Università degli Studi di Padova
Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria dell'Informazione

Tesi di laurea triennale

Uno studio sul consumo energetico nelle batterie per sensori wireless

A study on energy consumption in wireless sensor node batteries

Candidato:
Stefano Pretto
Matricola 609696

Relatore:
Prof. Leonardo Badia

Correlatore:
Dott. Riccardo Manfrin

Anno Accademico 2011–2012

a mio padre,

che mi ha indicato la strada,

e a mio nonno:

possiate essere orgogliosi di me

Indice

Elenco delle figure

Elenco delle tabelle

Sommario

Lo scopo di questo lavoro è eseguire uno studio del consumo energetico di un sensore di rete wireless, alimentato a batterie, per mezzo di simulazioni numeriche. Per fare questo verrà presentato ed analizzato il protocollo MAC utilizzato dal sensore (X-MAC), il sensore stesso, e le batterie. Quindi verrà progettato un modello di simulazione probabilistico basato su una semplice macchina a due stati, poi implementato tramite un programma scritto in Java. Infine verranno esposti e commentati i risultati ottenuti.

Abstract

This thesis is a study of energetic consumption for a wireless sensor network, supported by numerical simulation. The MAC protocol of the sensor (X-MAC), its battery characterization, and the sensor itself are introduced. A probabilistic simulation model is investigated, based on a two-state machine, and it is implemented in a software written in Java. In the end, the obtained results are presented and discussed.

Capitolo 1

Introduzione

L'efficienza energetica è uno dei temi più importanti in tutti i campi dell'ingegneria e, soprattutto ai giorni nostri, uno dei più studiati e dibattuti [4]. Nel mondo dell'elettronica e delle telecomunicazioni questo è particolarmente vero, in quanto si ha spesso a che fare con dispositivi alimentati da batterie, che per definizione hanno una carica di energia limitata, il cui uso efficiente è cruciale per un ottimo funzionamento. Questo obbiettivo può essere perseguito lavorando sul lato fisico del dispositivo, cercando quindi tecniche e tecnologie che permettano minori consumi a parità di lavoro svolto, oppure sul lato software, studiando programmi che comportino meno operazioni da fare, o che limitino quelle più dispendiose in termini di energia.

Le reti di sensori wireless (*wireless sensor networks* - WSN) sono un argomento di discussione e studio che negli ultimi anni è diventato sempre più presente (basti guardare all'interno della stessa Università di Padova, ad esempio [6]), e nel quale vi è una parte preponderante di ricerca riguardante i consumi enrgetici.

Sono reti composte da sensori autonomi (chiamati nodi), distribuiti su un'area con lo scopo di raccogliere dati e monitorare determinate condizioni ambientali (ad esempio la temperatura, l'umidità e la pressione, ma anche il livello di inquinamento atmosferico o altri parametri complessi [9]). I dati raccolti vengono passati da un nodo all'altro con un sistema a salti tra nodi adiacenti (*multi-hop*), fino a raggiungere il nodo di raccolta (*sink node*), che li trasferisce all'utente tramite internet. Il fatto che i nodi siano wireless e autonomi porta a molti vantaggi [9],

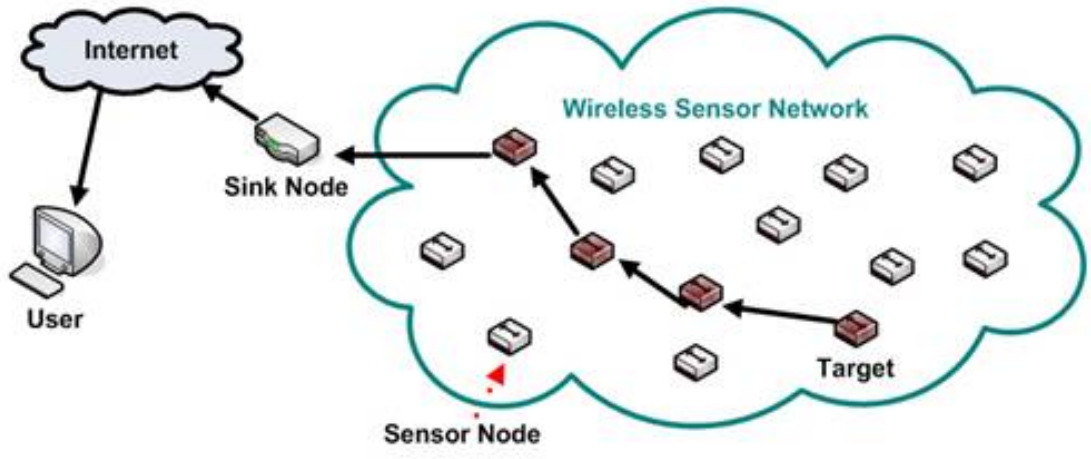


Figura 1.1: Semplice schema dell'architettura di una WSN

come ad esempio la facilità di installazione della rete, che non necessita cablaggi e quindi lavori onerosi, ma anche ad un evidente problema: i sensori devono essere alimentati da batterie, e quindi hanno un periodo di funzionamento limitato. Il problema dei consumi energetici diventa allora di fondamentale importanza, e con esso lo studio di metodi per limitarlo il più possibile.

Uno degli approcci più seguiti è lo studio di protocolli di accesso al mezzo (*Medium Access Control (MAC) protocols*) ottimizzati [8, 17], tra i quali troviamo il protocollo X-MAC [5], usato dal sensore di cui tratteremo in questa tesi. Una delle sue caratteristiche più importanti è proprio quella di consentire un ottimo risparmio sui consumi energetici dei sensori, come testimoniato dall'articolo di Suarez[21], cosa che lo rende un candidato ideale per uno studio in questo senso.

Il resto di questa tesi è organizzato come segue.

Nel secondo capitolo viene presentato lo stato dell'arte per quanto riguarda la ricerca e l'analisi sui consumi energetici nelle WSN.

Nel terzo capitolo vengono presentate e modellizzate le caratteristiche di un nodo reale e delle sue batterie, e progettata una simulazione numerica per la valutazione dei consumi dovuti a trasmissioni basate su protocollo X-MAC [5].

Nel quarto capitolo vengono presentati e commentati i risultati ottenuti.

L'appendice A riporta il codice sorgente Java scritto per la simulazione.

Capitolo 2

Lavori correlati e presentazione dell'X-MAC

Il problema del consumo energetico nelle WSN è uno dei punti chiave per l'effettivo successo di questa tecnologia, e sarà determinante per decretare se queste reti avranno un'applicazione sempre maggiore nel prossimo futuro.

Gli studi in questo senso si sono sviluppati seguendo approcci variegati: ad esmepio, Schurgers e Srivastava [18] si sono concentrati sullo studio del metodo di aggregazione dei dati e sullo sviluppo di differenti tecniche di instradamento dei dati nei nodi (*routing*), ottenendo risparmi fino al 90% rispetto alla situazione iniziale.

Tuttavia, l'approccio più seguito è lo studio di protocolli di accesso al mezzo (*MAC protocols*), in particolare quelli adottanti una routine di passaggio periodico del sensore da una fase di *sleep* ad una di attività e viceversa, chiamato *duty cycling*. In questo modo, il consumo totale è dato dalla somma del consumo durante il periodo di *sleep* e da quello durante il periodo di attività: come è facile notare, variando le lunghezze dei periodi è possibile far consumare più o meno energia per ogni ciclo, e quindi diminuire o estendere la durata della batteria utilizzata per alimentare un sensore con questo protocollo. Di conseguenza, la priorità diventa quella di ridurre il più possibile il tempo di attività, composto tra le altre parti da un periodo di ascolto del canale in assenza di trasmissioni (chiamato *idle listening*): questo ultimo è stato scoperto essere uno dei maggior fattori di consumo energetico [13, 19], che porta ad avere come obbiettivo l'evitare la sua comparsa.

Possiamo allora distinguere i protocolli MAC per reti di sensori wireless in due categorie: sincroni e asincroni [5]. I primi evitano completamente il fenomeno di *idle listening*, in quanto la fase di attività è simultanea per tutti i sensori, impedendo che uno di essi si attivi mentre tutti gli altri sono in *sleep* e quindi non possa trasmettere, nè ricevere nulla. Tuttavia, questo metodo comporta evidentemente che tutti i sensori della rete siano sincronizzati ad una stessa *schedule*. Viceversa, i protocolli asincroni non hanno questa limitazione, ma rischiano di portare alcuni sensori a periodi di *idle listening* molto lunghi in rapporto al loro *duty cycle*, e devono essere quindi progettati opportunamente.

Per quanto riguarda il primo caso possiamo citare il lavoro di Wu [22], in cui viene sviluppata una *schedule* per la raccolta dei dati che porta ad avere consumi per reti omogenee al massimo doppi rispetto all'ottimo, e nel quale vengono fatte diverse simulazioni numeriche per la valutazione dell'algoritmo. Ma data la sua maggiore appetibilità applicativa, è il secondo caso ad offrire una maggiore varietà di studi.

I protocolli MAC asincroni sono tanto più energeticamente efficienti, quanto più riescono a mantenere un basso periodo di *idle listening*. Miller [16] propone un'architettura del sensore a doppia radio, in cui viene utilizzato un segnale per svegliare i nodi adiacenti (*busy tone*) e permettere l'immediata sincronizzazione tra mittente e destinatario; questa soluzione tuttavia fa sì che vengano attivati anche sensori non implicati nella trasmissione, con i consumi che ne conseguono. S-MAC [23] è uno dei primi protocolli basati su *duty cycle* ad essere stati sviluppati per le reti di sensori wireless, ed è costituito da un periodo fissato di attività e uno variabile di *sleep*. All'inizio di quello di attività, i nodi devono scambiarsi informazioni per la sincronizzazione, e successivamente possono essere trasmessi i dati, nel rimanente tempo disponibile. S-MAC può raggiungere bassi livelli di assorbimento energetico, ma tuttavia deve mantenere una *schedule* dei nodi vicini, caratteristica che diventa di rilievo quando la rete aumenta di dimensioni e che, tra l'altro, mostra come S-MAC non sia un protocollo propriamente asincrono. Successivamente viene sviluppato T-MAC [8], con l'obbiettivo di migliorare S-MAC nell'ambito del

consumo energetico, ma eredita dal predecessore la maggior parte dei problemi che lo affliggevano.

B-MAC [17], sviluppato all'Università della California a Barkeley, è contraddistinto dal *low power listening* (LPL), chiamato anche *preamble sampling*, ossia il metodo usato per collegare tra loro un sensore che ha dati da mandare (*sender*) con il rispettivo sensore destinatario (*receiver*): questo avviene tramite l'invio da parte del *sender* di un preambolo atto a far sapere agli altri sensori di aver qualcosa da trasmettere, in modo che quando il *receiver* esce dalla fase di *sleep* riconosca di dover rimanere in ascolto del canale fino alla ricezione dei dati. Ovviamente, per avere la sicurezza di una trasmissione effettiva, l'invio del preambolo deve durare almeno per l'intero periodo di *sleep* del sensore destinatario, anche se c'è la possibiltà che questo si svegli, e quindi sia pronto a ricevere, già all'inizio della trasmissione, con conseguente consumo inutile da entrambe le parti (per l'invio continuo di preambolo inutile e per l'ascolto continuo del preambolo). Inoltre, si ha possibile consumo anche da parte di altri sensori della rete non implicati nello scambio, in quanto, nel caso si sveglino durante la fase di collegamento, riconosciuta la presenza del preambolo devono rimanere in ascolto fino all'inizio della trasmissione dei dati, prima di scoprire di non essere loro gli effettivi destinatari. Questo problema è chiamato *overhearing*. WiseMAC [12], basato su Aloha, usa tecniche simili a B-MAC, ma con la differenza che in questo caso il *sender* impara le routine degli altri nodi e imposta le successive comunicazioni di conseguenza. Tuttavia, non permette la presenza di un meccanismo per adattare i nodi a cambiamenti nel traffico dei dati.

Con l'obbiettivo di correggere questi difetti, viene quindi sviluppato il protocollo X-MAC [5], che utilizza un preambolo più corto e pacchettizzato (*strobed preamble*), contenente inoltre l'indirizzo del destinatario. Il sensore manda quindi pacchetti minimi di preambolo, e dopo ogni pacchetto si mette in ascolto del canale per ricevere una eventuale segnalazione di ricezione da parte del *receiver* (pacchetto di *acknowledgment*, o ACK), che quindi afferma di essere pronto a ricevere i dati. Inoltre si evita l'*overhearing*, visto che gli altri sensori, leggendo l'indirizzo

nel preambolo, hanno la sicurezza di poter tornare in *sleep* senza il rischio di perdere dati a loro destinati. Questo metodo permette ai due sensori di trasmettere i dati non appena il ricevitore si è svegliato, ha ricevuto un pacchetto di preambolo completo e ha mandato il suo ACK. Inoltre, nel peggiore dei casi, il consumo da parte del *sender* sarà equivalente a quello dovuto al preambolo lungo del B-MAC, in quanto al massimo trasmetterà pacchetti di preambolo per tutta la durata dello *sleep* del destinatario. Infine, migliora WiseMAC implementando meccanismi di adattamento al traffico, che però non analizzeremo in questo lavoro.

A latere a questi studi, si pone però anche il problema della valutazione del consumo energetico. A questo scopo, Shnayder [20] ha messo a punto un simulatore per analizzare i consumi di applicazioni sviluppate per WSN, con l'obbiettivo di fornire un ulteriore strumento ai progettisti. In alternativa, Dunkels [11] ha studiato un software che permette la valutazione online del consumo istantaneo dei sensori di una rete, permettendo valutazioni empiriche che in precedenza necessitavano di componenti aggiuntivi da installare al sensore. AEON [15] è un altro *tool* per sviluppatori, che simula i consumi di una rete basandosi sulle correnti assorbite dai sensori durante l'esecuzione dei programmi da valutare.

Studi predittivi sui consumi sono invece quelli fatti da Coleri [7], che mostra come modellizzare TinyOS (sistema operativo appositamente progettato per i sensori) come un automa ibrido, e usare questo modello per calcolare i consumi previsti: un modello paragonabile all'automa verrà usato in questo lavoro, per simulare il comportamento di una generica applicazione del sensore. Duarte-Melo [10] invece analizza le performance e i consumi in una WSN al variare della sua topologia, della potenza e del numero dei sensori.

Capitolo 3

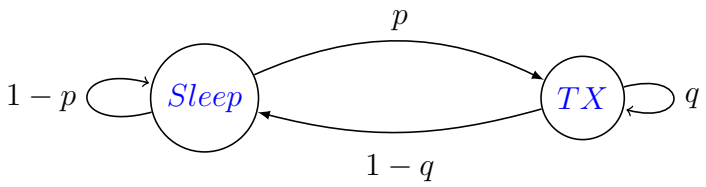
La simulazione

3.1 L'idea

L'idea di fondo di questo studio è quella di analizzare tramite simulazioni la durata delle batterie al variare di differenti carichi di lavoro del sensore trasmittente, con l'utilizzo del protocollo X-MAC.

Per fare questo, ipotizziamo che il nostro *sender* trasmetta e basta, senza mai dover ricevere nulla, e che quindi non abbia una routine di ascolto del canale per la ricerca di trasmissioni entranti; possiamo allora pensare che il programma fatto girare sul sensore lo possa portare in soli due differenti stati: *sleep* e trasmissione.

Il suo funzionamento può quindi essere sintetizzato da una macchina a stati [14], in questo modo:



La nostra simulazione consisterà allora nel generare dei pattern di funzionamento basati sulle probabilità p e q e, conoscendo il consumo energetico del sensore per i due stati, calcolare la durata di vita della batteria a seguito dell'esecuzione dei pattern creati.

3.2 Il sensore

Il sensore considerato è un progetto in via di sviluppo, le cui componenti principali (nonchè quelle più rilevanti ai fini della simulazione) sono il microcontrollore LPC1768, prodotto da NXP Semiconductors e basato su architettura ARM Cortex-M3, e il modulo di trasmissione radio AT86RF231, prodotto da ATMEL. Da specifiche, questi elementi necessitano di un'alimentazione di 3.0 V, con una soglia limite di funzionamento fissata a 2.8 V, e il modulo radio trasmette ad una velocità di 250 kbps. Le trasmissioni seguono lo standard *IEEE 802.15.4* [2].

3.3 Le batterie

Nonostante lo scopo finale sia quello di condurre una simulazione numerica, prendiamo in esame batterie reali, ovvero una coppia di pile stilo ricaricabili *Beghelli Carica 500*, di tipo AA, a tecnologia NiMh (*Nickel Metal Hydride*), i cui dati nominali dichiarati sono un voltaggio pari a 1.2 V, una capacità di 1200 mAh ciascuna [1].

Per verificare la loro effettiva capacità totale (per alimentare il sensore servirà un voltaggio di circa 3 V, per cui le due pile saranno messe in serie), carichiamo completamente le batterie, e le facciamo scaricare cortocircuitandole ad un carico puramente resistivo del valore di 155 Ω. Per effettuare le misurazioni, utilizziamo il convertitore AD del sensore stesso, opportunamente collegato alle batterie e opportunamente programmando il microcontrollore, in modo che faccia una misurazione del voltaggio ogni minuto. I dati raccolti sono quindi stati elaborati tramite il software MATLAB (*release* 2011) [3], ottenendo il grafico in figura 3.1.

Innanzitutto si può notare come il valore effettivo di tensione a batterie cariche (che si aggira attorno ai 3.25 V) sia molto più elevato di quello dichiarato dalla casa produttrice ($1.2\text{ V} \cdot 2 = 2.4\text{ V}$), cosa che ci consente di utilizzarle per alimentare il sensore. Inoltre, la doppia curva indica che una delle due pile ha esaurito la sua carica prima dell'altra (a circa 72 h), portando la linea del grafico ad abbassarsi a

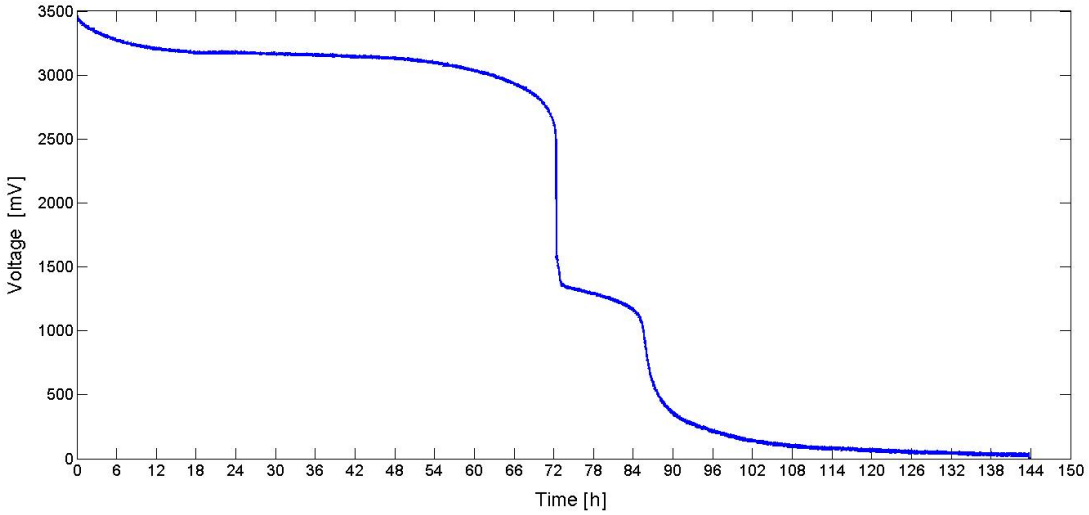


Figura 3.1: Andamento della differenza di potenziale ai capi della serie delle due batterie, scaricate su una resistenza da 155 Ω.

circa metà della tensione a regime di piena carica.

Per calcolare la carica utile effettiva allora, integriamo il grafico fino al punto in cui il voltaggio scende sotto la soglia dei 2.8 V (momento in cui il sensore smetterebbe di funzionare) e dividiamo il risultato per la resistenza moltiplicata per 60, in modo da ottenere un valore in mAh. Facendo questi calcoli tramite MATLAB, si giunge al risultato di 1418 mAh, valore più alto di quello dichiarato (1200 mAh) come era lecito attendersi dopo le considerazioni precedenti.

3.4 I consumi energetici

Per il calcolo dei consumi energetici, faremo riferimento a quelli riportati dai *datasheet* del microntrollore e del modulo radio, ovvero quelli riportati nelle tabelle 3.1, 3.2 e 3.3.

Tabella 3.1: Consumi del microcontrollore

LPC1768 power modes	I [mA]	V [V]	P [mW]	I @3.0 V [mA]
Active	42	3.3	138.6	46.2
Power-down	0.031	3.3	0.1023	0.0341

Tabella 3.2: Consumi del modulo radio

AT86RF231 activity	I [mA]	V [V]	P [mW]	I @3.0 V [mA]
TX	14	3.0	42	14
RX	12.3	3.0	36.9	12.3
Sensing	12.3	3.0	36.9	12.3
Sleep	0.000002	3.0	0.000006	0.000002

Tabella 3.3: Consumi nelle transizioni

LPL State transitions timings	[mAh/transition] @3.0 V	[µs/transition]
Active to Sleep	$4.41562500138889 \cdot 10^{-7}$	52,25
Sleep to Active	$7.59736116666667 \cdot 10^{-6}$	1005

3.5 Modellizzazione degli stati

Per poter utilizzare la macchina a stati nella simulazione, dobbiamo quindi modellizzare questi ultimi in modo da poter determinare esattamente il consumo di tempo ed energia per ognuno di essi.

3.5.1 Stato *Sleep*

Per quanto riguarda il tempo che il sensore passerà nello stato, in questo caso è del tutto arbitrario, e per la nostra simulazione sceglieremo un valore di 1 secondo.

$$t_{Sleep} = 1 \text{ s}$$

In questo stato, sappiamo che il microprocessore è in *Power-down*, mentre la radio si trova in *Sleep*, per cui il consumo energetico è pari a

$$Consumo_{Sleep} = (Power-down \ + \ Sleep) \text{ mA } \cdot t_{Sleep} = 0.034102 \text{ mA } \cdot \frac{1}{3600} \text{ h} =$$

$$= 9.473 \text{ nAh}$$

Siccome questo stato è anche sicuramente quello di partenza, non dobbiamo aggiungere consumi per transizioni acceso/spento o viceversa, che invece conteremo per lo stato *TX*.

3.5.2 Stato TX

Per questo stato, i calcoli da fare sono più complessi.

Per prima cosa, dobbiamo capire le azioni che vengono compiute, dovute al protocollo X-MAC.

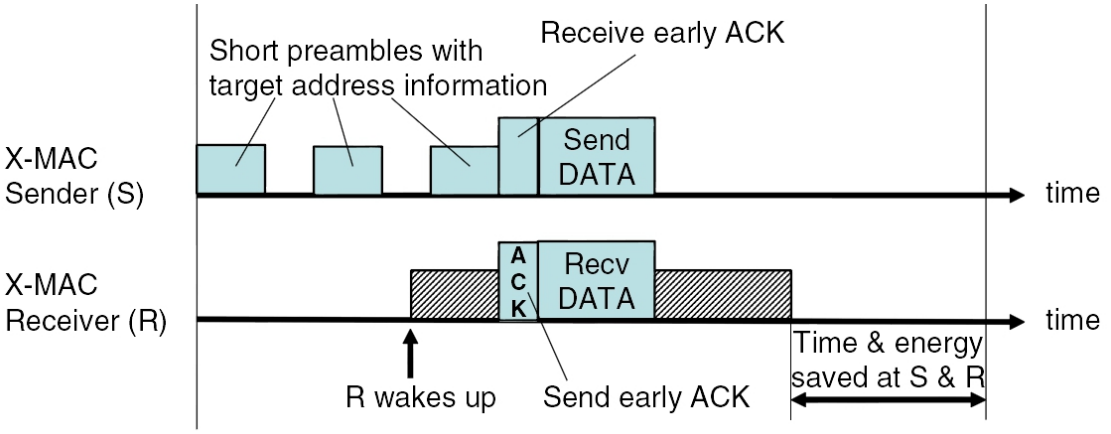


Figura 3.2: Visualizzazione grafica del protocollo X-MAC con *short and strobed preamble*, tratta da [5]

Notiamo dalla figura 3.2 che il *sender*, per la prima parte, passa continuamente da un periodo di trasmissione ad uno di ascolto del canale, fino alla ricezione del pacchetto ACK. Da quel momento, trasmette i suoi dati, e una volta conlusa la trasmissione torna in *sleep*.

Andiamo allora a capire come sono costruiti il preambolo, il pacchetto ACK, e quelli contenenti i dati, secondo lo standard *IEEE 802.15.4* seguito dal sensore, in modo da poter calcolare i tempi necessari alla radio per la trasmissione di ciascuno, e quindi calcolare i consumi.

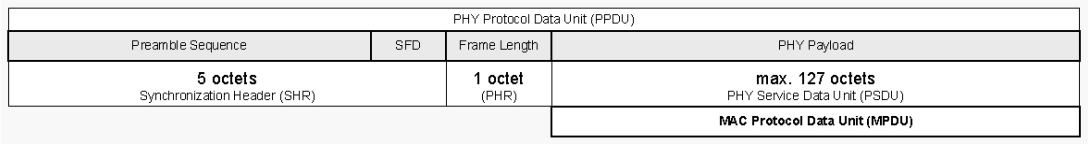


Figura 3.3: IEEE 802.15.4 Frame Format - PHY-Layer Frame Structure (PPDU)

Le immagini 3.3 e 3.4 (tratte dal *datasheet* del modulo radio) ci permettono di ricostruire i pacchetti citati in precedenza, e quindi calcolare la loro lunghezza in *Byte*. Ai fini del nostro lavoro, ci mettiamo nelle ipotesi che per il campo *Addressing* siano sufficienti 4 B, non ci sia *Auxiliary Security Header*, e un gruppo di dati utili

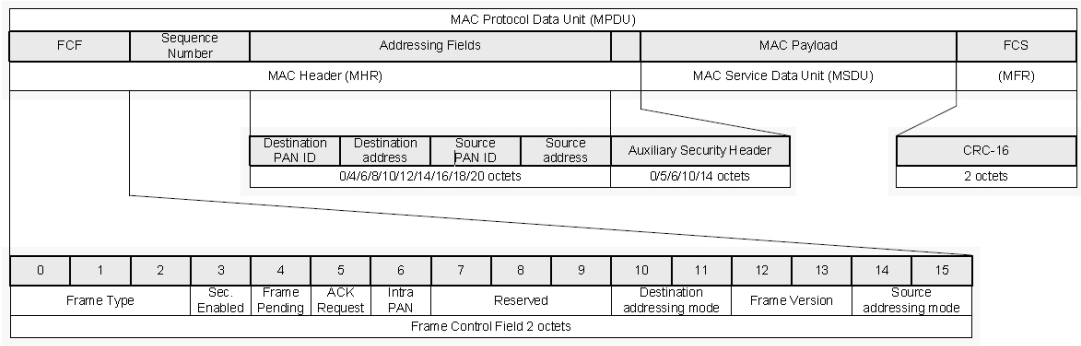


Figura 3.4: IEEE 802.15.4 Frame Format - MAC-Layer Frame Structure (MPDU)

da trasmettere sia lungo 116 B (ovvero 29 simboli, il numero massimo di simboli allegabili ad un solo PPDU).

$$L_{preambolo} = SHR + PHR + FCF + SequenceNumber + Addressing + FCS =$$

$$= (5 + 1 + 2 + 1 + 4 + 2) \text{ B} = 15 \text{ B}$$

$$L_{dati} = SHR + PHR + FCF + SequenceNumber + Addressing + Payload + FCS =$$

$$= (5 + 1 + 2 + 1 + 4 + 116 + 2) \text{ B} = 131 \text{ B}$$

$$L_{ACK} = SHR + PHR + FCF + SequenceNumber + FCS =$$

$$= (5 + 1 + 2 + 1 + 2) \text{ B} = 11 \text{ B}$$

Dalle lunghezze, sapendo la velocità di trasmissione della radio ($R = 250$ kbps), possiamo calcolare il tempo effettivo impiegato per ciascuno:

$$t_{preambolo} = \frac{L}{R} = \frac{15 \cdot 8 \text{ b}}{250 \text{ kbps}} = 480 \text{ µs}$$

$$t_{dati} = \frac{L}{R} = \frac{131 \cdot 8 \text{ b}}{250 \text{ kbps}} = 4192 \text{ µs}$$

$$t_{ACK} = \frac{L}{R} = \frac{11 \cdot 8 \text{ b}}{250 \text{ kbps}} = 352 \text{ µs}$$

Il preambolo viene mandato fino alla ricezione dell'ACK: ipotizzando un periodo di *sensing* del ricevitore di 400 ms, è lecito affermare che in media l'ACK verrà mandato esattamente a metà di questo periodo, in quanto non vi è alcuna correlazione tra i due sensori. Sapendo che, secondo le specifiche del modulo radio,

il tempo di attesa per il ricevimento dell'ACK da parte del *sender* è di 32 µs, possiamo calcolare quanti cicli preambolo/ascolto debbano essere compiuti in media, ovvero

$$t_{ciclo} = t_{preambolo} + 32\ \mu\text{s} = 512\ \mu\text{s}$$

$$Cicli = \frac{\text{attesa media}}{t_{ciclo}} = \frac{200\ \text{ms}}{(0.480 + 0.032)\ \text{ms}} = 390.625 \simeq 390$$

Verranno quindi compiuti 390 cicli di preambolo e ascolto a vuoto, e un ciclo di preambolo e ricevimento dell'ACK.

Ora possiamo calcolare il tempo medio passato nello stato TX per ogni transizione, ricordandoci di sommare anche i contributi dovuti alle transizioni di stato (*sleepToActive* e *activeToSleep*).

$$\tilde{t}_{TX} = 390 \cdot t_{ciclo} + (t_{ciclo} + t_{ACK}) + t_{dati} + t_{transizioni} =$$

$$= 390 \cdot (480 + 32)\ \mu\text{s} + (480 + 32 + 352)\ \mu\text{s} + 4192\ \mu\text{s} + 1005\ \mu\text{s} + 52.25\ \mu\text{s} =$$

$$= 205.79325\ \text{ms}$$

Per ricavare i consumi infine, dobbiamo moltiplicare il consumo in mA di ogni sottostato (trasmissione per il preambolo e dati, ascolto per le pause e l'ACK) per il tempo passato in esso. Abbiamo quindi

$$Cons._{preambolo} = (Active + TX)\ \text{mA} \cdot t_{preambolo} =$$

$$= (46.2 + 14)\ \text{mA} \cdot \frac{480 \cdot 10^{-6}\ \text{s}}{3600} = 8.027\ \text{nAh}$$

$$Cons._{dati} = (Active + TX)\ \text{mA} \cdot t_{dati} =$$

$$= (46.2 + 14)\ \text{mA} \cdot \frac{4192 \cdot 10^{-6}\ \text{s}}{3600} = 70.1\ \text{nAh}$$

$$Cons._{ascolto} = (Active + sensing)\ \text{mA} \cdot 32\ \mu\text{s} =$$

$$= (46.2 + 12.3)\ \text{mA} \cdot \frac{32 \cdot 10^{-6}\ \text{s}}{3600} = 0.52\ \text{nAh}$$

$$Cons._{ACK} = (Active + RX)\ \text{mA} \cdot t_{ACK} =$$

$$= (46.2 + 12.3)\ \text{mA} \cdot \frac{352 \cdot 10^{-6}\ \text{s}}{3600} = 5.72\ \text{nAh}$$

da cui, sommando opportunamente, otteniamo il consumo medio nello stato TX per transizione:

$$Cons._{TX} = 390 \cdot Cons_{preambolo+ascolto} + Cons_{preambolo+ascolto+ACK} + Cons_{dati} =$$

$$= 3.417697 \ \mu\text{Ah}$$

In realtà andrebbero sommati anche i consumi dovuti alle transizioni, ma per semplicità e chiarezza di calcolo (dato che i relativi valori sono molto minori di tutti gli altri consumi) non li riportiamo qua. Tuttavia ne teniamo ben presente nel codice sorgente per le simulazioni.

Capitolo 4

Risultati

4.1 Scenario

Ora abbiamo tutti i dati necessari per progettare ed eseguire la simulazione. Data la necessità di generare delle transizioni probabilistiche, ho deciso di scrivere un breve programma in Java, il cui codice sorgente si può trovare nell'appendice A, in cui sfruttare la funzione di generazione pseudocasuale dei numeri.

Il nostro obbiettivo è quello di analizzare la durata delle batterie di un sensore (che supponiamo debba trasmettere solamente) all'interno di una rete, con il modello di macchina a stati basato sui consumi del sensore reale e del protocollo X-MAC descritto precendentemente (e quindi al variare delle probabilità di passare a trasmettere, e di trasmettere più pacchetti).

Esaminando il codice, si può vedere come la simulazione crei un pattern di transizioni casuale, basato sulle probabilità p (passaggio da *sleep* a TX) e q (probabilità di rimanere in TX), e in base a questo consumi la batteria (settata secondo il valore misurato e calcolato precedentemente, 1418 mAh) di un sensore immaginario, al tempo stesso incrementandone il contatore del tempo di vita (*lifetime*). Questo procedimento viene fatto dieci volte per ogni combinazione delle due probabilità, e il dato riportato è poi la media aritmetica di queste. Nel metodo principale del codice, due cicli *for* reiterano questo pattern per tutte le probailità p da 0 a 1 (con passi di 0.01) mentre, per chiarezza del grafico finale, la q viene fatta variare a passi di 0.25. Infine, viene generato un file di testo con tutti i risultati, perfettamente formattato per poter essere immediatamente graficato tramite MATLAB.

Da notare, nella classe che genera il sensore fittizio, come solo nel metodo chiamato per il passaggio da *sleep* a *TX* siano presenti le variazioni dovute alle transizioni *sleepToActive* e *activeToSleep*: questo è valido in quanto se pensiamo a cosa succede quando il sensore rimane a trasmettere, è logico aspettarsi che non si sia spento dalla trasmissione precedente e non debba accendersi per quella successiva, mentre invece si spegnerà all'ultima di una serie consecutiva. Allora, contandole solo nel metodo citato sopra, e non contandole mai nel metodo chiamato nella situazione di trasmissione continua, commettiamo un errore praticamente nullo (la sola eccezione è dovuta al fatto che le batterie si scarichino nello stato *TX*, nel qual caso il valore della vita totale potrebbe essere più alto del dovuto, ma di un valore infinitesimale sul totale).

Quello descritto è il comportamento del programma di simulazione come riportato in appendice: tuttavia, è facile variare le condizioni della simulazione a seconda delle necessità, modificando opportunamente i valori all'interno del sorgente.

4.2 Risultati

Graficando i dati ottenuti dalle simulazioni, si ottengono le figure seguenti

La figura 4.1 ci dà una visione d'insieme dell'andamento della vita delle batterie al variare della probabilità di passare a trasmettere da una situazione di *sleep*: possiamo subito notare come l'andamento sia esponenziale decrescente all'aumentare di p, come è lecito attendersi. Infatti con $p = 1$ si trasmette una volta ogni 100 secondi, con $p = 2$ una ogni 50, con $p = 3$ una ogni 33, ecc. e i consumi aumentano di conseguenza, dato che è lo stato di trasmissione ad essere determinante in questo senso (avendo un consumo più alto di tre ordini di grandezza).

Inoltre, osserviamo come la vita delle batterie, in condizioni di attività abbastanza basse (con probabilità di transizione sleep/TX minore del 2%, ovvero circa una trasmissione ogni $50s$), sia nell'ordine degli anni: considerando anche che l'at-

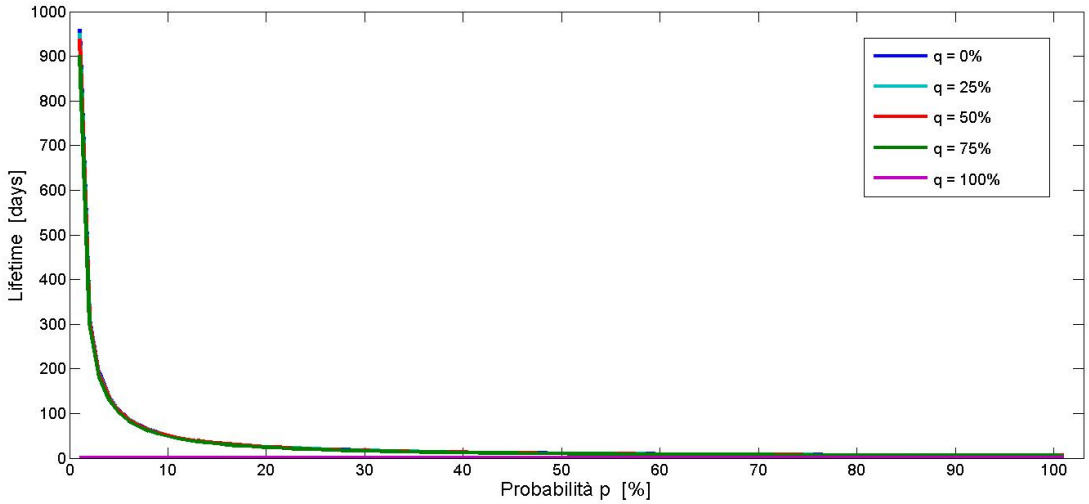


Figura 4.1: Andamento della vita del sensore al variare della probabilità p di passare da Sleep a TX, per differenti probabilità q di rimanere in TX

tività di questo sensore all'interno di una rete reale è stimata essere più bassa di tale valore, questo risultato è una conferma di come il protocollo X-MAC sia ottimizzato in termini dei consumi.

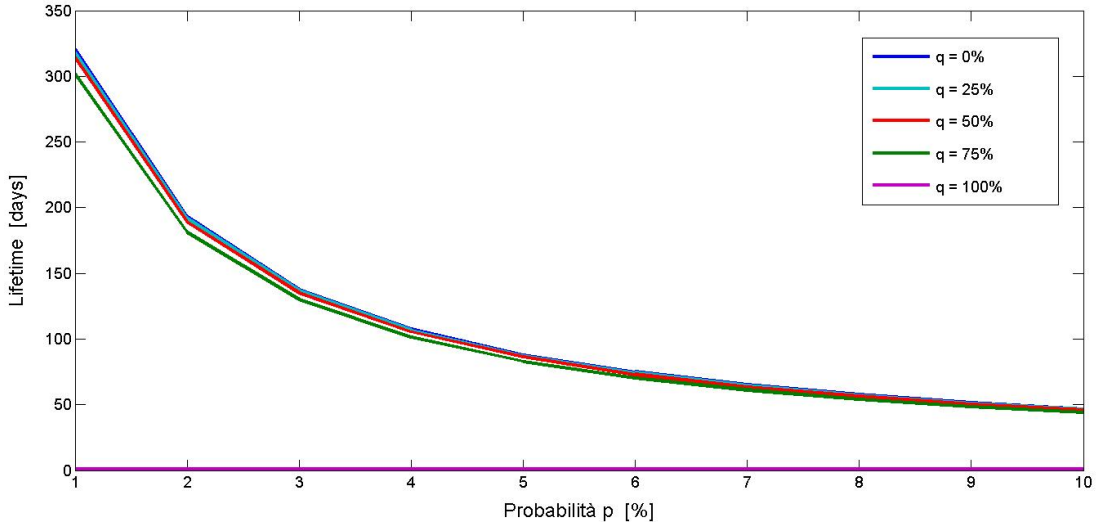


Figura 4.2: Andamento della vita del sensore al variare della probabilità p tra 1% e 10%, per differenti probabilità q di rimanere in TX

La figura 4.2 mostra più nel dettaglio la parte di grafico con p compresa tra lo 1% e il 10%: questo permette di osservare come i consumi siano quasi invarianti rispetto alla probabilità di rimanere in trasmissione, eccezion fatta naturalmente

per il caso limite di q pari a 1, dove il sensore (non uscendo mai dallo stato *TX* una voltra entratovi) esaurisce subito le batterie. Per verificare questa osservazione, facciamo una simulazione in cui, fissata la probabilità di passare da *sleep* a *TX*, variamo solamente la probabilità q.

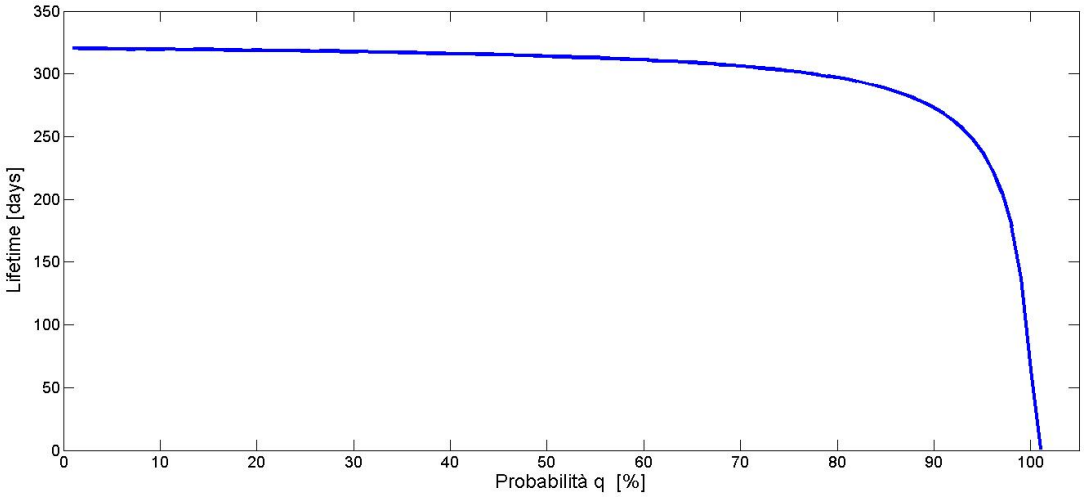


Figura 4.3: Andamento della vita del sensore in funzione della probabilità q di rimanere in TX, per una p fissata

La curva della figura 4.3 conferma la nostra osservazione, mostrando come la vita della batteria sia praticamente costante per valori di q minori del 90%, soglia oltre cui decade con pendenza sempre maggiore. Questo conferma il fatto che il punto dolente relativamente ai consumi è la fase di collegamento tra due sensori: infatti, l'andamento in figura si spiega considerando che solo la transizione da *Sleep* a *TX* comporta l'invio dello *strobed preamble*, mentre il rimanere in trasmissione no, con gli evidenti consumi minori che il secondo comporta. Questa è quindi un'ulteriore giustificazione al fatto che gli sforzi nello studio di nuovi protocolli si siano concentrati, e si concentrino, sull'ottimizzazione della fase di collegamento tra *sender* e *receiver*.

Capitolo 5

Conclusioni

In questo lavoro abbiamo presentato uno studio riguardante i consumi energetici per sensori di reti wireless. È stato descritto il protocollo X-MAC e il suo funzionamento basilare, e un sensore prototipo reale su cui verrà implementato. È stata calcolata la carica precisa di una coppia di batterie stilo ricaricabili, utilizzabili sul sensore. Si è quindi ipotizzato un modello a macchina a stati del funzionamento del sensore impostato solo come *sender*, e se ne sono calcolati i consumi di ogni stato partendo dai *datasheet* dei componenti del sensore, e dal funzionamento del protocollo MAC. Infine è stato scritto un semplice programma in Java per procedere con delle simulazioni basate su pattern di funzionamento casuali, per analizzare la durata delle batterie rispetto a differenti combinazioni di probabilità di transizione della macchina.

I risultati ottenuti portano un ulteriore conferma al fatto che il protocollo X-MAC sia energeticamente efficiente, e mostrano come all'aumentare della probabilità di passare da *sleep* a trasmissione, la durata della batteria cali con un andamento esponenziale. Inoltre, le simulazioni hanno evidenziato come la probabilità di rimanere in trasmissione sia praticamente ininfluente alla durata delle batterie, almeno fino a probabilità del 90%: questo porta ad osservare (e confermare) che la fase critica per lo scambio di dati tra due sensori implementanti un protocollo asincrono, sia quella atta a collegare il *sender* con il rispettivo *receiver*.

Infine, questo studio può essere ulteriormente sviluppato considerando un modello in cui il sensore in esame abbia anche una routine per l'ascolto e la ricezione

di dati da altri nodi, avvicinanando ancora di più la simulazione ad un caso reale, oppure può essere completato studiando il nodo duale, ossia quello ricevente.

Appendice A

Codice sorgente

Di seguito viene riportato il codice sorgente in Java usato per l'elaborazione delle simulazioni.

```
import java.util.*;
import java.io.*;
import java.math.*;

public class PatternGenerator
{     public static void main(String[] args) throws Exception
      {     Consumption[] sensor = new Consumption[10];
            double[] battery = new double[10];
            double[] lifetime = new double[10];
            PrintWriter writer = new PrintWriter("DatiSimulazione.txt");
            int alea=0;
            int state=0;                        // 0=sleep, 1=TX
            for(int p = 0; p<=100; p++)
            {     for(int q = 0; q<=100; q+=25)
                  {     for(int i = 0; i<10; i++)
                        {     state=0;
                              sensor[i]=new Consumption();
                              battery[i]=sensor[i].getBattery();
                              lifetime[i]=sensor[i].getLifetime();
                              while(battery[i]>0)
                              {     alea = (int) Math.round(Math.random()*100);
                                    if(state==0)
                                    {     if(alea<=p)             //sleep-->tx
                                          {     state=1;
                                                sensor[i].tx();
                                          }
                                          else                     //sleep-->sleep
                                                sensor[i].idle();
                                    }
                                    else
                                    {     if(alea<=q)             //tx-->tx
                                                sensor[i].stillTx();
                                          else                     //tx-->sleep
                                          {     state=0;
                                                sensor[i].idle();
                                          }
                                    }
                                    battery[i] = sensor[i].getBattery();
                              }
                              lifetime[i]=sensor[i].getLifetime();
                        }
                        double meanLifetime = 0;
                        double meanBattery = 0;
                        for(int l = 0; l<10; l++)
                        {     meanLifetime += lifetime[l];
                              meanBattery += battery[l];
                        }
```

```
                        meanLifetime = meanLifetime / 10;
                        meanBattery = meanBattery / 10;
                        writer.print(meanLifetime/(1440*60)+"  ");
                    }
                    writer.println();
                }
                writer.close();
        }
}

class Consumption
{       public void idle()      //idle di un secondo
        {       battery = battery-(0.034102*(1/3600));
                lifetime = lifetime + 1;
        }

        public void tx()              //activeToSleep + sleepToActive + invio
            pacchetto + ricezione ACK + invio preambolo + sensing a vuoto per l'ACK
        {       battery = battery-(4.41562500138889E-7+7.59736116666667E-6+((14+46.2)
            *(4.192e-3/3600))+(12.3+46.2)*(0.352e-3/3600)+(391*(((14+46.2)*(0.480e
            -3/3600))+((12.3+46.2)*(0.032e-3/3600)))));
                //116Byte di pacchetto, 32us di wait per l'ACK
                lifetime = lifetime + (52.25E-6 + 1005E-6 + 4192E-6 + 352E-6 +
                    391*(480E-6 + 32E-6));
        }

        public void stillTx()
        {       battery = battery - ((14+46.2)*(4.192e-3/3600));  //solo tx del
            pacchetto
                lifetime = lifetime + 4192E-6;
        }

        public double getBattery()
        {       return battery;
        }

        public double getLifetime()
        {       return lifetime;
        }

        protected double battery=1418;      //mAh
        protected double lifetime=0;        //secondi
}
```

Bibliografia

[1] *Catalogo pile Beghelli*. 2012. URL: http://www.beghelli.it/_documenti/pubblico/prodotti/Pile.pdf.

[2] *IEEE 802.15.4 Standard*. 2012. URL: http://www.ieee802.org/15/pub/TG4.html.

[3] *Sito web di Mathworks, software house produttrice di MATLAB*. 2012. URL: www.mathworks.org.

[4] B. Bougard et al. "Energy efficiency of the IEEE 802.15.4 standard in dense wireless microsensor networks: modeling and improvement perspectives". In: *Design, Automation and Test in Europe, 2005. Proceedings* (2005), pp. 196–201.

[5] Michael Buettner et al. "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks". In: *4th ACM Conference on Embedded Networked Sensor Systems (SenSys '06)* (2006), pp. 307–320.

[6] P. Casari et al. "The Wireless Sensor Networks for City-Wide Ambient Intelligence (WISE-WAI) Project". In: *Sensors* 9 (6 2009).

[7] Sinem Coleri, Mustafa Ergen e T. John Koo. "Lifetime analysis of a sensor network with hybrid automata modelling". In: *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications (WSNA '02)* (2002), pp. 98–104.

[8] Tijs van Dam e Koen Langendoen. "An adaptive energy-efficient MAC protocol for wireless sensor networks". In: *1st ACM Conference on Embedded Networked Sensor Systems (SenSys '03)* (2003), pp. 171–180.

[9] Waltenegus Dargie e Christian Poellabauer. *Fundamentals of Wireless Sensor Networks: Theory and Practice*. John Wiley & Sons, 2010.

[10] E.J. Duarte-Melo e Liu Mingyan. "Analysis of energy consumption and lifetime of heterogeneous wireless sensor networks". In: *Global Telecommunications Conference, 2002 (GLOBECOM '02)* (2002), pp. 21–25.

[11] Adam Dunkels et al. "Software-based On-line Energy Estimation for Sensor Nodes". In: *4th workshop on Embedded networked sensors (EmNets '07)* (2007), pp. 28–32.

[12] A. El-Hoiydi e J. Decotignie. "Low power downlink mac protocols for infrastructure wireless sensor networks". In: *ACM Mobile Networks and Applications* 10 (5 2005), pp. 675–690.

[13] L. Feeney e M. Nilsson. "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment". In: *IEEE INFOCOM* 3 (2001), pp. 1548–1557.

[14] J. E. Hopcroft, R. Motwani e J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison Wesley, 2006.

[15] O. Landsiedel, K. Wehrle e S. Gotz. "Accurate prediction of power consumption in sensor networks". In: *Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors (EmNets '05)* (2005), pp. 37–44.

[16] M.J. Miller e N. H. Vaidya. "A MAC protocol to reduce sensor network energy consumption using a wakeup radio". In: *Mobile Computing, IEEE Transactions on* 4 (2005), pp. 228–242.

[17] Joseph Polastre, Jason Hill e David Cueller. "Versatile low power media access for wireless sensor networks". In: *2nd ACM Conference on Embedded Networked Sensor Systems (SenSys '04)* (2004), pp. 95–107.

[18] C. Schurgers e M.B. Srivastava. "Energy efficient routing in wireless sensor networks". In: *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE* (2001), pp. 357–361.

[19] E. Shih, P. Bahl e M. Sinclair. "Wake on wireless: An event driven energy saving strategy for battery operated devices". In: *MobiCom* (2002), pp. 35–64.

[20] Victor Shnayder et al. "Simulating the power consumption of large-scale sensor network applications". In: *2nd ACM Conference on Embedded Networked Sensor Systems (SenSys '04)* (2004), pp. 188–200.

[21] Pablo Suarez et al. "Increasing ZigBee network lifetime with X-MAC". In: *Proceedings of the workshop on Real-world wireless sensor networks (REALWSN '08)* (2008), pp. 26–30.

[22] Wu Yanwei et al. "Energy-Efficient Wake-Up Scheduling for Data Collection and Aggregation". In: *Parallel and Distributed Systems, IEEE Transactions on* (2010), pp. 275–287.

[23] W. Ye, J. Heidemann e D. Estrin. "An energy efficient MAC protocol for wireless sensor networks". In: *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)* (2002), pp. 1567–1576.

Ringraziamenti

Desidero innanzitutto ringraziare il prof. Leonardo Badia e il dott. Riccardo Manfrin per la pazienza, il tempo e le risorse dedicate alla mia tesi. Voglio poi ringraziare di cuore la mia mamma, mia sorella e tutta la mia famiglia, per non avermi mai fatto mancare nulla e avermi dato tutto il supporto possibile per il raggiungimento dei miei obiettivi. Ringrazio Samanta per avermi accompagnato e sopportato fino ad ora. Ringrazio tutti i miei amici per avermi fatto crescere come persona, nonostante non sia stato un lavoro facile. Infine voglio ringraziare Francesco, per avermi insegnato cosa significhi *inseguire un sogno*.

Padova, 24 luglio 2012

Stefano Pretto