ALMA MATER STUDIORUM - UNIVERSITA' DI BOLOGNA
SEDE DI CESENA

SECONDA FACOLTA' DI INGEGNERIA CON SEDE A CESENA
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA
ELETTRONICA E TELECOMUNICAZIONI PER LO SVILUPPO SOSTENIBILE

IMPLEMENTAZIONE DI RETI DI SENSORI WIRELESS BASATE SUL PROFILO ZIGBEE LIGHT LINK

Tesi in
Reti di sensori wireless per monitoraggio ambientale LM

Relatore:
Chiar.mo Prof. Davide Dardari

Presentata da:
Dario Magnani

Correlatori:
Dott. Andrea Giorgetti
Ing. Vincenzo Zambianchi

Terza sessione
Anno accademico 2011/12

INDICE

INTRODUZIONE

Nella seguente trattazione si affronta lo studio e l'implementazione di reti di sensori wireless basate sul nuovo profilo ZigBee Light Link, nato per la gestione ed il controllo di punti luce.

Oggigiorno le reti wireless sono sempre più utilizzate e diffuse in quanto rappresentano potenti mezzi di controllo, apprendimento e scambio dati fornendo soluzioni semplici ed intelligenti in svariati scenari, come quello dell'illuminazione di ambienti. La loro evoluzione muove verso il concetto di "Internet of Things", ovvero l'ingresso in rete di oggetti e luoghi che compongono la realtà nella quale viviamo, offrendo la possibilità di un nuovo tipo di interazione con essa.

Quando il raggio di azione di una rete wireless, come quelle qui prese in esame, è ridotto a poche decine di metri, si parla di wireless personal area network (WPAN). Queste, coerentemente a tutte le reti di calcolatori in genere, hanno architettura data dal modello ISO/OSI. In particolare i livelli più bassi sono definiti dall'apposito standard IEEE 802.15.4, mentre livelli superiori possono essere invece sviluppati arbitrariamente. A tale riguardo è stata creata la specifica ZigBee che offre una soluzione completa di rete di scambio dati a basso costo, basso consumo per sistemi a bassa transfer rate.

Di seguito si parlerà quindi del mondo ZigBee e delle sue principali caratteristiche. Si farà poi riferimento ai profili ZigBee ed in particolare al profilo ZigBee Light Link (ZLL), che si propone come standard per i prodotti wireless dell'industria dell'illuminazione.

Questa ricerca ha come obiettivo l’analisi critica di tale profilo, eseguita avvalendosi dei risultati di test sperimentali eseguiti durante il tirocinio per tesi di laurea svoltosi presso l’azienda Aurel S.p.a. Si introdurrà quindi l’azienda stessa e il suo interesse verso questo tipo di tecnologia.

Verrà presentato poi il kit di valutazione Atmel acquistato dall’azienda e utilizzato nelle prove di laboratorio, comprendente hardware e software necessari all’implementazione di reti ZLL. Si descriveranno in seguito le modalità con le quali le prove sperimentali sono state eseguite. In fine sarà esposta la valutazione del profilo basata sulle evidenze dei risultati empirici sottolineando pregi, difetti e presentando alcune modifiche migliorative apportate a livello applicativo. In conclusione seguiranno riflessioni generali e a carattere aziendale, suggerendo possibili sviluppi futuri.

CAPITOLO 1: Il mondo ZigBee

1.1 Gli inizi

Alla fine degli anni ’90 nel campo delle telecomunicazioni si è sentita l’esigenza di realizzare reti radio digitali ad hoc con indipendenza organizzativa, bassa complessità e bassa consumo di potenza per soddisfare una domanda sempre crescente di prodotti e soluzioni aventi questi requisiti. Le esistenti tipologie di WPAN come WiFi e Bluetooth, regolamentate rispettivamente dagli standard IEEE 802.11 e 802.15.1, risultavano inadatte a tali scopi. Per questo IEEE ha creato il nuovo standard 802.15.4, reso pubblico nella sua prima versione nel maggio del 2003. In concomitanza e bastata su di esso, nasceva l’alleanza ZigBee e le relative specifiche, frutto degli studi congiunti di gruppo di note imprese. La versione 1.0 della specifica ZigBee fu completata il 14 dicembre 2004 ed divulgata a partire dal 13 giugno 2005 [1].

Market Name	ZigBee®	---	Wi-Fi™	Bluetooth™
Standard	802.15.4	GSM/GPRS CDMA/1xRTT	802.11b	802.15.1
Application Focus	Monitoring & Control	Wide Area Voice & Data	Web, Email, Video	Cable Replacement
System Resources	4KB - 32KB	16MB+	1MB+	250KB+
Battery Life (days)	100 - 1,000+	1-7	.5 - 5	1 - 7
Network Size	Unlimited (2^{64})	1	32	7
Maximum Data Rate (KB/s)	20 - 250	64 - 128+	11,000+	720
Transmission Range (meters)	1 - 100+	1,000+	1 - 100	1 - 10+
Success Metrics	Reliability, Power, Cost	Reach, Quality	Speed, Flexibility	Cost, Convenience

Figura 1: Confronto tra le principali tecnologie per WPAN **[2]**.

1.2 IEEE 802.15.4

Come sopracitato, la base di partenza della specifica ZigBee è lo standard IEEE 802.15.4 che definisce i livelli più bassi dello stack ISO/OSI, celebre modello strutturato sui sette strati costituenti l'architettura di rete. In particolare lo standard definisce il livello fisico e il livello di collegamento MAC (Media Access Control) e LCC (Logical Link Control) dettando caratteristiche e protocolli obbligatori che ogni WPAN deve adottare:

- Frequenze (bande ISM senza licenza) e numero di canali:
  - 1 canale a 868 MHz (Europa)
  - 10 canali a 915 MHz (USA, Canada, Giappone)
  - 16 canali a 2.4 GHz (Globale)
- Data rate:
  - 20 kb/s a 868 MHz
  - 40 kb/s a 915 MHz
  - 250 kb/s a 2.4 GHz
- Operatività a stella o peer-to-peer
- Supporto per dispositivi a bassa latenza
- Accesso di canale di tipo CSMA-CA
- Dispositivi ad indirizzo dinamico
- Protocolli fully handshaked per affidabilità di scambio dati
- Basso consumo di potenza

Per maggiori informazioni su protocolli, strutture dei pacchetti e tipologie di rete e dispositivi si faccia riferimento a [3].

1.3 ZigBee Alliance

Tra le società che contribuiscono allo studio e all'evoluzione di ZigBee sono presenti alcune delle più conosciute ed innovative organizzazioni del settore elettronico e delle telecomunicazioni come Philips, Samsung, Motorola, Mitsubishi e ST. Nel tempo, molte altre aziende, università, start-up ed enti pubblici e governativi hanno scelto di diventare membri della cosiddetta ZigBee Alliance. Istituita nel 2002, questa associazione aperta e non-profit lavora con lo scopo di creare e migliorare i prodotti ZigBee e di farne standard di mercato. Ad oggi l'Alliance conta decine di promotori e diverse centinaia di partecipanti e adottanti, numeri in continuo aumento a dimostrazione del fatto che soluzioni wireless di questo tipo sono sempre più richieste in diversi campi applicativi.

Figura 2: logo di ZigBee Alliance.

Come si può leggere dal sito ufficiale del gruppo [2], l'obiettivo principe di ZigBee Alliance è quello di fornire reti wireless affidabili a basso consumo energetico e basso costo che permettono l'interconnessione di dispositivi anche molto diversi tra loro dando agli utenti estremo controllo, flessibilità, connettività ed interoperabilità in qualsiasi ambiente e in ogni parte del mondo, tendendo alla visione di "Internet of Things".

1.4 Le specifiche

Ad oggi ZigBee Alliance offre due diverse specifiche: ZigBee e ZigBee RF4CE. Di interesse per questa trattazione è la prima, creata per implementare reti di sensori wireless connesse a maglia. La seconda specifica riguarda la semplice comunicazione wireless device-to-device. La più aggiornata versione della specifica di interesse è chiamata ZigBee 2012 e, a sua volta, si articola in due configurazioni: ZigBee e ZigBee PRO. In particolare quest'ultima è stata ottimizzata dalla prima per ottenere un basso consumo di potenza e sostenere grandi reti con anche centinaia di nodi ed è la più diffusa tra gli utilizzatori. Entrambe le alternative sono compatibili l'una con l'altra ed hanno le seguenti caratteristiche generali [2]:

- Frequenze e canali IEEE 802.15.4 (si veda la sezione 1.4.1)
- Soluzioni di cambio canale
- Funzioni di risparmio energetico
- Uso di ZigBee Cluster Library (si veda la sezione 1.4.4)
- Topologie di rete a stella multipla e comunicazioni inter-PAN (si veda la senzione 1.4.3)
- Trasmissioni unicast, multicast e broadcast
- Funzione di generazione di chiave di sicurezza
- Utilizzo dello standard di sicurezza AES-128
- Supporto degli standard ZigBee (si veda la sezione 1.4.3)

1.4.1 Tipologie di rete e dispositivi

La classica rete ZigBee è a maglia e composta principalmente da tre tipologie di nodo:

- Coordinator: il suo compito è quello di coordinare e controllare la formazione e la sicurezza della rete.
- Router: riceve ed inoltra dati estendendo così il raggio della rete grazie al multi-hop.
- End Device: svolge specifiche funzioni di controllo e monitoraggio.

Figura 3: Esempio di rete ZigBee che include un Coordinator, cinque Router e due End Device **[2]**.

1.4.2 ZigBee Cluster Library

Gli elementi di una rete con specifica ZigBee PRO comunicano tra loro utilizzando messaggi speciali definiti “Cluster”. Vi sono diverse tipologie di Cluster a seconda del tipo di messaggio e del tipo di applicazione. L’insieme di Cluster disponibili è descritto nel documento ZigBee Cluster Library (ZCL).

Gli sviluppatori possono creare nuovi Cluster ed espandere ZCL, oppure utilizzare Cluster già esistenti, pensati e raggruppati da ZigBee Alliance a seconda dei campi di utilizzo. Si prenda come esempio il Cluster On/Off, creato per lo scambio di messaggi ai fini di accensione e spegnimento di interruttori. L'uso di ZCL concorre quindi alla creazione di un linguaggio comune per lo sviluppo applicativo.

1.5 Applicazioni ed usi

Gli scenari d'uso in cui questi prodotti si collocano sono i più svariati, grazie alla loro versatilità. Le macro aree di impiego spaziano infatti da quella domestica alla commerciale fino all'industriale. L'Alliance si è concentrata in particolar modo nello sviluppo di standard (si veda la sezione 1.4.3) per la gestione e l'efficienza dell'energia, automazione in case ed edifici, assistenza sanitaria, fitness, telecomunicazioni ed elettronica di consumo. Esempi applicativi di successo sono riportati da ZigBee Alliance sul web [2].

Figura 4: Applicazioni ed utilizzi di ZigBee.

1.6 Evoluzione e standard

Ci si trova quindi di fronte ad una tecnologia flessibile e con un enorme numero di applicazioni possibili. Durante la sua evoluzione, data la sua versatilità, è stata naturale la scelta da parte dell'Alliance di sviluppare quelli che prendono il nome di profili o standard, differenti a seconda del tipo di impiego e concepiti per facilitare le particolari mansioni ed offrire versioni dedicate agli specifici campi di utilizzo.

Tali standard si basano sulla stessa specifica ZigBee PRO ma ognuno di questi presenta peculiarità atte a favorirne l'uso in un determinato scenario. Essi sono stati ideati in modo da comprendersi vicendevolmente appoggiandosi su ZCL ed assicurare così interoperabilità in caso di soluzioni che ne prevedono l'uso misto, tuttavia è anche possibile per gli sviluppatori programmare il proprio standard basandosi su ZigBee PRO.

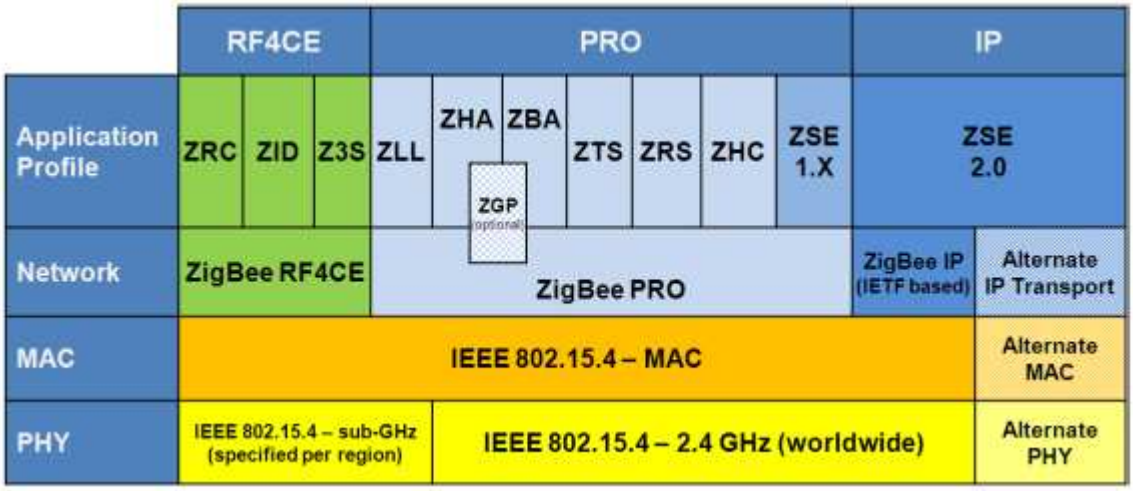


Figura 5: Lo stack ZigBee e gli standard.

La gamma di profili standard offerti da ZigBee Alliance è attualmente composta da:

ZBA: ZigBee Building Automation (Automazioni)

ZRC: ZigBee Remote Control (Telecomandi avanzati)

ZSE: ZigBee Smart Energy (Risparmio energetico domestico)

ZHC: ZigBee Health Care (monitoraggio per salute e fitness)

ZHA: ZigBee Home Automation (domotica)

ZID: ZigBee Input Device (per touchpad, tastiere, mouse)

ZLL: ZigBee Light Link (controllo dell'illuminazione)

ZRS: ZigBee Retail Services (shopping intelligente)

ZTS: ZigBee Telecom Services (servizi a valore aggiunto)

Di seguito viene riportata una significativa infographic da [2] che rappresenta e riassume l'evoluzione della specifica ZigBee dalla nascita alla creazione degli standard, l'ultimo dei quali è oggetto di discussione di questo elaborato: ZigBee Light Link.

Figura 6: Evoluzione e risultati di ZigBee dal 2002 al 2012.

CAPITOLO 2: Lo standard ZigBee Light Link

2.1 Obiettivi

Lo scopo primario di questo profilo è offrire all'industria dell'illuminazione uno standard per prodotti di consumo e controllo di estrema facilità di utilizzo per la gestione wireless di luci LED, lampade, interruttori e timer. Gli utilizzatori di una rete ZLL potranno gestire l'illuminazione dei propri ambienti a distanza, in maniera intelligente e volta al risparmio energetico e alla sostenibilità. La novità cardine e il principale punto di forza di ZLL è la semplicità di installazione e coordinamento della rete, che non necessita di nessun dispositivo speciale per lo svolgimento di questo compito. Non esistendo nessun nodo Coordinator, anche le funzioni di associazione e rimozione di nodi Router ed End Device sono del tutto rinnovate e rese più intuitive e veloci: l'utente finale dovrà interagire con semplici interruttori e dimmer wireless. Essendo uno standard ZigBee, le reti di dispositivi ZLL sono compatibili con altri prodotti che utilizzano standard come ZigBee Home Automation, ZigBee Input Device, ZigBee Remote Control e ZigBee HealthCare [2].

Figura 7: Esempio di rete ZigBee Light Link.

2.2 Caratteristiche

Di seguito vengono elencate le principali caratteristiche di ZLL riportate da ZigBee Alliance [2]:

- Facilità di installazione
  - la semplicità è tale da permettere di installare la rete in maniera “do-it-yourself”, senza appoggiarsi a installatori
  - ideale sia per case già esistenti sia per edifici in costruzione o ristrutturazione
  - La rete si auto organizza semplificando settaggi e interventi di manutenzione
- Basso costo
  - Costi di installazione e manutenzione abbattuti
  - Design di efficienza energetica e lunga vita delle batterie
  - Mercato competitivo e prezzi di prodotto al ribasso
- Facilità di utilizzo
  - Possibilità di connessione ad internet
  - Illuminazione personalizzabile su singoli punti luce o a gruppi, creazione personalizzata di scene e controllo intensità e colore dei punti luce
  - Associazione e rimozione di elementi semplice e ininfluente sulla rete
- Tecnologia ZigBee
  - Conformità alla specifica ZigBee
  - Lunga durata delle batterie che alimentano i dispositivi
  - Raggio d’azione fino a 70 metri in ambienti indoor e 400 metri outdoor
  - Flessibilità di rete per tutte le dimensioni

    - Specifica aperta e gratuitamente disponibile basata su IEEE 802.15.4
    - Soluzioni ad alta scalabilità per reti con grande numerosità
- Dispositivi supportati
    - Impianti di illuminazione
    - Lampade
    - Interruttori e dimmer
    - Telecomandi

2.3 Vantaggi

Presentando al pubblico ZLL, nel maggio 2012, ZigBee Alliance ha dedicato un webinary a questo standard [4]:

- Semplicità ed intuitività
    - Meccanismo di Touchlink (si veda la sezione 2.3.1) per aggiungere componenti e giostrare la propria rete di illuminazione.
    - Prodotti vendibili “off-the-shelf” e “do-it-yourself” (acquistabili e configurabili direttamente dall’utente finale)
    - Controllo estendibile via internet con smart phone, tablet e computer, oltre che da telecomando.
    - Dispositivi creati da diversi produttori certificati e riconoscibili con il simbolo del Light del profilo ZLL
- Sicurezza e affidabilità
    - Crittografia AES-128 a protezione di usi non autorizzati

    - L’autenticazione dei dispositivi protegge le reti dal traffico dati di reti vicine
    - Canale radio selezionabile per massimizzare performance e assicurare coesistenza con altri dispositivi wireless
- Interoperabilità
    - Dispositivi appartenenti a produttori diversi interfacciabili tra loro
    - Essendo standard ZigBee ed essendo certificato dall’Alliance, è assicurata l’interoperabilità con tutti i prodotti aventi anche essi queste due caratteristiche

2.3.1 Il Touchlink

Ciò che rende davvero interessante ZLL è la sua immediatezza nell’installazione e nella creazione (start) della rete. I meccanismi di associazione (join) e di rimozione (leave) di nodi, sono stati notevolmente semplificati rispetto agli algoritmi usati negli altri profili e in quello previsto dalla specifica ZigBee 2012. Come è già stato detto, nelle reti ZLL non esistono nodi Coordinator, ma solo Router e End Device. Lo start della rete avviene mediante una procedura che prende il nome di Touchlink. Questo metodo utilizza una comunicazione inter PAN per commissionare i messaggi necessari allo start di rete. Per questa comunicazione è stato creato da ZigBee Alliance un nuovo Cluster di ZCL, chiamato ZLL Commissioning. Un esempio di applicazione del razionale descritto dalla specifica ZLL è il seguente:

- Si considerino un dispositivo di controllo (Controller) e un dispositivo luminoso (Lighting Device)

- Avvicinando il Controller al Lighting Device che si desidera controllare ad una distanza che va dai 20 ai 50 centimetri, si preme un apposito pulsante di associazione per avviare la procedura di Touchlink
- La procedura verrà innescata automaticamente alla pressione del pulsante e i dispositivi si scambieranno messaggi con protocollo mostrato in Figura 8:
    - I due terminali si scambiano dapprima messaggi di tipo inter PAN (lo stack prende il comando del livello MAC esulando dalle normali comunicazioni ZigBee), scoprendosi attraverso un meccanismo di scansione e risposta
    - Il Controller comunica al Lighting Device che è stato selezionato per l’associazione lanciando un timer entro il quale si può abortire il processo rilasciando il pulsante. Contemporaneamente il Lighting Device mostrerà il suo interessamento attraverso un segnale luminoso.
    - Il Controller richiede al Lighting Device di descriversi e riceve risposta dal dispositivo interrogato
    - Il Controller invia i parametri di rete al Lighting Device che li utilizza per formare una rete
    - Il Controller esegue una richiesta di join in tale rete
    - La rete è formata
- Ora il Lighting Device può essere pilotato dal Controller alla distanza desiderata

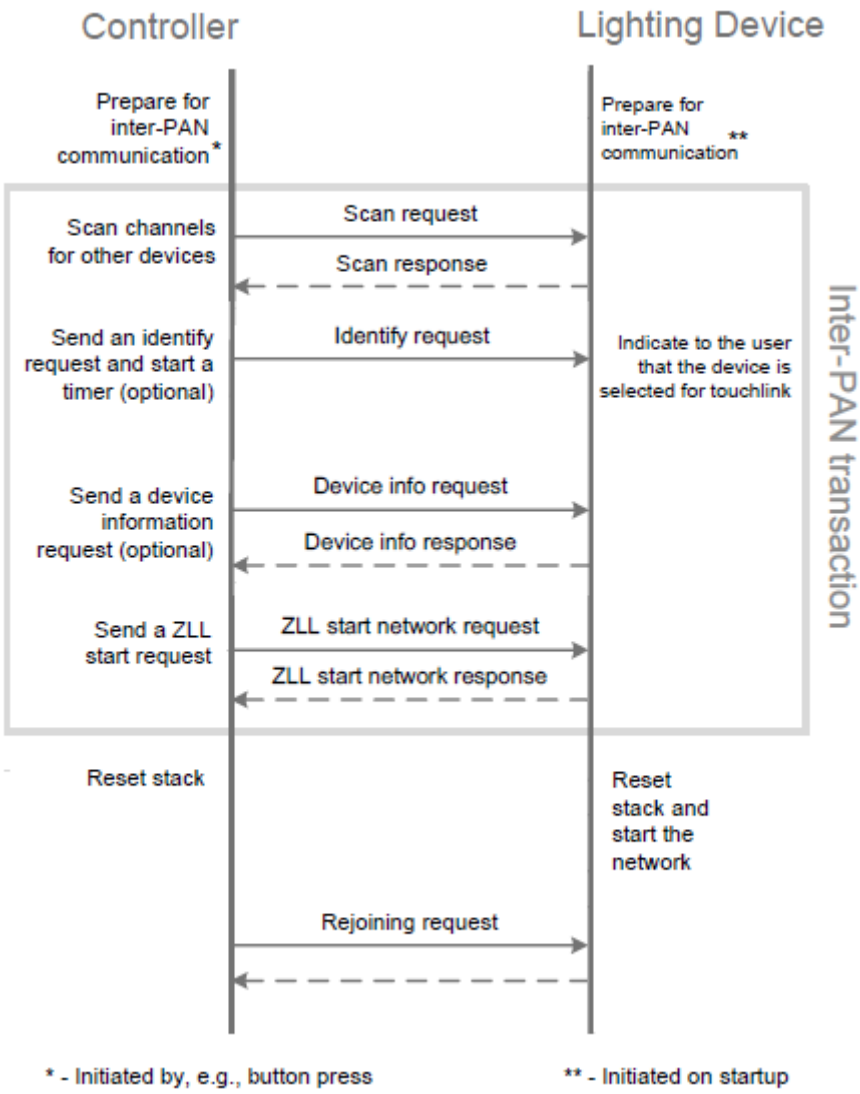


Figura 8: Protocollo Touchlink.

CAPITOLO 3: Aurel s.p.a.

3.1 L'azienda e ZigBee

Aurel s.p.a è un'azienda con sede a Modigliana (Forlì-Cesena) che nasce nel 1970 sviluppando la tecnologia del "thick film" per la realizzazione di microcircuiti elettronici. Negli anni l'azienda ha poi ampliato il suo raggio di azione e attualmente fornisce anche servizi di produzione elettronica, progetto e assemblaggio di circuiti ibridi custom, moduli wireless a radiofrequenza, circuiti elettronici per applicazioni in automotive, biomedicale, domotica e telecomunicazioni [5]. In particolare la sezione wireless del gruppo Aurel produce oltre il 60% dell'intero fatturato dell'azienda, e più del 70% dei suoi clienti si collocano nei settori home automation e industriale [6]. Nel 2008 l'azienda è entrata a far parte di ZigBee Alliance a livello "Adopter" [2] ed ha cominciato ad utilizzare questa tecnologia nell'implementazione di alcuni dei suoi prodotti e soluzioni wireless [7].

Figura 9: Logo di Aurel s.p.a.

3.2 Problematiche emerse e interessi su ZLL

L’esperienza dell’azienda nell’implementazione di moduli e reti wireless ZigBee con specifica ZigBee 2007 e in seguito ZigBee PRO ha permesso agli sviluppatori ed ai ricercatori Aurel di carpirne vantaggi e al contempo portare alla luce alcune problematiche. Alcuni di questi inconvenienti potrebbero essere risolti sfruttando i nuovi standard creati dall’Alliance (si veda la sezione 1.4.3), come ZLL. Le principali problematiche riscontrate dall’azienda riguardano i seguenti punti:

- Cambio automatico del canale radio in presenza di disturbi (non trattato in ZigBee 2007 e non del tutto risolto con ZigBee PRO)
- Pesantezza delle operazioni di installazione e monitoraggio
- Mobilità dei nodi di tipo End Device (ED) e dispendio energetico in caso di fuoriuscita degli stessi dal range di rete.
- Tempi di latenza in caso di assenza di risposta ad una interrogazione per applicazioni specifiche è desiderabile che l’utente abbia risposta immediata ad una interrogazione come ad esempio l’on/off di un punto luce

Gli ingegneri di Aurel hanno acquistato piattaforme di sviluppo certificate dall’Alliance e creato versioni aziendali di moduli ZigBee cercando di risolvere gli inconvenienti sopraelencati e muovendosi verso lo studio degli standard ZigBee PRO.

Nella fattispecie, lo standard ZLL, grazie alla semplicità di installazione e alle caratteristiche precedentemente descritte (si veda il capitolo 2), è stato preso in considerazione ed uno studio e una valutazione sono stati commissionati allo studente universitario autore della presente trattazione, attraverso un tirocinio per tesi di laurea.

CAPITOLO 4: Il kit di valutazione Atmel

4.1 RF4CE-EK

Per testare ed implementare le reti ZigBee Light Link, Aurel ha provveduto all'acquisto di un kit di valutazione Atmel, nota casa produttrice nel settore elettronico. Il prodotto è denominato RF4CE-EK. Nonostante questa dicitura possa trarre in inganno, esso fornisce una completa piattaforma hardware-software sia per lo stack ZigBee RF4CE sia per quello ZigBee PRO ed, in particolare, è munito degli standard ZRC e ZLL [8]. Il kit contiene:

- 2x Atmel RCB128RFA1 Radio Controller Board (RCB)
- 1x Atmel RCB Sensor Terminal Board (STB)
- 1x Atmel RCB Key Remote Control Board (K_RC)
- 1x Cavo RCB-BB RS232
- 1x Cavo USB
- 2x Antenne a 2.4 GHz
- 4x Batterie AAA

Figura 10: Kit di valutazione Atmel RF4CE.

4.1.1 Radio Controller Board

Questo piccolo dispositivo incarna il vero e proprio nodo ZigBee. Ha le seguenti caratteristiche:

- Transceiver single-chip con antenna a 2.4 GHz
- Microcontrollore AVR ATmega128RFA1 a bordo
- Alimentato a batterie (2x AAA)
- Munito di connettore standard 2x30 pin
- Indicatori LED
- Funge da nodo ZLL
    - Lighting Device
        - Stand-alone (effetto visivo: led a bordo)
        - Montato su Key Remote Control Board (effetto visivo: display LCD)
    - Controller
        - Montato su Key Remote Control Board

Figura 11: Atmel RCB128RFA1, vista frontale e vista dorsale.

4.1.2 Key Remote Control Board

Questo telecomando avanzato presenta una ricca interfaccia utente per gestire il controllo remoto di punti luce intelligenti:

- Tastiera e pulsanti vari
- Display LCD
- Indicatori LED
- Connettore JTAG
- Connettore standard 2x30 pin
- Ospita una RCB insieme al quale funge da nodo ZLL
  - Controller
  - Lighting Device (effetto visivo: display LCD)

Figura 12: Key Remote Control Board + RCB128RFA1.

4.1.3 Sensor Terminal Board

Questa scheda è usata per programmare il nodo RCB, aggiornare il firmware al suo interno, eseguire sessioni di debug e connetterlo al mondo esterno:

- Interfaccia per RCB
  - Connettore standard 2x30 pin
  - Connettore JTAG per la programmazione di RCB
  - Altri connettori

Figura 13: Sensor Terminal Board + RCB.

Per maggiori informazioni su questo hardware, riferirsi alla documentazione Atmel [8]. Grazie alla disponibilità da parte dell'azienda di due kit completi RF4CE, si sono potute svolgere prove comprendenti quattro RCB e due K_RC per la valutazione dello standard ZLL.

4.2 BitCloud ZLL software developement kit

Con BitCloud si intende la realizzazione dello stack ZigBee da parte di Atmel e certificato da ZigBee Alliance, comprendente il software necessario alla gestione dello stesso (API, Application Programming Interface). Insieme a RF4CE_EK esso costituisce una piattaforma di sviluppo per applicazioni utilizzanti ZigBee PRO e standard ZLL. Segue l'architettura dello stack BitCloud.

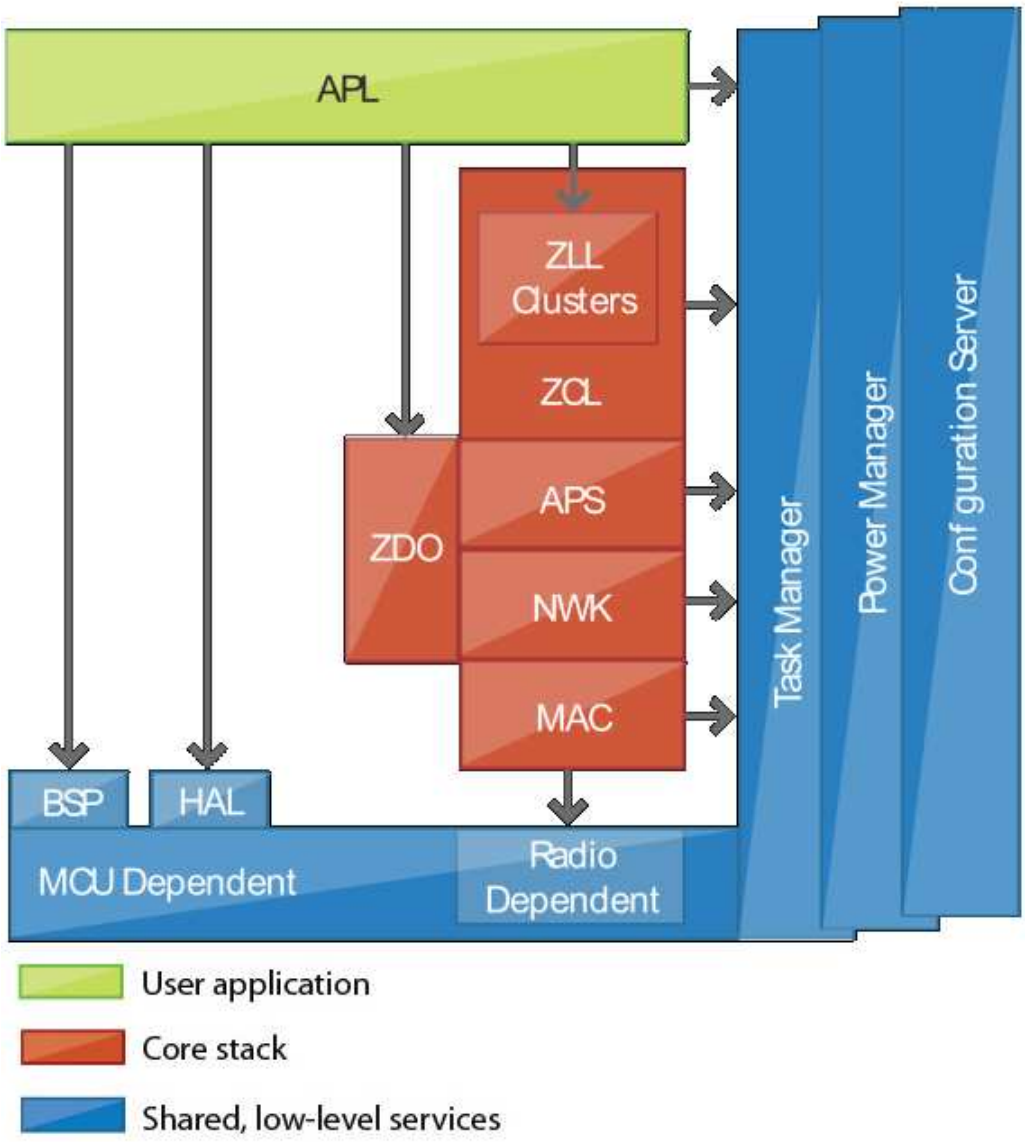


Figura 14: Architettura dello stack BitCloud per zigBee PRO.

Si noti che il livello più alto è quello sviluppabile dall'utente, attraverso un ambiente di programmazione che fa uso del linguaggio C/C++. Il BitCloud è contenuto in quello che prende il nome di BitCloud ZigBee Light Link software developement kit (SDK), un pacchetto software per lo sviluppo di applicazioni ZLL, scaricabile gratuitamente da [8]. Esso contiene:

- BitCloud ZigBee PRO
  - File header (con estensione .h) e sorgenti (con estensione .c) di componenti e driver necessari allo sviluppo di applicazioni in questo ambito
  - ZigBee Cluster Library, comprendenti anche
    - Cluster ZLL
    - API ZLL
  - Librerie compilate BitCloud
- Un'applicazione dimostrativa dello standard ZLL (ZLLDemo)
  - File di progetto IAR
  - File sorgenti (con estensione .c)
- Una serie di file immagini firmware precompilate dell'applicazione dimostrativa (con estensione .hex)
- Documentazione del kit di valutazione Atmel in formato pdf.

CAPITOLO 5: Strumenti di progetto e test

5.1 IAR Embedded Workbench

Per programmare le RCB del kit Atmel in modo da far svolgere loro il compito di nodi di rete ZLL, è necessario un firmware dedicato che contenga sorgenti, header e librerie da trasferire sul microcontrollore di ciascun dispositivo. A questo scopo si è utilizzato il codice contenuto nell'applicazione dimostrativa del SDK. Per poter visualizzare e sviluppare tale firmware è necessario l'utilizzo di uno specifico ambiente di sviluppo integrato (in inglese, IDE: Integrated Development Environment). La piattaforma Atmel descritta nel capitolo precedente richiede ambiente di programmazione IAR Embedded Workbench per Atmel AVR versione 6.11 o superiori, con compilatore IAR C/C++ per AVR versione 6.11.1.50453, che si può scaricare con licenza limitata dal sito di IAR [9].

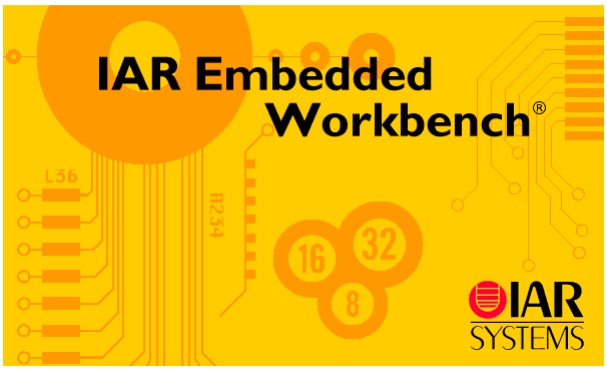


Figura 15: Logo di IAR Embedded Workbench e di IAR systems.

Installando su PC i contenuti del pacchetto SDK e l'IDE sopraindicato si avrà tutto il necessario per la gestione del software di reti ZLL.

5.1.1 Organizzazione del codice sorgente di ZLLDemo

Per workspace si intende il file con estensione .eww (Embedded Workbench Workspace) contenente i file header e i codici sorgenti necessari all'applicazione dimostrativa dello standard. Esso è denominato *ZLLdemo.eww* e si trova al percorso Application/ZLLDemo del pacchetto SDK. Il codice sorgente è diviso in codice specifico per i Lighting Device (qui chiamati Color Light), Controller (Color Scene Remote) e codice per componenti e tool in comune.

Le funzioni di accesso *main()* si trovano nei file *colorlight.c* e *colorsceneremote.c,* nelle omonime cartelle. Nel workspace sono presenti inoltre i file sorgenti dei Cluster supportati dai dispositivi. In questi, oltre ad essere definiti i Cluster, vengono inizializzate le strutture dati necessarie e definite le funzioni di callback, ovvero quelle funzioni chiamate a risposta dei comandi inviati attraverso i Cluster stessi. Nel file *configuration.h* (si veda l'appendice) in si trovano le configurazioni dei dispositivi chiamati Color Light e Color Scene Remote. La scelta del tipo di dispositivo per il quale si vuole compilare il firmware viene fatta attraverso i seguenti punti

- La definizione nel file *configuration.h*
  - *#define APP_DEVICE_TYPE_COLOR_LIGHT*
  - *#define APP_DEVICE_TYPE_COLOR_SCENE_REMOTE*
- La scelta dei file di progetto
  - Con prefisso *All_* per qualsiasi tipo di dispositivo
  - Con prefisso *Router_* per i Lighting Device
  - Con prefisso *EndDevice_* per i Controller

Segue uno screenshot di IAR Embedded Workbench dove è evidenziato il fatto che si sta lavorando ad un Lighting Device.

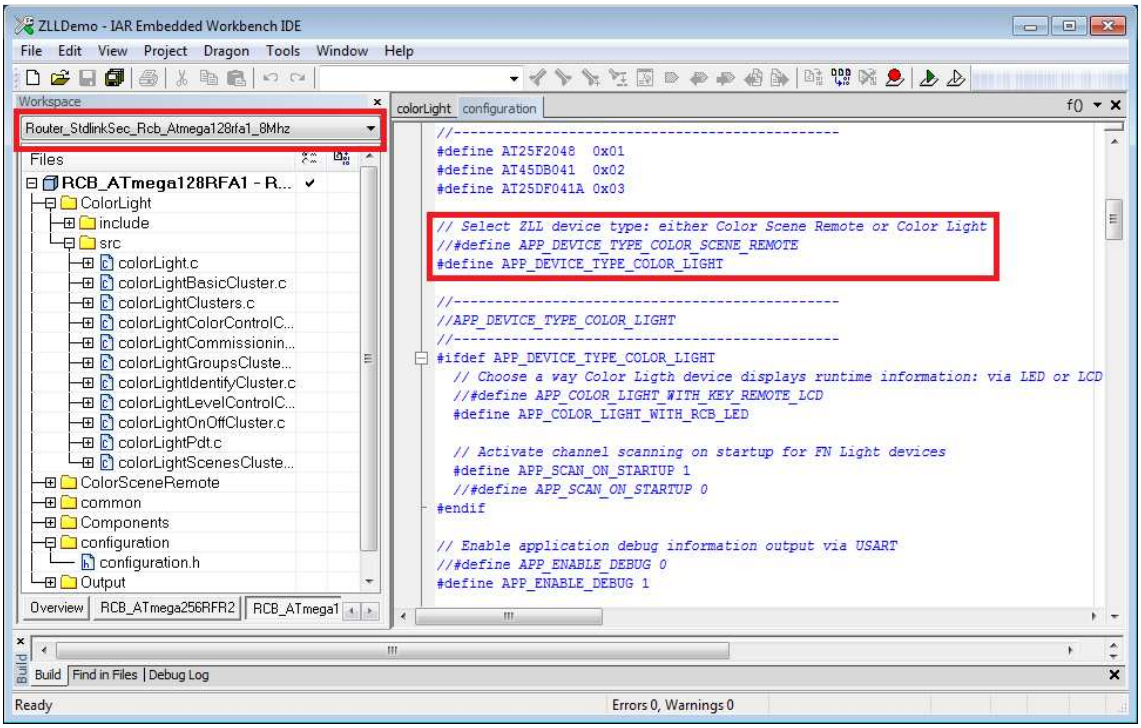


Figura 16: IAR Embedded Workbech, considerando un Lighting Device.

5.2 AVR Dragon

È necessario un programmatore che interfacci il computer sul quale si torva il firmware al nodo sul quale lo si vuole trasferire. Si è utilizzato il programmatore AVR Dragon, compatibile con il kit e già in possesso dell'azienda.

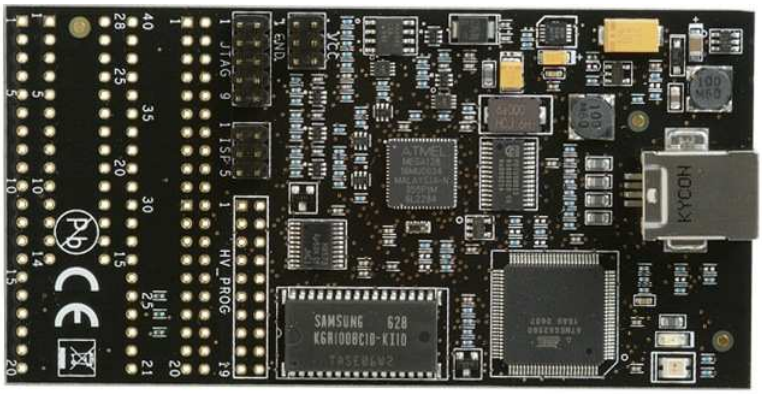


Figura 17: Programmatore AVR Dragon, vista dall'alto.

Esso si collega al PC tramite porta USB e, attraverso porta JTAG, alla scheda STB o indifferentemente alla K_RC, sulle quali viene montata la scheda RCB, dove risiede il microcontrollore da programmare. Affinché il trasferimento del firmware avvenga con successo è necessario che AVR Dragon sia quindi correttamente connesso con PC ed RCB. Si noti che quest'ultima scheda deve essere alimentata e accesa attraverso lo switch On/Off laterale.

5.3 Sniffer CC2531 USB dongle

Per monitorare i pacchetti che i nodi di una rete wireless si scambiano, viene utilizzato un apposito strumento chiamato Sniffer. Questo oggetto è anche esso un nodo ed composto da una semplice antenna unicamente ricevente e poca altra circuiteria annessa, si connette al PC tipicamente attraverso porta USB e si appoggia a software dedicati, a seconda del tipo di comunicazione RF che si vuole catturare. Esso permette di visualizzare a video la successione dei pacchetti di informazione captati rendendo possibile lo studio dell'effettiva attività radio che avviene nella rete, identificando le varie tipologie di pacchetto ed i campi da cui sono composti, come ad esempio il campo in cui risiede l'indirizzo del nodo sorgente e quello in cui si trova l'indirizzo del destinatario. Lo Sniffer posseduto da Aurel s.p.a. e utilizzato nei test sperimentali eseguiti sulle reti ZLL è denominato CC2531 USB dongle, prodotto da Texas Instruments.

Figura 18: CC2531 USB donlge, Texas Instruments.

Il software a cui si affianca è chiamato SmartRF Packet Sniffer, scaricabile dal sito della casa produttrice [10]. Esso dà la possibilità di selezionare in prima battuta il tipo di protocollo da monitorare. Ai fini di questa trattazione la voce desiderata è IEEE 802.15.4/ZigBee.

Segue uno screenshot della finestra principale di dialogo di SmartRF Packet Sniffer.

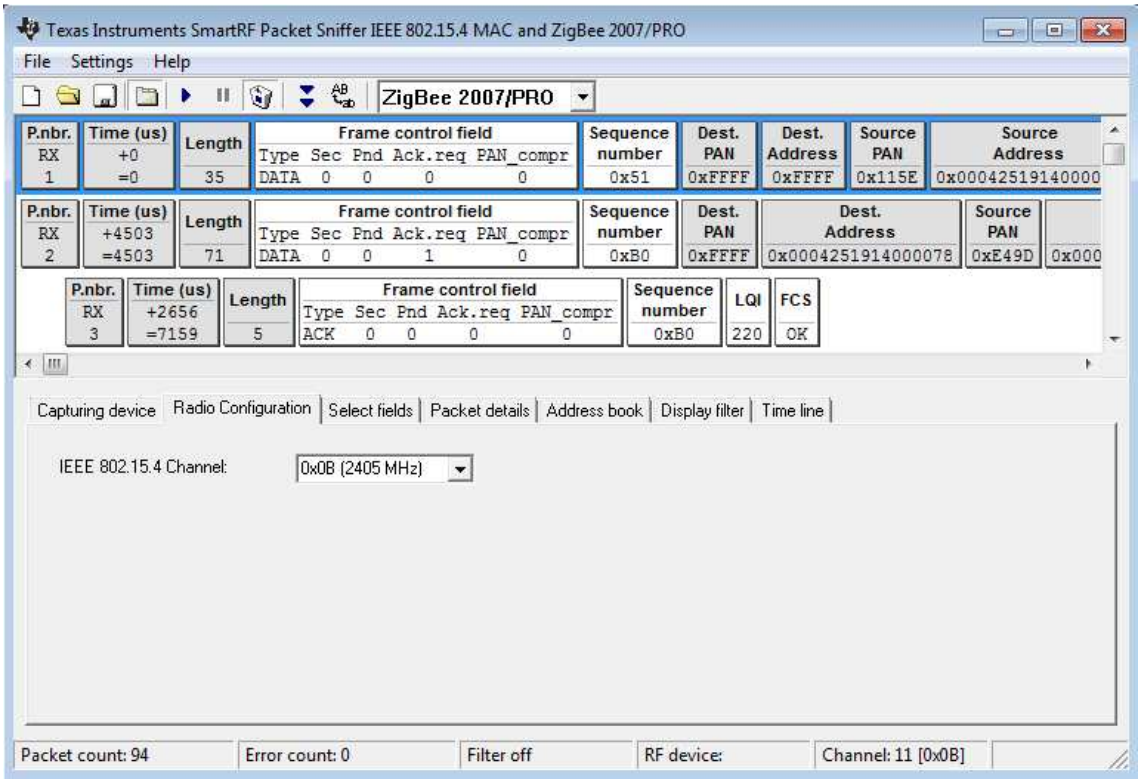


Figura 19: Screenshot SmartRF Packet Sniffer.

Si può operare un’ulteriore selezione scegliendo il tipo di specifica ZigBee voluta (ZigBee 2007/PRO). Grazie alla barra degli strumenti superiore si può poi controllare la cattura dei pacchetti con i pulsanti play, pausa, eccetera. Nei campi sottostanti alla finestra di visualizzazione dei pacchetti ricevuti ci sono poi altre opzioni come quella della scelta del canale radio sul quale porre il dispositivo in ascolto, scelto tra quelli previsti dallo standard IEEE 802.15.4. Per maggiori dettagli riguardo a questo strumento si rimanda alla documentazione rintracciabile attraverso la bassa dei menù alla voce Help.

5.4 Multimetro

Durante gli esperimenti con il kit fino ad ora descritto si è inoltre fatto uso della strumentazione da laboratorio presente in azienda, in particolare di un multimetro da banco, per misure di corrente nei nodi durante le loro varie attività, per verificarne il consumo raffrontandolo con i dati riportati nei manuali Atmel.

5.5 Schemi a blocchi

5.5.1 Programmazione e debug di un nodo ZLL

Seguono le figure che rappresentano la catena degli strumenti necessaria al trasferimento del firmware sulle RCB e il debug di tale firmware. Si noti che la RCB, per essere programmata, può essere montata indifferentemente su STB o K_RC. Per eseguire anche il debug del firmware invece, si è vincolati al tipo di dispositivo che si sta programmando. Se si vuole programmare e debuggare una Color Light stand-alone, è valido lo schema a blocchi di Figura 20. Se invece si sta lavorando con una Color Light con display LCD o un Color Scene Controller, è applicato lo schema di Figura 21.

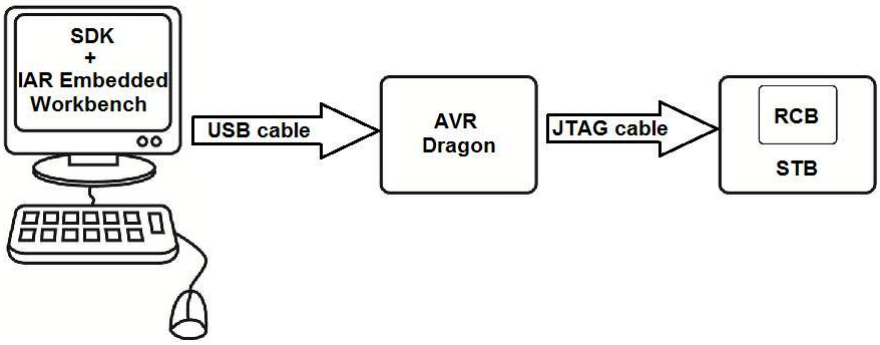


Figura 20: Schema a blocchi. Trasferimento del firmware di dimostrazione e debug di una RCB montata su STB, attraverso AVR Dragon.

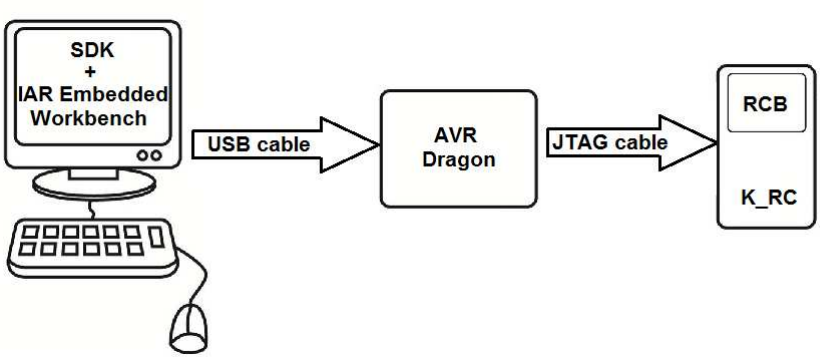


Figura 21: Schema a blocchi. Trasferimento del firmware di dimostrazione e debug di una RCB montata su K_RC, attraverso AVR Dragon.

5.5.2 Utilizzo dello sniffer con una rete ZLL

Una volta programmati i nodi e creata la rete ZLL, si possono studiare i pacchetti in aria che vengono scambiati dai dispositivi con lo sniffer.

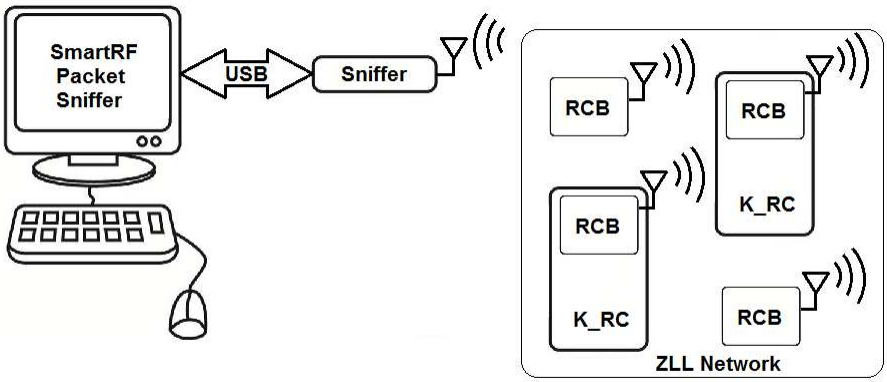


Figura 22: Studio dei pacchetti che vengono scambiati tra dispositivi di una rete ZLL attraverso lo sniffer.

CAPITOLO 6: valutazione del profilo

6.1 Panoramica dei test sperimentali effettuati

Per testare e valutare il funzionamento generale dello standard ZLL e capire se e come questo profilo possa risolvere le problematiche presentate al capitolo 3, si è proceduto all'implementazione di reti ZLL studiandone il comportamento in diversi scenari, ciascuno dei quali è mirato all'indagine di aspetti e comportamenti precisi della rete. La tipologia dei test svolti è riassumibile con la seguente schematizzazione:

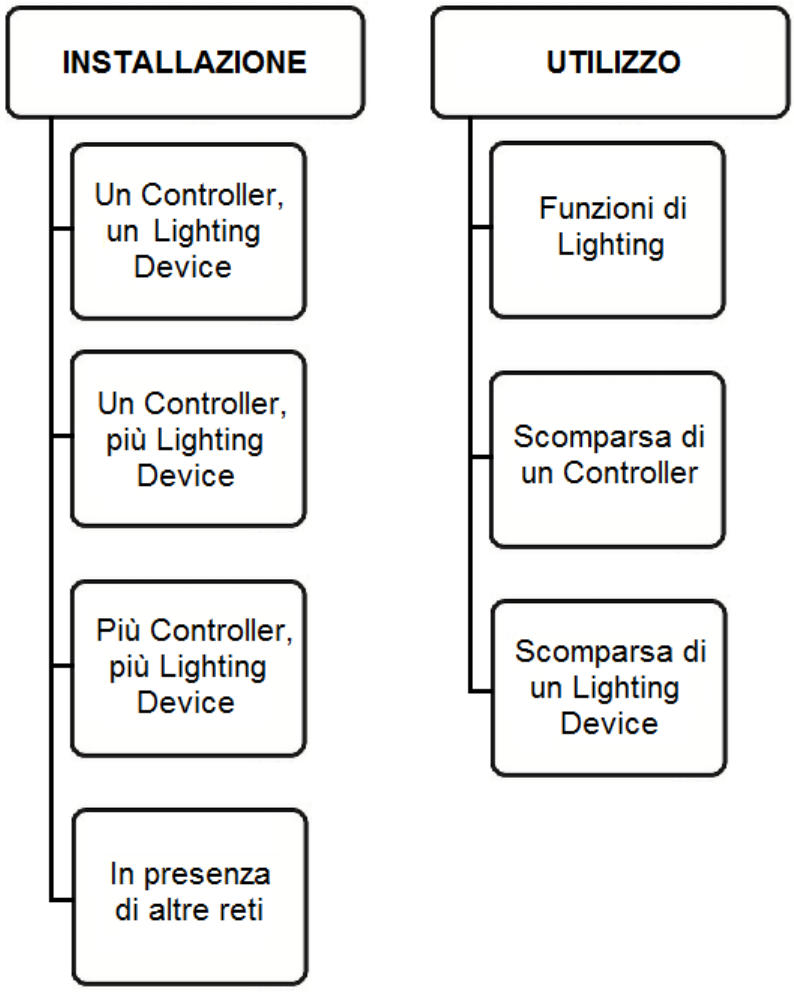


Figura 23: diagramma dei test eseguiti.

Si noti che, nell’esecuzione delle prove, la casistica è stata suddivisa in due macrogruppi: installazione ed utilizzo.

Il primo riguarda i test di associazione e di start di rete, mentre il secondo concerne il controllo dei dispositivi di una rete già avviata. Per quanto riguarda i test di installazione, verranno considerati i seguenti casi significativi: un primo scenario di base in cui si debba formare una rete popolata da un solo Controller e un Lighting Device, poi le estensioni ottenute aumentando il numero di Lighting Device e di Controller, ed infine un’ultima situazione in cui si procede all’installazione in un’area dove è già presente una rete di qualche tipo.

Anche per i casi di utilizzo sono stati considerati diversi scenari: nel primo si vuole affrontare il controllo classico dei punti luce, nel secondo si studia il caso di mobilità del Controller. Si immagini, ad esempio, un utente che si allontana dalla rete dimenticando nella propria tasca il Controller: questo costituisce ciò che è stato denominato “Scomparsa del Controller” dalla rete. Infine, come ultimo scenario di utilizzo, si è ipotizzato un guasto ad un Lighting Device che quindi viene a mancare costituendo il caso di “Scomparsa di un Lighting Device”.

6.2 Configurazione dei dispositivi in ZLLDemo

Prima di descrivere i test sperimentali effettuati, verranno qui riportate alcune importanti note sul firmware. Per maggiori informazioni consultare la documentazione del kit Atmel RF4CE.

- Reset delle funzionalità ZLL:

Ogniqualvolta nel firmware si utilizzano le funzioni specifiche dello standard ZLL, che prevede un uso speciale dello stack ZigBee, è prima necessario impostare quest'ultimo invocando l'apposita funzione *ZCL_ZllReset().*

- Lo stato Factory New:

Dopo la programmazione, i nodi di una rete ZLL si trovano in uno stato definito Factory New (FN), ovvero non facenti di parte di alcuna rete e non aventi nessun tipo di conoscenza su eventuali reti o nodi circostanti. Quando queste condizioni vengono meno in seguito ad un touchlink, i dispositivi sono etichettati come Non Factory New (NFN). Le API di BitCloud permettono di resettare un dispositivo da NFN a FN attraverso la funzione *ZCL_ZllIbResetToFactoryNew()*. Inoltre un Lighting Device può essere resettato a FN dal Controller, che manda un comando invocando la funzione *ZCL_ZllResetToFactoryNewRequest()* specificando nell'argomento l'extended ed il short address del dispositivo target. Questa funzione è utilizzata per forzare un nodo target a compiere il cosiddetto Leave di rete. Infine *ZCL_ZllIsFactoryNew()* è una funzione usata per verificare se un dispositivo è FN o meno.

- Organizzazione dei canali radio:

Si ricorda che, come riportato alla sezione 1.2, alla frequenza di funzionamento di 2.4 GHz, vi sono sedici canali disponibili. Essi sono numerati dall'11 al 26 (in esadecimale, da 0x0B a 0x1A). Il canale di default dei nodi ZLL è il numero 11 (0x0B).

Esso è modificabile nel file *configuration.h* (in appendice). Nello stesso file viene poi definita una maschera di canale che consiste in una *PRIMARY_CHANNEL_MASK* e una *SECONDARY_CHANNEL_MASK.* La prima viene usata per selezionare quattro canali dei sedici disponibili, la seconda invece prende in considerazione tutti i canali rimanenti. Anche per queste ci sono valori predefiniti che Atmel consiglia di non cambiare se non in fase di test, per assicurare l'interoperabilità e il corretto funzionamento dei dispositivi: la maschera primaria standard seleziona i canali 11, 15, 20, 25 e la secondaria tutti i restanti. La loro funzionalità è di dare un ordine ad un eventuale scorrimento dei canali da parte dell'applicazione. Per eseguire questo primo test i valori sono stati lasciati invariati.

6.3 Installazione

6.3.1 Un Controller, un Lighting Device

Il punto di partenza per la valutazione dello standard ZLL è quello in cui ci si accinge ad installare una nuova rete in un ambiente ideale, senza disturbi. Per fare ciò sono necessari almeno due dispositivi: un Controller ed un Lighting Device.

Questo è lo scenario base dal quale derivano tutti gli altri, per cui anche quello sul quale questa trattazione si focalizzerà maggiormente. Si è proceduto alla programmazione dei nodi (si vedano i capitoli 4 e 5) seguendo gli schemi di Figura 20 e Figura 21. Si sono trasferiti i firmware appositi sulle RCB in modo da ottenere un Controller (RCB + K_RC), che in questa trattazione verrà chiamato C1, e un Lighting Device (RCB stand-alone), L1.

- ACCENSIONE DI C1 (FN)

Alimentando C1, dallo sniffer non viene rilevata alcuna attività radio né sul canale di default né su nessun altro canale. Questo dato è in accordo con la filosofia dello standard ZLL: il Controller è uno speciale End Device ed è inizialmente in uno stato di risparmio energetico in cui il suo chip radio è spento.

Dall'esecuzione del file sorgente *colorSceneRemote.c,* monitorata con AVR Dragon in modalità debugger, si osserva infatti che all'accensione vengono svolte inizializzazioni interne tra cui il settaggio del parametro *CS_WriteParameter(CS_RX_ON_WHEN_IDLE_ID, &(bool){false})* che ha proprio la funzione di mantenere il transceiver normalmente disattivato. Dopo le inizializzazioni, C1 è portato in uno stato di idle, in attesa della pressione del pulsante dedicato all'associazione con altri nodi.

In questo stato si è misurata con multimetro la corrente assorbita dalla RCB. Il risultato riscontrato è in linea a quanto riportato nel datasheet del microcontrollore ATMega128RFA1, considerando il chip radio in sleep mode: 4.5 mA.

Si noti che per poter raffrontare questa misura con i dati del manuale, la RCB deve essere separata dalla K_RC, la quale assorbe una corrente fissa di circa 10mA per alimentare alcuni led e circuiteria accessoria. Misurando infatti il consumo dell'intero Controller Atmel in questo stato, si trova una corrente di 14.5 mA.

- ACCENSIONE DI L1 (FN)

Alla sua prima accensione, L1 invia un messaggio su tutti i canali radio, seguendo l'ordine definito dalla maschera primaria e poi passando ai canali della secondaria. Si tratta di un messaggio di Beacon. Tipico delle reti ZigBee, esso viene spedito dai nodi in modalità broadcast per scoprire la presenza di altri nodi, sincronizzarsi ed eventualmente procedere con il join di rete. L1 quindi esegue una scansione iniziale in ricerca di altri dispositivi. Questo comportamento è settabile attraverso la seguente definizione *di APP_SCAN_ON_STARTUP* al valore 1, nel file *configuration.h.*

P.nbr.	Time (us)	Length	Frame control field					Sequence number	Dest. PAN	Dest. Address		LQI	FCS
			Type	Sec	Pnd	Ack.req	PAN_compr						
RX	+0												
1	=0	10	CMD	0	0	0	0	0x5F	0xFFFF	0xFFFF	Beacon request	158	OK

Figura 24: Sniffer, Beacon inviata da L1 su tutti i canali definiti dalle maschere.

Se invece il valore del define è impostato a 0, L1 non avrà modo di ricercare altre eventuali reti al suo avvio. Essendo C1 programmato come End Device, non è in grado di rispondere all'interrogazione di L1 che rimane in ascolto, mantenendo il chip radio normalmente acceso e preparando lo stack ad una comunicazione inter-PAN con la funzione *ZCL_ZllInterPanStartReq()* in attesa di una richiesta di touchlink.

La corrente assorbita da L1 è di 15 mA con un picco di 18 mA corrispondente alla trasmissione del messaggio di Beacon, dati in linea anch’essi con quelli dichiarati da Atmel.

- TOUCHLINK

Conformemente al razionale descritto alla sezione 2.3.1, il touchlink implementato da Atmel è scatenato dalla pressione di un pulsante su C1. Nel kit specifico tale pulsante è denominato PWR e situato sulla tastiera della K_RC. Alla sua pressione, C1 si prepara ad aprire una comunicazione inter-PAN con la funzione *ZCL_ZllInterPanStartReq()* poi spedisce un messaggio sui canali definiti dalla sua maschera principale cerando un Lighting Device nei suoi dintorni.

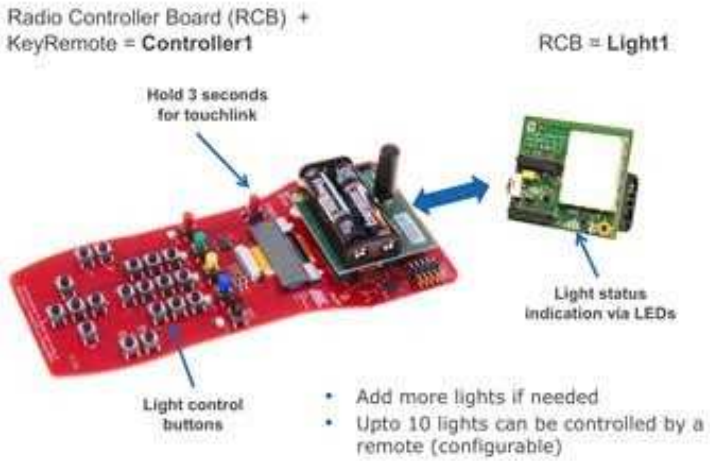


Figura 25: immagine esplicativa del touchlink tra i dispositivi del kit Atmel.

- Touchlink failure

Può succedere che, in base alla vicinanza, la visibilità, l'alimentazione e i diversi canali di default dei dispositivi, il touchlink non vada a buon fine e C1 non trovi nessun Lighting Device. Questo primo caso di touchlink fallito è stato simulato togliendo l'alimentazione a L1 e premendo il pulsante PWR su C1.

Lo sniffer rileva che, a fronte della chiamata a funzione *ZCL_ZllScanReq()*, vengono inviati cinque tentativi identici sul canale di default, prima di tornare in stato di idle. Questa attività radio dura complessivamente 1,0004 secondi (tempo rilevato dallo sniffer e visibile in Figura 26, dai campi Time), dopodiché non viene più spedito nessun pacchetto.

Si noti che il messaggio è spedito in broadcast (campo Destination Address 0xFFFF) da C1 che ha ancora un indirizzo esteso (campo Source Address).

P.nbr.	Time (us)	Length	Frame control field	Sequence number	Dest. PAN	Dest. Address	Source PAN	Source Address	MAC payload	NWK
RX	+0		Type Sec Pnd Ack.req PAN_compr						0B 00 0B 00 10 5E C0 11	Type Version
1	=0	35	DATA 0 0 0 0	0x16	0xFFFF	0xFFFF	0x2D39	0x000425FFFF175B8D	00 00 09 57 82 45 02 33	R11 0x2
RX	+251047		Type Sec Pnd Ack.req PAN_compr						0B 00 0B 00 10 5E C0 11	Type Version
2	=251047	35	DATA 0 0 0 0	0x17	0xFFFF	0xFFFF	0x2D39	0x000425FFFF175B8D	01 00 09 57 82 45 02 33	R11 0x2
RX	+250657		Type Sec Pnd Ack.req PAN_compr						0B 00 0B 00 10 5E C0 11	Type Version
3	=501704	35	DATA 0 0 0 0	0x18	0xFFFF	0xFFFF	0x2D39	0x000425FFFF175B8D	02 00 09 57 82 45 02 33	R11 0x2
RX	+248786		Type Sec Pnd Ack.req PAN_compr						0B 00 0B 00 10 5E C0 11	Type Version
4	=750490	35	DATA 0 0 0 0	0x19	0xFFFF	0xFFFF	0x2D39	0x000425FFFF175B8D	03 00 09 57 82 45 02 33	R11 0x2
RX	+249928		Type Sec Pnd Ack.req PAN_compr						0B 00 0B 00 10 5E C0 11	Type Version
5	=1000418	35	DATA 0 0 0 0	0x1A	0xFFFF	0xFFFF	0x2D39	0x000425FFFF175B8D	04 00 09 57 82 45 02 33	R11 0x2

Figura 26: Sniffer, canale di default. Touchlink failure.

In queste circostanze, da multimetro si rileva ciò che viene valutato come un primo inconveniente: sebbene l'attività termini dopo poco più di un secondo, il chip radio, acceso per spedire i 5 tentativi totali di touchlink, rimane attivato imponendo a C1 un consumo di circa 25 mA, valore che, sottraendo il consumo fisso di 10 mA della K_RC, corrisponde infatti al consumo del microcontrollore con transceiver attivo, riportato sul datasheet (15 mA). Come verrà evidenziato successivamente, ciò accade solamente nel caso in cui C1 sia FN.

- Touchlink succeeded

In Figura 27 viene mostrato il diagramma che rappresenta l'implementazione Atmel del protocollo di touchlink. Alcuni passi non obbligatori secondo le specifiche ZLL sono stati segnalati come opzionali. Di seguito viene riportata la descrizione del test di processo di touchlink (andato a buon fine) tra C1 ed L1, osservato sperimentalmente attraverso l'uso dello sniffer e facendo riferimento ai file *colorSceneRemote.c* e *colorLight.c*, eseguiti passo a passo utilizzando il debugger.

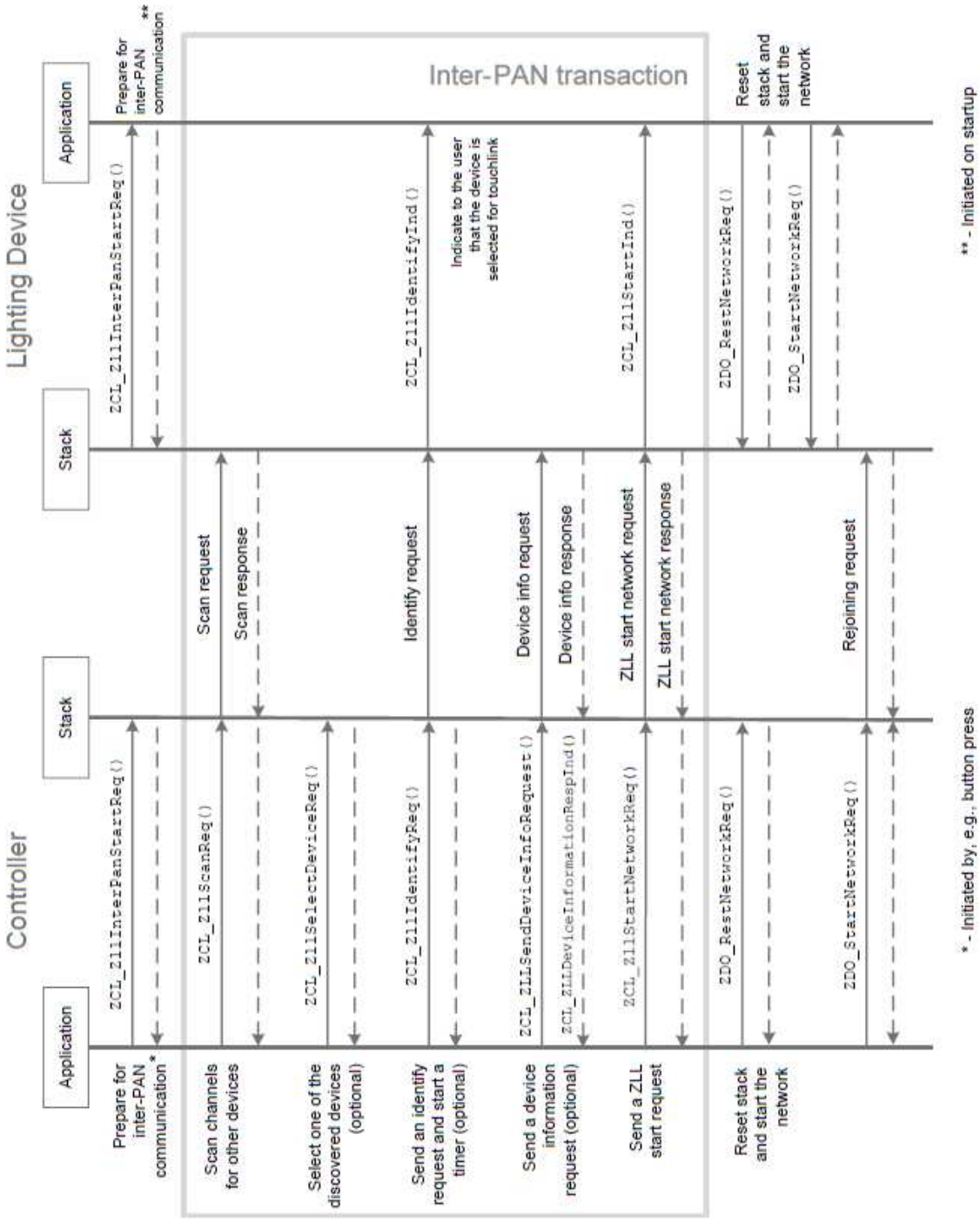
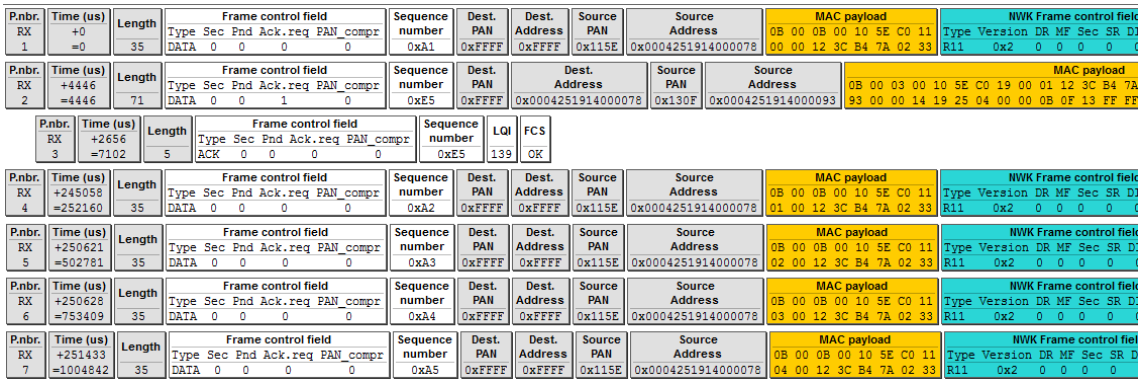


Figura 27: versione Atmel del meccanismo di touchlink.

Con questa prova si sono volute verificare le diverse fasi del processo, dando significato ai pacchetti catturati, associandoli alle corrispondenti chiamate a funzione del firmware, che possono essere ritrovate in Figura 27.



P.nbr.	Time (us)	Length	Frame control field	Sequence number	Dest. PAN	Dest. Address	Source PAN	Source Address	MAC payload	NWK Frame control field
RX 1	+0 =0	35	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 0	0xA1	0xFFFF	0xFFFF	0x115E	0x0004251914000078	0B 00 0B 00 10 5E C0 11 00 00 12 3C B4 7A 02 33	Type Version DR MF Sec SR DI R11 0x2 0 0 0 0 0
RX 2	+4446 =4446	71	Type Sec Pnd Ack.req PAN_compr DATA 0 0 1 0	0xE5	0xFFFF	0x0004251914000078	0x130F	0x0004251914000093	0B 00 03 00 10 5E C0 19 00 01 12 3C B4 7A 93 00 00 14 19 25 04 00 00 0B 0F 13 FF FF	
RX 3	+2656 =7102	5	Type Sec Pnd Ack.req PAN_compr ACK 0 0 0 0	0xE5	LQI 139	FCS OK				
RX 4	+245058 =252160	35	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 0	0xA2	0xFFFF	0xFFFF	0x115E	0x0004251914000078	0B 00 0B 00 10 5E C0 11 01 00 12 3C B4 7A 02 33	Type Version DR MF Sec SR DI R11 0x2 0 0 0 0 0
RX 5	+250621 =502781	35	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 0	0xA3	0xFFFF	0xFFFF	0x115E	0x0004251914000078	0B 00 0B 00 10 5E C0 11 02 00 12 3C B4 7A 02 33	Type Version DR MF Sec SR DI R11 0x2 0 0 0 0 0
RX 6	+250628 =753409	35	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 0	0xA4	0xFFFF	0xFFFF	0x115E	0x0004251914000078	0B 00 0B 00 10 5E C0 11 03 00 12 3C B4 7A 02 33	Type Version DR MF Sec SR DI R11 0x2 0 0 0 0 0
RX 7	+251433 =1004842	35	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 0	0xA5	0xFFFF	0xFFFF	0x115E	0x0004251914000078	0B 00 0B 00 10 5E C0 11 04 00 12 3C B4 7A 02 33	Type Version DR MF Sec SR DI R11 0x2 0 0 0 0

Figura 28: Sniffer, canale di default. Scan request e scan response tra C1 ed L1.

Alla pressione del pulsante PWR su C1, tenuto a distanza ravvicinata da L1, lo sniffer mostra una lunga serie di pacchetti intercettati. Considerando i primi sette e seminando opportunamente i breakpoint lungo il firmware in fase di debug si deduce che C1 ha invocato la funzione *ZCL_ZllScanReq()* provocando la trasmissione di cinque messaggi broadcast di tipo inter-PAN per scansionare il canale di default. Questi messaggi si identificano nei pacchetti con Packet Number 1, 4, 5, 6, 7 di Figura 28. L'indirizzo temporaneo di C1 è leggibile nel campo Source Address di ognuno di questi pacchetti. Inoltre C1 spedisce un messaggio di scan su ogni canale della maschera principale (si veda la sezione 6.3.4). Al pacchetto 1 risponde L1.

Il pacchetto di riposta ha Packet number 2 e si contraddistingue grazie al campo Destination Address, contenente l'indirizzo temporaneo di C1. Palesemente, al campo Source Address dello stesso pacchetto si trova quindi l'indirizzo temporaneo di L1. Questa risposta, inviata da L1 a C1, pretende un messaggio di Acknowledge (campo ACK settato a 1) che infatti è inviato da C1 con il pacchetto numero 3.

Per semplicità della trattazione, non verranno più menzionati i messaggi di ACK, sebbene sempre presenti. Gli altri messaggi di scan hanno lo scopo di dare la possibilità ad altri Lighting Device di concorrere al touchlink competendo tra loro. In caso di risposte multiple provenienti da dispositivi differenti, la scelta del dispositivo più idoneo è effettuata dall'applicazione del Controller attraverso la funzione *ZCL_ZllSelectDeviceReq()*. Il criterio implementato da Atmel è quello di selezionare il device che ha risposto allo scan con il segnale più potente, che si presuppone essere il più vicino a C1 e quindi quello che si intende associare. Per fare questo viene utilizzata la funzione *ZCL_ZllIsLowSignalStrength()*. In questo primo test è inizialmente presente il solo L1. Gli altri messaggi di scan vengono ignorati e la scelta del dispositivo più idoneo al touchlink ricade forzatamente su di esso.

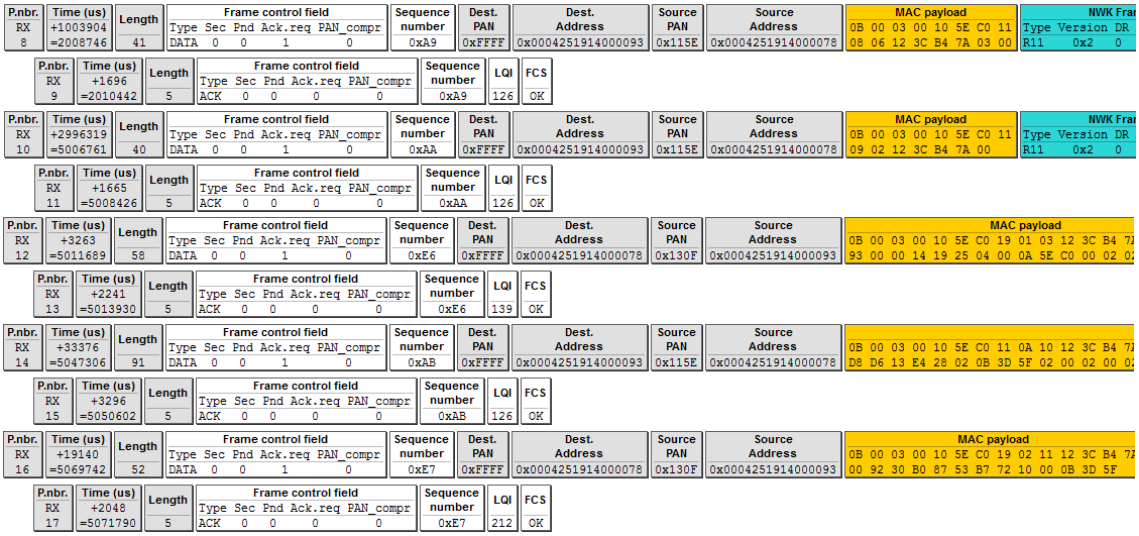


Figura 29: Sniffer, canale di default. Identify Request, Device Info Request e Response, Zll Start Network Request e Response.

Una volta che L1 è stato selezionato, C1 invia ad esso un messaggio di Identify Request chiamando la funzione *ZCL_ZllIdentifyReq()*. Questo messaggio è stato riscontrato essere corrispondente al pacchetto 8, di Figura 29. L'applicazione del Lighting Device implementa ora la funzione *ZCL_ZllIdentifyInd()*, chiamata di conseguenza alla ricezione di Identify Request e grazie alla quale esso comincia a fare lampeggiare un led: C1 ha identificato L1 che mostra all'utente di essere stato selezionato attraverso un segnale luminoso. I pacchetti 10 e 12 sono i messaggi di Device Info Request e Response, che sono gestiti dall'applicazione di C1 attraverso le funzioni *ZCL_ZLLSendDeviceInfoRequest()* e *ZCL_ZLLDeviceInformationRespInd()*.

Con questi messaggi C1 è venuto a conoscenza di tutte le informazioni del Lighting Deivice L1. All'invio di Identify Request l'applicazione di C1 fa partire un timer chiamato *identifyTimer*, entro il quale l'utente può rilasciare il pulsante PWR abortendo la procedura se ad esempio si accorge che è stato selezionato un Lighting Device non desiderato. Se L1 è stato correttamente identificato e il pulsante PWR è stato tenuto premuto fino allo scadere di *identifyTimer*, si scatena l'invio, da parte di C1, di un messaggio di ZLL Start Network Request attraverso la funzione *ZCL_ZllStartNetworkReq()* (pacchetto 14). Con questo messaggio, C1 inoltra i parametri di rete a L1, la cui applicazione è notificata da *ZCL_ZllStartInd()* che prenderà a carico i parametri di rete per poi farne lo start una volta fuori dalla transazione inter-PAN. L1 comunica a C1 l'avvenuta ricezione con il pacchetto 16, un messaggio di ZLL Start Network Response. Da ora in poi C1 ed L1 sono NFN.

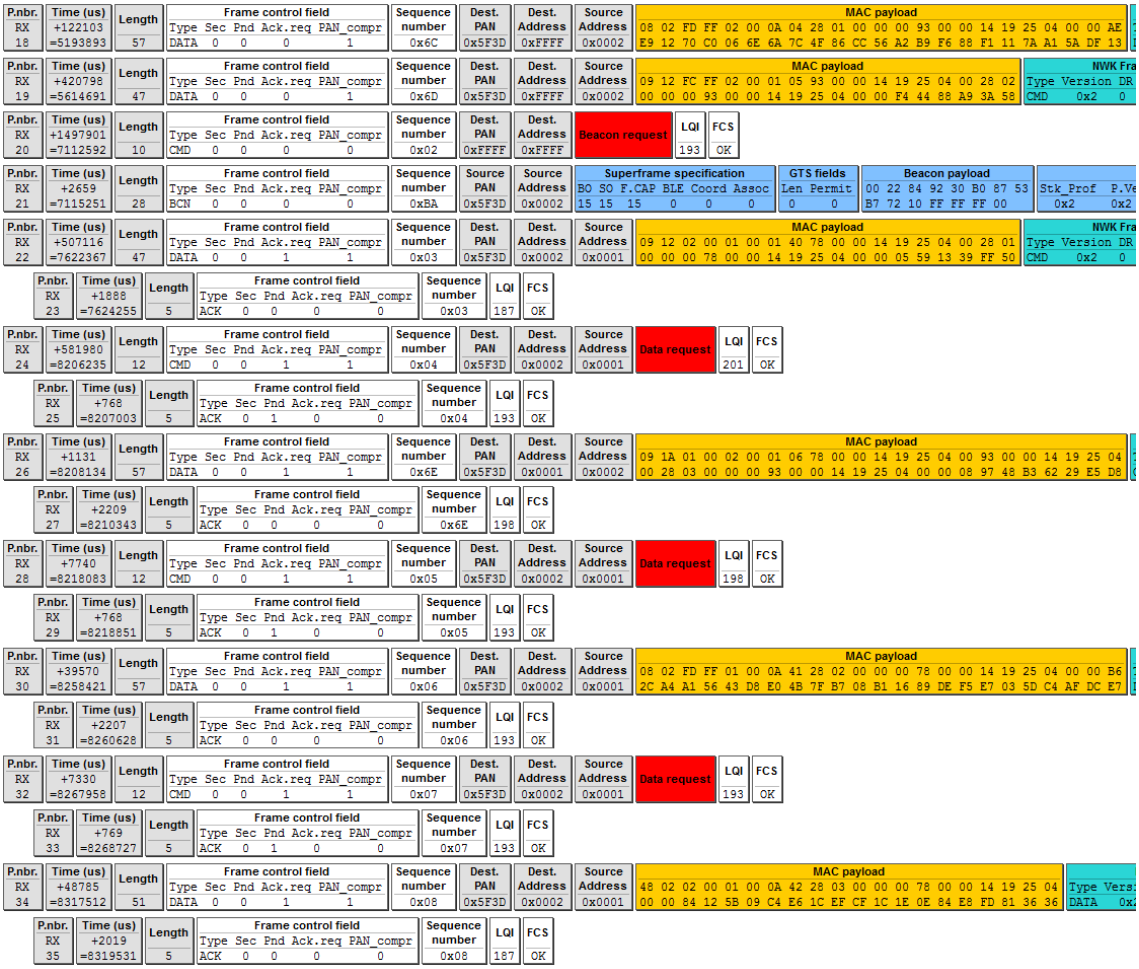


Figura 30: Sniffer, canale di default. Rejoining Request e Response.

Dal pacchetto 18 in avanti si è fuori dalla transazione inter-PAN. Le applicazioni di C1 e L1 resettano lo stack attraverso le funzioni *ZDO_ResetNetworkReq()* tornando al normale funzionamento ZigBee standard. Nelle rispettive callback verranno invocate le funzioni *ZDO_StartNetworkReq()*. L1 effettuerà finalmente lo start di rete annunciandolo con il broadcast del pacchetto 18 (il pacchetto 19 è un messaggio di Link Status che annuncia periodicamente ai vicini la presenza di L1) e conseguentemente C1 eseguirà una classica richiesta di associazione alla rete.

Questa procedura è riconosciuta nei pacchetti che vanno dal numero 20 al 23: C1 invia un messaggio di Beacon al quale L1 risponde con un Superframe e C1 entra a far parte della rete come End Device.

Da qui in avanti L1 è Parent di C1 il quale si rivolgerà sempre ad esso per l'invio di qualsiasi comando. Come si nota da Figura 30, dopo lo start di rete i nodi hanno cominciato ad utilizzare il Short Address che ha imposto L1. A questo punto l'applicazione di C1 fa partire un timer chiamato activityTimer entro lo scadere del quale l'End Device esegue polling sul prorpio Parent L1 ogni 500 ms, attraverso la funzione *ZDO_StartSyncReq()*. Il polling è originariamente un meccanismo di richiesta dati periodico tipico dell'End Device ZigBee il quale si trova normalmente in sleep mode e si sveglia di tanto in tanto interrogando il nodo Parent con un messaggio di tipo Data Request.

Il touchlink è terminato ma effettivamente la presenza di pacchetti con payload che vanno dal 24 al 25 significa che ancora ci sono informazioni che C1 e L1 devono scambiarsi, come ad esempio informazioni di gruppo previste dall'applicazione per la gestione intelligente dei Lighting Device in scene di luce. Dal pacchetto 35 in poi si catturano solo messaggi di Data Request inviati da C1 a L1 (non riportati in figura): è polling inutile, visto che L1 non ha più dati da consegnare a C1. Allo scadere di activityTimer il polling verrà arrestato con *ZDO_StopSyncReq()* e il chip radio disattivato. Se il timer viene sovradimensionato di ha una inefficienza energetica. C1 è pronto per pilotare L1 e aspetta la pressione dei pulsanti della sua tastiera per tornare ad attivare il chip radio e inviare comandi.

Considerando i campi Time dei pacchetti, il tempo impiegato per l’attività di touchlink è 7,62 secondi, dopo i quali C1 può già pilotare L1.

Lungo questo periodo vi è un’intensa attività radio che porta i dispositivi al consumo di una corrente misurata con multimetro di circa 18 mA, corrispondente al valore riportato sul manuale del microcontrollore con chip radio acceso e attività di trasmissione in corso (in ricezione si consuma qualche milliampere in meno). Il chip radio rimane attivo per permettere il polling di C1 su L1 fino allo scadere di activityTimer il cui dimensionamento quindi incide sul consumo di C1.

Il suo valore di default è sovrastimato di qualche secondo. Dopodiché si torna ad consumare 4,5 mA. D’ora in avanti se C1 viene spento, al suo riaccendi mento cercherà la connessione con L1 ed L1 manderà messaggi di Link Status ogni 15 secondi.

6.3.2 Un Controller, più Lighting Devices

Il test di installazione prosegue nell’allargare la rete, aggiungendo un altro (od altri N) Lighting Device, che verrà chiamato L2 (L3,L4,…,LN). Alimentandolo, anche L2 spedisce un messaggio di Beacon iniziale. Sebbene esso riceva in risposta un Superframe da L1, null’altro accade, poiché L1 non ha il potere di aggiungere automaticamente un nuovo Lighting Device alla sua rete: l’associazione può avvenire soltanto attraverso un touchlink con C1, come descrive la specifica ZLL. Si esegue quindi nuovamente la procedura affrontata alla sezione precedente.

Si nota che essa è identica per l'intera transazione inter-PAN, poi risulta velocizzata dal fatto che la rete è già stata creata precedentemente da L1, ed L2 deve sono eseguire il join ad essa. Si noti che L1 rimane Parent di C1 ed ogni messaggio che quest'ultimo deve effettuare dalla fine della transazione di inter-PAN in poi, passa forzatamente attraverso L1. Questo comportamento è stato osservato e catturato con lo sniffer al termine del tochlink tra C1 ed L2 e riportato in Figura 31.

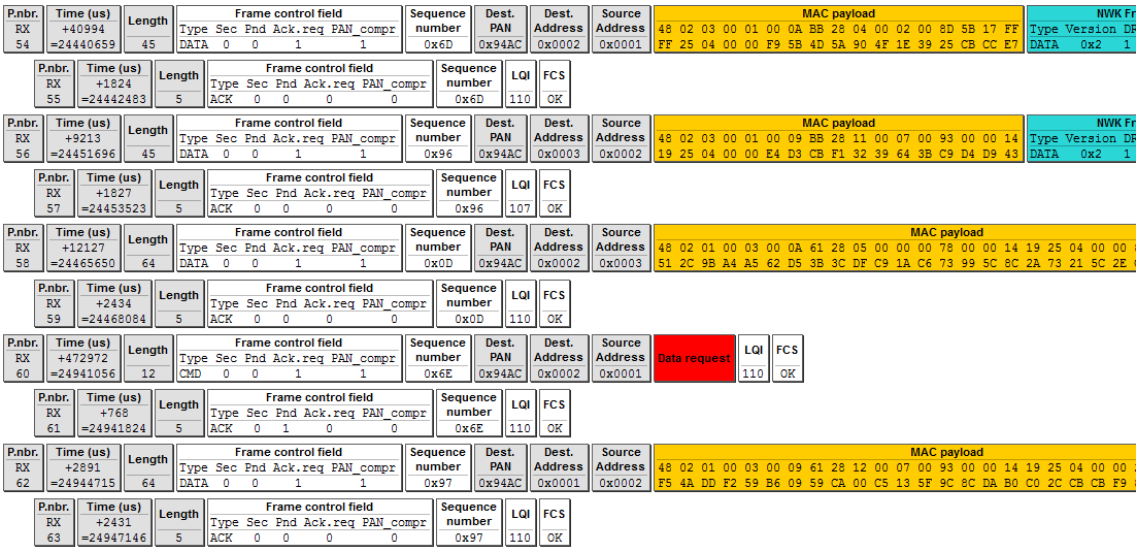


Figura 31: Sniffer, canale di default. Esempio di scambio dati tra C1 e un Lighting Device aggiuntivo L2, al termine di touchlink tra gli stessi.

6.3.3 Più Controller, più Lighting Device

Volendo generalizzare lo scenario precedente anche all'uso di più Controller, bisogna distinguere due casi possibili:

- Ogni Controller pilota Lighting Device diversi e quindi esistono tante reti ZLL quanti Controller

- Si desidera avere alcuni Lighting Device pilotati da più Controller contemporaneamente, quindi esistono reti ZLL con più di un Controller

Il primo caso si riduce banalmente alla ripetizione del primo test di installazione alla sezione 6.3.2, iterato tante volte quanti sono i Controller che si vogliono utilizzare. Più reti ZLL possono convivere anche sullo stesso canale, tuttavia si può anche scegliere di cambiare spostare il canale di utilizzo di una rete (si veda la sezione 6.3.4).

Per il secondo caso invece esiste la possibilità di eseguire touchlink tra due Controller, uno (C1) che fa già parte di una rete ZLL ed un secondo (C2) che invece è FN. È stato previsto da Atmel che, per inizializzare questa procedura, si premano simultaneamente i tasti PWR sui due Controller. Testando questa eventualità si è osservato che i Controller si preparano ad una sessione di inter-PAN e cominciano la scansione dei canali. C2 si interrompe non appena riceve la Scan Request di C1 (NFN). Quest’ultimo finisce lo scan normalmente e seleziona C1 con gli stessi criteri descritti per il touchlink con Lighting Device. C1 invia un messaggio di Start Network Request in formato speciale rivolto a C2, con tutte le informazioni necessarie per permettergli di eseguire il join alla rete.

Si noti infine che, se un Controller esegue touchlink con un Lighting Device che faceva già parte di una rete ZLL, quest’ultimo esegue il Leave dalla rete originaria e per crearne un’altra con i nuovi parametri commissionatigli.

6.3.4 Presenza di altre reti

Questo ultimo caso di installazione vuole indagare il comportamento di reti ZLL in luoghi ove sono presenti altri dispositivi operanti sulle frequenze di utilizzo. Nulla accade se la rete già presente è anche essa del tipo ZLL: le due reti possono coesistere anche sullo stesso canale grazie alla gestione della sicurezza come da ZigBee standard. Se invece il canale di default della rete è disturbato da qualche attività radio non compatibile con ZLL, è stata prevista la possibilità di cambio canale. Nonostante la soluzione sia implementata a livello applicativo da Atmel in questo kit di valutazione e non faccia parte quindi della specifica dello standard ZLL, è interessante analizzarla in quanto costituisce un tentativo di soluzione ad un problema cruciale per questa tecnologia: la scelta e lo switch intelligente del canale radio. Questa funzionalità è innescata alla pressione del tasto 4 sulla tastiera del Controller. Nel suo firmware, la funzione che se ne occupa è *sendNwkUpdateReqForFrequencyAgility()*. Il Controller manda un messaggio broadcast dove si informando tutti i Lighting Device a lui associati del cambio canale, invitandoli a fare altrettanto. La routine sceglie il nuovo canale senza l'uso di alcuna intelligenza, passando dal canale corrente al canale successivo presente nella maschera principale.

Con lo sniffer si sono visualizzati i pacchetti contenenti questa comunicazione sul canale di default (canale 11) e trascorso il tempo necessario allo switch si è verificato l'effettivo interrompimento delle trasmissioni su tale canale. Mettendo poi in ascolto lo sniffer sul canale 15, successivo a quello di default nella maschera principale, si è effettivamente ritrovata l'attività della rete.

Questa implementazione di cambio canale ha diversi punti deboli riscontrati nei test che verranno discussi nella sessione finale di questo capitolo. Il più grave è costituito dal fatto di perdere l'associazione di quei Lighting Device che, mentre si effettua il salto di frequenza, per qualche motivo non sono raggiungibili dalla notifica broadcast inviata dal Contoller.

È inoltre d'uopo ricordare che, alla pressione del pulsante PWR sul Controller, esso invia uno scan sui cinque canali definiti dalla maschera principale, pertanto il meccanismo di touchlink riesce ad associare anche Lighting Device operanti in canali diversi che per esempio avevano subito un cambio di canale, a condizione che i canali in cui si trovano siano presenti in maschera. Si è verificato che è il Controller ad imporre il canale al Lighting Device operante su canale diverso. In particolare, dapprima il touchlink avviene sul canale in cui quest'ultimo è operativo, poi, al momento dello scambio delle informazioni di rete (ZLL Start Network Request), avviene il salto al canale del Controller.

6.4 Utilizzo

Si passa ora ai test delle funzionalità lato utente della rete ZLL. Questo standard prevede molte possibilità di controllo dei Lighting Device attraverso controller. In questa sede si vedranno solo le funzionalità principali in quanto questa valutazione verte sulla novità rappresentata dal metodo di associazione, discusso ampiamente nelle sezioni precedenti.

È comunque interessante verificare la comunicazione tra dispositivi ZLL una volta creata la rete per constatare se l’uso specifico dei nodi di tipo End Device in questo nuovo standard soffre o meno dei limiti tipici da cui invece era affetta la specifica classica ZigBee.

6.4.1 Funzioni principali di Lighting

Tutti i comandi che un Controller può inoltrare ai Lighting Device ad esso associati per pilotarli seguono circa gli stessi step:

- Si attiva il chip radio del Controller e si attiva il polling con *ZDO_StartSyncReq()* di modo da poter ricevere i dati del nodo Lighting Device Parent
- Si spedisce il comando contenente le informazioni sul desinatario al nodo Parent
- Il nodo Parent inoltra il comando al Lighting Device in modo unicast, multicast o broadcast a seconda di una scelta effettuata a lato Controller
- Il Lighting Device esegue il comando e risponde al nodo Parent
- Il nodo Parent, interrogato a polling dal Controller, riporta Il messaggio di notifica del Lighting Device
- Il Controller arresta il polling con *ZDO_StopSyncReq()* e spegne il chip radio

- COMANDO DI SELEZIONE DEL LIGHTING DEVICE

Premendo il tasto SEL sulla tastiera del Controller prima di un qualsiasi comando di Lighting, si seleziona il Lighting Device a cui il comando è destinato. Premendo ripetutamente il pulsante si potranno scorrere i dispositivi che è possibile comandare e che, se selezionati, paleseranno il loro stato di interessamento attraverso il lampeggio di un led di segnalazione. Una volta premuto il tasto SEL si ha un certo lasso di tempo in cui è possibile comandare via unicast il Lighting Device selezionato. Questo tempo è specificato nel campo *cmd.payload.identyfy.identifyTime* (da non confondere con il timer *identifyTimer,* usato durante il touchlink), nell'applicazione del Controller, alla funzione chiamata *identifySelectedDevice()*. Si è proceduto ad aumentare il suo valore per rendere più agevoli le prove.

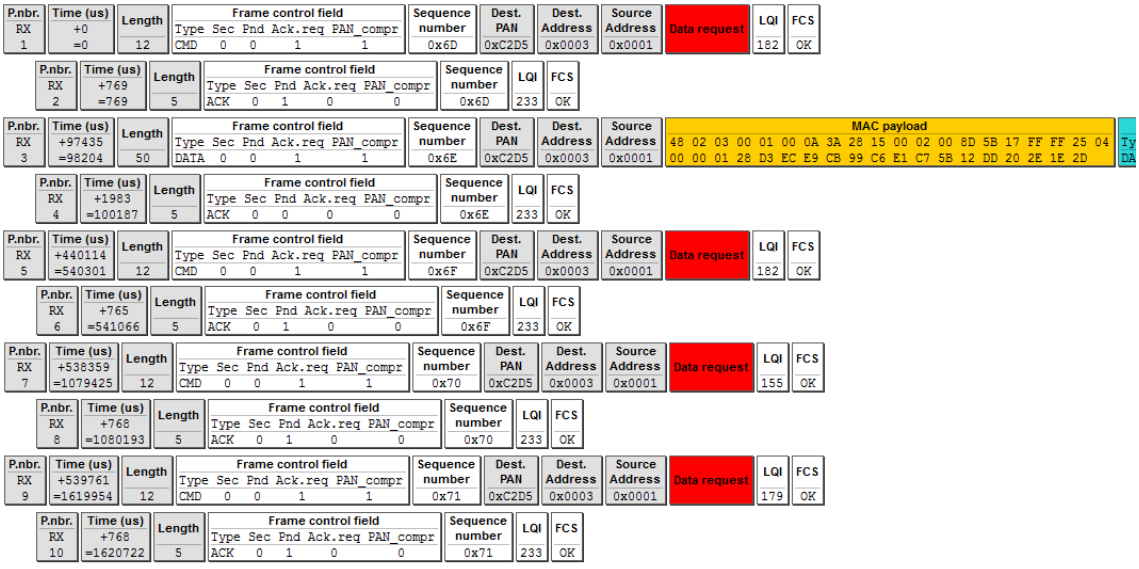


Figura 32: Sniffer, canale di default. Invio del comando di selezione da C1 a L2.

- COMANDO DI ON/OFF

Come la maggior parte dei comandi il messaggio di accensione/spegnimento di un punto luce può essere mandato in unicast, usando prima il tasto SEL per selezionare il Lighting Device da azionare, multicast, tramite la formazione di scene, che ai fini di questa trattazione non sono state considerate, o infine in broadcast, tramite la sola pressione dei pulsanti della tastiera K_CR che ne scatenano gli eventi, denominati L+ ed L-, in basso a sinistra.

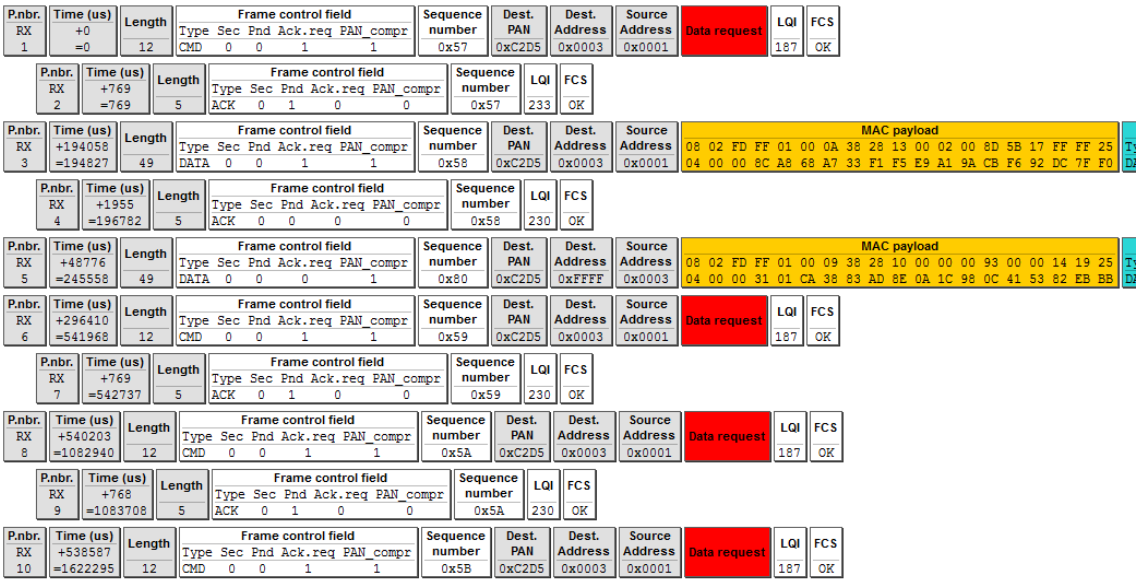


Figura 33: Sniffer, canale di default. Invio broadcast del messaggio di accensione dei punti luce.

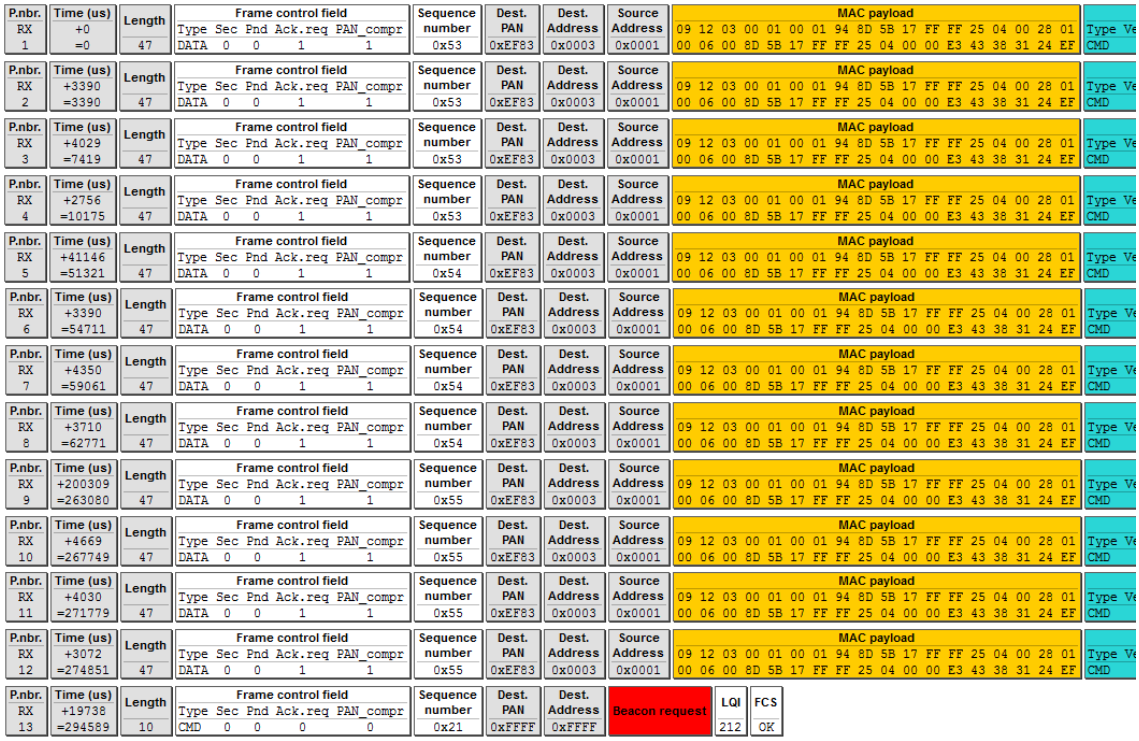
Sono disponibili molti altri comandi di Lighting, come ad esempio quello di aumento e diminuzione dell’intensità luminosa di un Lighting Device la cui implementazione è tutto sommato simile a quella dei comandi sopra descritti e che quindi non si ritiene sensato presentare ai fini di una valutazione sugli aspetti ingegneristici dello standard.

6.4.2 Scomparsa di un Controller

Si consideri una situazione di utilizzo in cui la rete costituisce un impianto di illuminazione casalingo e ad esempio l'utilizzatore, uscendo di casa, porta inavvertitamente con se il Controller. Se si trattasse di una rete ZigBee classica, si avrebbe il risveglio periodico dell'End Device che, interrogando a polling il suo nodo Parent e non ricevendone risposta poiché portato fuori range, si auto dichiarerebbe Orphan dopo una serie di tentativi. A questo punto l'End Device tenta un rejoin alla rete di nuovo non ricevendo risposta. Questo meccanismo si ripeterebbe di continuo fino ad esaurire le batterie del nodo.

Si è verificato che, in una rete ZLL, il Controller ha il chip radio normalmente spento e non si accende mai se non alla pressione di un pulsante di comando. Se ciò accade e l'End Device interroga il suo nodo Parent e, se dopo alcuni veloci tentativi ripetuti non ha risposta, manda una Beacon affinché un eventuale altro Lighting Device nelle vicinanze diventi il suo nuovo Parent. Se non riceve immediata risposta perché, ad esempio, è lontano dalla rete, l'End Device spegne il suo chip radio e aspetta la pressione di un altro pulsate come ad esempio potrebbe essere quello di touchlink.

Questa attività dura 0,294 secondi nei quali il consumo è di 18 mA per l'invio dei tentativi di connessione al Parent e del messaggio di Beacon, del resto il consumo è al solito valore di 4.5 mA a chip radio in power save.



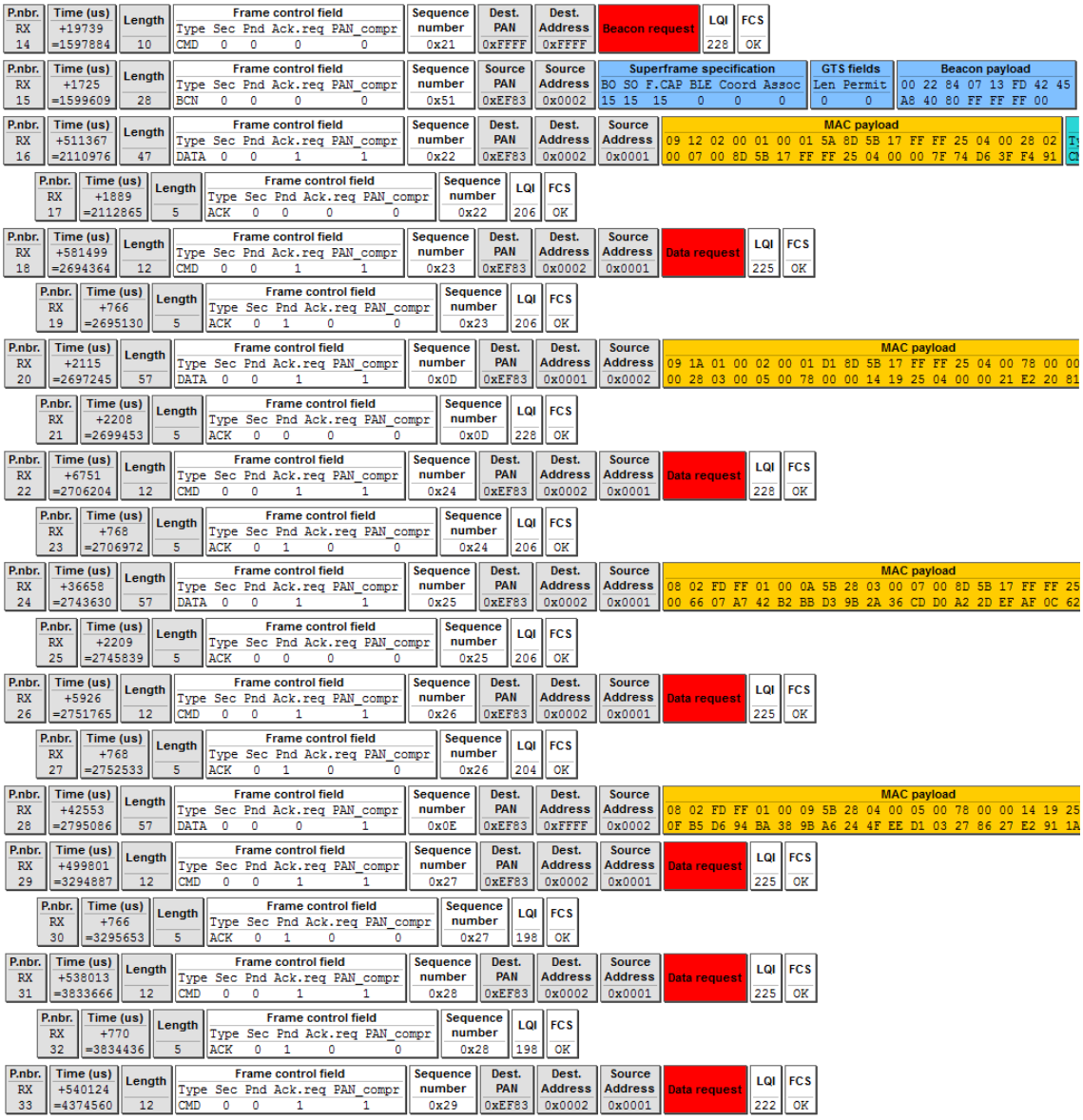
P.nbr.	Time (us)	Length	Frame control field	Sequence number	Dest. PAN	Dest. Address	Source Address	MAC payload	
RX 1	+0 =0	47	Type Sec Pnd Ack.req PAN_compr DATA 0 0 1 1	0x53	0xEF83	0x0003	0x0001	09 12 03 00 01 00 01 94 8D 5B 17 FF FF 25 04 00 28 01 00 06 00 8D 5B 17 FF FF 25 04 00 00 E3 43 38 31 24 EF	Type Ve CMD
RX 2	+3390 =3390	47	Type Sec Pnd Ack.req PAN_compr DATA 0 0 1 1	0x53	0xEF83	0x0003	0x0001	09 12 03 00 01 00 01 94 8D 5B 17 FF FF 25 04 00 28 01 00 06 00 8D 5B 17 FF FF 25 04 00 00 E3 43 38 31 24 EF	Type Ve CMD
RX 3	+4029 =7419	47	Type Sec Pnd Ack.req PAN_compr DATA 0 0 1 1	0x53	0xEF83	0x0003	0x0001	09 12 03 00 01 00 01 94 8D 5B 17 FF FF 25 04 00 28 01 00 06 00 8D 5B 17 FF FF 25 04 00 00 E3 43 38 31 24 EF	Type Ve CMD
RX 4	+2756 =10175	47	Type Sec Pnd Ack.req PAN_compr DATA 0 0 1 1	0x53	0xEF83	0x0003	0x0001	09 12 03 00 01 00 01 94 8D 5B 17 FF FF 25 04 00 28 01 00 06 00 8D 5B 17 FF FF 25 04 00 00 E3 43 38 31 24 EF	Type Ve CMD
RX 5	+41146 =51321	47	Type Sec Pnd Ack.req PAN_compr DATA 0 0 1 1	0x54	0xEF83	0x0003	0x0001	09 12 03 00 01 00 01 94 8D 5B 17 FF FF 25 04 00 28 01 00 06 00 8D 5B 17 FF FF 25 04 00 00 E3 43 38 31 24 EF	Type Ve CMD
RX 6	+3390 =54711	47	Type Sec Pnd Ack.req PAN_compr DATA 0 0 1 1	0x54	0xEF83	0x0003	0x0001	09 12 03 00 01 00 01 94 8D 5B 17 FF FF 25 04 00 28 01 00 06 00 8D 5B 17 FF FF 25 04 00 00 E3 43 38 31 24 EF	Type Ve CMD
RX 7	+4350 =59061	47	Type Sec Pnd Ack.req PAN_compr DATA 0 0 1 1	0x54	0xEF83	0x0003	0x0001	09 12 03 00 01 00 01 94 8D 5B 17 FF FF 25 04 00 28 01 00 06 00 8D 5B 17 FF FF 25 04 00 00 E3 43 38 31 24 EF	Type Ve CMD
RX 8	+3710 =62771	47	Type Sec Pnd Ack.req PAN_compr DATA 0 0 1 1	0x54	0xEF83	0x0003	0x0001	09 12 03 00 01 00 01 94 8D 5B 17 FF FF 25 04 00 28 01 00 06 00 8D 5B 17 FF FF 25 04 00 00 E3 43 38 31 24 EF	Type Ve CMD
RX 9	+200309 =263080	47	Type Sec Pnd Ack.req PAN_compr DATA 0 0 1 1	0x55	0xEF83	0x0003	0x0001	09 12 03 00 01 00 01 94 8D 5B 17 FF FF 25 04 00 28 01 00 06 00 8D 5B 17 FF FF 25 04 00 00 E3 43 38 31 24 EF	Type Ve CMD
RX 10	+4669 =267749	47	Type Sec Pnd Ack.req PAN_compr DATA 0 0 1 1	0x55	0xEF83	0x0003	0x0001	09 12 03 00 01 00 01 94 8D 5B 17 FF FF 25 04 00 28 01 00 06 00 8D 5B 17 FF FF 25 04 00 00 E3 43 38 31 24 EF	Type Ve CMD
RX 11	+4030 =271779	47	Type Sec Pnd Ack.req PAN_compr DATA 0 0 1 1	0x55	0xEF83	0x0003	0x0001	09 12 03 00 01 00 01 94 8D 5B 17 FF FF 25 04 00 28 01 00 06 00 8D 5B 17 FF FF 25 04 00 00 E3 43 38 31 24 EF	Type Ve CMD
RX 12	+3072 =274851	47	Type Sec Pnd Ack.req PAN_compr DATA 0 0 1 1	0x55	0xEF83	0x0003	0x0001	09 12 03 00 01 00 01 94 8D 5B 17 FF FF 25 04 00 28 01 00 06 00 8D 5B 17 FF FF 25 04 00 00 E3 43 38 31 24 EF	Type Ve CMD
RX 13	+19738 =294589	10	Type Sec Pnd Ack.req PAN_compr CMD 0 0 0 0	0x21	0xFFFF	0xFFFF	Beacon request	LQI 212	FCS OK

Figura 34: Sniffer, canale di default. C1 (0x0001) cerca il Parent L1 (0x0003) e non ricevendo risposta tenta un rejoin con altri Lighting Device.

6.4.3 Scomparsa di un Lighting Device

Se invece il Controller fosse ancora entro il range di rete ma fosse il suo nodo Parent a scomparire, il comportamento dell'End Device sarebbe il medesimo. Si è tuttavia ritenuto opportuno testare questa variante di scenario, in caso la rete fosse popolata anche da altri Lighting Device. In questo caso il messaggio di Beacon mandato al termine dei tentativi di connessione al Parent, avrebbe risposta da uno degli altri Router, che diventerebbe il nuovo Parent. Si sono quindi misurate le tempistiche entro le quali un Controller di una rete ZLL voglia.

La rete impiega 2.537 secondi a dare un nuovo Parent al Controller ed altri 0.782 secondi per attivare tutti i punti luce.



P.nbr.	Time (us)	Length	Frame control field	Sequence number	Dest. PAN	Dest. Address	Source Address	Superframe specification	GTS fields	Beacon payload	MAC payload	LQI	FCS
RX 14	+19739 =1597884	10	Type Sec Pnd Ack.req PAN_compr CMD 0 0 0 0	0x21	0xFFFF	0xFFFF	Beacon request					228	OK
RX 15	+1725 =1599609	28	Type Sec Pnd Ack.req PAN_compr BCN 0 0 0 0	0x51	Source PAN 0xEF83	Source Address 0x0002	BO SO F.CAP BLE Coord Assoc 15 15 15 0 0 0	Len Permit 0 0	00 22 84 07 13 FD 42 45 A8 40 80 FF FF FF 00				
RX 16	+511367 =2110976	47	Type Sec Pnd Ack.req PAN_compr DATA 0 0 1 1	0x22	0xEF83	0x0002	0x0001				09 12 02 00 01 00 01 5A 8D 5B 17 FF FF 25 04 00 28 02 00 07 00 8D 5B 17 FF FF 25 04 00 00 7F 74 D6 3F F4 91		
RX 17	+1889 =2112865	5	Type Sec Pnd Ack.req PAN_compr ACK 0 0 0 0	0x22								206	OK
RX 18	+581499 =2694364	12	Type Sec Pnd Ack.req PAN_compr CMD 0 0 1 1	0x23	0xEF83	0x0002	0x0001				Data request	225	OK
RX 19	+766 =2695130	5	Type Sec Pnd Ack.req PAN_compr ACK 0 1 0 0	0x23								206	OK
RX 20	+2115 =2697245	57	Type Sec Pnd Ack.req PAN_compr DATA 0 0 1 1	0x0D	0xEF83	0x0001	0x0002				09 1A 01 00 02 00 01 D1 8D 5B 17 FF FF 25 04 00 78 00 00 00 28 03 00 05 00 78 00 00 14 19 25 04 00 00 21 E2 20 81		
RX 21	+2208 =2699453	5	Type Sec Pnd Ack.req PAN_compr ACK 0 0 0 0	0x0D								228	OK
RX 22	+6751 =2706204	12	Type Sec Pnd Ack.req PAN_compr CMD 0 0 1 1	0x24	0xEF83	0x0002	0x0001				Data request	228	OK
RX 23	+768 =2706972	5	Type Sec Pnd Ack.req PAN_compr ACK 0 1 0 0	0x24								206	OK
RX 24	+36658 =2743630	57	Type Sec Pnd Ack.req PAN_compr DATA 0 0 1 1	0x25	0xEF83	0x0002	0x0001				08 02 FD FF 01 00 0A 5B 28 03 00 07 00 8D 5B 17 FF FF 25 00 66 07 A7 42 B2 BB D3 9B 2A 36 CD D0 A2 2D EF AF 0C 62		
RX 25	+2209 =2745839	5	Type Sec Pnd Ack.req PAN_compr ACK 0 0 0 0	0x25								206	OK
RX 26	+5926 =2751765	12	Type Sec Pnd Ack.req PAN_compr CMD 0 0 1 1	0x26	0xEF83	0x0002	0x0001				Data request	225	OK
RX 27	+768 =2752533	5	Type Sec Pnd Ack.req PAN_compr ACK 0 1 0 0	0x26								204	OK
RX 28	+42553 =2795086	57	Type Sec Pnd Ack.req PAN_compr DATA 0 0 0 1	0x0E	0xEF83	0xFFFF	0x0002				08 02 FD FF 01 00 09 5B 28 04 00 05 00 78 00 00 14 19 25 0F B5 D6 94 BA 38 9B A6 24 4F EE D1 03 27 86 27 E2 91 1A		
RX 29	+499801 =3294887	12	Type Sec Pnd Ack.req PAN_compr CMD 0 0 1 1	0x27	0xEF83	0x0002	0x0001				Data request	225	OK
RX 30	+766 =3295653	5	Type Sec Pnd Ack.req PAN_compr ACK 0 1 0 0	0x27								198	OK
RX 31	+538013 =3833666	12	Type Sec Pnd Ack.req PAN_compr CMD 0 0 1 1	0x28	0xEF83	0x0002	0x0001				Data request	225	OK
RX 32	+770 =3834436	5	Type Sec Pnd Ack.req PAN_compr ACK 0 1 0 0	0x28								198	OK
RX 33	+540124 =4374560	12	Type Sec Pnd Ack.req PAN_compr CMD 0 0 1 1	0x29	0xEF83	0x0002	0x0001				Data request	222	OK

Figura 35: Sniffer, canale di default. L2 (0x0002) risponde aC1 (0x0001) diventando il suo nuovo Parent.

CAPITOLO 7: Report, considerazioni e possibili sviluppi futuri

In questo ultimo capitolo verranno riportati sinteticamente i vantaggi e gli svantaggi che lo standard ha presentato a fronte dei test condotti, suggerendo alcune possibili soluzioni e sviluppi.

7.1 Installazione

Si è visto come per installare una rete ZLL si debba effettuare la procedura di touchlink. Essa avviene manualmente e nodo per nodo, alla semplice pressione del pulsante PWR sul Controller avvicinato al Lighting Device da associare. Questo tipo di associazione è estremamente semplice ed intuitiva e costituisce il principale punto di forza e fattore innovativo rispetto alle modalità ZigBee classiche che hanno algoritmi di associazione automatici e veloci ma complessi, macchinosi e difficilmente pilotabili dall'utente. La facilità di installazione delle reti ZLL rende i suoi prodotti vendibili on-the-shelf e do-it-yoursef abbattendo costi di assistenza e manutenzione tipici invece delle reti ZigBee classiche.

7.2 Tempistiche

- La sua durata, nella versione Atmel, (dati riferiti alle figure 28, 29 e 30) è di circa 7.6 secondi distribuiti come segue:

    - 2.008 s di scan e valutazione delle risposte ricevute per scelta del dispositivo più consono all'associazione
        - 0.008 s per il primo scan e la sua prima risposta
        - 1 s di altri scan (senza risposta) per ricercare eventuali dispositivi concorrenti al touchlink (tempistica migliorabile)
        - 1 s di inattività radio per attendere altre risposte, processarle e, in base ad esse, scegliere il dispositivo più adatto al touchlink (tempistica migliorabile)
    - 5.109 s per selezionare il dispositivo ed inviargli le informazioni di rete necessarie allo start
        - 3 s tempo di identificazione del dispositivo da parte dell'utente (tempistica migliorabile)
        - 0.109 s per scambiarsi le informazioni di rete
        - 2 secondi per il reset degli stack, l'uscita dalla transazione inter-PAN e l'attesa dello start di rete (tempistica migliorabile)
    - 0.509 s di operazioni di associazione ZigBee standard

Per rendere possibile l’associazione tra l’End Device Controller e il Router Lighting Device senza l’utilizzo di un nodo Coordinator sono necessari speciali messaggi inter-PAN per lo scambio delle informazioni di rete necessarie allo start della stessa che richiedono una tempistica del tutto assente nel metodo di associazione ZigBee standard.

Quest’ultima è automatica, priva di interazioni con l’utente ed avviene attraverso un nodo Coordinator che ha già le informazioni di rete necessarie, senza il bisogno di transazioni inter-PAN. In questo caso si stima che coordinare l’associazione tra un Router e un End Device, per avere un dato rapportabile al caso di touchlink, richiederebbe statisticamente meno di un secondo [11].

Essa quindi risulta attualmente molto più veloce del touchlink, sebbene il confronto risulti anomalo in quanto i due metodi sono profondamente differenti: il touchlink è nato per essere gestito e controllato da un’utenza umana. Infatti l’applicazione Atmel è gestita in modo da dilazionare la transazione inter-PAN molto più del necessario includendo funzionalità facoltative rispetto alla specifica ZLL, tra cui l’utilizzo del tempo di identificazione (*identifyTimer*, settato a 3 secondi) entro il quale l’utente ha la possibilità di abortire la procedura rilasciando il pulsante PWR.

Sottraendo al tempo di transazione inter-PAN la durata delle suddette operazioni opzionali e delle latenze giudicate migliorabili, questa si ridurrebbe notevolmente potendo essere ragionevolmente confrontabile con i tempi di un’associazione ZigBee standard.

- Con ZLL si migliorano invece le tempistiche di gestione della rete nei casi in cui L’End Device perda il nodo Parent e debba riassociarsi ad un altro nodo Router che prenda il suo posto: questa procedura ha una durata di circa 3 secondi mentre le reti ZigBee standard impiegano di default algoritmi di retry e ricalcolo dei percorsi di routing più insistenti e pesanti che tengono conto anche del nodo Coordinator e che fanno esplodere questi tempi fino a decine di secondi a meno che non si impieghino scelte applicative di provvedimento, come segnalato dall’esperienza dell’azienda Aurel s.p.a.

7.3 Consumo ed efficienza energetica

- Sarebbe opportuno approfondire il problema, apparentemente dovuto a un malfunzionamento dello stack, a causa del quale un Controller Factory New è impossibilitato a spegnere il suo chip radio dopo una scansione avvenuta senza successo, facendolo consumare una corrente di 15 mA senza alcuna utilità.
- Inoltre, per diminuire maggiormente il consumo dei Controller durante i periodi di inattività, si può pensare di porre in sleep mode anche il microcontrollore e non solo il chip radio. Questo farebbe passare i valori di consumo tipici di queste fasi da 4,5 mA ad 1 uA ma renderebbe la gestione dei pulsanti molto più complessa.

- Si può pensare poi ad un metodo per sincronizzare lo stop di polling (*ZDO_StopSyncReq()*) del Controller con l'effettivo termine delle comunicazioni con il Lighting Device Parent, in modo da evitare l'inutile invio di Data Request, che attualmente termina con l'espirazione di *activityTimer*, sovradimensionato rispetto al necessario.

7.4 Firmware: la funzionalità di cambio canale

- La utility di cambio canale andrebbe rivista a fondo. Prima di tutto il suo innesco dovrebbe essere automatico e avvenire ogniqualvolta il canale corrente è affetto da disturbi.
- Inoltre la funzione stessa implementata nel firmware potrebbe essere migliorata controllando, prima del cambio di canale, se tutti i Lighting Device sono pronti alla procedura e notificare l'eventuale assenza di qualche dispositivo per evitare di perderne l'associazione, come invece succede attualmente con l'applicazione Atmel.
- La scelta di canale sul quale trasferirsi dovrebbe tenere conto della qualità del canale stesso utilizzando ad esempio la medesima procedura con il quale il cambio potrebbe venire innescato.

7.5 Porting e customizzazione

Al di là delle applicazioni nell'industria dell'illuminazione, si può pensare di sviluppare quindi un firmware proprietario che utilizza il touchlink come metodo di associazione, start di rete e leave basandosi sull'impronta Atmel per poi creare una versione custom snellita nelle tempistiche, migliorata nei consumi e con funzionalità intelligente di cambio automatico di canale. L'azienda Aurel s.p.a., che crea moduli wireless proprietari basati su ZigBee anche con microcontrollore ATMega128RFA1, potrebbe pensare di fare un porting tra le schede che compongono il kit di valutazione e le board aziendali, per realizzare una applicazione forte del metodo user-friendly di touchlink, che si può spendere in campi più ampi rispetto a quello per cui è stato ideato.

CONCLUSIONI

Dallo studio effettuato e dai test eseguiti è stato riscontrato che il nuovo metodo di associazione caratteristico dello standard ZLL e basato sul meccanismo di touchlink è leggero ed intuitivo sia da lato utente che da un punto di vista ingegneristico. Esso è per ora esclusivamente utilizzato in questo profilo, concepito appositamente per gestire dispositivi quali impianti di illuminazione, lampade e switch. L'associazione avviene manualmente e nodo per nodo, limitandone l'uso a reti con un ridotto numero di dispositivi ma dando al contempo il vantaggio di un'estrema praticità e adeguatezza nei campi per cui lo standard è nato. Per reti ad uso generico e con un grande numero di nodi è invece ancora preferibile l'uso classico di ZigBee per l'automatismo e le minori tempistiche di associazione e start di rete. Si ritiene tuttavia che alcune caratteristiche del touchlink come il fare a meno del nodo Coordinator, possano rappresentare a livello generale una potente alternativa alla classica gestione di rete ZigBee, appesantita da procedure ed algoritmi più complessi e con lunghi tempi di convergenza. La semplicità di installazione e monitoraggio, punto di forza di questo standard, rende i prodotti ZLL potenzialmente commercializzabili come device on-the-shelf e do-it-yourself con l'unica perplessità causata da alcune funzionalità avanzate che risulterebbero eccessivamente oscure agli utenti finali, quali, ad esempio, il cambio di canale radio manuale, che si ritiene invece dover essere automatico e non percepibile dagli utilizzatori finali.

A proposito di questa caratteristica, c’è da dire che, nonostante già nelle precedenti versioni della specifica ZigBee PRO si cercava di dare nuovi strumenti e parziali soluzioni per implementare il cambio automatico del canale radio in presenza di disturbi, questa necessità è ancora lontana dall’essere soddisfatta. La funzione di cambio canale sviluppata da Atmel all’interno del firmware dell’applicazione dimostrativa di ZLL è priva di automatismo, intelligenza e robustezza. L’impiego di questo standard migliora invece le problematiche riguardo al consumo degli End Device in certi scenari specifici. I Controller ZLL hanno chip radio normalmente spento e attivato solo durante l’effettivo utilizzo del nodo. Questi quindi consumano meno dei classici End Device ZigBee che invece svolgono attività radio periodicamente. I tempi di latenza e ricalcolo in caso di mancata risposta ad una interrogazione sono soddisfacentemente brevi, come del resto ci si aspettava da uno standard nato per applicazioni nel campo dell’illuminazione, dove è spesso richiesta l’istantaneità come requisito principale. Concludendo, ZLL presenta lati interessanti che però dovrebbero essere maggiormente sviluppati, come proposto al capitolo 7, per non rimanere confinati al restrittivo mondo delle applicazioni lighting.

APPENDICE

Firmware ZLLDemo: configuration.h

```
#ifndef _CONFIGURATION_H_
#define _CONFIGURATION_H_

//------------------------------------------------
// Disables board-specific peripherals support
//------------------------------------------------
//#define APP_DISABLE_BSP 1
#define APP_DISABLE_BSP 0
//------------------------------------------------
// Includes board-specific peripherals support in application.
//------------------------------------------------
#include <BoardConfig.h>
//------------------------------------------------
// Atmel communication interfaces identifiers.
// Supported interfaces are platform and application dependent.
//------------------------------------------------
#define APP_INTERFACE_USART 0x01
#define APP_INTERFACE_VCP 0x02
#define APP_INTERFACE_SPI 0x03
#define APP_INTERFACE_UART 0x04
#define APP_INTERFACE_USBFIFO 0x05

//------------------------------------------------
// Atmel external memory identifiers.
// Supported memory is platform and application dependent.
//------------------------------------------------
#define AT25F2048  0x01
#define AT45DB041  0x02
#define AT25DF041A 0x03

// Select ZLL device type: either Color Scene Remote or Color Light
#define APP_DEVICE_TYPE_COLOR_SCENE_REMOTE
//#define APP_DEVICE_TYPE_COLOR_LIGHT
//------------------------------------------------
//APP_DEVICE_TYPE_COLOR_LIGHT
//------------------------------------------------
#ifdef APP_DEVICE_TYPE_COLOR_LIGHT
  // Choose a way Color Ligth device displays runtime information: via LED or LCD.
  #define APP_COLOR_LIGHT_WITH_KEY_REMOTE_LCD
  //#define APP_COLOR_LIGHT_WITH_RCB_LED

  // Activate channel scanning on startup for FN Light devices
  #define APP_SCAN_ON_STARTUP 1
  //#define APP_SCAN_ON_STARTUP 0
#endif
```

```
// Enable application debug information output via USART
//#define APP_ENABLE_DEBUG 0
#define APP_ENABLE_DEBUG 1

// Fix PAN settings for debug purpose
#define APP_ZLL_FIXED_PAN 0
//#define APP_ZLL_FIXED_PAN 1

//-----------------------------------------------
//APP_ZLL_FIXED_PAN == 1
//-----------------------------------------------
#if (APP_ZLL_FIXED_PAN == 1)
  // Fixed extended PAN ID value for debug purpose
  #define APP_EXT_PANID 0xDEADBEEFCAFEACABLL

  // Fixed short PAN ID value for debug purpose
  #define APP_SHORT_PANID 0xACAB
#endif

// Size of the buffer used by the MAC component for data frames
#define CS_MAC_FRAME_RX_BUFFER_SIZE 300

// Channels set configuration

#define APP_ZLL_CHANNELS   {11, 15, 20, 25}
#define APP_ZLL_DEFAULT_WORKING_CHANNEL 11
#define APP_PRIMARY_CHANNELS_MASK             0x2108800 //standard
#define APP_SECONDARY_CHANNELS_MASK           0x5EF7000 //standard

//#define APP_ZLL_CHANNELS   {12, 16, 21, 26}
//#define APP_ZLL_DEFAULT_WORKING_CHANNEL 12
//#define APP_PRIMARY_CHANNELS_MASK             0x4211000 //standard+1
//#define APP_SECONDARY_CHANNELS_MASK           0x3DEE800 //standard+1

//#define APP_ZLL_CHANNELS   {13, 17, 19, 22}
//#define APP_ZLL_DEFAULT_WORKING_CHANNEL 13
//#define APP_PRIMARY_CHANNELS_MASK             0x4A2000 //standard+2
//#define APP_SECONDARY_CHANNELS_MASK           0x7B5D800 //standard+2

//#define APP_ZLL_CHANNELS   {14, 18, 23, 24}
//#define APP_ZLL_DEFAULT_WORKING_CHANNEL 14
//#define APP_PRIMARY_CHANNELS_MASK             0x1844000 //standard+3
//#define APP_SECONDARY_CHANNELS_MASK           0x67BB800 //standard+3

// A period in ms of polling a parent for data by an end device. On a sleeping end
// device the parameter determines a period with which poll requests are sent to
// the parent while the end device is awaken. A parent of a sleeping end device
// uses the parameter to calculate estimates of the time when the next poll request
// from a child will be received.
//  Value range: any value valid for the C-type
//  C-type: uint32_t
```

```
//  Can be set: at any time
//  Persistent: No
#define CS_INDIRECT_POLL_RATE 500

// While scanning channels during network join the node keeps listening to each
// channel specified by the ::CS_CHANNEL_MASK for a period of time calculated
// according to the formula that for the 2.4GHz frequency band is: 960 * 16 * (2
// raised to a power n + 1) microseconds, providing n is a value of this parameter.
// Note that the formula for the Sub-GHz employs another constant instead of 16.
#define CS_SCAN_DURATION 5

// Determines the maximum number of attempts to enter a network performed by the
// stack during network start. Upon each attempt ZDO sends a beacon request and
// collects beacon responses from nearby devices all over again.
#define CS_ZDO_JOIN_ATTEMPTS 1

// If the parameter being switched between 0xff and 0x00, determines whether the
// device accepts or not a child joining the network via MAC association, that is,
// if the joining device does not possess the PANID value of the network and its
// PANID parameter is set to 0.
#define CS_PERMIT_DURATION 0x00
//MAC association is on.
//#define CS_PERMIT_DURATION 0xFF

// Is used to calculate the length of time after which a not responding end device
// child is considered lost. A sleeping end device is considered lost and a
// corresponding notification is raised on the parent, if the end device does not
// polls for data for the time span which duration is calculated by the following
// formula: CS_NWK_END_DEVICE_MAX_FAILURES * (CS_END_DEVICE_SLEEP_PERIOD +
// CS_INDIRECT_POLL_RATE)
#define CS_NWK_END_DEVICE_MAX_FAILURES 4

// The parameter is used to determine the security type.
//
//  Value range: 0,3 - for standard security; 1,2 - for high security.
//  0 - network key is preconfigured ;
//  1 - network join without master key, but with a trust center link key, which
// must be set via APS_SetLinkKey();
//  2 - network join employs a master key, which must be set APS_SetMasterKey();
//  3 - network key is no preconfigured, but rather received from the trust center
// in an unencrypted frame. <br.
//  C-type: uint8_t
//  Can be set: at any time before network start
//  Persistent: Yes
#define CS_ZDO_SECURITY_STATUS 1
//#define CS_ZDO_SECURITY_STATUS 3
//#define CS_ZDO_SECURITY_STATUS 2
//#define CS_ZDO_SECURITY_STATUS 0

// Maximum amount of records in the Group Table.
//  The Group Table size cannot be 0. The group table stores pairs of a group
```

```
// address and an endpoint. Upon receiving a frame addressed to members of a
// certain group which include the current node as well the stack fires indications
// on all endpoints registered with the group address.
//  C-type: uint8_t
//  Can be set: at compile time only
//  Persistent: No
#define CS_GROUP_TABLE_SIZE 10

// Maximum amount of records in the Neighbor Table.
//
//  The parameter determines the size of the neighbor table which is used to store
// beacon responses from nearby devices. The parameter puts an upper bound over the
// amount of child devices possible for the node.
//
//  Value range: at minimum 1, the maximum value is limited to the available memory
//  C-type: uint8_t
//  Can be set: at compile time only
//  Persistent: No
#define CS_NEIB_TABLE_SIZE 10

// Maximum amount of records in the network Route Table.
//
//  The parameter sets the maximum number of records that can be kept in the NWK
// route table. The table is used by NWK to store information about established
// routes. Each table entry specifies the next-hop short address for a route from
// the current node to a given destination node. The table is being filled
// automatically during route discovery. An entry is added when a route is
// discovered.
//
//  Since the end device always sends a frame directly to its parent its route
// table size should be set to 0.
//
//  C-type: uint8_t
//  Can be set: at compile time only
//  Persistent: No
#define CS_ROUTE_TABLE_SIZE 10

// Maximum amount of records in the network Route Discovery Table.
//
//  The parameter specifies the size of the route discovery table used by NWK to
// store next-hop addresses of the nodes for routes that are not yet established.
// Upon exhausting the capacity of the table, the stack starts rewriting old
// entries. If the size of the route table is big enough after all used routes are
// established the table may not be used.
//
//  Since the end device always sends a frame directly to its parent its route
// discovery table size should be set to 0.
//
//  C-type: uint8_t
//  Can be set: at compile time only
//  Persistent: No
```

```
#define CS_ROUTE_DISCOVERY_TABLE_SIZE 10

// The number of buffers for data requests on the APS layer.
//  The parameter specifies the number of buffers that are allocated by APS to
// store data requests parameters. The parameter puts an upper bound to the number
// of data requests that can be processed by APS simultaneously. If all buffers are
// in use and a new data request appears, it is kept in a queue until a buffer is
// released.
//
//  C-type: uint8_t
//  Can be set: at compile time only
//  Persistent: No
#define CS_APS_DATA_REQ_BUFFERS_AMOUNT 4

// The number of buffers for acknowledgement messages sent by APS.
//
//  This parameter determines the amount of memory that needs to be allocated for a
// special type of buffers used by APS to store payloads for acknowledgement
// frames. The need to use the buffers occurs when the node receives a frame that
// has to be acknowledged. That is, the APS component on the node has to send an
// acknowledgement frame. For frames initiated by the application, the memory for a
// payload is to be allocated by the application on its own, while the payload
// memory for an acknowledgement frame shall be reserved by APS. The request
// parameters are still stored in the data request buffers.
//
//  Typically, a value of this parameter equals CS_APS_DATA_REQ_BUFFERS_AMOUNT - 1.
//
//  C-type: uint8_t
//  Can be set: at compile time only
//  Persistent: No
#define CS_APS_ACK_FRAME_BUFFERS_AMOUNT 3

// The maximum number of retries that will be performed by APS layer before
// reporting failed transmission.
//
//  The parameter sets the number of attempts that will be made by APS layer to
// transmit a data frame. If all these attempts fail due to underlying layers
// failures, then APS response with an error status.
//  C-type: uint8_t
//  Can be set: at any time before network start
//  Persistent: No
#define CS_APS_MAX_FRAME_RETRIES 1

// If the parameter is set to true multicasting on the NWK level is used,
// otherwise, multicasting on the APS level is applied. The parameter is
// recommended to be set to true. For detail refer to ZigBee specification.
#define CS_NWK_USE_MULTICAST false
//#define CS_NWK_USE_MULTICAST true


#endif // _CONFIGURATION_H_
```

INDICE DELLE FIGURE

BIBLIOGRAFIA E SITOGRAFIA

[1] Wikipedia, the free encyclopedia, «en.wikipedia.org/wiki/ZigBee,» [Online]. Available: http://en.wikipedia.org/wiki/ZigBee.

[2] ZigBee Alliance, «www.zigbee.org,» [Online]. Available: http://www.zigbee.org/.

[3] D. Dardari e M. Lucchi, *Tecnologies for wireless sensor network,* University of Bologna at Cesena, 2012.

[4] ZigBee Alliance, «www.zigbee.org,» [Online]. Available: https://docs.zigbee.org/zigbee-docs/dcn/12/docs-12-0255-01-0mwg-exploring-new-opportunities-with-zigbee-light-link.pdf.

[5] AUREL s.p.a., «www.aurel.it,» [Online]. Available: http://www.aurel.it/.

[6] AUREL s.p.a., «www.aurelwireless.com,» [Online]. Available: http://www.aurelwireless.com/wp-content/uploads/aurel_company-profile_it.pdf.

[7] AUREL s.p.a., «www.aurel-zigbee.com,» [Online]. Available: http://www.aurel-zigbee.com/.

[8] Atmel, «www.atmel.com,» [Online]. Available: http://www.atmel.com/tools/rf4ce-ek.aspx.

[9] IAR systems, «www.iar.com,» [Online]. Available: http://www.iar.com/.

[10] Texas Instruments, «www.ti.com,» [Online]. Available: http://www.ti.com/.

[11] T3lab, «http://www-micrel.deis.unibo.it/~t3lab/PDF/TR007-07.pdf,» [Online]. Available: http://www-micrel.deis.unibo.it/~t3lab/PDF/TR007-07.pdf.

RINGRAZIAMENTI

Seguono i ringraziamenti personali dell’autore.
“Ringrazio mio padre e mia madre per avermi spronato allo studio, per avermi incoraggiato all’impegno, per avermi insegnato ad avere sete di conoscenza. Senza i loro sacrifici ed il loro sostegno morale ed economico, forse non sarei arrivato fino a qui. Ringrazio i miei fratelli Emanuele ed Eleonora, che prima di me hanno conseguito una laurea dandomi la possibilità di vedere nelle loro vite un esempio e uno spunto per quello che si prospetta essere il mio futuro, per essere stati faro nella notte e pionieri nella vita, spianandomi spesso la strada in modo naturale ed affettuoso. Ringrazio i miei piccoli nipoti Francesco e Chiara, per la loro purezza, creatività geniale e tenerezza. Avendo dedicato la tesi della mia laurea triennale all’allora appena nato Francesco, dedico questa tesi a Chiara. Un ringraziamento speciale è rivolto a mio zio Claudio che mi ha messo in contatto con l’azienda Aurel s.p.a. Ringrazio poi Franco Perugini per aver accolto la mia richiesta di tirocinio in azienda, l’ing. Mattia Visani, mentore della mia attività su ZigBee in Aurel e figura di riferimento per dubbi e difficoltà che ho incontrato durante l’attività. Ringrazio tutti i dipendenti Aurel che ho conosciuto e che mi hanno ospitato calorosamente in quello che si è dimostrato essere un qualificato e gradevole ambiente di lavoro. Ringrazio il Prof. Davide Dardari per la disponibilità dimostrata nel accettare il ruolo di relatore per questa tesi e la tempestività nelle risposte via mail a miei quesiti e richieste di revisione dell’elaborato. Un grazie è rivolto a tutti i professori che in questi anni hanno saputo trasmettermi passione ed interesse per l’elettronica e le telecomunicazioni.

Ringrazio l’università di Bologna in particolar modo per avermi dato l’opportunità di usufruire del programma Erasmus. Ringrazio la DTU di Copenhagen e tutti gli amici con cui ho condiviso esperienze di crescita immensa durante gli indimenticabili cinque mesi che ho trascorso in Danimarca: Daniele, Moussa, Paolo, Simone, Michele, Francesco, Sylvain, Krisha, Steven, Christof, Emily, Ezgi, Claudia, Fabio e tutti i ragazzi che hanno reso il mio soggiorno all’estero così prezioso ed unico. Ringrazio Eleonora per essermi stata accanto con pazienza in questo periodo, rispettando i miei ritmi e i miei impegni, facendomi coraggio e compagnia nei momenti più duri e gioendo con me in quelli più belli, dimostrandomi il grande affetto e l’amore di cui una fedele amica e dolce compagna è capace. Ringrazio l’altra mia grande compagna, passione ed espressione della mia vita: la musica. In particolare ringrazio la band The Same Old Shoes per avermi insegnato a credere nelle mie idee e nelle mie capacità creative, usarle per realizzare progetti con metodo e perseveranza, rispettare gli impegni presi e godere dei i risultati ottenuti insieme. Ringrazio quindi i miei colleghi e amici musicisti: Mattia, Davide, Filippo, Mattia, Fabrizio e tutti gli amici con cui ho condiviso il palco in questi anni, poi i mastri Elvis Presley, Johnny Cash, Buddy Holly, Hank Williams, Son House, Muddy Waters, Howlin Wolf, Robert Johnson, John Lennon, Paul McCartney, George Harrison, Ringo Starr, Roger Daltrey, Pete Townshend, John Entwistle, Keith Moon, Robert Plant, Jimmy Page, John Paul Jones, John Bonham, John e Tom Fogerty, Stu Cook, Douglas Cosmo Clifford e tanti altri.

Ringrazio le aule studio e il circolo arci Valverde di Forlì, la biblioteca Ruffilli, l’associazione Koinè, Microsoft Paint, la rosticceria Miky’s Pizza di Cesena, la rosticceria la Mela di Forlì, la pizzeria al taglio Altero di Forlì, la pizzeria del Sole di Forlì, la Feltrinelli di Forlì, la Conad di Corso Diaz a Forlì, il parcheggio dell’ospedale Villa Serena di Forlì, il parcheggio del Lice Artistico di Forlì, il parco urbano di Forlì In fine, ma non meno importanti, gli amici di sempre con cui sono in cammino da anni verso il futuro: Luca, Riccardo, Jacopo, Beppe, Lorenzo, Alessandro, Nicola, Federico, Tomas, Mattia, Alessio, Chiara, Erica, Caterina, Stefania, Alice, Carlotta, Michela, Lucia, Maria Chiara, Anna, Angelita, Elena, Margherita.”