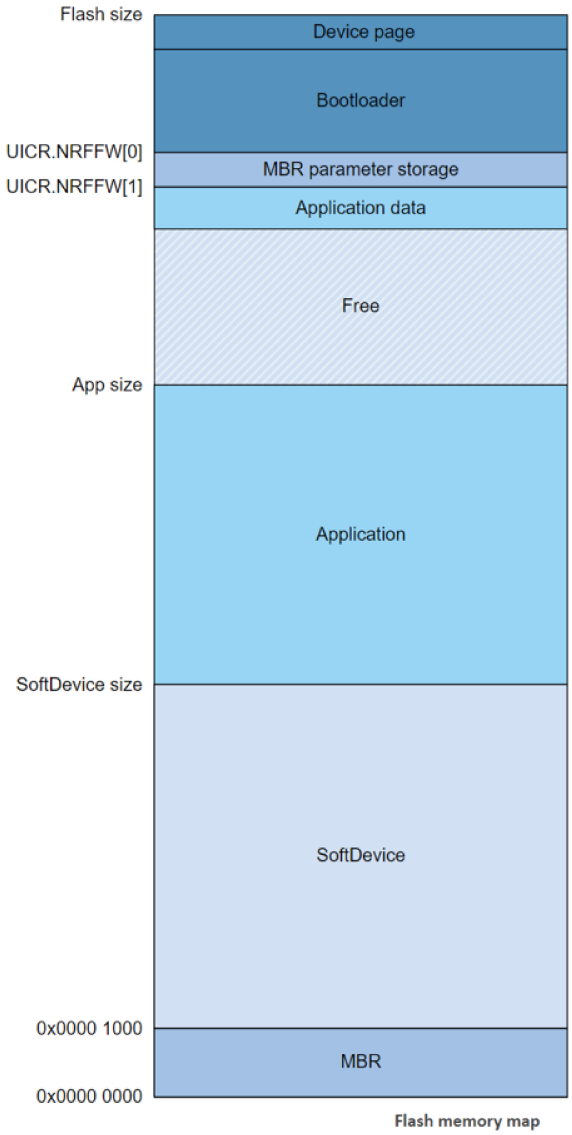


Figura 5.2: Mappa della memoria flash.

della realizzazione della presente realizzazione). Inoltre è stato necessario installare Python 2.7 e i relativi file e moduli associati per una corretta esecuzione del tool *nrfutil*[5]

[5]https://github.com/NordicSemiconductor/pc-nrfutil/tree/meshdfu.

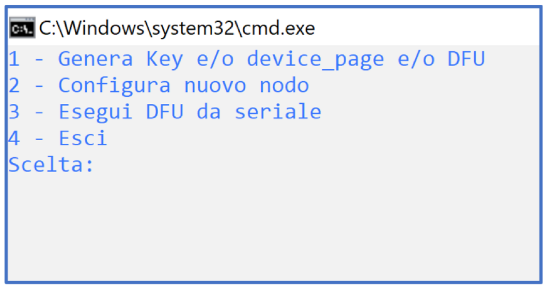


Figura 5.3: Menù creato per la configurazione del DFU mesh.

5.2.1 Fasi preliminari

Prima della preparazione vera e propria del dispositivo per la configurazione DFU mesh, come prima fase si procede eseguendo delle operazioni cosidette preliminari, scegliendo da menù creato (Figura 5.3) la prima voce: generazione di una coppia di chiavi (pubblica e privata) per rendere più sicura la procedura OTA-DFU; creazione della *device page*; creazione dell' archivio DFU. Ognuno dei precedenti termini sarà spiegato nel dettaglio nel seguito di questo paragrafo. Si procede quindi alla:

1. Generazione (opzionale) di una coppia di chiavi (pubblica e privata) tramite *nrfutil*. Questo comando, alla riga 3 del seguente codice, crea un file *private_key.txt* nella *directory* corrente (in cui si sta lavorando). Questa chiave deve essere condivisa solo con fonti attendibili. Se persa, si perde anche l'autorizzazione per eseguire in futuro gli aggiornamenti DFU sui dispositivi. L'unico modo per inserire una nuova chiave privata valida è eseguire nuovamente il suo caricamento nel dispositivo manualmente.

```
1 ...
2
3 nrfutil keys --gen-key private_key.txt
4 nrfutil keys --show-vk hex private_key.txt
```

Listing 5.1: Generazione coppia di chiavi

Il risultato sarà un *output* simile a quello nelle seguenti righe di codice. L'unione delle due stringhe HEX, nelle righe 1, 2, 3 e 4 costituiscono la chiave pubblica.

```
1 Verification key Qx: ed09a58df6db5cd15b8637304f31d31
2 f4042492ed7c7e4839fbe903f260a2ba1
3 Verification key Qy: a855e92b72885825481ad56282bcf549
4 ad7455ec46f000f0f62d97eeec883ba6
5 ...
```

Listing 5.2: Strighe HEX

2. Aggiunta della chiave pubblica generata con *nrfutil* alla *device page*. La chiave pubblica deve essere aggiunta nel file di configurazione del *bootloader*, quest'ultimo usato per creare la *device page*. Nella cartella *tools/dfu* dell'SDK mesh si procede quindi alla modifica del file *bootloader_config_default.json* per preparare la *device page*. Ciò consente al dispositivo di verificare che la persona che ha avviato il trasferimento DFU abbia la chiave privata associata a questa chiave pubblica. Inoltre, è anche possibile modificare la voce dell'ID dell'azienda, *company_id* nella *device page* per non rischiare di avere conflitti con altri ID di altre aziende.

```
    ...

{
    "bootloader_config":
    {
        "bootloader_id": 1,
        "bootloader_version": 1,
        "company_id": 89,
        "application_id": 1,
        "application_version": 1,
        "public_key":"
    ed09a58df6db5cd15b8637304f31d31f4042492ed7c7e4839fbe903f26
        0
    a2ba1a85e92b72885825481ad56282bcf549ad7455ec46f000f0f62d97
        eeec883ba6"
    }
}

...
```

Listing 5.3: Device page

3. Preparazione del DFU. Per eseguire un trasferimento DFU, è necessario creare un archivio DFU con *nrfutil*, fornendo alcuni argomenti che corrispondono alla *device page* del dispositivo. L'archivio DFU è un *file .zip* che contiene il binario dell'applicazione insieme ad alcuni metadati. Dal sottomenù è possibile scegliere se creare o no un nuovo pacchetto DFU, come nel codice seguente.

```
 ...


SET /P Inserisci la patch del file .hex per la generazione del
     pacchetto DFU
set PATH=
echo Hai digitato: %PATH%
SET /P Inserisci il nome da assegnare al pacchetto DFU
set NOME=
```

```
9  echo Hai digitato: %NOME%
10 SET /P Inserisci nuovo application version (application
       version attuale + 1)
11 set APP_VER=
12 set /P APP_VER=Inserisci application version
13 echo Hai digitato: %APP_VER%
14 nrfutil dfu genpkg --application %PATH%
15     --company-id 0x00000059
16     --application-id 1
17     --application-version %APP_VER%
18     --key-file private_key.txt
19     --sd-req 0x00AF rem for NRF52832
20     --mesh %NOME%.zip
21
22 ...
```

Listing 5.4: Preparazione del DFU

Il comando *nrfutil dfu genpkg*, nella riga 14, genera un archivio DFU nella directory corrente, dopo aver inserito, come mostrato dal codice riportato sopra, tutte le informazioni richieste. È necessario quindi inserire la *patch* per il file HEX di un'applicazione, nel nostro caso il file *Light_Switch_Server_DFU_NRF52832*[6]. I valori *–company-id* e *–application-id* nelle righe 14 e 15 devono corrispondere ai valori utilizzati per generare la *device page* del dispositivo. In particolare, *–Application-version* deve inoltre essere maggiore del numero di versione utilizzato per l'immagine *firmware* precedente. *–key -file private_key.txt* deve contenere le chiavi generate nelle fasi preliminari, quindi la sua chiave pubblica deve corrispondere con quella presente nella *device page* del dispositivo. Bisogna quindi utilizzare il file HEX che corrisponde al *chip* e al *Softdevice* che si sta utilizzando: in questo caso la versione 0x00AF per la *board* nRF52832 utilizzata per i test svolti. Ad esempio, per un primo aggiornamento del firmware è necessario settare *–Application-version* su 2. Un dispositivo inoltre accetta solo trasferimenti di applicazioni che corrispondano alla propria azienda e *application IDs* e hanno un numero di versione superiore.

4. Generazione della versione HEX della *device page*. Lo *script device_page _generator.py* situato nella cartella del file della *device page*, in *tools/dfu* crea un file HEX della *device page* nella cartella *tools/dfu/bin*. Questo file deve essere caricato, insieme alla prima applicazione, su tutti i dispositivi da aggiornare. È necessario specificare il numero seriale del dispositivo (opzione -d) e la versione Softdevice (opzione -sd). Tutte le *device page* contengono una voce *SD_VERSION*, che deve corrispondere al valore –sd-req passato a *nrfutil* durante la generazione dell'archivio DFU

[6]Il file .hex viene generato quando si crea lo stack mesh.

nel passaggio precedente. La mancata corrispondenza dei parametri dei requisiti della versione del Softdevice farà sì che il dispositivo rifiuti il trasferimento, poiché il proprio *firmware* ID non corrisponderà a quello nel trasferimento. Il comando nel seguente codice crea la *device page* che verrà caricata insieme alla prima applicazione su tutti i dispositivi.

```
1
2 ...
3
4 device_page_generator.py -d nrf52832_xxAA -sd "s132_6.1.0"
5
6 ...
```

Listing 5.5: Generazione della *device page*

5.2.2 Caricamento dei file in tutti i dispositivi

Una volta eseguiti tutti gli step precendenti, l'uso di *nrfjprog* nel seguente codice permette di effettuare nell'ordine:

- L'*erase* di tutti i dati nel dispositivo (incluso UICR[7]), come mostrato nella riga 5.
- Il caricamento del *Softdevice*, disponibile nella cartella al percorso *bin/-softdevice*, mostrato nella riga 8.
- Il caricamento del *bootloader* seriale, stando attenti che la sua versione corrisponda a quella del *chip*, mostrato nella riga 11.
- Il caricamento della prima applicazione, alla riga 14.
- Il caricamento della *device page* nei dispositivi, alla riga 18.
- Il *reset* di ciascun dispositivo, nella riga 19.

```
1
2 ...
3
4 echo. Erase all chip memory (including UICR) on all devices
5 nrfjprog --eraseall
6
7 echo. Flash the SoftDevice on all devices
8 nrfjprog --program MESH\bin\softdevice\s132_nrf52_6.1.0_softdevice
    .hex --chiperase --verify
9
10 echo. Flash the serial bootloader on all devices
11 nrfjprog --program MESH\bin\bootloader\gccarmemb\
    mesh_bootloader_serial_gccarmemb_nrf52832_xxAA.hex --verify
```

[7]

```
12
13 echo. Flash the first application on all devices
14 nrfjprog --program  MESH\examples\dfu\build\
       dfu_nrf52832_xxAA_s140_6.1.0_Debug\dfu_nrf52832_xxAA_s132_6
       .1.0.hex  --verify
15
16 echo. Flash the device page on all devices
17 nrfjprog --program MESH\tools\dfu\bin\
       device_page_nrf52832_xxAA_s132_6.1.0.hex --verify
18
19 nrfjprog --reset
20
21 ...
```

Listing 5.6: Caricamento dei file

A questo punto tutto è pronto per eseguire il DFU attraverso la mesh.

5.2.3 Trasferimento dell'archivio DFU su seriale tramite *nrfutil*

Tramite la scelta 3 del menù iniziale è ora possibile connettere la prima *board* tramite seriale al PC per iniziare la fase di trasferimento del firmware anche alle altre *board*, come descritto nel seguente codice di per se auto-esplicativo.

```
1 ...
2
3 nrfutil --verbose dfu serial -pkg Light_Switch_Server_DFU_NRF52840
      .zip -p %INPUT% -b 115200 -fc --mesh
4 ...
```

Listing 5.7: Trasferimento dell'archivio DFU su seriale tramite *nrfutil*

Per ottenere un output più dettagliato, è possibile aggiungere *–verbose* prima degli argomenti. Il baud rate è impostato di default a 115200. Al termine, il bootloader passa all'applicazione. Non si può eseguire due volte il DFU con lo stesso archivio DFU, perché l' *application version* nella *device page* nel dispositivo viene incrementata alla versione più recente. Pertanto, il bootloader rifiuterà qualsiasi tentativo di trasferire nuovamente lo stesso firmware. In questo caso l'applicazione appena trasferita include anche il supporto DFU, quindi può essere di nuovo aggiornata in futuro quando necessario. Per eseguire un successivo trasferimento DFU, sarà necessario eseguire nuovamente le fasi di preparazione e il trasferimento, ma con un *application version* aumentato: ad esempio, –application-version 3.

5.3 Setup sperimentale

Lo stesso setup utilizzato per i test condotti nel capitolo è stato impiegato per eseguire l'aggiornamento OTA-DFU: tutti e 10 i *server* che necessitavano dell'aggiornamento sono rimasti nelle medesime posizioni, con gli stessi parametri di configurazione impostati precedentemente per la rete, come descritto nella sottosezione 4.2.1. Durante la procedura di aggiornamento il file .zip contenente il nuovo *firmware Light Switch Server*, aveva un peso totale di 353 KB.

5.4 Analisi delle prestazioni

Durante le prove effettuate nel trasferimento del *firmware Light Switch Server* la procedura di aggiornamento è risultata piuttosto lenta: questo comportamento era prevedibile dato che il DFU mesh implementato dalla Nordic deve garantire una comunicazione affidabile anche tramite la ridondanza dei dati scambiati, a discapito del *throughput*.

Inoltre, sono riportate qui di seguito alcune caratteristiche di rete che possono abbassare il *throughput*[8], ovvero:

- Densità di rete: la quantità di dispositivi che si trovano nel raggio d'azione radio l'uno dell'altro influisce in modo significativo sulla velocità di ricezione dei pacchetti. Un numero maggiore di dispositivi entro la distanza radio l'uno dall'altro provoca più collisioni di pacchetti, il che riduce il throughput totale.

- Estensione della rete: ogni salto nella rete comporta un certo rischio di perdita di pacchetti e crea un ritardo nel traffico. Maggiore è il numero di salti nella rete, maggiore è il rischio che alcuni dispositivi perdano il trasferimento.

- Topologia di rete: sebbene un'elevata densità di nodi possa avere un impatto negativo sulla percentuale di successo del trasferimento, avere un numero troppo basso di percorsi per raggiungere un nodo di destinazione può causare perdite di pacchetti. Più un nodo di destinazione si basa su diversi dispositivi di inoltro per avere successo con le loro trasmissioni, maggiore è la probabilità di perdere quel nodo di destinazione ad un certo punto durante il trasferimento.

- Rumore esterno: se distribuito in un ambiente rumoroso, la DFU Mesh offre prestazioni peggiori (come tutte le tecnologie wireless).

[8] Infocenter Nordic, *Mesh DFU protocol*

I dati DFU vengono inviati in blocchi di 16 byte a intervalli regolari, con alcune trasmissioni ridondanti per ogni pacchetto per garantire che tutti i dispositivi lo ricevano. L'intervallo del pacchetto è controllato dal dispositivo di origine del trasferimento.

Durante le prove condotte la velocità di trasferimento del DFU mesh è stata lasciata come impostazione predefinita a 500 ms per ogni pacchetto inviato, rendendo la velocità di trasferimento 16 B/500 ms = 32 B/s). Durante la procedura di aggiornamento dei 10 nodi *Light Switch Server*, (353 KB in totale), i risultati ottenuti hanno portato a concludere, come ci si aspettava, che il tempo necessario per il loro aggiornamento completo non dipende dal numero dei nodi coinvolti. Infatti effettuando alcune prove di aggiornamento rispettivamente con 2, 6 e 10 nodi, in tutte e tre le situazioni il tempo di aggiornamento totale è stato di 3 ore, utilizzando il valore ottimale di *default* di 500 ms/pacchetto.

Capitolo 6

Integrazione della tecnologia beacon nella rete Bluetooth mesh

Una volta creata la rete Bluetooth mesh, occorre poter veicolare informazioni aggiuntive, relative a servizi basati sulla tecnologia Bluetooth. L'universo IoT intravede enormi potenzialità e possibilità di implementare nuovi servizi basati su questa tecnologia.

Un'infrastruttura di rete, come quella proposta nel capitolo 4 si pone come la candidata ideale per poter veicolare informazioni di questo tipo. A tal fine, la funzionalità *beacon* può essere integrata nei dispositivi stessi della rete mesh o su dispositivi sensori o trasduttori esterni ad essa. Le informazioni che i *beacon* forniscono possono essere definite statiche o dinamiche: le prime non possono essere modificate dall'utente una volta inserite a livello di programmazione, mentre le seconde possono essere modificate in qualunque momento dall'utente.

Nel seguito di questo capitolo verrà descritta l'integrazione realizzata tra due applicativi con funzionalità *beacon* basati sul protocollo BLE, e l'applicativo *Light Switch Server* basato sul protocollo Bluetooth mesh. In particolare, nello sviluppo dei due applicativi sono stati presi in esame due formati *standard* di *beacon*: l'iBeacon di Apple e l'Eddystone di Google. Verrano descritti quindi i relativi formati dei *frame* che si trovano all'interno dei pacchetti BLE periodicamente inviati, unitamente ai due tipi di implementazioni *firmware* realizzate. Nella prima, verranno implementati entrambi i formati *beacon* statici iBeacon ed Eddystone all'interno del *firmware Light Switch Server* precedentemente realizzato. Nella seconda verrà presentata la coesistenza del formato dinamico Eddystone beacon all'interno del *firmware Light Switch*, sfruttando le caratteristiche dei servizi GATT *Eddystone Configuration* e PB PROXY GATT.

6.1 Implementazione di beacon statici in un applicativo mesh per il *lighting*

In questa sezione verrà illustrata l'implementazione di *beacon* statici, ovvero che non possono essere cambiati una volta che un dispositivo è stato programmato, all'interno dell'applicativo *Light Switch Server*.

6.1.1 Implementazione iBeacon

L'applicatico sviluppato, utilizzando l'SDK mesh messo a disposizione dalla Nordic, mostra come sia possibile eseguire il *beaconing* simultaneo, nel caso specifico iBeacon e Eddystone, mentre i nodi partecipano alla rete mesh.

Per inviare *beacon*, l'applicazione sviluppata utilizza il *packet manager* interno alla mesh e la struttura del modulo *advertiser* del livello *Bearer*. L'applicazione inizializza prima l'*advertiser*, necessaria solo una prima volta. Quindi alloca e riempie i campi del pacchetto. Non è necessario impostare il tipo di pacchetto o l'indirizzo dell'advertisement, poiché questo viene gestito dal modulo *advertiser* stesso. Dopo l'inizializzazione, l'applicazione schedula il pacchetto per la trasmissione mettendolo nella coda TX dell'*advertiser*, con un parametro che indica il numero di ripetizioni che l'*advertiser* farà. Il conteggio delle ripetizioni è impostato di default, su BEARER_ADV_REPEAT_INFINITE, il che fa sì che il pacchetto venga ritrasmesso per sempre o fino a quando non viene sostituito da un pacchetto diverso.

Quindi, nello *script* principale all'interno del *main(void)* si utilizzano le funzioni *initialize()* e *start()*: la prima, per inizializzare il provisioning dei dispositivo nella rete mesh e, successivamente, l'*advertiser*; la seconda, un volta che il dispositivo è stato aggiunto nella rete mesh, consente di iniziare la trasmissione *beacon*.

Nella funzione *adv_start()*, all'interno di *start()*, viene definita una variabile chiamata *adv_data* che contiene i bit del payload del pacchetto di advertising da inviare. Se nel campo Manufacturer ID del *beacon* si inserisce 0x004C si hanno *beacon* per dispositivi Apple. La variabile *adv_data* è stata quindi modificata per mandare iBeacon, come nel codice sotto riportato.

```
1
2
3 static const uint8_t adv_data[] =
4     {
5         APP_FLAG_BEACON_LENGTH, /* Flag Beacon length (including
      type, but not itself) */
6         BLE_GAP_AD_TYPE_FLAGS, /**< Flags for discoverability. */
7
      BLE_GAP_ADV_TYPE_EXTENDED_CONNECTABLE_NONSCANNABLE_UNDIRECTED,
```

```
    /**< Connectable non-scannable undirected advertising events
    using extended advertising PDUs. */
8       APP_ADVERTISER_LENGTH, /* Advertiser data length (
    including type, but not itself) */
9       BLE_GAP_AD_TYPE_MANUFACTURER_SPECIFIC_DATA, /**<
    Manufacturer Specific Data. */
10      APP_COMPANY_IDENTIFIER, /* Company ID */
11      APP_iBEACON_TYPE, /* iBeacon Type */
12      APP_iBEACON_LENGTH, /* iBeacon Lenght */
13      APP_BEACON_UUID,  /* UUID */
14      APP_MAJOR_VALUE, /* Major Value*/
15      APP_MINOR_VALUE, /* Minor Value */
16      APP_MEASURED_RSSI  /* RSSI at 1 m */
17  };
```

Listing 6.1: Configurazione della variabile *adv_data* per iBeacon

avendo dichiarato nel main i seguenti valori costanti:

```
1  /* Defines for iBeacon packet */
2  #define APP_FLAG_BEACON_LENGTH          0x02
               /**< Length of the discoverability Beacon. */
3  #define APP_ADVERTISER_LENGTH           0x1A
               /**< Total length of the iBeacon. */
4  #define APP_iBEACON_LENGTH              0x15
               /**< Length of manufacturer specific data in the
       advertisement. */
5  #define APP_iBEACON_TYPE                0x02
               /**< iBeacon Type. */
6  #define APP_MEASURED_RSSI               0xC3
               /**< The Beacon's measured RSSI at 1 meter distance in
        dBm. */
7  #define APP_COMPANY_IDENTIFIER          0x59, 0x00
               /**< Company identifier for Nordic Semiconductor ASA.
       as per www.bluetooth.org. */
8  #define APP_MAJOR_VALUE                 0x01, 0x02
               /**< Major value used to identify Beacons. */
9  #define APP_MINOR_VALUE                 0x03, 0x04
               /**< Minor value used to identify Beacons. */
10 #define APP_BEACON_UUID                 0x01, 0x12, 0x23, 0x34, \
11                                         0x45, 0x56, 0x67, 0x78, \
12                                         0x89, 0x9A, 0xAB, 0xBC, \
13                                         0xCD, 0xDE, 0xEF, 0xF0
             /**< Proprietary UUID for Beacon. */
```

Listing 6.2: Definizione dei dati del pacchetto iBeacon

ed i valori nella libreria "BLE_GAP.h":

Per trasmettere i *beacon*, nel *firmware Light Switch Server* sono state aggiunte le seguenti funzioni prese dall'SDK Mesh. Nella funzione mesh_init, all'interno di initialize() è stata aggiunta nrf_mesh_rx_cb_set così definita:

```
1  void nrf_mesh_rx_cb_set(nrf_mesh_rx_cb_t rx_cb)
```

```
{
    m_rx_cb = rx_cb;
}
```

Listing 6.3: Definizione della funzione RX *callback*

e la seguente dichiarazione:

```
static void rx_cb(const nrf_mesh_adv_packet_rx_data_t * p_rx_data)
{
    LEDS_OFF(BSP_LED_0_MASK);  /* @c LED_RGB_RED_MASK on pca10031
    */
    char msg[128];
    (void) sprintf(msg, "RX [@%u]: RSSI: %3d ADV TYPE: %x ADDR:
    [%02x:%02x:%02x:%02x:%02x:%02x]",
                   p_rx_data->p_metadata->params.scanner.timestamp
    ,
                   p_rx_data->p_metadata->params.scanner.rssi,
                   p_rx_data->adv_type,
                   p_rx_data->p_metadata->params.scanner.adv_addr.
    addr[0],
                   p_rx_data->p_metadata->params.scanner.adv_addr.
    addr[1],
                   p_rx_data->p_metadata->params.scanner.adv_addr.
    addr[2],
                   p_rx_data->p_metadata->params.scanner.adv_addr.
    addr[3],
                   p_rx_data->p_metadata->params.scanner.adv_addr.
    addr[4],
                   p_rx_data->p_metadata->params.scanner.adv_addr.
    addr[5]);
    __LOG_XB(LOG_SRC_APP, LOG_LEVEL_INFO, msg, p_rx_data->
    p_payload, p_rx_data->length);
    LEDS_ON(BSP_LED_0_MASK);  /* @c LED_RGB_RED_MASK on pca10031
    */
}
```

Listing 6.4: Dichiarazione della funzione RX *callback*

La funzione *Packet RX callback* consente di ricevere tutti i pacchetti di *advertising*, non filtrati e conformi alle specifiche BLE, all'interno dell'applicazione. Questi pacchetti vengono catturati dal modulo *Scanner* del livello *bearer*. Il tipo *ble_packet_type_t* elenca tutti i pacchetti di *advertising* che è possibile ricevere. Una volta che un nuovo pacchetto viene catturato dallo *scanner*, viene passato attraverso la RX *callback* fornita all'applicazione utente.

Per ascoltare i pacchetti di *advertising*, viene eseguita una RX callback chiamando la funzione *nrf_mesh_rx_cb_set()*. Come input, la funzione di RX *callback* accetta un puntatore ad una struttura di parametri che contiene tutti i dati disponibili sul pacchetto in arrivo. La RX *callback* viene richiamata per tutti i pacchetti che vengono elaborati dalla mesh dopo che la mesh stessa li

ha elaborati. Ovviamente, la mesh presuppone che tutti i pacchetti in arrivo aderiscano al formato del pacchetto di *advertising* BLE.

Nel *main* sono inoltre state aggiunte le seguenti dichiarazioni:

```
static advertiser_t m_advertiser ;
static uint8_t m_adv_buffer [ADVERTISER_BUFFER_SIZE] ;
#define ADVERTISER_BUFFER_SIZE (64)
```

Listing 6.5: Dichiarazioni *m_advertiser* e *m_adv_buffer*.

e la libreria *advertiser.h* che definisce il tipo di variabile *advertiser_t*.

Sempre nella funzione *mesh_init()* è stata aggiunta *adv_init()* definita come di seguito:

```
static void adv_init (void)
{
 advertiser_instance_init(&m_advertiser, NULL, m_adv_buffer,
    ADVERTISER_BUFFER_SIZE);
}
```

Listing 6.6: Definizione della funzione *mesh_init()*.

La funzione *advertiser_instance_init()* può essere richiamata più volte per inizializzare differenti *advertiser*. Nella funzione *start()* è stata aggiunta *adv_start()* definita come nel codice seguente:

```

static void adv_start(void)
{
    bearer_adtype_add(BLE_GAP_AD_TYPE_COMPLETE_LOCAL_NAME);

    advertiser_enable(&m_advertiser);

#ifdef iBeacon
    static const uint8_t adv_data[] =
    {
        APP_FLAG_BEACON_LENGTH, /* Flag Beacon length (including
    type, but not itself) */
        BLE_GAP_AD_TYPE_FLAGS, /**< Flags for discoverability. */

    BLE_GAP_ADV_TYPE_EXTENDED_CONNECTABLE_NONSCANNABLE_UNDIRECTED,
    /**< Connectable non-scannable undirected advertising events
    using extended advertising PDUs. */
        APP_ADVERTISER_LENGTH, /* Advertiser data length (
    including type, but not itself) */
        BLE_GAP_AD_TYPE_MANUFACTURER_SPECIFIC_DATA, /**<
    Manufacturer Specific Data. */
        APP_COMPANY_IDENTIFIER, /* Company ID */
        APP_iBEACON_TYPE, /* iBeacon Type */
        APP_iBEACON_LENGTH, /* iBeacon Lenght */
        APP_BEACON_UUID,  /* UUID */
        APP_MAJOR_VALUE, /* Major Value*/
        APP_MINOR_VALUE, /* Minor Value */
```

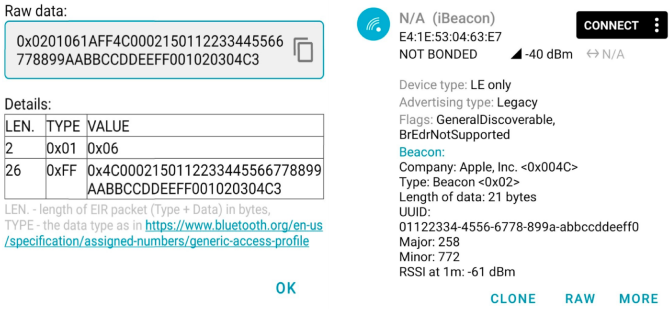


Figura 6.1: Beacon ricevuti dall'app nRF Connect: iBeacon Apple (destra), Raw Data (sinistra)

```
22          APP_MEASURED_RSSI  /* RSSI at 1 m */
23      };
24       /* Allocate packet */
25      adv_packet_t * p_packet = advertiser_packet_alloc(&
        m_advertiser, sizeof(adv_data));
26      if (p_packet)
27      {
28          /* Construct packet contents */
29          memcpy(p_packet->packet.payload, adv_data, sizeof(adv_data
        ));
30          /* Repeat forever */
31          p_packet->config.repeats = ADVERTISER_REPEAT_INFINITE;
32
33          advertiser_packet_send(&m_advertiser, p_packet);
34      }
35  }
```

Listing 6.7: Definizione della funzione *adv_start()*.

avendo definito nel *main* le *macro* come nel codice 6.1. La funzione bearer_adtype_add aggiunge il tipo BLE_GAP_AD_TYPE_COMPLETE_LOCAL_NAME alla lista di tipi di AD accettati, mentre advertiser_enable abilita l'istanza *m_advertiser*. La funzione advertiser_packet_alloc alloca un buffer dalla instanza *m_advertiser* e *advertiser_packet_send* trasmette il pacchetto p_packet usando l'instanza m_advertiser. Il risultato è mostrato in Figura 6.1

Se la *macro* APP_COMPANY_IDENTIFIER viene cambiata in 0x0059 nel campo Manufacturer ID, allora il produttore è Nordic Semiconductor.

```
1  #define APP_COMPANY_IDENTIFIER 0x59, 0x00
       /**< Company identifier for Nordic Semiconductor ASA. as per
       www.bluetooth.org. */
```

I *beacon* trasmessi con Manufacturer ID pari a 0x0059 sono mostrati in Figura 6.2.

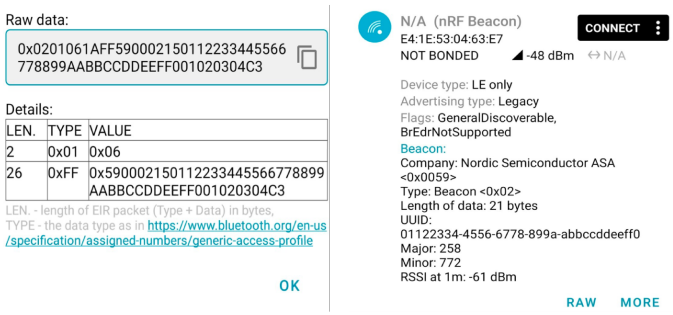


Figura 6.2: Beacon ricevuti dall'app nRF Connect: iBeacon Nordic (destra), Raw Data (sinistra)

6.1.2 Implementazione Eddystone

Per implementare il *beacon* Eddystone, secondo il suo formato *standard* URL[1] è stata modificata la variabile *adv_data* precedentemente descritta nel modo seguente:

```
1
2  #ifdef EddystoneURL
3      static const uint8_t adv_data[] =
4      {
5          APP_FLAG_BEACON_LENGTH, /* Flag Beacon length (including
       type, but not itself) */
6          BLE_GAP_AD_TYPE_FLAGS, /**< Flags for discoverability. */
7
       BLE_GAP_ADV_TYPE_EXTENDED_CONNECTABLE_NONSCANNABLE_UNDIRECTED,
       /**< Connectable non-scannable undirected advertising events
       using extended advertising PDUs. */
8          APP_SERVICE_UUID_LENGTH, /**< Server UUID advertising data
        length */
9          BLE_GAP_AD_TYPE_16BIT_SERVICE_UUID_COMPLETE,  /**<
       Complete list of 16 bit service UUIDs. */
10         APP_EDDYSTONE_UUID, /**< Eddystone UUID. */
11         APP_EDDYSTONE_DATA_LENGTH, /**< Eddystone data length. */
12         BLE_GAP_AD_TYPE_SERVICE_DATA, /**< Service Data - 16-bit
       UUID. */
13         APP_EDDYSTONE_UUID, /**< Eddystone UUID. */
14         APP_EDDYSTONE_URL_FRAME_TYPE, /**< Frame Type: Eddystone
       URL. */
15         APP_MEASURED_RSSI, /**< Ranging data. */
16         APP_URL_PREFIX, /**< URL prefix http://www. */
17         'n', /**< URL. */
18         'o',
19         'r',
20         'd',
21         'i',
```

[1] https://github.com/google/eddystone/tree/master/eddystone-url

```
        'c',
        's',
        'e',
        'm',
        'i',
        '.',
        'c',
        'o',
        'm',
        '/'
    };
#endif
```

Listing 6.8: Configurazione della variabile adv_data per il formato beacon Eddystone

avendo definito nel *main* le seguenti macro:

```
/*Defines for Eddystone packet */
#define APP_SERVICE_UUID_LENGTH         0x03
            /**< Complete list of UUID service advertising data
    length. */
#define APP_EDDYSTONE_DATA_LENGTH       0x15
            /**< Eddystone Frame Length. */
#define APP_EDDYSTONE_UUID              0xAA, 0xFE
            /**< Eddystone UUID. */
#define APP_EDDYSTONE_URL_FRAME_TYPE    0x10
            /**< Eddystone URL Frame Type. */
#define APP_URL_PREFIX                  0x00
            /**< Eddystone URL Prefix. */
```

Listing 6.9: Definizione macro per il pacchetto Eddystone.

ed avendo usato i valori nella libreria "BLE_GAP.h":

- "BLE_GAP_AD_TYPE_FLAGS" = 0x01
- "BLE_GAP_ADV_TYPE_EXTENDED_CONNECTABLE _NONSCANNABLE_UNDIRECTED" = 0x06
- "BLE_GAP_AD_TYPE_16BIT_SERVICE_UUID_COMPLETE" = 0x03
- "BLE_GAP_AD_TYPE_SERVICE_DATA" = 0x16

Per l'implementazione della trasmissione Eddystone la procedura è la stessa di quella descritta per iBeacon nella sottosezione precedente. Il risultato è mostrato in Figura 6.3

In conclusione, per trasmettere contemporaneamente sia *beacon* Eddystone che iBeacon sono state dichiarate due variabili chiamate *iBeacon* e *eddystone*, in sostituzione di *adv_data*. Sono state dunque dichiarate due istanze necessarie

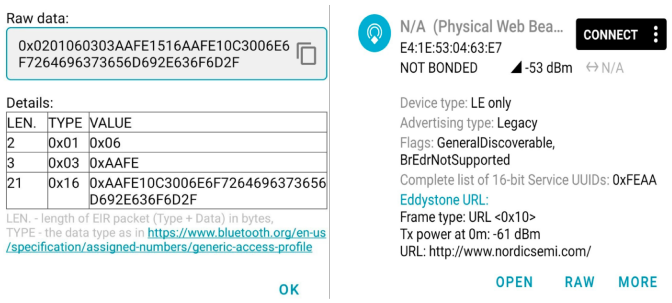


Figura 6.3: Beacon ricevuti dall'app nRF Connect: Eddystone (destra), Raw Data (sinistra)

per trasmettere i due messaggi di *advertising* consecutivamente. Nel codice seguente è riportanto solo il caso dell'istanza iBeacon per Android, ma il discorso è analogo qualora si voglia istanziare l'iBeacon Apple:

```
1 static advertiser_t m_advertiser_ibeacon_android;
2 static advertiser_t m_advertiser_eddystone;
```

Listing 6.10: Dichiarazione istanze per trasmettere i due messaggi di *advertising* consecutivamente.

ed i vettori che contengono i dati:

```
1 static uint8_t       m_adv_buffer_iBeaconAndroid[
      ADVERTISER_BUFFER_SIZE];
2 static uint8_t       m_adv_buffer_eddystone[ADVERTISER_BUFFER_SIZE
      ];
```

Listing 6.11: Vettore contenente i dati Eddystone.

Ovviamente, ne consegue che sia la funzione *adv_init()* che *adv_start* debbano essere modificate in riferimento alle nuove istanze create. Ciò che si ottiene quindi utilizzando l'app nRF Connect è osservare come entrambi i *beacon* Eddystone ed iBeacon vengano visualizzati alternativamente.

6.2 Implementazione di beacon dinamici in un applicativo mesh per il *lighting*

Come discusso precedentemente all'inizio di questo capitolo, la modalità dinamica di trasmissione dei *beacon* consente di variare a piacimento dell'utente i valori informativi trasmessi dal *beacon* anche dopo la sua configurazione iniziale. Questi valori sono l'ID piuttosto che l'URL, nel caso del *beacon* Eddystone ad esempio. Le finalità possono essere molteplici, dal proporre a un *client* l'URL del proprio sito internet, ad esempio nel caso di un *retail*, a una informazione

presente in un dabase centrale a cui uno smartphone accede, una volta ricevuto l'ID di un *beacon*.

6.2.1 Coesistenza dei servizi GATT Eddystone e PROXY GATT

Il problema principale che ci si ritrova ad affrontare quando si vuole implementare la funzionalità *beacon* in configurazione dinamica in un dispostivo operante in una rete mesh è la coesistenza tra i servizi GATT Eddystone e PROXY GATT contemporaneamente. Occorre in questo caso che il dispositivo inizializzi i suoi servizi PB-GATT per consentire ad uno smartphone di comunicare con lui al fine di portare a termine il suo *provisioning* all'interno della rete mesh. Poi il dispositivo deve essere resettato e di nuovo inizializzato per aggiungere i servizi BLE, in questo caso GATT Eddystone per la configurazione dinamica del *beacon*.

A tal fine si è partiti dal progetto *coexistence* dell'SDK della Nordic che fornisce una base di partenza per procedere all'integrazione tra un applicativo BLE con uno Bluetooth mesh. Infine, è stata utilizzata l'app nRF Connect per vedere i *beacon* trasmessi e i servizi BLE. L'app nRF Mesh consente invece di configurare i nodi nella rete mesh, solo se è presente il PB-GATT o se il *proxy* è abilitato.

6.2.2 Operazioni preliminari

In questa sezione verranno descritte alcune modifiche iniziali necessarie da apportare al progetto *coexistence*:

Il file *header* di configurazione dell'SDK (*sdk_config.h*) aiuta a gestire la configurazione statica di un'applicazione che si basa sull'SDK nRF5. Le opzioni di configurazione incluse in questo file possono essere modificate rapidamente in una procedura guidata usando l'interfaccia grafica dello strumento *Java open source CMSIS Configuration Wizard*. Ogni modulo dell'SDK contiene almeno un'opzione di configurazione che lo abilita. Se il modulo è disabilitato, anche se il codice sorgente viene aggiunto al progetto, questo non viene compilato perché il modulo non è implementato. Osservando il file *sdk_config.h*, è possibile verificare quali moduli siano utilizzati nell'applicazione:

```
1 example_module.c
2 #include " sdk_config .h"
3 #if EXAMPLE_MODULE_ENABLED
4 . . .
5 #endif //EXAMPLE_MODULE_ENABLED
```

Listing 6.12: Esempio di moduli abilitati nel file sdk_config.

Per ogni dispositivo supportato, Segger Embedded Studio e GCC ARM mettono a disposizione file di *linker* generici che contengono tutte le sezioni di memoria utilizzate dai moduli SDK. In questo caso, utilizzando Segger Embedded Studio, *config/<device_name>/ses/flash_placement.xml.*

Lo strumento CMSIS Configuration Wizard può essere facilmente integrato in Segger Embedded Studio seguendo la procedura *File -> Open Studio Folder... -> External Tools Configuration*, aggiungendo le seguenti righe di codice all'interno del file *tools.xml*, prima del *tag </tools>*:

```
<item name="Tool.CMSIS_Config_Wizard" wait="no">
    <menu>&amp;CMSIS Configuration Wizard</menu>
    <text>CMSIS Configuration Wizard</text>
    <tip>Open a configuration file in CMSIS Configuration Wizard</
    tip>
    <key>Ctrl+Y</key>
    <match>*config*.h</match>
    <message>CMSIS Config</message>
    <commands>
      java -jar &quot;$(CMSIS_CONFIG_TOOL)&quot; &quot;$(InputPath
    )&quot;
    </commands>
</item>
```

Listing 6.13: Integrazione dello strumento CMSIS Configuration Wizard in Segger Embedded Studio.

E' necessario quindi abilitare, a partire dal file *sdk_config.h* dell'esempio *coexistency*, i moduli che sono abilitati per l'Eddystone, oltre ad aggiungere alcune *directory* e *macro* del preprocessore, rispettivamente in *Options -> Prepocessor -> User Include Directories* e *Options -> Prepocessor -> Preprocessor Definitions*. Occorre infine aggiungere tutti file con estensione *.c* nelle rispettive cartelle, al fine di far coesistere il servizio Eddystone con la mesh.

6.2.3 Implementazione delle funzionalità Eddystone

In questa sezione verranno descritte le funzioni implementate che gestiscono il servizio Eddystone nel progetto. Nel file *main.c* è stato inserita la libreria *nrf_ble_es.h*

e le *macro*

```
#define NON_CONNECTABLE_ADV_LED_PIN     BSP_BOARD_LED_0     //!<
    Toggles when non-connectable advertisement is sent.
#define CONNECTED_LED_PIN               BSP_BOARD_LED_1     //!<
    Is on when device has connected.
#define CONNECTABLE_ADV_LED_PIN         BSP_BOARD_LED_2     //!<
    Is on when device is advertising connectable advertisements.
```

Listing 6.14: Macro aggiunte nel file *main.c*.

Nella funzione *bsp_event_handler* è stato aggiunta *nrf_ble_es_on_start _connectable_advertising* necessaria per mandare *connectable advertising* quando si preme il pulsante 2 (BSP_EVENT_KEY_2).

```
switch (event)
   {
        case BSP_EVENT_KEY_2:
        {
          nrf_ble_es_on_start_connectable_advertising();
        }
        break;
        default:
        break;
    }
```

Listing 6.15: Pressione del pulsante 2 per mandare *_connectable_advertising.*

Inoltre è stata implementata la funzione *on_es_evt* che gestisce gli eventi Eddystone accendendo dei LED per dare un segnale visivo sullo stato degli *advertising* trasmessi.

```

static void on_es_evt(nrf_ble_es_evt_t evt)
{
    switch (evt)
    {
        case NRF_BLE_ES_EVT_ADVERTISEMENT_SENT:
            bsp_board_led_invert(NON_CONNECTABLE_ADV_LED_PIN);
            break;

        case NRF_BLE_ES_EVT_CONNECTABLE_ADV_STARTED:
            bsp_board_led_on(CONNECTABLE_ADV_LED_PIN);
            break;

        case NRF_BLE_ES_EVT_CONNECTABLE_ADV_STOPPED:
            bsp_board_led_off(CONNECTABLE_ADV_LED_PIN);
            break;

        default:
            break;
    }
}
```

Listing 6.16: Accensione dei LED per verifica dello stato degli *advertising* trasmessi.

A questo punto, nel *main()* dopo la funzione *services_init()* è stata aggiunta la chiamata alla funzione *nrf_ble_es_init(on_es_evt)* la cui definizione è presente nel file *nrf_ble_es.c* precedentemente aggiunta.

Tutte le funzionalità Eddystone sono ora state implementate. Per trasmettere gli iBeacon, nel file *mesh_main.c* sono state aggiunte le funzioni *adv_init()* e

Softdevice	Version	Minimum RAM Start	FLASH Start
S132	7.0.0	0x20001668	0x26000

Tabella 6.1: Dimensioni RAM e FLASH Start

adv_start() per implementare e trasmettere iBeacon. Facendo il *debug* del codice si sono presentati alcuni errori. Il primo è ERROR 4 [NRF_ERROR_NO_MEM] quando il programma ritorna dalla chiamata alla funzione *nrf_ble_escs_init()* che contiene *sd_ble_uuid_vs_add()*. Quest'ultima ritorna NRF_ERROR_NO_MEM quando non ci sono slot liberi per "vendor UUID". Per risolverlo, nel file *sdk_config.h* si deve settare *#define NRF_SDH_BLE_VS_UUID_COUNT 1*.

Dopo aver modificato questa *macro*, sono state cambiate le dimensioni della *ram* dedicata al *Softdevice* come suggerito dalle informazioni di debug. La Tabella 6.1 mostra quali siano i valori da impostare.

Per cambiare le dimensioni della RAM occorre andare in *Project -> Edit Options -> Common Configuration -> Linker -> Section Placement Macros*. Nel progetto finale, composto da

- FLASH_PH_START=0x0
- FLASH_PH_SIZE=0x80000
- RAM_PH_START=0x20000000
- RAM_PH_SIZE=0x10000
- FLASH_START=0x26000
- FLASH_SIZE=0x5a000
- RAM_START=0x20002430
- RAM_SIZE=0xdbd0

sono state cambiate le macro RAM_START e RAM_SIZE precedentemente uguali a 0x20002420 e 0xdbe0 rispettivamente. Infatti per ogni nuovo UUID che si aggiunge occorre aumentare di 16 byte RAM_START e sottrarre 16 byte da RAM_SIZE, come indicato in *sd_ble_uuid_vs_add error code 4*. FLASH_SIZE non deve essere più grande di (FLASH_PH_SIZE - FLASH_START), e (RAM_SIZE + RAM_START) non deve essere più grande di (RAM_PH_START + RAM_PH_SIZE).

Infine è stato necessario risolvere alcuni errori che si sono presentati in fase di debug, perlopiù dovuti a chiamate a funzioni per doppie inizializzazioni di moduli o altre divergenze dovute al *merge* dei due applicativi. Altri file di

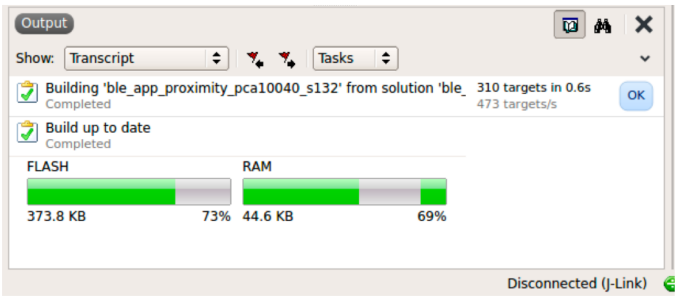


Figura 6.4: Risorse

progetto e alcune librerie inutilizzate, in relazione a funzionalità non necessarie, sono state rimosse.

Le risorse utilizzate dal programma sono 373.8 KB di memoria flash su 512 KB disponibile e 44.6 KB di memoria RAM su 64 KB disponibili come si può vedere dalla Figura 6.4

Per modificare l'intervallo di advertising per i *beacon* Eddystone è possibile cambiare le macro APP_CFG_NON_CONN_ADV_INTERVAL_MS e PP_CFG_CONN_ADV_INTERVAL_MS nella libreria es_app_config.h. Per cambiare l'intervallo di advertising degli iBeacon si può usare la macro BEARER_ADV_INT_DEFAULT_MS nella libreria nrf_mesh_config_bearer.h.

In conclusione, l'applicazione realizzata permette di inviare beacon Eddystone mentre il dispositivo partecipa alle rete Bluetooth mesh. In particolare è possibile connettere il dispositivo all'applicazione nRF *Connect* per mostrare il contenuto del *beacon* Eddystone trasmesso, modificando il suo contenuto tramite il servizio GATT Eddystone Configuration Service. Al fine di evitare conflitti tra i servizi GATT Eddystone e PROXY GATT contemporaneamente si è proceduti all'inizializzazione dei servizi PB-GATT per consentire ad uno smartphone di comunicare con il dispositivo al fine di portare a termine il suo *provisioning* all'interno della rete mesh. Dopodichè il dispositivo deve essere resettato e di nuovamente inizializzato per aggiungere il servizio BLE, in questo caso GATT Eddystone per la configurazione dinamica del *beacon*.

Capitolo 7

Studio ed analisi di segnali fotopletismografici per Wireless Body Sensor Networks

In questo capitolo verrà descritta l'implementazione di un algoritmo per l'elaborazione dei segnali PPG acquisiti durante intensa attività fisica e corrotti da MA [52].

L'innovazione tecnologica nell'ambito delle *wireless body sensor networks* ha permesso di monitorare segnali e parametri vitali in maniera continuativa, anche da remoto h24 [53, 54]. Ad esempio, il monitoraggio delle malattie cardiovascolari (CVD) è una delle principali aree di applicazione dei sensori indossabili, laddove da sempre l'elettrocardiografia (ECG) rappresenta lo *standard* di riferimento e la tecnologia più affidabile per la valutazione clinica di questi parametri [55, 56]. Negli ultimi anni è inoltre cresciuta l'attenzione rivolta allo sviluppo di tecniche di denoising dei segnali ECG acquisiti sui soggetti tramite sensori indossabili durante attività quotidiane [57].

Negli ultimi anni, l'attenzione dei ricercatori si è rivolta all'adozione della tecnica della fotopletismografia (PPG) che offre la possibilità di essere facilmente utilizzata in dispositivi indossabili e *low power* per il monitoraggio h24. La fotopletismografia è una tecnica *low cost*, non invasiva, implementata dai pulsossimetri ottici per misurare diversi parametri clinici, come l'*heart rate* (HR), la saturazione di ossigeno (SpO_2), piuttosto che valutare parametri legati alla respirazione o al monitoraggio della pressione sanguigna, e all'attività cardiaca. Infine la tecnica PPG è impiegata per valutare alcune malattie a carico del sistema cardio vascolare [58]. Ultimamente la richiesta di sensori PPG nella vita quotidiana è cresciuta moltissimo, sia in ambito ambulatoriale che nello svolgimento di attività quotidiane. Ad oggi la tecnologia PPG offre quindi una valida e ormai popolare alternativa all'ECG, anche perchè i sensori possono essere posti in vari parti del corpo, come lobi delle orecchie, dita, polsi, rendendoli idonei per un uso ambulatoriale quotidiano [59]. Ad esempio, gli orologi intelligenti o i braccialetti smart che misurano la frequenza cardiaca in

real time [60] consentono anche il monitoraggio remoto dei parametri fisiologici attraverso le tecnologie web [61].

Sebbene il monitoraggio di questi segnali a riposo non presenti particolari problemi, in alcune situazioni il monitoraggio PPG è estremamente impegnativo, a causa degli artefatti da movimento (MA) generati durante l'acquisizione sotto intensa attività fisica. Essendo tutte le parti del corpo in movimento, il contatto del sensore PPG con la pelle viene spesso alterato, specialmente sotto un intenso esercizio fisico, compromettendo pesantemente la bontà della misurazione. L'elaborazione dei segnali PPG durante l'attività fisica intensiva pone ad oggi sfide ancora aperte nel campo della ricerca, perché i MA sono difficili da rimuovere anche con le più complesse tecniche di analisi del segnale. Sebbene in letteratura siano presenti numerosi dettagli e proposte riguardanti sia lo sviluppo di sensori che le varie applicazioni basate sull'analisi PPG, la vera attenzione va rivolta al problema degli MA. La tendenza è quella quindi di adottare tecnologie a basso costo, come dispositivi indossabili, che contengano una o più unità di sensing al loro interno, unitamente l'adozione di opportune tecniche ed algoritmi di data fusion, per trovare il giusto compromesso tra affidabilità e carico computazionale in dispositivi *low power*.

Numerose tecniche per il rilevamento e la rimozione degli MA sul segnale PPG per una stima accurata dell'HR sono state proposte in letteratura nel dominio tempo - frequenza [62, 60] e con approcci basati sul machine learning [63]. Un'iniziativa in particolare, la IEEE Signal Processing Cup (SPC) database del 2015, ha promosso lo sviluppo di algoritmi per dispositivi low power da polso, rendendo disponibile un database pubblico[1] con acquisizioni sia durante la corsa che durante l'attività fisica intensa. Molte di queste tecniche sviluppate però sono poco accurate nella stima del segnale PPG o risultano inadatte per essere utilizzate in dispositivi *low power*. Altre presentano l'adozione di troppi sensori per essere utilizzati in questi tipi di dispositivi. Tuttavia, se ottenere una stima accurata dell'HR è una sfida tutt'altro che semplice in condizioni di intensa attività fisica, la corretta ricostruzione del segnale PPG nel tempo si presenta come una sfida ancora più difficile. Un'accurata ricostruzione del segnale PPG è di fondamentale importanza per ottenere una valutazione della SpO_2 e dei suoi parametri associati, tantopiù nel periodo di emergenza pandemica che stiamo tutti vivendo. In generale, recentemente in letteratura sono state introdotte alcune tecniche per stimare la saturazione di ossigeno dall'ampiezza di alcuni picchi contenuti nello spettro nel dominio della frequenza del segnale rosso e infrarosso **??**. Tuttavia questi metodi funzionano bene in presenza di artefatti da movimento minimi e quando la qualità del segnale PPG è ottima.

Lo scopo del presente lavoro è stato quello sviluppare un algoritmo per ricostruire accuratamente il segnale PPG proveniente da un dispositivo indossabile

[1] IEEE SPC, https://sites.google.com/site/researchbyzhang/ieeespcup2015.

da polso. A tal fine è stato scelto uno dei noti database della *IEEE Signal Processing Cup* su cui testare l'algoritmo proposto e validare i risultati ottenuti. Il database comprende due segnali PPG, un segnale accelerometrico tri-assiale e quello ECG di riferimento. Tutti i segnali sono stati acquisiti contemporaneamente sui soggetti. Quello che si intende proporre in questo lavoro è un algoritmo affidabile che, sfruttando tecniche di signal processing, permette la sua implementazione in dispositivi con basse richieste computazionali *low power* al fine di effettuare una stima real time dell'HR con l'obiettivo finale di effettuare un'accurata ricostruzione del segnale PPG in presenza di MA.. Il lavoro mostra la fattibilità di un algoritmo per la ricostruzione del segnale PPG durante la camminata e la corsa.

7.1 La pulsossimetria ottica

Un pulsossimetro è un dispositivo economico e di piccole dimensioni, basato sulla tecnica della pulsossimetria ottica. Questa tecnica viene utilizzata per rilevare il cambiamento del volume del sangue arterioso durante il ciclo cardiaco, attraverso la misurazione dell'aumento dell'assorbimento della luce dovuto all'aumento sistolico del volume del sangue arterioso per HbO_2 e Hb, generalmente nelle regioni del rosso e dell'infrarosso [64]. La tecnica della pulsossimetria ottica utilizzata nei moderni pulsossimetri è stata ampiamente discussa in diverse pubblicazioni [65, 66, 67]. Ad ogni modo, nel seguito ne verrà fatta una rapida trattazione per coglierne alcuni concetti fodamentali.

La Figura 7.1 mostra un esempio di una forma d'onda fotopletismografica, costituita dalla sua componente continua (DC) e quella pulsatile (AC). La componente DC della forma d'onda PPG corrisponde al segnale ottico trasmesso o riflesso rilevato dal tessuto e dipende dalla struttura del tessuto e dal volume sanguigno medio del sangue arterioso e venoso. Da notare che la componente DC varia lentamente con la respirazione. La componente AC mostra i cambiamenti nel volume sanguigno che si verificano tra la fase sistolica e quella diastolica del ciclo cardiaco; la frequenza fondamentale della componente AC dipende dalla frequenza cardiaca e si sovrappone alla componente DC. Esiste una relazione matematica definita legge di Beer-Lambert, descritta in letteratura [68], che mette infine in relazione le componenti DC e AC con la SpO_2.

Ad oggi sono disponibili sul mercato diversi tipi di pulsossimetri. Oltre ai più costosi e complessi degli ospedali, generalmente non trasportabili all'esterno, esistono, come discusso all'inizio di questo capitolo, altri dispositivi più leggeri, indossabili, per applicazioni domestiche e non solo. In riferimento a questi ultimi dispositivi, il segnale PPG può essere rilevato in vari siti anatomici, tra cui i principali sono: il dito, l'orecchio, la fronte ed il polso. Possiamo

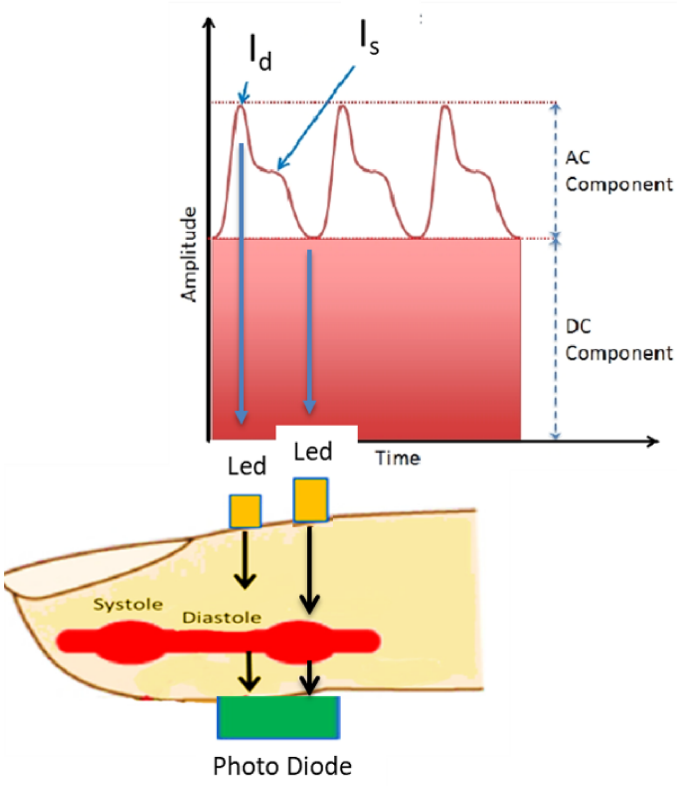


Figura 7.1: Forma d'onda PPG: componenti AC e DC. **Fonte:** [2]

quindi proporre un'ulteriore distinzione sulla base della tecnologia che questi dispositivi adottano, ovvero:

- Pulsossimetria a trasmissione: è il caso del dito e dell'orecchio, in cui la luce viene rilevata dopo essere stata trasmessa attraverso un organo ed è quindi limitata alla punta delle dita e ai lobi delle orecchie.
- Pulsossimetria riflessione: è il caso della fronte e del polso, in cui le sorgenti luminose e il fotorilevatore si trovano sulla stessa superficie della pelle, può essere applicata a qualsiasi sito accessibile.

In condizioni normali nel caso della pulsossimetria a trasmissione gli impulsi PPG hanno un elevato rapporto segnale/rumore (S/R). La pulsossimetria a riflessione può essere applicata a qualsiasi sito accessibile, ed è quindi vantaggiosa in condizioni di bassa perfusione periferica [69], a differenza di quella a trasmissione limitata nella pratica solo al dito e all'orecchio. Inoltre dalla letteratura si evince che la pulsossimetria riflessiva offre prestazioni migliori

nel riconoscimento della variazione del segnale pulsatile in termini di rapporto segnale/rumore (S/R), quindi migliore indice di perfusione [70]. In definitiva, la pulsossimetria a riflessione è più adatta di quella trasmissiva nel caso di:

- basso indice di perfusione, come esempio nel caso della vasocostrizione.
- basso flusso cardiaco
- basse temperature esterne

Occorre quindi scegliere il miglior punto di perfusione, compatibilmente con il tipo di utilizzo che verrà fatto e limitare le altre cause che possono generano artefatti

- Interferenza tra LED e fotodiodi
- Interferenza della luce ambientale
- Rumore introdotto durante l'acquisizione (e.g. dovuto ad artefatti da movimento tra l'elettrodo e la pelle.

Tutti questi fattori elencati sono quelli che influenzano maggiormente la qualità del segnale che si sta acquisendo [71]. Pertanto, l'S/R deve essere valutato attentamente in fase di test. Inoltre, l'ultimo in particolare, può essere parzialmente superato con l'implementazione di sofisticati algoritmi di analisi del segnale.

A tutti questi fattori fin qui presentati vanno aggiunti: lo scattering dei tessuti e la presenza di carbossiemoglobina che in particolare influenzano la stabilità sia del coefficiente R^2 e, più in generale, la stima della SpO_2. Questi fattori influenzano anche la stima dell'HR seppure in generale in maniera minore.

7.1.1 Il problema degli artefatti da movimento

Il tipo e l'entità degli artefatti da movimento che vengono generati sul segnale PPG dipende dal tipo di esercizio fisico, dal movimento della mano, o molte altre anomalie, compreso lo shift della lunghezza d'onda a causa dei cambiamenti nell'accoppiamento tra tessuto e sensore, la pulsazione meccanica del sangue arterioso irregolare con la frequenza cardiaca reale e l'aumento della pressione di contatto del sensore sulla pelle. Quindi, la contaminazione degli MA sul segnale PPG danno luogo a una stima inaffidabile di SpO_2 e dell'HR, rendendo queste ultime problemetiche importanti.

Le cause principali di rumore durante l'acquisizione del segnale PPG [62] sono:

[2]Coefficiente che viene calcolato a partire dalle componenti AC e DC del sangue. Il coefficiente R viene utilizzato nella legge di *Beer-Lambert*.

- Interferenze della luce ambientale
- Rumore di rete (50-60 Hz)
- Interferenze della respirazione (0.4 Hz)
- Scarsa perfusione
- Artefatti da movimento

Le prime quattro influenze possono essere più o meno facilmente limitate grazie all'adozione di tecniche di filtraggio digitali, a livello hardware e software, sia nel dominio del tempo che in quello della frequenza. I MA, invece, sono interferenze casuali a bassa frequenza, difficili da identificare perché hanno le stesse componenti spettrali di quelle del segnale utile, e non hanno una morfologia pseudo-periodica. Ciò rende difficile la loro rimozione, specialmente sotto un intenso esercizio fisico.

I metodi di filtraggio comuni rimuovono efficacemente gli MA su pazienti a riposo, ma risultano pressochè inefficaci quando il segnale PPG è acquisito durante il movimento del soggetto, poichè lo spettro di frequenza di questo rumore si sovrappone a quello del vero segnale PPG. Presenta quindi una sfida sviluppare un robusto algoritmo per dispositivi indossabili in grado di stimare adeguatamente l'HR in presenza di severi MA.

7.1.2 Tipi di segnali PPG

Esistono principalmente tre tipi di segnali PPG [72]: i cosiddetti segnali PPG buoni, cattivi e peggiori. Il primo presenta nello spettro un solo picco che rappresenta quello vero dell'HR. Tuttavia, questo tipo di segnale PPG non è così comune durante un intenso esercizio fisico come il segnale PPG cattivo. Quest'ultimo contiene più di uno picco dominante correlato sia alla vera frequenza cardiaca che agli MA e le sue armoniche. Solo con un algoritmo di tracking appropriato l'HR reale può essere rilevato sul segnale PPG rimuovendo gli MA. Infine, ancora più complessa è la stima della frequenza cardiaca nel caso del segnale PPG peggiore in cui la posizione dell'armonica del picco degli MA è molto vicina a quella del vero HR.

7.1.3 Metodo di rimozione degli MA

Al fine di rimuovere gli artefatti da movimento tecniche comunemente adottate in letteratura sono rappresentate dal *moving average filter* (MAF) e dal *multi rate filtering*, le quali si sono dimostrate utili in applicazioni limitate, dimostrandosi inefficaci contro i cambiamenti repentini nell'intensità e caratteristiche spettrali del rumore generato [73]. Il rumore in banda si verifica quando

gli spettri dell'artefatto da movimento e quello del segnale PPG si sovrappongono in modo significativo. Tuttavia, non è consigliabile utilizzare tecniche di filtraggio a frequenza fissa per eliminare artefatti da movimento dovuti all'interferenza in banda e agli spettri di frequenza sovrapposti nel segnale PPG. Tecniche come la trasformata *wavelet* [74] e *third-order independent component analysis* (ICA) [75] sono state proposte per la riduzione degli artefatti da movimento. Tuttavia, gli studi hanno indicato che le pulsazioni arteriose non sono statisticamente indipendenti dal movimento [76].

La tecnica del filtraggio adattivo

La tecnica del filtraggio adattivo è largamente utilizzata in letteratura per rimuovere i MA dai segnali biomedici e in particolare dai segnali PPG [77, 78]. Questa tecnica, in generale, si basa sulla cancellazione adattiva del rumore (ANC) [79, 80], una tecnica diffusamente utilizzata per stimare i segnali corrotti da disturbi o interferenze aggiuntive come gli MA, secondo un sistema segnale - disturbo variabile nel tempo. Come esempio pratico, molto spesso si fa uso dei segnali accelerometrici presi come rumore di riferimento, cercando di eliminare quel rumore comune sia nei dati di accelerazione che in quelli PPG, tramite tecniche di ottimizzazione basate sulla sottrazione dello spettro tramite la minimizzazione dell'errrore quadratico medio [81].

Dopo un approfondito studio teorico e pratico sui possibili algoritmi implementabili per il nostro scopo, la scelta si è rivolta a questa ultima tecnica descritta sopra. Oltretutto, l'uso di accelerometri tri-assiali ormai presenti in tutti gli orologi e smart band, consente di utilizzare l'hardware del dispositivo per generare questi segnali come ingresso di riferimento per il rumore. Il filtraggio adattivo è intrinsecamente auto-configurante attraverso l'uso di un algoritmo ricorsivo che aggiorna i parametri del filtro al variare del segnale PPG nel tempo. Questo approccio inoltre può essere utilizzato per ottenere il livello di reiezione del rumore desiderato senza una stima a priori del segnale o del rumore. Questo metodo perciò richiede due ingressi: un segnale PPG corrotto e un segnale preso come riferimento del rumore (RNS) che presenti una correlazione con il rumore presente sul primo segnale, nel nostro studio, quello accelerometrico. Il vantaggio nell'utilizzo di un metodo adattivo sta nel tempo di risposta veloce e la capacità di elaborazione continua in condizioni variabili nel tempo. I risultati sperimentali hanno dimostrato prestazioni molto affidabili e stabili nel tempo contro gli MA, come l'oscillazione delle braccia [82], rendendo questa tecnica probabilmente una delle migliori candidate per acquisizioni su soggetti che corrono o camminano sul tapis roulant, o altre attività come le immersioni subacque, per il monitoraggio dell'HR e SpO_2.

7.1.4 Dataset utilizzato

Al fine di validare l'algoritmo sviluppato, si è deciso di adottare un dataset pubblico online fornito dalla IEEE Signal Processing Cup[3], uno dei più utilizzati in letteratura per validare algoritmi sviluppati per la stima dell'HR basati su segnali PPG in presenza di artefatti di movimento [62]. Il dataset contiene due segnali PPG identici (LED verde, $\lambda = 609$ nm) generati dai due canali di un pulsossimetro, tre segnali dell'accelerometro tri-assiale (x, y, z), ed il segnale di riferimento ECG a un canale, registrato contemporaneamente dal torace di ogni soggetto. Sia i segnali PPG che quelli accelerometrici sono stati acquisiti da un dispositivo da polso. Infine, tutti i segnali, sono stati campionati a 125 Hz. I dati sono stati acquisiti da 12 soggetti maschi sani di età compresa tra 18 e 35 anni. Durante la registrazione i soggetti hanno camminato o corso su un tapis roulant a velocità variabile da 6-8 km/h a 12-15 km/h, ogni prova per una durata approssimativa di 5 minuti. Al fine di poter permettere un confronto prestazionale dell'algoritmo sviluppato con quelle di altri algoritmi analoghi presentati in passato alla IEEE Signal Processing Cup, il dataset è stato utilizzato sulla base delle seguenti regole per la rilevazione dell'*heart rate* dal segnale PPG: la frequenza cardiaca deve essere calcolata su una finestra temporale di otto secondi sovrapposta a sei secondi dalla finestra precedente. E' possibile inoltre utilizzare anche i dati precedenti fino alla finestra attuale, ma non è possibile effettuare nessuna operazione come il filtraggio della media mobile o la modifica della frequenza cardiaca di stime su valori dell'HR passati basate sulla stima dell'attuale valore dell'HR calcolato.

7.2 L'algoritmo real time implementato

Lo scopo principale del presente lavoro è stato quello di sviluppare un algoritmo in grado di ricostruire correttamente il segnale PPG acquisito durante un'intensa attività fisica. Questo è possibile solo se è disponibile una stima accurata della frequenza cardiaca. Perciò, l'algoritmo proposto è stato realizzato procendendo per fasi, ed è composto da due parti principali: la prima riguarda la stima della frequenza cardiaca mentre la seconda riguarda la ricostruzione del segnale PPG utilizzando le informazioni precedentemente fornite dell'HR stimato nella prima fase. Una semplice rappresentazione per illustrare l'intera idea alla base della sua realizzazione è mostrata in Figura 7.2.

7.2.1 Algoritmo per la stima dell'HR

La stima dell'HR è stata ottenuta processango i dati grezzi dei segnali PPG (p_i) e i dati accelerometrici (a_j(n)). a_j(n) denota i segnali affetti da MA ac-

[3] zhilinzhang.com/spcup2015/data.html

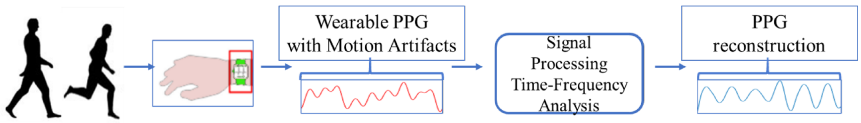


Figura 7.2: Framework dell'algoritmo proposto.

Figura 7.3: Diagramma a blocchi del metodo proposto.

quisiti dal sensore accelerometro tri-assiale, dove n = 0,1,2,... denota l'indice del campione $j = x,y,z$ identifica l'indice dell'asse. $p(n)$ può essere descritto con una formulazione matematica [83], [84], e modellato come:

$$p(n) = d_k(n) + w_k(n) \tag{7.1}$$

dove n = 0,1,2,... denota l'indice del campione, d_k e w_k rappresentano, rispettivamente, il segnale PPG non affetto da movimento (desiderato) e il segnale MA, entrambi con indici k = 1,2,3.

Per stimare l'heart rate, per ogni finestra, è stata effettuata una operazione di filtraggio passa-banda sui dati PPG e i dati accelerometrici al fine di ridurre il rumore random introdotto a causa degli elettrodi del sensore durante l'acquisizione. L'HR è stato calcolato usando la tecnica di filtraggio adattivo basata sull'uso dell'algoritmo Recursive-Least-Squares (RLS) [85, 86] e l'algoritmo Sum Slope Function (SSF) [87], quest'ultimo con uno schema a soglie adattative [88]. Entrambi gli algoritmi verranno descritti in modo più dettagliato nel seguito di questa sezione. Infine, è stata applicata una semplice tecnica di verifica del picco per calcolare l'HR appropriato al fine di prevenire una sua errata valutazione. In Figura 7.3 è mostrato un diagramma a blocchi dei passi eseguiti per stimare l'HR dai segnali PPG corrotti da movimento e descritti nel seguito di questo capitolo.

Pre-processamento

Al fine di rimuovere il rumore corrispondente a quelle frequenze fuori dal range di interesse [89], sui entrambi i due segnali PPG e i tre segnali accelerometrici è stato applicato un filtraggio passabanda, tra 0.4 Hz e 3.5 Hz. Successivamente, i due segnali PPG sono stati normalizzati sulla base della loro energia e mediati tra di loro al fine di sopprimere indesiderati rumori random [90]. In particolare, l'adozione di questa prima fase di pre-processamento consente di

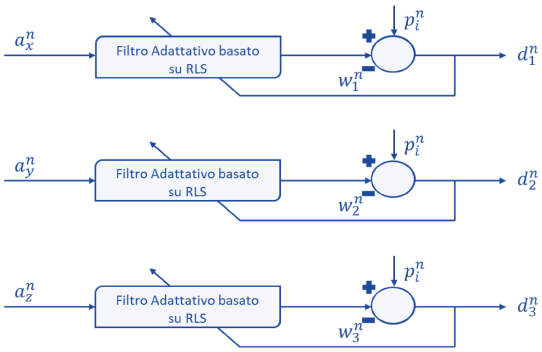


Figura 7.4: Diagramma a blocchi dello schema di filtraggio tramite RLS delle tre componenti del segnale accelerometrico a_x, a_y, a_z.

ottenere, analizzando lo spettro del segnale PPG risultante, una rilevazione più affidabile della posizione reale dell'HR [72].

Filtraggio adattivo

Dopo la fase di pre-processamento, è stato effettuata la cancellazione adattativa del rumore tramite il filtraggio RLS prendendo in input sia il segnale PPG che quello accelerometrico, quest'ultimo come riferimento del rumore generato dagli MA, secondo la procedura adottata in [72]. In questo studio si evidenzia la relazione più spiccata tra i picchi dominanti dello spettro presi sui singoli canali del segnale accelerometrico (x,y,z), con quelli associati agli MA presenti nello spettro del segnale PPG, rispetto agli stessi picchi generati dalla somma vettoriale dei tre assi accelerometrici. In generale quindi prendendo in considerazione uno solo dei canali dei segnali accelerometrici come riferimento per il rumore, questo approccio funziona bene nella maggior parte dei casi. Una sua rappresentazione è descritta in Figura 7.4, dove la stima dell'HR in generale è ottenuta processando i dati grezzi dei segnali PPG, ($p(n)$) e i dati accelerometrici, (a_j(n)). a_j(n) denota i segnali degli MA acquisiti dal sensore accelerometro tri-assiale, dove n = 0,1,2,... denota l'indice del campione e j = x,y,z identifica l'indice dell'asse. d_k e w_k rappresentano, rispettivamente, il segnale PPG non affetto da movimento che si vuole ottenere e il segnale MA, entrambi con indici k = 1,2,3. Prendendo ognuno di loro tuttavia come riferimento per gli MA, come ingresso in cascata ai filtri RLS, richiederebbe più risorse hardware e potenza computazionale.

In particolare in questo lavoro, in una prima fase, è stata calcolata la potenza di banda per le tre componenti (x,y,z) dei segnali di accelerazione [72]; si può osservare empiricamente che il segnale di accelerazione avente una banda di potenza maggiore, P_{band}, in uno specifico range di frequenze, nello specifico

compreso tra 0.5 Hz e 2 Hz, mantiene chiaramente i picchi massimi dominanti correlati con i picchi degli MA presenti nello spettro PPG. Pertanto, il segnale di accelerazione con la potenza di banda più elevata è stato scelto come segnale di riferimento per gli MA e da questo momento in poi definito segnale di riferimento MA per le successive fasi dell'algoritmo. Per ottenere i migliori risultati, la lunghezza del filtro RLS è stato empiricamente impostata su 32. Prima di procedere verso quest'ultima scelta, uteriori prove preliminari sono state sperimentate provando un filtraggio ricorsivo per ognuno dei 3 segnali accelerometrici, al variare anche del numero dei coefficienti del filtro adottato, ma con risultati non soddisfacenti. L'inverso della matrice di covarianza è stata inizializzata come $10I$, dove 10 esprime la varianza iniziale dell'autocorrelazione, scelta dettata in generale da esperienza pratica, dove questa si assume piccola rispetto alla σ^2 del segnale PPG e nel nostro caso di due ordini di grandezza inferiore rispetto alla varianza media del segnale PPG affetto da MA e di circa tre ordini di grandezza rispetto al segnale PPG senza MA. I è la matrice identità dell'ordine della lunghezza del filtro RLS. Il *weight vector* è stato inizializzato a 0. Il *forgotten factor* dell'RLS è stato impostato a 0.999.

È stato inoltre sperimentato che nessuno di questi segnali accelerometrici, presi singolarmente, ed usati come riferimento per la generazione degli MA, produce una stima sufficientemente accurata dell'HR per ciascuna finestra, dove anche il vettore somma combinato dei tre assi dell'accelerometro RNS fallisce nella maggior parte dei casi. D'altra parte, è stato osservato empiricamente che a volte, soprattutto nelle prime due finestre, quando il soggetto è considerato quasi a riposo, in presenza di un forte rumore random, la stima dell'HR dall'analisi dello spettro non è accurata. Allo stesso tempo, il tentativo di ridurre gli MA sul segnale PPG utilizzando un solo stadio di filtraggio RLS spesso fallisce. Pertanto, solo per le prime due finestre, è stato scelto di implementare un secondo stadio con un altro segnale accelerometrico, in cascata al primo, migliorando il S/R e l'accuratezza dell'HR stimata, solo se la differenza tra la P_{band} del valore maggiore rispetto a quello del secondo più grande è inferiore al 10%. Inoltre, poiché alcune finestre sono così compromesse dagli MA che solamente l'azione del filtro RLS non è sufficiente, ciò sembra dovuto sia alla debole correlazione del disturbo presente sul segnale accelerometrico e PPG che all'inizializzazione del filtro, è stata adottata un ulteriore fase di verifica per monitorare la frequenza cardiaca in finestre consecutive del segnale PPG, come verrà descritto nel seguito.

Algoritmo Sum Slope Function

Il monitoraggio della frequenza cardiaca è un altro aspetto chiave del metodo proposto. Dall'osservazione risulta che poiché il soggetto nelle prime due finestre è considerato a riposo, i picchi dell'HR, in diversi casi, possono essere

rilevati più accuratamente nel dominio del tempo rispetto ad un monitoraggio nel dominio della frequenza tramite la sua FFT, in cui la presenza di un basso S/R dovuto ad un rumore random può rendere imprecisa la stima dell'HR. È stato quindi scelto di implementare la Sum Slope Function (SSF) unitamente ad una tecnica con schema a soglia adattativa per la ricerca dei picchi [88]. Infatti, l'uso della SSF enfatizza le caratteristiche morfologiche della forma d'onda PPG reale e sopprime le altre componenti non desiderate [91]. L'SSF, al tempo i, SSF_i è definita come 7.2 -

$$SSF_i = \sum_{k=i-w}^{i} \Delta x_k \; where \; \Delta x_k = \begin{cases} \Delta s_k : \Delta s_k > 0 \\ 0 \quad : \Delta s_k \leq 0 \end{cases} \tag{7.2}$$

dove w e s_k sono, rispettivamente, la lunghezza della finestra analizzata ed il segnale PPG filtrato con il passabanda. In questo studio, è stata utilizzata la dimensione della finestra di analisi di 16 campioni, 128 ms per la frequenza di campionamento a 125 Hz. $\delta \; s_k = s_k$ - s_{k-1}, e s_k è il k-esimo campione del segnale PPG. Generalmente l'*onset* della SSF coincide approssimativamente con l'*onset* dell'impulso, ciò dipende dall'heart rate e dalla dimensione della finestra da analizzare). L'idea quindi è quella di ricercare il picco dell'impulso che appare nell'intervallo tra l'*onset* dell'SSF e l'*offset* dell'SSF. L'algoritmo quindi individua prima l'*onset* della SSF e l'*offset* della SSF ed infine determina il picco dell'impulso all'interno dell'intervallo come limite locale. Quindi viene impostata una soglia adattativa sul segnale SSF e tramite un semplice schema di condizionamento del segnale, viene impostato un intervallo di ricerca per il successivo rilevamento del picco dell'impulso tramite la ricerca del picco massimo all'interno dell'intervallo di ricerca per un rilevamento del picco dell'impulso più affidabile. Inoltre, è necessario correggere lo sfasamento temporale dei picchi ricavati sulla SSF in funzione della dimensione della finestra w. Infine, i picchi *overdetected* e quelli *skipped* vengono eliminati e restimati usando *knowledge-based rules* [92].

Tracking e verifica dell'heart rate

Solamente per le prime due finestre, è stato usato sia il metodo SSF che RLS per stimare meglio il picco HR. In particolare, in questa situazione si è scelto di applicare la SSF al segnale PPG dopo essere stato filtrato con RLS. Per le altre finestre, l'HR è stato calcolato solo sull'analisi spettrale di entrambi i segnali PPG filtrati con RLS, denominato nel seguito RLS-processato, e con il passabanda, denominato PPG pre-processato.

1. Stima dell'HR per le finestre iniziali: è necessario prestare particolare attenzione durante l'analisi delle prime due finestre. Infatti l'accuratezza nella stima dell'HR in questa fase migliora i risultati nell'accuratezza del

tracking dell'HR per le finestre successive. Per la prima finestra quindi, se la differenza tra la stima attuale dell'HR calcolata sia con il metodo RLS che con quello SSF è biù bassa di una soglia fissata, scelta empiricamente come 4 (considerando la limitata variazione di range biologica dei battiti in due finestre consecutive), ($BPM_{\text{est(SSF)}} - BPM_{\text{est(RLS)}} < 4$), l'HR per la finestra attuale, BPM_{est}, è calcolato come la media dei due valori dell'HR di cui sopra ($BPM_{\text{est}} = 0.50 BPM_{\text{est(SSF)}} + 0.50\ BPM_{\text{est(RLS)}}$). Altrimenti, se la P_{band} del segnale accelerometrico MA è più bassa di una soglia fissata P_{th}, ($P_{\text{band}} < P_{\text{th}}$), empiricamente scelta sulla base di considerazioni fatte sulla potenza di banda dei segnali di accelerazione nella banda di interesse compresa tra 0.5 Hz e 2 Hz in condizione di riposo iniziale e sotto la quale il segnale PPG è considerato libero da MA, l'HR della finestra attuale è calcolato prendendo la posizione del picco massimo dominante nello spettro del segnale RLS-processato ($BPM_{\text{est}} = BPM_{\text{est(RLS)}}$), altrimenti, prendendo quello stimato con il metodo SSF ($BPM_{\text{est}} = BPM_{\text{est(SSF)}}$). Per quanto concerne la seconda finestra, se la differenza tra l'HR stimato con entrambi i metodi è inferiore ad un valore fissato, scelto come 4, ($BPM_{\text{est(SSF)}} - BPM_{\text{est(RLS)}} < 4$), l'HR della finestra attuale è calcolata come la media dell'HR stimato con entrambi i metodi RLS e SSF, come sopra. Altrimenti, viene semplicemente mantenuta per la finestra attuale il valore dell'HR della stima precedente.

2. Selezione dei picchi: per le altre finestre, l'HR attuale è calcolato, nella maggior parte delle finestre, trovando il picco massimo dominante e la relativa posizione nello spettro del segnale RLS-processato. Tuttavia, possono presentarsi situazioni in cui il filtro RLS fallisce nel rimuovere gli MA dal segnale PPG originale, poichè gli MA sono troppo dominanti nel segnale PPG o la posizione dei picchi degli MA nel segnale PPG sono troppo vicini a quella dei picchi dell'HR per discriminarli. A tal fine, è stato settato un ristretto range di ricerca per la posizione attuale del picco dell'HR, ridotto ad uno specifico intervallo, dato che la differenza tra i battiti di finstre temporali consecutive rimangono all'interno di un limitato range (a causa della natura biologica del segnale in relazione all sovrapposizione delle finestre). L'intervallo di ricerca quindi, per il picco con il massimo valore dominante, è stato settato sperimentalmente come $[R_0{=}f_0{-}\Delta_R, \ldots, f_0 + \Delta_R]$, dove f_0 è la posizione di picco dell'HR stimato nella finestra precedente. Da questo intervallo di ricerca si ottiene la posizione della frequenza dominante più alta, f_{curr} e dalla quale BPM è

calcolato come:

$$\widehat{BPM}_{\text{est}} = \frac{f_{\text{curr}} - 1}{N_{\text{FFT}}} \times 60 \times Fs \tag{7.3}$$

dove Δ_s è settato a 6. N_{FFT} è il numero di punti della FFT utilizzati per il calcolo dello spettro, impostato a 3000 e F_s è la frequenza di campionamento. Successivamente, è necessario un secondo controllo sulla stima dell'attuale posizione del picco, prima di continuare con ulteriori fasi di verifica: solo se la distanza tra il valore dell'HR attuale stimato prendendo il valore del picco massimo sul segnale PPG pre-elaborato, $BPM_{\text{pre-processed}}$, e quello precedentemente stimato, $BPM_{\text{est(prev)}}$, è minore della distanza tra il valore dell'HR correntemente stimato sul segnale RLS-processato, $\widehat{BPM}_{\text{est}}$, e quello precedentemente stimato, e il picco in frequenza corrispondente nello spettro del segnale PPG pre-processato, $f_{\text{pre-processed}}$, non è presente nel segnale di riferimento, (f_{acc}, $|(BPM_{\text{pre-processed}} - BPM_{\text{est(prev)}}| < |\widehat{BPM}_{\text{est}} - BPM_{\text{est(prev)}}| \wedge f_{\text{pre-processed}} \neq f_{\text{acc}}$), è stato preso $BPM_{\text{pre-processed}}$ come valore attuale dell'HR, ($\widehat{BPM}_{\text{est}} = (BPM_{\text{pre-processed}}$).

Altrimenti, se la posizione stimata del picco dal segnale RLS-processato è esso stesso indentificato nel segnale di riferimento MA, e P_{band} di quest'ultimo è compreso in alcuni valori scelti sperimentalmente, P_{th1} e P_{th2}, oppure se la posizione stimata del picco attuale sul segnale RLS-processato non coincide con quella del segnale del rumore di riferimento MA e la potenza di banda di quest'ultimo è superiore alla soglia impostata (basata sul precedente valore ricavato dall'analisi della potenza di banda dei segnali di accelerazione nella banda di interesse compresa tra 0.5 Hz e 2 Hz in condizione di riposo iniziale), P_{th2}, ($f_{\text{cur}} \equiv f_{\text{acc}} \wedge P_{\max} > P_{\text{th1}} \wedge P_{\max} < P_{\text{th2}}$) $\vee$ ($f_{\text{cur}} \neq f_{\text{acc}} \wedge (P_{\max} > P_{\text{th2}})$, il picco attuale dell'HR viene scartato e preso il secondo più alto in quello del segnale RLS-processato. D'altra parte, se la potenza del segnale di riferimento MA supera un'altra soglia impostata sperimentalmente e la posizione del picco massimo sul segnale PPG pre-processato corrisponde a quella dal segnale RLS-processato, allora viene mantenuto come valido il valore del picco stimato come HR attuale. Infine, viene preso quindi come valore dell'HR attuale, in tutti gli altri casi, quello del picco massimo sul segnale RLS-processato.

Segue la verifica del picco corrente secondo le seguenti fasi: tracking e smussamento della frequenza cardiaca e, una procedura di verifica per evitare valori BPM stimati estremamente alti o bassi per evitare di perdere il monitoraggio per un lungo periodo [72].

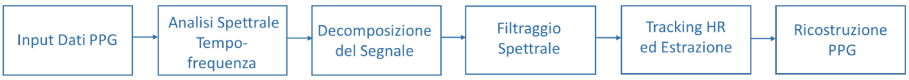


Figura 7.5: Schema a blocchi dell'algoritmo precedente per la ricostruzione del segnale PPG [3].

7.2.2 Algoritmo per la ricostruzione del segnale PPG

Al fine di ricostruire correttamente la forma d'onda del segnale PPG, è stato adottato un promettente approccio basato sull'analisi tempo-frequenza [3], basato sulla selezione ottimale delle componenti nel dominio della frequenza che si ritengono rappresentare al meglio la forma d'onda del segnale PPG e che non includano gli MA. Inoltre, per migliorare l'accuratezza nella ricostruzione del segnale PPG, per ogni finestra temporale, è stato scelto di utilizzare le informazioni sull'HR precedentemente ottenute nell'analisi della sottosezione 7.2.1. Infine, per ogni finestra temporale, è stata comparata la forma d'onda PPG ricostruita con in risalto la posizione dei picchi associati, e la forma d'onda del segnale ECG di riferimento, fornita con il dataset utilizzato.

L'algoritmo di base che è stato adottato ed implementato quindi, denominato da qui in poi per semplicità come "precedente", è costituito da 5 fasi, illustrate in Figura 7.5. Nel dettaglio, i passaggi principali saranno descritti nel seguito di questo paragrafo.

Analisi spettrale tempo-frequenza

Il segnale PPG pre-processato è dapprima sottocampionato da 125 Hz iniziale a 20 Hz, dopodichè il segnale viene demodulato in sottobande utilizzando la tecnica denominata *variable frequency complex demodulation* (VFCDM), basata sullo spettro tempo-frequenza (TFS).

Decomposizione del segnale

Il segnale PPG è quindi decomposto in 12 componenti in frequenza, in relazione alle bande in frequenza nella TFS.

Filtraggio spettrale

Partendo dalla considerazione che l'HR si trovi nel range [0.5 Hz - 3 Hz], il quale tipicamente contiene il vero valore dell'HR, considerando sia il valore più basso che quello più alto della frequenza cardiaca, viene assunto che i due picchi maggiori e le loro corrispettive frequenze nello spettro PPG possano fornire le informazione sull'HR.

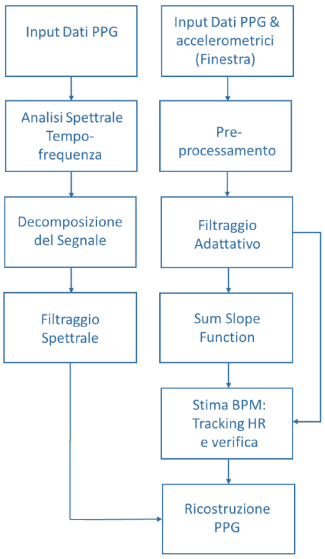


Figura 7.6: Diagramma a blocchi dell'algoritmo proposto per la ricostruzione del segnale PPG.

Tracking ed estrazione dell'HR

Se il picco più grande si trova non più distante dei 10 bpm dal valore HR precedente, viene scelto questo; in caso contrario, si procende verificando se il secondo picco più grande rientra o meno nell'intervallo dei 10 bpm. Se il valore dell'HR devia di oltre 10 bpm, viene utilizzata l'HR della finestra precedente.

Ricostruzione del segnale PPG

Il segnale PPG è quindi ricostruito, sommando le componenti della VFCDM, prendendo solo le cinque all'interno del range precedentemente citato, con le frequenze più vicine all'HR selezionato durante ogni finestra. Quindi, viene usato l'attuale BPM_{est}, stimato per ogni finestra temporale, invece di quello calcolato con l'algoritmo precedente, cercando di migliorare l'accuratezza della ricostruzione PPG (Figura 7.6).

7.3 Validazione dei risultati ottenuti

In questo lavoro, è stato implementato un algoritmo per la ricostruzione del segnale PPG utilizzando i dati dell'HR precedentemente calcolati per ciascuna finestra temporale. Al fine di valutare le prestazioni dell'algoritmo proposto per la stima della frequenza cardiaca, sono stati eseguiti dei test su segnali *standard* presi da uno dei dataset disponibili in letteratura maggiormente utilizzato per

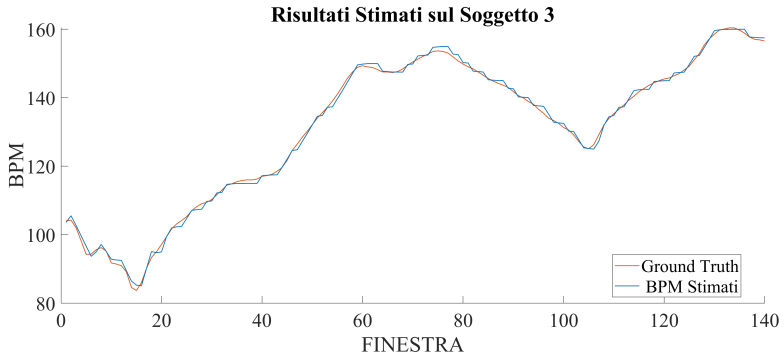


Figura 7.7: Performance del metodo proposto sul soggetto 3 (miglior soggetto).

testare la bontà degli algoritmi per segnali PPG [89] comprendente anche un segnale ECG di superficie acquisito simultaneamente con i suoi relativi valori BPM_{true} utilizzati come riferimento (HR true). Infine è stato utilizzato il ben noto algoritmo di Pan–Tompkins [4] per graficare il segnale ECG e confrontarlo con il segnale PPG ricostruito.

I risultati della stima della frequenza cardiaca sono presentati in termini di Average Absolute Error (AAE), definito come:

$$AAE = \frac{1}{N}\sum_{w=1}^{N} |BPM_{\text{est}}(w) - BPM_{\text{true}}(w)| \tag{7.4}$$

dove BPM_{est} e BPM_{true} sono i valori stimati e reali della frequenza cardiaca in BPM della finestra w-esima e N è il numero totale di finestre temporali. L'AAE e il suo valore di deviazione *standard* (σ) sono rispettivamente 1,20 e 1,09. In Figura 7.7 e Figura 7.8 vengono mostrati i risultati rispettivamente del soggetto in cui l'algoritmo ottiene, rispettivamente, le migliori e le peggiori prestazioni, rispetto ai loro valori di ground truth, calcolati dall'ECG.

Al fine di valutare le performance del metodo proposto per la stima dell'HR, sono stati presi in considerazione due indici: l'indice Pearson correlation (r) ed il Bland-Altman plot. L'indice Pearson correlation indica la misura del grado di somiglianza tra il valore vero e quello stimato dell'heart rate; nel nostro caso, rispettivamente, il valore dell'ECG e quello stimato dall'algoritmo proposto. Più alto è il valore di r, migliori sono le stime. Il Bland-Altman plot misura l'accordo tra il valore reale e la stima dell'*heart rate*. Limit of agreement (LOA) è l'intervallo che viene calcolato utilizzando la differenza media μ e la deviazione *standard* σ, che è definita come $[\mu - 1.96\sigma, \mu + 1.96\sigma]$. Il coefficiente di correlazione di Pearson, r, risulta essere 0,9964, come mostrato

[4] L'algoritmo di Pan-Tompkins è comunemente usato per rilevare complessi QRS nei segnali elettrocardiografici (ECG)

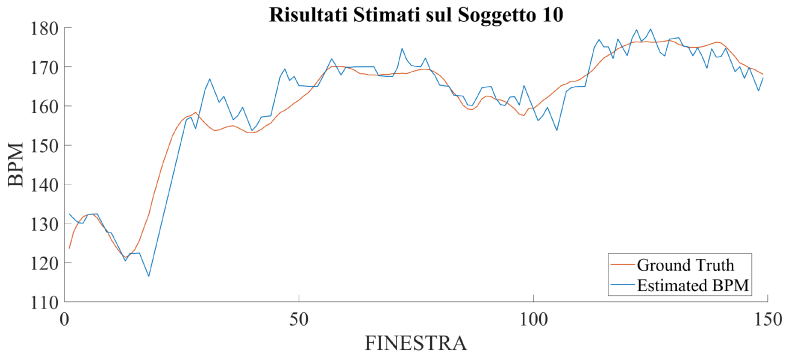


Figura 7.8: Performance del metodo proposto sul soggetto 10 (peggior soggetto)

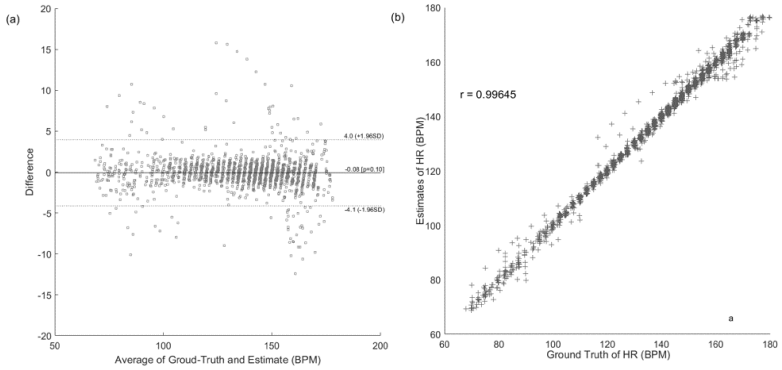


Figura 7.9: Stima dei risultati sui 12 dataset: Blant-Altman plot (a) Pearson Correlation plot (b).

nella Figura 7.9 (a sinistra). Successivamente, il Blant-Altman plot è mostrato in Figura 7.9 (a destra), usando tutti i frame di tutti e 12 i soggetti. Il LOA ottenuto è $[-4.1, 4.0]$.

Sia il valore AAE che σ stimato per l'HR con il nostro metodo discusso sono risultati significativamente inferiori su tutti i 12 soggetti (Tabella 7.1). Dalla nostra conoscenza, l'algoritmo proposto in questa tesi presenta valori di accuratezza molto simili rispetto ai migliori presenti in letteratura, e che presenta un minore potenza computazionale rispetto alla maggior parte di loro.

La Figura 7.10 mostra il confronto tra il segnale PPG pre-processato e quello ricostruito con il metodo proposto, mentre la Figura 7.11 mostra una versione ingrandita delle serie temporali HRV ottenute, rispettivamente, dal nostro metodo e da quello precedente, confrontato con il segnale ECG di riferimento acquisito contemporaneamente. Fisiologicamente, l'intensità del segnale PPG è massima alla fine della diastole, diminuendo durante la sistole, quando il

Tabella 7.1: Comparazione delle prestazioni in termini di AEE considerando tutti e 12 le acquisizioni sui soggetti.

Soggetti	Metodo precedente	Metodo proposto
Set. 1	3.65±2.74	1.39±1.67
Set. 2	4.10±2.86	1.28±1.70
Set. 3	4.59±3.31	0.73±0.54
Set. 4	3.78±2.77	1.34±1.67
Set. 5	3.97±2.95	0.82±1.07
Set. 6	3.57±2.41	1.15±1.42
Set. 7	3.70±2.96	0.88±0.99
Set. 8	3.79±2.67	0.86±0.79
Set. 9	4.54±2.82	0.73±0.50
Set. 10	3.43±2.97	3.09±0.50
Set. 11	3.82±2.20	1.42±1.70
Set. 12	4.15±2.90	0.84±0.61
Media±sd	3.92±2.80	1.20±1.09

sangue viene espulso dal ventricolo sinistro nel sistema vascolare, aumentando quindi il volume del sangue arterioso periferico, come mostrato in Figura 7.12. Quindi, da un punto di vista qualitativo, siamo in grado di stimare l'*heart rate* in maniera abbastanza accurata, laddove si possono osservare dinamiche di frequenza simili tra il riferimento ECG e le serie temporali HRV ricostruite con il nostro metodo, migliori di quelle ottenute con il metodo precedente.

Infine, l'algoritmo scritto in Matlab, versione R2019 e CPU 3.4 GHz richiede solo 80 ms per l'analisi di un segmento di 8 s per la stima dell'HR, circa 1 s per la ricostruzione completa del segnale PPG. Pertanto, questo metodo ha il potenziale per essere applicabile per l'implementazione su dispositivi indossabili come orologi intelligenti e sensori di fitness basati su PPG;

7.4 Conclusioni

Questo studio propone un algoritmo per ridurre gli artefatti da movimento sui segnali PPG, durante intensa attività fisica, in particolare durante la camminata e la corsa, al fine di calcolare l'HR e ricostruire la forma d'onda associata al segnale PPG. La principale sfida ad oggi nell'analisi di questo tipo di segnali è quella di riuscire ad ottenere una accurata stima dell'HR e della ricostruzione del segnale PPG nonostante il sovrapporsi sul segnale utile dei disturbi MA che affliggono pesantemente questo tipo di acquisizioni. Il particolare, ad oggi non sono disponibili algoritmi robusti in grado di risolvere questo tipo di inconveniente.

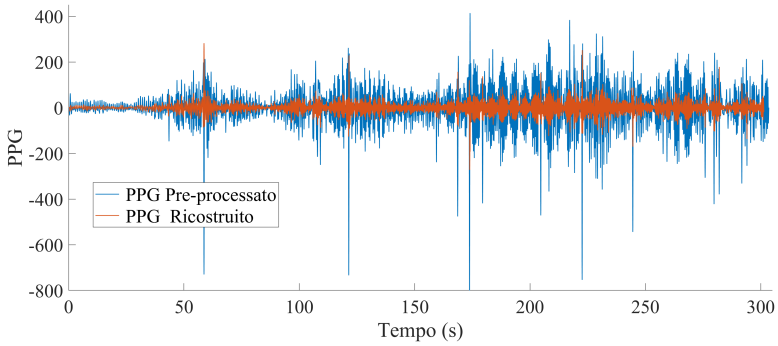


Figura 7.10: Ricostruzione segnale PPG (Dataset 1): segnale PPG precedente pre-processato e segnale PPG ricostruito con il metodo proposto.

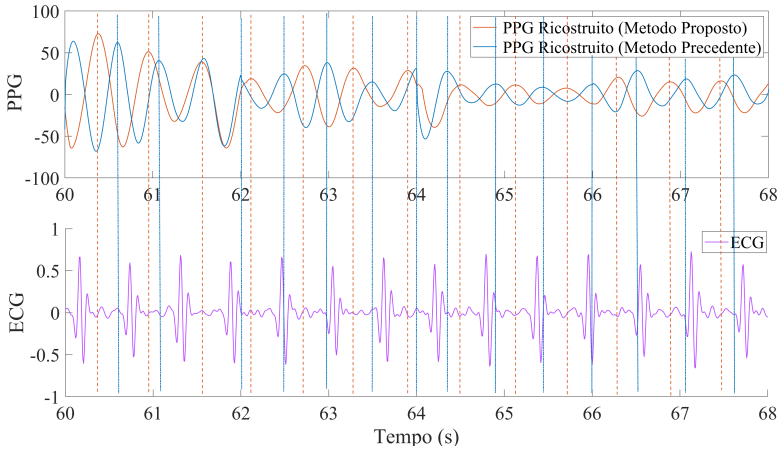


Figura 7.11: Ricostruzione segnale PPG (Dataset 1): comparazione tra il segnale PPG ricostruito con il metodo proposto e con il metodo precedente con il segnale di riferimento ECG.

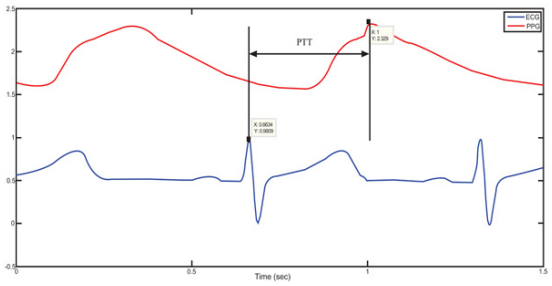


Figura 7.12: *Pulse transit time* (PTT), definito come l'intervallo di tempo tra i picchi R dell'ECG e quello sul PPG all'interno dello stesso ciclo cardiaco. **Fonte: [4].**

Per stimare l'*heart rate* è stato effettuato un pre-filtraggio iniziale dei dati PPG grezzi e dei segnali accelerometrici al fine di ridurre il rumore random introdotto a causa degli elettrodi del sensore durante l'acquisizione. Sulla base delle informazioni dei segnali accelerometrici usati come riferimento per il rumore generato dagli artefatti da movimento (MA) sul segnale PPG è stato calcolato l'HR tramite algoritmi noti di *data fusion* nel dominio tempo-frequenza unitamente a ulteriori tecniche di verifica proposte in questo lavoro per calcolare e migliorare la stima dell'HR prevenendo una sua errata valutazione. Al fine di ricostruire correttamente la forma d'onda del segnale PPG, è stato adottato un promettente approccio di analisi del segnale nel dominio tempo-frequenza che sfrutta la tecnica denominata *variable frequency complex demodulation* (VFCDM). Inoltre, per migliorare l'accuratezzo nella ricostruzione del segnale PPG si è scelto di utilizzare le informazioni sull'HR precedentemente stimato.

I risultati ottenuti, sono stati validati con l'HR proveniente dal segnale *gold standard* ECG, contenuto in un dataset che comprende inoltre segnali affetti da MA durante lo svolgimento di un'attività fisica che va dalla camminata alla corsa dei soggetti. E' stato quindi ottenuta una stima accurata dell'HR, ottenendo prestazioni simili ai migliori algoritmi presenti in letteratura, laddove nella ricostruzione del segnale PPG si osservano inoltre dinamiche di frequenza simili tra il riferimento fornito dall'ECG e le serie temporali HRV ricostruite.

7.5 Limitazioni e possibili sviluppi

Avendo lo studio proposto come obiettivo la stima dell'HR e la ricostruzione del segnale PPG nel tempo su soggetti in movimento durante attività di camminata e corsa, alcune limitazioni riguardano la mancanza di valutazioni della bontà dell'algoritmo su differenti dataset con differenti *task* o dati dei sogget-

ti. Le performance, potrebbero variare considerevolmente. Inifine un dataset con maggiore numerosità di acquisizioni potrebbe rendere più robusta la scelta degli iperparametri dell'algoritmo.

Possibili sviluppi futuri potranno comprendere l'implementazione dell'algoritmo su dispositivi indossabili tramite una sua ottimizzazione. Inoltre, potrà essere utilizzato per calcolare l'analisi della variabilità della frequenza cardiaca sui risultati, così come la ricostruzione del segnale PPG nel tempo essere d'ausilio per la stima della pressione sanguigna come supporto clinico nello studio delle malattie cardiovascolari. Inoltre, la tecnica proposta per la riduzione degli artefatti da movimento, integrata in un sistema sanitario, potrà essere utilizzata per fornire un monitoraggio sanitario continuo senza interrompere la vita quotidiana. L'ottica dell'implementazione dell'algoritmo descritto è quello di essere utilizzato per un monitoraggio continuo h24 su soggetti che svolgono attività quotidiane, piuttosto che in ambito ospedaliero, con il fine di progettare un dispositivo indossabile capace di fare *edge computing* al suo interno, e di trasmettere *in real time* all'occorrenza in maniera accurata i dati di interesse biomedico basati sulla corretta analisi del segnale PPG ricostruito, trasmettendo i dati attraverso tecnologia Bluetooth e Bluetooth mesh.

Capitolo 8

Conclusioni

L'obiettivo di questo lavoro di tesi è stato quello di intraprendere un'importante attività di studio, ricerca e ottimizzazione del recente protocollo Bluetooth mesh rivolto alle più evolute applicazioni nell'ambito delle reti di sensori wireless e dell'Internet of Things. La realizzazione della rete Bluetooth mesh per la gestione dei comandi di On/Off per gruppi di luci ha consentito di affrontare una delle maggiori sfide nel campo del lighting: la corretta ricezione dei messaggi di Status in presenza di collisioni. I test condotti sulla rete nei due scenari proposti, controllato e reale, hanno consentito di analizzare e successivamente migliorare l'affidabilità della ricezione di tali messaggi a partire da una implementazione Bluetooth mesh*standard* base. E' stata inoltre proposta, sulla base dei risultati ottenuti, un'analisi approfondita per migliorare ulteriormente la conoscenza della percentuale di comandi On/Off realmente eseguiti. Lo scopo potrà essere in futuro quello di estendere questo tipo di analisi sulla base dei test condotti anche a tutti quei dispositivi che rispondono a comandi di On/Off. L'integrazione tra un applicativo con funzionalità beacon basato sul protocollo Bluetooth Low Energy con un applicativo per il *lighting*, precedentemente sviluppato per la creazione della rete, ha permesso di ottenere un buon punto di partenza per implementare nuove funzionalità nei dispositivi operanti nella rete Bluetooth mesh al fine di creare servizi IoT veicolabili da questo tipo di rete. Infine un'ulteriore attività rientrante nel campo delle Wireless Body Sensor Networks è stata quella dell'implementazione di un algoritmo per il monitoraggio di segnali biomedici su soggetti h24. La particolarità dell'algoritmo risiede nel riuscire a rimuovere efficacemente gli artefatti da movimento presenti su segnali fotopletismografici acquisiti su soggetti in movimento durante la camminata e la corsa utilizzando tecniche di signal processing e data fusion relativamente poco dispendiose a livello computazionale. I risultati ottenuti per la stima dell'*heart rate* e la ricostruzione della curva fotopletismografica sono piuttosto incoraggianti in termini di accuratezza ottenuti. L'algoritmo proposto potrà essere ottimizzato e potenzialmente impiegato in dispositivi indossabili con basse capacità computazionali, le cui informazioni trasmesse potranno essere veicolate da una rete Bluetooth mesh.

Bibliografia

[1] D. L. Hernández-Rojas, T. M. Fernández-Caramés, P. Fraga-Lamas, and C. J. Escudero, "Design and practical evaluation of a family of lightweight protocols for heterogeneous sensing through ble beacons in iot telemetry applications," *Sensors*, vol. 18, no. 1, p. 57, 2018.

[2] C. Manlises, J. D. Cruz, J. Fausto, L. Muralla, D. Payas, and M. Posada, "Monitoring of blood pressure using photoplethysmographic (ppg) sensor with aromatherapy diffusion," in *2016 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*. IEEE, 2016, pp. 476–480.

[3] J. Harvey, S. M. Salehizadeh, Y. Mendelson, and K. H. Chon, "Oxima: A frequency-domain approach to address motion artifacts in photoplethysmograms for improved estimation of arterial oxygen saturation and pulse rate," *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 2, pp. 311–318, 2018.

[4] H. T. Ma, "A blood pressure monitoring method for stroke management," *BioMed research international*, vol. 2014, 2014.

[5] K. Ashton *et al.*, "That 'internet of things' thing," *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009.

[6] I. ITU, "Internet report–the internet of things, november 2005," 2005.

[7] B. Hammi, R. Khatoun, S. Zeadally, A. Fayad, and L. Khoukhi, "Iot technologies for smart cities," *IET Networks*, vol. 7, no. 1, pp. 1–13, 2017.

[8] Y. Mehmood, F. Ahmad, I. Yaqoob, A. Adnane, M. Imran, and S. Guizani, "Internet-of-things-based smart cities: Recent advances and challenges," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 16–24, 2017.

[9] P. Ramaswamy, "Iot smart parking system for reducing green house gas emission," in *2016 International Conference on Recent Trends in Information Technology (ICRTIT)*, 2016, pp. 1–6.

[10] P. Sadhukhan, "An iot-based e-parking system for smart cities," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2017, pp. 1062–1066.

[11] B. Son, Y.-s. Her, and J.-G. Kim, "A design and implementation of forest-fires surveillance system based on wireless sensor networks for south korea mountains," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 6, no. 9, pp. 124–130, 2006.

[12] D. Han and J. Lim, "Smart home energy management system using ieee 802.15.4 and zigbee," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 3, pp. 1403–1410, 2010.

[13] H. Jiang, C. Cai, X. Ma, Y. Yang, and J. Liu, "Smart home based on wifi sensing: A survey," *IEEE Access*, vol. 6, pp. 13 317–13 325, 2018.

[14] P. Pierleoni, L. Pernini, A. Belli, L. Palma, L. Maurizi, and S. Valenti, "Indoor localization system for aal over ipv6 wsn," in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2016, pp. 1–7.

[15] P. Pierleoni, A. Belli, A. Gentili, L. Incipini, L. Palma, S. Valenti, and S. Raggiunto, "A ehealth system for atrial fibrillation monitoring," in *Italian Forum of Ambient Assisted Living*. Springer, 2018, pp. 229–241.

[16] P. Pierleoni, A. Belli, R. Concetti, L. Palma, F. Pinti, S. Raggiunto, S. Valenti, and A. Monteriù, "A non-invasive method for biological age estimation using frailty phenotype assessment," in *Italian Forum of Ambient Assisted Living*. Springer, 2018, pp. 81–94.

[17] O. Elijah, T. A. Rahman, I. Orikumhi, C. Y. Leow, and M. N. Hindia, "An overview of internet of things (iot) and data analytics in agriculture: Benefits and challenges," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3758–3773, 2018.

[18] C. Brewster, I. Roussaki, N. Kalatzis, K. Doolin, and K. Ellis, "Iot in agriculture: Designing a europe-wide large-scale pilot," *IEEE communications magazine*, vol. 55, no. 9, pp. 26–33, 2017.

[19] "A review of wireless sensors and networks' applications in agriculture," *Computer Standards & Interfaces*, vol. 36, no. 2, pp. 263 – 270, 2014.

[20] E. Manavalan and K. Jayakrishna, "A review of internet of things (iot) embedded sustainable supply chain for industry 4.0 requirements," *Computers & Industrial Engineering*, vol. 127, pp. 925–953, 2019.

[21] A. Anzanpour, A.-M. Rahmani, P. Liljeberg, and H. Tenhunen, "Internet of things enabled in-home health monitoring system using early warning score," in *Proceedings of the 5th EAI international conference on wireless mobile communication and healthcare*, 2015, pp. 174–177.

[22] A. Alphonsa and G. Ravi, "Earthquake early warning system by iot using wireless sensor networks," in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. IEEE, 2016, pp. 1201–1205.

[23] M. Y. Mukta, M. A. Rahman, A. T. Asyhari, and M. Z. A. Bhuiyan, "Iot for energy efficient green highway lighting systems: Challenges and issues," *Journal of Network and Computer Applications*, vol. 158, p. 102575, 2020.

[24] P. Song, C. Chen, K. Li, and L. Sui, "The design and realization of embedded gateway based on wsn," in *2008 International Conference on Computer Science and Software Engineering*, vol. 4. IEEE, 2008, pp. 32–36.

[25] *Bluetooth SIG, Bluetooth specification version 4.0*, Std., 2016. [Online]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification

[26] *Bluetooth SIG, "Mesh Profile Version 1.0.1*, Std., 2019. [Online]. Available: https:www.bluetooth.com/specifications/mesh-specifications

[27] S. S. Ahmeda and E. A. Esseid, "Review of routing metrics and protocols for wireless mesh network," in *2010 Second Pacific-Asia Conference on Circuits, Communications and System*, vol. 1. IEEE, 2010, pp. 27–30.

[28] M. Kusek, "Internet of things: Today and tomorrow," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2018, pp. 0335–0338.

[29] J. Yin, Z. Yang, H. Cao, T. Liu, Z. Zhou, and C. Wu, "A survey on bluetooth 5.0 and mesh: New milestones of iot," *ACM Transactions on Sensor Networks (TOSN)*, vol. 15, no. 3, pp. 1–29, 2019.

[30] *Bluetooth SIG, "Bluetooth mesh networking: paving the way for smart lighting*, Std. [Online]. Available: https://www.bluetooth.com/bluetooth-resources/bluetooth-mesh-paving-the-way-for-smart-lighting

[31] B. Lee, S. Im, S. Lee, B. Kim, B. Roh, and Y. Ko, "The beacon identification using low pass filter for physical web based iot services," in *2015 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, 2015, pp. 354–358.

[32] Q. Wan and J. Liu, "Smart-home architecture based on bluetooth mesh technology," in *IOP Conf. Ser. Mater. Sci. Eng*, vol. 322, 2018, p. 072004.

[33] "Bluetooth sig, bluetooth mesh – what a difference a year makes. accessed: Oct. 14, 2020." [Online]. Available: https://www.bluetooth.com/blog/what-a-year-for-bluetooth-mesh/

[34] *IEEE, "IEEE 802.15 WPAN Task Group 1 (TG1)*, Std., 2019. [Online]. Available: https://grouper.ieee.org/groups/802/15/pub/TG1.html

[35] *Bluetooth SIG, "Bluetooth specification version 5.0*, Std., 2016. [Online]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification

[36] J. Lindh, "Bluetooth low energy beacons," *Texas Instruments*, p. 2, 2015.

[37] A. Developer, "Getting started with ibeacon," *Retrieved May*, vol. 10, p. 2018, 2014.

[38] *Eddystone, Eddystone Protocol Specification*, Std., 2015. [Online]. Available: https://github.com/google/eddystone/blob/master/protocol-specification.md

[39] S. M. Darroudi and C. Gomez, "Bluetooth low energy mesh networks: A survey," *Sensors*, vol. 17, no. 7, p. 1467, 2017.

[40] C. A. G. da Silva, E. L. dos Santos, A. C. K. Ferrari, and H. T. dos Santos Filho, "A study of the mesh topology in a zigbee network for home automation applications," *IEEE Latin America Transactions*, vol. 15, no. 5, pp. 935–942, 2017.

[41] W. Rzepecki and P. Ryba, "Iotsp: Thread mesh vs other widely used wireless protocols–comparison and use cases study," in *2019 7th International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE, 2019, pp. 291–295.

[42] P. Pierleoni, A. Belli, L. Palma, S. Valenti, S. Raggiunto, L. Incipini, and P. Ceregioli, "The scrovegni chapel moves into the future: An innovative internet of things solution brings new light to giotto's masterpiece," *IEEE Sensors Journal*, vol. 18, no. 18, pp. 7681–7696, 2018.

[43] D. Pérez-Díaz-De-Cerio, M. García-Lozano, A. V. Bardají, J.-L. Valenzuela *et al.*, "Bluetooth mesh analysis, issues, and challenges," *IEEE Access*, vol. 8, pp. 53 784–53 800, 2020.

[44] *Bluetooth SIG, Mesh Model Version 1.0.1"*, Std., 2019. [Online]. Available: https:www.bluetooth.com/specifications/mesh-specifications

[45] R. Rondón, A. Mahmood, S. Grimaldi, and M. Gidlund, "Understanding the performance of bluetooth mesh: Reliability, delay and scalability analysis," *IEEE Internet of Things Journal*, 2019.

[46] A. S. Brandão, M. C. Lima, C. J. B. Abbas, and L. J. G. Villalba, "An energy balanced flooding algorithm for a ble mesh network," *IEEE Access*, 2020.

[47] P. Pierleoni, A. Gentili, M. Mercuri, A. Belli, R. Garello, and L. Palma, "Performance improvement on reception confirmation messages in bluetooth mesh networks," *IoT Journal*, 2020, (under review).

[48] *Nordic Semiconductor. nRF5 SDK for Mesh*, Std. [Online]. Available: https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF5-SDK-for-Mesh

[49] E. Whitley and J. Ball, "Statistics review 3: hypothesis testing and p values," *Critical Care*, vol. 6, no. 3, pp. 1–4, 2002.

[50] *Nordic Semiconductor. DFU implementation for Mesh*, Std. [Online]. Available: https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.meshsdk.v5.0.0/md_doc_user_guide_modules_dfu_integrating_into_app.html

[51] *Nordic Semiconductor. DFU configuration for Mesh*, Std. [Online]. Available: https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.meshsdk.v5.0.0/md_doc_user_guide_modules_dfu_configuring_performing.html

[52] A. Gentili, A. Belli, L. Palma, and S. M. Egi, "A real-time algorithm for ppg signal processing during intense physical activity," *IoT Technologies for HealthCare*, p. 22.

[53] T. Rault, A. Bouabdallah, Y. Challal, and F. Marin, "A survey of energy-efficient context recognition systems using wearable sensors for healthcare applications," *Pervasive and Mobile Computing*, vol. 37, pp. 23–44, 2017.

[54] S. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, "The internet of things for health care: a comprehensive survey," *IEEE access*, vol. 3, pp. 678–708, 2015.

[55] B.-S. Lin, A. M. Wong, and K. C. Tseng, "Community-based ecg monitoring system for patients with cardiovascular diseases," *Journal of medical systems*, vol. 40, no. 4, p. 80, 2016.

[56] D. G. Benditt, W. O. Adkisson, R. Sutton, R. K. Mears, and S. Sakaguchi, "Ambulatory diagnostic ecg monitoring for syncope and collapse: An assessment of clinical practice in the united states," *Pacing and Clinical Electrophysiology*, vol. 41, no. 2, pp. 203–209, 2018.

[57] A. Nasim, F. Pinti, A. Gentili, A. Belli, L. Palma, and P. Pierleoni, "Dynamic segmented beat modulation method for denoising ecg data from wearable sensors," in *2019 International Conference on Sensing and Instrumentation in IoT Era (ISSI)*. IEEE, 2019, pp. 1–4.

[58] J. Allen, "Photoplethysmography and its application in clinical physiological measurement," *Physiological measurement*, vol. 28, no. 3, p. R1, 2007.

[59] ——, "Photoplethysmography and its application in clinical physiological measurement," *Physiological measurement*, vol. 28, no. 3, p. R1, 2007.

[60] T. Tamura, Y. Maeda, M. Sekine, and M. Yoshida, "Wearable photoplethysmographic sensors—past and present," *Electronics*, vol. 3, no. 2, pp. 282–302, 2014.

[61] P. Pierleoni, L. Pernini, L. Palma, A. Belli, S. Valenti, L. Maurizi, L. Sabbatini, and A. Marroni, "An innovative webrtc solution for e-health services," in *2016 IEEE 18th international conference on e-health networking, applications and services (Healthcom)*. IEEE, 2016, pp. 1–6.

[62] V. Periyasamy, M. Pramanik, and P. K. Ghosh, "Review on heart-rate estimation from photoplethysmography and accelerometer signals during physical exercise," *Journal of the Indian Institute of Science*, vol. 97, no. 3, pp. 313–324, 2017.

[63] E. Grisan, G. Cantisani, G. Tarroni, S. K. Yoon, and M. Rossi, "A supervised learning approach for the robust detection of heart beat in plethysmographic data," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2015, pp. 5825–5828.

[64] M. Nitzan, A. Romem, and R. Koppel, "Pulse oximetry: fundamentals and technology update," *Medical Devices (Auckland, NZ)*, vol. 7, p. 231, 2014.

[65] O. Wieben, "Light absorbance in pulse oximetry," in *Design of pulse oximeters*. CRC Press, 1997, pp. 53–68.

[66] P. D. Mannheimer, J. Cascini, M. E. Fein, and S. L. Nierlich, "Wavelength selection for low-saturation pulse oximetry," *IEEE transactions on biomedical engineering*, vol. 44, no. 3, pp. 148–158, 1997.

[67] A. Zourabian, A. M. Siegel, B. Chance, N. Ramanujam, M. E. Rode, and D. A. Boas, "Trans-abdominal monitoring of fetal arterial blood oxygenation using pulse oximetry," *Journal of biomedical optics*, vol. 5, no. 4, pp. 391–406, 2000.

[68] J. Moyle, "Pulse oximetry london," *BMJ*, 2002.

[69] Y. Mendelson and B. D. Ochs, "Noninvasive pulse oximetry utilizing skin reflectance photoplethysmography," *IEEE Transactions on Biomedical Engineering*, vol. 35, no. 10, pp. 798–805, 1988.

[70] D. Castaneda, A. Esparza, M. Ghamari, C. Soltanpur, and H. Nazeran, "A review on wearable photoplethysmography sensors and their potential future applications in health care," *International journal of biosensors & bioelectronics*, vol. 4, no. 4, p. 195, 2018.

[71] M. Nitzan, A. Romem, and R. Koppel, "Pulse oximetry: fundamentals and technology update," *Medical Devices (Auckland, NZ)*, vol. 7, p. 231, 2014.

[72] S. T. Ahamed and M. T. Islam, "An efficient method for heart rate monitoring using wrist-type photoplethysmographic signals during intensive physical exercise," in *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*. IEEE, 2016, pp. 863–868.

[73] H.-W. Lee, J.-W. Lee, W.-G. Jung, and G.-K. Lee, "The periodic moving average filter for removing motion artifacts from ppg signals," *International Journal of Control, Automation, and Systems*, vol. 5, no. 6, pp. 701–706, 2007.

[74] C. Lee and Y. T. Zhang, "Reduction of motion artifacts from photoplethysmographic recordings using a wavelet denoising approach," in *IEEE EMBS Asian-Pacific Conference on Biomedical Engineering, 2003.* IEEE, 2003, pp. 194–195.

[75] B. S. Kim and S. K. Yoo, "Motion artifact reduction in photoplethysmography using independent component analysis," *IEEE transactions on biomedical engineering*, vol. 53, no. 3, pp. 566–568, 2006.

[76] J. Yao and S. Warren, "A short study to assess the potential of independent component analysis for motion artifact separation in wearable pulse

oximeter signals," in *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*. IEEE, 2006, pp. 3585–3588.

[77] R. Yousefi, M. Nourani, S. Ostadabbas, and I. Panahi, "A motion-tolerant adaptive algorithm for wearable photoplethysmographic biosensors," *IEEE journal of biomedical and health informatics*, vol. 18, no. 2, pp. 670–681, 2013.

[78] R. Yousefi, M. Nourani, and I. Panahi, "Adaptive cancellation of motion artifact in wearable biosensors," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2012, pp. 2004–2008.

[79] M. R. Ram, K. V. Madhav, E. H. Krishna, N. R. Komalla, and K. A. Reddy, "A novel approach for motion artifact reduction in ppg signals based on as-lms adaptive filter," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 5, pp. 1445–1457, 2011.

[80] R. Yousefi, M. Nourani, S. Ostadabbas, and I. Panahi, "A motion-tolerant adaptive algorithm for wearable photoplethysmographic biosensors," *IEEE journal of biomedical and health informatics*, vol. 18, no. 2, pp. 670–681, 2013.

[81] H. Fukushima, H. Kawanaka, M. S. Bhuiyan, and K. Oguri, "Estimating heart rate using wrist-type photoplethysmography and acceleration sensor while running," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2012, pp. 2901–2904.

[82] W. Pengfei, "A new wristband wearable sensor using adaptive reduction filter to reduce motion artifact," in *Information Technology and Applications in Biomedicine, 2008. ITAB 2008. International Conference on. IEEE*, 2008.

[83] L. B. Wood and H. H. Asada, "Low variance adaptive filter for cancelling motion artifact in wearable photoplethysmogram sensor signals," in *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2007, pp. 652–655.

[84] S. Seyedtabaii and L. Seyedtabaii, "Kalman filter based adaptive reduction of motion artifact from photoplethysmographic signal," in *Proceedings of world academy of science, engineering and technology*, vol. 27, 2008.

[85] R. Yousefi, M. Nourani, S. Ostadabbas, and I. Panahi, "A motion-tolerant adaptive algorithm for wearable photoplethysmographic biosensors," *IEEE journal of biomedical and health informatics*, vol. 18, no. 2, pp. 670–681, 2013.

[86] R. Yousefi, M. Nourani, and I. Panahi, "Adaptive cancellation of motion artifact in wearable biosensors," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2012, pp. 2004–2008.

[87] S. A. Rankawat, M. Rankawat, and R. Dubey, "Heart rate estimation from non-cardiovascular signals using slope sum function and teager energy," in *2015 International Conference on Industrial Instrumentation and Control (ICIC)*. IEEE, 2015, pp. 1534–1539.

[88] M. Hahn, "An adaptive ssf-based pulse peak detection algorithm for heart rate variability analysis in home healthcare environments," in *International Conference on Ubiquitous Healthcare*. International Conference on Ubiquitous Healthcare, 2010, pp. 70–71.

[89] Z. Zhang, Z. Pi, and B. Liu, "Troika: A general framework for heart rate monitoring using wrist-type photoplethysmographic signals during intensive physical exercise," *IEEE Transactions on biomedical engineering*, vol. 62, no. 2, pp. 522–531, 2014.

[90] A. Temko, "Estimation of heart rate from photoplethysmography during physical exercise using wiener filtering and the phase vocoder," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2015, pp. 1500–1503.

[91] W. Zong, G. Moody, and R. Mark, "Reduction of false arterial blood pressure alarms using signal quality assessement and relationships between the electrocardiogram and arterial blood pressure," *Medical and Biological Engineering and Computing*, vol. 42, no. 5, pp. 698–706, 2004.

[92] D.-G. Jang, U. Farooq, S.-H. Park, and M. Hahn, "A robust method for pulse peak determination in a digital volume pulse waveform with a wandering baseline," *IEEE transactions on biomedical circuits and systems*, vol. 8, no. 5, pp. 729–737, 2014.