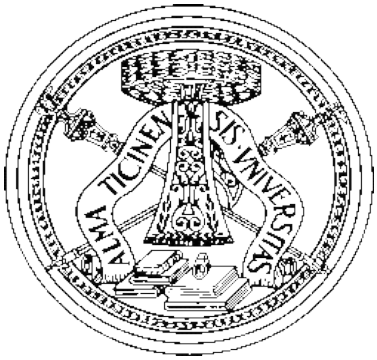


Università degli Studi di Pavia
Facoltà di Ingegneria

Corso di Laurea Specialistica in Ingegneria Informatica

Sistema di monitoraggio ambientale tramite WSN

Relatore:
Prof. Paolo Ettore Gamba

Correlatore:
Ing. Emanuele Goldoni

Elaborato di laurea
di Filippo De Stefani

Anno accademico 2007/2008

INDICE

APPENDICI

CAPITOLO 1

INTRODUZIONE

Il rapporto tra l'uomo e la natura spesso risulta problematico. Fenomeni come grandinate, uragani, inondazioni, siccità e terremoti possono causare la perdita di vite umane e ingenti danni materiali. Altre volte invece è l'uomo stesso ad innescare meccanismi disastrosi, come gli incendi boschivi, l'inquinamento e il riscaldamento globale.
Purtroppo, molti di questi fenomeni naturali sono inevitabili, e l'unica cosa che può fare l'uomo è prendere delle precauzioni prima del loro verificarsi. Ovviamente non è possibile prevedere il futuro, ma è possibile creare dei modelli che aiutino a capire in quali condizioni certi eventi si verificano. Per la formulazioni di tali modelli, serve un'intensa fase di studio, basata sull'osservazione delle variabili dell'ambiente. Osservazioni estese sia nello spazio che nel tempo sono fondamentali per capire meglio certi fenomeni, come quello del riscaldamento globale. Tuttavia, monitorare vaste aree e per molto tempo non è possibile, oppure è difficile e costoso. Infatti, solitamente, per misurare dati come ad esempio temperatura, umidità e velocità del vento si usano delle stazioni meteorologiche. Ne esistono di vari tipi, dimensione, costo e accuratezza, ma tutte hanno in comune certe caratteristiche: il costo di ognuna è piuttosto alto, l'ingombro di tutto l'apparecchio può non essere trascurabile, il consumo energetico non è molto basso e quindi devono essere collegate alla rete elettrica o a dei pannelli solari per funzionare a lungo e, infine, non hanno la capacità di formare una rete con le altre centraline. In questo modo, se sono state posizionate in un'area remota, ognuna deve essere dotata di collegamenti radio a lunga distanza (ad esempio tramite GPRS), che sono costosi e richiedono ulteriori risorse energetiche, oppure sono collegate con dei cavi, con tutti i limiti che ne derivano.
Se si dovesse monitorare un ghiacciaio con queste stazioni, il *deployment* risulterebbe molto costoso e la posizione delle singole stazioni verrebbe determinata da vincoli come, ad esempio, ingombro, copertura del segnale di connessione e presenza di irradiazione solare per i pannelli fotovoltaici.
Tuttavia, grazie all'evoluzione tecnologica, oggi stanno nascendo nuove sistemi atti a questi scopi.
Queste nuove soluzioni sono state rese possibili dal progresso tecnologico, che ha consentito la creazione di hardware sempre più piccolo, performante ed efficiente in termini di consumo energetico, e sono conosciute come *Wireless Sensor Network (WSN)*.
Una WSN è un insieme di nodi dotati di CPU, memoria, radiotrasmettitore e sensori che effettuano delle misurazioni dell'ambiente (*sensing*) e le trasmettono ad un punto di raccolta, il quale poi le inoltra ad un sistema di elaborazione remoto. La caratteristica di questi nodi è che sono piccoli, consumano pochissima energia, sono collegati tra di loro e costano poco. La rete che formano è auto-configurante ed autonoma, non richiede nessun intervento dell'uomo, può arrivare a coprire un area molto estesa e può rimanere attiva per molto tempo.
A differenza del precedente metodo di monitoraggio con le stazioni meteo, una soluzione tramite WSN consente di risparmiare parecchio sui costi, monitorare aree molto più estese, con condizioni ambientali estreme e in punti molto ristretti e difficili da raggiungere. Inoltre, dato il basso costo di ogni nodo, sono utilizzabili in situazioni di emergenza, dove è possibile che vengano distrutti o persi in qualche modo, come ad esempio in aree a pericolo di incendio o di frane.
Un'altra importante applicazione delle WSN, oltre a quella per scopi scientifici, è la cosiddetta *agricoltura di precisione.* In tale ambito, la rete di sensori è utilizzata per monitorare i parametri ambientali di un campo agricolo, in modo che l'agricoltore sappia, per esempio, quando doverlo irrigare o quando adoperare certi pesticidi.
Le possibilità di utilizzo di una Wireless Sensor Network non si limitano solo a ciò: possono essere adoperate per fini militari, di sorveglianza, di *home automation*, di monitoraggio di parametri biologici e tanti altri.
Il progetto realizzato in questa tesi è un sistema di monitoraggio ambientale adatto ad uno scopo di

agricoltura di precisione e studio del territorio.
L'obiettivo è stato quello di realizzare un sistema completo, economico e pronto all'uso, con il quale l'utente, tramite il proprio browser, possa visualizzare e gestire i dati rilevati nell'area monitorata.
Per rendere possibile ciò è stata progettata una Wireless Sensor Network, ovvero sono stati creati i protocolli di rete e l'applicazione necessari al suo funzionamento, dopodiché è stato sviluppato il sistema di memorizzazione, consultazione e analisi dati, tramite l'implementazione di un database, di un web server e di un applicazione web.
Nonostante esistano già soluzioni simili sviluppate da altre università o aziende, il sistema realizzato si distingue per il suo basso costo e l'hardware alternativo utilizzato per i nodi, basato su *Arduino* e *XBee*, più semplice ed economico di quello solitamente adoperato.
Una volta terminata la fase progettuale, sono stati svolti dei test di tutto il sistema per determinare la validità delle architetture sviluppate, in particolare le performance della rete di sensori.
Le fasi seguite nella realizzazione del progetto sono state riprese nell'ordine in cui è stato strutturato questo elaborato di tesi.
Nel *Capitolo 2* vengono introdotte le WSN attraverso una descrizione delle loro origini, delle loro applicazioni, della loro struttura, della loro architettura hardware e software, delle sfide progettuali.
Il *Capitolo 3* illustra le soluzioni di reti di sensori per il monitoraggio ambientale già realizzate da altre università ed aziende.
Nel *Capitolo 4* viene presentato l'obiettivo del progetto realizzato in questa tesi e viene fornita la specifica dei requisiti del sistema. Dopodiché viene accennata l'architettura implementata, insieme ad una descrizione dei componenti hardware adoperati. Infine vengono illustrate le due fasi seguite nello sviluppo del sistema: la progettazione della WSN e la progettazione del sistema di memorizzazione, consultazione ed analisi dei dati.
Il *Capitolo 5* è dedicato alla prima fase, ovvero la progettazione della Wireless Sensor Network. Questo capitolo, insieme al sesto, costituisce il nucleo vero e proprio della tesi. In tale capitolo, vengono prima presentate le varie soluzioni, già esistenti, che compongono l'architettura di rete di una WSN; vengono poi spiegate quelle create per questo progetto, confrontandole con quelle precedenti. Con il termine “soluzioni”, in questo caso, si intendono tutti i vari protocolli di trasmissione dei bit a livello fisico, di accesso al canale, di rete, di *duty-cycling*, di risparmio energetico e di *topology control*.
Nel *Capitolo 6* viene descritta la seconda fase, ovvero la progettazione del sistema di memorizzazione, consultazione ed analisi dati. Qui viene descritta la sua architettura, insieme ai vari componenti software che la compongono. Dopodiché vengono presentate le funzionalità e l'interfaccia utente dell'applicazione web sviluppata.
Nel *Capitolo 7* viene proposta un analisi di propagazione del segnale radio dei nodi utilizzati, che riporta i risultati di test effettuati in ambienti chiusi e aperti. Poi vengono illustrati i risultati dei test del deployment del sistema completo, corredati da un'analisi delle prestazioni della rete.
Infine, nel *Capitolo 8*, vengono fatte delle considerazioni finali sul sistema realizzato e vengono proposti eventuali sviluppi futuri del progetto.

CAPITOLO 2

INTRODUZIONE ALLE WIRELESS SENSOR NETWORK

2.1 DEFINIZIONE DI UNA WSN

Una *sensor network* è un'infrastruttura composta da elementi in grado di compiere misurazioni, elaborarle e comunicarle ad un punto centrale, dove i dati vengono gestiti.
Una *WSN (Wireless Sensor Network)* può quindi essere definita come un insieme di nodi wireless interconnessi (anche detti *mote, sensor node, wireless node o smart dust),* aventi poca RAM e una CPU a basse prestazioni.
La struttura di una Wireless Sensor Network prevede solitamente diversi nodi wireless sparsi in un'area, che inviano periodicamente dati rilevati tramite sensori ad un punto di raccolta, detto *sink* o *base station* o *gateway,* il quale gestisce la rete, raccoglie i dati dei nodi e li inoltra ad un altro sistema remoto per ulteriori elaborazioni (vedasi Figura 2.1).

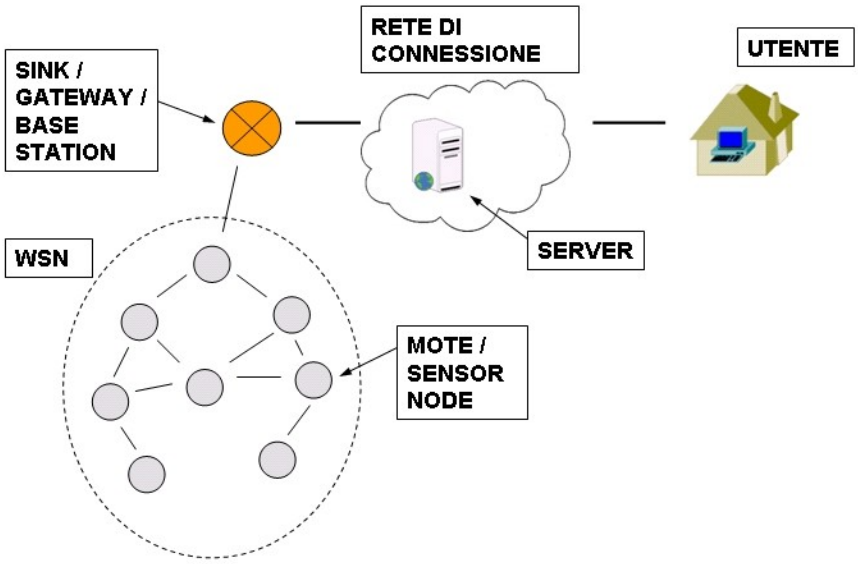


Figura 2.1: Struttura tipica di una WSN

I possibili utilizzi di una Sensor Network sono la raccolta dati, la sorveglianza, il monitoraggio, la telemetria medica ed altri ancora. Oltre alla semplice misurazione, è possibile anche azionare degli attuatori o dei sistemi di controllo.
I componenti basilari di una rete di questo tipo sono:

- un insieme di sensori distribuiti;
- una rete di interconnessione (solitamente wireless);
- un punto di raccolta dati;
- un insieme di risorse computazionali nel punto di arrivo dei dati (oppure oltre di esso), al fine di effettuare *data mining*, correlazioni dati, monitoraggio dello stato, etc...

Parte dell'elaborazione dei dati può essere già svolta all'interno della rete, in modo da alleggerire il carico di lavoro dell'unità centrale (che altrimenti dovrebbe effettuare calcoli su molti dati).
Le architetture di elaborazione e di comunicazione dei dati sono specifiche del tipo di applicazione che si sta realizzando; quindi, un sistema di monitoraggio ambientale sarà diverso da uno di sorveglianza militare.

2.2 BACKGROUND DELLE WSN

I sensori di una rete WSN possono avere varie funzioni, scopi e possibilità e, grazie alla continua evoluzione tecnologica, si scoprono sempre più ambiti di utilizzo. Le reti di sensori esistono già da parecchio tempo: le centraline meteorologiche, i radar di controllo del traffico areo e la rete elettrica nazionale rappresentano alcuni esempi. Tuttavia esse richiedono hardware specializzato e specifici protocolli, che rendono queste reti particolarmente costose, l'esatto contrario delle WSN.
Lo sviluppo delle Wireless Sensor Network è iniziato molto recentemente ed è un ambito multidisciplinare poiché richiede conoscenze di radio e *networking*, *signal processing,* gestione di database, intelligenza artificiale, ottimizzazione delle risorse, algoritmi di risparmio energetico, architetture di sistemi per l'utente e piattaforme tecnologiche (hardware e software). Inoltre, proprio il continuo sviluppo della tecnologia, spinge sempre più lontano eventuali limiti progettuali. Ad esempio, oggi esistono molti tipi di sensori, in grado di valutare diverse grandezze, di dimensioni veramente ridotte, dall'ottima affidabilità e dal basso consumo energetico.
I sensori attualmente sul mercato possono rilevare elettricità, campo magnetico, onde radio, raggi infrarossi, coordinate geografiche, suoni, pressione, parametri ambientali (luminosità, temperatura, umidità, vento,...), parametri biologici, parametri biochimici e altri ancora. Un obiettivo commerciale che è stato prefissato è quello di realizzare sistemi con sensori basati sui *micro sistemi elettro-meccanici (MEMS)* della dimensione totale di 1 mm^3 [75].
I dati rilevati dai sensori vengono spediti all'interno della rete tramite dei collegamenti wireless a bassa potenza al punto di raccolta detto gateway*,* che è spesso connesso ad Internet e che a sua volta può inoltrare i dati ad un punto di analisi. Il meccanismo di accesso al canale wireless è solitamente di tipo *contention-oriented* ad accesso casuale, come definito nello standard IEEE 802.
Le WSN sono caratterizzate da alcuni aspetti chiave come, per esempio, i vincoli di potenza elaborativa, la durata limitata della batteria, un basso *duty-cycle* e connessioni molti-a-uno, che creano delle sfide progettuali nell'ambito di trasporto dei dati, gestione della rete, disponibilità, confidenzialità, integrità e *in-network processing*. Per di più, lo stack del protocollo realizzato deve essere quanto più leggero possibile, per poter soddisfare i vincoli imposti dai limiti hardware dei nodi.
Il superamento di queste difficoltà, ha comportato un primo passo verso la creazione di uno standard per le WSN: già nei primi anni del 2000 furono definiti degli standard, che tuttavia erano proprietari. In seguito si è cercato di crearne uno aperto.
Dapprima sono stati esaminati quelli già disponibili: l'*IEEE 802.11* non va bene perché richiede troppe risorse hardware e offre una banda ben oltre le necessità di un nodo della WSN (che trasmette pochissimi byte alla volta); i sistemi ad infrarossi non sono adatti poiché richiedono un allineamento visivo tra i nodi; l'*IEEE 802.15.1 (Bluetooth)* è troppo complesso e costoso. Così si è arrivati a definire un nuovo standard, l'*IEEE 802.15.4* e il *ZigBee* (ZigBee definisce solo i layer software sopra l'802.15.4 e supporta diverse applicazioni).
L'IEEE 802.15.4 opera nella banda radio ISM a 2.4 Ghz, permette data rate fino a 250 kbps e un range tipico tra i 10 e i 75 metri.
È stato pronosticato che, entro il 2010, il volume di vendita di apparecchi basati sull'IEEE 802.15.4/ZigBee sarà tre volte il volume di vendita di quelli Wi-Fi (i nodi ZigBee compliant dovrebbero passare da 1 milione nel 2005 a 100 milioni nel 2008) [75].
Ben prima della definizione di questi standard, furono create delle reti che potrebbero sembrare molto simili alle WSN, le *MANET* , ma che in realtà presentano sostanziali differenze.
Le MANET, ovvero *Mobile Ad Hoc Network*, sono reti costituite ad un preciso scopo e per soddisfare una necessità comunicativa immediata. Solitamente le MANET sono reti wireless multi-hop e i nodi sono mobili. Una tipica MANET è quella che viene creata tra le varie squadre di soccorso durante una situazione di *disaster-recovery*. Le sfide principali di una MANET sono quelle di riorganizzare la rete quando i nodi si muovono e gestire i problemi derivanti dalla comunicazione wireless (perdita di connessione, distanza massima di trasmissione,...); in parte queste sono anche le sfide di una WSN, ma i punti dove le due reti si differenziano veramente sono i seguenti:

1. **Equipaggiamento e applicazioni:** nelle MANET i terminali sono laptop, alimentati da

batterie grosse e vi è un utente umano; le applicazioni tipiche sono le chiamate vocali o l'accesso a computer remoti (Web Server,...).

2. **Specificità d'uso:** le MANET non hanno la stessa versatilità d'uso delle WSN, e sono progettate solo a specifici scopi ed usi, nonché numero di nodi.
3. **Interazione con l'ambiente:** le WSN interagiscono con l'ambiente. Ciò significa, ad esempio, che per un periodo in cui non vi sono cambiamenti ambientali particolari, il traffico di rete è molto basso, mentre durante una situazione di allarme, il traffico può essere molto elevato. Le MANET invece supportano un tipo di traffico classico, consumer, caratterizzato da voce e dati.
4. **Scalabilità**: le WSN possono gestire migliaia di nodi, le MANET molti meno.
5. **Energia:** le reti di sensori hanno vincoli di consumo energetico molto più restrittivi delle MANET, per le quali la sostituzione di una batteria non presenta particolari problemi.
6. **Auto-configurazione**: entrambi i tipi di rete richiedono questa funzionalità, anche se è più difficile implementarla sulle WSN, a causa dei molti vincoli già presenti. In ogni caso, l'auto-configurazione è l'aspetto in cui le due reti più si assomigliano.
7. **Affidabilità e QoS**: in una MANET ogni nodo deve garantire una certa affidabilità; in una rete di sensori, la perdita di un nodo non deve destare preoccupazione. Per quanto riguarda il Quality of Service (QoS), una MANET può implementare delle politiche di QoS (ad esempio per il traffico voce), mentre una WSN possiede una tipologia di QoS totalmente diversa, dettata da altre regole (risparmio energetico *in primis*).
8. **Data-Centric:** il termine *data-centric* è totalmente estraneo ad una MANET, mentre è un approccio fondamentale per la progettazione di una WSN.
9. **Semplicità e risorse limitate:** il software di un nodo di una WSN deve essere il più leggero possibile e occupare poca RAM; in una MANET i tipi di protocolli adoperati non soddisfano nessuna di queste condizioni.
10. **Mobilità**: in entrambi le reti ci possono essere nodi o gruppi di nodi in movimento, ma nelle WSN ciò può implicare complicazioni maggiori.

Quindi, il fatto che le Sensor Network supportino vari tipi di applicazione, interagiscano con l'ambiente e debbano ponderare vari trade-off, permette di considerarle un sistema diverso dalle MANET.
Dunque, anche a causa di queste differenze tra le WSN e le MANET, è stato necessario definire un nuovo standard, ovvero il già citato IEEE 802.15.4.

2.2.1 Origini delle Sensor Network

La storia delle reti di sensori si divide in quattro fasi [75]:

1. **Sensor Network dell'epoca della guerra fredda**: durante la guerra fredda furono sviluppate negli Stati Uniti delle reti acustiche di sorveglianza sottomarina, che vengono usate ancora oggi per rilevare attività sismiche sottomarine. Inoltre, nel Nord America, furono create delle reti di radar per la difesa aerea, i cui sensori erano costituiti da degli aerei *Airborn Warning and Control System (AWACS) .*
2. **Iniziative del DARPA**: Il maggior stimolo allo sviluppo delle reti di sensori fu nei primi anni 80, quando il *DARPA (Defence Advanced Research Projects Agency)* sponsorizzò la creazione di *Distributed Sensor Network (DSN)* per determinare se protocolli TCP-IP, insieme alla rete *ARPA*, potessero essere usati come reti di sensori. Le DSN prevedevano diversi nodi autonomi equamente spaziati, in grado di collaborare tra loro, il cui obiettivo era quello di raccogliere i dati in nodi che potessero utilizzare al meglio quelle informazioni. Le applicazioni delle DNS erano il calcolo distribuito, il signal processing e il *tracking*. I componenti di queste reti erano sensori acustici, protocolli di comunicazione di alto livello, algoritmi di calcolo e software distribuito.
3. **Applicazioni militari degli anni 80 e 90:** in questo periodo furono sviluppati i primi prodotti

commerciali derivanti dalla tecnologia delle DSN che, proprio grazie alla commercializzazione, erano caratterizzati da un costo minore e dall'implementazione dei primi standard. Tali reti di sensori erano adoperati in situazioni di *network-centric warfare,* ovvero in ambienti in cui i sistemi d'arma devono cooperare tra di loro scambiandosi informazioni sull'obiettivo. I sensori infatti erano in grado di tracciare obiettivi multipli, anche molto distanti e con ottimi tempi di risposta. Esempi di questa tecnologia erano le reti di radar per rilevare obiettivi aerei, reti di sensori acustici negli oceani per rilevare sottomarini e reti di sensori schierate sul campo di battaglia.

4. **Ricerca attuale:** dopo l'evoluzione tecnologica degli anni 90 e dei primi anni del 2000, si è giunti ad una nuova generazione di sensori, che può essere definita come seconda generazione di prodotti commerciali. Il progresso ha consentito lo sviluppo di tecnologie ad alta densità come i MEMS e i recenti *NEMS (Nanoscale Electromechanical Systems)*.
   La standardizzazione è diventata un fattore determinante per la diffusione delle WSN e la nascita di sistemi Bluetooth, Wi-Fi, Wimax e ZigBee è un fattore che abilita la connettività totale. Tutto ciò, insieme a nuovi processori a basso costo e basso consumo energetico, consente oggi di adoperare le reti di sensori per molteplici applicazioni.
   La ricerca nelle soluzioni commerciali si sta muovendo verso la definizione di topologie di rete *mesh*, *peer-to-peer* e *cluster-tree*, di standard di sicurezza e di profili di applicazioni comuni.

2.3 APPLICAZIONI DELLE RETI DI SENSORI

Inizialmente le Sensor Network erano adoperate solo in contesti particolari come la rilevazione di radiazioni, tracciamento di obiettivi e sorveglianza militare, rilevazione di dati biomedici, monitoraggio di un'area e rilevazione di attività sismiche. Più recentemente, l'attenzione è stata rivolta verso reti di sensori biologici e chimici per applicazioni di sicurezza nazionale, nonché verso lo sviluppo di soluzioni commerciali alla portata di molti.
Un piccolo elenco di possibili applicazioni è il seguente:

- **Ambito militare**:
  - monitoraggio forze nemiche
  - monitoraggio forze alleate ed equipaggiamenti
  - sorveglianza di campi di battaglia
  - tracking di obiettivi
  - stima dei danni
  - rilevazione di attacchi nucleari e biochimici

- **Ambito ambientale**:
  - rilevamento di incendi boschivi
  - rilevamento di inondazioni
  - monitoraggio di microclimi
  - agricoltura di precisione

- **Ambito biomedico**:
  - monitoraggio a distanza di parametri fisiologici
  - tracking di medici e pazienti all'interno di un ospedale
  - gestione dei medicinali

- **Ambito domestico**:
  - Home Automation
  - lettura contatori (Smart Metering)

- **Ambito industriale**:
  - controllo ambientale in costruzioni industriali ed uffici

- controllo dell'inventario
- tracking di veicoli
- monitoraggio della *supply chain*

- **Ambito dei trasporti:**
  - monitoraggio del traffico
  - sensori *intra-veicolo*

Sensori chimici, fisici, d'immagine e acustici possono essere adoperati per monitorare ecosistemi.
Sul campo di battaglia, i sensori possono essere usati per identificare e tracciare oggetti, veicoli e personale nemico. Microsensori possono essere posizionati ovunque, in aria, per terra, in acqua, nel corpo umano, in veicoli, in edifici. Quest'ultimi, vengono definiti in tal caso *smart space*.
La diffusione delle WSN riguarda anche l'ambito consumer: nei prossimi anni alcuni produttori di questo ambito inizieranno ad inserire sensori nei loro prodotti, per migliorarne prestazioni e durata.
In questo modo sarà possibile azionare il riscaldamento di casa o l'illuminazione in remoto, richiedere la telemetria alla propria auto o monitorare i propri parametri di salute [102, 103], tanto per citare alcuni esempi. La meta che si vuole raggiungere è quella di consentire ad una persona il controllo totale di tutti i suoi oggetti.

2.4 SFIDE PROGETTUALI DI UNA WSN

Non è possibile creare un prototipo di WSN in grado di gestire tutte le applicazioni sopra elencate. Tuttavia, anche se ognuna di esse avrà sfide particolari, molte saranno condivise.
Tali sfide comuni si possono riassumere nelle seguenti caratteristiche:

- **Type of Service**: le WSN non sono delle reti tradizionali che si limitano a comunicare dei bit, bensì devono provvedere informazioni utili o agire ad eventi particolari. Per questo motivo servono nuovi paradigmi, nuove interfacce utente e nuovi modi di pensare ai servizi della rete.
- **Quality of Service (QoS):** i parametri QoS adottati tradizionalmente come ritardo, banda minima e *jitter*, non servono se si stanno utilizzando software che tollerano i ritardi e se i nodi scambiano pochi pacchetti alla volta. Inoltre, in alcuni casi basta ricevere qualche dato ogni tanto, mentre in altri tutti i dati devono essere ricevuti oppure devono essere ricevuti entro un certo tempo. Ciò che conta è quindi la quantità e la qualità dei dati ricevuti da un nodo di raccolta di una certa area. Ad esempio, delle metriche valide possono essere l'affidabilità nel rilevamento di certi eventi o l'approssimazione di certe misure.
- **Tolleranza ai guasti:** un nodo potrebbe esaurire la batteria, subire danni fisici o perdere il collegamento wireless con gli altri nodi. Una simile situazione non deve però danneggiare il resto della rete e un altro nodo dovrebbe possibilmente prendere il posto del nodo perso.
- **Lifetime:** nella maggior parte degli scenari, i nodi sono alimentati da batterie che non è possibile o non è conveniente sostituire. In ogni caso, l'obiettivo di una WSN è quello di rimanere attiva più a lungo possibile o, almeno, per la durata della sua missione. Quindi, il risparmio energetico assume un ruolo fondamentale in una Wireless Sensor Network. Anche nelle applicazioni in cui è previsto l'ausilio di un piccolo pannello solare per ricaricare la batteria dei nodi, il risparmio energetico rimane un fattore critico e in queste situazioni l'obiettivo è quello di mantenere la rete attiva per un tempo indeterminato.
  Tuttavia, l'implementazione di meccanismi di risparmio energetico richiede dei compromessi con la qualità del servizio: la soluzione è ovviamente quella di trovare un giusto bilanciamento tra le due caratteristiche.
  La definizione di *lifetime* non è univoca, nel senso che dipende dall'applicazione che si vuole misurare: a volte infatti si indica con lifetime il tempo entro il quale il primo nodo della rete finisce la propria energia (o comunque non funziona più); altre volte invece corrisponde al momento in cui il 50% dei nodi vengono persi; oppure indica la prima volta in cui una regione

sotto controllo non è più monitorata da alcun nodo.

- **Scalabilità:** una WSN può contenere fino a migliaia di nodi e la sua architettura deve essere in grado di supportarli tutti.
- **Densità di nodi non uniforme**: in una rete di sensori possono esserci zone molto affollate (con molti nodi) e zone con pochissimi nodi sparsi. La densità dei nodi può variare nello spazio e nel tempo (ad esempio perché i nodi finiscono la batteria) e la rete deve essere in grado di adattarsi a queste variazioni.
- **Programmabilità:** i nodi devono essere in grado di poter cambiare i propri compiti in qualunque momento, ovvero devono poter essere riprogrammabili.
- **Manutenibilità:** dato che sia la WSN sia l'ambiente in cui si trova sono in continuo mutamento, la rete deve essere in grado di adattarsi, monitorando il proprio stato di salute, aggiornando i propri parametri, decidendo tra nuovi compromessi (ad esempio diminuendo la qualità del servizio quando l'energia sta per terminare). Quindi deve essere in grado di auto-mantenersi.

Per risolvere tutte queste sfide bisogna creare meccanismi di comunicazione, architetture e protocolli adatte allo scopo che supportino le seguenti funzionalità:

- **Connessioni wireless multi-hop:** la comunicazione diretta tra due nodi non è sempre possibile, poiché potrebbero esserci ostacoli oppure perché i nodi sono molto distanti tra loro e l'utilizzo di una potenza trasmissiva elevata comporterebbe un rapido esaurimento della batteria. Quindi la soluzione è quella di adoperare dei nodi che fungano da *relay* verso altri nodi.
- **Operazioni energeticamente efficienti:** è importante che tutte le operazioni compiute tengano in considerazione il risparmio energetico e bisogna, possibilmente, evitare la formazione di *hotspot*, ovvero regioni o gruppi di nodi che esauriscono la propria energia molto più rapidamente degli altri.
- **Auto-configurazione:** la rete deve essere in grado di configurare automaticamente tutti i suoi parametri vitali. Per esempio deve gestire autonomamente l'ingresso di un nuovo nodo oppure l'aggiornamento delle tabelle di routing dopo la perdita di alcuni nodi.
- **Collaborazione e in-network processing:** in alcune applicazioni, un singolo nodo non è in grado di capire se si è verificato un evento. Per questo motivo è necessario che i nodi collaborino tra di loro e compiano *in-network processing*, ovvero eseguano alcuni calcoli sui dati, tipicamente *data-aggregation* (ad esempio il calcolo della temperatura media di una zona)
- **Data-centric:** in una rete di comunicazione tradizionale, lo scambio di dati avviene tra entità aventi ognuna un indirizzo di rete specifico, quindi si tratta di un'architettura *address-centric*.
  In una WSN, non importa tanto chi fornisce il dato, ma da quale regione proviene. D'altronde un nodo può essere ridondato da più nodi, e quindi si perde l'individualità dei vari componenti. Ciò che interessa è richiedere una certa informazione ad una certa area monitorata, e non richiedere una certa informazione ad un certo nodo. È un concetto simile alla query di un database “Visualizza tutte le aree in cui la temperatura è maggiore di X” oppure “Richiedi i dati di umidità della regione X”.
- **Località:** per risparmiare risorse hardware, il nodo deve interessarsi e memorizzare informazioni di routing solo verso i nodi vicini a lui. Così facendo, nel momento in cui la rete dovesse crescere esponenzialmente, le risorse hardware occupate rimarrebbero inalterate. Chiaramente, conciliare località e protocolli di routing efficienti è una delle sfide da affrontare.
- **Bilanciamento dei trade-off:** sia durante la fase di progettazione della Wireless Sensor Network sia durante il suo *runtime*, bisogna ponderare diversi trade-off, anche contraddittori tra loro. Alcuni di questi sono già stati accennati: lifetime e qualità del servizio, lifetime della rete e lifetime del singolo nodo, densità della rete ed efficienza del routing, solo per citarne alcuni.

Riuscire a combinare tutti questi meccanismi in maniera semplice ed efficiente è una sfida non da poco.

2.5 ARCHITETTURA HARDWARE DI UN NODO

I componenti hardware dei nodi di una WSN comprendono CPU, memoria, radio, sensori, attuatori e fonte di alimentazione, come riportato in Figura 2.2. Una breve descrizione di questi componenti e le piattaforme più diffuse vengono fornite nel paragrafo successivo.

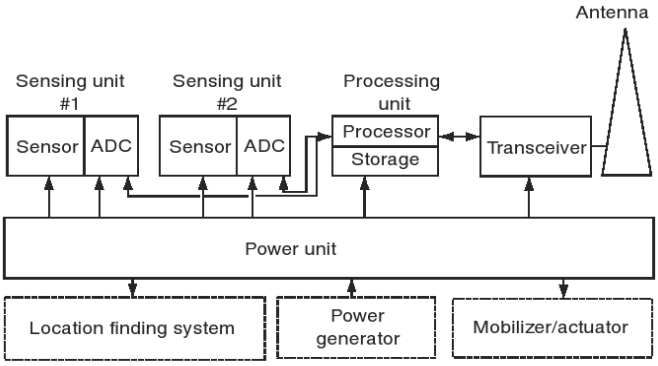


Figura 2.2: Architettura hardware di un nodo

2.5.1 Componenti hardware

- **Controller:** è la CPU del nodo e si occupa di rilevare i dati tramite i sensori, elaborarli, decidere dove e quando spedirli, ricevere dati da altri nodi, gestire gli attuatori e i sensori.
  I componenti più adatti a questo scopo sono i microcontrollori, ovvero processori per sistemi *embedded*, poiché richiedono poca energia, hanno spesso delle memoria integrata e non hanno controller di memoria, sono facilmente programmabili (anche tramite software open-source), si integrano bene con altri componenti come sensori e attuatori e hanno delle modalità di risparmio energetico.
  I microcontrollori maggiormente diffusi in ambito WSN sono il *Texas Instrument MP 430* e l'*Atmel ATmega*. Il primo è un processore *RISC (Reduced Instruction Set Computer)* a 16 bit con frequenza fino a 4 Mhz, da 2 a 10 kb di RAM on-chip, diversi convertitori analogico/digitale a 12bit e un real-time clock.
  L'ATmega è sempre un RISC ma a 8 bit, con memoria on-chip da 4 a 256 kb, frequenza fino a 20 Mhz e convertitori A/D a 10 o 12 bit.
- **Memoria:** la memoria è di tipo RAM per le variabili temporanee, mentre per il codice si usano le EEPROM o le flash.
- **Dispositivo di comunicazione:** è costituito da un *transceiver* ad onde radio che opera solitamente sui 2.4Ghz (per reti 802.15.4).
- **Sensori e attuatori:** i sensori possono essere di tre tipi: *passivi omnidirezionali*, *passivi direzionali* e *attivi*. Con il termine passivo si indica il fatto che il sensore non modifica in nessun modo l'oggetto della misurazione, mentre con omnidirezionale si intende che la grandezza misurata non varia a seconda della direzione in cui la si misura; ad esempio, sensori di temperatura ed umidità sono omnidirezionali. Sono direzionali invece i sensori di immagine, come le fotocamere, che restituiscono risultati differenti in base a come sono posizionate. I sensori attivi, invece, effettuano delle misurazioni dopo aver sollecitato l'ambiente in qualche modo, ad esempio creando delle piccole esplosioni nel terreno.
  Chiaramente, ogni sensore è caratterizzato da una certa accuratezza, affidabilità, costo e dimensione. Nelle WSN si usano quasi esclusivamente sensori di tipo passivo omnidirezionale.
  Per quanto riguarda gli attuatori, le WSN si limitano ad aprire o chiudere un interruttore o ad impostare un valore in qualche modo, perciò non sono attuatori particolari.

- **Fonte di alimentazione:** per dare energia ad un nodo si ricorre a batterie, che possono essere di qualsiasi tipo, a seconda dell'hardware e dell'applicazione da realizzare. Le reti che devono avere un lifetime molto lungo ricorrono a fonti energetiche ausiliare per ricaricare le batterie dei nodi. La soluzione più ricorrente è rappresentata dall'utilizzo di celle fotovoltaiche installate su ogni nodo, ma altre idee sono in fase di sviluppo; tra queste, le più adatte alle WSN sono i sistemi che sfruttano le vibrazioni meccaniche per produrre energie tramite piezo o altri componenti elettromagnetici ed elettrostatici, e quelle che sfruttano il flusso di aria o liquidi attraverso minuscole turbine basate su MEMS. Inoltre, se la disposizione dell'energia nell'ambiente viene tenuta in considerazione per l'allocazione dei compiti all'interno di una rete, il lifetime può aumentare fino al 200% [1].
  È importante però tenere presente che queste fonti ausiliare hanno un costo energetico supplementare a causa dei circuiti di regolazione necessari.

2.5.2 Piattaforme hardware

Sono disponibili in commercio diverse piattaforme. Quale scegliere dipende dall'applicazione che si vuole creare, dal contesto in cui verrà adoperato, dalle dimensioni massime ammissibili, dal costo sostenibile, dalla loro efficienza energetica e dalla robustezza richiesta.
I modelli più diffusi sul mercato sono questi:

- **Mica Mote:** sono stati sviluppati dall'Università della California a Berkeley, in parte con la collaborazione di Intel, verso la fine degli anni 90 [79]. Ci sono quattro modelli appartenenti a questa serie: Mica, Mica2, Mica2Dot (Figura 2.3) e MicaZ (Figura 2.4) che differiscono in base a caratteristiche tecniche e dimensioni. Sono distribuiti dalla Crossbow e sembrano essere i più utilizzati nelle WSN.
  Sono dotati tutti di microcontroller Atmel, in particolare il più piccolo della famiglia possiede un Atmel 8 bit a 4Mhz, 4 K di RAM, 128 K di flash programmabile e 512 K flash per i dati, una radio da massimo 40 kbps a 900/450/300 Mhz. Un interessante evoluzione dei Mica è lo Spec [80]. Lo Spec è un single-chip mote, ovvero integra un core Atmel di tipo RISC, 3K di memoria, una porta input e una output da 4bit, 8 bit ADC on-chip, trasmettitore radio e oscillatore a 32 Khz nello spazio di soli 2 mm x 2,5 mm. Lo Spec può rappresentare un grosso passo avanti nelle reti di sensori. Al momento non è ancora in commercio, ma non appena sarà disponibile avrà anche un costo molto basso, dato che lo strato di silicio di 5 mm^2 costa 0,30 € e gli altri componenti ancora meno, quindi la cifra totale sarà molto bassa.

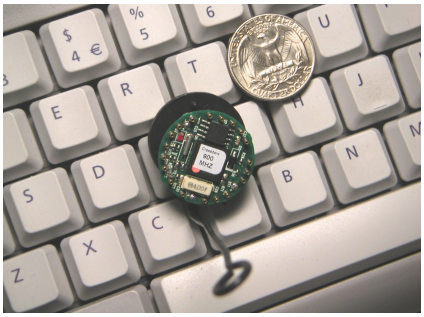


Figura 2.3: Un Mica2Dot

Figura 2.4: Un nodo spesso utilizzato nelle WSN, il MicaZ

- **EYES:** questi nodi sono stati sviluppati da Infineon nell'ambito del progetto sponsorizzato dall'Unione Europea “Energy Efficient Sensor Networks” [82]. Sono equipaggiati con microcontrollore Texas Instrument MP 430 e radio TDA 520 (Figura 2.5).

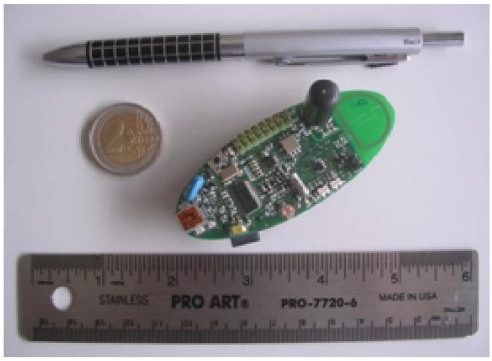


Figura 2.5: Un mote EYES

- **BTnodes:** creati dall'ETH di Zurigo [81], sono dotati di Atmel Atmega128L, 64+180 K RAM e 128 K di flash. Per la radio utilizzano una combinazione di Bluetooth e un Chipcon CC1000 tra i 433 Mhz e 915 Mhz (Figura 2.6).

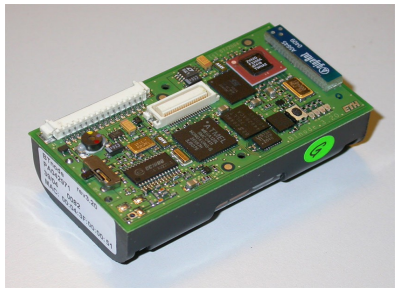


Figura 2.6: Il BTnode

- **Scatterweb:** nati dalla ricerca della Freie Universitaet Berlin [83], sono una famiglia di piattaforme che variano dal nodo standard con Texas Instruments MP 430 (Figura 2.7) fino a nodi con web server integrato e diverse possibilità di interconnessione (oltre Bluetooth o radio a bassa potenza), come ad esempio CAN .

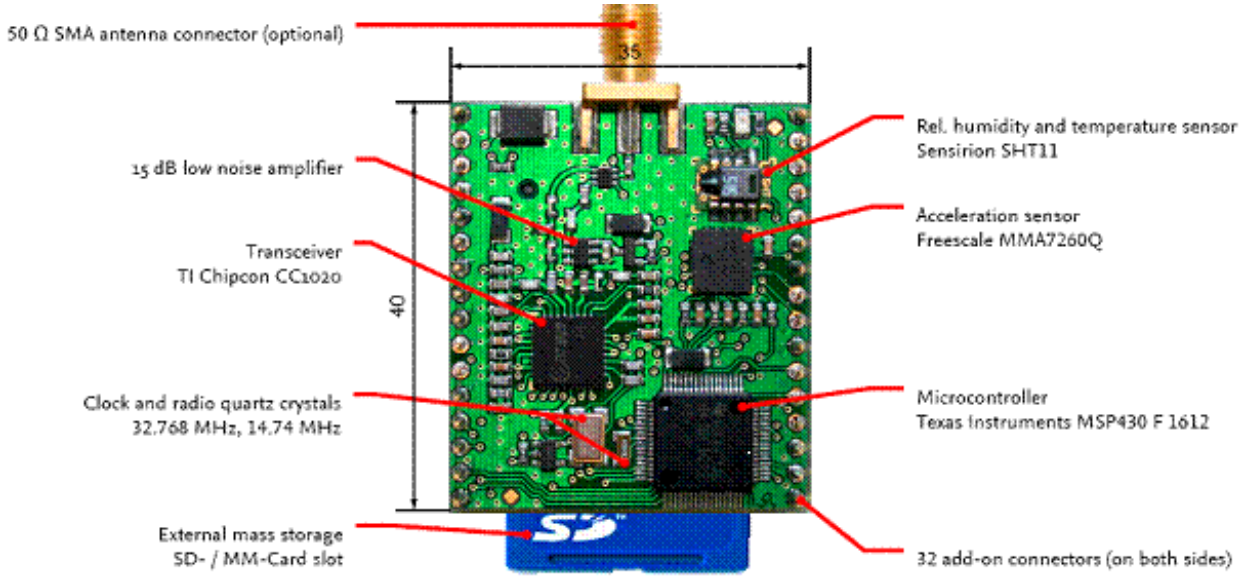


Figura 2.7: La struttura di un nodo Scatterweb

- **FireFly:** questi nodi [84] sono anch'essi basati su Atmel Atmega128, con 8K di RAM e 128K di ROM e radio 802.15.4 compliant della Chipcon. La particolarità sta nel fatto che sono dotate di modulo clock hardware per la sincronizzazione globale e di una porta *SDIO (Secure Digital Input Output)* per l'utilizzo di schede di memoria flash, come si nota in Figura 2.8.

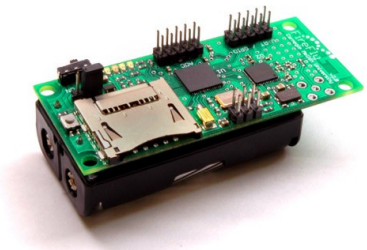


Figura 2.8: Un nodo FireFly

- **TelosB**: prodotti dalla Crossbow [85] ma ideati dall'università di Berkeley, sono caratterizzati dalla presenza del microcontrollore Texas Instrument MSP 430 con 10K di RAM, modulo radio compatibile con IEEE 802.15.4 e porta USB per la programmazione e raccolta dati (Figura 2.9).

Figura 2.9: Un nodo TelosB

- **altre soluzioni commerciali:** a differenza delle soluzioni fin ora citate, queste piattaforme sono sviluppate interamente da aziende, invece che da centri di ricerca universitari. Alcuni produttori di queste tecnologie sono Millenial [86] ed Ember [87].

2.6 ARCHITETTURA DI RETE

Per descrivere l'architettura di rete di una Wireless Sensor Network, si può ricorrere al modello *OSI (Open Systems Interconnection)*, riportato in Figura 2.10, tenendo però ben presente che la separazione dei layer in una WSN non è così netta; al contrario, i layer si sovrappongono tra loro, in particolare il livello 2 e il livello 3. Questa è una scelta progettuale dovuta, per motivi che vanno ricondotti alle caratteristiche e ai limiti di una Sensor Network. È insomma una conseguenza dei trade-off elencati nei paragrafi precedenti: per creare dei protocolli di comunicazione efficienti, leggeri, capaci di supportare avanzate funzionalità di risparmio energetico e flessibili ai cambiamenti della rete stessa, il livello Mac e il livello Network si trovano spesso a lavorare in stretta collaborazione tra loro, violando così il principio di separazione dei livelli.

OSI Model			
	Data unit	Layer	Function
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation and encryption
		5. Session	Interhost communication
	Segments	4. Transport	End-to-end connections and reliability (TCP)
Media layers	Packets	3. Network	Path determination and logical addressing (IP)
	Frames	2. Data link	Physical addressing (MAC & LLC)
	Bits	1. Physical	Media, signal and binary transmission

Figura 2.10: Il modello OSI

I livelli che costituiscono il cuore di una WSN sono il secondo e il terzo, proprio poiché vanno a definire il "carattere" della rete, ovvero sono questi due livelli che si occupano di risolvere le sfide più difficili di una reti di sensori wireless. I layer rimanenti assumono invece un'importanza minore, specialmente dal quinto in su. È vero che anche il livello fisico svolge un ruolo fondamentale, senza il quale la rete non esisterebbe nemmeno; tuttavia tutte le reti di sensori senza fili a bassa potenza sfruttano apparati radio con caratteristiche simili, ovvero bassa potenza di trasmissione, basso consumo, basso bitrate. Quindi, al di là di fattori come la frequenza o il tipo di modulazione adoperato, il profilo di una WSN a bassa potenza non è definito dal livello Fisico.
Il livello del Trasporto, se viene implementato, non è definito da particolari regole e, in ogni caso, non è complicato come nelle reti classiche TCP/IP.
I livelli 5, 6 e 7 sono solitamente validi solo per i gateway o per gli host. Fanno eccezioni le reti ZigBee e altre soluzioni OEM, che implementano anche parte del livello Applicazione nei nodi.
Nei paragrafi successivi viene presentata una descrizione generale dei primi tre layer, lasciando gli approfondimenti al Capitolo 5.

2.6.1 Livello Fisico

Esistono varie tecnologie utilizzabili per le Wireless Sensor Network: Wi-Fi, 3G, Wimax, Bluetooth e 802.15.4/ZigBee. Dato che solitamente esse ricorrono all'802.15.4/Zigbee, le altre tipologie di connessione non verranno trattate. Infatti connessioni Wi-Fi, 3G o Wimax vengono per lo più sfruttate per il collegamento del nodo gateway ad Internet o ad un altro host, ma non per creare collegamenti tra i vari nodi. La Figura 2.11 mostra queste tecnologie a confronto per range di trasmissione e data rate, mentre la Tabella 2.1, propone un confronto su più aspetti.

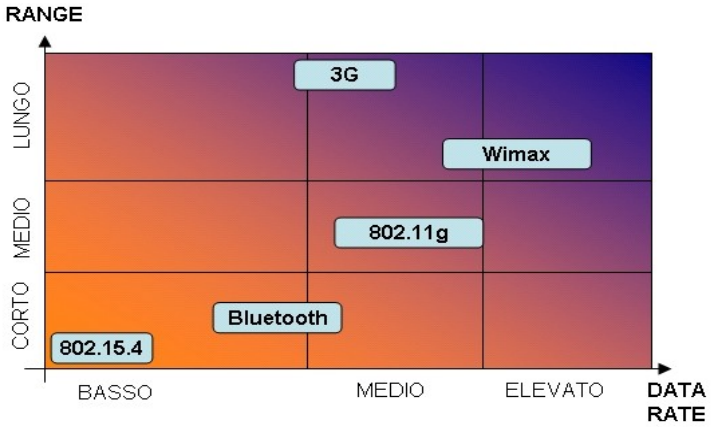


Figura 2.11: Confronto tra le varie tecnologie wireless disponibili

	802.11	802.15.1/Bluetooth	802.15.4/ZigBee
Copertura (metri)	100	10 - 100	10 - 75
Throughput (Mbps)	2 - 30	1 - 2	0.25
Consumo energetico	Medio	Basso	Molto basso
Autonomia	Minuti - poche ore	Diverse ore - pochi giorni	Giorni - pochi anni
Dimensioni	Medie	Piccole	Molto piccole
Rapporto costo/complessità	Alto	Medio	Basso

Tabella 2.1: Confronto tra varie tecnologie wireless (fonte: [20])

Gli aspetti che devono essere tenuti in considerazione a livello fisico, sono quelli tipici di una connessione senza fili:

- **Riflessione**: un'onda che si sta propagando incontra un oggetto più grande della sua lunghezza d'onda (muri, edifici,...).
- **Diffrazione**: la traiettoria tra trasmettitore e ricevitore è ostruita da superfici con contorni irregolari: l'onda si piega attorno all'ostacolo anche quando è assente il *line-of-sight (LOS).*
- **Diffusione (Scattering):** quando vengono incontrati oggetti che hanno una dimensione inferiore alla lunghezza d'onda dell'onda in propagazione (foglie, cartelli stradali, ...).

Questi fenomeni causano distorsione e attenuazione del segnale radio, che risulta essere la somma di varie componenti riflesse, diffratte e diffuse dagli ostacoli incontrati dall'onda. Tale effetto viene denominato *multipath;* infatti, riflessione, diffrazione e diffusione danno origine a traiettorie *(path)* di propagazione diverse, oltre a quella LOS tra mittente e destinatario. Quando ci sono più percorsi di propagazione radio, il segnale finale ricevuto è la somma vettoriale di tutte queste componenti provenienti da ogni direzione e da ogni angolo. Alcune di esse si sommeranno al segnale del path diretto, dando origine ad un'interferenza costruttiva, altre invece si sottrarranno, causando così un interferenza distruttiva.
Un altro fattore critico nella propagazione radio è la mobilità*:* anche quando le due parti comunicanti sono ferme, l'ambiente circostante può essere in movimento (ad esempio i rami delle piante che si muovono con il vento), provocando così improvvise fluttuazioni del segnale, dovute ai fenomeni appena descritti. Se si è all'interno di edifici, la situazione risulta ancora più difficile, a causa della presenza di muri, pavimenti, tubature e persone in movimento che degradano ulteriormente il segnale.
Infine, un problema critico che si verifica con i protocolli MAC di tipo *CSMA (Carrier Sense Multiple Access)* è quello del *nodo nascosto* e del *nodo esposto.*
Questo problema deriva dalle proprietà tempo-dipendenti del canale wireless, che è soggetto ai fenomeni appena spiegati di attenuazione, interferenza e diffusione. Queste limitano la massima distanza di trasmissione di un nodo. Proprio questi limiti e il fatto che CSMA tenta di evitare le collisioni ascoltando il segnale nella vicinanza del trasmettitore, danno origine ai problema del terminale nascosto e del terminale esposto.
Il nodo nascosto è definito come il nodo entro il raggio d'azione del destinatario ma fuori dalla copertura del nodo trasmettitore. Un esempio può essere il seguente (fare riferimento alla Figura 2.12): si supponga che B sia nel raggio di copertura di A e di C, ma A e C siano fuori copertura reciprocamente. A vuole trasmettere a B: effettua il sensing del canale, sente che non è occupato e avvia la trasmissione. Ad un certo punto, C deve spedire un messaggio a B, mentre A sta ancora trasmettendo a B. C effettua il sensing del canale, lo considera libero, perché non sente il segnale di A, e inizia a trasmettere B, causando così una collisione in B, senza che né A né C se ne accorgano. B a questo punto perde entrambi i messaggi, di A e di C.

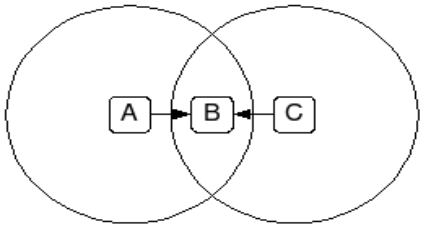


Figura 2.12: Il problema del nodo nascosto

Il nodo esposto è quel nodo nel raggio di trasmissione del mittente ma non del destinatario. L'esempio in questo caso è raffigurato in Figura 2.13: B è nel raggio di trasmissione di A e C, ma A e C sono fuori copertura reciprocamente, e D è nel raggio di trasmissione di C. Ora, B vuole mandare un pacchetto ad A: effettua il sensing del canale, sente che è libero e trasmette. A questo punto, C vuole trasmettere a D: effettua il sensing del canale, sente che è occupato dalla trasmissione di B e posticipa la sua trasmissione. Tuttavia, C avrebbe potuto trasmettere lo stesso subito, poiché D è fuori dalla portata di B e non si sarebbero verificate collisioni.

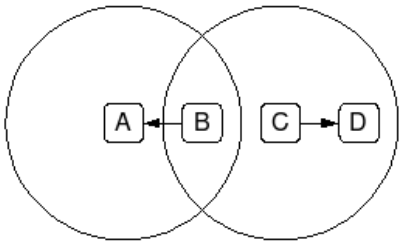


Figura 2.13: Il problema del nodo esposto

Per risolvere questi due problemi sono state proposte due soluzioni: la prima consiste nell'utilizzo di due canali, uno per i dati e uno di controllo. Quest'ultimo serve a segnalare agli altri nodi che è in corso la ricezione di un pacchetto. Tale soluzione però non è quella solitamente implementata nei dispositivi wireless. Infatti, il meccanismo generalmente usato è quello del *Collision Avoidance* (da cui il protocollo *CSMA/CA)*, che risolve però solo il problema del nodo nascosto. Tale tecnica consiste nella prenotazione del canale, da parte del nodo trasmettitore, tramite un pacchetto *RTS (Request to Send)*. Gli altri nodi nel raggio di copertura del trasmettitore confermano la ricezione dell'RTS spedendo un messaggio *CTS (Clear to Send)* al nodo e non occupano il canale per il tempo specificato nell'RTS.
Le caratteristiche del livello fisico dipendono ovviamente dalla tecnologia scelta, ma, come già accennato, per le WSN la tecnologia predominante è l'IEEE 802.15.4/ZigBee. Questa infatti consente di creare collegamenti a brevi distanza tra i vari nodi (tipicamente dai 10 ai 75 metri), con bitrate bassi, non oltre i 250 kbps, e bassi consumi energetici (grazie alla bassa potenza trasmissiva).
Dal punto di vista implementativo, nell'IEEE 802.15.4 è stato deciso l'utilizzo di una modulazione *DSSS (Direct Sequence Spread Sprectum)*, operativa in tre bande: 868 Mhz in Europa, 915 Mhz in USA e 2400 Mhz in tutto il mondo. Ulteriori dettagli sul livello fisico verranno forniti nel Capitolo 5.

2.6.2 Livello Data Link

Il livello *Data Link* è composta da due sottolivelli: *LLC* e *MAC*. Il primo, detto *Logical Link Control* layer, fa da tramite tra i livelli superiori e il livello fisico, consentendo l'interoperabilità tra diversi tipi di rete. Tuttavia, l'uso di questo *sublayer* è molto limitato, dato che l'interoperabilità è garantita da altri protocolli del network layer.
Il livello *MAC (Medium Access Control)* riveste invece un ruolo molto importante e si occupa di tre

funzioni fondamentali:

- l'assemblaggio dei dati in frame tramite l'aggiunta di un *header,* contenente informazioni sull'indirizzo, e un *trailer*, contenente informazioni per la correzione degli errori;
- il disassemblamento dei frame ricevuti per estrarre informazioni sull'indirizzo e la correzione degli errori;
- la gestione dell'accesso al mezzo di trasmissione condiviso.

Il requisito fondamentale di un MAC per reti WSN è la necessità di risparmiare più energia possibile. Ci sono diversi fattori che contribuiscono allo spreco energetico, come l'eccessivo *overhead*, l'*idle listening*, le collisioni di pacchetti e l'*overhearing*. Inoltre l'accesso regolato al mezzo trasmissivo comporta lo scambio di informazioni di controllo e di sincronizzazione tra i nodi. Il continuo scambio di tali dati può provocare perdite significative di energia. Lo stesso effetto e un *throughput* di rete diminuito possono essere causati anche da un prolungato ascolto del canale in attesa di pacchetti. La ritrasmissione di pacchetti persi a causa di collisioni è un'ennesima fonte di spreco, nonché di degrado delle prestazioni. Allo stesso modo l'overhearing, ovvero la ricezione e la decodifica di pacchetti destinati ad altri nodi, comporta inutili perdite prestazionali ed energetiche.
L'obiettivo di un MAC è quello di impedire queste situazioni appena descritte. Ci sono tre gruppi principali di protocolli MAC: *Schedule-based*, *Contention-based e ibridi.*
I primi regolano l'accesso al canale tramite una schedulazione, pre-assegnata ad ogni nodo, affinché un solo nodo alla volta possa occupare il mezzo trasmissivo.
I protocolli Contention-based non effettuano invece alcuna preallocazione delle risorse e l'accesso al canale condiviso è garantito *on-demand.* Se due nodi tentano l'accesso contemporaneo, si verifica una collisione. Il compito di questo tipo di MAC è quello di minimizzare, e non di evitare completamente, l'occorrenza di queste collisioni.
Per ridurre il consumo energetico, questi protocolli MAC differiscono tra loro nel meccanismo adoperato per minimizzare la probabilità di collisione, l'overhearing e l'overhead per il controllo di accesso al canale.
Il problema delle collisioni viene risolto tipicamente ricorrendo ad algoritmi casuali distribuiti per schedulare l'accesso al canale tra i contendenti. Per evitare l'overhearing, si mettono in stato dormiente i nodi (disattivando la radio) il prima possibile. Ciò può però provocare difficoltà di comunicazione tra nodi vicini, che possono essere risolte usando protocolli MAC con scheduling meno restrittivi.
Una descrizione esaustiva dei MAC studiati per le WSN si trova nel Capitolo 5.

2.6.3 Livello Network

Il livello Network si occupa principalmente del routing. Il routing ha un ruolo importantissimo, dato che lo scopo di una rete di sensori è quello di raccogliere dati da vari punti di un dominio, processarli e farli convogliare tutti in un unico punto (detto sink) dove risiede un'applicazione specifica. Per far sì che questi dati raggiungano la destinazione in maniera efficiente, cioè in maniera affidabile e con un basso costo energetico, è necessario che i protocolli di routing individuino dei cammini ottimali tra i nodi e il sink (o i sink).
Le caratteristiche di una WSN come la dinamicità (nodi che perdono il link o esauriscono la batteria, nodi aggiunti,...), la densità (che può variare molto di zona in zona), le risorse hardware limitate e la bassa disponibilità energetica, rendono complicato lo sviluppo di protocolli di routing.

2.6.3.1 Diffusione e raccolta dei dati

Il routing in una Wireless Sensor Network può essere scomposto in due fasi: diffusione dei dati e raccolta dati. Tipicamente, il processo di raccolta dati viene innescato dal verificarsi di un evento nell'ambiente o dall'applicazione: a questo punto i nodi effettuano le rilevazioni con i propri sensori, li diffondono eventualmente ad altri nodi per effettuare in-network processing (ad esempio

aggregazione di dati) e li trasmettono a nodi intermedi, i quali faranno lo stesso finché questo processo di raccolta dati raggiunge il nodo sink.
Questo schema di rete appena descritto viene definito *multi-hop,* poiché i nodi possono essere distanti più di un hop dal sink. L'approccio multi-hop è molto più conveniente rispetto al *sigle-hop*, sebbene possano esserci scenari in cui quest'ultimo è più indicato. In effetti, un approccio single-hop non consente una grande estensione geografica della rete e inoltre richiede alte potenze trasmissive, non adatte ad una WSN; inoltre usando MAC di tipo contention-based, i contendenti al canale potrebbero essere numerosi, aumentando così il rischio di collisioni.
In ambito multi-hop, invece, si possono facilmente raggiungere distanze molto lunghe, nonostante i nodi siano posizionati a poche decine di metri. Essendo così vicini tra loro, i nodi necessitano di una bassa potenza trasmissiva e la possibilità di contesa del canale con altri nodi è minore rispetto al caso single-hop. Purtroppo però ci sono anche aspetti negativi: far sì che il pacchetto spedito da un nodo lontano molti hop dal sink arrivi integro e senza un eccessivo ritardo è una delle sfide che il routing deve affrontare. Il suo compito è proprio quello di trovare quei nodi intermedi, situati tra il nodo lontano diversi hop e il sink, che consentiranno al pacchetto di arrivare a destinazione, tenendo sempre presente i limiti di energia e di banda di una WSN.

2.6.3.2 Sfide progettuali nel routing

Il routing in una Wireless Sensor Network si differenzia particolarmente dal routing di una qualsiasi rete cablata o *wireless ad hoc* proprio per le caratteristiche peculiari delle WSN:

- **Dinamicità e grandezza della rete**: come già accennato, la densità di una rete può variare molto di zona in zona, nuovi nodi posso entrare e altri posso uscire dalla rete in qualsiasi momento e i link tra nodi potrebbero cadere o degradarsi in seguito a variazioni ambientali. In tutti questi casi, una Sensor Network deve dimostrare la propria capacità di auto-configurazione, cambiando alcuni parametri e compiendo decisioni in base alla situazione presente. Gli algoritmi di routing sono *self-configuring*, ovvero sono in grado di reagire a questi cambiamenti improvvisi.
- **Risorse limitate:** le risorse limitate sono quelle hardware ed energetiche. Per rientrare nel primo limite, serve che gli algoritmi abbiano una ridotta occupazione di memoria (*footprint)* e siano facilmente eseguibili da CPU poco potenti. Per quanto riguarda il secondo, il fatto di avere una rete multi-hop è già di per sé un costo energetico maggiore rispetto ad una a single-hop, poiché nel primo caso più nodi devono essere tenuti accesi, ricevere e inoltrare il pacchetto, mentre in un single-hop, solo due nodi sono coinvolti. Quindi, l'unico modo per diminuire questo costo, è creare dei percorsi di instradamento che bilancino fattori come il numero di nodi intermedi, le risorse energetiche residue di ogni nodo e la potenza trasmissiva richiesta.
- **Data model dell'applicazione:** i *data model* descrivono il flusso di informazioni tra nodi e sink. Questi modelli sono dipendenti dall'applicazione in rapporto a quali informazioni, come vengono usate e quanto spesso vengono richieste. Sono stati proposti vari modelli di raccolta dati, a seconda delle applicazioni. Alcuni di essi richiedono i dati periodicamente o in base al verificarsi di certi eventi nell'ambiente. In altri, ogni nodo rileva, salva, elabora e aggrega i dati prima di inviarli al sink. In altri ancora, i dati vengono prelevati interattivamente tramite una comunicazione bidirezionale tra sink e nodi.
  Un protocollo di routing ideale dovrebbe essere ottimizzato per l'applicazione e al contempo essere in grado di supportare diversi data model, oltre a risultare affidabile, scalabile, efficiente e reattivo. Dal punto di vista pratico ciò non è facilmente realizzabile.

2.6.3.3 Strategie di routing

Il trade-off che un protocollo di routing deve bilanciare è quello tra capacità di rispondere ai cambiamenti ed utilizzo efficiente delle risorse. In altre parole, serve il giusto compromesso tra

risorse hardware e di banda limitate e overhead necessario per adattarli a queste. L'overhead in una WSN viene misurato in termini di occupazione di banda, utilizzo energetico e risorse di calcolo.
Per valutare se esistessero già protocolli in grado di soddisfare questo trade-off, furono esaminati quelli adoperati nelle Ad Hoc Wireless Network.
In queste reti si usano tre tipi di routing che si differenziano tra loro in base a come l'informazione è acquisita e mantenuta e al modo in cui essa viene utilizzata per calcolare i percorsi ottimali.
La prima strategie è detta *routing proattivo* (o *table driven)* e consiste nel disseminare le informazioni di routing aggiornato a tutti i nodi. La rete può avere un organizzazione *flat* o *gerarchica*. In caso sia flat, il routing sarà ottimale ma l'overhead in seguito ad un cambiamento nella rete sarà enorme. Il routing gerarchico invece è adatto per reti dalle grandi dimensioni.
La seconda strategia è detta di *routing reattivo* e calcola il percorso solo per alcuni nodi *on demand*. Questi protocolli non prevedono il mantenimento di un informazione globale della rete, perciò, per scoprire nuovi cammini è necessario interrogare tutta la rete tramite un *flooding*, con le risposte che tornano indietro seguendo il percorso inverso.
Infine l'ultima strategia è detta di *routing ibrido,* adatta per reti molto grandi. In questo caso, i nodi vengono uniti in gruppi, definiti *cluster*, adiacenti. All'interno di un cluster il routing è di tipo proattivo, mentre tra i cluster il routing è reattivo. Uno svantaggio di questa tecnica è che mantenere dei cluster provoca un ulteriore overhead.
In definitiva, i metodi di routing delle ad hoc network non vanno bene per una WSN, poiché essi non sono adatti a situazioni di rete dinamiche o che crescono in dimensione, dato che overhead e traffico di rete sarebbero in continua crescita. Per questo motivo la ricerca scientifica ha prodotto diversi tecniche di routing, specifiche per le WSN; le principali verranno esaminate in dettaglio nel Capitolo 5.

2.6.4 Livello Trasporto

La natura data-centric delle WSN combinata con le risorse hardware limitate, rende il protocollo TCP inutilizzabile in questo ambito. Inoltre, le rete di sensori sono caratterizzate da un concetto di affidabilità differente dalle reti classiche. Per di più, i livelli stessi di affidabilità o di controllo di congestione possono dipendere dal tipo dei dati trasportati.
Quindi, le funzionalità di questo livello devono essere progettate trovando il giusto compromesso tra consumo energetico e politiche implementate. Queste ultime sono però dipendenti anche dal percorso seguito dai pacchetti: *forward path* (dai nodi verso il sink) o *reverse path* (dal sink verso i nodi).
Nel forward path si applica un principio di *event-reliability,* ovvero non ha importanza che tutti i pacchetti arrivino a destinazione, ma serve che arrivino almeno quelli necessari per un corretto monitoraggio dell'evento.
Al contrario, nel percorso inverso (reverse path), è richiesto che tutti i messaggi spediti dal sink giungano ai nodi destinatari. Questo perché i messaggi mandati dal sink contengono informazioni critiche per il controllo dell'attività del nodo, come query dei dati o istruzioni di programmazione. Quindi, in questo caso, sono necessarie delle regole di consegna dei pacchetti più severe.

2.6.5 Livello Applicativo

Protocolli di questo tipo per una generica WSN non sono ancora diffusi, sebbene in passato ne siano stati proposti alcuni, tra cui *Sensor Management Protocol (SMP), Task Assignment e Data Advertisment Protocol (TADAP) e Sensor Query Data Dissemination (SQDDP)*.
SMP fornisce operazioni software per compiti di amministrazione della rete, come data aggregation, clustering, sincronizzazione, localizzazione, accensione sensori, monitoraggio e riconfigurazione dei nodi, autenticazione e distribuzione di chiavi nelle comunicazione sicure.
TADAP gestisce l'assegnazione di “interessi” riguardo un attributo o un evento e la pubblicizzazione dei dati disponibili.

SQDDP fornisce all'utente le interfacce per avviare e rispondere a query e si occupa di assegnare identificativi ai nodi in base ad attributi o alla loro locazione.
Il discorso è diverso per le reti ZigBee. Infatti questo standard definisce anche il livello applicativo, che è composto dall'*Application Support Sublayer (APS)*, dal *Zigbee Device Object (ZDO)* e da oggetti applicativi definiti dal produttore. Nella Figura 2.14, si può osservare come è strutturato tutto lo stack di ZigBee.

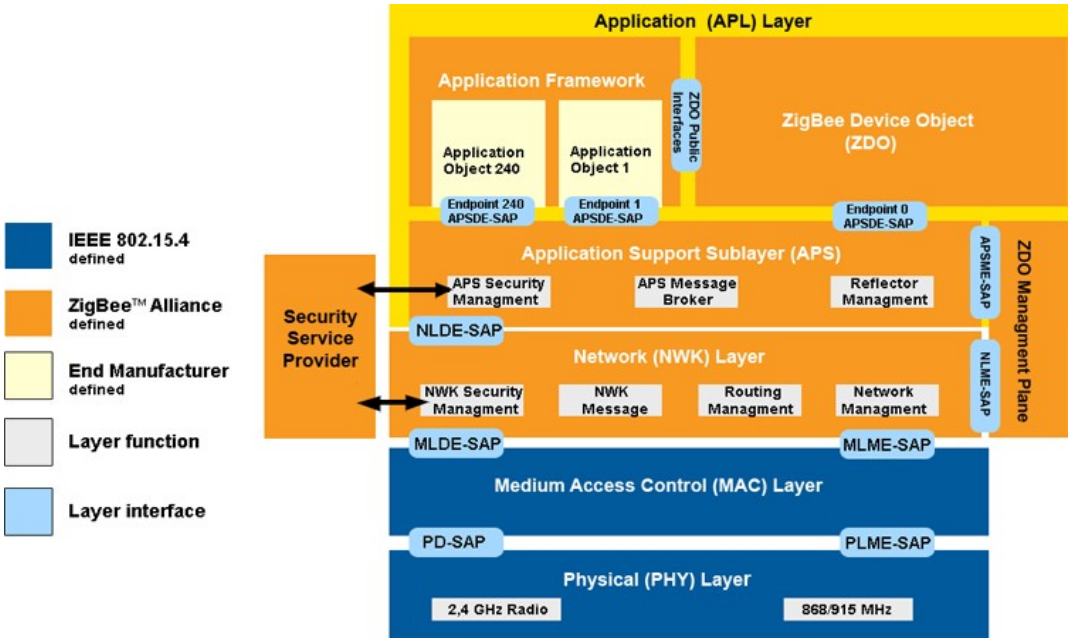


Figura 2.14: Architettura dello stack ZigBee

L'APS si occupa di mantenere le tabelle delle associazioni tra dispositivi caratterizzati da servizi ed esigenze proprie, e di inoltrare i messaggi tra dispositivi associati. Un altro suo compito è quello del *discovery,* cioè della ricerca di altri dispositivi che stanno operando nella zona.
ZDO definisce il ruolo dei nodi, inizializza e risponde a richieste di connessione e crea connessioni sicure tra nodi.
Gli oggetti applicativi definiti dal produttore implementano le funzionalità scelte in conformità alle regole dello standard ZigBee.

2.7 SISTEMA OPERATIVO

Ogni nodo di una WSN necessita di un sistema operativo che controlli l'hardware, fornisca un'astrazione del livello hardware al software applicativo e faccia da intermediario tra hardware e applicazioni. I sistemi operativi per le reti di sensori sono caratterizzati dai seguenti vincoli, determinati dalle scarse risorse computazionali:

- allocare uno stack di memoria per ogni processo non è possibile;
- la memoria condivisa tra processi non è protetta;
- il kernel è stato progettato secondo un modello *event-driven* o *Finite State Machine (FSM)*; il secondo approccio è conveniente per realizzare la sincronizzazione;
- devono essere fornite *API (Application Programming Interface)* modulari e generiche;
- il sistema operativo deve essere aggiornabile e riprogrammabile;
- non serve un file system, dato l'assenza del disco .

Le caratteristiche richieste ad un *SOS (Sensor Operating System)* sono queste:

- deve essere leggero e piccolo, al massimo qualche centinaio di kilobyte;
- dovrebbe possibilmente essere real-time;
- deve allocare efficacemente le risorse;

- deve supportare funzionalità di risparmio energetico;
- deve fornire un'interfaccia di programmazione generica per l'implementazione di middleware o applicazioni.

Il più famoso SOS è TinyOS [88], sviluppato secondo un modello event-driven usando il linguaggio *nesC*, che è un derivato di *C*. Le applicazioni per questo sistema operativo vengono create usando lo stesso linguaggio. Altri SOS, che supportano i linguaggio C, sono Contiki [89], Nano-RK [90], Btnut [81] e MANTIS [91].

2.8 MIDDLEWARE

Il middleware serve a colmare il gap tra i protocolli di rete e le applicazioni, attraverso delle funzioni di adattamento. La sua collocazione architetturale, come visibile in Figura 2.15, è tra il livello delle applicazioni ed i protocolli di rete; la sua funzione sta nel nascondere i dettagli implementativi dei livelli più bassi e facilitare lo sviluppo, il deployment e la gestione delle applicazioni.
Le sfide progettuali di un middleware sono: controllo della topologia della rete, elaborazioni data-centric efficienti, integrazione delle applicazioni nei protocolli di rete (per migliorare consumi energetici e prestazioni), sfruttamento efficiente delle risorse e supporto per applicazioni real-time.
Le funzioni basilari di un middleware sono quindi:

- fornire servizi di sistema alle applicazioni;
- coordinare le diverse applicazioni;
- adottare meccanismi per lo sfruttamento ottimale delle risorse di rete e hardware;
- allocare e ottimizzare le risorse di rete, bilanciando i trade-off di QoS.

Il middleware aiuta la negoziazione tra applicazioni e protocolli di rete, migliorando le prestazioni e l'occupazione di risorse. Per far ciò, deve conoscere le caratteristiche di entrambe le parti, analizzando le loro richieste e calcolando il miglior compromesso in base allo stato attuale della rete e ai QoS desiderati dalle applicazioni. Questa funzionalità può essere un servizio del middleware, che viene invocato dalle applicazioni.
I servizi di un middleware forniscono informazioni su un'applicazione e la qualità dei suoi servizi, oltre che sullo stato della rete attuale. In questo modo possono gestire al meglio le risorse della rete stessa. È anche possibile che i middleware informino le applicazioni di cambiare QoS, ma ciò richiede che le applicazioni siano adattive. Infine, il middleware si può occupare anche di diffusione dei dati, della loro compressione e della loro memorizzazione.

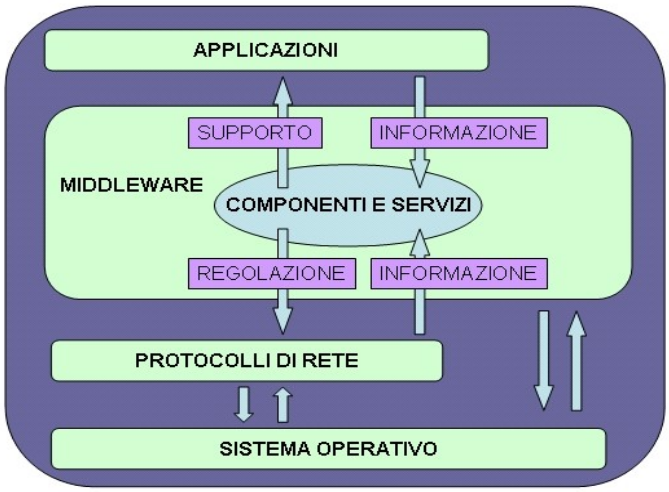


Figura 2.15: Architettura software di una WSN

I middleware ad oggi disponibili sono diversi: *MiLAN* [92] si propone di mantenere in vita la rete il più a lungo possibile; *AMF* [93] cerca un compromesso tra risorse e prestazioni dell'applicazione in fase

di raccolta dati; altri ancora, come *IrisNet* [94], considerano la WSN come un database distribuito su cui effettuare delle query; *Dfuse* [95] è usato per la fusione di dati; *DsWare* [96] è un middleware affidabile ed a basso consumo energetico per il rilevamento di eventi; *Em** [97] fornisce una serie di tool per lo sviluppo di applicazioni WSN; *SensorWare* [98] è un *agent-based* middleware, dove "un agente", ovvero un piccolo programma o *mobile control script,* viene iniettato nella rete per raccogliere i dati di certi nodi.

2.9 SVILUPPI FUTURI

Dal punto di vista hardware, l'evoluzione tecnologica consentirà lo sviluppo di microcontrollori sempre più piccoli, efficienti e performanti, come si è già visto nei primi paragrafi, quando si è parlato del chip Spec (Figura 2.16), che integra microcontrollore e radio in una piattaforma grande quanto la punta di una penna. Anche i sensori subiranno miglioramenti non trascurabili: avranno dimensioni sempre più piccole grazie alle tecnologie MEMS (Figura 2.17) e NEMS, saranno più accurati, meno costosi, più efficienti, affidabili, in grado di sopportare condizioni di utilizzo estreme e potranno rilevare nuove sostanze chimiche, individuando quelle tossiche, esplosivi e agenti biologici.

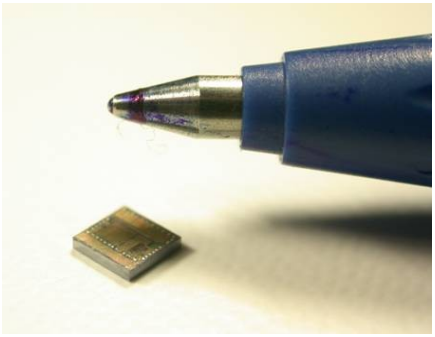


Figura 2.16: Il chip Spec

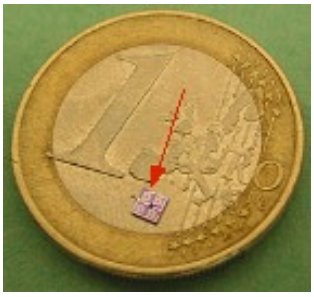


Figura 2.17: Sensore di pressione MEMS

Da una prospettiva software sarà necessaria l'adozione di standard di comunicazione tra sensori e lo sviluppo di sistemi di *service discovery,* in modo che si possano creare dialoghi tra nodi di diverse reti.

Grazie a questi fattori, le WSN diventeranno sempre più diffuse in qualsiasi contesto, in una moltitudine di applicazioni: in ambito domestico, i sensori si troveranno in molti elettrodomestici e dispositivi, compresi quelli di illuminazione e riscaldamento; in ambito biomedico sarà possibile monitorare diversi parametri corporei in tempo reale; in ambito militare i sensori saranno disseminati sui campi di battaglia, sui veicoli e sugli equipaggiamenti delle truppe; in ambito industriale e commerciale si assisterà alla rivoluzione dell'agricoltura di precisione, al tracking dell'inventario e delle catene di montaggio, al monitoraggio degli impianti produttivi. Nell'ambito dei trasporti, le strade saranno delle lunghe reti di sensori per segnalare in tempo reale traffico o condizioni di pericolo e le automobili potranno essere dotate di nuovi meccanismi di prevenzione degli incidenti e dei furti; nell'ambito della robotica, sarà possibile creare micro-robot autonomi e in grado di comunicare tra loro.

Tutta questa innovazione però solleverà problemi legali e di sicurezza. Infatti serviranno nuove leggi per regolamentare l'utilizzo di queste tecnologie in situazioni particolari di privacy e sarà necessario implementare meccanismi di sicurezza per evitare il *tampering* dei dati, ovvero la modifica volontaria dei dati rilevati dai sensori, piuttosto che la lettura di dati personali (ad esempio dei propri sensori biologici) o la diffusione di virus nella rete.
In definitiva, il potenziale per un'importante rivoluzione della nostra vita quotidiana c'è, ma il lavoro da affrontare è ancora molto.

CAPITOLO 3

MONITORAGGIO AMBIENTALE TRAMITE WSN

3.1 COSA È IL MONITORAGGIO AMBIENTALE

L'obiettivo di questa tesi è creare un sistema di monitoraggio ambientale. Monitorare un ambiente significa raccogliere, tramite sensori, informazioni in un'area, salvare e/o visualizzare queste informazioni ed eventualmente passarle ad altri sistemi per analizzarle. Quali debbano essere le informazioni da prelevare dall'ambiente dipende dagli obiettivi.
Il sistema tipico di monitoraggio ambientale tramite WSN, è costituito dai nodi, solitamente dotati di sensori di luminosità, temperatura e umidità, distribuiti nell'area interessata. Tali nodi mandano i dati rilevati ad un sink o gateway, il quale è connesso ad un host atto alla memorizzazione e alla gestione dei dati.
Il risultato finale è che l'utente (o un sistema) può consultare (ad esempio tramite browser), anche in tempo reale, le informazioni di un'area remota e agire di conseguenza.

3.2 UTILITÀ DEL MONITORAGGIO AMBIENTALE

Monitorare l'ambiente ha diversi scopi, a seconda dell'ambito di utilizzo e del fine che si vuole perseguire. Un primo fine è la ricerca scientifica: avere informazioni dettagliate sul nostro pianeta, aiuta a comprendere meglio i fenomeni naturali o causati dall'uomo. Sicuramente, uno degli ambiti che trae gran beneficio dal monitoraggio ambientale è la ricerca sul riscaldamento globale: infatti, per creare modelli teorici su questo fenomeno servono quanti più dati possibili da qualsiasi area del pianeta, dal fondo degli oceani fino all'atmosfera.
Un altro fine è quello di prevenire i disastri naturali: ad esempio sarebbe possibile avvertire la popolazione in caso di inondazioni imminenti, terremoti [3] e smottamenti. Progetti simili sono già in fase di sviluppo avanzato: in Honduras il MIT sta lavorando per creare un sistema che avverta la popolazione quando il fiume sta per esondare [4]; altri ricercatori ne stanno sviluppando uno per le piene improvvise in un paese del Venezuela [2]. Un altro disastro ambientale che sarebbe possibile prevenire è quello degli incendi boschivi: se le guardie forestali venissero avvertite non appena un piccolo focolaio si sviluppa, sarebbe possibile estinguerlo subito, anche in virtù del fatto che si conoscerebbe la sua posizione nel bosco.
Un altro fine ancora è il controllo dei parametri ambientali, come il controllo della qualità dell'acqua di un fiume o di un lago, piuttosto che il rilevamento di sostanze tossiche disperse nel terreno.
Un ulteriore applicazione è quella dell'agricoltura di precisione, che, insieme allo studio sul riscaldamento globale, è uno degli ambiti in cui le Wireless Sensor Network sono ora più adoperate.
L'agricoltura di precisione, è una forma di agricoltura che prende in considerazione diverse variabili per ottenere un ottimo raccolto usando solo le risorse necessarie. Essa consiste nel continuo monitoraggio dei campi tramite sensori, immagini satellitari e GPS, al fine di compiere le azioni giuste nel punto giusto al momento giusto. Per fare un esempio, l'agricoltore sa quale piante irrigare, in che quantità e in quale momento. Ciò porta a notevoli vantaggi sotto diversi punti di vista: vantaggi economici, poiché le risorse vengono utilizzate nella quantità necessaria e solo dove servono effettivamente, vantaggi produttivi, poiché conoscendo meglio il proprio terreno e trattandolo meglio, il raccolto aumenta e vantaggi ambientali, poiché è possibile risparmiare importanti risorse come l'acqua e moderare il ricorso a pesticidi, agenti chimici e fertilizzanti.

3.3 RETI WSN PER IL MONITORAGGIO ESISTENTI

Sistemi per il monitoraggio ambientale tramite WSN sono già stati realizzati sia in ambito universitario che commerciale. La maggior parte di essi sono stati creati per l'agricoltura di precisione. Nei paragrafi seguenti, verranno illustrate le caratteristiche e il funzionamento di tali sistemi.

3.3.1 Monitoraggio di una risaia

In [5] viene presentata una soluzione per il monitoraggio di una risaia, che permette di salvare i dati rilevati in un database centrale e di avvisare via *SMS (Short Message Service)* il responsabile di eventuali parametri anomali. I nodi della rete effettuano rilevazione periodiche ma è anche possibile richiederle manualmente. Per la consultazione dei dati è stato sviluppato anche un applicazione *MIDP (Mobile Information Device Profile)* per cellulari, di cui sono visibili due screenshot in Figura 3.1.
L'architettura del sistema, presentata in Figura 3.2, prevede una serie di nodi disposti a griglia nella risaia, che mandano i dati ad una stazione centrale, che a sua volta li spedisce via Internet ad un database centrale per ulteriori elaborazioni.

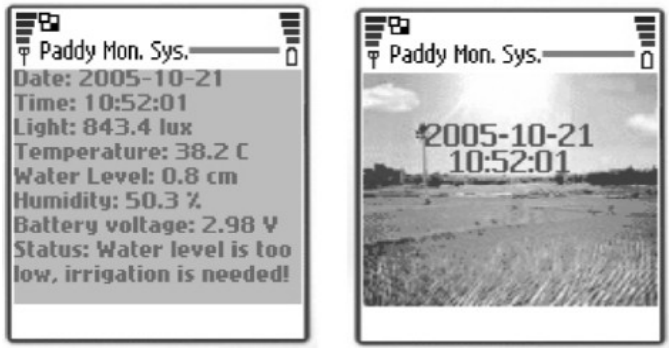


Figura 3.1: L'applicazione MIDP per cellulari sviluppata

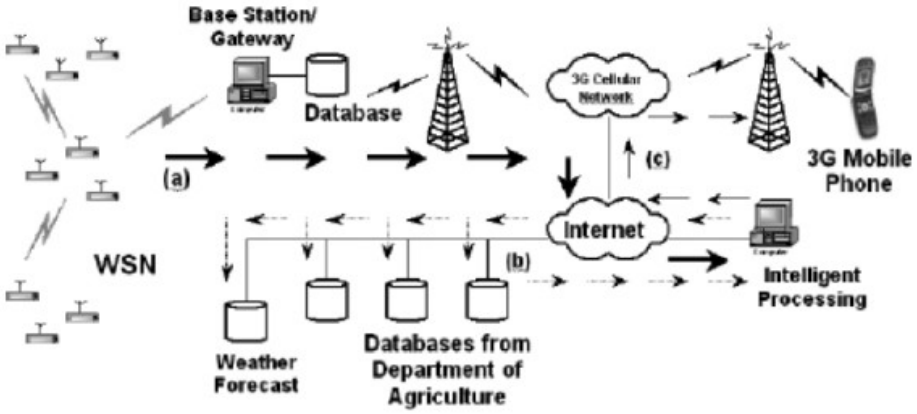


Figura 3.2: Architettura di rete del sistema realizzato

Le informazioni raccolte sono temperatura, umidità e livello dell'acqua. Sono state scelte queste grandezze per motivi precisi: la temperatura deve essere tra i 10°C e i 45°C per consentire la germinazione e, inoltre, la presenza di microclimi nel campo attrae diverse specie di insetti. Differenti livelli di umidità determinano lo sviluppo di differenti malattie delle piante; infine, un giusto livello dell'acqua determina una corretta crescita della pianta.
I nodi sono dei Mica Mote con sistema operativo TinyOS e comunicano con il gateway, o stazione base, tramite un protocollo di routing detto *ISBG (Information Selection Branch Growing)*, sviluppato dagli stessi autori della rete.
Tale protocollo è basato su un algoritmo di routing ad albero: l'idea centrale è quella di creare una rete in cui tutti i nodi gestiscano la stessa quantità di traffico, in modo che nessun nodo sia più

utilizzato degli altri e, quindi, che nessun nodo esaurisca la propria batteria molto prima degli altri.
Inoltre, per minimizzare il *delay end-to-end*, vengono sviluppati i rami della rete dove i nodi foglia hanno un minor numero di hop dal gateway.
Dai test effettuati, questo protocollo si è comportato molto meglio rispetto ad altri protocolli esistenti, sia in termini di lifetime della rete (vedasi Figura 3.3), sia nella propagazione dei dati (vedasi Figura 3.4).

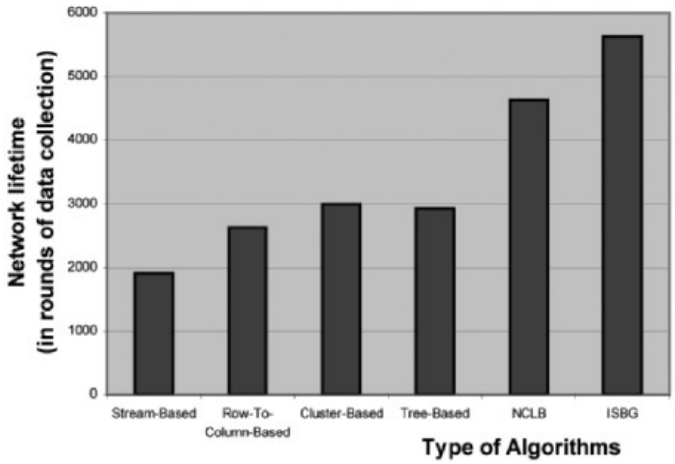


Figura 3.3: Confronto del lifetime tra vari algoritmi di routing

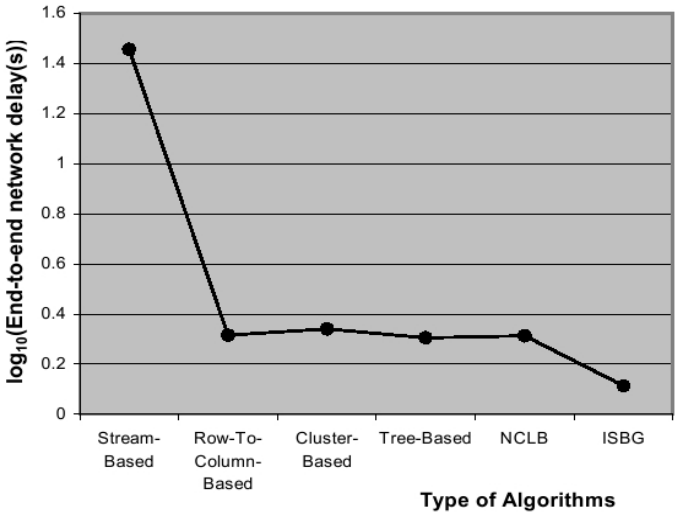


Figura 3.4: Confronto del delay in vari algoritmi di routing

3.3.2 A^{2}S

A^{2}S è l'acronimo di *Automated Agriculture System* ed è un sistema di monitoraggio ambientale sviluppato per campi agricoli e serre [6]. È composto dalla rete WSN e da un'applicazione di gestione della rete.
Gli autori hanno sviluppato su piattaforme compatibili 802.15.4 a 2.4Ghz con sistema operativo ANTS-EOS. L'algoritmo di routing scelto è ad albero e la possibilità di far dormire i nodi della rete è comandata da un pacchetto inviato dai sink; la durata dello stato di *power-saving* è impostabile dal software di gestione. Oltre alla raccolta dei dati di luminosità, temperatura e umidità, è possibile anche attivare un attuatore, che comanda la regolazione delle luci. Il deployment è stato effettuato in una serra di meloni e cavoli, illustrata in Figura 3.5.

Figura 3.5: Il deployment dei nodi nella serra

L'architettura del sistema, schematizzata in Figura 3.6, prevede la presenza di tre sottoreti distinte, con tre relativi sink, connessi via Wi-Fi ad un unico gateway che inoltra i dati via Internet ad un web server, dove risiede il software applicativo.
Le informazioni rilevate sono consultabili tramite l'applicazione apposita via pc e via palmare.

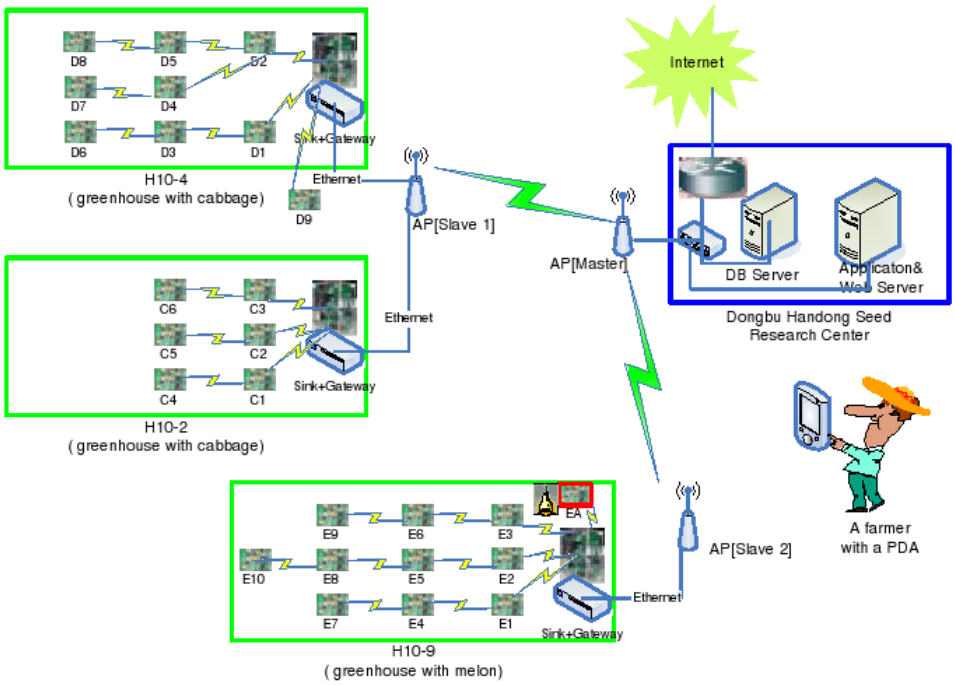


Figura 3.6: L'architettura di rete di A²S

Un risultato importante di questo studio riguarda la disposizione dei nodi: sebbene in situazione di Line of Sight i nodi possano essere a 70 metri di distanza, all'interno della serra non è possibile mantenerli a più di 20 metri l'uno dall'altro, probabilmente a causa della presenza di molti fili metallici e delle foglie delle piante.

3.3.3 Monitoraggio in serra

Il sistema di monitoraggio descritto in [7], prevede il rilevamento di temperatura, umidità e luminosità all'interno di una serra; i nodi utilizzati sono dei MicaZ con sistema operativo TinyOS. L'architettura di

rete, riassunta in Figura 3.7, è composta dai nodi della WSN che mandano i dati ogni 5 minuti al sink. Quest'ultimo è collegato via seriale ad un terminale GSM che spedisce, tramite SMS, i dati rilevati ad un pc remoto, dove verranno salvati in un database. L'utilizzo di tale tecnologia per comunicare i dati al punto centrale di analisi è la caratteristica di questo progetto: non ci sono altri studi che propongono una soluzione simile e la sua validità è discutibile, soprattutto da punto di vista economico, che deve essere aggiunto al costo di 400€ di ogni nodo.

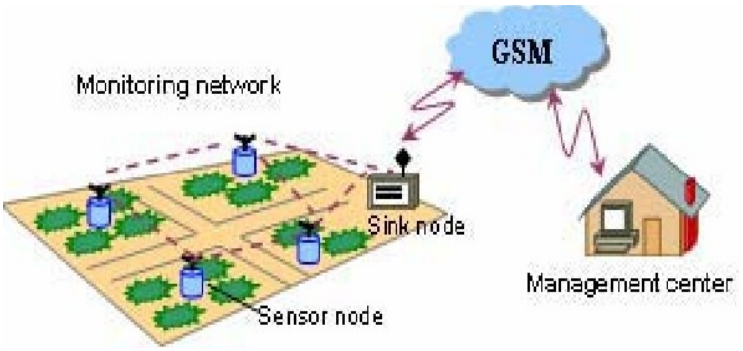


Figura 3.7: Architettura del sistema di monitoraggio serre

Purtroppo in questo articolo non viene detto nulla sui protocolli di rete, né sulle prestazioni del sistema. Vengono però riportate interessanti riflessioni sul posizionamento delle antenne, che verranno illustrate nel Capitolo 7.

3.3.4 Lofar Agro

Lofar Agro [8] è il nome di un progetto che prevede l'utilizzo di un sistema di monitoraggio tramite rete di sensori con l'obiettivo di combattere il batterio *Phytophtora*, che affligge le coltivazioni di patate.
Questo studio, insieme a quello di *habitat monitoring* [11], viene spesso citato in letteratura.

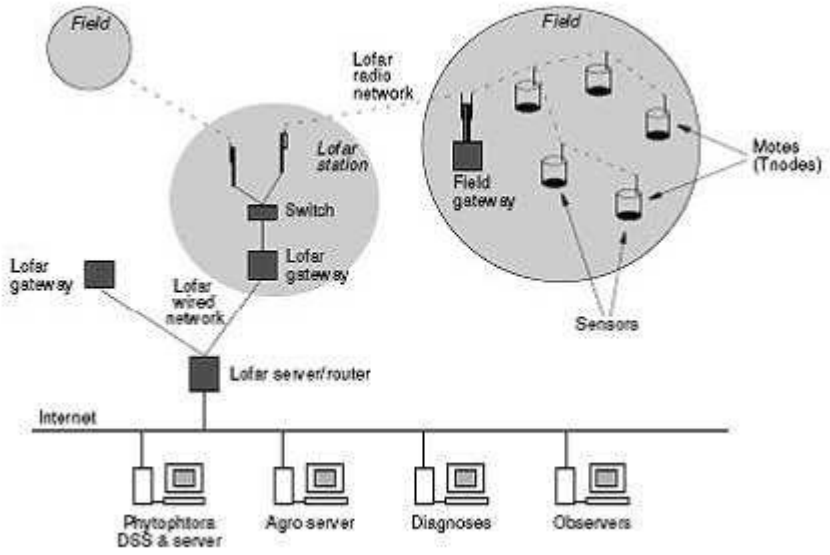


Figura 3.8: Architettura del sistema Lofar Agro

Il sistema, schematizzato in Figura 3.8, prevede una WSN composta da nodi *Tnodes* (vedasi Figura 3.9), simili ai Mica Motes, con sistema operativo TinyOS, dotati di sensori di temperatura e umidità, che mandano periodicamente i dati ad un sink collegato ad un gateway, il quale li inoltra ad un server via Internet, dove le rilevazioni sono accessibili via XML.

Figura 3.9: Un nodo della rete, chiamato Lofar Mote

Un server *DSS (Decision Support System)* elabora poi queste informazioni per dare suggerimenti agli utenti sulle azioni da intraprendere per prevenire la formazione del batterio (ad esempio consigli sull'uso dei pesticidi).
Questa rete si differenzia dalle altre appena descritte in molti aspetti: il MAC utilizzato è di tipo *T-MAC* (verrà spiegato nel Capitolo 5) e il protocollo di routing è il *MintRoute*, con aggiornamento delle tabelle di routing ogni ora. L'affidabilità richiesta è molto elevata: il gateway, una volta al giorno, verifica di aver ricevuto tutti i dati spediti dai nodi e, in caso negativo, richiede la loro rispedizione. Il nodo può recuperare il dato, poiché nella propria EEPROM mantiene tutti i dati rilevati non ancora confermati dal gateway.
Questo progetto, oltre allo scopo di informare l'agricoltore delle possibili condizioni alla sviluppo del batterio Phytophtora, è servito anche come test sul campo di una WSN con le caratteristiche riportate sopra.
Studi correlati [9,10] hanno inoltre dimostrato che quando il campo di patate è in fiore, il raggio d'azione dei nodi è di soli 10 metri e che l'*RSSI (Received Signal Strenght Indicator)* migliora in condizioni di umidità alta.

3.3.5 WSN per colture con irrigazione intensa

Il sistema descritto in [12] è nato con lo scopo di pianificare efficientemente l'utilizzo dell'acqua nei campi che richiedono molta irrigazione. L'architettura della rete, schematizzata in Figura 3.10 e in Figura 3.11, è composta da nodi MicaZ raggruppati in reti gestite da sink. I dati rilevati sono temperatura, umidità dell'aria, luminosità, acidità e umidità del suolo e vengono spediti periodicamente, tramite un gateway collegato ad Internet, ad un server con database che funge anche da DSS.

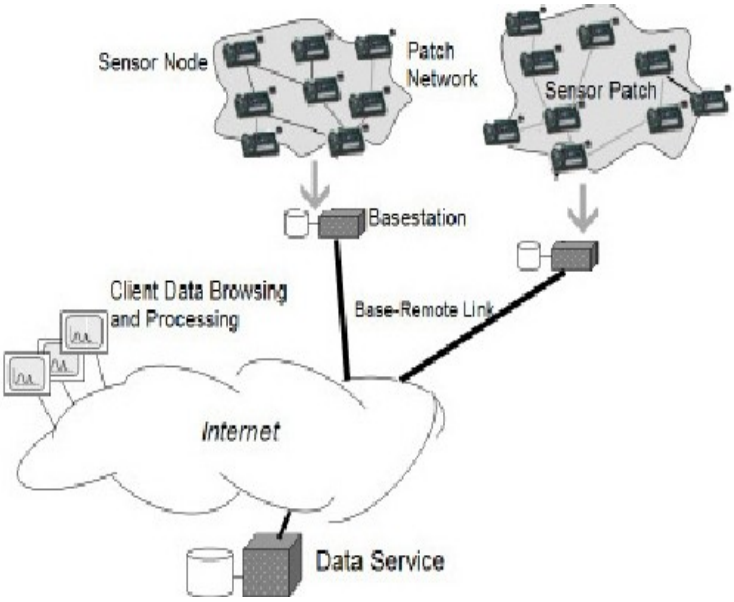


Figura 3.10: Architettura di rete

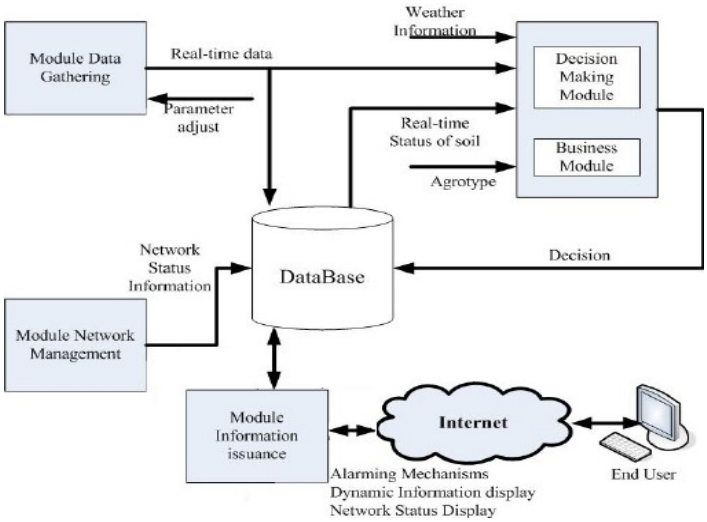


Figura 3.11: Architettura software del sistema

Il protocollo di routing scelto è ad albero e le tabelle di routing dei nodi vengono aggiornate ad intervalli regolari tramite messaggi scambiati tra i nodi. La coordinazione tra questi è garantita da un MAC di tipo S-MAC, con meccanismi di RTS (Request to Send) e CTS (Clear to Send) per minimizzare le collisioni.
È stata sviluppata anche un'applicazione per la consultazione dei dati rilevati e dello stato della rete e per la gestione della WSN. Alcuni screenshot si trovano in Figura 3.12.

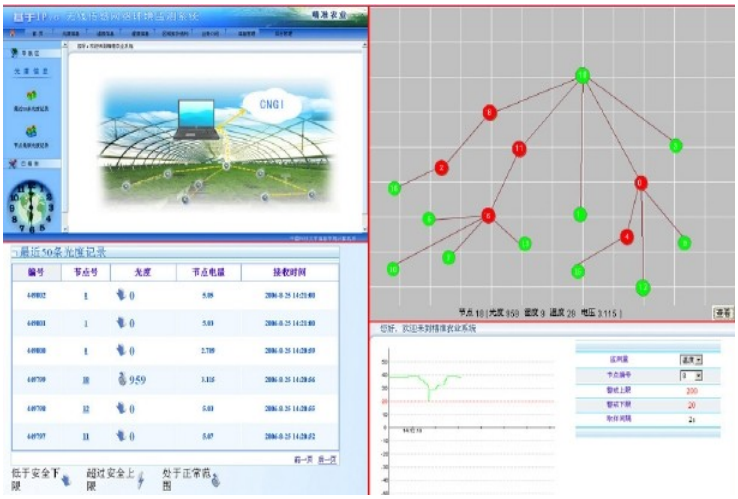


Figura 3.12: Software per la visualizzazione dei dati rilevati e per la gestione della WSN per l'irrigazione

3.3.6 WSN reattiva per la misura dell'umidità del suolo

Il progetto presentato in [13] ha avuto come obiettivo quello di misurare l'umidità del suolo in una foresta vicino a Perth per lo studio e la modellizzazione delle acque sotterranee della zona.
I ricercatori hanno preparato una rete ad albero, composta da sensori Mica2 con TinyOS, equipaggiati con sensori di umidità del terreno; uno di essi, è stato dotato di un pluviometro. I dati rilevati vengono spediti ad un sink connesso, tramite un interfaccia apposita, ad un pc con connessione GPRS (chiamato *Superlite*), che a sua volta li spedisce ad un database remoto. L'utente può poi visualizzare questi rilevamenti tramite un'interfaccia.
La caratteristica che contraddistingue questa rete è il suo comportamento di tipo reattivo, ovvero che reagisce ad un evento. Infatti, durante i momenti di pioggia, i nodi devono effettuare misurazioni più frequentemente, ad esempio ogni 10 minuti, mentre nei periodi di secca, è sufficiente anche una sola volta al giorno.

A questo fine, gli sviluppatori hanno implementato un MAC di tipo *S-MAC* che mantiene sincronizzati i nodi, un routing ad albero e un meccanismo di segnalazione in caso di pioggia. In tale situazione, i nodi cambiano intervallo di sleep e aumentano la frequenza di campionamento del suolo, poiché proprio in questi momenti si hanno veloci cambiamenti nelle acque sotterranee.
Le prove sul campo del sistema hanno mostrato non poche difficoltà: il tasso di pacchetti dati persi a livello di WSN ha sfiorato il 40% e anche quelli persi in fase di trasmissione GPRS non sono trascurabili. In quest'ultimo caso la causa è probabilmente la scarsa qualità della connessione mobile, con frequenti disconnessioni; a livello WSN la causa è riconducibile probabilmente alla notevole dinamicità della qualità del link. Nonostante i nodi fossero dotati di antenna esterna sopraelevata, a volte il range massimo tra due nodi non superava i 5 metri. Inoltre l'evento di pioggia non è stato comunicato correttamente in alcuni giorni. Un altro punto debole dei nodi è stato la loro robustezza: infatti il *case* stagno, costruito appositamente, non ha tenuto bene e in alcuni nodi è penetrata dell'acqua all'interno, mandando in tilt il sensore di umidità. Un ultimo problema, risolto però prima del deployment sul campo, è stato riscontrato con il software driver del sensore di umidità che non era corretto e restituiva valori errati di misurazione.
Per quanto riguarda l'autonomia della rete, il risultato migliore è stato di 191 ore con batterie *LISO2* da 8Ah, con un duty cycle dell'1%. Il consumo energetico durante le fasi di sleep è stato più alto rispetto alle specifiche dei Mica2, poiché la radio è stata lasciata spenta ma non la CPU e quindi, in fase di sleep, il nodo consumava 8 mA contro i 5 µA di specifica. Durante la fase di attività si sono registrati valori tra 5 mA e 20 mA. Infine è interessante che venga fornita anche una tabella riassuntiva dei costi sostenuti (Tabella 3.1 e Tabella 3.2), aspetto che di solito è difficile trovare sia in articoli di ricerca che in *case study* commerciali.

COMPONENTE	**COSTO**
Mica2 Mote	220€
Scheda di gestione dei sensori	326€
2 sonde di umidità per il suolo	200€
Pluviometro	165€
Box del nodo	9€
Batteria LISO2	29€
Scheda di interfaccia tra sink e Superlite	106€
terminale GPRS Superlite con antenna whip	351€

Tabella 3.1: Riassunto dei costi per il progetto

VOLTAGGIO	**TECNOLOGIA**	**COSTO**	**CAPACITÀ (Ah)**	**AUTONOMIA sul campo**
2 x 1.5V	Alcaline	2.6€	1.5	2 giorni
2 x 1.2V	NiMh	9.4€	2.3	8 giorni
1 x 3V	LISO2	19.8€	8	28 giorni

Tabella 3.2: Confronto tra le batterie adoperate

3.3.7 Solar Biscuit

Solar Biscuit [14] è un prototipo di rete creato con l'obiettivo di eliminare la necessità di una batteria per alimentare i nodi. Per raggiungere questo scopo, vengono utilizzati dei piccoli pannelli solari che caricano un capacitore *EDLC (Electric double-layer capacitor)*, il quale fornisce l'energia richiesta ai nodi (illustrati in Figura 3.13), equipaggiati di microcontrollore Microchip Pic a 7 Mhz, radio Chipcon a 315 Mhz e sensori di umidità e temperatura.

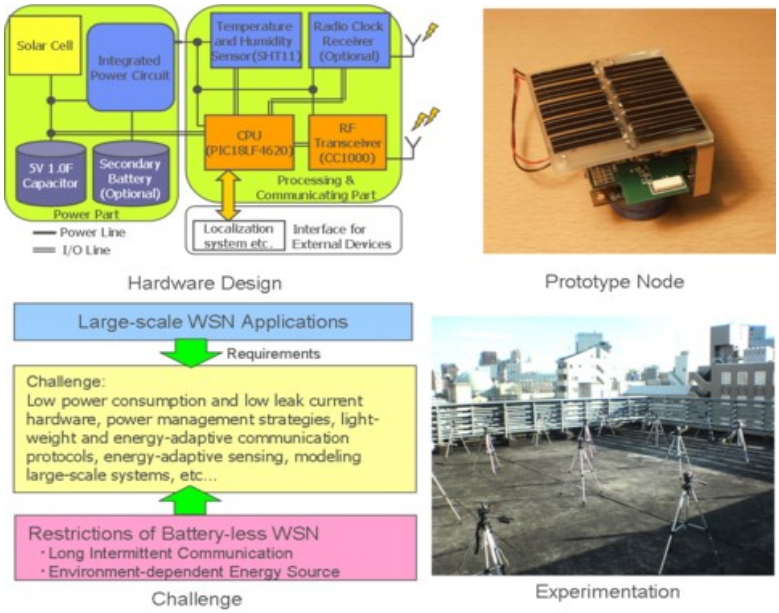


Figura 3.13: Architettura hardware del nodo e immagini di alcune fasi del progetto Solar Biscuit

Il problema principale nel realizzare un sistema simile è che il pannello fotovoltaico non fornisce abbastanza energia per un utilizzo immediato e, dunque, l'energia va accumulata per un certo periodo prima di poter essere utilizzata. Il consumo di un nodo nella sua fase attiva è pari a 25 mA, mentre il pannello solare fornisce al massimo 20 mA. Al fine di raggiungere lo stato di carica, ogni nodo esegue due fasi: la fase di *bootstrap,* in cui accumula energia nel capacitore, e la fase di *attività***,** in cui compie le misurazioni e le trasmette verso il sink. La prima fase termina non appena viene rilevato un certo voltaggio nel capacitore. Non appena il nodo entra nella seconda fase, attende un intervallo random, manda in broadcast dei messaggi di sveglia e prima di tornare nella fase di sleep invia i dati rilevati ai nodi che hanno risposto al messaggio di sveglia (vedasi Figura 3.14). Siccome i nodi non entrano nella seconda fase contemporaneamente, solo alcuni riceveranno il pacchetto mentre sono attivi e lo ritrasmetteranno in broadcast a loro volta. Per evitare la creazione di loop, è definito un parametro *TTL (Time To Live)*. Quando il nodo rileva una carica sotto una certa soglia oppure scade l'intervallo della fase attiva, ritorna alla fase di bootstrap.
In condizioni di emergenza, invece, il nodo fonte continua a mandare in broadcast il messaggio di allarme e i nodi che lo ricevono lo inoltrano in broadcast in continuazione, fino a quando non esauriscono la batteria.
Nell'ambito di monitoraggio ambientale, le grandezze solitamente misurate non variano velocemente e quindi attendere intervalli di trasmissione lunghi è una condizione accettabile. Tuttavia gli autori si propongono di realizzare un sistema che supporti anche le condizioni di emergenza, ovvero quelle situazioni in cui un nodo rileva un dato fuori i valori norma e deve notificare l'evento al sink il più presto possibile. Come soluzione è stato implementato un generico MAC CSMA (Carrier Sense Multiple Access) e un protocollo di routing a flooding per inoltrare i pacchetti al sink, in modo da non appesantire i nodi con tabelle di routing o messaggi di aggiornamento delle stesse.

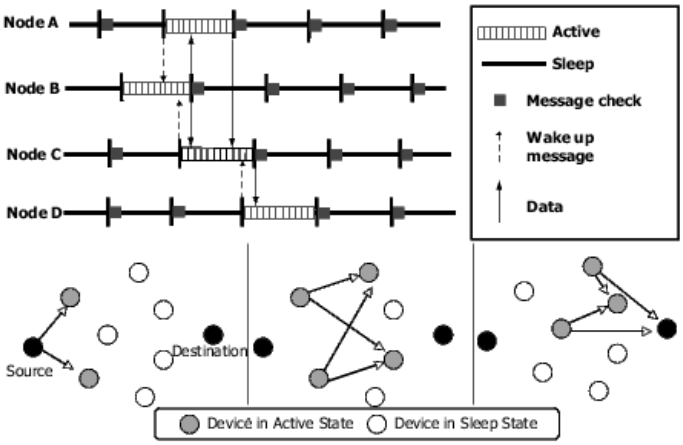


Figura 3.14: Schema riassuntivo delle due fasi di un nodo

Questa tipo di rete non è adatta ad applicazioni in cui si richiede una certa affidabilità nella ricezione dei dati: infatti i test condotti hanno dimostrato che con certi valori di intervallo e TTL, la perdita dei pacchetti può raggiungere circa il 70%, con delay di quasi 4.8 secondi. Queste pessime prestazioni sono dovute alla presenza di numerosi pacchetti duplicati che circolano nella rete e creano congestione. Buoni risultati sono stati ottenuti impostando intervalli di rilevazione pari a 60 secondi, con rate di perdita pacchetti del 10% e delay di 0.89 secondi.

3.3.8 SensorScope

SensorScope [15, 17] è un sistema di monitoraggio ambientale *out-of-the-box*, ovvero è un sistema completo e subito operativo. È stato sviluppato dal politecnico di Losanna inizialmente per monitoring all'interno di edifici [16]; recentemente è stato adattato anche per ambienti esterni, in particolare con lo scopo di raccogliere dati per migliorare i modelli che prevedono disastri naturali come valanghe, inondazioni, smottamenti e degrado ambientale.
La rete è composta da nodi *Shockfish TinyNode*, dotati di sistema operativo TinyOS, con CPU MSP430 a 16bit e 8Mhz, radio XE1205 a 868 Mhz da massimo 76kbps, 48 KB di ROM e 10 KB di RAM, alimentati da batterie ricaricate da pannelli solari e interfacciati con una serie di sensori di: temperatura, umidità, luminosità, velocità e direzione vento, pioggia e umidità del terreno. Il costo totale di ognuna di queste stazioni di rilevamento è di circa 1000€ e il loro ingombro non è molto ridotto, come visibile in Figura 3.15, essendo alte 150 centimetri.

Figura 3.15: Installazione di un nodo della WSN SensorScope

Riguardo ai protocolli di rete, il MAC implementato gestisce i duty-cycle della radio e gli acknowledge dei pacchetti ma non supporta il *Carrier Sense*, per cui viene adoperato un meccanismo di *backoff*

per l'accesso al canale (nella versione indoor del sistema il MAC era di tipo B-MAC).
Le tabelle di routing sono mantenute e aggiornate all'interno dai nodi dai beacon trasmessi dal sink e contengono l'età dei nodi vicini, la distanza in hop dal sink e la qualità del link. La scelta del percorso di instradamento dipende, oltre che dalla qualità del link rilevata non tramite RSSI (poiché è un metro di giudizio poco affidabile) ma tramite la trasmissione di bit di prova, anche da un fattore casuale, per bilanciare meglio il carico di lavoro tra i vari nodi. Per questo motivo, il tree routing che si crea viene definito *opportunistico.*
Tutti i nodi sono sincronizzati in maniera globale e per mantenere questa sincronizzazione, il nodo periodicamente manda un messaggio di richiesta sincronizzazione ad un nodo più vicino di lui al sink, che fornirà la risposta in broadcast, in modo da aggiornare il timing anche degli altri nodi. L'origine del segnale di sincronizzazione arriva dal sink, che inserisce nei suoi pacchetti di beacon il clock.
I dati, una volta giunti al sink, vengono spediti insieme al time offset al server centrale via GPRS, il quale imposta il giusto timestamp alle rilevazioni e le pubblica sul sito di *SensorScope* e su *Microsoft SensorMap*. Tutta questa architettura è schematizzata in Figura 3.16.

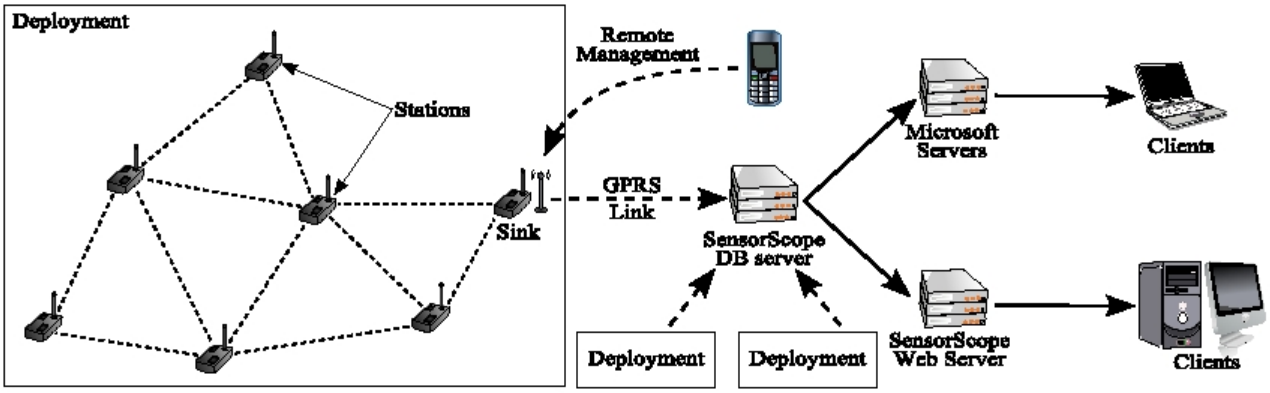


Figura 3.16: Architettura della rete SensorScope

I ricercatori hanno sviluppato anche una comoda e interessante interfaccia di consultazione dei dati e dello stato dei nodi (vedasi Figura 3.17): navigando all'interno della mappa è possibile selezionare un nodo, rappresentato da un *marker*, e visualizzare le rilevazioni dell'ultima ora in un grafico pop-up e in griglia.

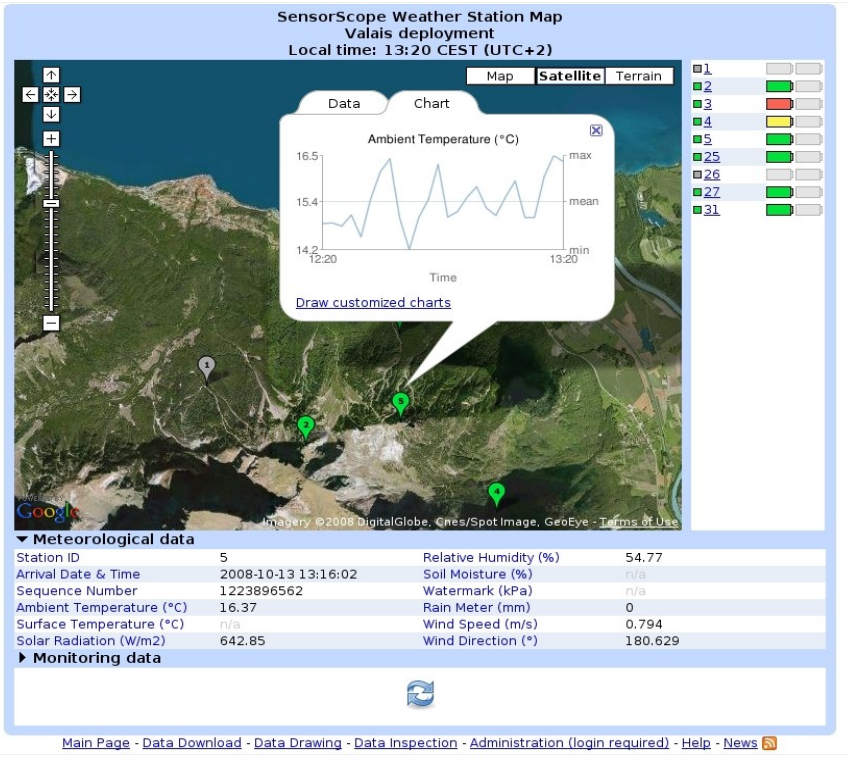


Figura 3.17: L'interfaccia del software di visualizzazione dati di SensorScope

Oltre a questo, è stato sviluppato un altro software per un analisi dati più approfondita (vedasi Figura 3.18).

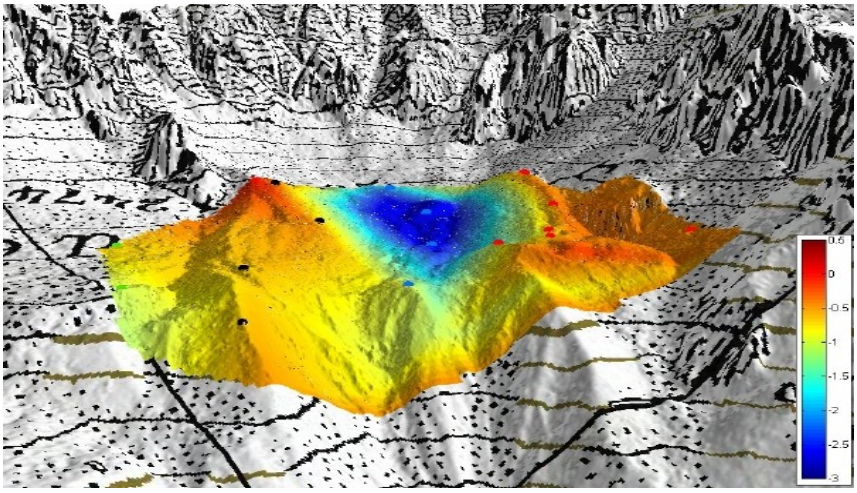


Figura 3.18: Analisi di temperatura di un territorio montuoso

I risultati delle prove sul campo, effettuati in alta montagna, dimostrano un buon comportamento del sistema. Dal grafico di Figura 3.19, si può notare come la perdita di pacchetti e il rate di pacchetti duplicati siano contenuti.

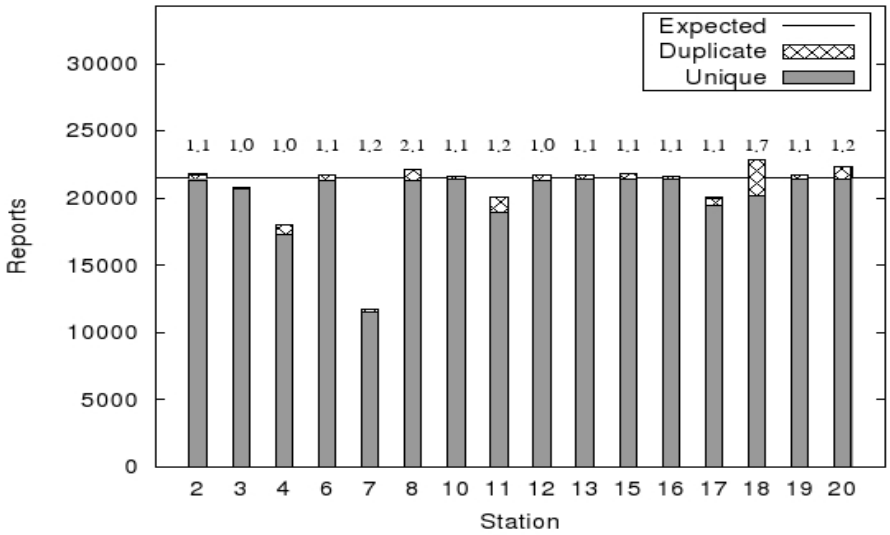


Figura 3.19: Numero di rilevazioni ricevute dai vari nodi

Negli sviluppi futuri del sistema, verranno aggiunte funzionalità di *network management*, con parametri gestibili dall'utente, verrà aumentato il numero di nodi per migliorare la robustezza della rete e meglio bilanciare il consumo energetico e infine verrà migliorata l'interfaccia, per consentire la visualizzazione di movimenti (come rocce che franano), di neve, nebbia e nuvole.

3.3.9 Camalie Net Wireless Sensing

La soluzione proposta da *Camalie Net* [18,19] è, come SensorScope, pronta all'uso ma, a differenza di quest'ultima, è di tipo commerciale e si rivolge in particolare ai viticoltori. La scopo del sistema è fornire ai coltivatori d'uva informazioni su come irrigare al meglio i propri terreni, così da massimizzare il raccolto e minimizzare il consumo dell'acqua. Inoltre è possibile monitorare umidità e temperatura per prevenire le gelate dei grappoli o la diffusione di batteri nocivi; monitorare il flusso e la pressione nell'impianto di irrigazione, nonché azionare attuatori per il controllo di pompe e valvole dello stesso.
Il primo prototipo del sistema è stato creato nel 2005, utilizzando sensori Mica2Dot con radio a 433

Mhz, ma a partire dalla fine del 2007 sono stati sostituiti con nodi della serie *eKo Pro* della *Crossbow* operanti sui 2.4 Ghz.
I dettagli implementativi della rete non vengono forniti, data la natura proprietaria della soluzione. In ogni caso la rete è di tipo mesh, auto-configurante e alimentata da batterie ricaricate da pannelli solari. I dati rilevati vengono mandati alla base-station, che è collegabile via Ethernet ad un pc. Da qui è possibile consultare direttamente le rilevazioni tramite il software *eKoView,* che crea grafici dei parametri monitorati oppure li sovrappone a immagini satellitari.
Il kit completo, denominato eKo Pro Series Starter Kit, rappresentato in Figura 3.20, viene venduto da Camalie Net a $3359 (circa 2500€) e comprende tre nodi con sensori di temperatura, umidità del suolo e del terreno, base station e software applicativo completo (web server, database e visualizzatore).

Figura 3.20: L'eKo Pro Series Starter Kit

Statistiche prestazionali della rete non sono note. Si conoscono invece i risultati del raccolto dopo l'implementazione di tale tecnologia. Mark Holler, proprietario del vigneto californiano dove questa soluzione è stata implementata per prima, nonché fondatore di Camalie Net, afferma che il suo raccolto è raddoppiato nel 2005 rispetto al 2004 (quando era privo della WSN) e il consumo di acqua è rimasto costante. L'anno successivo il raccolto è di nuovo raddoppiato, da 4 a 8 tonnellate. Stesso discorso nel 2007, da 8 a 16 tonnellate.
Sicuramente i fattori ambientali hanno una notevole influenza su questi risultati, ma è indubbio che l'utilizzo di una rete di sensori per monitorare i parametri dei propri vigneti abbia contribuito a questo successo. Naturalmente questa tecnologia può essere sfruttata non solo per i vigneti ma anche per qualsiasi altra agricoltura e qualsiasi forma di monitoraggio ambientale. Difatti, Crossbow vende la soluzione generica del kit, mentre Camalie Net quella specifica per i viticoltori.

3.3.10 Great Duck Island Habitat Monitoring

Questo sistema di monitoraggio dell'habitat di animali e piante è stato uno dei primi ad utilizzare una Wireless Sensor Network a tale scopo. Sebbene il progetto risalga al 2002, il lavoro presentato in [11] è da considerarsi molto valido anche oggi e, difatti, è uno degli articoli più citati in letteratura quando si parla di monitoraggio ambientale tramite WSN.
L'obiettivo del progetto, la cui architettura realizzata è schematizzata in Figura 3.21, era quello di studiare in che condizioni di habitat alcune specie animali e vegetariane vivessero. Per far ciò, sono stati posizionati 32 nodi Mica Mote in vari punti dell'isola Great Duck nel Maine (vedasi Figura 3.22), anche nelle tane degli animali. Ogni nodo è stato dotato di sensore di temperatura, umidità, luminosità, infrarossi e pressione atmosferica.

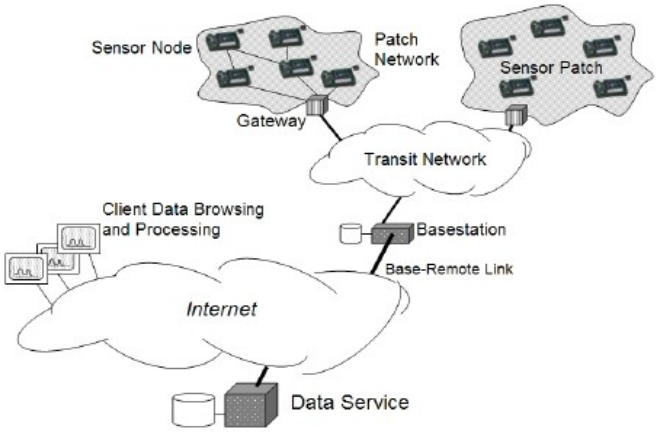


Figura 3.21: Architettura del sistema di habitat monitoring

I dati raccolti periodicamente dai nodi vengono salvati nella flash del nodo e spediti poi al sink, il quale è connesso tramite una rete transitoria alla stazione base. Questa è costituita da un mini-pc (con batteria alimentata da un pannello solare) che salva in un database locale i dati ricevuti ed è connesso ad Internet tramite un collegamento satellitare. I dati rilevati sono quindi prelevabili sia dal database di Great Duck che dal server centrale di Berkeley e sono interfacciabili, ad esempio, con applicazioni *GIS (Geographical Information System)*. In loco è possibile anche consultare tramite un palmare sia il database sia un nodo della WSN direttamente.
Riguardo i protocolli di rete, gli autori hanno scelto un duty-cycle MAC, ovvero l'accesso al canale radio viene schedulato, e il routing è ad albero. L'inoltro dei pacchetti avviene a livelli all'interno di ogni ramo, cioè le foglie più in basso svegliano quelle del livello superiore e inoltrano loro i pacchetti. Queste ultime faranno lo stesso con quelle del livello superiore.
Un altro aspetto che è stato ben tenuto in conto, è quello del consumo energetico, dato che la rete doveva durare 9 mesi. Il comportamento dei nodi è stato modellato sulla base dei consumi energetici di ogni componente, riportati nella Tabella 3.3.

Operation	nAh
Transmitting a packet	20.000
Receiving a packet	8.000
Radio listening for 1 millisecond	1.250
Operating sensor for 1 sample (analog)	1.080
Operating sensor for 1 sample (digital)	0.347
Reading a sample from the ADC	0.011
Flash Read Data	1.111
Flash Write/Erase Data	83.333

Tabella 3.3: Consumi di ogni componenti del nodo

Figura 3.22: Uno dei nodi posizionati

I risultati ottenuti hanno denotato un buon comportamento della rete, sebbene l'autonomia sia stata di 7 mesi circa, invece dei nove inizialmente previsti.

3.3.11 Libelium WSN

Libelium è uno spin-off dell'università di Saragoza che produce componenti per le WSN, come nodi, gateway e software di gestione della rete. I nodi sono chiamati *Squidbee* (vedasi Figura 3.23) e sono basati sulla piattaforma open source *Arduino*, con modulo radio *XBee* compatibile con l'802.15.4 e sensori di temperatura, umidità e luminosità. Questi sono i nodi che sono stati scelti per il progetto di questa tesi, perciò verrà fornita una descrizione più dettagliata nel prossimo capitolo. Il costo di ognuno è di circa 120€.

Figura 3.23: Un nodo Squidbee

Il gateway, chiamato *Meshlium*, è un *mini-pc* con processore *Via Eden* e può essere equipaggiato con moduli dual-radio Wi-Fi, GPRS, Bluetooth e ovviamente ZigBee. Il prezzo del modello base è di 525 €. Insieme al gateway è incluso un applicativo che consenti di gestire e monitorare i parametri della rete (Figura 3.24). Non vengono fornite particolari funzionalità per la raccolta dei dati dai nodi e la loro visualizzazione ed analisi. Anche i nodi vengono dotati di un semplice codice di base che non implementa alcun protocollo di rete (tranne quello MAC già presente nel modulo radio XBee). Questo poiché Libelium intende distribuire solo i mezzi per sviluppare quello che si vuole, e non dare all'utente una soluzione pronta. Quindi è necessario che l'eventuale acquirente sviluppi da sé tutta l'architettura di rete.

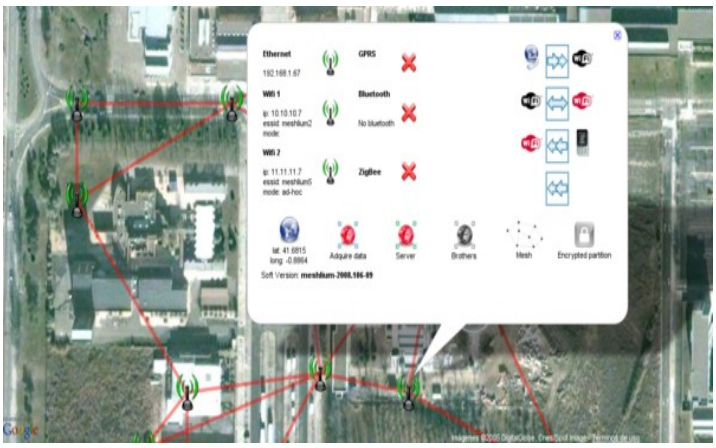


Figura 3.24: L'interfaccia di gestione della WSN

3.4 PREGI E DIFETTI DELLE SOLUZIONI DISPONIBILI

Le soluzioni di monitoraggio appena illustrate rappresentano la maggior parte di quelle esistenti ora, sia sotto forma open source che commerciale. Inoltre sempre più aziende si stanno lanciando nel campo delle Wireless Sensor Network, e nuovi prodotti continuano ad essere sviluppati. Tra le soluzioni descritte nei paragrafi precedenti, alcune sono servite soprattutto come test sul campo, per provare nella realtà il comportamento di una WSN invece che simularlo con il computer o in laboratorio, altre invece sono veri e propri sistemi maturi, affidabili e utilizzabili senza problemi nel loro scopo.
La soluzione migliore tra quelle presentate è sicuramente SensorScope, poiché è quella che riassume meglio le caratteristiche di una WSN: è totalmente auto-configurante, basta posizionare i

nodi, è facile e immediata da implementare e tutto è già pronto; il software applicativo permette un efficace monitoraggio e una grande versatilità nella gestione dei dati; tutto il codice sviluppato è open source ed è liberamente modificabile; la rete è performante e robusta; l'architettura di rete è flessibile e adattabile anche a condizioni con collegamenti difficili. Gli unici difetti, se così si possono definire, sono dovuti ai sensori installati, che sono ingombranti e soprattutto costosi, pur permettendo una notevole accuratezza delle rilevazioni.
Un'altra soluzione interessante è quella di Camalie Net (e Crossbow). Anche questa, come SensorScope, riassume bene le caratteristiche di base di una WSN: auto-configurabilità, semplicità di utilizzo, comodo sistema di monitoraggio e analisi dati. Ciononostante, presenta gli stessi difetti di SensorScope, ovvero costo e ingombro. In questo caso, il costo è ancora maggiore rispetto a SensorScope, poiché, nonostante il prezzo totale sia circa di 2400 €, i sensori di ogni nodo sono solo tre e sicuramente non al livello di quelli di SensorScope. Anche il pannello solare, che influisce notevolmente sul prezzo, è molto più piccolo in questo caso. D'altronde, essendo un prodotto commerciale, è normale che i margini di profitto siano più alti rispetto ad un prodotto “universitario”.
Le soluzioni come quella per i campi dall'irrigazione intensa descritta nella Sezione 3.3.5, oppure quella descritta nella Sezione 3.3.1, oppure A^2S**,** hanno le potenzialità di essere sistemi molto validi: l'architettura di rete sembra affidabile e performante (fatta eccezione per quella descritta nella Sezione 3.3.5, che non è possibile giudicare perché non riporta statistiche a proposito) e la dimensione dei nodi non è eccessivamente grande. L'aspetto in cui però devono maturare è quello dell'interfaccia, che deve essere più flessibile e potente e al contempo facile da utilizzare. Stesso discorso per l'habitat monitoring di Great Duck Island, che comunque si discosta dalle altre per epoca realizzativa e scopi (puramente di ricerca scientifica).
Per quanto riguarda il progetto Lofar, purtroppo ha avuto qualche problema: oltre a quelli architetturali, ci sono state poi anche questioni economiche e di tempo che hanno ostacolato il progetto.
Libelium di fatto non fornisce una soluzione e, quindi, non è valutabile.
Solar Biscuit è un interessante progetto di ricerca, che ha avuto lo scopo di creare nodi *battery-less*, più che creare un vero sistema completo per il monitoraggio ambientale.
Il sistema descritto nella Sezione 3.3.3 per il monitoraggio in serra, oltre a non fornire molti dati utili a valutare il progetto, è caratterizzato da una scelta progettuale molto discutibile, ovvero quella di adoperare gli SMS come mezzo di trasporto delle informazioni dal campo verso il server remoto. Oltre ad essere un sistema costoso, non permette una versatilità di utilizzo, poiché non è una comunicazione bidirezionale e i messaggi sono limitati alla lunghezza massima degli SMS.
In definitiva, la vera soluzione per il monitoraggio ambientale tramite WSN, economica, efficiente, non invasiva, facilmente gestibile, versatile e con interfaccia utente completa non esiste ancora.

CAPITOLO 4

PIATTAFORMA REALIZZATA

4.1 OBIETTIVO

L'obiettivo di questa tesi è stato quello di creare un sistema di monitoraggio ambientale tramite Wireless Sensor Network. Il sistema è stato realizzato in modo da soddisfare i seguenti requisiti:

- **out-of-the-box**: la soluzione deve essere pronta all'uso, non deve richiedere configurazioni particolari o interventi hardware da parte dell'utente;
- **completezza**: la soluzione deve comprendere oltre all'architettura di rete, anche un sistema di gestione, memorizzazione, visualizzazione e analisi dati;
- **facilità di utilizzo**: il sistema deve essere installabile e utilizzabile da un qualsiasi utente con conoscenze basilari di informatica;
- **economicità**: sia i costi dei componenti utilizzati sia i costi operativi del sistema devono essere contenuti;
- **non invasività**: i nodi della WSN devono essere quanto più piccoli possibili e non devono alterare in alcun modo l'ambiente;
- **versatilità**: il sistema non deve essere pensato per un uso specifico in un ambito preciso o in un territorio particolare, ma deve poter adattarsi ad una moltitudine di deployment;
- **espandibilità**: il sistema deve consentire, senza particolari modifiche software, l'aggiunta di nuovi sensori;
- **interfacciabilità**: i dati rilevati e memorizzati devono essere facilmente interfacciabili con altri applicativi software;
- **open source**: il sistema deve essere composto esclusivamente da software open source.

In definitiva, la piattaforma realizzata si è proposta di colmare le lacune ed eliminare i difetti di quelle ad oggi disponibili, che sono state presentate nel capitolo precedente. In particolar modo, si è puntato sull'economicità, aspetto che manca in tutte le soluzioni, e sul fatto di essere un sistema completo out-of-the-box.
Come in qualsiasi progetto, ci sono stati anche dei trade-off da bilanciare. In particolare, il fattore economico ha giocato un ruolo importante sulla scelta dell'hardware, più precisamente sulla scelta dei nodi nodi e dei sensori.

4.2 AMBITO DI UTILIZZO

Il sistema è stato creato con il fine di monitorare un ambiente, tramite la rilevazione di parametri come temperatura, luminosità e umidità dell'aria. Gli ambiti generici di utilizzo del monitoraggio ambientale sono già stati elencati nei capitoli precedenti. In particolare, il sistema realizzato è adatto per contesti come l'agricoltura di precisione e lo studio del territorio per fini di ricerca scientifica.
Dato il basso costo dei sensori al momento adottati, e quindi la loro accuratezza non elevata, si sconsiglia l'utilizzo del sistema in ambiti in cui è richiesta un'elevata precisione della misurazione.
L'architettura del sistema è stata progettata per applicazioni in cui le grandezze da monitorare cambiano lentamente, ovvero con una frequenza non inferiore al minuto. Inoltre, essendo una rete di tipo *time-based*, e non *event-based*, le rilevazioni vengono effettuate periodicamente ad intervalli regolari, e non in base al verificarsi di eventi ambientali. Il sistema quindi potrebbe anche venir adoperato, ad esempio, per la prevenzione degli incendi boschivi, sebbene questo non sia il suo scopo ideale. Infatti, una rete di tipo event-based può essere programmata per reagire in un certo

modo non appena rileva un parametro ambientale fuori norma. In altre parole, se un nodo rileva una temperatura di 150°C invece di 30°C, capisce che è successo qualcosa ed aumenta la frequenza delle misurazioni, inviando quanti più dati possibili alla stazione centrale, cosicché le guardie forestali possano capire meglio cosa stia succedendo. Il sistema realizzato, invece, essendo time-based, non cambia automaticamente la frequenza dei campionamenti in base all'evento, ma è possibile modificarla manualmente. Quindi, se l'intervallo impostato è di 30 minuti, è possibile diminuirlo a 1 minuto, ma bisognerà aspettare il prossimo intervallo perché ciò diventi attivo (quindi aspettare altri 30 minuti nel caso peggiore).

4.3 FASI DEL PROGETTO

Il progetto è stato articolato in tre fasi principali:

1. Sviluppo della WSN.
2. Sviluppo del sistema di gestione della WSN.
3. Testing finale e prove sul campo.

4.3.1 Progettazione della WSN

La progettazione della WSN è consistita nella creazione dei protocolli di rete da implementare nei nodi. Questa fase può essere scomposta nelle seguente sottofasi:

1. Studio delle WSN presenti in letteratura.
2. Studio delle piattaforma hardware adottata.
3. Creazione del meccanismo di duty-cycling.
4. Creazione del protocollo di routing.
5. Creazione dell'algoritmo di sincronizzazione dei nodi.
6. Creazione dell'algoritmo di scheduling.
7. Implementazione delle funzioni di risparmio energetico.
8. Testing.

Le fasi elencate non sono necessariamente in ordine cronologico, anche perché, come già spiegato, l'implementazione dei protocolli di rete richiede uno sviluppo cross-layer, in cui è spesso necessario fondere i livelli dell'architettura OSI (in particolare quello MAC e quello Network).

4.3.2 Progettazione del sistema di gestione della WSN

In questa seconda fase è stato creato il sistema che permette di salvare i dati rilevati in un database, prelevarli da remoto e consultarli tramite browser, nonché gestire e visualizzare, sempre tramite browser, alcuni parametri della WSN. Essenzialmente, è stato implementato un database ed un web server ed è stato sviluppato codice *SQL, HTML, PHP, AJAX, XML e Javascript* atto a questo scopo.
Le sottofasi in cui è scomponibile questa fase sono le seguenti:

1. Documentazione sulle tecnologie database e web server per *RIA (Rich Internet Application).*
2. Implementazione del database.
3. Sviluppo del sistema di memorizzazione dati.
4. Sviluppo del sistema di consultazione dati.
5. Sviluppo del sistema di gestione della WSN in remoto.
6. Testing.

Tutto il codice sviluppato sia per questo sistema sia per la WSN, è disponibile nel CD-ROM allegato.

4.3.3 Testing finale e prove sul campo

Nell'ultima fase, è stato effettuato il testing finale del sistema completo, ovvero di rete WSN e sistema

di consultazione dati e gestione della rete. Dopodiché sono state effettuate delle prove all'aperto per verificare le prestazioni della rete, come perdita di pacchetti ed autonomia.

4.4 SPECIFICA DEI REQUISITI SOFTWARE

Prima dello sviluppo del sistema, è stato redatto il documento della specifica dei requisiti software, conforme allo standard IEEE 830-1998, che è possibile trovare nell'Appendice A.
Nel documento viene descritto un sistema *multi-tier*, costituito dalla WSN, da un gateway, da un database e da un web server. Questi componenti devono interfacciarsi tra loro, come indicato in Figura 4.1, in modo da permettere la memorizzazione dei dati della rete e la loro consultazione in remoto, da parte di un utente, tramite browser. Inoltre questi dati devono essere facilmente interfacciabili con altri software, quindi devono essere forniti in formato XML ed EEML [78], e devono transitare su reti TCP/IP.

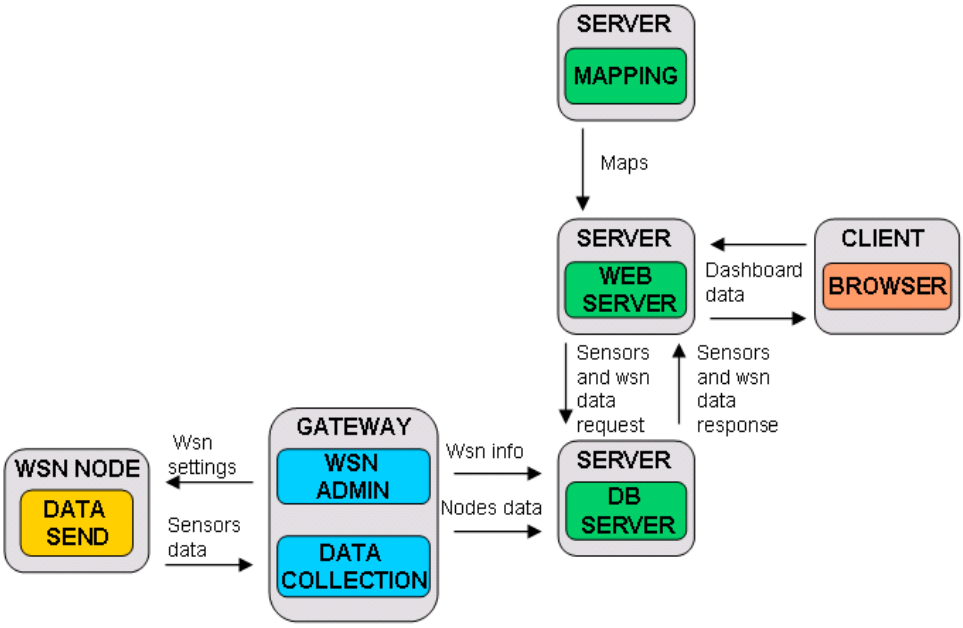


Figura 4.1: Architettura software del sistema

L'utente interagisce col sistema tramite un'applicazione web interattiva, una sorta di *dashboard o cruscotto,* che deve fornire quattro funzionalità principali:

- **Monitoraggio**: i dati provenienti dai sensori devono essere rappresentati su una mappa, su dei grafici e su delle tabelle. I dati devono essere aggiornati in tempo reale, devono essere facilmente interpretabili e l'utente deve essere avvertito visivamente e via mail in caso di valori fuori norma.
- **Analisi dati**: questa funzione permette all'utente di richiedere al sistema i dati scelti secondo certi parametri e visualizzarli su dei grafici o in un foglio di calcolo.
- **Gestione allarmi**: l'utente può definire delle aree di allarme, impostando dei valori soglia delle grandezze misurate, in modo da essere avvertito in caso queste vengano oltrepassate.
- **WSN**: l'utente può controllare lo stato della WSN e impostare il parametro di intervallo di trasmissione dei dati, pari almeno ad un minuto.

Un ultimo requisito importante richiesto, è quello della leggerezza dei componenti software, in modo da consentire la loro installazione ed esecuzione su sistemi a basse prestazioni.
Vincoli come la gestione della multi-utenza e l'implementazione di standard di sicurezza non sono presenti, onde evitare un'eccessiva estensione del progetto.

4.5 ARCHITETTURA MULTI-TIER IMPLEMENTATA

L'architettura finale realizzata, schematizzata in Figura 4.2, è di tipo multi-tier:

- **Mote Tier**: comprende tutte le componenti del nodo della WSN (mote), ovvero il bootloader di Arduino, il protocollo di MAC, il protocollo di routing, l'algoritmo di schedulazione e il meccanismo di risparmio energetico.
- **Gateway Tier**: è l'architettura del sink o gateway, che si occupa di gestire la WSN e salvare i dati nel database; comprende il protocollo di MAC e di routing, gli algoritmi di gestione della WSN e il database.
- **Server Tier**: si occupa di prelevare i dati dal database e renderli disponibili in formato XML, costruire i grafici e le mappe richieste e gestire gli allarmi. È costituito quindi dal web server a da tutte le tecnologie necessarie al suo funzionamento, come AJAX, PHP, HTML, XML, Google Maps.
- **Client Tier**: è costituito dai componenti che consentono di visualizzare e analizzare i dati rilevati dai nodi e lo stato di salute delle rete, impostare allarmi, e modificare alcuni parametri del sistema e della WSN stessa.

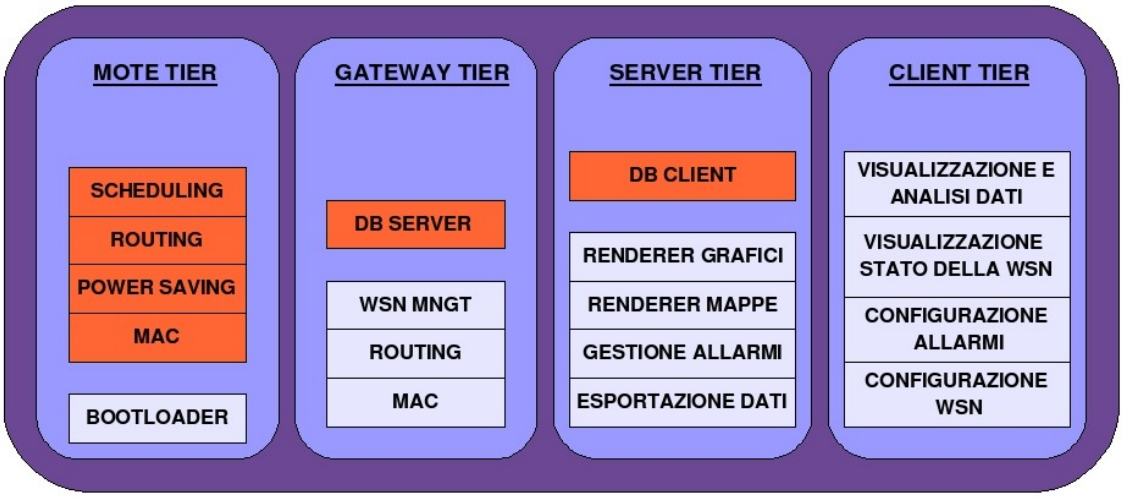


Figura 4.2: Architettura multi-tier del sistema realizzato

4.6 HARDWARE ADOPERATO

L'hardware necessario per il sistema è rappresentato dai nodi, dal gateway e dal web server. Il fattore determinante per la scelta di questi è stato il costo.

4.6.1 Nodi della WSN

Come mote sono stati utilizzati gli *Squidbee* della Libelium (vedasi Figura 4.3). Lo Squidbee è un dispositivo wireless sensor con hardware e sorgenti open source, per questo definito “open mote”. È costituito da una piattaforma di base, Arduino, collegata ad una scheda con modulo radio, detta XBee shield, da tre sensori e da una scatola protettiva. Il costo di un Squidbee è di circa 120 €.

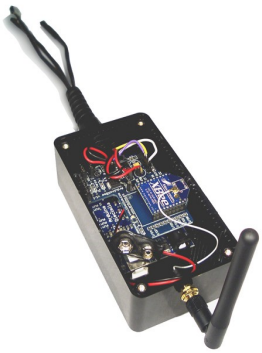


Figura 4.3: Nodo Squidbee

4.6.1.2 Arduino

Arduino è una piattaforma hardware open source per il *physical computing*, il cui linguaggio di programmazione deriva da *Wiring*, un'altra piattaforma per il physical computing.

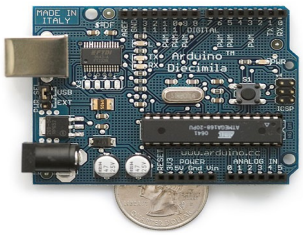


Figura 4.4: Arduino

Esistono più versioni di Arduino: quella usata nello Squidbee si chiama *Arduino Diecimila***,** rappresentata in Figura 4.4. Essa è basata sul microcontrollore Atmel Atmega168, con 16K di memoria flash programmabili (di cui 2K occupati dal bootloader), 1K di SRAM, 512 byte di EEPROM e 16Mhz di clock. La scheda Diecimila è dotata di 14 porte I/O digitali, di cui 6 con PWM (Pulse Width Modulation), e 6 input analogici. È alimentabile con una tensione consigliata compresa tra i 7 e i 12 Volt.
I vantaggi di questa piattaforma rispetto ad altre simili con microcontrollori ATmega sono:

- **il costo**: Arduino Diecimila viene venduto a circa 30€;
- **l'open source hardware**: i modelli dei circuiti sono distribuiti con licenza Creative Commons e possono essere modificati o migliorati senza molte difficoltà;
- **il software open source ed espandibile**: l'ambiente di sviluppo, nonché le librerie fornite, sono open source e facilmente modificabili a proprio piacimento usando il linguaggio C e C++;
- **la community:** dietro Arduino si è creata una vasta comunità di utenti, pronta a fornire supporto e utili librerie.

4.6.1.3 XBee

L'XBee Shield (vedasi Figura 4.5) è una scheda che permette di montare l'XBee su Arduino ed è stata sviluppata da Libelium in collaborazione con il team di Arduino. Il costo dell'XBee Shield, comprensivo di XBee, è di circa 60 €.

Figura 4.5: XBee Shield

XBee è un modulo radio compatibile con l'IEEE 802.15.4 e ZigBee. Ci sono due versioni di XBee: XBee e XBee-PRO, che differiscono nella potenza di trasmissione e nel range massimo. La versione di base (l'XBee) ha una potenza trasmissiva di 1mW e supporta bitrate fino a 250 kbps [104].
Facendo un confronto con un MicaZ mote, dotato di CPU ATMega 128L con 512 K di Flash e radio a 2.4Ghz compatibile 802.15.4/Zigbee, si nota che i consumi dell'Arduino con XBee sono molto superiori: in trasmissione e ricezione il MicaZ è quattro volte più efficiente, mentre in sleep e in idle ci sono addirittura tre ordini di grandezza di differenza, come si può notare dai grafici di Figura 4.6 e Figura 4.7.

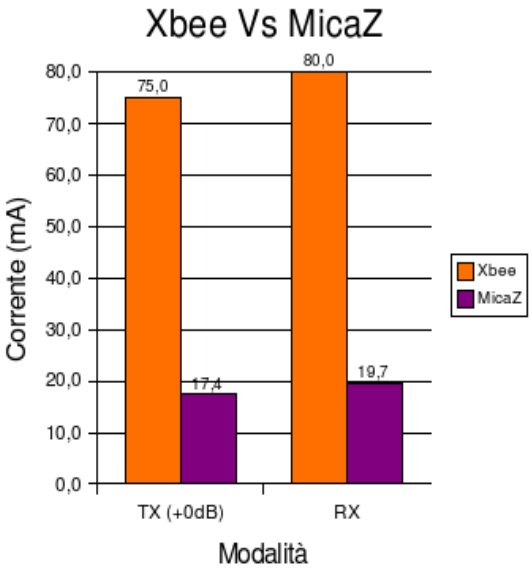


Figura 4.6: Consumo energetico (mA) in trasmissione e ricezione

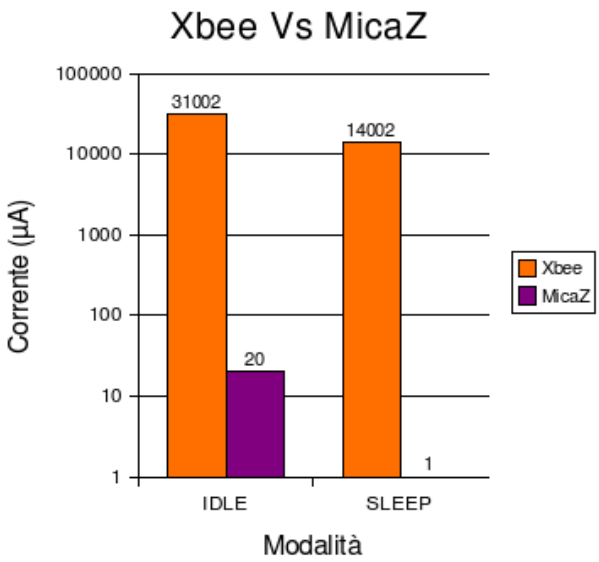


Figura 4.7: Consumo energetico (µA) in idle e in sleep

4.6.1.4 Sensori

I sensori già installati nello Squidbee sono tre: temperatura, umidità e luminosità. Il primo è un *LM35*, sensore a circuito integrato capace di misurare temperature tra i -55°C e +150°C, con accuratezza di 0.5°C (a +25°C) e con consumo massimo di 60 µA. Il sensore di umidità è un *808H5V5*, in grado di rilevare l'umidità relativa dell'aria tra 0 e 100%, con accuratezza minore di ±4% (a 25°C e umidità tra il 30 e l'80%). La corrente tipica assorbita è di 0.38 mA, quella massima è inferiore a 0.5 mA. Infine, il sensore di luminosità è un comune *LDR (Light Dependent Resistor)* da 10K. Ognuno di questi sensori costa meno di 1 Euro.

4.6.1.5 Scatola

La scatola protettiva è fatta di comune plastica, misura 120 x 65 x 40 mm e dovrebbe proteggere

l'hardware dagli agenti atmosferici e dalla polvere. In realtà l'apertura della scatola non sembra bene isolata e non ci sono guarnizioni di alcun tipo. I sensori sono all'esterno e non sono protetti, ma i cavi passano all'interno di piccoli tubi che convogliano in una guaina collegata al corpo principale della scatola.

4.6.2 Gateway

Il gateway è l'insieme di tre apparati distinti: un mini pc, un sink ed eventualmente un router adsl. Quest'ultimo serve solo nel caso in cui il web server sia raggiungibile esclusivamente tramite Internet. Inoltre la funzione di router è utile nel caso in cui ci siano più sottoreti WSN e quindi più sink. Negli altri casi, il router adsl non serve. Per le prove effettuate per questa tesi, non è stato utilizzato, poiché è stata adoperata una *Local Area Network (LAN)*.

4.6.2.1 Sink

Il sink è semplicemente uno Squidbee privo di sensori e privo di microcontrollore (vedasi Figura 4.8). Fondamentale funge solo da modulo radio. Il microcontrollore è assente perché altrimenti non sarebbe possibile comunicare con l'XBee. Quindi il codice per gestire il sink deve essere installato ed eseguito da un pc collegato via USB ad esso (il mini pc serve anche a questo).

Figura 4.8: Il nodo sink

4.6.2.2 Mini-pc

Dato che il gateway, e quindi anche il mini-pc, si trova sul campo da monitorare, ci sono buone possibilità che non sia fornito di un accesso alla rete elettrica. Dunque, in tale situazione, dovrebbe essere alimentato da batterie ricaricate da, ad esempio, pannelli fotovoltaici. Ciò definisce un importante requisito per il pc: il consumo energetico. Un'altra caratteristica desiderabile, è quella dell'ingombro ridotto, in modo da poterlo posizionare ovunque. Infine il costo rimane sempre una determinante. Un attributo non richiesto è invece la potenza di calcolo: infatti sul mini-pc deve girare il programma scritto in C del sink e un database molto leggero, uSQLite (che verrà presentato nel Capitolo 6).
Le soluzioni che ben soddisfano questi requisiti sono i mini-pc. In particolare, è stato scelto il *Koala Nano PC* dell'azienda italiana *KOAN S.A.S.*, riportato in Figura 4.9. Il Nano PC è un sistema dalla dimensioni molto ridotte (115x115x35,4mm), dotato di CPU *Vortex* a 300 Mhz (senza coprocessore matematico), RAM di 128 MB DDR2, scheda video integrata *XGI Volari Z9S*, scheda Ethernet 10/100 integrata, 2 porte USB, 2 porte seriali, 1 slot Compact Flash e 1 porta PS/2. Viene alimentato da un trasformatore a 5V ed è caratterizzato da un consumo massimo di 1080mA a 5V (massimo 6W). Il Nano PC viene venduto a 159 €.

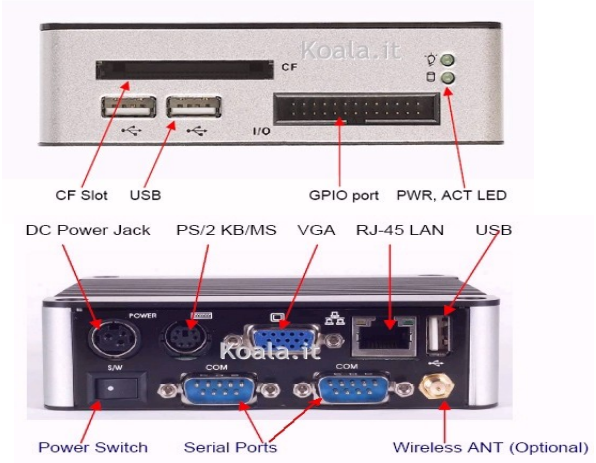


Figura 4.9: Fronte e retro del Nano Pc

4.6.3 Web server

Il web server non necessita di requisiti hardware particolari. Su di esso verrà installato un web server molto leggero (*Lighttpd*), un *Java Toolkit (Dojo)* e *Php*. Quindi, praticamente qualsiasi computer in grado di far girare una distribuzione Linux va bene. Naturalmente, se si prevedono molti accessi contemporanei al server, è meglio dimensionare adeguatamente la macchina.
Nell'ambito di questo progetto di tesi, è stato usato un pc con CPU AMD Athlon 3200+ con distribuzione Ubuntu.

CAPITOLO 5

PROGETTAZIONE DELLA WSN

5.1 APPROCCIO CROSS-LAYER

Solitamente, per la progettazione di un'architettura di rete, si fa riferimento al modello OSI. Per una rete WSN però il discorso cambia. Infatti, sebbene esistano diverse soluzioni costruite seguendo il consueto modello a livelli, esse non risultano le più indicate. La motivazione è che, nonostante l'architettura così definita consenta di raggiungere ottimi risultati prestazionali nei singoli layer, essa non riesce a massimizzare le prestazioni della rete in rapporto all'energia consumata. Quindi, in una WSN, date le scarse risorse energetiche e hardware, è opportuno progettare e ottimizzare in modo congiunto i diversi layer dell'architettura. Quest'ultimo approccio è definito *cross-layer*.
Ci sono diversi articoli in letteratura che dimostrano che le tecniche di design ed integrazione cross-layer consentono miglioramenti significativi in termini di consumo energetico. Tali miglioramenti sono dovuti a tre ragioni: la prima è che le limitate risorse energetiche, computazionali e di memoria dei nodi della rete, richiedono un tale approccio. L'overhead che una architettura a livelli produce, comporta una grande inefficienza. La seconda ragione consiste nel fatto che spesso è necessario considerare le proprietà del modulo radio e del canale wireless nel design dei protocolli.
Infine, le reti di tipo *event-centric*, richiedono dei protocolli di comunicazione *application-awar*e.
Esistono due tipi di approccio cross-layer: uno che segue le linee guida di un'architettura a strati ma tiene conto delle integrazioni cross-layer in ogni livello; l'altro, invece, definisce un modello in cui i vari strati non esistono più e si fondono tra loro indistintamente. Al momento, il primo metodo è quello più usato ed è stato quello seguito in questo progetto di tesi.
In letteratura, le interazioni cross-layer proposte sono diverse. Di seguito vengono riportati alcuni esempi:

- **MAC + PHY:** in [21], la correlazione spaziale di un fenomeno fisico osservato è sfruttata per il controllo di accesso al canale. L'idea è quella di considerare un nodo il rappresentante di altri nodi, in base ad un modello teorico. Il MAC proposto viene definito *Correlation-based Collaborative Medium Access Control (CC-MAC),* e consente alte prestazioni in termini di consumo energetico, rate di pacchetti persi e latenza.
- **MAC + ROUTING**: un esempio di interazione tra MAC e routing si trova in [22], dove viene proposto un algoritmo di routing che seleziona il next-hop in base alla potenza trasmissiva necessaria a raggiungere i vari nodi confinanti.
- **ROUTING + PHY**: in [23], viene presentato un metodo di ottimizzazione di throughput della rete. Gli autori mettono in rilievo il fatto che il throughput è collegato alla capacità dei singoli link e quindi al livello della potenza del segnale radio ricevuto da ogni nodo. Inoltre, l'allocazione della potenza dipende dalle interferenze e dal data rate del link. Quindi, viene proposta una soluzione CDMA/OFDM in cui il controllo della potenza trasmissiva e il routing vengono eseguiti in maniera distribuita.
- **TRASPORTO + PHY**: un'interazione tra il livello fisico e quello di trasporto per il controllo della potenza trasmissiva e per il controllo di congestione viene illustrato in [24]. Gli autori propongono un sistema CDMA per la regolazione di queste due variabili.
- **INTERAZIONE A 3 LIVELLI:** oltre agli approcci che accoppiano due livelli, esistono anche quelli che ne congiungono tre. Un esempio di questi lo si trova in [25], dove viene modellata una rete di tipo TDMA (Time Division Multiple Access) con le funzionalità di ottimizzazione di potenza trasmissiva, rate di trasmissione e scheduling di accesso al canale.

Come si può dedurre dagli esempi appena citati, il principio di progettazione cross-layer cerca di rendere possibile un'allocazione uniforme delle risorse tra tutta la rete, in modo da massimizzare il lifetime, la capacità e il risparmio energetico della rete intera.
Si tratta insomma di capire le interdipendenze tra i vari livelli. Solo così si possono progettare sistemi dove, per esempio, il rate di trasmissione dipende dalle interferenze che agiscono sul link e queste interferenze dipendono dall'algoritmo di controllo della potenza di trasmissione. O ancora, situazioni in cui le scelte di routing vengono determinate dalle performance dei link, a loro volta dipendenti dallo scheduling di trasmissione, regolato dal MAC.
Al di là di questi vantaggi, l'approccio cross-layer presenta purtroppo anche alcuni aspetti negativi, illustrati in [26], che impattano sulle seguenti caratteristiche del sistema:

- **Modularità**: nell'approccio classico a livelli, i componenti dell'architettura possono essere suddivisi in moduli isolati l'uno dall'altro, consentendo così facilitazioni nella fase di sviluppo e di risoluzione di problemi. Nell'approccio cross-layer, questi moduli non esistono più, poiché l'architettura tende ad essere unificata e di conseguenza non è più possibile lavorare su uno strato senza considerare gli altri.
- **Miglioramenti del sistema**: per i motivi spiegati sopra, cambiare un componente in un sistema cross-layer può facilmente comportare problemi ad altre parti del sistema stesso. Per questo motivo gli autori di [26] consigliano di creare, al posto dei moduli, delle *entità* delle funzioni, che aiutano anche ad evitare la creazioni di funzionalità duplicate (cosa che succede spesso nelle architetture a livelli).
- **Instabilità**: una singola scelta progettuale può avere un impatto negativo sulla stabilità del sistema intero. Perciò, è necessario prestare la massima attenzione e sviluppare tecniche di controllo della stabilità del sistema.

5.2 FUNZIONALITÀ DELL'ARCHITETTURA

Le principali funzionalità che compongono l'architettura di una WSN sono le seguenti:

- trasmissione a livello fisico
- medium access control
- topology control
- duty-cycling
- routing
- power management
- data aggregation

Per raggiungere gli obiettivi di risparmio di risorse energetiche e hardware è necessario tenere presente le interdipendenze reciproche di questi componenti. Per capire questo concetto, si può partire dalla definizione del duty-cycling.

5.2.1 Duty-Cycling

Il *duty-cycling* è la tecnica più efficiente di risparmio energetico, e consiste nel spegnere il nodo quando non serve e accenderlo quando serve. *Duty cycle* è l'intervallo in cui una stazione è attiva. L'attivazione o lo spegnimento del nodo è determinato dal topology control o dal power management o da entrambi. In Figura 5.1 è rappresentato uno schema riassuntivo delle soluzioni per il duty-cycling, proposto in [77].

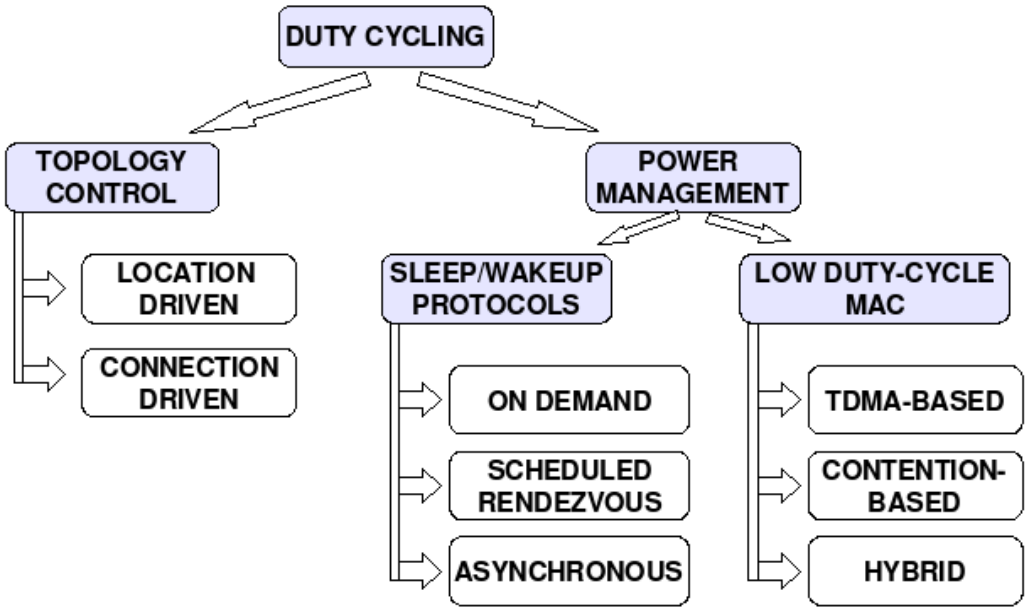


Figura 5.1: Schema riassuntivo delle soluzioni per il duty-cycling

5.2.2 Topology control

Il *topology control* è definito da protocolli che mirano a sfruttare la ridondanza nella rete. Con il termine ridondanza si intende che un nodo è posizionato molto vicino ad altri nodi e, quindi, può essere sostituito senza problemi da uno di essi, poiché le rilevazioni riportate sarebbero molto simili, se non identiche.
I problemi di un protocollo di questo tipo sono: quanti sensori attivare, quali e quando.
Per quanto riguarda la prima domanda, bisogna trovare il giusto trade-off tra numero nodi attivi e potenza trasmissiva richiesta o qualità del link. Infatti, se ci sono pochi nodi accesi e sono molto distanti tra loro, serve una potenza trasmissiva maggiore e il link potrebbe essere più instabile.
Per rispondere alla seconda domanda, sono state proposte varie soluzione riassumibili in due classi:

- **Location driven**: la decisione di attivazione dei nodi dipende dalla loro posizione.
- **Connectivity driven**: i nodi sono attivati dinamicamente, in modo da garantire la connessione o la copertura dell'area monitorata.

Il topology control può migliorare il lifetime della rete di un fattore 2 o 3 rispetto alle reti in cui i nodi sono sempre attivi [99], ma è possibile incrementarlo se oltre a questa tecnica si implementa anche il power management.

5.2.3 Power Management

Il *power management* è una tecnica di risparmio energetico che può essere implementata o in un livello sopra al MAC (di solito Network o Application) o nel MAC stesso. I protocolli appartenenti alla prima categoria si possono definire protocolli di *Sleep/Wakeup* indipendenti e hanno il vantaggio di essere indipendenti dal MAC, mentre quelli della seconda sono protocolli *low duty-cycle MAC* e hanno il vantaggio di ottimizzare l'accesso al canale in base allo scheduling di Sleep e Wakeup.
I protocolli di Sleep/Wakeup indipendenti possono allora volta essere suddivisi in tre sottoclassi, in base all'approccio usato per decidere quando attivare/disattivare i nodi:

- **On-Demand:** è la forma più efficiente di Sleep/Wakeup, poiché un nodo viene svegliato solo quando c'è un altro nodo che deve comunicare con lui. Questo approccio, oltre a permettere un risparmio energetico elevato, consente anche una bassissima latenza della rete.
  Il problema principale è però come svegliare il nodo nell'istante giusto. Al momento, l'unica soluzione è quella di utilizzare due moduli radio: uno, detto data radio, usato per la

trasmissione e ricezione dei dati; l'altro, detto wakeup radio, è usato per svegliare il nodo. Quando un nodo A deve spedire un dato al nodo B dormiente, manda un impulso tramite la wakeup radio a B per svegliarlo. Così B accende la data radio e A può iniziare a trasmettere.

- **Scheduled Rendezvous:** l'idea di questo approccio è che un nodo si sveglia quando anche i suoi vicini si svegliano. Solitamente i nodi si accendono in base ad un wakeup schedule e rimangono attivi per un tempo predeterminato breve, per comunicare con i vicini. Poi tornano a dormire fino al prossimo rendezvous time. Lo svantaggio di questa tecnica è che i nodi devono essere sincronizzati tra loro e la latenza della rete è alta. Mantenere i nodi sincronizzati significa però avere ulteriore overhead, ovvero messaggi di sincronizzazione, che costano energia.
- **Asincroni:** in questo approccio, i nodi seguono un wakeup schedule senza essere sincronizzati tra loro. Quindi, ogni nodo si accende e si spegne secondo la sua schedulazione e se deve comunicare, lo fa con i nodi attivi in quel momento. Questa tecnica è più facile da implementare delle altre, ma ha il difetto che costa più energia, dato che i nodi devono svegliarsi più spesso rispetto agli altri approcci, per evitare che ci siano momenti in cui solo un nodo è attivo.

Per quanto riguarda i low duty cycle MAC, esistono tre categorie, già accennate nel Capitolo 2:

- **Time Division Multiple Access (TDMA)**: ad ogni nodo viene assegnato uno slot di tempo durante il quale può trasmettere o ricevere. Nel tempo restante, rimane spento. Questa soluzione ha molti difetti, tra cui una scarsa flessibilità, una limitata scalabilità, la necessità di un'elevata sincronizzazione, l'alta probabilità di avere nodi vicini che interferiscono tra loro e le scarse prestazioni in condizioni di traffico leggero. Per questi motivi, non vengono implementati solitamente MAC basati solo sul TDMA.
- **Contention-based**: è la categoria di MAC più diffusa. Implementano il duty-cycling tramite un'integrazione tra funzionalità di accesso al canale e scheduling Sleep/wakeup.
- **Ibridi:** cercano di combinare i punti di forza ed eliminare i difetti dei TDMA e dei Contention-based. I MAC di questo tipo si comportano come dei Contention-based quando il livello di contesa del canale è basso, mentre diventano dei TDMA quando il livello di contesa è alto.

5.2.4 Routing

I protocolli di routing si possono dividere in due categorie [77], schematizzate in Figura 5.2: la prima è formata dai protocolli che sfruttano la struttura della rete mentre la seconda è composta da quelli che si basano su politiche di QoS o sui flussi della rete.

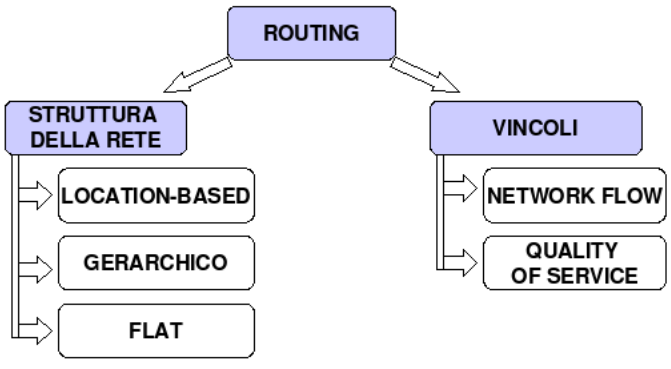


Figura 5.2: Le categorie di protocolli di routing per WSN

La prima permette soluzioni energy-aware più efficienti e può essere scomposta in tre sottocategorie:

- **Location-based routing**: il protocollo di routing sfrutta le informazioni di posizione o vicinanza dei nodi.
- **Routing gerarchico (o clustering routing)**: viene imposta una struttura alla rete, cioè

alcuni nodi assumono un ruolo particolare nel processo di comunicazione. All'interno di un gruppo di nodi, detto cluster**,** un nodo viene nominato *cluster-head* e si occupa di coordinare le attività del cluster. Oltre a questi cluster-head, ci sono i nodi ordinari, che comunicano solo con il loro cluster-head, e gateway, che sono il punto di riferimento dei cluster-head. Quando un nodo deve spedire un pacchetto ad un altro nodo, lo manda al cluster-head il quale lo inoltrerà direttamente al destinatario, se questo si trova all'interno del cluster oppure al gateway, se il destinatario appartiene ad un altro cluster. Solo i gateway e i cluster-head partecipano alla propagazione di messaggi di controllo e routing. Di conseguenza, l'overhead di routing viene ridotto significativamente e reti di grande dimensione non soffrono di problemi di scalabilità. Inoltre il clustering può essere sfruttato per operazioni di data-aggregation e di power management.

- **Flat routing (o Data-centric routing)**: in questo approccio, tutti i nodi sono considerati sullo stesso piano per quanto riguarda il processing e il trasporto dei dati. In questa categoria viene seguito il modello di comunicazione data-centric, in cui i dati sono più importanti dei singoli nodi. Quindi, routing e forwarding richiedono una forma di *data-centric data dissemination.* In questo caso, le informazioni sono identificate da attributi riguardanti il fenomeno. Per esempio "Richiedi la temperatura della regione X" deve essere disseminata ai nodi della regione X. Viceversa, i dati dalla regione X devono tornare al punto dove è stata lanciata la query. A tal fine si possono usare tecniche come flooding e gossiping, ma non sono le tecniche migliori a causa della ridondanza di informazione che creano (con il conseguente spreco energetico). Sono stati quindi proposti diversi algoritmi application-aware che si occupano di disseminare l'informazione nella rete. Questi si basano sul modello *publish/subscribe*: i nodi pubblicano i dati disponibili che poi vengono consegnati solo ai quei nodi che li hanno richiesti. Inoltre, questi meccanismi consentono risparmi energetici grazie all'in-network processing che viene effettuato per il data aggregation.

La seconda categoria di protocolli di routing è scomponibile in schemi basati su:

- **Network Flow**: l'algoritmo di routing prende in considerazione valori come il throughput e il traffico presente nei vari link.
- **Quality of Service:** ai singoli pacchetti viene assegnata una priorità che permette loro di transitare su determinati path piuttosto che altri.

5.2.5 Data aggregation

Il Data aggregation è una tecnica usata soprattutto nelle reti con routing di tipo data-centric, in cui i nodi intermedi possono ricevere dati dello stesso tipo dai nodi figli. Considerando il fatto che le WSN sono spesso progettate seguendo un modello di *reverse multicast tree* (vedasi Figura 5.3), questa funzione risulta molto conveniente da applicare [100].
Il data aggregation si occupa di aggregare i dati provenienti da più nodi che si riferiscono allo stesso attributo.

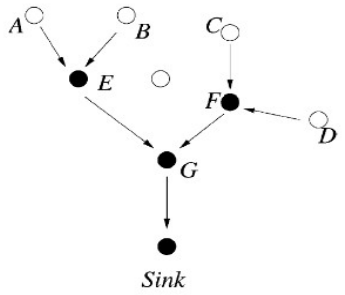


Figura 5.3: Data aggregation in una struttura reverse multicast tree: i dati vengono aggregati a livello dei nodi intermedi, colorati in nero

Quindi, lo scopo di questa tecnica è risparmiare traffico di rete ed energia dei nodi, inoltrando solo certi dati. In tal caso, si parla anche di *data fusion***.** Per fare un esempio, un nodo intermedio riceve i dati di temperatura dai suoi figli, appartenenti tutti alla stessa area. Il nodo può a questo punto calcolare e inoltrare la media dei dati di temperatura, in modo da avere un valore unico e significativo per quell'area. Il data aggregation è la forma più comune di in-network processing.

5.3 SOLUZIONI ESISTENTI

Per la progettazione dell'architettura del sistema, si è dapprima svolta una ricerca sulle varie funzionalità (MAC, routing, scheduling, power saving, topology control,...) presentate in lettura e poi, sfruttando le conoscenze apprese, si è creata un'architettura adatta allo scopo di questa tesi.
Di seguito vengono prima presentate le soluzioni proposte in letteratura e poi quelle implementate per il progetto di questa tesi.
In questo paragrafo vengono illustrate le funzionalità esistenti di un sistema WSN. Ad eccezione della trasmissione dei bit a livello fisico, che è definita da uno standard, le funzionalità proposte dai ricercatori, all'interno di uno stesso ambito, come il routing o il medium access control, sono numerose e caratterizzate ognuna da pregi e difetti, che sono serviti a capire quali scelte effettuare per l'architettura finale del sistema.

5.3.1 Trasmissione dei bit a livello fisico

La trasmissione dei bit a livello fisico dipende ovviamente dall'hardware a disposizione. Dato che i nodi Squidbee utilizzati in questo progetto di tesi sono dotati di modulo radio XBee, la modulazione radio avviene secondo lo standard IEEE 802.15.4 [76]. Perciò non verranno considerati altri standard.
Questo standard, rilasciato nel 2003, adotta una tecnica di *Direct Sequence Spread Spectrum (DSSS)*. Le trasmissioni DSSS moltiplicano i dati da trasmettere per un segnale di "rumore". Questo segnale di rumore è una sequenza pseudocasuale di 1 e -1, ad una frequenza molto maggiore di quella del segnale originale, distribuendo così l'energia di quest'ultimo su una banda molto più larga.
Il segnale derivante sembra rumore bianco. Tuttavia, questo segnale di "rumore" viene adoperato dal ricevente per ricostruire il segnale originale, attraverso la moltiplicazione per la stessa sequenza pseudocasuale di 1 e -1. Questo processo è conosciuto con il nome di *despreading*. Affinché il despreading funzioni correttamente, le sequenze di trasmissione e ricezione devono essere sincronizzate. Ciò richiede che il ricevitore sincronizzi la sua sequenza con quella del trasmettitore attraverso un processo di ricerca del timing. Un beneficio derivante dalla sincronizzazione è la possibilità di usare il timing relativo per determinare la posizione del ricevitore se si conosce quella del trasmettitore. L'effetto causato del despreading è un miglioramento del rapporto *SNR (Signal to Noise Ratio)* ed è definito *process gain.*
Se un terzo trasmettitore invia una sequenza pseudocasuale diversa (o non la trasmette) sullo stesso canale, il despreading non determina alcun process gain per quel segnale. Per questo motivo il DSSS implementa il *Code Division Multiple Access (CDMA)*, che permette a diversi trasmettitori di utilizzare lo stesso canale, all'interno dei limiti delle proprietà di correlazione tra le sequenze pseudocasuali.
Ritornando allo standard 802.15.4, le bande definite sono tre: 868 Mhz per l'Europa, 915 Mhz per gli Stati Uniti e 2.4 Ghz in tutto il mondo. Per consentire la coesistenza di più reti nella stessa area, è stato adottato un approccio *Frequency Division Multiplexing*, dividendo le bande disponibili in canali (1 canale per la banda 868 Mhz, 10 per la 915 Mhz con *channel spacing* di 2 Mhz, 16 per la 2.4Ghz con channel spacing di 5 Mhz). I canali delle prime due bande sono stati definiti per applicazioni a basso bit rate. Quest'ultimo è pari a 20 kbps per la banda 868 Mhz e 40 kbps per 915 Mhz. L'ampiezza dei canali della banda 2.4 Ghz consente invece bit rate maggiori, fino a 250 kbps. Nel 2006 però sono stati introdotti nuovi schemi di modulazione che permettono di raggiungere un bit

rate di 250 kbps anche nelle prime due bande.
Non molto tempo dopo il rilascio dell'IEEE 802.15.4, ci si accorse che numerose applicazioni avrebbero potuto trarre vantaggio dalla possibilità di poter misurare, con grande accuratezza, la distanza tra i vari nodi della rete. Tale possibilità era preclusa all'802.15.4, data l'ampiezza di banda limitata. Per questo motivo fu creato un nuovo *Task Group*, l'*802.15.4a*, con lo scopo di definire un nuovo livello fisico. Furono così adottate due nuove tecnologie: l'*Ultra Wide Band (UWB)* e il *Chirp Signal*. L'UWB implementato usa tre frequenze: la prima sotto il Ghz, la seconda nel range 3-5Ghz, la terza nel range 6-10 Ghz, divisi in diversi canali. Non è necessario che una radio supporti tutti i canali, ma solo quelli definiti come obbligatori. Questi canali hanno una banda di 500 Mhz e consentono un *ranging* (misurazione delle distanza) con l'accuratezza di 1 metro. Alcuni canali offrono ancora maggiore banda e quindi accuratezza. Nell'802.15.4a, due reti possono inoltre condividere lo stesso canale, grazie all'utilizzo di *preamble* caratterizzati da una bassa cross-correlazione. Il livello fisico dell'UWB adotta un approccio *Impulse Radio*, in cui sono trasmessi impulsi corti con una larghezza di banda pari a quella del canale. In base alla tecnica di modulazione adoperata, l'UWB consente bit rate tra i 0.1Mbps e 26 Mbps. L'altra soluzione che adotta la tecnica del Chirp Signal, ha il vantaggio di operare sulla banda dei 2.4 Ghz, disponibile in tutto il mondo. Lo standard definisce 14 canali distanziati 5 Mhz l'uno dall'altro. In questo caso, il bit rate non supera 1Mbps e non viene supportato il ranging, anche se sono state presentate alcune soluzioni a proposito.
Le funzioni del livello fisico dell'802.15.4 sono le seguenti:

- **Attivazione/Spegnimento** del modulo radio.
- **Energy Detection (ED)**: stima la potenza del segnale ricevuto (utilizzato per la selezione del canale).
- **Link Quality Indication (LDI):** caratterizzazione del pacchetto ricevuto in base alla potenza e alla qualità del segnale.
- **Selezione del canale.**
- **Clear Channel Assessment**: rilevazione dell'energia del canale per determinare se è già occupato da una stazione che sta trasmettendo.
- **Trasmissione e ricezione** dei bit a livello fisico.

Il *protocol data unit,* schematizzato in Figura 5.4, è costituito dai seguenti campi:

- **SHR**: permette al ricevente di sincronizzarsi con il bit stream.
- **PHR**: contiene informazioni sulla lunghezza del frame.
- **Payload** di lunghezza variabile che contiene il MAC frame.

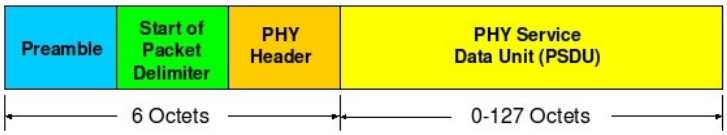


Figura 5.4: Protocol Data Unit del livello fisico dell'802.15.4

5.3.2 Protocolli di topology control

5.3.2.1 GAF

GAF (Geographical Adaptive Fidelity) [27] è un protocollo location-driven. Si basa sulla conoscenza della posizione dei nodi, che può essere fornita tramite GPS o altri sistemi. L'area monitorata viene divisa in piccole griglie virtuali (come in Figura 5.5). Ognuna di queste griglie è definita in modo che, per ogni coppia di griglie adiacenti A e B, tutti i nodi di A riescono a comunicare con B e viceversa. All'interno di una stessa griglia, il routing è uguale per tutti i nodi e solo un nodo alla volta viene attivato. Quindi, i nodi devono coordinarsi tra loro per decidere chi deve rimanere acceso e per quanto tempo.

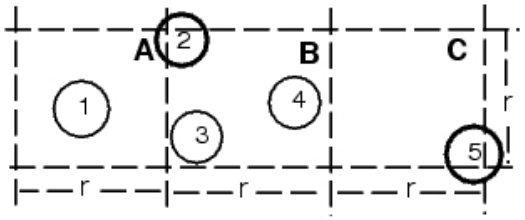


Figura 5.5: La divisione in griglie virtuali in GAF

Le fasi che ogni nodo attraversa sono *sleeping*, *discovery* e *active,* rappresentate in Figura 5.6.

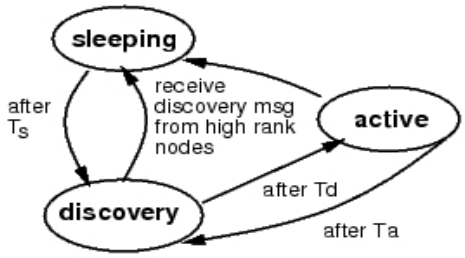


Figura 5.6: Le fasi attraversate da un nodo in GAF

All'inizio il nodo si trova nella fase discovery e scambia messaggi di discovery con altri nodi. Questi messaggi vengono spediti alla scadenza di un timer, dopodiché si passa alla fase active. A questo punto, periodicamente continua a trasmettere in broadcast il suo messaggio di discovery. Se viene rilevato che un altro nodo si occupa del routing, allora entra nella fase sleep. Da questa fase si sveglierà periodicamente per entrare nello stato di discovery. Per bilanciare il carico del routing tra i vari nodi, viene utilizzato un algoritmo basato su un punteggio per decidere di volta in volta chi sarà il nodo *leader* che si occuperà di effettuare il routing all'interno della griglia. Il punteggio viene determinato da alcune regole: un nodo nella fase attiva ottiene un punteggio maggiore di uno in fase discovery e i nodi che hanno una vita residua stimata maggiore ricevono più punti. In questo modo, è possibile cambiare l'intervallo di un nodo nella fase attiva, per farlo tornare nella discovery, facendogli così risparmiare energia. Inoltre, non appena tal nodo ritorna attivo, avrà meno possibilità di essere eletto di nuovo.
Questo protocollo di topology control è indipendente dal protocollo di routing e può quindi essere implementato insieme a qualsiasi algoritmo di routing.
I difetti di GAF consistono nel fatto che, per effettuare il routing, tutti i nodi all'interno di una griglia sono intercambiabili tra loro, e ciò potrebbe portate ad un sottoutilizzo della copertura radio a disposizione. Inoltre, sapere la posizione di ogni nodo può risultare costoso. Questi svantaggi non si presentano nei protocolli di tipo connection-driven, dato che la decisione di dormire o rimanere attivo viene presa in base alle informazione riguardanti le connessione presenti nella rete.

5.3.2.2 Span

Span [101] è un protocollo connection-driven che ha la caratteristica di eleggere dei coordinatori di tutti i nodi nella rete. I coordinatori stanno sempre svegli e si occupano del routing multi-hop, mentre gli altri nodi dormono e si svegliano ogni tanto solo per controllare se è necessario diventare un coordinatore.
Gli obiettivi di Span sono quattro: avere sempre almeno un coordinatore per qualsiasi nodo; raggiungere un efficace *load balancing* facendo ruotare il ruolo di coordinatore tra i vari nodi; cercare di minimizzare il numero di coordinatori per massimizzare il network lifetime, senza però che ciò comporti un degrado in termini di latenza o capacità della rete; scegliere i coordinatori in base a informazioni disponibili localmente.

Per eleggere un numero sufficiente di coordinatori viene applicata la seguente regola: se due nodi vicini ad un nodo non coordinatore non sono raggiungibili tra loro, né direttamente né tramite uno o più coordinatori, quel nodo deve diventare un coordinatore. Potrebbe però succedere che più nodi scoprano contemporaneamente la mancanza di connessione e tentino di diventare dei coordinatori simultaneamente. Per evitare ciò, il nodo aspetta un delay casuale di backoff e si annuncia agli altri candidati. Se alla fine di questo ritardo, il nodo non ha ricevuto alcun messaggio dagli altri canditati coordinatori, allora diventa un coordinatore. Il tempo di backoff viene calcolato in base al numero di nodi che verrebbero connessi tra loro se esso diventasse un coordinatore e la sua energia residua.
Ogni coordinatore periodicamente controlla se può cessare di essere un coordinatore, ad esempio perché i nodi che connette possono comunicare direttamente tra loro o tramite un altro coordinatore.
Nel momento in questa condizione sussiste, manda un messaggio annunciando il suo ritiro e rimane in servizio ancora un istante per evitare che i nodi possono rimanere scoperti.
Un difetto di Span è che necessita delle modifiche nel processo di lookup del routing, dato che dipende strettamente da esso.

5.3.2.3 ASCENT

ASCENT (Adaptive Self-configuring sEnsor Network Topologies) [29] è un protocollo connection-driven che non dipende da informazioni di routing. In ASCENT i nodi decidono se dormire o rimanere attivi in base a informazioni, ricavate in locale, come il tasso di perdita di pacchetti e la connettività.
L'idea di ASCENT è che, all'inizio, solo alcuni nodi sono nello stato *active*, mentre gli altri sono in *passive* (ricevono pacchetti ma non trasmettono). Se il numero di nodi intermedi non è sufficiente, il sink registra un elevato tasso di perdita pacchetti. A questo punto, il sink invia dei messaggi di *help* ai nodi in fase passive, per chiedere loro di svegliarsi e passare in fase active. Non appena questi nodi sono connessi alla rete, mandano dei messaggi *neighbor announcement.* Questo processo continua finché il tasso di perdita pacchetti registrato dal sink non scende sotto una soglia specificata a livello applicativo.
Gli stati di un nodo in ASCENT, sono quattro (vedasi Figura 5.7): *test, sleep, active, passive*.

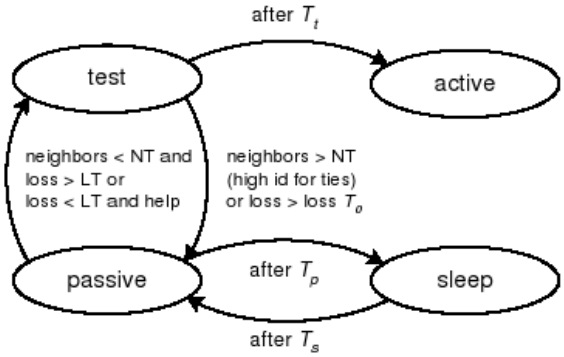


Figura 5.7: Diagramma degli stati in ASCENT

In fase di avvio, il nodo si trova nella fase test, momento in cui il nodo scambia messaggi di controllo routing e dati con gli altri nodi, per capire se è necessario un altro nodo intermedio. Dopo un intervallo in cui manda messaggi di *active neighbor announcement*, passa alla fase active e vi rimane per un certo intervallo. Se però ci sono già nodi vicini attivi oppure il tasso di perdita pacchetti è peggiorato dopo che lui è entrato nella rete, passa subito alla fase passive. In questa fase, vi rimane un tempo preciso, per poi passare nello stato sleep. Se tuttavia il numero di nodi attivi è sotto una certa soglia e il tasso di perdita di pacchetti è oltre un certo limite, oppure il tasso di perdita pacchetti è minore di una certa soglia ma riceve messaggi di help, allora il nodo passa alla fase test. Nella fase passive, il nodo raccoglie solo informazioni sullo stato della rete. Dalla fase sleep, esce una volta scaduto il timer ed entra nella fase passive. Nello stato attivo i nodi vi rimangono, per inoltrare dati e messaggi di routing, finché non terminano l'energia. In definitiva, il protocollo ASCENT ha buone proprietà di scalabilità, ma il risparmio energetico non dipende dalla densità dei nodi.

5.3.3 Protocolli di Sleep/Wakeup

5.3.3.1 On demand

L'idea alla base di questi protocolli è di svegliare il nodo solo quando deve ricevere o trasmettere pacchetti. Grazie alla bassa latenza derivante da questo approccio, i protocolli on demand sono molto indicati nelle reti di tipo event-driven. In questi casi, i nodi mantengono la radio spenta, continuando però il sensing dell'ambiente e, solo quando rilevano un evento, accendono la radio per comunicarlo.
Per poter svegliare il nodo in qualsiasi momento, viene mandato un messaggio di sveglia sul canale radio. Per far ciò servono due radio, una usata per il trasporto dati, l'altra per i messaggi di sveglia, oppure è possibile usare la stessa radio ma su canali diversi. Tuttavia, potrebbe verificarsi la situazione in cui un nodo sta già usando la radio per trasmettere un pacchetto e non può quindi ricevere dati su un altro canale.
Gli approcci solitamente usati sono due: usare due radio della stessa potenza [30, 31] oppure usare la radio a bassa potenza per i messaggi di sveglia [32, 33, 34]. Nel primo caso serve però applicare uno scheduling a entrambe le radio mentre nel secondo, la radio a bassa potenza è sempre accesa, tuttavia il range d'azione è molto basso.

STEM
STEM (Sparse Technology and Energy Management) [30] usa due radio della stessa potenza. Non viene adoperata una radio wakeup a bassa potenza per evitare il problema di range di trasmissioni diversi. Perciò, viene applicato uno schema duty-cycle ad entrambe.
STEM è un protocollo abbastanza vecchio e superato da altri protocolli sotto molti aspetti, perciò non verrà trattato nei dettagli.

PTW
PTW (Pipeline Tone Wakeup) [31] usa due radio come STEM. L'idea è che i nodi periodicamente accendono la radio e se un nodo A deve comunicare un messaggio dopo aver rilevato un certo evento, manda un tono di wakeup (che viene ricevuto da tutti i suoi vicini). Dopo l'invio del wakeup, chi ha ricevuto questo impulso accende la radio per la ricezione dei dati e A spedisce un messaggio in cui informa chi è il destinatario delle comunicazione che deve avvenire. In questo modo, gli altri nodi che sono stati svegliati ma non sono il destinatario, tornano a dormire. Il destinatario vero (nodo B) risponde con un *wakeup acknowledge.* A questo punto A inizia la trasmissione dei dati. Se però B è solo un nodo intermedio e il destinatario finale è il nodo C, allora, non appena A inizia la trasmissione dei dati, B manda un wakeup ai suoi vicini, così da ripetere il processo appena descritto.
L'accesso al canale rimane in ogni caso regolato dal MAC.

Picoradio
Picoradio [32] è stato il primo sistema con radio *low power listening* ad essere proposto. Il principio di funzionamento è lo stesso di STEM, con la differenza che la wakeup radio è sempre in stand-by, ovvero è sempre in ascolto di impulsi di sveglia. Il consumo del circuito della radio wakeup è minore di 10 mA.

Radio Triggered Wake-Up
Un problema dei sistemi con wakeup radio è il continuo consumo di energia (seppur di bassa entità).
Una delle soluzioni proposte è stata presentata in [33], dove viene illustrato un circuito che è in grado di sfruttare l'energia contenuta nell'impulso radio di wakeup per svegliare un nodo dormiente. In questo modo si raggiunge un'altissima efficienza energetica. Tuttavia, il difetto di questa soluzione è che i nodi devono essere molto vicini tra loro, circa 3 metri. Con un circuito un poco più complesso e una latenza di poco superiore è possibile arrivare a 30 metri circa.

RfidImpulse

Un'alternativa a Radio Triggered Wake-Up viene fornita da *RfidImpulse* [34, 35], che ha sempre lo scopo di svegliare il nodo usando solo l'energia dell'impulso radio, ma l'hardware utilizzato è diverso. Infatti RfidImpulse impiega la tecnologia *RFID (Radio-frequency Identification)*, sia per trasmettere che ricevere l'impulso radio, come schematizzato in Figura 5.8.

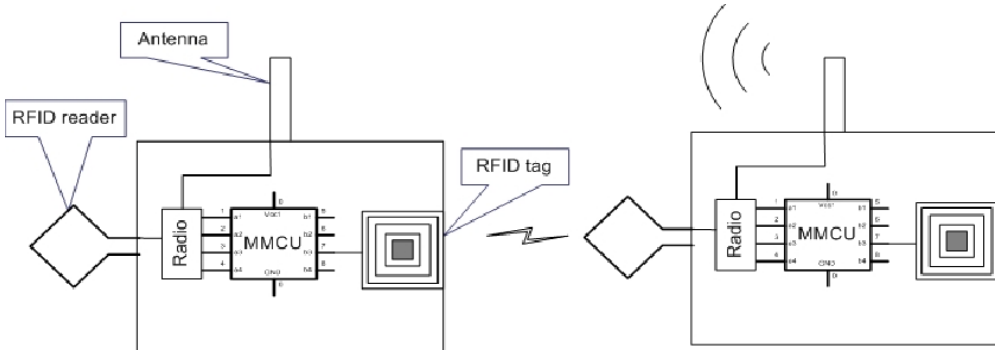


Figura 5.8: Schema hardware dei nodi in RfidImpulse

Ogni nodo viene dotato di un *RFID tag* ed un *RFID reader*. Quando deve trasmettere, manda un impulso tramite l'RFID reader. Tale impulso alimenta l'RFID tag, che genera un interrupt in grado di svegliare il nodo. A questo punto il nodo destinatario, accende la radio principale per ricevere i pacchetti (vedasi Figura 5.9).

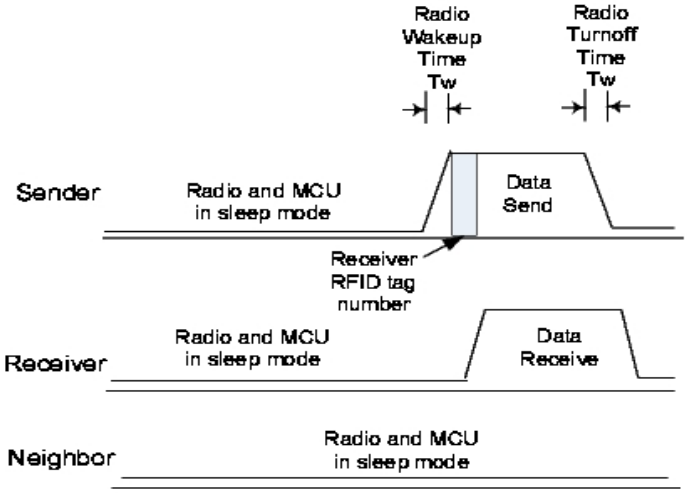


Figura 5.9: Schema temporale dell'invio di un pacchetto in RfidImpulse

Un altro vantaggio derivante dall'utilizzo della tecnologia RFID è che l'impulso mandato dall'RFID reader è diretto solo al destinatario e non a tutti i nodi vicini. Inoltre è possibile conoscere l'ID dei nodi vicini.

Gli svantaggi invece si riassumono nel raggio di copertura e nel costo. Infatti, usando RFID tag passivi (non alimentati), il massimo raggio d'azione è di 10 metri, che sale a 75 metri se si usano tag attivi (alimentati). L'altro difetto è il costo, soprattutto degli RFID reader, che al momento non è ancora molto basso.

5.3.3.1 Scheduled rendezvous

Questi schemi richiedono che tutti i nodi si sveglino contemporaneamente. Tipicamente i nodi si attivano periodicamente per controllare se ci sono comunicazioni in corso. Poi tornano a dormire fino al prossimo momento prestabilito. Il vantaggio di questo approccio è che tutti i nodi sono attivi contemporaneamente, mentre lo svantaggio è che serve una schedulazione.

Fully Synchronized Pattern
Il *Fully Synchronized Pattern* [37] è lo schema più semplice: tutti i nodi si svegliano secondo un pattern, rimangono attivi per un tempo determinato, e tornano a dormire fino al prossimo momento di sveglia. Questo protocollo è stato adottato ad esempio in *TASK* [36].
Il vantaggio di questo approccio è che non richiede una grande sincronizzazione, dato che i periodi di sleep e active time sono lunghi. Il problema è che però tutti i nodi, dato che si svegliano assieme, spediscono contemporaneamente, causando quindi molte collisioni. Inoltre, non è uno schema molto flessibile, poichè il pattern di wakeup è prestabilito e non si adatta dinamicamente a diverse condizioni di traffico o topologia della rete.

Staggered Wakeup Pattern
In questo schema, nodi che si trovano a differenti livelli dell'albero di routing si svegliano in momenti diversi. Ovviamente, nodi adiacenti hanno intervalli di attività che coincidono almeno in parte. Quindi, in tale intervallo ci sarà un periodo in cui il nodo riceve pacchetti dai suoi nodi figli e un periodo in cui trasmette i suoi pacchetti al suo nodo padre.
Esistono due tipi di *staggered pattern: backward* e *forward.* Il primo, mostrato in Figura 5.10, è di tipo backward, poiché viene ottimizzata la latenza a partire dalle foglie verso la radice. Nei forward succede il contrario (quest'ultimo è poco usato nelle WSN).

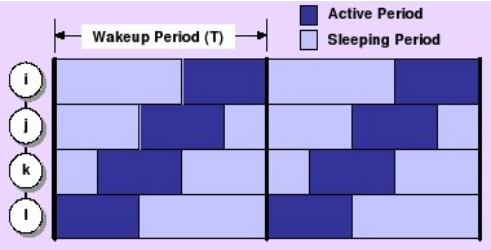


Figura 5.10: Pattern di attivazione dei nodi in uno schema Staggered Wakeup

Questo approccio viene usato ad esempio nel *D-MAC* e ha i seguenti vantaggi: innanzitutto, i nodi non sono tutti attivi contemporaneamente, quindi le collisioni e la contesa del canale sono diminuite. Inoltre, proprio per questo motivo, i nodi possono rimanere attivi meno a lungo. Infine, il fatto che i nodi padri raccolgano i dati dei figli consente una facile implementazione di data aggregation.
I difetti di questo schema sono gli stessi del Fully Synchronized Pattern, ovvero scarsa flessibilità e rischio di collisioni.

Adaptive Staggered Wakeup Pattern
Questi schemi sono stati creati con lo scopo di mitigare i problemi descritti nel paragrafo precedente. Infatti, in [38], l'intervallo di attività del nodo viene impostato al minimo valore possibile, minimizzando così il consumo energetico e la latenza della rete. Inoltre permette l'impostazione di intervalli diversi per nodi appartenenti allo stesso livello dell'albero, in modo da consentire che ogni nodo si adatti al traffico che deve gestire.

Flexible Power Scheduling
FPS [39], è sempre uno schema Adaptive Staggered Wakeup Pattern, ma con un approccio a *slot*, ovvero il tempo viene diviso in slot. Il problema con questo tipo di schemi è che non sono flessibili e serve un'accurata sincronizzazione tra i nodi. Per cercare di risolvere quest'ultimo difetto, FPS propone un sistema di prenotazione degli slot: quando un nodo deve trasmettere o ricevere, prenota un certo numero di slot, che verranno a lui riservati.

5.3.3.2 Asincroni

Gli schemi asincroni non necessitano di alcuna sincronizzazione. Ogni nodo si sveglia in maniera

indipendente, sapendo che comunque ci sarà qualche altro nodo attivo in quel momento.
Un esempio di questo approccio è già stato visto quando è stato spiegato *Solar Biscuit* nel Capitolo 3.

Asyncronous Wakeup Protocol
AWP [40] è in grado di rilevare i nodi vicini in un tempo finito senza richiedere un allineamento degli slot. L'idea è che ad ogni nodo è associato un *Wakeup Schedule Function*, usato per generare uno scheduling di wakeup. Affinché due nodi possano comunicare tra loro, i loro wakeup scheduling devono sovrapporsi. Ciò è possibile grazie al fatto che tutti i nodi hanno lo stesso duty-cycle. Un esempio di wakeup scheduling per 7 nodi, lo si può vedere in Figura 5.11.

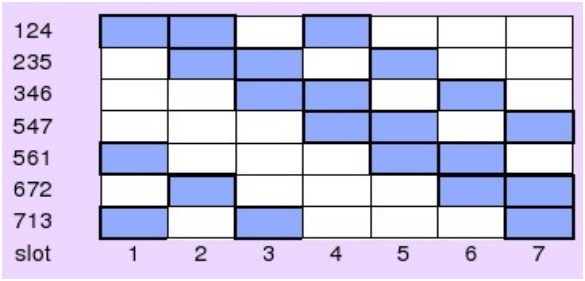


Figura 5.11: Esempio di scheduling asincrono. Le righe indicano i nodi, le colonne gli slot

I difetti di questa soluzione sono la latenza e il fatto che difficilmente tutti i nodi saranno attivi in contemporanea, impedendo di fatto di spedire messaggi broadcast.

Random Asynchronous Wakeup
RAW [41] si basa sull'assunto che tipicamente le Sensor Network sono caratterizzate da un'alta densità di nodi. Quindi, ci sono molte cammini possibili tra un mittente e il suo destinatario. Effettivamente, RAW è un misto di protocollo di routing e algoritmo di wakeup random. Il routing utilizzato è una variante del routing geografico: il nodo intermedio scelto è uno dei nodi attivi in quel momento che soddisfano certi criteri.
L'algoritmo di wakeup serve a far dormire i nodi per periodi casuali. Quando si svegliano, effettuano una procedura di *Neighbor Discovery,* per sapere a chi inoltrare i pacchetti. Una volta spediti i dati e dopo che è scaduto un tempo fissato, tornano a dormire.
Il pregio di questo schema è che è semplice da implementare e si adatta ai cambiamenti topologici della rete. Il principale difetto consiste nel fatto che non è detto che quando un nodo si sveglia, ci siano altri nodi raggiungibili, e quindi la garanzia di consegna dei pacchetti e la latenza sono i punti deboli.

5.3.4 Protocolli low duty-cycle MAC

5.3.4.1 TDMA-based

Nei protocolli TDMA il tempo è diviso in *frame* e i frame sono suddivisi in *slot***.** Ad ogni nodo vengono assegnati più slot di un frame, durante il quale esso può trasmettere o ricevere. Spesso i nodi sono raggruppati in cluster, dove il cluster-head si occupa di assegnare gli slot ai nodi [42].
Il vantaggio del TDMA è che esegue in maniera naturale, senza overhead aggiuntivi, il duty-cycle e viene minimizzata la possibilità di interferenze tra i nodi.
I problemi però non mancano: la sua scarsa scalabilità, la necessità di un'elevata sincronizzazione, la mancanza di flessibilità e adattamento ai cambianti topologici della rete, l'impossibilità di creare sempre uno scheduling senza interferenze e le scarse prestazioni in termini di utilizzo del canale e ritardo medio dei pacchetti in condizione di traffico basso, rendono protocolli di questo tipo poco efficienti. In quest'ultima situazione, il TDMA è peggiore del CSMA perché ogni nodo deve attendere

il proprio slot, anche quando il canale è subito libero.

TRAMA
TRAMA [42] divide il tempo in due parti, uno ad accesso casuale, l'altro ad accesso schedulato. Il periodo ad accesso casuale serve per l'assegnazione degli slot e impone ai nodi un protocollo di contesa per l'accesso al canale. L'altro invece è costituito dagli slot assegnati ai nodi.
L'algoritmo di assegnazione degli slot funziona in questo modo: prima i nodi ottengono informazioni sui nodi distanti al più due hop e che stanno richiedendo degli slot liberi, poi viene avviata una procedura per la scelta degli slot stessi. L'assegnazione avviene dopo aver applicato una funzione *hash* dipendente dall'identificativo del nodo e dal numero dello slot.

5.3.4.2 Contention-based

I protocolli di questo tipo sono i più utilizzati nelle Wireless Sensor Network. Sono robusti, scalabili e si adattano a diverse situazioni di traffico, tuttavia non consentono di ottenere lo stesso risparmio energetico dei protocolli TDMA, se non viene implementato uno schema duty-cycle adatto.

B-MAC
B-MAC (Berkeley MAC) [43] è il protocollo usato di default dai nodi con TinyOS. B-MAC implementa funzionalità basilari di accesso al canale, come lo schema a backoff, una stima accurata del canale ed acknowledge opzionali. Per raggiungere un basso duty cycle, ricorre ad uno schema sleep/wakeup basato su una fase di listening periodico chiamato *Low Power Listening (LPL)*. I nodi si attivano periodicamente per controllare l'attività del canale. Il tempo di wakeup è fisso mentre quello di controllo del canale può essere variato a livello applicativo.
I pacchetti del B-MAC sono costituiti da un lungo *preamble* e dal payload. La durata del preamble è almeno pari a quella dell'intervallo di controllo del canale, affinchè un nodo rilevi sempre quando c'è una trasmissione in corso.

S-MAC
In S-MAC (Sensor MAC) [44], i nodi si scambiano dei pacchetti di *sync* per coordinare i periodi di sleep/wakeup. Ogni nodo può seguire la sua schedulazione o quella di un vicino o entrambe se non si sovrappongono. Nodi che usano la stessa schedulazione formano un *cluster virtuale*.
Il periodo di accesso al canale è diviso in due parti. Durante il periodo di listen, i nodi si scambiano i pacchetti di sync e altri di controllo per evitare le collisioni. Nel periodo restante avviene il trasferimento dei dati. I nodi che non sono interessati dalla comunicazione in corso dormono fino al prossimo periodo di listen.
Per evitare alte latenze nelle reti multi-hop, viene adottato uno schema adattivo di listening: un nodo che riceve le trasmissioni del vicino, si sveglia per un breve periodo alla fine di queste. Se il nodo è il next hop della comunicazione, allora può già ricevere i pacchetti senza dover aspettare la prossima schedulazione. I periodi di listen e sleep sono costanti e non posso essere variati.

T-MAC
T-MAC (Timeout-MAC) [68] è un miglioramento di S-MAC, progettato per condizioni di traffico variabile. T-MAC implementa uno schema di sincronizzazione basato su cluster virtuali come in S-MAC. Una schedulazione definisce dei frame all'interno dei quali avviene la comunicazione tra nodi. Pacchetti in coda vengono spediti in *burst* all'inizio di un frame. Tra questi burst, il nodo può andare a dormire. Un nodo va nella fase di sleep se non si verificano eventi nel canale radio durante la sua fase attiva. È importante evitare che il nodo vada troppo presto a dormire e, a tal fine, T-MAC utilizza appositi meccanismi. Inoltre vengono spediti dei pacchetti di controllo speciali quando un nodo deve trasmettere. T-MAC riesce così ad avere prestazioni migliori dell'S-MAC in termini di latenza ed efficienza energetica.

D-MAC

D-MAC [45] è un protocollo adattivo ottimizzato per la raccolta dati in reti organizzate ad albero.
Questo protocollo diminuisce il problema della latenza che affligge solitamente i MAC a basso duty-cycle. Tale latenza è dovuta al fatto che un nodo deve svegliarsi prima di poter inoltrare i pacchetti ricevuti da un altro nodo. Le cose peggiorano quando gli hop da attraversare sono tanti. Per questo motivo in S-MAC e T-MAC il processo di forwarding è limitato a pochi hop.
In D-MAC invece, lo scheduling dei nodi dipende dalla loro posizione nell'albero, cioè nodi che appartengono allo stesso livello hanno periodi di attività molto ravvicinati o coincidenti, così da diminuire la latenza (vedasi Figura 5.12).

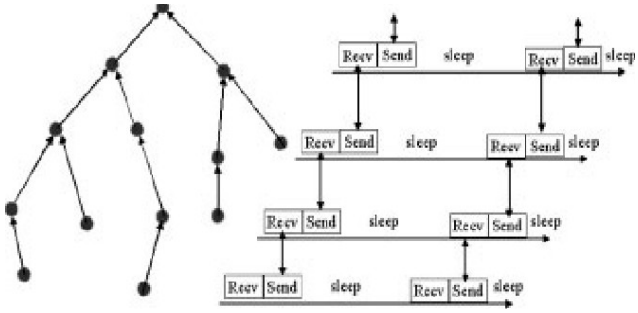


Figura 5.12: Schema di funzionamento del D-MAC in una rete con raccolta dati ad albero

Ad ogni nodo viene assegnato uno slot, sufficientemente lungo per la trasmissione di un pacchetto. Se deve trasmettere più pacchetti, richiede più slot. In questo modo i periodo di attività si adattano alle condizioni di traffico. Infine, D-MAC adotta uno schema di predizione per consentire a tutti i nodi figli di poter trasmettere.

IEEE 802.15.4 MAC

Una rete *PAN (Personal Area Network)* 802.15.4, è formata da un nodo coordinatore della PAN e, opzionalmente, da altri coordinatori. Gli altri nodi, detti *Reduced Function Device*, devono associarsi a uno di questi coordinatori (denominati *Full Function Device*), i quali gestiscono le comunicazioni all'interno della rete. Le tipologie di rete supportate, rappresentate in Figura 5.13, sono a *stella* (single-hop), *cluster-tree* e *mesh* (multi-hop).

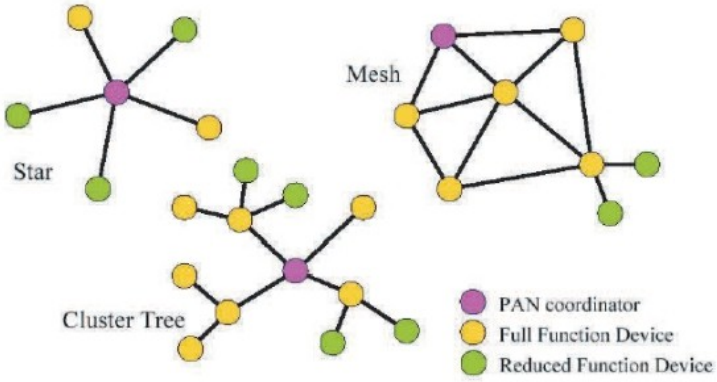


Figura 5.13: Topologie di una rete 802.15.4

L'802.15.4 definisce due modi di accesso al canale: *beacon enabled* e *non-beacon enabled.*
La prima consente una funzionalità di risparmio energetico basato su duty-cycle. In particolare utilizza una struttura *superframe* (vedasi Figura 5.14), delimitata dai beacon, che sono frame di sincronizzazione generati dai coordinatori periodicamente.

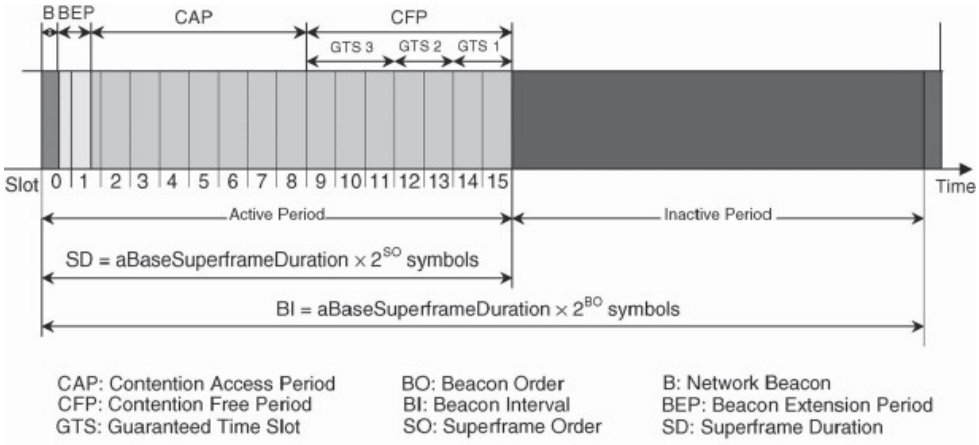


Figura 5.14: Struttura di un superframe con energy saving

Ogni superframe è composto da un *active period* e un *inactive period*. Nell'active period, i nodi comunicano con il loro coordinatore. Tali periodi possono essere suddivisi in *Contention Access Period (CAP)* e *Contention Free Period (CFP)*. Durante il CAP, i nodi accedono al canale usando uno schema CSMA/CA, mentre nel CFP vengono assegnati dei *Guaranteed Time Slots (GTS)* ai singoli nodi. Nell'inactive period, i nodi si trovano in uno stato di basso consumo energetico.
Nella modalità non-beacon, non esiste la struttura superframe e i nodi accedono al canale usando sempre il CSMA/CA.
La modalità beacon è adatta per situazioni single-hop. Tuttavia ci sono soluzioni, come quella in [46], che consentono l'utilizzo in ambito multi-hop, attraverso una divisione della rete in cluster.

X-MAC
X-MAC [48] mira a risolvere i difetti di B-MAC. Infatti, il lungo preamble utilizzato nel B-MAC comporta sprechi energetici, nonché latenze elevate e nodi non destinatari vengono coinvolti nella comunicazione (con ulteriore perdita di energia).
Per raggiungere questi obiettivi, X-MAC implementa un meccanismo con *short preamble.* Inoltre, in questo preambolo viene inserito l'indirizzo del nodo destinatario, in modo da evitare che i nodi estranei stiano svegli a ricevere dati non destinati a loro. L'altra novità consiste nell'utilizzo di uno *strobed preamble,* ovvero il nodo ricevente interrompe il lungo preamble non appena si sveglia e capisce di essere il destinatario. Così facendo si riduce l'energia consumata e il tempo di attesa per la ricezione dell'intero *long preamble.* Infine, X-MAC implementa un algoritmo per la regolazione del duty-cycle in base alle condizioni di traffico.
In Figura 5.15, si nota facilmente la differenza tra un classico protocollo Low Power Listening (LPL), come il B-MAC, e l'X-MAC.
In uno schema LPL, un nodo deve ricevere tutto il preamble di tipo long per capire se è il destinatario e solo a quel punto si sveglia per ricevere i pacchetti della comunicazione.
Invece, l'X-MAC manda una serie di short preamble (da qui il termine *short strobed preamble*); il nodo, non appena capisce da uno di questi short strobed preamble di essere il destinatario, si sveglia, manda un ack per informare il mittente di essere pronto e inizia la ricezione dei dati. In questo modo si guadagna tempo e si consuma meno energia rispetto all'approccio LPL.

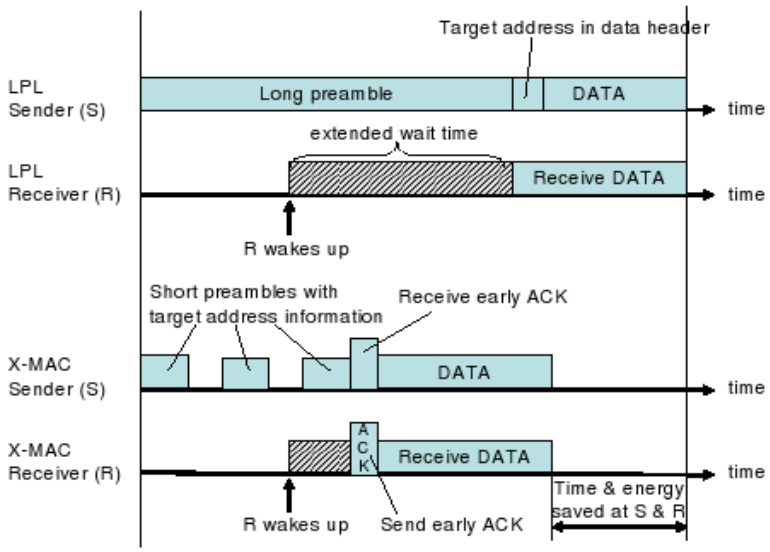


Figura 5.15: Confronto delle timeline tra uno schema LPL e l'X-MAC

5.3.4.3 Ibridi

Questi protocolli cercano di combinare i punti di forza dei protocolli TDMA e Contention-based, rafforzando i loro punti deboli.

Z-MAC

Z-MAC [47] prevede una fase preliminare in cui vengono eseguite le operazioni di *neighbor discovery*, *slot assignment*, *local frame exchange* e *global time synchronization.*

Attraverso il neighbor discovery, ogni nodo costruisce una lista di nodi vicini distanti al massimo due hop. Questa lista viene poi usata da un algoritmo per distribuire gli slot ad ogni nodo, in modo che a nessun nodo della lista venga assegnato lo stesso slot. Facendo così, si ha la certezza di non avere interferenze tra nodi all'interno dei due hop di distanza.

Il local frame exchange serve a stabilire il time frame, dato che Z-MAC non usa uno frame unico per tutta la rete, poiché sarebbe molto difficile altrimenti adattarsi ad un cambiamento di topologia della rete. Z-MAC invece consente ad ogni nodo di avere il proprio time frame locale che dipende dal numero di vicini ed evita che ci siano contese con gli stessi.

Infine, il global time syncronization, sincronizza tutti i nodi ad un clock comune.

Lo slot locale di ogni nodo e il time frame assegnato vengono comunicati ai vicini (entro la distanza dei due hop). A questo punto termina la fase di setup e inizia quella di accesso al canale, regolata dal controllo di trasmissione: i nodi possono essere in due modalità, che sono definite *Low Contention Level (LCL)* e *High Contention Level (HCL).* Di default un nodo si trova nella LCL, tranne quando riceve un *Explicit Contention Notification (ECN)* entro un certo tempo. Questi ECN vengono spediti dai nodi quando notano un'alta contesa del canale. In HCL, solo i nodi proprietari dello slot corrente e i vicini del primo hop possono competere per l'accesso al canale. In LCL, qualsiasi nodo può competere, anche se i proprietari dello slot hanno sempre la priorità.

In questo modo, Z-MAC riesce a raggiungere un elevato utilizzo del canale anche in condizioni di bassa contesa, poiché un nodo può trasmettere non appena il canale è libero.

5.3.5 Protocolli di routing

5.3.5.1 Location-based

Questi protocolli sfruttano l'informazione di posizione o vicinanza dei nodi. Molti di questi, come GAF, SPAN e ASCENT, che sono già stati presentati, sfruttano questo dato anche per spegnere la radio

nei nodi non coinvolti nel routing.

GEAR
GEAR [49] divide il processo di forwarding in due step: forwarding verso la regione destinataria e forwarding all'interno della regione stessa. Il primo utilizza una stima dei costi basata sulla distanza dei nodi e l'energia residua. Il secondo comprende una combinazione di forwarding geografico e flooding ristretto.

Low Power GPS
In [50], si utilizzano ricevitori GPS a basso consumo per ottenere il *Minimum Energy Communication Network (MECN)*. Questo protocollo costruisce un grafo che tiene conto del consumo energetico necessario per trasmettere o ricevere pacchetti. Una volta che è disponibile questo grafo, viene costruita, da un algoritmo distribuito, una sottorete ottimale da usare per le comunicazioni.
Un'estensione del protocollo [51] trova la più piccola MECN, che consente un maggiore risparmio energetico se la regione di broadcast intorno al trasmettitore è circolare.

5.3.5.2 Gerarchici

LEACH
LEACH (Low Energy Adaptive Clustering Hierarchy) [52] è forse il più famoso protocollo di clustering. È composto da due fasi: la fase di *setup* e la fase di *steady*. Nella prima, i cluster head sono scelti attraverso un algoritmo casuale distribuito. I rimanenti nodi si uniscono ad un cluster head che consente loro di minimizzare l'energia per la comunicazione. Dopo questa associazione, i cluster creano uno scheduling per tutta la rete. Le trasmissioni dei dati avvengono durante la fase di steady. I nodi effettuano le rilevazione e le mandano ai cluster head, i quali le inoltrano al sink, dopo averle aggregate, come rappresentato in Figura 5.16.

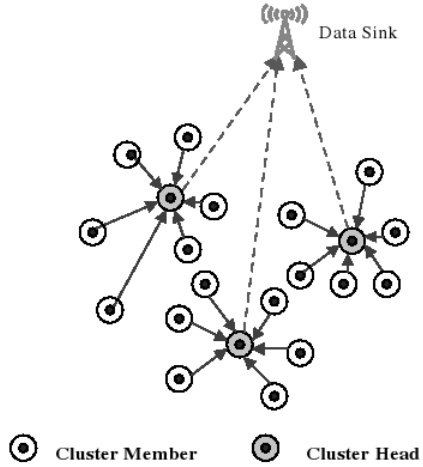


Figura 5.16: Modello della rete in LEACH

Questa fase dura molto di più di quella di setup, al fine di ridurre l'overhead del protocollo. Inoltre, la fase di setup si ripete periodicamente, per consentire una rotazione dei cluster-head.

Pegasis
Pegasis [53] è un protocollo che riprende e migliora LEACH, ricorrendo ad uno schema a catena.
All'inizio vengono costruite delle catene usando un algoritmo di tipo *greedy*. Poi, i dati vengono trasferiti e aggregati lungo queste catene. Solo un nodo della catena, il leader, trasmette i dati al sink. I leader si alternano per risparmiare energia.

TEEN

TEEN [54] è un protocollo di clustering *threshold-based*, progettato per applicazioni *time critical*, come la rilevazione di eventi.

In TEEN i cluster-head pubblicizzano due parametri: l'*hard threshold* e il *soft threshold.*

I nodi continuano a fare il sensing dell'ambiente, ma trasmettono ai cluster head solo se il dato è maggiori dell'hard threshold. Ciò serve a limitare il consumo energetico, dato che la radio rimane spenta la maggior parte del tempo. Per incrementare tale risparmio, le trasmissioni consecutive vengono autorizzate solo se la variazione della grandezza monitorata è maggiore della soft threshold. Inoltre, i cluster head ruotano periodicamente.

APTEEN

APTEEN [55] è un'estensione di TEEN, avente l'obiettivo di raggiungere una maggiore flessibilità.

Infatti, APTEEN può cambiare dinamicamente i parametri operativi per adattarsi alle esigenze dell'applicazione. Inoltre, il risparmio energetico è superiore grazie alle funzioni di aggregazione e scheduling delle trasmissioni implementate.

AODV

AODV (Ad-hoc On-demand Distance Vector) è un protocollo di routing reattivo ed è implementato nell'architettura ZigBee. Esso prevede l'utilizzo di tabelle di routing, salvate in ogni nodo, contenenti informazioni sul *next hop*. Se il next hop non è conosciuto, viene eseguito il processo di *route discovery*. Dato che il numero di percorsi che un nodo router può salvare è limitato, tale processo di route discovery avverrà più spesso in reti di grandi dimensione con molte comunicazioni fra i nodi.

Quando un nodo fonte deve scoprire il percorso di instradamento verso un nodo destinatario, manda in broadcast un comando di richiesta route. Questo messaggio contiene l'indirizzo del mittente, l'indirizzo del destinatario e un costo del path. Man mano che il comando di richiesta del percorso si propaga nella rete, i nodi che ricevono e ritrasmettono il pacchetto incrementano il campo del costo del path e aggiungono una entry alla loro tabella di route discovery. Quando finalmente il nodo destinatario riceve il messaggio di richiesta route, compara il costo del cammino appena ricevuto con quello degli altri messaggi ricevuti. Se il costo del percorso è più basso degli altri, trasmette al nodo fonte un pacchetto di *route reply*. I nodi intermedi ricevono il pacchetto e lo inoltrano fino al nodo da cui è partita la richiesta iniziale.

5.3.5.3 Flat

La forme più semplici di flat routing (o data-centric routing) sono il *flooding* e il *gossiping*.

La prima tecnica consiste nel mandare un pacchetto a tutti i nodi confinanti. In questo modo si è sicuri che il pacchetto arrivi a destinazione. Per evitare però che circoli nella rete all'infinito, un nodo deve inoltrare solo i pacchetti che non ha ancora ricevuto; quindi è necessario che vi sia un identificatore della fonte e un numero di sequenza nel pacchetto, per esempio. Inoltre, di solito è presente un'informazione che riporta la data di scadenza del pacchetto stesso, per evitare che continui a propagare nella rete anche quando il destinatario non è raggiungibile.

Il gossiping, invece consiste nel mandare il pacchetto ad un nodo vicino a caso, sperando di trovare alla fine quello giusto. Chiaramente, in questo approccio, il delay può essere assai elevato.

Un evoluzione di questi due schemi, consiste ad esempio nel mandare più pacchetti a random ai nodi vicini, oppure mandare il pacchetto a un sottogruppo di nodi (*controlled flooding*).

Come si può dedurre facilmente, sebbene tutte queste tecniche siano facili da implementare, sono in ogni caso poco efficienti, in termini di delay e di risparmio energetico. Per questo motivo, si ricorre spesso ai protocolli elencati di seguito.

Directed Diffusion

In *Directed Diffusion* [56], ogni dato è contraddistinto da una coppia *attributo-valore.* Il sink manda in broadcast un *interesse*, che è la descrizione di un task, contenente un *timestamp* e un gradiente.

L'interesse è collegato al dato specificato tramite la coppia attributo-valore. Ogni nodo salva l'interesse in una cache appena lo riceve. La disseminazione dei dati, ovvero la propagazione dell'interesse, imposta i gradienti relativi ai dati che rispondono all'interesse. Quando il nodo fonte ha dati di questo tipo, li manda attraverso il path del gradiente dell'interesse. La propagazione e aggregazione dei dati avviene localmente.

GBR

GBR (Gradient Based Routing) [57] migliora Directed Diffusion attraverso due accorgimenti. Il primo è che l'interesse contiene un contatore degli hop in riferimento al sink, in modo che il gradiente venga impostato in base alla distanza minima dal sink.
In secondo luogo, vengono implementati degli schemi di diffusione e fusione dei dati per bilanciare il carico tra i nodi e, quindi, migliorare il lifetime della rete.

EAR

EAR (Energy Aware Routing) [58] è uguale a GBR, tranne nel fatto che i dati vengono inoltrati al sink seguendo una cammino con link a basso costo energetico, invece che lungo un percorso con il minore numero di hop come avviene in GBR.
Per evitare di esaurire presto l'energia dei nodi lungo il *minimum energy path,* EAR sceglie uno dei molti path che consente di massimizzare il lifetime della rete.

SPIN

SPIN (Sensor Protocol for Information Negotiation) [59] manda i dati solo ai nodi che li hanno richiesti esplicitamente, proprio come avviene in Directed Diffusion.
SPIN si basa su una fase di negoziazione in cui vengono scambiati i descrittori dei dati (metadata). Le comunicazioni sono più efficienti poiché vengono trasmessi i descrittori invece dei dati stessi.
La procedura, riassunta in Figura 5.17, è composta da due fasi: prima i nodi pubblicizzano nuovi dati utilizzando i descrittori e aspettano che eventuali nodi li richiedano; poi, avviene la trasmissione dei dati. Un'altra caratteristica di questo protocollo è che è in grado di adattarsi in base all'energia residua dei nodi.

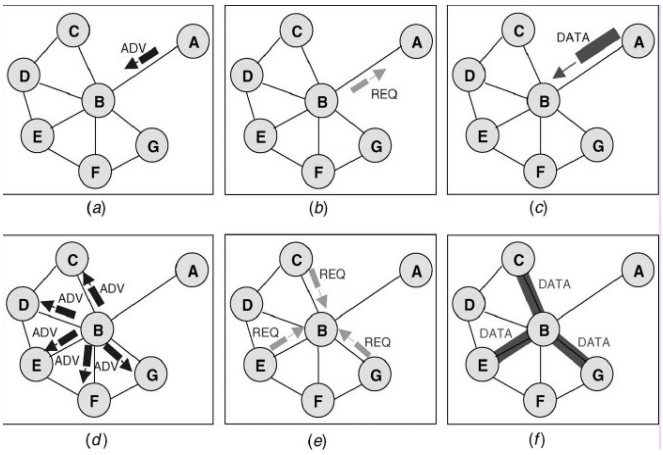


Figura 5.17: Funzionamento di SPIN: a) A pubblicizza i suoi nuovi dati (ADV) b) B richiedi i dati che gli interessano (REQ) c) A trasmette a B i dati che gli interessano d) B pubblicizza i suoi nuovi dati e) i nodi richiedono i dati che interessano loro f) B consegna i dati richiesti ai rispettivi nodi

Cougar

Cougar [60] considera la WSN un database distribuito e utilizza delle query per richiedere i dati ai nodi. Dei nodi intermedi tra il sink e il nodo destinatario, denominati nodi leader, effettuano data

aggregation.
I problemi di questo approccio sono fondamentalmente tre: serve un layer aggiuntivo nell'architettura dei nodi, il *query layer*, che consuma risorse hardware ed energetiche; è necessaria la sincronizzazione dei nodi per aspettare l'arrivo di tutti i dati; infine, i nodi leader devono ruotare, in modo da impedire la formazione di hotspot.

ACQUIRE
ACQUIRE [61] è l'acronimo di *ACtive QUery forwarding In sensoR nEtworks*, e come Cougar, considera la WSN una database distribuito. Il principio dietro questo protocollo è quello di iniettare nella rete un pacchetto contenente la query, che segue una traiettoria casuale (o predeterminata o guidata) attraverso la rete. Ad ogni passo, il nodo che riceve questa query effettua un update per ottenere informazioni da tutti i nodi vicini distanti *d* hop. Man mano che la query continua il suo percorso, viene risolta in componenti sempre più piccoli finché risulta completamente eseguita. A tal punto, ritorna al nodo che l'ha originata come una risposta completa, seguendo il percorso inverso o il più corto possibile.
Il vantaggio di questo approccio è che si possono creare query complesse. Inoltre, risulta essere un routing efficiente se il parametro *d* è scelto correttamente: infatti, se *d* è troppo grande si comporta come il flooding; se invece è troppo piccolo, la query deve attraversare molti più hop.

5.3.5.4 Network flow-based

I protocolli di routing di questo tipo calcolano i percorsi che i pacchetti devono seguire in base ai flussi di rete. Ad esempio, *Maximum Lifetime Energy Routing* [62] mira ad aumentare il network lifetime definendo i costi dei link come una funzione dipendente dall'energia rimanente del nodo e dalla potenza trasmissiva richiesta. Quindi cerca di calcolare la distribuzione del traffico (*network flow problem*). Il path meno costoso sarà quello con la maggiore energia residua tra tutti i path possibili.
Un altro protocollo di questo tipo è *Maximum Lifetime Data Gathering* [63], che cerca di ottimizzare la schedulazione della raccolta dati (*data gathering*) per minimizzare l'energia spesa in tale processo, effettuando il data aggregation se possibile. L'algoritmo utilizzato è costoso dal punto di vista computazionale.
Infine, il *Minimum Cost Forwarding* [64], mira a trovare il path a costo minimo in una rete estesa, semplice e scalabile. Per far ciò, la funzione di costo implementata tiene conto di variabili come il delay, il throughput e parametri energetici tra il nodo destinatario e il sink. Per trovare il costo ottimale, ad ogni nodo serve un solo messaggio.

5.3.5.5 QoS-based

SAR
SAR (Sequential Assignment Routing) [65] prende in considerazione le risorse energetiche e il QoS in ogni path, oltre al livello di priorità del pacchetto. La selezione del percorso è fatta dal nodo che origina il pacchetto. Ogni link è definito da un costo energetico e un delay, e una “resistenza” al passaggio dei pacchetti che può essere rappresentata in forma metrica per ogni path. Per superare questa resistenza, ogni pacchetto possiede un punteggio che gli può garantire la priorità di passaggio sul link, ad esempio, a bassa latenza ma con nodi con poca energia residua.
Per ogni pacchetto inoltrato nella rete viene calcolato un valore pesato di QoS che è il prodotto di parametri QoS del link e un coefficiente peso associato alla priorità del pacchetto. In questo modo la metrica QoS dipende dal QoS attribuito ad ogni pacchetto in relazione al livello di priorità dello stesso.
L'obiettivo di SAR è minimizzare la media pesata della metrica QoS per l'intero lifetime della rete.
Il difetto principale di questo approccio è l'elevato overhead dovuto al mantenimento delle tabelle di routing e degli stati.

Energy Aware QoS Routing

Questo protocollo [66] trova path energeticamente efficienti a costo minimo, in grado di minimizzare il delay end-to-end durante la connessione. I parametri considerati per il calcolo del costo sono la riserva di energia, l'energia necessaria per la trasmissione e il tasso d'errore.
Il protocollo supporta sia traffico best-effort sia real-time, grazie all'adozione di un modello a coda basato su classi (vedasi Figura 5.18).

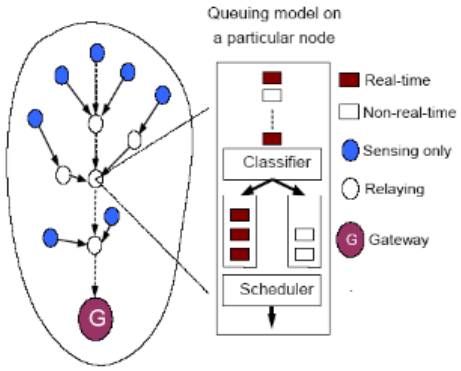


Figura 5.18: Modello a coda di un nodo

SPEED

SPEED [67] è un protocollo che prevede tre tipi di servizi di comunicazione real-time: *unicast, area-multicast, area-anycast*. L'obiettivo principale che si propone è quello di assicurare una certa velocità per ogni pacchetto nella rete. Per raggiungere questa meta, effettua delle deviazioni di traffico a livello Network e regola localmente i pacchetti spediti a livello MAC.
I componenti di SPEED, schematizzati in Figura 5.19, sono i seguenti:

- un API (Application Programming Interface);
- l'algoritmo *Stateless Non-deterministic Geographic Forwarding (SNGF)*: è il modulo di routing responsabile della scelta del canditato next hop che può supportare la velocità di consegna desiderata;
- *NFL (Neighborhood Feedback Loop)* e *Backpressure Rerouting*: sono moduli che si occupano di ridurre o deviare il traffico in caso di congestione, cosicché SNGF abbia candidati tra cui scegliere;
- *last mile process*: è un processo che serve a supportare i tre moduli appena descritti sopra;
- *delay estimation scheme*: è un meccanismo con il quale un nodo determina se è avvenuta congestione;
- *neighbor beacon exchange scheme*: serve a fornire la locazione geografica dei nodi vicini, affinché SNGF possa effettuare il routing geografico.

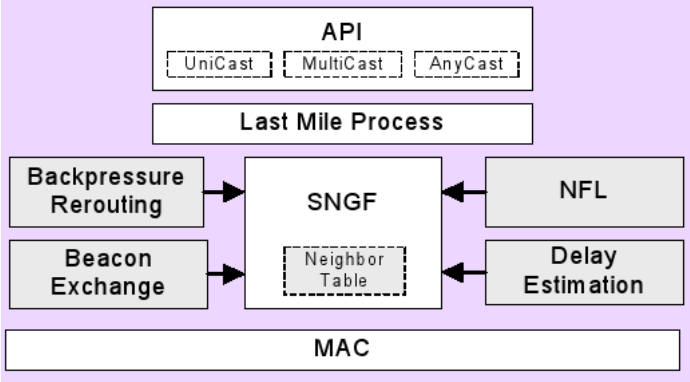


Figura 5.19: I componenti di SPEED

5.4 VINCOLI PROGETTUALI

I principali vincoli del progetto di questa tesi derivano dai nodi utilizzati, gli Squidbee, e sono:

- **memoria limitata:** la memoria flash di Arduino in cui è possibile salvare il software dei nodi è di soli 14 K. Non è possibile realizzare un sistema complesso in così poco spazio;
- **modulo radio pre-programmato**: il modulo radio XBee implementa già il livello fisico e il livello MAC dello standard 802.15.4. Perciò non è possibile eliminare le funzionalità di questi livello, ma solo modificare qualche parametro;
- **modulo GPS assente**: il nodo non è dotato di modulo GPS, dato che il suo costo è elevato, consuma parecchia energia e la posizione dei nodi è statica.

Un altro vincolo, di natura applicativa, richiede che l'intervallo minimo di reporting delle rilevazioni sia di 1 minuto. Ciò significa che i nodi devono essere in grado di effettuare il sensing dell'area e riportare i dati al sink almeno ogni minuto. Ovviamente l'utente potrà poi decidere se impostare un intervallo di 1 minuto o 10 o 50 o maggiore ancora. Inoltre, questo intervallo deve essere modificabile anche durante il funzionamento del sistema.

5.5 ARCHITETTURA REALIZZATA

5.5.1 Trasmissione dei bit a livello fisico

L'implementazione del livello fisico non è stato modificato in alcun modo, anche perché non è possibile né conveniente farlo. Dunque è stato mantenuto quello del modulo XBee, compatibile con lo standard 802.15.4. Per ulteriori informazioni sulle caratteristiche del livello fisico implementato nei moduli XBee, fare riferimento alla Sezione 5.3.1.

5.5.2 Topology control

Nella rete realizzata non è stato implementato il topology control. Questa scelta è stata effettuata per diversi motivi. Innanzitutto, dato che uno degli obiettivi del sistema è di essere economico, utilizzare più nodi del necessario significa costi maggiori. Quindi, la topologia ideale della WSN è quella in cui i nodi sono distanziati il più possibile l'uno dall'altro, in modo da coprire un'area estesa con un numero basso di nodi. È vero che così facendo, la perdita di un nodo non consente più di avere un sensing della zona che occupava ma, d'altronde, il sistema che si deve realizzare non è mission-critical, e la perdita di un nodo o di qualche pacchetto non comporta danni.
Il secondo motivo dell'assenza del topology control è la mancanza del GPS, che sarebbe tornato utile per applicare protocolli di questo tipo.
Infine, anche se si fosse usata un'altra tecnica, invece del GPS, per la rilevazione dei nodi vicini, come ad esempio l'RSSI, il codice per implementare il topology control avrebbe occupato una quantità consistente di memoria nella flash. Però, come già detto, questa risorsa è assai limitata.

5.5.3 Power management

Innanzitutto, come già accennato nella Sezione 5.4, non è possibile costruire un MAC da zero per i nodi Squidbee, poiché l'XBee implementa già il MAC dell'802.15.4. Per questo motivo è stato possibile solamente modificare alcuni parametri. L'implementazione di un altro protocollo low duty-cycle MAC, al posto del CSMA/CA adotatto dall'XBee, non sarebbe stata fattibile.
Il modulo XBee permette di scegliere tra tre modalità di MAC: *802.15.4 con ack, 802.15.4 senza ack, MaxStream compatibile 802.15.4*.
La modalità utilizzata nel progetto è la *Maxstream*. Questo tipo di MAC è compatibile con quello 802.15.4 ma in più aggiunge un header Maxstream nel frame 802.15.4, che permette di rilevare i

pacchetti duplicati, di ripetere il *Clear Channel Assestment* in caso di errore e di utilizzare dei comandi di *Network Discovery* e di risoluzione di identificativi (comandi non utilizzati nel progetto). Questa modalità MAC è stata preferita alle altre due perché sembra garantire una migliore efficienza della rete in termini di integrità dei pacchetti. Purtroppo ha come difetto di contenere un header aggiuntivo, che costa un poco di più in termini di energia e tempo.
A questo MAC è stato affiancato un protocollo di sleep/wakeup di tipo scheduled rendezvous.
Infatti, questo tipo di schema si adatta molto bene alle caratteristiche del sistema di monitoraggio ambientale, dove è sufficiente che in nodi si sveglino periodicamente e inviino i dati, senza particolari vincoli di latenza. I protocolli *sleep/wakeup on demand* sarebbero risultati troppi costosi e oltre le necessità: questi richiedono infatti la presenza di un secondo modulo radio su ogni nodo e permettono di massimizzare il tempo di sleep della rete e minimizzare la latenza. Sono la soluzione ideale per quelle applicazioni in cui non si sa quando un singolo nodo potrebbe svegliarsi, ma nel sistema di monitoraggio ambientale, i nodi devono attivarsi insieme periodicamente.
Viceversa, i protocolli *sleep/wakeup asincroni*, si basano sull'idea che in qualsiasi momento ci sia almeno un nodo attivo per ricevere i pacchetti di un altro nodo. Questa supposizione però è poco credibile nella rete di sensori che si vuole realizzare e non pone le basi per un sistema sufficientemente affidabile.
Nella WSN di questo progetto, i nodi sono pochi e sono distanziati il più possibile l'uno dall'altro, per risparmiare sui costi. Perciò non è detto che ci sia un nodo attivo nelle vicinanze per ascoltare le comunicazioni di un nodo trasmittente. Inoltre, avendo la necessità di un reporting dei dati periodico, con un protocollo asincrono questa esigenza non è soddisfacibile.
Invece, gli schemi scheduled rendezvous prevedono una schedulazione delle fasi di sleep e wakeup che ogni nodo deve seguire, in modo che tutti i componenti della rete si attivino contemporaneamente e spediscano i dati al sink. Questo è proprio quello che richiede il sistema di monitoraggio di questa tesi.
Per la creazione del protocollo scheduled rendezvous di questo progetto, sono stati ripresi alcuni concetti e idee di S-MAC, T-MAC, e sono state inventate nuove soluzioni in modo da adattarlo alle esigenze e ai vincoli del sistema.
Infatti, anche potendo, molto probabilmente non si sarebbe potuto implementare uno di quei MAC, così come molti altri presenti in letteratura, poiché il codice avrebbe richiesto troppo spazio in memoria.
Come in S-MAC, il protocollo implementato utilizza una schedulazione sleep/wakeup e divide il periodo di attività in due parti: nella prima i nodi ricevano i messaggi di sincronizzazione, mentre nella seconda ricevono pacchetti di dati o di controllo. Un'altra somiglianza con S-MAC è che i nodi sono *loosely synchronized*, ovvero non è necessaria un'elevata sincronizzazione. I concetti che sono presenti anche in T-MAC sono la divisione del tempo in frame in cui un nodo può trasmettere e il fatto di spegnere la radio non appena non ci sono più nodi che devono comunicare con il nodo sveglio. Inoltre, il risultato finale è un protocollo di sleep/wakeup con idee simili a quelle del Flexible Power Scheduling, poiché ad ogni nodo viene assegnato uno slot di tempo in cui trasmettere, ma non viene assegnato on demand.
Il meccanismo di risparmio energetico implementato si occupa solo dell'attivazione e dello spegnimento della radio, mentre la gestione dello stand-by del microcontrollore è previsto come sviluppo futuro.
Il principio su cui si basa il protocollo realizzato consiste nell'assegnazione di uno slot univoco ad ogni nodo della rete. Tale slot corrisponde al valore dell'indirizzo identificativo del nodo a 16 bit (detto *MY*), per cui non viene introdotto alcun overhead per la gestione dello scheduling, ovvero non vengono scambiati ulteriori messaggi a questo scopo. Ogni nodo segue tre sottofasi: *Beacon, Connection, Data*.
Nella prima, il nodo aspetta l'arrivo del beacon del proprio cluster-head per la sincronizzazione.
Una volta ricevuto il beacon, il nodo entra nella fase di Connection, dove può ricevere richieste di connessione da nodi orfani, inoltrare tali richieste di altri nodi e ricevere/inoltrare i messaggi dal gateway. Queste due primi fasi sono di durata fissa.
Poi il nodo passa alla fase di Data, di durata fissa $T_{data,}$ durante la quale spedisce le proprie

rilevazioni al proprio padre e inoltra quelle provenienti dai nodi figli.
In questa fase il nodo si attiva solo quando sa che ci deve essere una trasmissione in ingresso o in uscita. Dunque, accende la radio solo quando è il turno del suo slot, per spedire i propri dati rilevati, oppure quando c'è un nodo figlio che deve comunicare con esso. Gli slot assegnati ai figli, vengono mantenuti in memoria dal nodo padre, in questo modo si sveglia solo al momento opportuno. Questa conoscenza degli slot viene appresa nel momento in cui esso manda il messaggio di tipo *answer orphan* al nodo orfano, contenente il *MY* assegnato.
Questa suddivisione in slot serve ad evitare che più nodi trasmettano contemporaneamente, causando così collisioni e un rapido riempimento dei buffer in ingresso dei nodi.
Sicuramente mantenere una tabella degli slot dei figli in ogni nodo può consumare memoria, tuttavia il vantaggio è che viene risparmiata energia che altrimenti sarebbe stata utilizzata per la trasmissione di messaggi per chiedere ogni volta ai vari nodi qual è il loro slot.
La soluzione adottata va bene finché la rete non ha un numero molto elevato di nodi, dopodiché la dimensione delle tabelle degli slot diventa eccessiva. Inoltre, la schedulazione delle trasmissioni rende la rete poco flessibile. Tuttavia questi sono dei compromessi, che hanno consentito la creazione di una rete caratterizzata dall'affidabilità della consegna dei pacchetti pacchetti e da un buon valore di latenza. Infatti, se fosse stato implementato un protocollo con la raccolta dei dati a livelli, si sarebbe causato un certo ritardo nell'arrivo dei dati, dovuto alla necessità di raccogliere i dati dei nodi livello per livello.
Il funzionamento in dettaglio del protocollo di sleep/wakeup creato è il seguente:
la struttura della rete è ad albero, ed ogni nodo trasmette, ad ogni intervallo di trasmissione, dei pacchetti di beacon. Non appena un nodo riceve un beacon dal proprio nodo padre, resetta un timer e da quel momento rimane “pronto” per tutta la fase *Active* (vedasi Figura 5.20) di durata T_{active}, dopodiché si spegne per un tempo pari a $T_{sleep} = T_{interval} - (T_{active} + T_{safety})$. T_{sleep} indica la lunghezza della fase *Sleep;* T_{active} è un valore fisso prestabilito, pari a 40 secondi. T_{safety} è anch'esso un valore fisso predeterminato pari a 3 secondi; la sua funzione è quella di salvaguardare il nodo dal drift della sincronizzazione. Trascorso il periodo di T_{sleep}, si entra nella fase *Safety*,, dove il nodo accende la radio e i primi tre secondi da questo momento servono solo per evitare che un nodo non ben sincronizzato mandi pacchetti in anticipo senza che nessuno li riceva.
Dopo questi tre secondi, il nodo passa alla fase Active, più precisamente nella sottofase *Beacon* (vedasi Figura 5.21), dove dovrebbe ricevere il beacon dal proprio padre entro T_{beacon} pari a 300 msec. Non appena lo riceve, riazzera il suo timer. Se invece non riceve il beacon, dopo T_{beacon} passa alla sottofase *Connection* e azzera “manualmente” il suo timer. Se un nodo non riceve beacon dal proprio padre per quattro volte consecutive, presume che il padre abbia qualche problema e inizia la procedura di ricerca di un nuovo padre.
Trascorso $T_{beacon,}$ il nodo passa alla sottofase Connection di durata $T_{connection}$, dove invia il proprio beacon e attende la ricezioni di messaggi di controllo come richieste di connessione *ask connection* da nodi orfani, o inoltra messaggi di questo tipo o riceve le risposte dal gateway *gateway response*. Sia in Beacon sia in Connection, la radio è sempre accesa.
Passato il tempo $T_{connection}$, il nodo entra nella sottofase Data, utilizzata per la trasmissione/ricezione delle rilevazioni dei sensori. In questa fase, il nodo attiva la radio solo quando deve trasmettere i propri dati durante il suo slot di tempo di durata T_{slot}, oppure quando deve ricevere i dati dai propri figli. Come già spiegato, il nodo sa quando un proprio figlio trasmetterà e, di conseguenza, attiva la radio un poco prima della trasmissione e la spegne un poco dopo, in modo da avere dei margini di sicurezza.
Questa fase termina allo scadere del tempo T_{active}. A questo punto il nodo entra nella fase di Sleep vera e propria, in cui non fa nulla, se non aspettare la fase Safety, così da ripetere il ciclo nuovamente. Sebbene nella fase Active e Sleep il nodo non faccia nulla per la maggior parte del tempo, le due si differenziano perché, in linea teorica, nella prima un nodo con moltissimi figli potrebbe rimanere attivo per tutta questa fase, mentre nella fase Sleep sicuramente il nodo dorme.
Uno schema che riassume le fasi di un nodo è riportato nell'Appendice C.

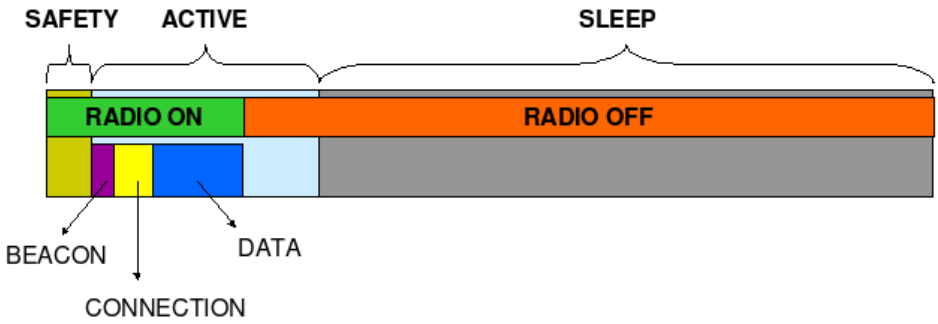


Figura 5.20: Le fasi seguite da un nodo

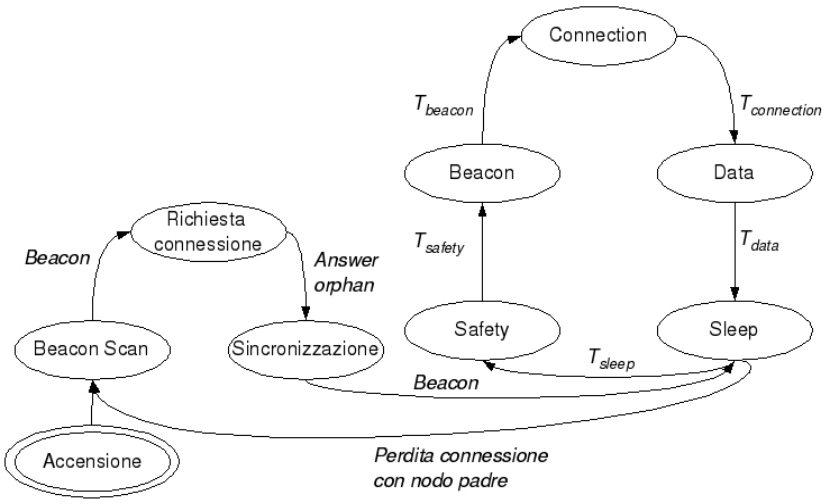


Figura 5.21: Diagramma a stati finiti di un nodo

5.5.4 Compiti di un nodo

5.5.4.1 Ingresso nella rete

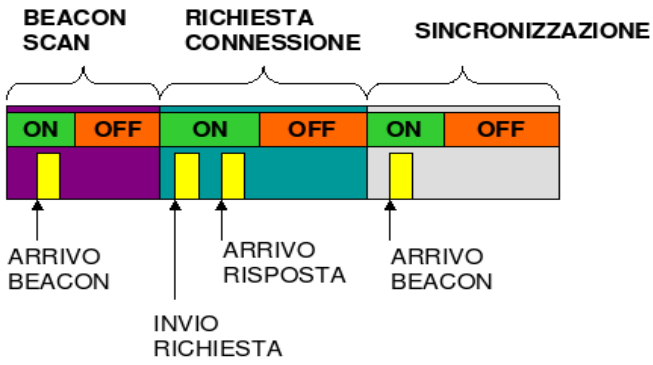


Figura 5.22: La schedulazione seguita da un nodo che vuole entrare nella rete

La prima cosa che deve fare un nodo non appena viene acceso è cercare di entrare nella rete WSN presente. Per far ciò, esegue il seguente meccanismo, schematizzato in Figura 5.21 e in Figura 5.22:

1. Subito dopo l'accensione, il nodo entra nella fase di Beacon Scan: il nodo tiene la radio attiva finché non riceve almeno un beacon. Dopodiché, aspetta ancora qualche istante, per ricevere anche eventuali beacon in ritardo (la spedizione dei beacon avviene a livelli in maniera quasi sincronizzata). Il beacon contiene questi campi: indirizzo MY mittente, numero di hop di distanza dal sink, numero figli del nodo che lo ha spedito, intervallo di trasmissione dei dati.

2. A questo punto, il nodo esegue un algoritmo per trovare a quale nodo connettersi, e quindi quale nodo diventerà suo padre. Esso viene scelto in base al numero di hop e al numero di figli, specificati nel beacon: la regola è che viene selezionato quello con il minor numero di hop, salvo che questo abbia già un certo numero di figli in più rispetto agli altri. Se però, il nodo disponibile come alternativa ha un numero inferiori di figli, ma ha un numero di hop maggiore di un *Hop threshold*, allora viene tenuto lo stesso il nodo scelto inizialmente.
   Questa politica di selezione ha lo scopo di bilanciare il carico della rete tra tutti i nodi ed evitare così che un nodo debba inoltrare i pacchetti di tanti figli.
3. Una volta scelto il candidato padre, il nodo spegne la radio ed entra nella fase di sleep, aspettando il prossimo intervallo in cui i nodi si sveglieranno. Per calcolare la durata dello sleep, utilizza l'intervallo di trasmissione specificato nel beacon, a cui viene sottratto l'intervallo di attività.
4. Non appena termina il periodo di sleep, il nodo si sveglia e calcola il valore *Join Backoff*. Join Backoff è un valore di delay casuale in millisecondi non superiore alla durata di $T_{connection}$. Lo scopo di Join Backoff è evitare che tutti i nodi orfani inviino contemporaneamente la richiesta di connessione al padre e di separare il traffico dati regolare da questi messaggi di controllo, che vengono trasmessi nella sottofase Connection della fase Active.
5. A questo punto, il nodo, dopo aver aspettato Join Backoff, invia al candidato padre un messaggio di richiesta connessione. In questo pacchetto vi è l'indirizzo Serial Low del nodo che l'ha spedito, l'indirizzo MY del nodo padre, l'indirizzo MY del nodo. Il Serial Low è la parte bassa di 32 bit dell'indirizzo univoco IEEE a 64 bit che ogni nodo possiede.
6. Quando il padre candidato riceve questo messaggio, se non è lui stesso il sink, lo inoltra verso il sink.
7. Gli eventuali nodi intermedi, salvano nella propria tabella di routing l'indirizzo MY dell'ultimo nodo che ha inoltrato questo messaggio e l'indirizzo MY del candidato padre.
8. Una volta che la richiesta giunge al sink. questi assegna un nuovo indirizzo MY al nodo entrante e salva nella tabella dei nodi l'indirizzo Serial Low del nuovo nodo, l'indirizzo MY del padre e l'indirizzo MY assegnato.
9. A questo punto il sink spedisce indietro la risposta al nodo padre, detta *gateway response*, contenente l'indirizzo MY del padre, l'indirizzo Serial Low del nodo entrante, l'indirizzo MY assegnatogli e il numero di hop di distanza dal nodo.
10. Quando i nodi intermedi ricevono un gateway response ma non sono loro i padri del nodo entrante, controllano nella loro tabella di routing a chi devono inoltrare il pacchetto e gli spediscono il messaggio. Di fatto, la risposta segue il percorso inverso rispetto all'andata.
11. Una volta che il messaggio di gateway response giunge al padre, questi prepara un nuovo messaggio, detto *answer orphan*, contenente il Serial Low del nodo entrante, l'indirizzo MY assegnatogli e il numero di hop dal sink e lo spedisce a tutti i nodi orfani che sono nella sua portata radio, poiché essi non hanno ancora un indirizzo MY proprio e il nodo padre non può indirizzare il nodo specifico.
12. Ora il nodo orfano controlla, tramite il campo del Serial Low del pacchetto, di essere lui il destinatario. In tal caso imposta l'indirizzo MY che gli è stato assegnato e ricava il valore del suo slot $T_{slot} = (MY+3)*Slot\ Duration$, dove *Slot Duration* indica la durata di uno slot, fissata a 500 msec.
13. A questo punto, il nodo torna a dormire, in attesa nella prossima fase di attività
14. Nella seguente fase di attività, detta fase di Sincronizzazione, il nodo attende il beacon del padre e passa alla fase sleep, aspettando il prossimo intervallo di attività. Questa fase serve solo a garantire che il nodo venga sincronizzato correttamente.
15. Nella nuova fase di attività il nodo è di fatto un nuovo nodo della rete, e svolge tutti i compiti di un nodo della WSN: manda il suo beacon, trasmette e riceve dati, inoltra pacchetti e accetta richieste da nodi orfani.

Uno schema che riassume questo meccanismo è riportato nell'Appendice C.

5.5.4.2 Trasmissione dei beacon

I beacon vengono trasmessi da tutti i nodi connessi all'inizio di ogni intervallo di trasmissione. Il loro scopo è quello di sincronizzare i nodi, nonché informare della presenze di un potenziale padre i nodi non connessi.

5.5.4.3 Sensing

Ogni nodo, eccetto il sink, effettua nella propria fase Active una rilevazione con i sensori. Questi dati vengono trasmessi, tramite un messaggio *sensor data,* verso il sink durante gli slot assegnati. Tale messaggio contiene l'indirizzo MY del nodo che ha rilevato i dati, un numero di sequenza, il valore del sensore luminosità, il valore del sensore umidità, il valore del sensore di temperatura. Viene quindi spedito il valore campionato del sensore, e non la grandezza finale in lux o °C, ad esempio.
In questo modo, i nodi vengono sgravati dal compito di effettuare la conversione, che coinvolge anche variabili *float* ed è pesante sia per la CPU sia per la memoria. Questa conversione verrà effettuata poi dal web server.

5.5.5 Routing

Per la scelta del protocollo di routing, sono stati esclusi quelli di tipo network flow-based, QoS-based e location-based. I primi due non si adattano al tipo di WSN da realizzare, poiché non vi è nessuna necessità di rispettare vincoli di traffico o di Quality of Service nella rete. I dati che attraversano la rete sono sempre gli stessi, ovvero le rilevazioni dei sensori, e non c'è nessuna esigenza di dare priorità ad alcuni di essi o risolvere problemi di saturazione dei link, dato il protocollo di sleep/wakeup scelto; i messaggi di controllo della WSN, come quelli di richiesta di connessione, sono trattati come dei pacchetti uguali agli altri.
I protocolli location-based sono stati scartati perché richiedono o un GPS per ogni nodo o un informazione di vicinanza tra nodi tipo RSSI. Il GPS è costoso e consuma risorse; l'RSSI è un indicatore inaffidabile, perché è in continua fluttuazione, a cause delle molte variabili che influiscono sulla propagazione radio.
I due tipi di algoritmi di routing rimanenti sono quelli flat e gerarchici. Per quanto riguarda i flat, sono stati esclusi subito quelli che seguono schemi di flooding e gossiping, poiché molto inefficienti. Rimangono quindi quelli basati sull'assegnazione di coppie attributo-valore, come Directed Diffusion, oppure basati su meccanismi di publish/subscribe, come SPIN. Queste ultime tecniche vanno bene in quei contesti in cui vi è un'applicazione che interroga la rete di sensori per ottenere informazione specifiche da essa. Il sistema che si vuole realizzare però si limita a richiedere alla WSN tutti i dati rilevati da ogni nodo, in modo da avere sempre una visione globale dell'area monitorata. Perciò, si presume che non vi sia un utente che desideri solo certi dati da certe aree: non è stato seguito, per questo progetto, l'approccio di considerare la rete come un database distribuito, onde evitare di rendere troppo complicato il sistema.
Infine, i protocolli gerarchici, consentono di creare strutture della rete ad albero, che ben si adattano al meccanismo di power management scelto, in cui i vari cluster-head si occupano di tenere sincronizzati i propri figli e di inoltrare i loro pacchetti. Inoltre, la rete e i nodi non vengono appesantiti da informazioni di routing eccessive, e vige il principio della località, ovvero un nodo sa come raggiungere solo i suoi vicini.
Quindi il protocollo di routing scelto è di tipo gerachico ed è stato realizzato cercando di renderlo semplice e basilare, per consentire una bassa occupazione delle risorse hardware. Più precisamente, è stato creato un *routing reattivo gerarchico*: reattivo poiché ricerca il percorso per un nodo sconosciuto solo se l'ha bisogno; gerarchico perché la rete è strutturata ad albero. In tale struttura ad albero i nodi padre, o cluster-head, ricevono i pacchetti dai nodi figli e li inoltrano verso il sink.
Solitamente i dati seguono un forward path, ovvero dai nodi verso il sink. In questo caso, ogni nodo

figlio sa chi è il proprio nodo padre e trasmette i messaggi a lui solo. L'informazione dell'indirizzo MY del proprio nodo padre viene appresa in fase di ingresso nella rete.
Tuttavia esiste la possibilità di un reverse path (dal sink ai nodi) nel momento in cui il sink spedisce il messaggio di *gateway reply* verso il padre di un nodo entrante. In questo caso, come già spiegato, i nodi mantengono una tabella di routing, per sapere quale nodo indirizzare nel percorso inverso. Questa tabella ha una dimensione piuttosto limitata; perciò se molti nodi avviano la procedura di ingresso nella rete, la tabella non potrà contenere il percorso verso ognuno di essi e il pacchetto verrà scortato. Ciò non rappresenta un grave problema, poiché non è molto alta la probabilità che tanti noti inviano quasi contemporaneamente la richiesta di connessione. Inoltre, se un nodo orfano non riceve alcuna risposta dal gateway entro un certo timeout, invia nuovamente la richiesta.

5.5.6 Data aggregation

Non è stato implementato alcun meccanismo di data aggregation, per motivi riconducibili alla tipologia di applicazione del sistema. Infatti, l'utente imposta un valore di intervallo di trasmissione dei dati e tale valore indica sia ogni quanto tempo i nodi devono effettuare il sensing dell'ambiente, sia ogni quanto devono trasmettere le rilevazioni al sink. Quindi, se l'intervallo scelto è di 5 minuti, significa che l'utente vuole visualizzare sul proprio computer tutti i dati di tutti i nodi rilevati ogni 5 minuti.
Il data aggregation sarebbe tornato l'utile nel caso in cui l'utente non avesse avuto questa esigenza di visualizzare i dati “in tempo reale” di ogni nodo. Ad esempio, se l'utente fosse stato interessato a visualizzare la media dei dati dell'ultima ora, con misurazioni effettuate ogni 10 minuti, allora si sarebbe potuto effettuare un data aggregation per salvare i dati e spedirli al sink solo una volta all'ora.
Inizialmente si era pensato di implementare un algoritmo di *delta enconding*, che consiste nello spedire la prima serie di dati normalmente e le serie successive calcolate come differenza rispetto alle precedenti. Ciò avrebbe consentito la trasmissione di pacchetti più piccoli e, quindi, un risparmio energetico. Ciononostante, alla fine è stato deciso di non adottarlo per l'assenza di garanzia di consegna dei pacchetti e per la mancanza di spazio in memoria per implementarlo. Il primo motivo è stato quello determinante, infatti, in caso di perdita di una serie, quella successiva non avrebbe avuto senso senza i valori di riferimento della precedente.

CAPITOLO 6

PROGETTAZIONE DEL SISTEMA DI GESTIONE DELLA WSN

6.1 DESCRIZIONE GENERALE DEL SISTEMA

Nella specifica dei requisiti software, riportata nell'Appendice A, sono state illustrate in dettaglio tutte le caratteristiche che questo sistema deve avere.
Per rispondere a tali requisiti, è stato implementato un database nel gateway ed è stato sviluppata la logica applicativa per permettere al nodo sink di salvare i dati nel gateway.
I dati che il gateway salva sono le rilevazioni provenienti dai nodi e le informazioni riguardanti la WSN, come le associazioni e lo stato dei nodi. Inoltre, il database è consultato anche dal sink stesso per sapere alcuni parametri che deve applicare alla WSN, come l'intervallo di trasmissione. Oltre a ciò, è stato realizzata un'applicazione web per consentire agli utenti di visualizzare, via browser, i dati dei sensori e dei nodi della WSN, nonché di impostare alcuni parametri della rete. Per garantire all'utente risposte del sistema in tempo reale, una buona usabilità ed una certa interattività, è stata creata una *Rich Internet Application* usando le tecnologie più recenti e più adatte allo scopo.

6.2 DATABASE

Il primo passo per l'implementazione del database è stato scegliere quale DBMS utilizzare.
Il database ideale avrebbe dovuto avere le seguenti caratteristiche (oltre a quelle tipiche di un database):

- **leggerezza**: è assolutamente importante che il database occupi pochissime risorse hardware, sia per l'installazione sia durante il suo funzionamento;
- **buone performance**: dato il contesto in cui il database viene adoperato, le tabelle assumono abbastanza velocemente grandi dimensioni. Il database deve essere in grado di rispondere comunque in tempi ragionevoli alle query che riceve;
- **disponibilità API**: l'esistenza di API complete e ben documentate, preferibilmente per il linguaggio *C*, consente uno sviluppo più rapido dell'applicazione;
- **open source**: ovviamente, come tutto il software in questo progetto, il DBMS deve essere open source e deve essere installabile su un sistema Linux.

Dunque, caratteristiche come la gestione avanzata di più utenti o funzioni di sicurezza particolari non vengono richieste. Dati quindi questi vincoli, sono stati scartati i più diffusi DBMS come *PostgreSQL* e *MySQL*, poiché contengono tante funzionalità avanzate che non servono al progetto in questione e, inoltre, occupano molto spazio in memoria in fase di installazione. In particolare, MySQL occupa quasi 100 Mb, mentre PostgreSQL circa 20 Mb ma è più lento di MySQL.
Alla fine la scelta è caduta su *SQLite* [105], che è proprio il database con le caratteristiche ideali per il sistema realizzato.

6.2.1 SQLite

SQLite è un database relazionale embedded open source. È stato sviluppato con lo scopo di fornire un modo comodo di gestire dati senza l'overhead introdotto dai classici database relazionali. SQLite è famoso per la sua alta portabilità, la facilità di utilizzo, la compattezza, l'efficienza e l'affidabilità.

Esattamente le caratteristiche del database cercato.
SQLite è di tipo embedded, ovvero non gira come un'applicazione indipendente, bensì coesiste all'interno dell'applicazione che serve. Quindi, il suo codice è integrato come una parte del programma che lo ospita. Il vantaggio che ne deriva è che server e client sono nello stesso processo e non richiedono dunque configurazioni di rete.
SQLite offre molte funzionalità, nonostante le sue dimensioni di soli 250 KB. Supporta un gran numero di query *ANSI SQL92* (transazioni, viste, controlli sui dati, query composte), trigger, indici, colonne con auto-incremento, clausole *LIMIT* e *OFFSET*.
I principi che SQLite segue sono:

- **Nessuna configurazione**: è assente qualsiasi forma di Database Administration; tutto ciò che serve è incluso nel nucleo del programma.
- **Portabilità**: SQLite gira sotto qualsiasi piattaforma che sia Linux, Windows, Mac, Solaris, Symbian o Palm Os. Inoltre funziona su architetture a 16, 32 e 64 bit. La portabilità poi riguarda non solo il software ma anche il file del database. Infine, SQLite può contenere fino a 2 TB di dati (se il sistema operativo lo permette).
- **Compattezza**: SQLite è stato progettato per essere piccolo e leggero: infatti comprende solo un file header e una libreria e il tutto non occupa più di mezzo megabyte. Altrettanto compatti sono i database creati, che sono dei normali file del sistema operativo: tutti gli oggetti, come tabelle, trigger, schemi, indici e viste sono contenute all'interno di un singolo file. Inoltre, i record sono a lunghezza variabile, quindi viene allocato solo lo spazio necessario per mantenere il dato del campo.
- **Semplicità**: l'API di SQLite è semplice e facile da utilizzare, ben documentata e intuitiva; grazie a ciò, una vasta community sviluppa in continuazione nuove librerie.
- **Flessibilità**: SQLite combina la potenza e la versatilità del front-end di un database relazionale con la semplicità e compattezza del back-end utilizzato.
- **Nessun costo di licenza**: tutto il codice è di pubblico dominio.
- **Affidabilità**: tutto il codice in SQLite viene ampiamente testato.
- **Convenienza**: SQLite ha usa serie di caratteristiche, come la possibilità di aggiungere diversi database ad una singola sessione, che lo rendono adatto a molti scopi.

In termine di prestazioni, SQLite è tanto veloce, se non più veloce, dei più diffusi DBMS, nell'effettuare operazioni tipo *INSERT, UPDATE o SELECT* su una tabella. Se invece si utilizzano query molto complicate, allora SQLite inizia a mostrare i suoi limiti e i normali database relazionali rispondono più agilmente.
I limiti di SQLite sono essenzialmente tre:

- **Concorrenza**: il sistema permette più entità in lettura, ma solo una in scrittura; le operazioni di scrittura bloccano il database.
- **Dimensione del database**: anche se la dimensione massima è di 2 TB, avere database molto grandi costa in termine di occupazione della RAM. Ad esempio, un database di 100 GB richiede l'allocazione di 25 MB di RAM prima di ogni transazione.
- **Networking**: il supporto di rete è assente. Sebbene i database possano essere condivisi su *Networking File System*, le prestazioni con questi sistemi possono essere basse a causa della latenza introdotta da tali file system.

Al di là di questi limiti, che derivano dallo scelte progettuali, e che comunque non sono un problema per il sistema in cui si vuole usare SQLite, questo database non implementa alcune funzionalità:

- **vincoli Foreign Key:** questo tipo di vincoli non sono supportati;
- **supporto completo dei trigger**: mancano alcune caratteristiche dei *trigger*, come il trigger ricorsivo;
- **support completo di ALTER TABLE:** è possibile solamente eseguire operazioni di ridenominazione della tabelle o aggiunta di una colonna;
- **transazioni nidificate**: SQLite permette che solo una query alla volta sia attiva;

- **RIGHT e FULL OUTER join**: questi tipi di join non sono implementati;
- **viste aggiornabili**: le viste sono read-only;
- **GRANT e REVOKE**: questi due comandi non sono implementati; l'unico modo per garantire o negare l'accesso a un database è impostare i permessi di accesso a livello di file system.

In definitiva, SQLite ha tutte le caratteristiche desiderate, tranne una: non è accessibile in remoto. Per risolvere questo problema, è stato utilizzato *uSQLite*.

6.2.2 USQLite

uSQLite [106] è un *network wrapper* per SQLite. Serve a rendere SQLite accessibile in remoto utilizzando un approccio non convenzionale per rendere sia il client che il server molto leggeri, semplici e portabili.
USQLite sfrutta il protocollo *TechFell* [107] per le comunicazioni tra client e server. Inizialmente questo protocollo era stato creato per consentire a dispositivi embedded di accedere direttamente a un database, ma si è poi rivelato utile anche ad altri scopi e con un server appropriato (come *uSQLiteServer*) è possibile usarlo per comunicazione tra processi e sistemi diversi.
In molti casi uSQLite evita il ricorso ad un middleware per la comunicazione, mentre in uno scenario generale permette di costruire applicazioni prive di un layer per l'interfacciamento con il database. Di fatto, permette la creazione di un database embedded distribuito.
In questo modo è stato risolto il problema di accesso remoto remoto al database uSQLite.
Sul mini-pc del gateway è stato dunque installato uSQLiteServer, che è accessibile tramite un client scritto in C fornito insieme a uSQLiteServer. Questo client è stato leggermente modificato per adeguarlo all'applicazione.

6.2.2.1 Ulteriore utilizzo di uSQLite

USQLite rappresenta l'unico canale di comunicazione con la rete WSN. Questo perché l'installazione di un web server o una qualsiasi altra tecnologia per l'accessibilità in remoto avrebbe consumato ulteriori preziose risorse hardware del mini-pc.
Quindi, il canale creato da uSQLite è stato sfruttato anche per scopi che non sono propriamente di un database. Questo “uso improprio” consiste nel salvare i parametri della WSN impostati dall'utente, in una tabella del database. Questi parametri vengono poi letti dal sink periodicamente per eventualmente applicare delle modifiche alla rete di sensori. Al momento, l'unico parametro della WSN impostabile dall'utente è l'intervallo di trasmissione dati.

6.3 WEB SERVER

Il Web Server funge da intermediario tra l'utente e il database del gateway. Quindi il suo compito è di prelevare i dati dal database, elaborarli e mandarli all'utente.
Più precisamente, si occupa di:

- richiedere al database i dati delle rilevazioni più recenti, quelle delle ultime 24 ore o quelle specificate dall'utente;
- salvare nel database il parametro di intervallo di trasmissione dati, le aree di allarme e i dati degli utenti;
- trasformare i dati dei sensori in unità di misura del SI o derivate;
- preparare grafici;
- comunicare con il server di Google Maps;
- generare dati XML ed EEML;
- autenticare gli utenti;
- gestire eventi di allarme (tramite notifica email e aggiornamento della mappa).

6.4 SCELTA DEL WEB SERVER

La scelta del web server è dipesa dai vincoli di leggerezza, compattezza, performance e tipo di licenza. Avere un web server leggero e compatto significa poterlo installare su macchine molto vecchie o addirittura sul mini-pc del gateway, il che vuol dire risparmiare sul costo totale del progetto.
Esistono diverse soluzioni web server, di cui quella open source più famosa è *Apache*. Apache è pero troppo pesante ed ha molte caratteristiche non necessarie allo scopo di questo progetto. Il web server cercato doveva essere in grado di servire dei contenuti semplici occupando poche risorse e compiere le operazioni basilari di un web server.
La ricerca si è quindi ristretta a due soli contendenti: *Lighttpd* [115] e *Nginx* [116]. Questi sono web server open source rinomati per il loro bassissimo consumo di risorse, per la stabilità, per la flessibilità, per la scalabilità e per le prestazioni notevoli, in termine di richieste servite al secondo.
Sono entrambi web server *asincroni*, ovvero sono *event-driven* e gestiscono le richieste in un solo (o al più in pochi) *thread*. Apache invece è *process-based*, cioè per ogni singola richiesta simultanea richiede un thread, che introduce dell'overhead non trascurabile. Questa architettura asincrona ben si adatta all'implementazione in questo progetto, poiché consente prestazioni ottimali in ambito *Web 2.0* (*Ajax*, *Ruby On Rails*, ...), che è poi lo stesso ambito del sistema realizzato in questa tesi.
Le funzionalità offerta da Lighttpd sono le seguenti:

- Bilanciamento del carico FastCGI, SCGI e supporto HTTP-proxy
- Supporto chroot
- Web server basato su select()-/poll()
- Supporto per schemi di notifica eventi più efficienti come kqueue ed epoll
- Riscrittura degli URL in base a condizioni (mod_rewrite)
- Supporto SSL e TLS, via OpenSSL.
- Autenticazione su server LDAP
- Statistiche rrdtool
- Download basato su regole, con possibilità di uno script per il supporto della sola autenticazione
- Supporto SSI
- Hosting virtuale flessibile
- Supporto di moduli
- Cache Meta Language (al momento sostituito da mod_magnet)
- Supporto WebDAV minimale
- Supporto delle servlet (AJP), nelle versioni 1.5.x e superiori
- Compressione HTTP utilizzando mod_compress e il più recente mod_deflate (1.5.x)

Nginx offre simili funzionalità. Tuttavia la scelta non è stata determinata dai servizi offerte dal web server, poiché appunto il progetto non richiede funzionalità particolari. Il fattore più importante da valutare è stato il consumo di risorse hardware, mentre le prestazioni pure non hanno costituito un metro di giudizio. Infatti, entrambi i web server sono in grado di servire un numero impressionante di richieste simultanee: si pensi che Lighttpd viene usato in alcuni server di *youtube.com, wikipedia.org, SourceForge.net, ImageShack.us*, mentre Nginx su altri siti russi ad altissimo traffico. Quindi entrambi possono gestire senza problemi migliaia di accessi contemporanei. Per il web server del progetto si prevede un traffico molto più basso, perciò Nginx e Lighttpd vanno più che bene.
Un ultimo elemento di paragone è stato il consumo delle risorse. Non è stato facile fare confronti sotto questo punto di vista, poiché non ci sono benchmark ufficiali ma solo opinioni di utenti che hanno provato le due soluzioni sul proprio web server. L'ideale sarebbe stato fare delle prove con i due web server nel contesto di utilizzo reale di questo progetto di tesi ma, dato che lo scopo di questa tesi è un altro, ci si è fidati delle opinioni degli utenti trovate su vari siti o blog.
Dal punto di vista di occupazione della RAM e della CPU, diverse fonti [69,70] sostengono che Nginx sia migliore. Molti utenti riportano un utilizzo della CPU minore con Nginx: c'è chi nota 52% contro 98% di Lighttpd e chi 5% contro 10% [71], per esempio. Inoltre sembra che Lighttpd possa soffrire di

memory leak in alcune condizioni, ovvero l'occupazione di RAM continua ad aumentare senza mai scendere. Tuttavia ciò non si è verificato nel contesto di utilizzo di questo progetto di tesi. Inoltre, c'è chi sostiene che Lighttpd abbia dei picchi di CPU al 100% casuali [72]. In definitiva, Nginx sembrerebbe essere il web server più adatto allo scopo.
Ciononostante la scelta è caduta su Lighttpd, per un semplice motivo: la documentazione. Infatti, al tempo, la documentazione di Nginx era piuttosto scarsa e quella originale era in russo, mentre quella di Lighttpd era completa e arricchita da diversi tutorial.
Perciò, come prima implementazione del sistema è stato scelto Lighttpd, mentre Nginx è stato riservato per un eventuale implementazione futura, nel caso in cui Lighttpd non si rilevi all'altezza del compito.

6.5 ARCHITETTURA DEL WEB SERVER

Il web server è costituito da DOJO, PHP, HTML, XML, EEML, Google Maps API e uSQLite Client. Ognuno di questi elementi ha dei compiti specifici che verranno illustrati nei paragrafi seguenti.
Una caratteristica importante del sistema è che è stato sviluppato secondo un approccio AJAX.

6.5.1 PHP

PHP (PHP Hypertext Preprocessor) [108] è un linguaggio di scripting interpretato con licenza open source adatto a sviluppare applicazioni web lato server, anche se può essere usato per scrivere script a linea di comando o applicazioni *standalone* con interfaccia grafica.
Nel sistema realizzato è servito per costruire diverse applicazioni tra cui: gestire il client uSQLite, convertire i dati dei sensori in unità di misura del SI, trasformare i dati in XML ed EEML.

6.5.2 Dojo

Dojo [109] è un JavaScript toolkit open source per la costruzione di applicazioni browser-based. È stato costruito usando JavaScript lato client e serve ad espandere le funzionalità di un browser. Con il suo aiuto è possibile costruire interfacce utente browser-based simili ad applicazioni locali. In questo modo il browser diventa la piattaforma dell'interfaccia utente, indipendentemente da dove si trova il back-end dell'applicazione.
Dojo fornisce componenti che rendono i siti più utilizzabili, funzionali e interattivi. Tra questi componenti vi sono i *widget*, realizzati con codice JavaScript, HTML e CSS, che consentono di creare facilmente menu, tabelle, grafici, effetti animati, form, text editor e altre cose ancora. Dojo è stato implementato nell'architettura proprio con lo scopo di creare grafici, in particolare i grafici che rappresentano l'andamento dei dati rilevati dai sensori.
Il grande pregio di questa tecnologia è di essere potente, comoda da utilizzare e molto leggera. Infatti occupa non più di 100 KB.
Dojo non è l'unico toolkit open source disponibile, ma ce ne sono molti altri, come *Prototype* [111], *JQuery* [112] *e Google Toolkit* [113]. In Tabella 6.1 viene proposto un paragone tra i toolkit più diffusi. Come si può notare Dojo possiede una maggiore collezione di widget e una maggiore facilità di utilizzo rispetto ai concorrenti, e per questo motivo è stato scelto per il progetto.

	Prototype	Dojo	Mochikit	Yahoo!	Google	JQuery
Simple AJAX	✓	✓	✓	✓	✓	✓
Drag n Drop	✓	✓	✓	✓	✓	✓
Basic Visual Effects	✓	✓	✓	✓	✓	✓
Advanced Visual FX	✓	✓	✓	✓	✓	✓
Java integration					✓	
Event handling	✓	✓	✓	✓	✓	✓
Back button support with Ajax		✓			✓	
Developer tools			✓		✓	
Rated Features (0-4 stars)						
Minimal Learning Curve	★★★	★★	?	★★★	★	★★★★
Ease of use (API)	★★★	★★	?	★★★	★	★★★
Widget Collection (useful or not)	★★	★★★★	★★	★★★	★★★★	★
Documentation	★★★	★★	★	★★★★	★	★★
Developer Community	★★★	★★	★	★★★	★★★	★★
Refined UI effect examples	★★★	★★★	★	★★★★	★★★	★★
Filesize Range (KB)	46-137	18-276	5-113	2-300	?	10-44
Licensing	MIT	AFL / BSD	MIT/AFL	BSD	Apache **	MIT

Tabella 6.1: Confronto tra diversi Java Toolkit (fonte:[73])

6.5.3 XML

XML (eXtensible Markup Language) [110] è il metalinguaggio utilizzato per descrivere tutti i dati scambiati tra web server e client. È stato adoperato per due motivi: rendere l'applicazione versatile e facilmente interfacciabile con altre applicazioni e consentire l'approccio AJAX (spiegato nella Sezione 6.5.7).

6.5.4 EEML

EEML [78] è l'acronimo di *Extended Environments Markup Language*, uno standard per la condivisione di dati di sensori tra ambienti remoti, sia virtuali che fisici. Più precisamente, è un markup language che descrive l'output di sensori ed attuatori, basato su uno schema scritto in XML. In questo modo i dati dell'ambiente seguono una struttura sofisticata e possono essere scambiati facilmente con altri software e applicazioni.
Un esempio di documento EEML è il seguente:

```
<eeml xmlns="http://www.eeml.org/xsd/005"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.eeml.org/xsd/005 http://www.eeml.org/xsd/005/005.xsd" version="5">
    <environment updated="2007-05-04T18:13:51.0Z" creator="http://www.haque.co.uk" id="1">
        <title>A Room Somewhere</title>
        <feed>http://www.pachube.com/feeds/1.xml</feed>
        <status>frozen</status>
        <description>This is a room somewhere</description>
        <icon>http://www.roomsomewhere/icon.png</icon>
        <website>http://www.roomsomewhere/</website>
        <email>myemail@roomsomewhere</email>
        <location exposure="indoor" domain="physical" disposition="fixed">
            <name>My Room</name>
            <lat>32.4</lat>
            <lon>22.7</lon>
            <ele>0.2</ele>
        </location>
        <data id="0">
            <tag>temperature</tag>
            <value minValue="23.0" maxValue="48.0">36.2</value>
            <unit symbol="C" type="derivedSI">Celsius</unit>
        </data>
        <data id="1">
            <tag>blush</tag>
            <tag>redness</tag>
            <tag>embarrassment</tag>
            <value minValue="0.0" maxValue="100.0">84.0</value>
            <unit type="contextDependentUnits">blushesPerHour</unit>
        </data>
        <data id="2">
            <tag>length</tag>
            <tag>distance</tag>
            <tag>extension</tag>
            <value minValue="0.0">12.3</value>
            <unit symbol="m" type="basicSI">meter</unit>
        </data>
    </environment>
</eeml>
```

6.5.5 Google Maps API

La *Google Maps API* [114] permette di inglobare Google Maps nelle pagine web con JavaScript. L'API serve a fornire varie utilità per la manipolazione delle mappe e l'aggiunta di oggetti e contenuti alle stesse.
Nella realizzazione dell'interfaccia, la mappa assume un ruolo fondamentale, perché è il mezzo di monitoraggio principale. Su di essa infatti, vengono raffigurati con dei simboli colorati i valori riportati dai sensori e le aree monitorate. In questo modo l'utente ha un interfaccia molto comoda, pratica ed intuitiva che permette un monitoraggio più efficiente, dato che l'utente può vedere immediatamente il valore del dato e a quale zona è riferito.

6.5.6 USQLite Client

USQLite Client è un programma scritto in linguaggio C che spedisce le query a uSQLServer (che si trova sul mini-pc) e riceve le risposte dal database.
Viene eseguito dal web server tramite un applicazione PHP, che si occupa di invocarlo e di trasformare la risposta ricevuta sotto forma di documento XML.

6.5.7 AJAX

L'approccio AJAX ha un ruolo importante nell'applicazione perché ha consentito di renderla real-time,

caratteristica essenziale per un sistema di monitoraggio. Per spiegare i vantaggi di questa tecnologia, è opportuno spiegare il suo funzionamento e le differenze rispetto ad un approccio classico.
Senza AJAX, per creare un'applicazione per browser dinamica si può ricorrere a due soluzioni: utilizzare *Flash, Java applet* o tecnologie simili oppure una combinazione di HTML, PHP e JavaScript. La prima viene spesso scartata perché troppo pesante per un browser oppure perché è dipendente da una libreria specifica. La seconda soluzione è adatta in quei contesti dove si richiede che il browser non venga "appesantito" o necessiti di software particolari. Essa è riassumibile in questo scenario: l'utente invia una richiesta via HTTP al server, il quale prepara la risposta tramite un'applicazione PHP e la spedisce al client sotto forma di HTML e JavaScript. Il problema di questo scenario è che ogni volta che il client necessita di nuovi dati dal server, deve inviare una nuova richiesta HTTP per ricaricare la pagina, bloccando così l'attività dell'utente. AJAX è nato con lo scopo di eliminare questo ricaricamento della pagina.
AJAX è l'acronimo di *Asynchronous JavaScript and XML*. È una sorta di JavaScript potenziato, poiché offre a JavaScript lato client una tecnica per inviare chiamate in background al server e recuperare dati aggiuntivi quando necessario, aggiornando determinate porzioni della pagina senza dover ricaricare la pagina stessa.
AJAX consente di bilanciare il carico di lavoro tra utente e server, permettendo loro di comunicare in background mentre l'utente sta lavorando sulla pagina.
Per citare un esempio, si pensi al form di una pagina web che richiede all'utente di inserire i propri dati come nome, cognome, data di nascita, etc... Questi dati inseriti vanno validati prima di essere elaborati dal server. Senza AJAX, ci sono due metodi per validarli. Nel primo, l'utente compila tutti i campi e clicca sul bottone di "Invia". Il server riceve i dati, li controlla e invia l'esito all'utente. Così facendo però l'utente deve aspettare l'arrivo della risposta e il caricamento di una nuova pagina.
L'alternativa è validare i dati a livello del client tramite JavaScript, ma ciò spesso non è fattibile, perché richiede di inviare troppi dati al client. Se si dovesse validare il nome di una città, bisognerebbe spedirgli un elenco di tutte le città esistenti!
In uno scenario *AJAX-enabled*, invece, l'applicazione web può validare i dati inseriti facendo chiamate al server in background, mentre l'utente continua a compilare gli altri campi del form. Per esempio, non appena l'utente seleziona lo Stato, l'applicazione avvisa il server di controllare se quello Stato è valido, senza interrompere l'utente.
AJAX non richiede alcuna installazione di software, le tecnologie necessari sono già implementate nei browser. Infatti esso è composto da:

- **JavaScript**, l'elemento principale, che permette di costruire funzionalità lato client. Per manipolare parti della pagina HTML si usa il *DOM (Document Object Model*);
- **XMLHttpRequest**, ovvero l'oggetto che permette a JavaScript di effettuare connessioni asincrone al server, così da consentire all'utente di continuare a lavorare. L'accesso al server è effettuato tramite una comune richiesta HTTP di un file o uno script sul server;
- una **tecnologia server-side**, come PHP, per gestire le richieste dal client JavaScript.

La comunicazione tra client e server necessita un metodo per l'inoltro dei dati e la loro interpretazione.
Per l'inoltro dei dati, lo script del client, che effettua una chiamata al server tramite l'oggetto XMLHttpRequest, può mandare coppie attributo-valore con *GET* o *POST*. Queste coppie sono facilmente leggibili da qualsiasi script lato server.
Il server manda indietro la risposta via HTTP, in un formato comprensibile da JavaScript. Questo formato è l'XML. In alternativa ad XML si può usare anche semplice testo o *JSON (JavaScript Object Notation)*. Nella Figura 6.1 e nella Figura 6.2, viene mostrata un confronto tra questa architettura AJAX e un architettura web classica.

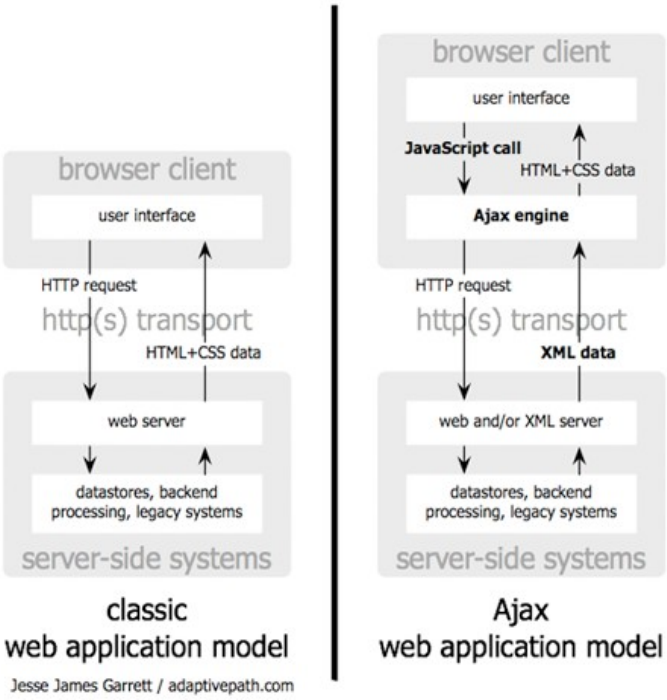


Figura 6.1: Confronto fra l'architettura di un'applicazione web classica e una AJAX-enabled

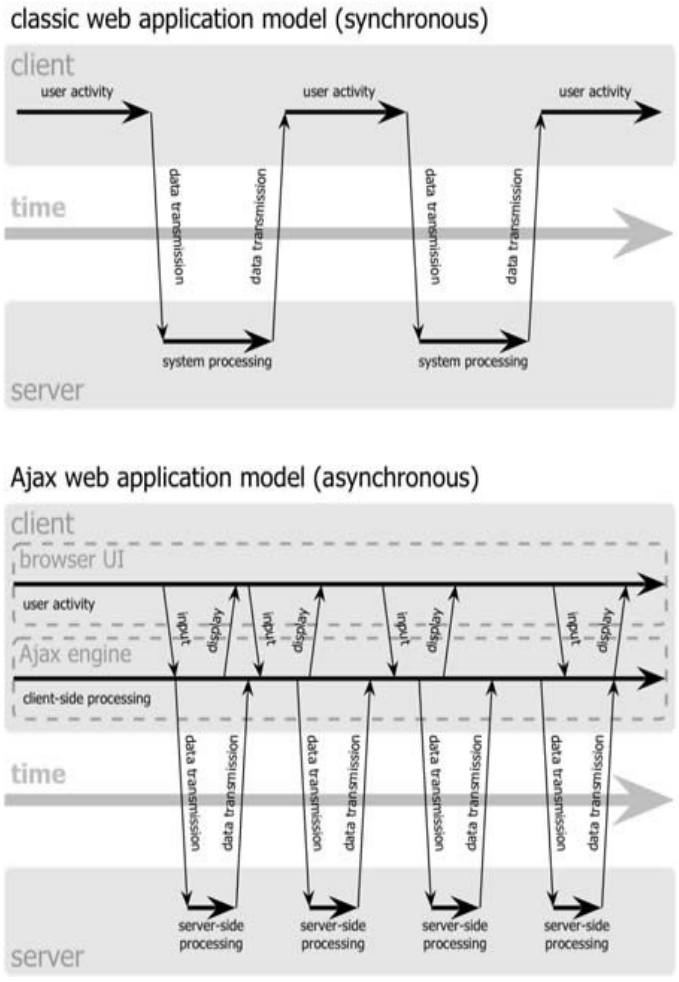


Figura 6.2: Timeline raffigurante il comportamento di un'applicazione web classica e una AJAX-enabled

AJAX, in definitiva, ha molte potenzialità, che permettono ad esempio:

- la creazione di siti e applicazioni web più reattivi ed interattivi;
- l'utilizzo di tecnologie già esistenti;
- l'integrazione con funzionalità presenti nei browser;
- l'utilizzo in ambiti dove sono richiesti aggiornamenti in tempo reale.

Purtroppo però, ci sono anche degli aspetti negativi dell'approccio AJAX:

- il *bookmarking* di una pagina realizzata con AJAX non è possibile, poiché l'indirizzo URL della pagina non cambia durante l'utilizzo;
- i motori di ricerca potrebbero non indicizzare tutte le parti della pagina;
- il bottone di navigazione del browser non funzionano, perché la pagina è sempre la stessa;
- JavaScript può essere disabilitato dall'utente sul proprio browser, rendendo AJAX inutilizzabile.

Per il sistema di questo progetto di tesi, tali aspetti negativi non comportano nessun problema: infatti, il problema del bookmark, oltre a non essere di fatto un problema, può essere in parte risolto creando più pagine web “intermedie”. L'unico aspetto che può creare qualche preoccupazione è l'ultimo. In genere però, le persone che disabilitano Javascript sono poche e lo fanno per scopi particolari. In ogni caso i vantaggi introdotti da AJAX sono troppo importanti per potervi rinunciare.

6.6 FUNZIONI DEL SISTEMA ED INTERFACCIA UTENTE

Le principali funzioni del sistema sono divise in quattro aree: monitoraggio, analisi dati, gestione allarmi e WSN**.** Per ognuna di queste, è stata creata una pagina, selezionabile tramite un *tab* posto in alto nell'interfaccia utente. Grazie a ciò, l'utilizzo dell'applicazione risulta molto intuitivo e viene in parte risolto il difetto di AJAX per cui non è possibile aggiungere ai bookmark una pagina. Se la pagina fosse stata una sola, ad esempio, non sarebbe stato possibile inserire un bookmark per ogni funzionalità.

6.6.1 Monitoraggio

Questa pagina, illustrata in Figura 6.3, è quella principale. L'area più importante è la mappa, che raffigura i nodi della rete con dei simboli, detti *marker*. Tali simboli hanno il colore che rispetta una scala precisa per ogni grandezza, così da poter visualizzare all'istante la situazione generale dell'area monitorata. In caso di allarme, ovvero i valori rilevati sono oltre una soglia specificata, le icone lampeggiano in modo da attirare l'attenzione dell'utente. In tale situazione il colore dell'icona indica se la soglia superata è quella di primo o di secondo livello. Nella mappa viene rappresentata una sola grandezza alla volta, ad esempio temperatura o umidità, ma è possibile cambiarla dal menu in fondo a destra della mappa.
I nodi raffigurati nella mappa sono interattivi, nel senso che è possibile cliccarci sopra. Questo evento provoca la comparsa di un pop-up sopra il nodo che riporta un sommario dei dati del nodo stesso: indirizzo MY e dati dell'ultima rilevazione. Inoltre, vengono creati i grafici di temperatura, umidità e luminosità delle ultime 24 ore, riferiti a quel nodo.
L'area sottostante contiene una tabella, che riporta le ultime rilevazioni ricevute dai nodi. Ogni riga è composta dal numero identificativo del nodo (indirizzo MY), data e ora di ricevimento del dato e i valori di temperatura, umidità e luminosità. Anche in questa tabella è implementata la funzione di notifica visiva di allarme. Infatti, in tal caso, viene colorato lo sfondo della cella riportante il dato oltre la soglia. Per indicare quale soglia è stata superata (la prima o la seconda) e se in misura minore o maggiore, vengono usati colori diversi.
Sotto questa tabella si trova l'area che raffigura i grafici delle rilevazioni delle ultime 24 ore per il nodo selezionato. Questi grafici vengono creati non appena l'utente clicca sul marker di un nodo nella mappa e riguardano tutte e tre le grandezze monitorate.

Infine, in alto a destra della pagine, vi è una semplice area di login. Se un utente (il concetto di utente viene spiegato meglio nella Sezione 6.6.3) registrato si autentica, il sistema sa quali allarmi costui ha impostato ed effettua le notifiche visive di allarme relative. In questo modo, non si creano falsi allarmi. In ogni caso, qualsiasi utente sia autenticato (o se non c'è un utente autenticato), il sistema supporta sempre la funzione di avviso via mail. Tale funzione consiste nel spedire una mail ad un certo indirizzo per notificare un evento di allarme. Se nessun utente registrato è connesso, il sistema considera tutte le aree di allarme impostate.

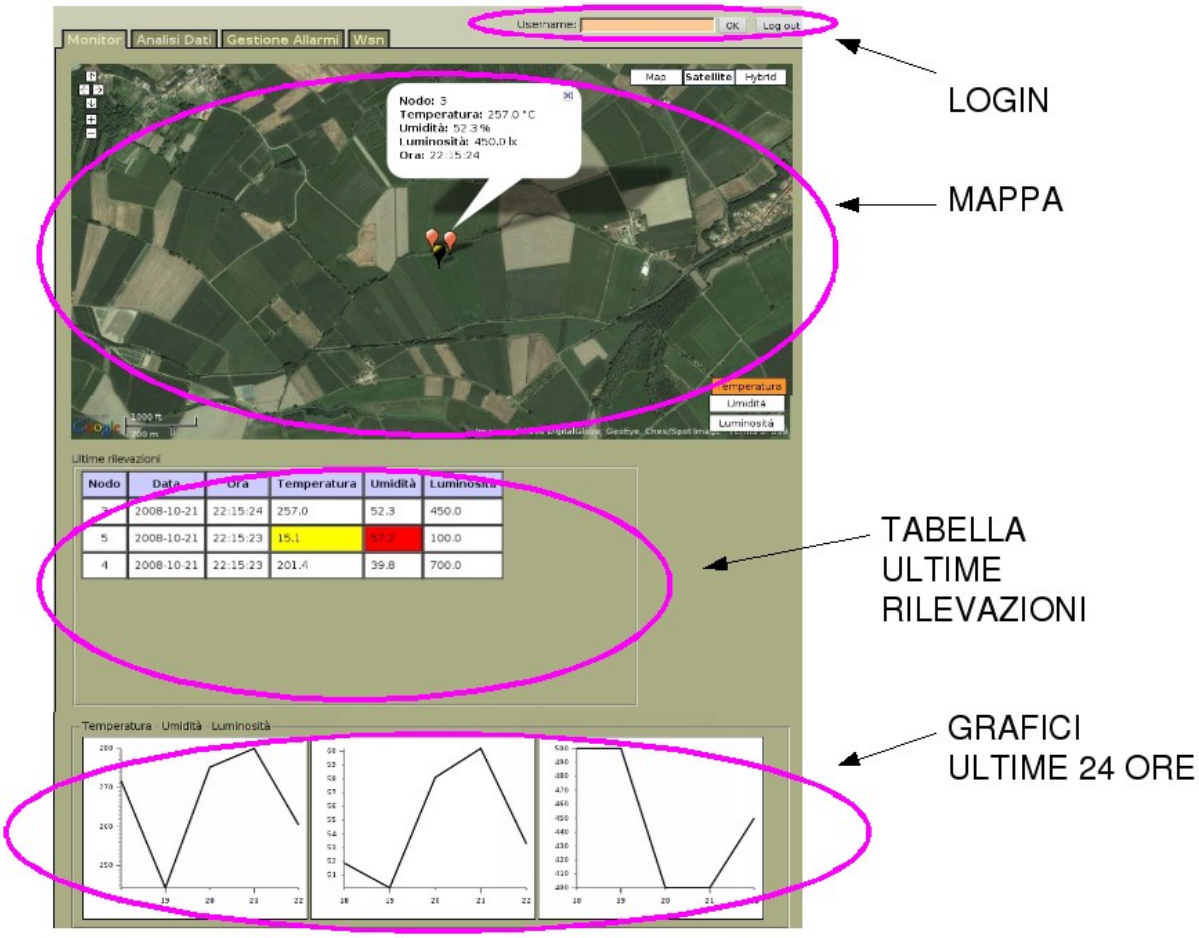


Figura 6.3: Screenshot delle pagina di Monitor

6.6.2 Analisi dati

Questa pagina, riportata in Figura 6.4, consente un'analisi basilare dei dati riportati dai nodi. È divisa in due aree: la prima è un form in cui l'utente imposta i parametri di selezione dei dati, ovvero la grandezza misurata, la data e il numero di nodo. Inoltre vi è la possibilità di richiedere la creazione di un foglio di calcolo con i dati di risposta.
Dopo che l'utente effettua l'invio dei parametri della richiesta, vengono creati, nella seconda area della pagina, i grafici relativi alle grandezze selezionate. Inoltre, se è stato selezionata l'opzione di generazione del foglio di calcolo, l'utente riceve un file in formato *xls* con una tabella dei dati richiesti riportante il numero identificativo del nodo, data e ora, e valori rilevati dai sensori.
In questa pagina non è presente il form di login, poiché non è necessaria alcuna identificazione dell'utente per la funzionalità di analisi dei dati.

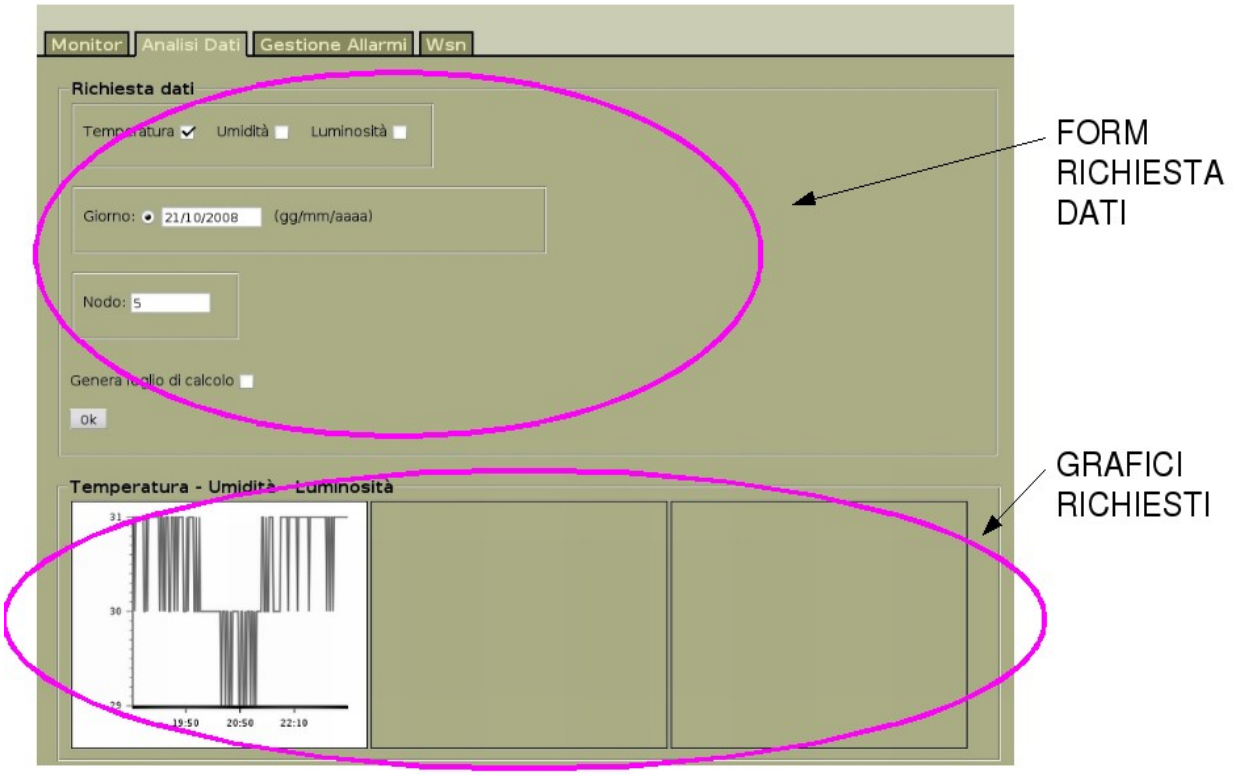


Figura 6.4: Screenshot della pagina Analisi Dati

6.6.3 Gestione allarmi

La pagina di “Gestione allarmi”, di cui è disponibile una screenshot in Figura 6.5, riprende le funzioni e l'interfaccia di “Monitoraggio”, ma aggiunge delle funzionalità e degli elementi. L'obiettivo di questa pagina è fornire all'utente la possibilità di aggiungere delle aree di allarme e impostarne le proprietà.
Le aree di allarme sono delle zone geografiche definite dall'utente, per le quali è possibile impostare delle soglie di allarme sulle grandezze monitorate. Quando il sistema rileva il superamento di tali soglie, aziona le procedure di notifica visiva (marker che lampeggiano sulla mappa, celle della tabella delle rilevazioni che vengono colorate) e notifica via mail.
Per ognuna di queste aree sono configurabili due soglie, dette soglia di primo livello e soglia di secondo livello, per ogni grandezza monitorata. Inoltre, per ciascuna soglia, è possibile impostare il segno (ad esempio, “ x < Soglia 1, x > Soglia 2” oppure “ x > Soglia 1, x > Soglia 2”, etc...).
Ogni area è assegnata ad un responsabile o proprietario (*owner*). Questi responsabili sono poi di fatto gli utente riconosciuti dal sistema in fase di login. Così facendo, quando un responsabile si autentica, il sistema sa quali sono le sue aree di allarme e di conseguenza, aziona gli eventi di notifica solo se il dato oltre soglia proviene da una delle sue aree.
Una lista completa dei responsabili registrati, viene visualizzata in una tabella sotto la mappa. I campi visualizzati sono il nome identificativo dell'utente e il suo indirizzo email (al quale viene spedita la notifica in caso di allarme).
Per definire un'area di allarme, l'utente la disegna sulla mappa. Per disegnarla, deve posizionare i vertici dell'area sulla mappa, facendo un click sul menu a destra della mappa “Nuova area di allarme” e poi un click nella mappa sulle coordinate desiderate, a partire dal vertice a sud-ovest e proseguendo in senso orario. Una volta definiti i vertici, viene raffigurato un poligono colorato che rappresenta l'area appena definita. Cliccando su di esso compare un pop-up contenente un form, dove è possibile impostare le varie soglie e il proprietario. Questi parametri sono modificabili in qualsiasi momento, e vengono visualizzati ogniqualvolta il poligono venga cliccato. Se l'utente inserisce un nuovo responsabile, viene automaticamente aggiunto e visualizzato nella tabella dei

responsabili.
È possibile inoltre definire più aree di allarme sovrapposte: in tal caso, l'ultima area creata determina i parametri di allarme. Questa funzione è comoda quando si vuole che una certa area sia entro certi valori, mentre per una zona ristretta al suo interno valgano altre soglie.
Infine tutte le rimanente funzionalità sono le stesse della pagina di monitoraggio, perciò è possibile utilizzare “Gestione Allarmi” anche per il semplice monitoraggio.

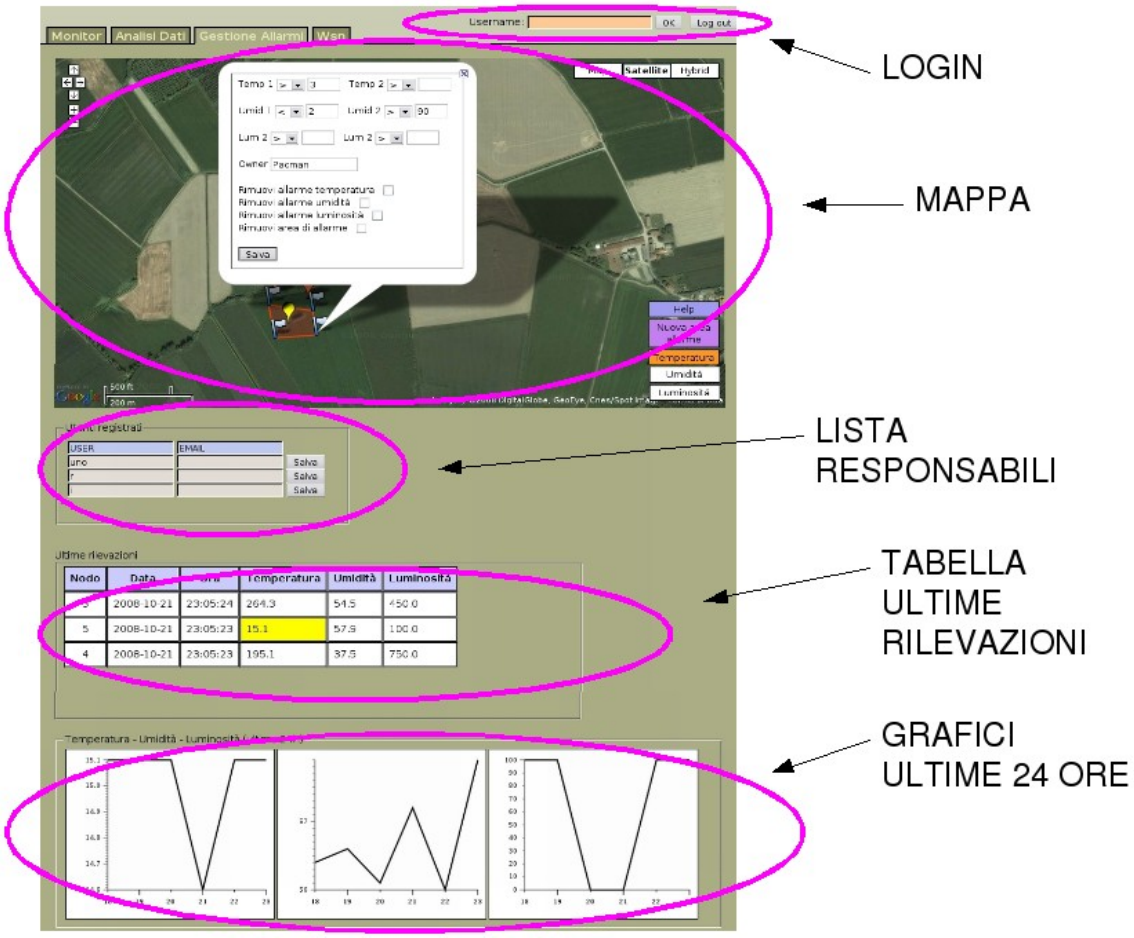


Figura 6.5: Screenshot della pagina Gestione Allarmi

6.6.4 WSN

Questa sezione è dedicata allo visualizzazione dello stato delle rete WSN e all'impostazione di alcuni parametri di setup. L'elemento principale della pagina, come si nota dalla Figura 6.6, è sempre la mappa, che raffigura la struttura della rete: i nodi sono rappresentati sotto forma di marker, mentre i link di connessione tra essi sono rappresentati da delle linee. I nodi che fungono da intermediari o cluster-head hanno un icona con un pallino in centro, per distinguerli dai semplici nodi foglia. Inoltre, il colore del marker, indica se il nodo è attivo o no.
Cliccando su di essi, compare un pop-up che riporta i dati riassuntivi del nodo: indirizzo Serial Low del nodo, indirizzo MY del nodo selezionato, indirizzo MY del nodo padre (o cluster-head) a cui fa riferimento, breve descrizione dello stato del nodo (attivo o non attivo), data e ora di ingresso nella WSN.
Sotto la mappa si trova un form con le impostazioni di setup. Il primo campo visualizza e permette di modificare l'intervallo di rilevamento e trasmissione dati. Il secondo permette di impostare l'indirizzo IP della macchina su cui gira il database server (ovvero l'IP del mini-pc).
Dopo questo form, c'è un link, “Genera EEML”, il quale indirizza ad una pagina PHP che genera e restituisce al client un documento EEML delle ultime rilevazioni.

Infine, l'ultima area è quella dei “Nodi senza coordinate”. Tale area è dinamica, cioè visualizza un form per l'inserimento delle coordinate di un nodo non appena il sistema riconosce un nuovo nodo della rete.
Questa funzione è stata creata per risolvere il problema dell'assenza di GPS. Si presuppone quindi che, in fase di deployment della WSN, un addetto rilevi, tramite un GPS portatile, la posizione dei nodi e utilizzi poi questo form per inserire le coordinate. Per ogni nodo senza coordinate, il form visualizza il suo indirizzo Serial Low (non modificabile), latitudine e longitudine (da impostare). Non appena l'utente compila il modulo, la riga corrispondente sparisce e il nodo viene aggiunto alla mappa.

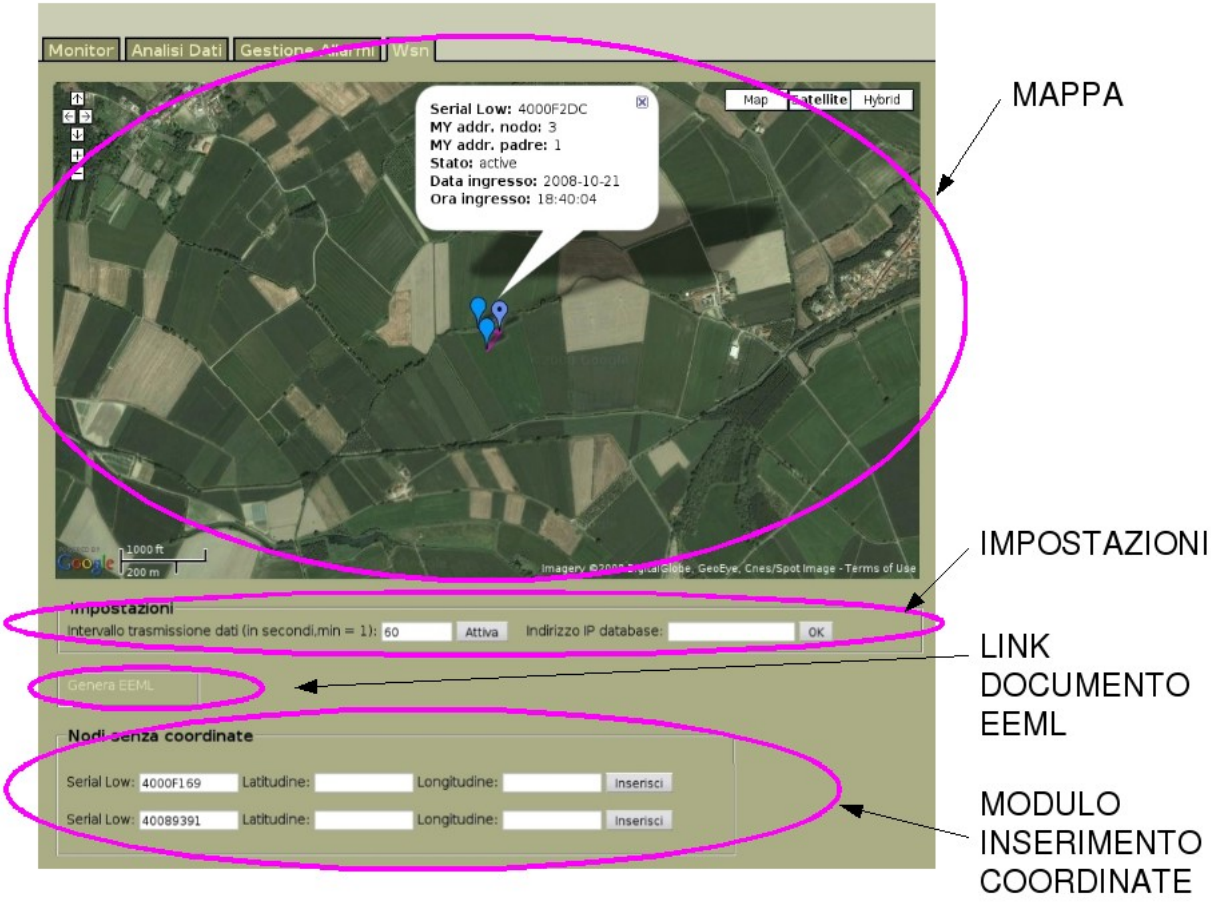


Figura 6.6: Screenshot della pagina WSN

CAPITOLO 7

DEPLOYMENT E ANALISI DEI RISULTATI

7.1 TEST EFFETTUATI

I test effettuati possono essere suddivisi in due fasi: nella prima sono state svolte delle prove per valutare le prestazioni radio dei nodi in diverse situazioni; nella seconda è stato effettuato il deployment vero e proprio ed è stato testato sul campo il comportamento del sistema intero nelle sue condizioni tipiche di utilizzo.

7.2 ANALISI DELLA PROPAGAZIONE RADIO

In questa fase sono stati condotti dei test per meglio capire le capacità trasmissive dei nodi con radio XBee adoperati. Per eseguire queste prove, è stato creato un semplice sistema che esegue un ping dei nodi e registra i tempi di *RTT (Round Trip Time)*.
Nella prima serie di test, i nodi sono stati posizionati in un ambiente chiuso e piccolo (un appartamento, vedasi Figura 7.3), nella seconda in un ambiente interno/esterno (ovvero un nodo dentro un edificio e l'altro fuori) e nella terza sono stati disseminati all'aperto in un campo agricolo. In queste prime tre serie la rete era di tipo single-hop, come illustrato in Figura 7.1 e in Figura 7.2. La struttura multi-hop è stata testata nella quarta e ultima serie, svoltasi in ambiente indoor.
Come nodi sono stati utilizzati gli Squidbee (con antenna omnidirezionale esterna) e dei nodi Arduino con XBee dotato di antenna omnidirezionale integrata (di tipo *chip*), in modo da valutare anche le differenze prestazionali tra queste due tipologie di antenne.

7.2.1 Test indoor

7.2.1.1 Ambiente piccolo

Il programma di ping realizzato spedisce 100 pacchetti, in formato "| a | n.sequenza | 34 | 97 | 21 | @".
Alla fine di ogni messaggio vi è "@", in modo da far capire al ricevente che il messaggio è terminato, mentre "a" è un semplice identificativo del messaggio.
Il nodo destinatario legge il contenuto del messaggio e spedisce ciò che ha ricevuto al mittente.
Quest'ultimo a sua volta controlla che il pacchetto ricevuto sia uguale a quello spedito e in caso affermativo salva in due file di testo il tempo impiegato: in uno vi sono registrati i tempi assoluti di spedizione e ricezione del pacchetto, mentre nell'altro vi è il Round Trip Time (RTT) in msec di ogni pacchetto.

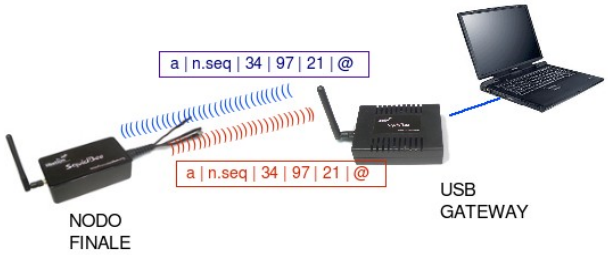


Figura 7.1 Configurazione del test 1 hop con nodo Squidbee

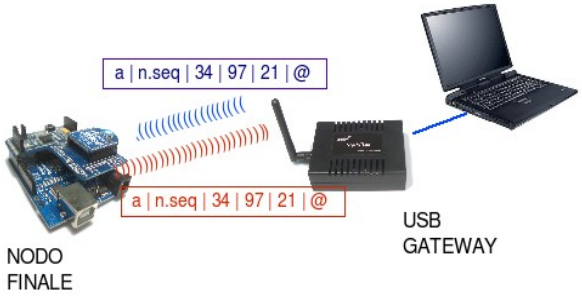


Figura 7.2: Configurazione del test 1 hop con nodo Arduino+XBee (antenna integrata)

Il pc con l'USB gateway è stato posizionato nella stanza A, mentre la posizione del nodo ricevente era in A, B o C a seconda del test.
Come nodi riceventi sono stati utilizzati: uno Squidbee con antenna omni e un Arduino con XBee Shield.

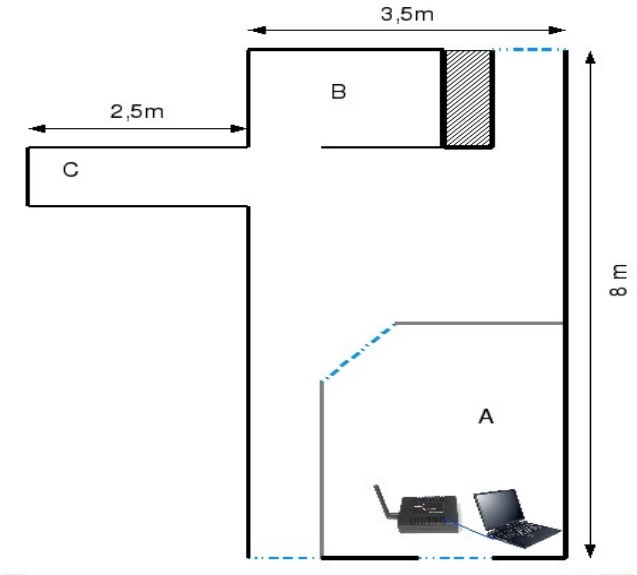


Figura 7.3: Pianta dell'ambiente di test

Il primo test (Figura 7.4 e Figura 7.5) riporta i risultati del ping al variare della posizione del nodo ricevente (Squidbee).

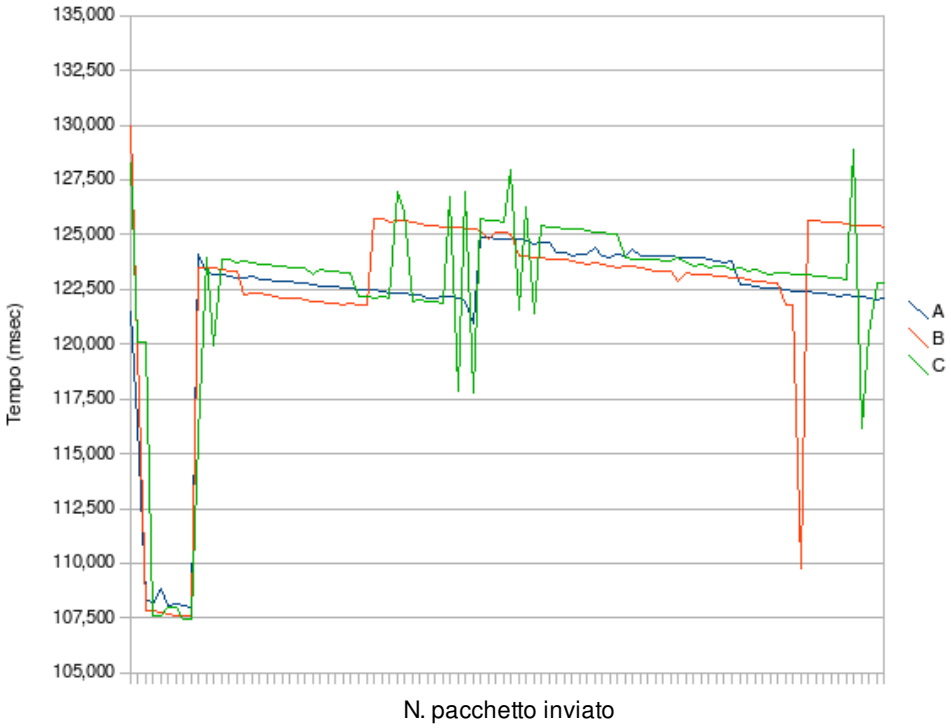


Figura 7.4: Valore del tempo di ping con antenna omni

Come si può notare, non ci sono particolari differenze tra le varie posizioni del nodo ricevente.
Stesso discorso usando l'Arduino con l'antenna chip. Il fatto che si alternino momenti a più basso ping con momenti a maggiore ping, è tipico con questi sistemi radio e sarà un comportamento ancora più accentuato nei test outdoor o in ambienti chiusi più grandi.

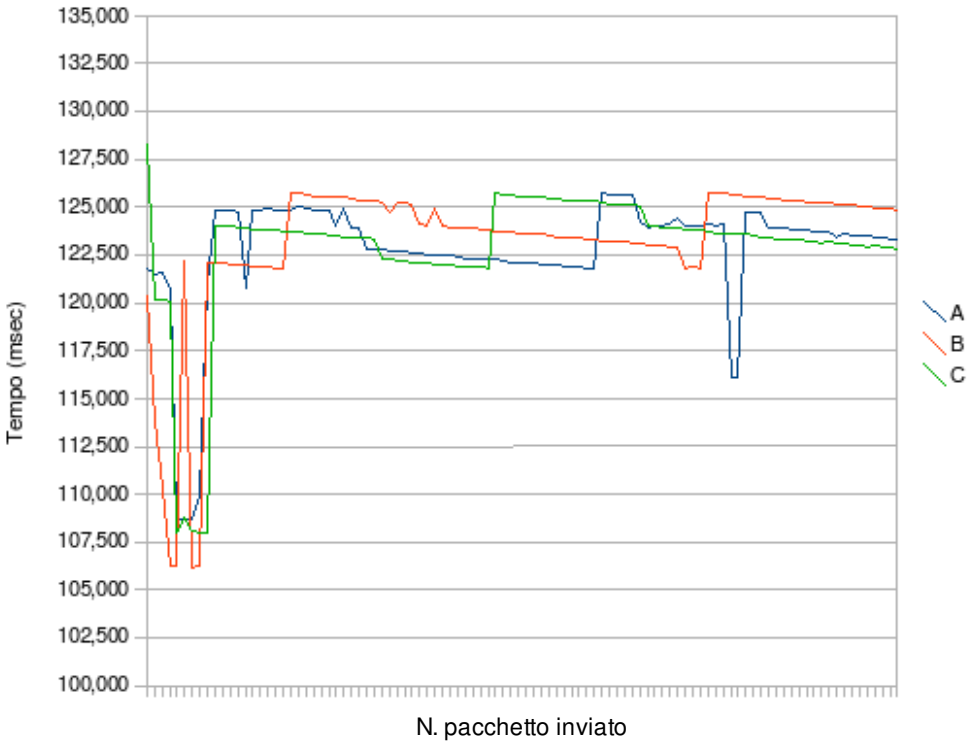


Figura 7.5: Valore del tempo di ping con antenna integrata

Confrontando direttamente le due soluzioni (Figura 7.6), si nota che l'antenna omni esterna consente migliori prestazioni, anche se di pochissimo (i valori di ping indicati sono la media dei tempi di ping dei pacchetti ricevuti). In nessuna prova si è verificata perdita di pacchetti.

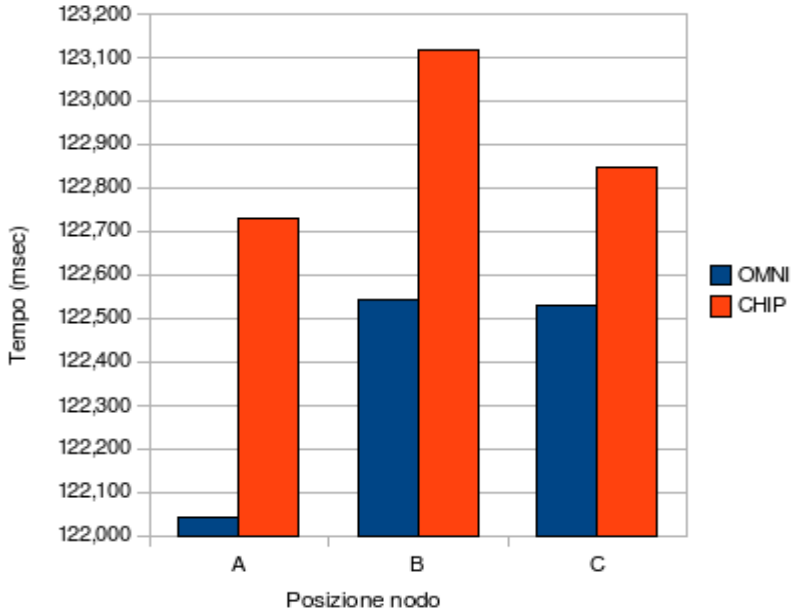


Figura 7.6: Tempo medio di ping - Confronto diretto tra antenna omni e chip

Sono poi stati fatti dei test (Figura 7.7) per verificare quanto tempo si perde nell'impostare il destinatario prima di ogni invio di pacchetto. In una rete a due nodi, non ha senso impostare ogni volta il destinatario ma, quando la rete aumenterà di dimensione, ciò sarà necessario. Tuttavia anche il tal caso, il comando di cambio destinatario avverrà solo quando opportuno, e non ad ogni invio di messaggio.

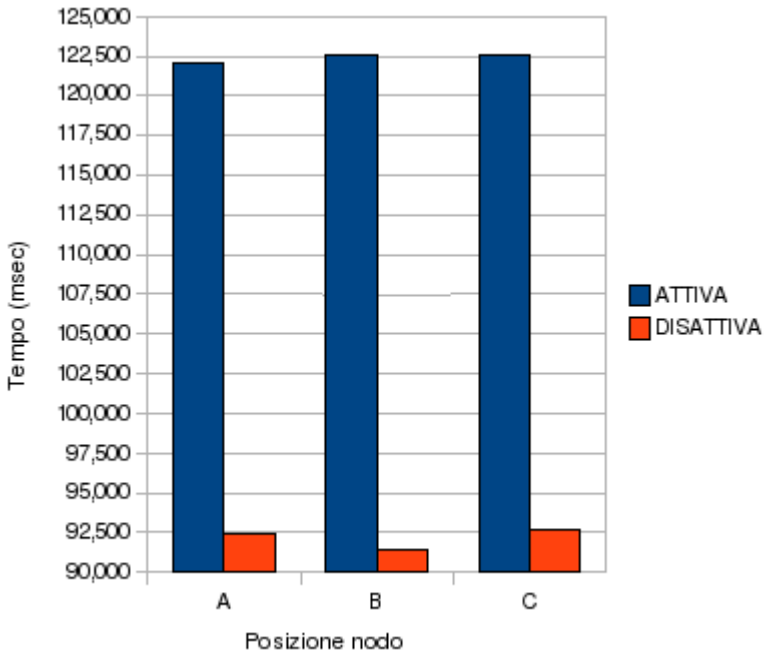


Figura 7.7: Tempo medio di ping - Influenza sulle prestazioni del meccanismo di impostazione destinatario

Per eseguire la procedura di impostazione del destinatario, occorrono 12 ms; ciononostante, i risultati ottenuti dimostrano che il tempo perso è maggiore, intorno ai 25-30 ms.

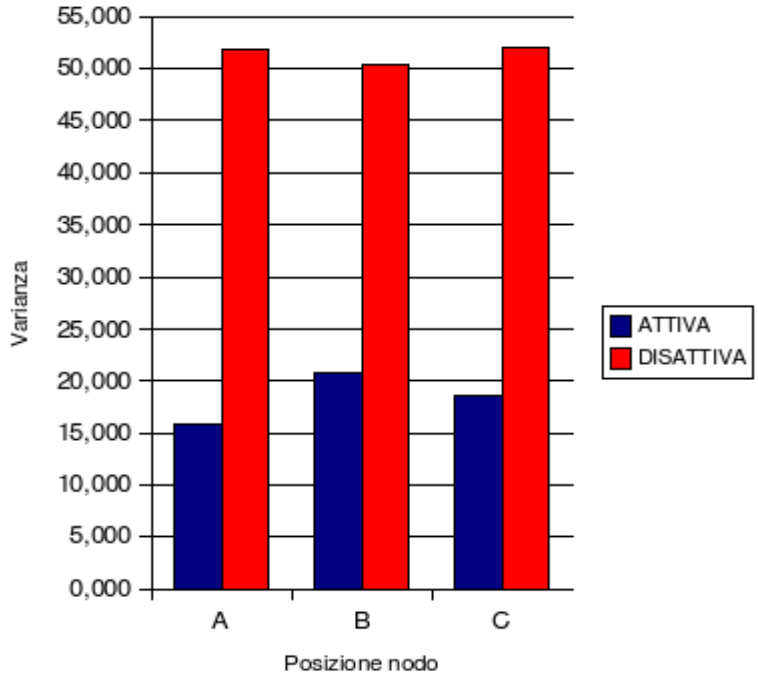


Figura 7.8: Varianza del tempo medio di ping - Influenza sulle prestazioni del meccanismo di impostazione destinatario

L'assenza dell'impostazione del destinatario è caratterizzata da un'elevata varianza (Figura 7.8).

7.2.1.2 Ambiente grande

Questi test sono stati effettuati al piano H del dipartimento di Telecomunicazioni. Qui di seguito è riportato una bozza della pianta della struttura (Figura 7.9) e una foto (Figura 7.10).

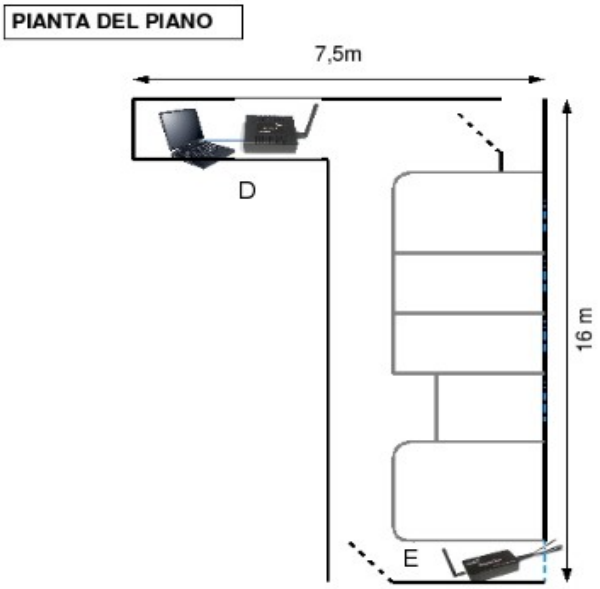


Figura 7.9: Pianta della struttura in cui sono stati svolti i test in ambiente interno vasto

Figura 7.10: Test Indoor - i paletti verdi indicano dove è stato posizionato il nodo gateway

Il nodo gateway è posizionato su una scrivania vicino ad un grosso armadio di metallo. I punti di posizionamento indicati con D1 e D2 differiscono soltanto di 15 cm (D2 posizionato più a sinistra e sollevato di un qualche centimetro). Tuttavia le prestazioni ottenute sono molto diverse, come si potrà osservare fra poco. Il nodo finale è invece posizionato per terra (E1) oppure sopra un calorifero spento a circa 80 cm di altezza (E2), a seconda del test.
Per quanto riguarda i risultati, è importante notare da subito un aspetto: se anche solo un nodo è posizionato per terra**,** su una qualsiasi superficie, le prestazioni sono pessime, tali addirittura da non consentire la connessione in molti casi. Ciò è evidente nei grafici di Figura 7.11 e di Figura 7.12, in cui vengono mostrati i risultati usando sia un nodo con antenna esterna omni sia un nodo con antenna chip. Il gateway è sempre dotato di antenna esterna omni.

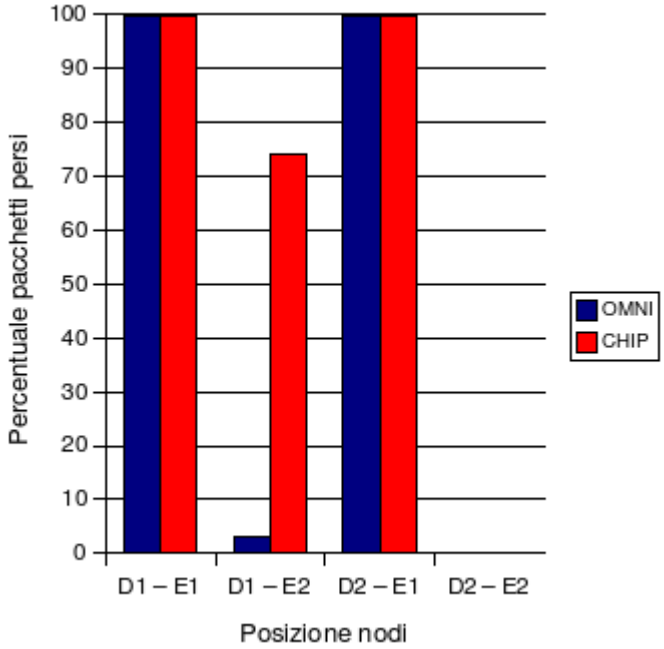


Figura 7.11: Test Indoor - Ambiente grande - Perdita pacchetti

La perdita dei pacchetti è totale quando il nodo ricevente è per terra. Ciò potrebbe essere dovuto al fenomeno del multipath e quindi alla presenza di interferenze distruttive.
Tuttavia anche il test E2, in cui il nodo è sollevato, l'antenna chip non consente trasmissioni accettabili; è bastato però spostare di 15 cm la stazione (in D2) per poter aver di nuovo un link stabile. Con l'antenna omni esterna, questo problema non si è registrato: sia in D1 che in D2 il link era buono, come confermato dai valori di ping nel grafico di Figura 7.12.

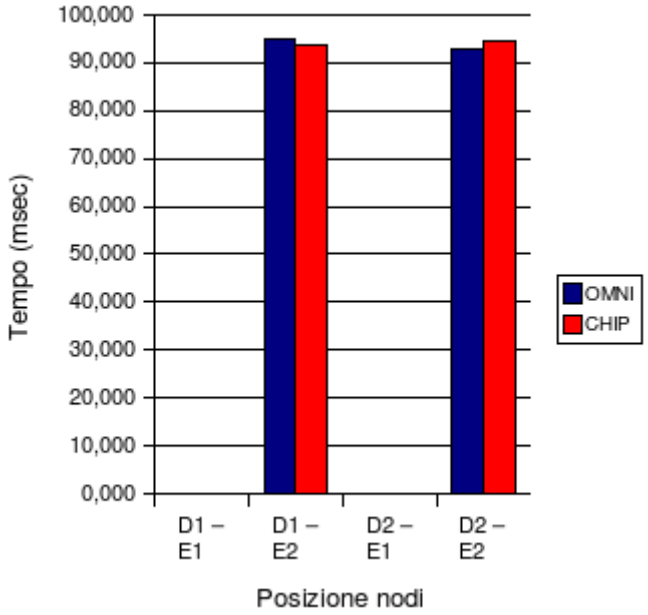


Figura 7.12: Test Indoor - Ambiente grande – Tempo medio ping

La prima e la terza colonna del grafico di Figura 7.12 sono vuote, perché tutti i pacchetti sono andati persi.
La varianza invece è molto alta quando il gateway si trova in posizione D2 e l'end node si trova in posizione E2 (Figura 7.13).

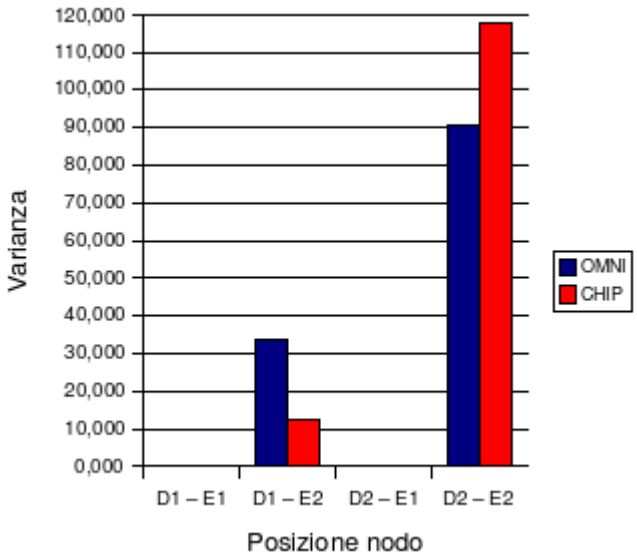


Figura 7.13: Test Indoor - Ambiente grande – Varianza del tempo di ping

Infatti se si osserva, ad esempio, l'andamento del ping nell'arco di una sessione, riportato in Figura 7.14, si nota l'alternarsi di momenti a minore e maggiore ritardo, con differenze di ben 14 ms.

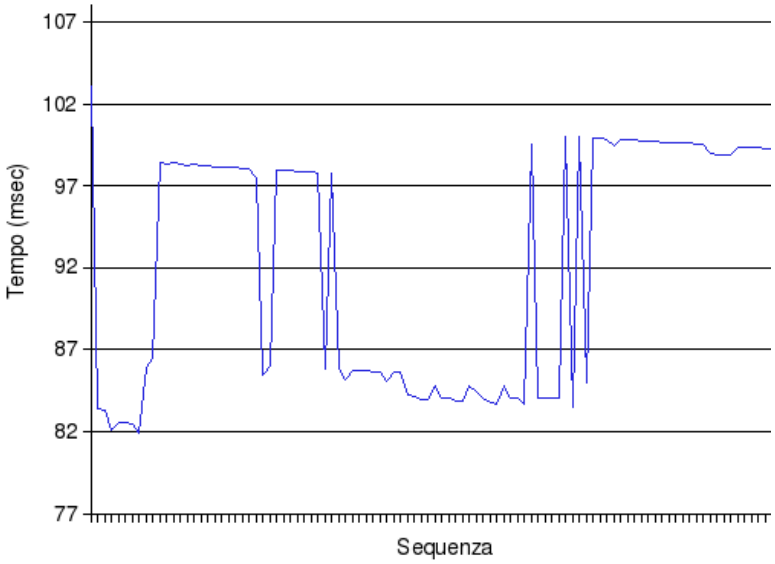


Figura 7.14: Andamento del tempo di ping nel test "D2 - E2"

7.2.2 Test outdoor

7.2.2.1 Ambiente non complesso

Questa prima serie di test outdoor è stata effettuata sul tetto all'esterno del piano H del dipartimento di Telecomunicazioni.
L'end node è stato posizionato su una sedia nel punto X (vedasi Figura 7.15), mentre il nodo gateway è stato messo in due punti Line of Sight (LOS), ovvero in A e B, e poi in 2 punti non-LOS (C ed E), nascosto dalle pareti dell'edificio. Il gateway è stato dotato dell'antenna omni esterna, mentre il nodo end cambiava tra chip ed omni esterna in base al test.
I termini che si trovano nei grafici indicano se il nodo gateway si trova per terra (caso 1) oppure a circa 120 cm dal suolo (caso 2).

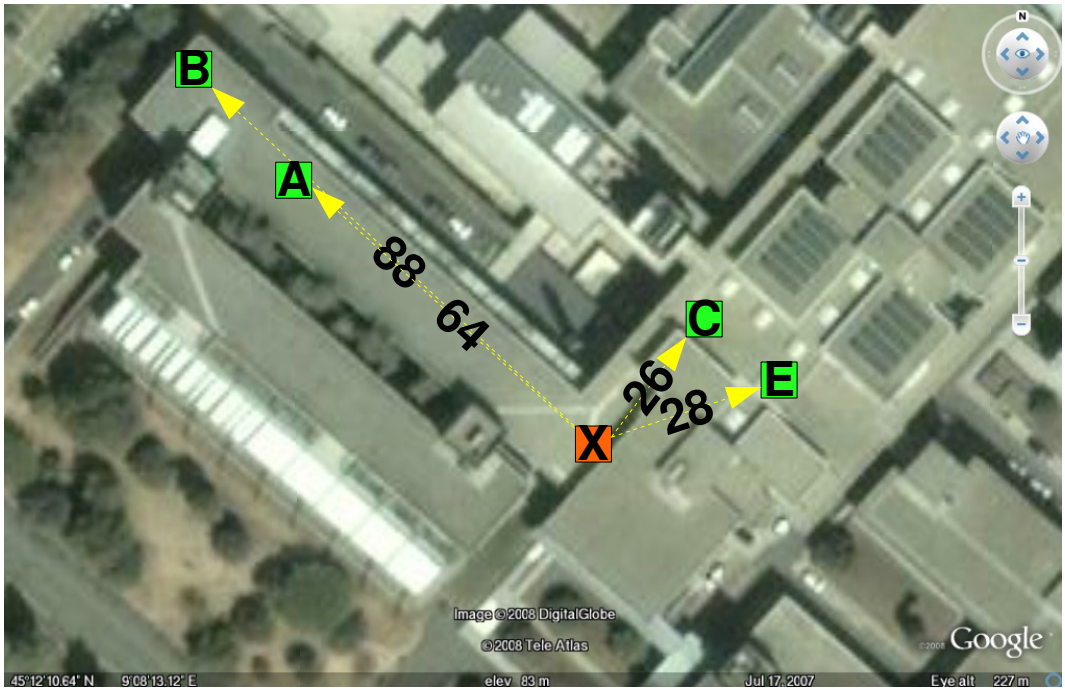


Figura 7.15: Configurazione del test outdoor. I numeri indicano la distanza in metri

Nei grafici in Figura 7.16, Figura 7.17 e Figura 7.18 vengono riportati i risultati conseguiti.

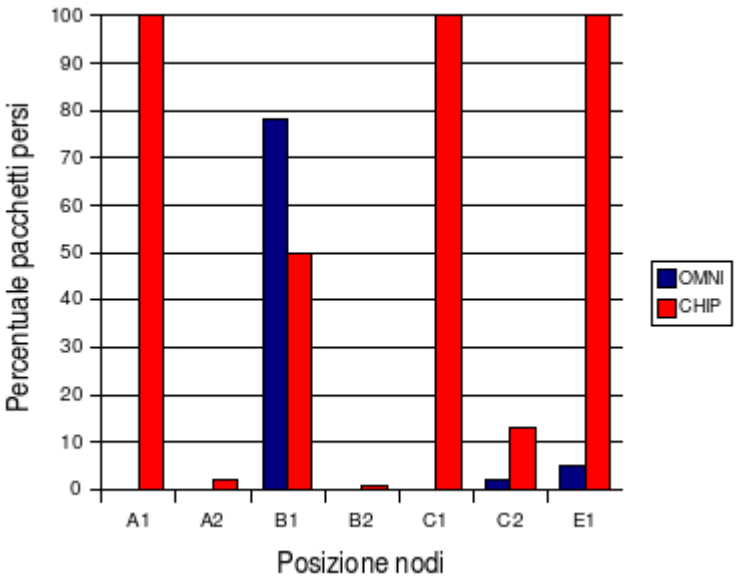


Figura 7.16: Test Outdoor - Ambiente grande – Pacchetti persi

Come già evidenziato, posizionare un nodo con antenna chip per terra significa praticamente eliminare il link, indipendentemente dal tipo di terreno sottostante. Anche con antenna esterna il problema non è da sottovalutare, come si può vedere dal test “B1” in cui il nodo gateway si trova a circa 80 m LOS dall'end node, la perdita pacchetti è troppo elevata e quindi il link molto fragile.
Per quanto riguarda le prestazioni in riferimento al posizionamento del nodo, l'antenna esterna omni garantisce chiaramente una migliore copertura, come dimostra il fatto che in situazione non-LOS estrema (punto E) o LOS a lunga distanza (punto B) l'antenna Chip non riceve il segnale (caso test E) o comunque lo riceve poco (caso test B). Sempre nei casi B ed E, si registrano inoltre i tempi di ping più lunghi, rispettivamente 98 ms e 96 ms, contro i 92-93 ms di tutti gli altri casi.
L'antenna Chip ha comportato tempi maggiori di 1-3 ms nei vari test rispetto a quella esterna, tuttavia il suo tempo di ping è caratterizzata da una varianza minore.

La varianza molto alta del test “C2” è stata causata da qualche pacchetto arrivato con molto ritardo. Di seguito vengono riportati i grafici di ping (Figura 7.17) e varianza (Figura 7.18).

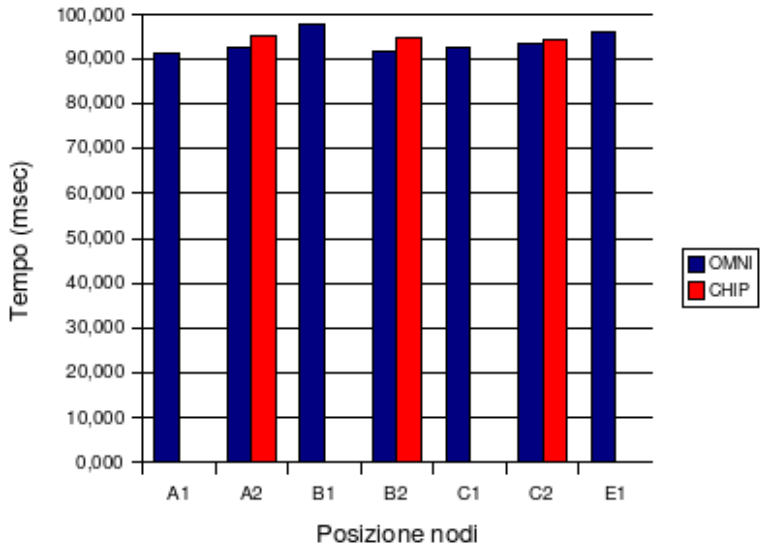


Figura 7.17: Test Outdoor - Ambiente Grande – Tempo medio di ping

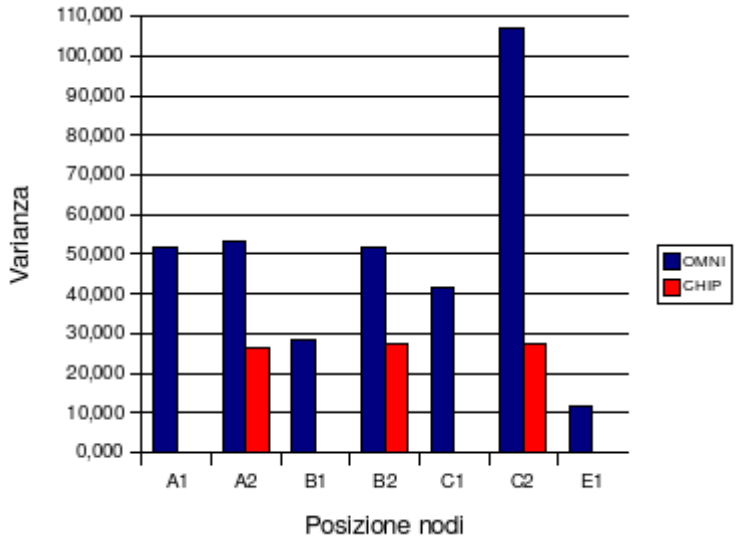


Figura 7.18: Test Outdoor - Ambiente Grande - Varianza del tempo di ping

7.2.2.2 Campo agricolo

Questi test sono stati eseguiti in un parco, più precisamente in un campo di piccole piante e in un boschetto. Le foto di Figura 7.19 e Figura 7.20 illustrano l'ambiente dei test.

Figura 7.19: Il campo dei test, con il nodo in posizione rialzata


Figura 7.20: Il nodo finale a 50 cm dal suolo

L'obiettivo dei test, è stato quello di capire qual è il massimo raggio di copertura della rete wireless, in che modo è meglio posizionare i nodi e quanto influisce la presenza di alberi e foglie in un concreto ambito di utilizzo di una WSN.
È stato scelto di adoperare solo antenne omni esterne in quanto, a parità di costo e con delle dimensioni molto contenute, consentono performance superiori, soprattutto nelle situazioni più critiche.
Sono state svolte quattro serie di test che differiscono per la posizione del nodo finale, indicato nella Figura seguente con “X” in arancione. Le locazioni del nodo gateway sono invece rappresentate dalle altre lettere con sfondo verde.
Per capire il miglior posizionamento dei nodi, sono state provate quattro configurazioni di altezza da

terra: per terra, a 50 cm dal suolo, a 120 cm, su un palo di legno di 2 m ai lati del campo, su un palo di legno sopraelevato rispetto al campo di circa 3 m.
Nella prima sessione di prove il nodo finale si trova ad una estremità del campo (punto X), per terra o a 50cm, a seconda del test (vedasi Figura 7.21).

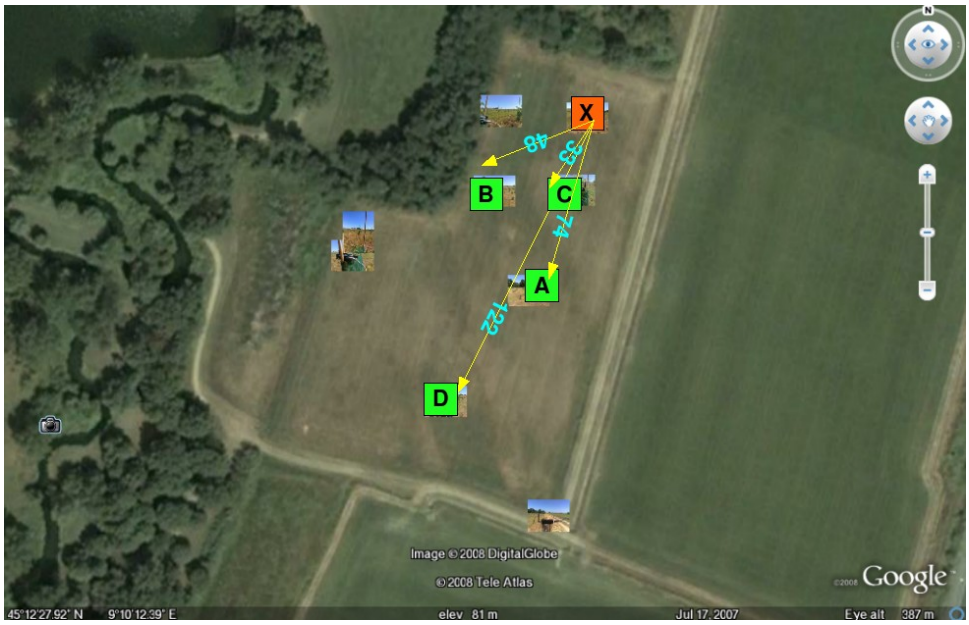


Figura 7.21: Configurazione della prima sessione di test: nodo nel campo

Nella seconda sessione, in nodo finale si trova rialzato rispetto al campo di circa 3 metri, su un palo di legno (vedasi Figura 7.22).

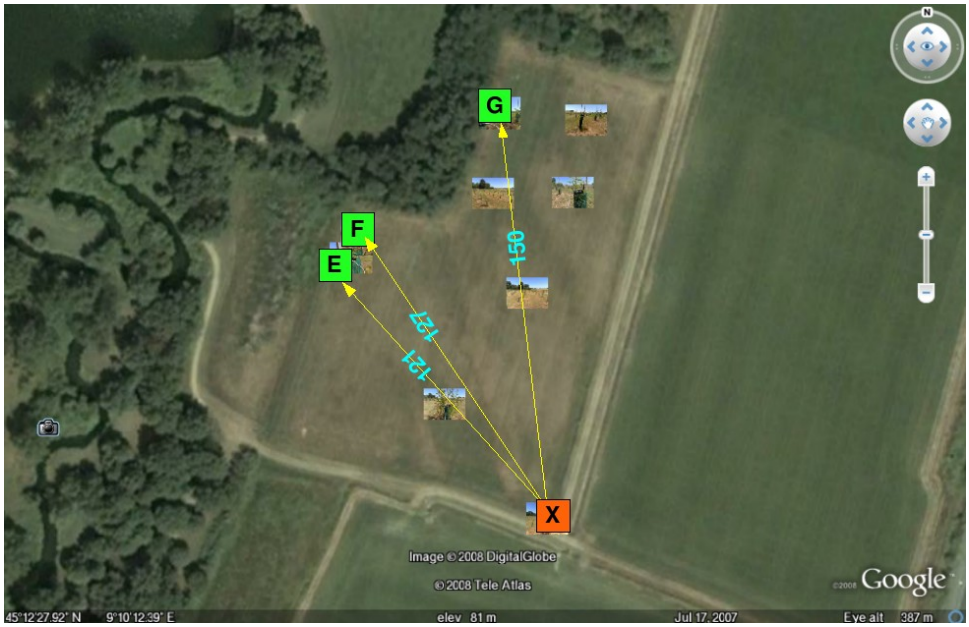


Figura 7.22: Configurazione della seconda sessione di test: nodo a 3 m dal suolo

Nella terza sessione, il nodo è stato spostato su un palo alto 2 metri rispetto al suolo del campo (fare riferimento alla Figura 7.23).

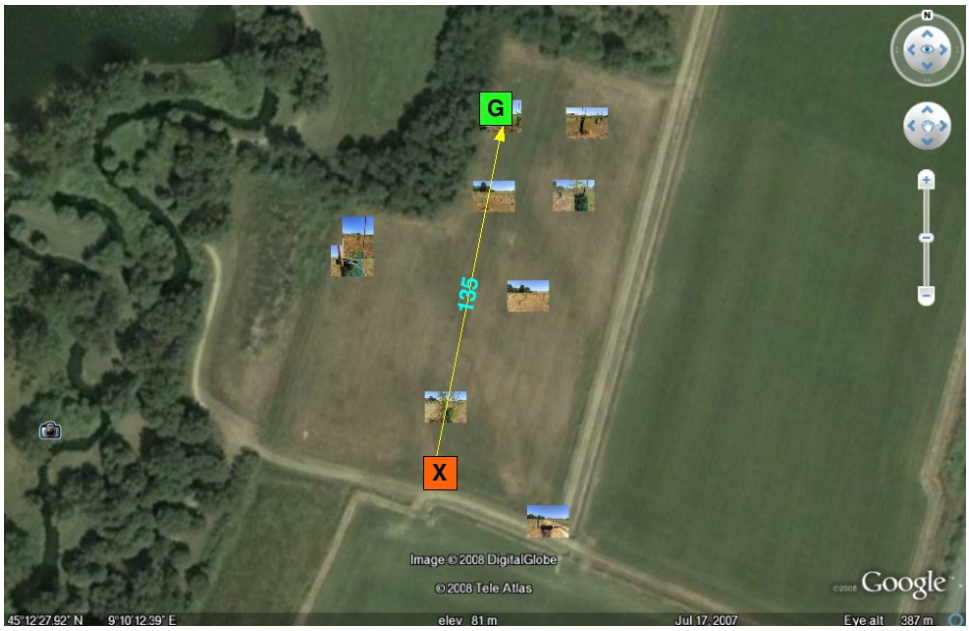


Figura 7.23: Configurazione della terza sessione di test: nodo a 2 m dal suolo

I risultati dei test svolti nel campo, riportati in Tabella 7.1 e in Tabella 7.2, confermano ancora una volta che i nodi per terra non permettono la creazione di buoni link, sebbene un nodo posizionato in alto possa migliorare la situazione. Quindi, mettendo entrambi i nodi per terra, il link è inesistente (vedasi i test in A, B, C), ma se un nodo viene posizionato in alto, allora il link diventa affidabile, con qualche perdita di pacchetto (minore del 5%).

	0 m	1.2 m
A	100	3
B	100	0
C	100	-

Tabella 7.1: Percentuale di pacchetti persi al variare dell'altezza del gateway e della posizione, con l'end node collocato per terra

	0 m	0.5 m	2 m	3.0 m
C	0	-	-	-
D	-	1	-	-
E	-	-	-	0
F	-	-	-	0
G	-	-	2	0

Tabella 7.2: Percentuale di pacchetti persi al variare dell'altezza del nodo finale e della posizione, con il gateway collocato a 50 cm di altezza

In “D” i due nodi sono ad una grande distanza (122 m), ma sono a 50 cm dal suolo. È possibile andare oltre questa distanza, ma in tal caso un nodo deve essere posto più in alto. Infatti il gateway posizionato in “G” a 2 metri di altezza, dove la distanza tra i nodi è di 135 metri, si perde il 2% dei pacchetti, mentre con il gateway nello stesso punto ma a 3 metri di altezza, distante 150 metri dal nodo finale, non si verificano perdite di pacchetto. Ciò grazie al fatto che un nodo si trova in posizione rialzata rispetto a tutto il campo. I tempi di ping, riportati nella Tabella 7.3 e nella Tabella 7.4, sono compresi tra i 92 e 96 msec e, più che dalla distanza, dipendono dall'altezza e quindi dalla “visibilità” tra i nodi stessi, nonostante la vegetazione presente non sia fitta, anzi piuttosto rada.

	0 m	1.2 m
A	-	95.669
B	-	91.966
C	-	-

Tabella 7.3: Tempo medio di ping al variare dell'altezza del gateway e della posizione, con l'end node collocato per terra

	0 m	0.5 m	2 m	3.0 m
C	91.543	-	-	-
D	-	95.033	-	-
E	-	-	-	94.211
F	-	-	-	92.752
G	-	-	94.075	93.369

Tabella 7.4: Tempo medio di ping al variare dell'altezza del nodo finale e della posizione, con il gateway collocato a 50 cm di altezza

Nella Tabella 7.5 e nella Tabella 7.6 viene riportata la varianza registrata, che dipende sia dall'altezza di posizionamento dei nodi sia dalla distanza reciproca.

	0 m	1.2 m
A	-	22.267
B	-	51.193
C	-	-

Tabella 7.5: Varianza del tempo medio di ping al variare dell'altezza del gateway e della posizione, con l'end node collocato per terra

	0 m	0.5 m	2 m	3.0 m
C	51.583	-	-	-
D	-	30.267	-	-
E	-	-	-	48.908
F	-	-	-	53.809
G	-	-	40.613	68.596

Tabella 7.6: Varianza del tempo medio di ping al variare dell'altezza del nodo finale e della posizione, con il gateway collocato a 50 cm di altezza

Nel grafico di Figura 7.24, viene riassunta la differenza tra nodo per terra e nodo rialzato.

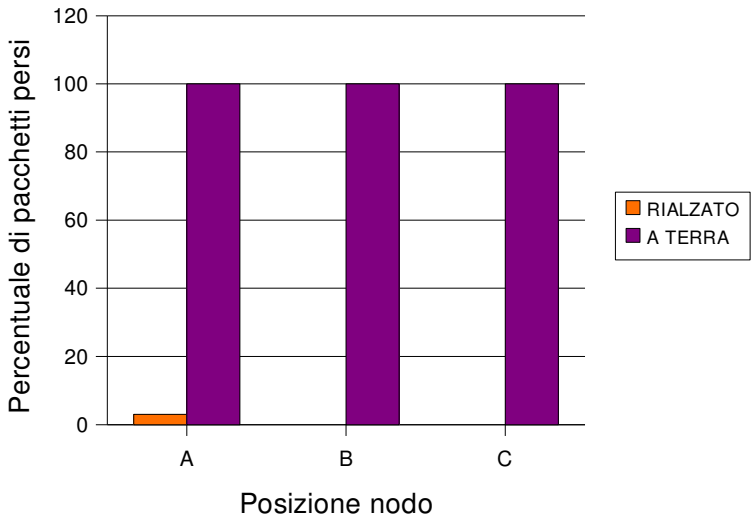


Figura 7.24: Differenza tra nodo a terra e nodo rialzato, espressa in termini di pacchetti persi

Le prestazioni del link non risentono molto della distanza, in questi test in cui l'ambiente non è così ostile. Infatti nel grafico di Figura 7.25, che mette in relazione distanza tra nodi e tempi di ping, si nota come quest'ultimo rimanga praticamente costante all'aumentare della distanza. Inoltre, cosa ancora più importante, non si verifica perdita di pacchetti, o comunque solo in pochi casi e solo del 2% circa.

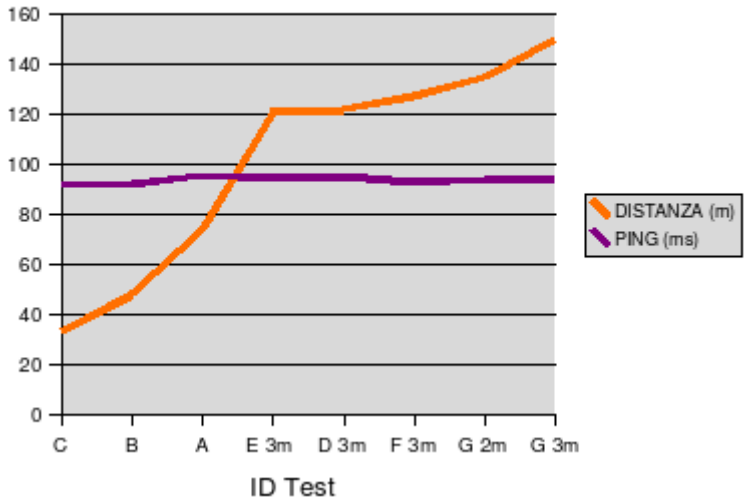


Figura 7.25: Andamento del tempo medio di ping in rapporto alla distanza dei nodi

La quarta serie di test è stata effettuata in un piccolo bosco, con il nodo end fissato su un ramo vicino al tronco, a circa a 180 cm, e il gateway a 120 cm (non su una pianta). In Figura 7.26 viene illustrata la configurazione del test.

Figura 7.26: Configurazione quarta serie di test: nodi tra gli alberi

In questo caso, la presenza di foglie e rami (vedasi Figura 7.27) impatta molto negativamente le prestazioni; inoltre il vento, muovendo essi, causa un link molto variabile: ad esempio, una folata di vento e, quindi, lo spostamento delle foglie e dei rami, può modificare la “visibilità” tra i nodi e causare perdita di pacchetti.

Figura 7.27: Ambientazione della quarta serie di test

Dal grafico di Figura 7.28, si nota un'alta varianza dei tempi di ping, probabilmente dovuta appunto alla natura molto mobile degli ostacoli presenti.
I tempi di ping sono in linea con quelli ottenuti dai test nel campo, mentre la perdita di pacchetti è significativa già a 70 metri circa (punto H), mentre a 50 metri (punto I) tutti i pacchetti arrivano a destinazione.

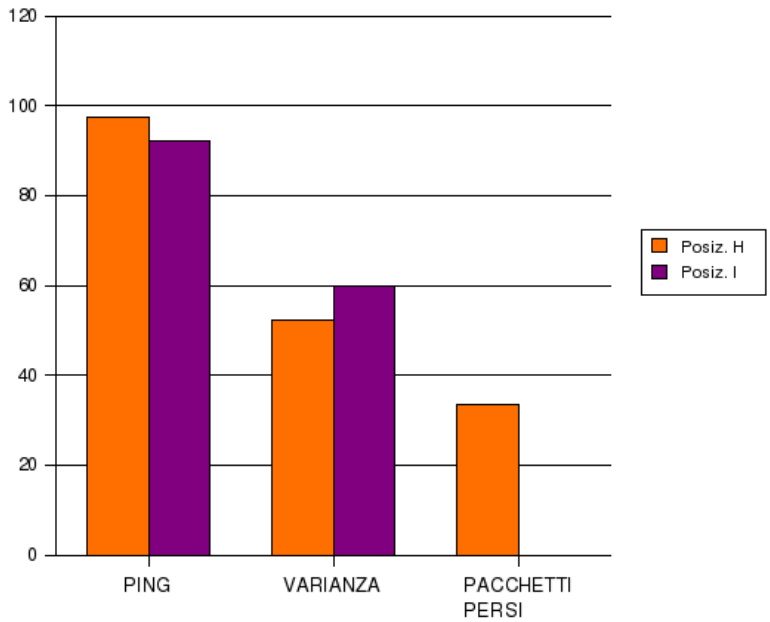


Figura 7.28: Tempo medio di ping (msec), varianza del tempo medio di ping e percentuale di pacchetti persi della quarta serie di test

Infine, l'ultimo test effettuato ha l'obiettivo di stabilire la massima distanza tra nodi raggiungibile, nella situazione migliore possibile, ritratta in Figura 7.29 (nodo finale nel punto X e gateway nel punto M).

Figura 7.29: Massima distanza raggiunta tra due nodi

Figura 7.30: Posizione dei due nodi nel test di massima distanza

Il nodo end è stato posizionato a 2 metri di altezza su un palo di legno, come si nota in Figura 7.30, poi il nodo gateway è stato allontanato fino al punto oltre il quale il link non consentiva la consegna di almeno il 90% dei pacchetti. La traiettoria tra i 2 nodi non presenta ostacoli e il nodo gateway è stato tenuto a 120 cm di altezza circa.
La distanza massima così raggiunta è stata sorprendente: 328 metri! Il costruttore, nel datasheet dichiara 100 metri all'esterno in LOS.
Pochi metri oltre tale limite, il link crollava e perdeva un numero eccessivo di pacchetti.
Per concludere si riporta un grafico riassuntivo (Figura 7.31) delle distanze massime raggiunte, in base alla collocazione dei nodi.

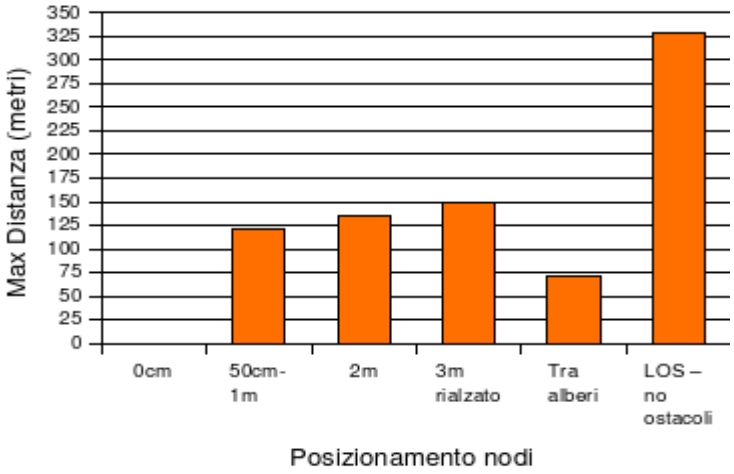


Figura 7.31: Massima distanza raggiunta nelle varie situazioni (i valori delle ascisse delle prime quattro colonne indicano a che altezza sono stati posizionati i nodi nel campo agricolo)

7.2.3 Test multi-hop

È stato creato un caso con due hop e con indirizzamento statico, ovvero ogni nodo sa già chi è il prossimo hop a cui spedire. Nella Figura 7.32, si può osservare come è stata strutturata la rete.

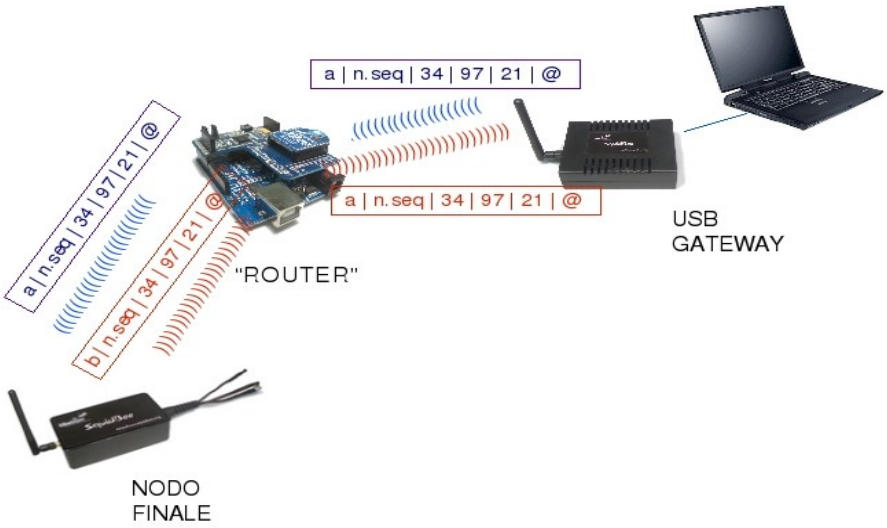
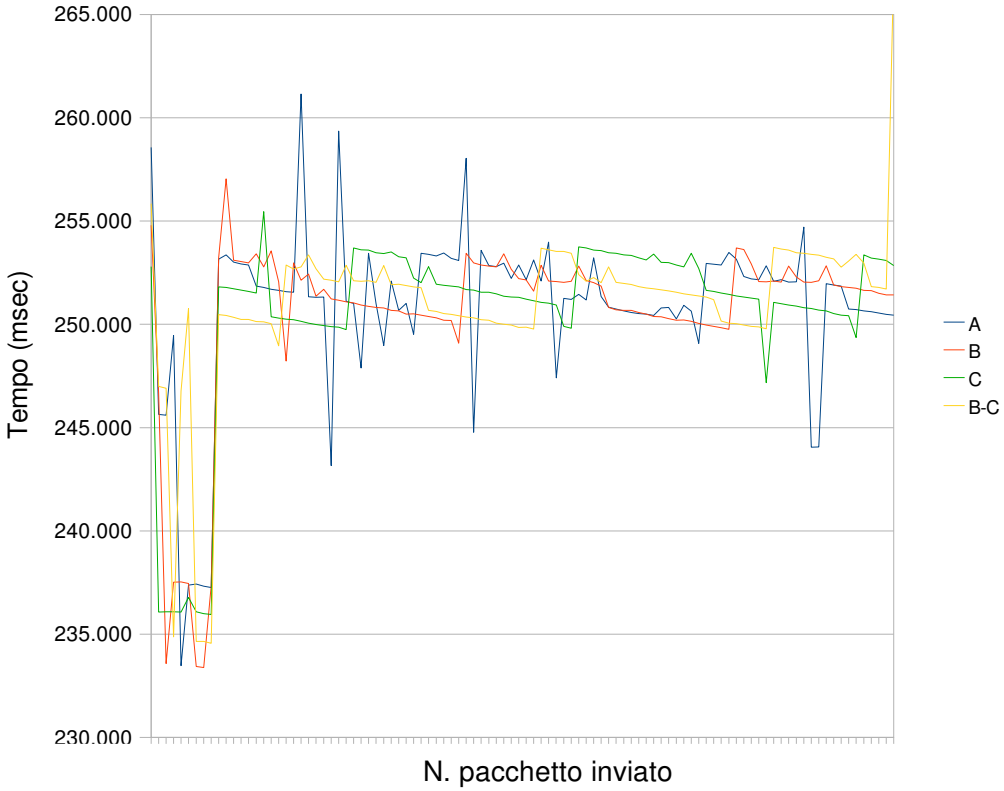


Figura 7.32: Struttura della rete nel test multi-hop

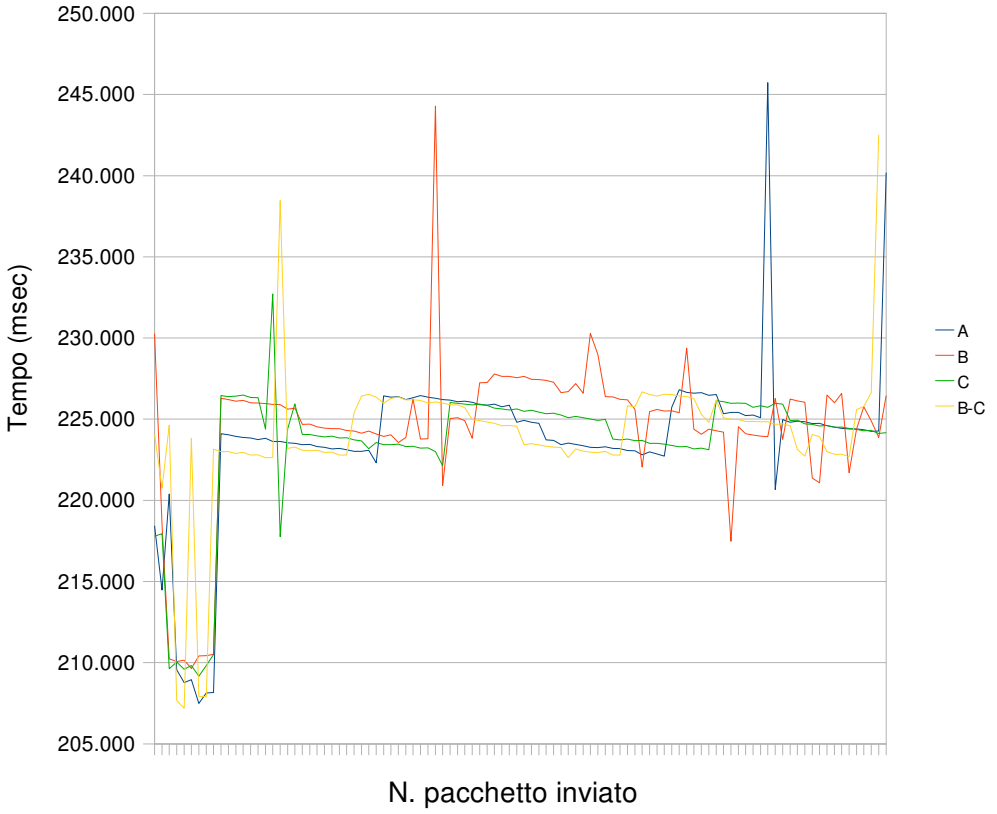
Avviando il programma ping dal pc, l'USB gateway invia 100 pacchetti ogni 10 ms al nodo “Router”, il quale li inoltra al “Nodo Finale”. Il contenuto di ogni pacchetto è: “ | a | n.sequenza | 34 | 97 | 21 | @|”. Il campo “a” è un identificativo del pacchetto, il n.sequenza è il numero del pacchetto inviato, 34 97 21 sono dati qualsiasi, “@” è il marker di fine messaggio.
Quando un nodo riceve dati in ingresso, li legge finché non trova una “@”, dopodiché controlla che il messaggio inizi con una “a” (o “b” per il router) e finisca con “@”. In caso affermativo trasmette indietro al mittente un pacchetto simile.
Nel momento in cui il pacchetto arriva al nodo finale, egli risponde con “b” invece che con “a”, affinché il router capisca la provenienza del pacchetto ricevuto e sappia se inoltrare verso l'alto o verso il basso il messaggio.
I test sono stati eseguiti variando la posizione dei nodi secondo i quattro scenari di Figura 7.33.

Figura 7.33: Piante raffiguranti la collocazione dei nodi nei vari test

I grafici di Figura 7.34, Figura 7.35 e Figura 7.36, mostrano i tempi di ping in confronto alle diverse posizioni dei nodi.



a)



b)

Figura 7.34: Confronto tra i tempi di ping con impostazione del destinatario (a) e senza (b).

I tempi di ping in questi caso non dipendono particolarmente dalla posizione dei nodi e lo scenario in teoria migliore (scenario A), in cui tutti i nodi sono vicini tra loro, non comporta nessun vantaggio rispetto agli altri scenari.

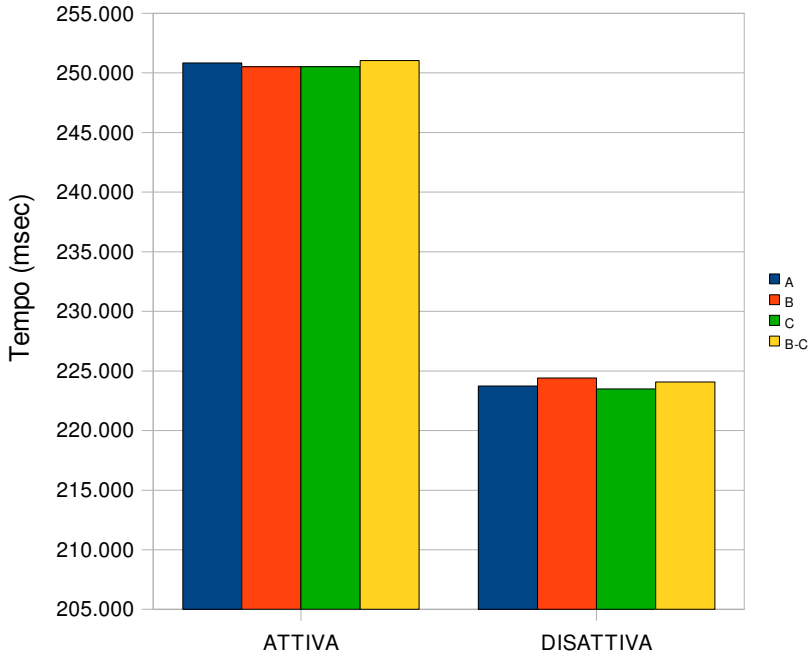


Figura 7.35: Tempo medio di ping in relazione allo scenario e all'attivazione del meccanismo di impostazione destinatario

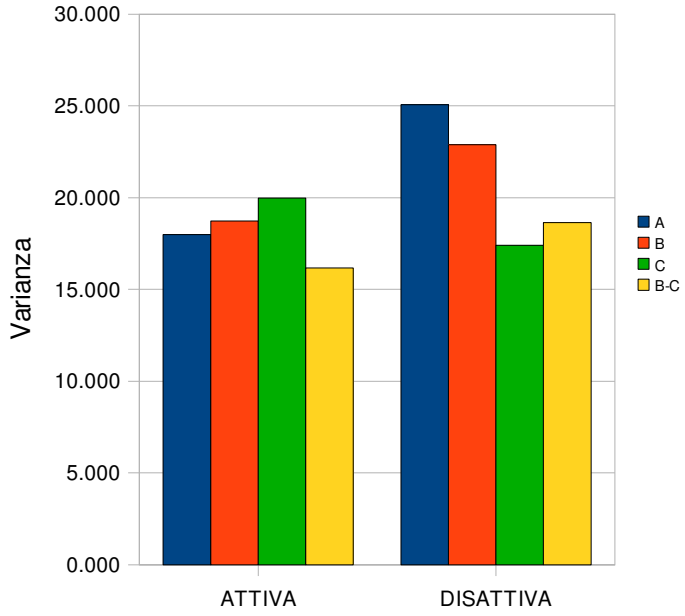


Figura 7.36: Varianze del tempo medio di ping in relazione allo scenario e all'attivazione del meccanismo di impostazione destinatario

La varianza è quasi sempre maggiore laddove l'impostazione destinatario è disattiva.
Il fatto di abilitare/disabilitare l'impostazione del destinatario in questo test a 2 hop è possibile solo

per il nodo finale e l'USB gateway, ma non ovviamente per il router che deve commutare ogni volta a seconda del pacchetto da inviare.
I ritardi introdotti dal impostazione del destinatario, sono in linea con quelli del single-hop, ovvero intorno ai 25-30 ms, come si nota dalla Figura 7.37.

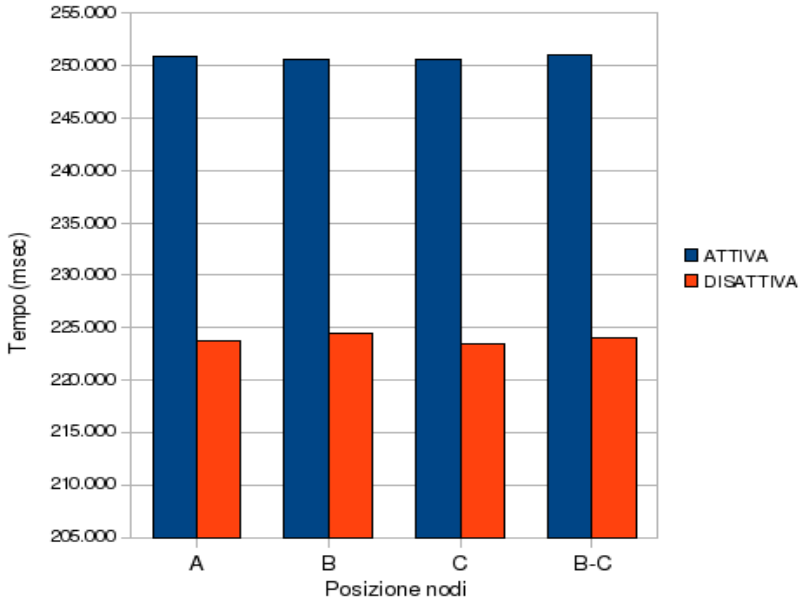


Figura 7.37: Tempo medio di ping in relazione allo scenario e all'attivazione del meccanismo di impostazione destinatario

7.2.4 Confronto con altri test

I risultati ottenuti nei test effettuati trovano conferma anche in letteratura. Infatti, in [74], vengono esposti i risultati delle prove svolte con dei nodi *XBee Pro* in campi all'aperto. L'XBee Pro è la versione dell'XBee con una potenza trasmissiva maggiore, 18 dB invece di 0 dB, mentre le altre caratteristiche sono equivalenti.
Anche qui, i risultati evidenziano il problema della collocazione in altezza del nodo. Se il nodo viene posizionato in basso, si creano fenomeni di riflessione e di multipath, che degradano il segnale. Per trovare l'altezza teorica ideale a cui elevare i nodi, è possibile fare riferimento al modello della *Zona di Fresnel.*

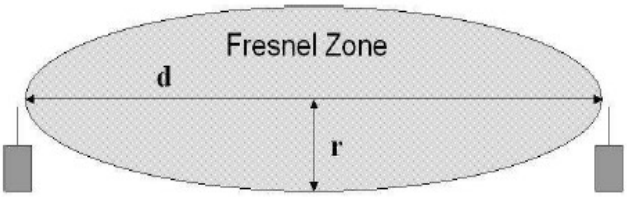


Figura 7.38: Zona di Fresnel. "d" indica la distanza tra trasmettitore e ricevitore, "r" il raggio della zona di Fresnel

La zona di Fresnel, schematizzata in Figura 7.38 è una delle elissoidi concentriche che definiscono i volumi nel *pattern* della radiazione di un'apertura (solitamente) circolare. Le zone di Fresnel derivano dalla diffrazione nell'apertura circolare.
Per massimizzare la potenza del segnale ricevuto, bisogna minimizzare l'effetto di segnali fuori fase attraverso la rimozione degli ostacoli dall'RF Line of Sight.
Quindi, la zona di Fresnel rappresenta un volume privo di effetti di riflessione, ma se un ostacolo si trova in questa zona, si verifica un'attenuazione del segnale maggiore e l'RSSI (Received Signal Strength Indication) sarà in continua fluttuazione a causa del fenomeno del multipath.

Il *Fresnel Zone Radius (FZR)* è dato dalla seguente formula:

$$r_{meters} = 17.32\sqrt{\frac{d_{km}}{4\, f_{GHz}}}$$

dove *d* è la distanza tra le antenne in chilometri, *f* è la frequenza in Ghz.
Si ritiene che antenne alte meno del 60% dell'FZR comportino un drastico *fading* con conseguente attenuazione del segnale.
La tipologia di test svolto dagli autori, consiste nella spedizione di pacchetti a nodi single-hop per registrare l'RSSI del pacchetto ricevuto, il delay e il tasso di perdita dei pacchetti.
Nel grafico di Figura 7.39 viene riportato l'RSSI al variare della distanza e della posizione dei nodi.

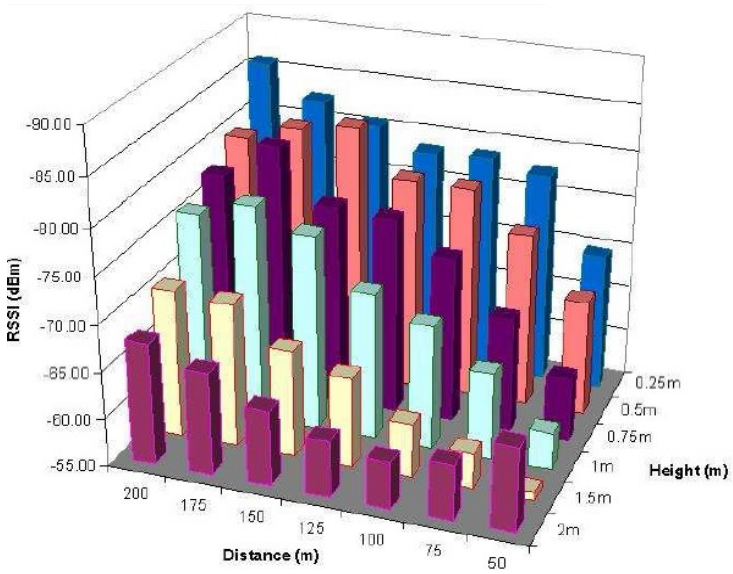


Figura 7.39: RSSI rilevato nei test condotti in [74]

Come si può notare, quando i nodi sono collocati in basso, l'RSSI è peggiore. Addirittura un nodo distante 200 metri e posto a 2 metri di altezza ha all'incirca lo stesso RSSI di un nodo distante 50 metri ma a 25 centimetri di altezza. Un altro elemento rilevabile da questi risultati è la presenza di alcuni comportamenti anomali difficilmente spiegabili, come ad esempio il fatto che nel test con distanza di 50 metri, l'RSSI del nodo elevato a 2 metri è molto più basso di quello in cui il nodo è a 1,5 metri di altezza.
Dall'analisi della deviazione standard dell'RSSI (Figura 7.40), si osserva come i nodi posti in basso siano caratterizzati da maggiori fluttuazioni del segnale, dovute al multipath.

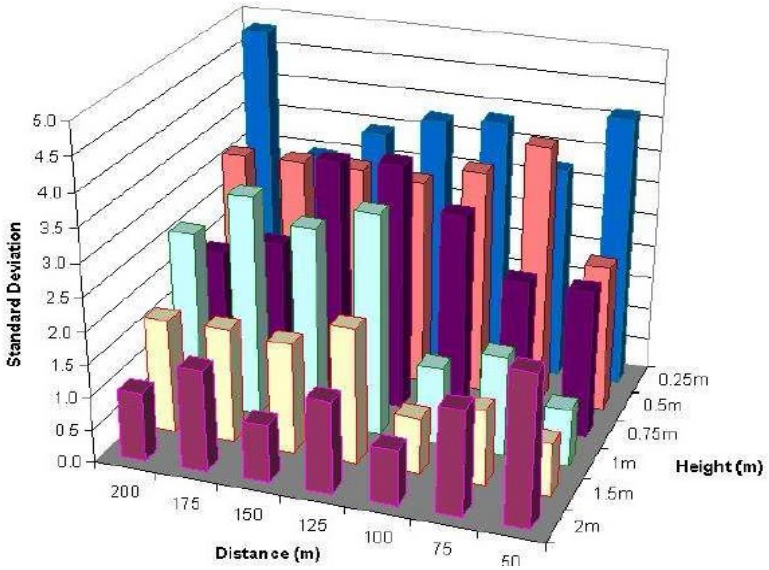


Figura 7.40: Standard Deviation dell'RSSI del grafico precedente

In Figura 7.41 viene riportato il numero di tentavi di ritrasmissione di pacchetto effettuati dai nodi. Quando i nodi sono posizionati in basso, il numero è maggiore, poiché maggiore è il numero di pacchetti considerati persi.

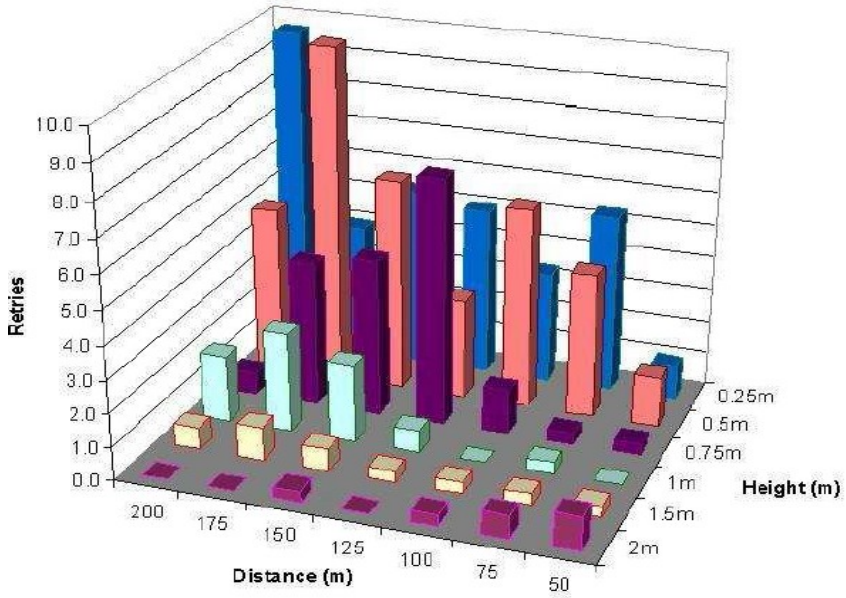


Figura 7.41: Numero di tentativi di ritrasmissione effettuati dai nodi

Infine, sempre in [74], sono stati effettuati dei test per valutare che influenza ha la pioggia sul segnale ricevuto.
Dall'analisi dei risultati si è dedotto che l'RSSI è di poco inferiore in caso di pioggia intensa e i canali (della banda 2.4Ghz) in cui si sono registrate migliori prestazioni sono quelli sulle frequenze più basse, come si nota dai grafici in Figura 7.42 e in Figura 7.43, riportanti RSSI e deviazione standard nella situazione di pioggia e asciutto. In entrambe le condizioni ambientali non è stato comunque perso neanche un pacchetto.

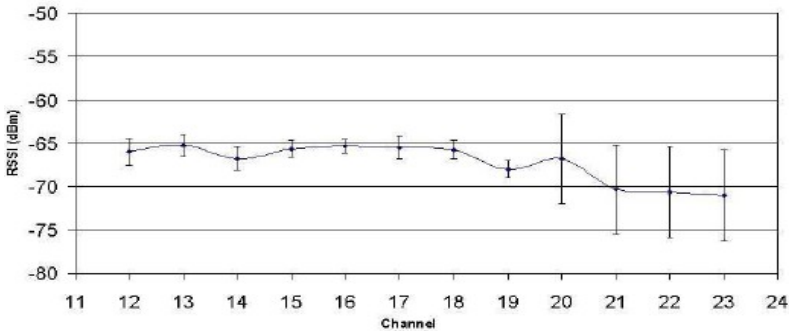


Figura 7.42: RSSI in condizioni di pioggia

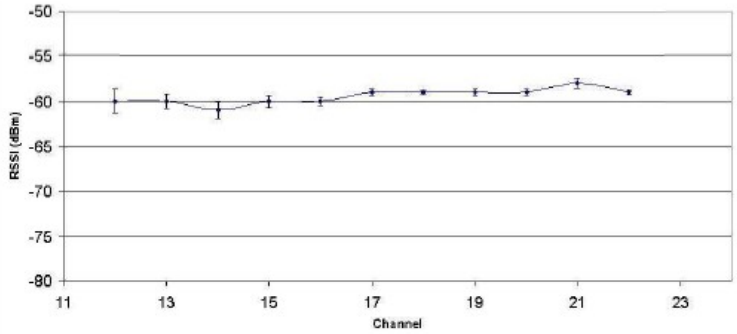


Figura 7.43: RSSI in condizione di asciutto

La variazione dell'RSSI in base al posizionamento in altezza dell'antenna è stato rilevato anche in [7], dove sono stati adoperati nodi MicaZ in una serra.
Dai grafici di Figura 7.44 e di Figura 7.45, si nota come l'RSSI e il range di trasmissione migliorino all'aumentare dell'altezza dell'antenna.

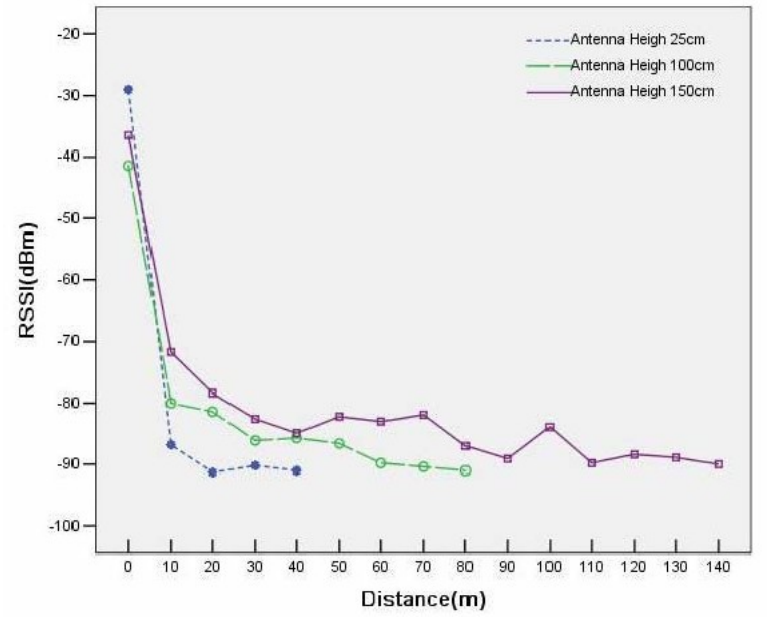


Figura 7.44: Andamento dell'RSSI al variare dell'altezza dell'antenna

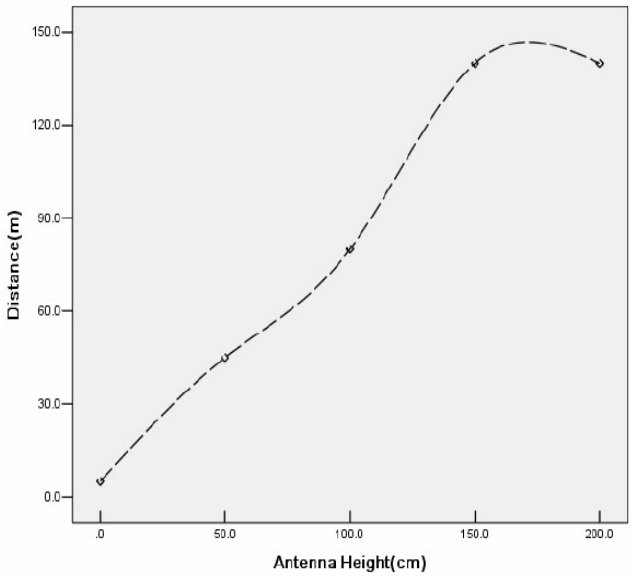


Figura 7.45: Range di trasmissione al variare dell'altezza dell'antenna

7.3 TEST DEL SISTEMA FINALE

Una volta terminato lo sviluppo delle varie componenti del progetto, sono stati effettuati dei test per verificare il corretto funzionamento dell'intero sistema e valutare le prestazioni della WSN.

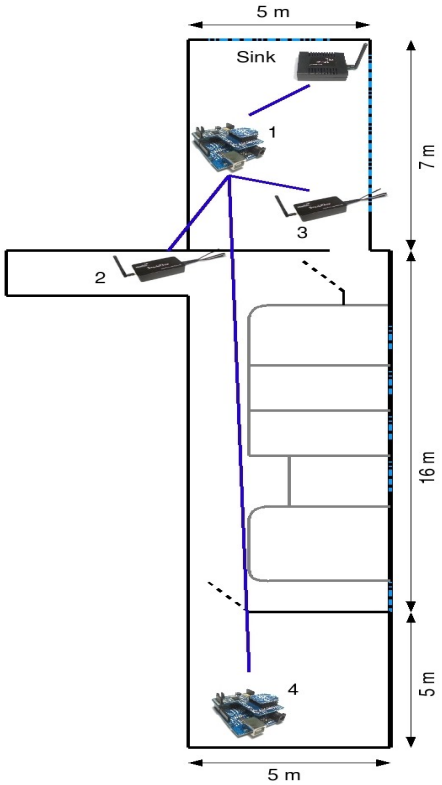


Figura 7.46: Pianta raffigurante la disposizione dei nodi nel dipartimento di telecomunicazioni

Sono stati installati i vari componenti software, ovvero il database, il web server e l'applicazione del sink su due pc, dopodiché sono stati posizionati quattro nodi della rete di sensori nel piano del

dipartimento di telecomunicazioni, di cui viene riportata una pianta in Figura 7.46 e alcune foto in Figura 7.47.

Figura 7.47: L'ambientazione del test finale (i cartelli in verde indicano la posizione dei nodi)

I nodi, due Squidbee e due Arduino con XBee (con antenna integrata e non dotati di sensori) sono stati collegati a delle batterie da 9 Volt, dalla capacità di circa 150 mAh (anche se non tutte le batterie erano cariche al 100%) e l'intervallo di trasmissione dei dati è stato impostato a 1 minuto. L'attivazione dei nodi è avvenuta in maniera graduale, cioè sono stati attivati l'uno dopo l'altro a distanza di qualche minuto.
La rete che si è creata è rappresentata in Figura 7.46 e in Figura 7.48: il sink ha un nodo figlio, il quale è il nodo padre dei nodi rimanenti.

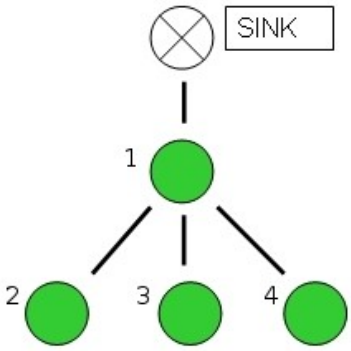


Figura 7.48: Struttura della WSN nel test finale

La qualità del link tra le varie stazioni è risultata buona, nonostante la presenza di vari ostacoli come muri, strutture metalliche e fonti di interferenza: sono bastati al più due hop per raggiungere il sink. Il basso numero di hop, a parità di potenza trasmissiva, è da considerarsi un fattore positivo per la rete, dato che consente un risparmio energetico.
Il sistema è stato lasciato attivo per 4 ore e 41 minuti, poi è stato fermato. Terminato questo periodo di tempo, le batterie risultavano ancora cariche. La percentuale di perdita di pacchetti è risultata molto bassa: la media, riferita a tutta la rete, è pari a 1,24%, mentre il valore per i singoli nodi è rispettivamente di 0%, 0,35%, 0% e 4,6% (vedasi Figura 7.49). Quest'ultima percentuale, più alta rispetto alle altre, è probabilmente dovuta a qualche difficoltà di comunicazione del nodo in questione, che è un nodo foglia dotato di antenna integrata e posizionato in un punto nascosto agli altri nodi.

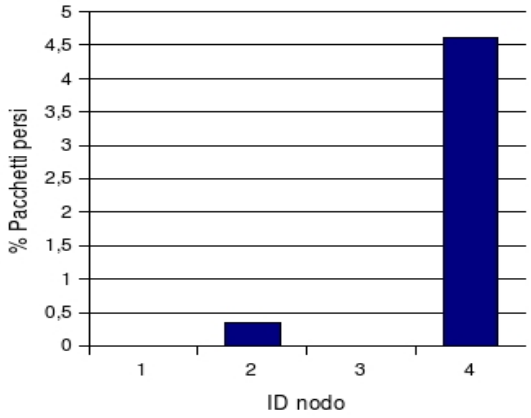


Figura 7.49: Percentuale di pacchetti persi per ogni nodo nel test finale

Per quanto riguardo l'autonomia, è probabile che i nodi avessero energia a sufficienza per qualche ora ancora, ma non di più. Se si considera il fatto che: le batterie utilizzate hanno una capacità molto bassa, ovvero circa 150 mAh contro 2000 – 2500 mAh delle batterie che vengono spesso adoperate nei nodi delle WSN; manca il meccanismo di risparmio energetico del microcontrollore, che verrà possibilmente implementato in futuro; l'intervallo di trasmissione dei dati è molto breve e, quindi, i nodi “riposano” poco; allora, si può dedurre che il lifetime della rete è buono e può migliorare notevolmente considerando questi tre fattori appena elencati, ovvero usando una batterie molto più capace, attivando il power saving della CPU e impostando un intervallo di trasmissione più lungo. In questo modo, si potrebbe ottenere un'autonomia anche di pochi giorni.
Il sistema di gestione della WSN è funzionato correttamente: i dati arrivati dai nodi sono stati salvati nel database, che al termine della prova conteneva più di 1000 entry in un file da 196 KB. Questi dati poi sono stati consultati da un altro pc via LAN, adoperando l'applicazione web sviluppata. Il web server non è stato installato sul mini-pc, poiché a causa delle sue limitate risorse hardware, non è consigliabile installare ed eseguire altre applicazioni oltre a quella che gestisce il sink ed a uSQLiteServer.

CAPITOLO 8

CONCLUSIONI E SVILUPPI FUTURI

Con questo progetto di tesi si è cercato di realizzare un un sistema di monitoraggio ambientale tramite WSN completo, pronto all'uso, intuitivo ed economico.
Lo scopo principale è stato quello di creare una soluzione che finora nessuno aveva proposto, ovvero un pacchetto completo, dalla WSN all'applicazione web, creato con tecnologie economiche, come Arduino e XBee. A differenza della maggior parte degli studi sulle WSN che si trovano in letteratura, i quali propongono molti progetti puramente teorici, questa tesi ha proposto una soluzione pratica, un'applicazione reale di una WSN, per ottenere un obiettivo preciso.
Si può dire che questo obiettivo sia stato raggiunto, sebbene ci siano alcuni aspetti migliorabili, come le funzionalità del sistema e la sicurezza. Infatti, data l'estensione di questo progetto non è stato possibile entrare nei dettagli di tutti gli aspetti, mentre altri non sono stati nemmeno trattati.
Il primo tra questi, è la sicurezza. La sicurezza entra in gioco in più parti del sistema. Innanzitutto, andrebbe creata una gestione della multi-utenza dell'applicazione web, ovvero bisognerebbe gestire l'accesso di ogni utente con username e password tramite connessione sicure al web server, per evitare che qualche malintenzionato modifichi ad esempio le aree di allarme di un utente oppure il suo profilo, o che acceda ad altre informazioni senza autorizzazione. Eventualmente, per casi particolari, andrebbe protetta la connessione tra web server e database, che al momento si scambiano dati in chiaro, consentendo così l'intercettazione e la modifica dei dati provenienti dalla WSN. Implementate queste contromisure, il sistema sarebbe pronto per essere installato su macchine accessibili in remoto via Internet, mentre adesso è adatto solo ad un utilizzo in un ambiente fidato (ad esempio una rete LAN).
Inoltre, c'è l'aspetto della sicurezza nelle reti di sensori, che è più critico da risolvere rispetto agli altri problemi. Infatti, è più facile attaccare una WSN. Un malintenzionato potrebbe senza troppe difficoltà intercettare i dati spediti dai nodi o, ancora peggio, effettuare del data tampering. Quest'ultimo termine significa manomissione dei dati, che può avvenire in più modi: il malintenzionato potrebbe sostituire il nodo con un altro che spedisce dati falsi oppure potrebbero alterare fisicamente i sensori in modo da sfalsare le rilevazioni (ad esempio coprendo il sensore di luminosità).
Un altro aspetto migliorabile è l'applicazione web: per ora fornisce funzioni quasi basilari, ma in futuro potrebbe essere estesa con nuovi strumenti di analisi dati, in modo da permettere uno studio più efficace dell'ambiente.
Infine, si potrebbero provare nuovi protocolli nella WSN in grado di migliorare ulteriormente l'efficienza energetica e diminuire il tasso di perdita dei pacchetti. Oltre a ciò, è opportuno procedere all'implementazione del meccanismo di power saving per il microcontrollore e pensare a come migliorare certi aspetti secondari, ad esempio come modificare il case del nodo per una migliore protezione dai fattori atmosferici ed ambientali.

APPENDICE A

SPECIFICA DEI REQUISITI SOFTWARE

Di seguito, viene riportato il documento di specifica dei requisiti software del sistema di gestione e visualizzazione dei dati della WSN, formattato secondo le specifiche dello standard IEEE 830-1998.

A1 Introduzione

A1.1 Obiettivo

L'obiettivo di questo documento è fornire una descrizione generale ed un elenco delle funzioni di un sistema di monitoraggio ambientale via web.

A1.2 Ambito

Il sistema è uno strumento di monitoraggio ambientale via web, ovvero un cruscotto (dashboard) accessibile tramite browser, con il quale è possibile visualizzare dati provenienti dai nodi della Wireless Sensor Network e modificare alcuni parametri di funzionamento della rete stessa.
Il sistema è accessibile da più utenti contemporaneamente, riporta i dati in tempo reale e può generare eventi di allarme in base al verificarsi di condizioni stabilite dall'utente stesso.

A1.3 Definizioni, acronimi e abbreviazioni

WSN: Wireless Sensor Network - rete wireless costituita da nodi autonomi dotati di sensori per il monitoraggio di ambienti o sistemi.

XML: eXtensible Markup Language - metalinguaggio utilizzato per creare nuovi linguaggi, atti a descrivere documenti strutturati.

PHP: PHP: Hypertext Preprocessor - Linguaggio di scripting interpretato, con licenza open source, originariamente concepito per la realizzazione di pagine web dinamiche.

DBMS: Database Management System - Insieme di sistemi software che consente la creazione e la manipolazione efficiente di banche dati.

RIA: Rich Internet Application - applicazione web avente le caratteristiche e le funzionalità di un programma desktop.

A1.4 Prefazione

Nella Sezione A.2 verranno illustrate le caratteristiche principali del prodotto, al fine di creare il contesto per la definizione dei requisiti utente e di sistema del capitolo successivo.
Questa sezione è costituita da sei sottosezioni:

1. Prospettiva del prodotto
2. Funzioni del prodotto
3. Caratteristiche dell'utente

4. Vincoli
5. Assunzioni e dipendenze
6. Divisione dei requisiti

Nella Sezione A.3, verranno approfonditi i requisiti funzionali e non funzionali del progetto. Tale sezione è suddivisa in sei sottosezioni:

- Interfacce esterne
- Funzioni
- Requisiti prestazionali
- Requisiti logici del database
- Vincoli progettuali
- Attributi del sistema software
- Ulteriori commenti

Questo documento segue la struttura raccomandata dallo standard IEEE 830-1998.

A2 Descrizione generale

A2.1 Prospettiva del prodotto

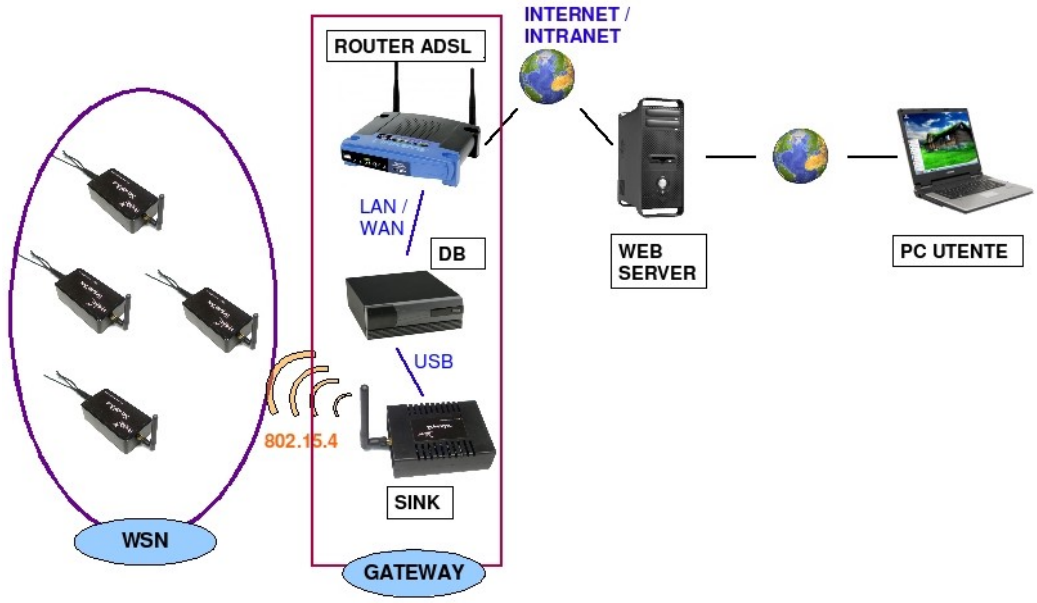


Figura A1.1: Architettura hardware del sistema

L'architettura del sistema (Figura A1.1) è costituita dai nodi della WSN che trasmettono i dati dei propri sensori al gateway. Quest'ultimo gestisce la rete e manda via USB i dati ricevuti ad un sistema embedded, sul quale vi è il database. I dati dei nodi vengono salvati nel database e diventano accessibili dall'utente grazie al web server. Il sistema embedded è connesso via LAN/WAN ad un router Wi-Fi (eventualmente con modem ADSL), che a sua volta è connesso a Internet.
L'utente accede alla dashboard aprendo il proprio browser e collegandosi al web server. E' previsto l'utilizzo da un solo tipo di utente, che userà principalmente il sistema per monitorare ed analizzare i dati raccolti, oltre che controllare lo stato della WSN.
La dashboard è caratterizzata da quattro funzionalità principali:

- monitoraggio ambiente
- analisi dati
- gestione allarmi

- supervisione rete

La funzione di monitoraggio ambientale è la principale. Essa permette di visualizzare su una mappa interattiva la posizione dei nodi e i dati rilevati da essi, codificati tramite colore e continuamente aggiornati. I dati sono inoltre riassunti in una tabella e sono riportati su dei grafici in funzione del tempo e del nodo selezionato dall'utente. È possibile impostare due soglie dall'allarme, oltre le quali si attivano degli eventi di notifica visiva sulla schermata stessa oltre alla notifica via e-mail. Le impostazioni di allarme sono configurabili dall'utente con la funzione di gestione allarmi.
Per quanto riguarda l'analisi dei dati, l'utente può creare fino a tre grafici personalizzabili: può infatti scegliere quali dati raffigurare, di qualsiasi giorno e qualsiasi nodo. E' inoltre possibile esportare i dati selezionati su foglio di calcolo.
Attraverso la gestione degli allarmi, l'utente può attivare delle soglie d'allarme sui dati rilevati dai sensori, oltre le quali il sistema avvisa l'utilizzatore con delle notifiche visive e via e-mail. Possono essere impostate soglie diverse per nodi diversi e tutti i valori impostati devono essere visualizzabili.
Infine, la supervisione della rete permette la visualizzazione della configurazione attuale della rete: uno schema raffigura la topologia, ovvero le connessioni stabilite tra i vari nodi. Se un nodo si spegne o se ne viene aggiunto un altro, lo schema cambia di conseguenza, rispettivamente segnalando il nodo caduto o disegnando il nuovo nodo. Inoltre è possibile stabilire l'intervallo di ricezione dei dati dai nodi.

A2.1.1 Interfacce

Tutte le schermate della dashboard sono accessibili via browser dall'utente tramite una qualsiasi connessione al server ospitante la dashboard.
Non è richiesto alcun hardware particolare all'utente finale.
L'interfaccia deve essere visualizzabile e gestibile da un qualsiasi browser recente (Firefox 2+, Opera 6+, Safari, IE6+).
Il web server e il database server devono risiedere su uno o più sistemi a basse prestazioni e a basso consumo energetico, alimentati tramite rete elettrica. Il gateway della WSN è collegato al database tramite porta USB. Il database è a sua volta collegato al web server, che gira su una macchina connessa ad Internet.
Il database deve scambiare dati in XML con la dashboard.
Le comunicazioni tra web server e dashboard avvengono tramite HTTP su protocollo TCP/IP.
In caso di errore di trasmissione dati, l'utente deve essere avvertito con un messaggio.

A2.2 Funzioni del prodotto

Le funzioni principali sono quattro: monitoraggio ambientale, analisi dati, impostazione allarmi, supervisione rete.

A2.2.1 Monitoraggio ambientale

E' la funzione fondamentale della dashboard. Consente di visualizzare tutti i dati rilevati dai sensori della WSN.

Attivazione monitoraggio

- L'utente inizializza il monitoraggio dei dati accedendo alla pagina web della dashboard
- Il sistema rappresenta i nodi della rete con dei simboli colorati sulla mappa. Di default, vengono caricati i dati attuali di temperatura, codificati tramite il colore dei simboli dei nodi. Nella griglia vengono invece caricati tutti i dati dei sensori per ogni nodo.

Scelta dati raffigurati sulla mappa

- L'utente seleziona, tramite un pulsante, quale dati visualizzare sulla mappa interattiva

(temperatura, umidità o luminosità)
- Il sistema carica i dati scelti nella mappa, rappresentando con un simbolo la posizione del nodo da cui proviene quel dato e con il colore il valore del dato stesso. Una legenda mette in corrispondenza il colore con il valore approssimato.

Visualizzazione dati in griglia
- L'utente visualizza nella griglia gli ultimi dati ricevuti da tutti i nodi (temperatura, umidità, luminosità)
- L'utente può far scorrere la tabella in senso verticale
- Se un nodo comunica un valore che supera la soglia di allarme impostata, il sistema evidenzia con un colore diverso la riga della tabella di quel nodo
- La tabella è ordinabile in maniera crescente o decrescente rispetto al numero del nodo, al valore attuale di temperatura, di umidità o di luminosità.

Visualizzazione dell'andamento dei valori riferiti ad un nodo
- L'utente seleziona un nodo dalla mappa cliccando sul suo simbolo
- Il sistema crea tre grafici raffiguranti l'andamento nel tempo di temperatura, umidità e luminosità per quel nodo.

Visualizzazione valori massimi e minimi di un nodo
- L'utente seleziona un nodo dalla mappa cliccando sul suo simbolo
- Il sistema riporta in delle caselle di testo i valori di massimo e minimo, a partire dalle ore 24 fino al momento attuale, di temperatura, umidità e luminosità. Oltre a ciò, riporta anche l'ora in cui è stato registrato il massimo/minimo.

Visualizzazione del valore di massimo e minimo in tutta la rete
- Il sistema aggiorna automaticamente i valori di massimo e minimo rispetto a tutta la rete e li visualizza in una casella di testo. I valori vengono calcolati a partire dalle ore 24 e azzerati ogni giorno.

A2.2.2 Analisi dati
Questa funzione permette di analizzare più in profondità i dati rilevati dai sensori, creando grafici personalizzabili o esportando i valori registrati su un foglio di calcolo.

Creazione grafici personalizzati
- L'utente accede alla schermata "analisi dati"
- L'utente seleziona quale valore gli interessa, di quale periodo, da quali nodi
- L'utente sceglie quanti grafici (da 1 a 3) visualizzare
- Il sistema crea il grafico.

Esportazione su foglio di calcolo
- L' utente accede alla schermata "analisi dati"
- L'utente seleziona quale valore gli interessa, di quale periodo, da quali nodi
- L'utente avvia, tramite pulsante, l'esportazione
- Il sistema crea e invia il foglio di calcolo
- L'utente salva o apre il foglio di calcolo ricevuto.

A2.2.3 Gestione allarmi
Questa funzione consente all'utente di definire delle soglie di allarme sui valori riportati dai sensori. Oltrepassata una soglia, il sistema crea delle notifiche visive e/o invia una e-mail.

Impostazione soglie di allarme

- L'utente accede alla schermata "gestione allarmi"
- L'utente seleziona dalla mappa il nodo o i nodi su cui attivare gli allarmi
- L'utente sceglie quali valori monitorare (temperatura, umidità, luminosità)
- L'utente imposta il valore di soglia
- Il sistema riassume gli allarmi impostati in una tabella.

A2.2.4 Supervisione rete

Questa funzionalità permette di visualizzare la configurazione topologica della rete e di impostare l'intervallo di trasmissione dei dati dai nodi.

Visualizzazione della configurazione della rete WSN

- L'utente accede alla schermata "WSN"
- Il sistema raffigura automaticamente la topologia della rete. Non appena un nodo si aggiunge o si disconnette dalla rete, il sistema aggiorna automaticamente la topologia della WSN.

Impostazione intervallo di trasmissione dati

- L' utente accede alla schermata "WSN"
- L'utente inserisce il valore dell'intervallo nelle casella di testo apposita
- Il sistema applica la modifica a tutta la rete.

A2.3 Caratteristiche dell'utente

Utenti tipici di questo prodotto sono ricercatori scientifici, geologi, guardie forestali, ingegneri civili, gestori di terreni agricoli e addetti alla sorveglianza del territorio in generale.
Si tratta dunque di persone con background scientifico molto differente e con background tecnologico che potrebbe essere anche molto scarso.
Per poter utilizzare il sistema nelle sue funzioni principali, devono poter bastare le conoscenze di base di navigazione di una pagina web.
Per quanto riguarda la frequenza di utilizzo della dashboard, alcuni utenti potrebbero necessitare di accedervi qualche minuto ogni giorno, ma la maggior parte farà uso attivo del sistema per poche ore in una settimana oppure vi accederà solo periodicamente (ogni mese, ogni semestre, ogni anno).
Alcuni applicazioni (tipo monitoraggio incendi), necessitano però che il sistema sia attivo 24 ore su 24, anche se l'utente è lontano dal monitor, in modo da avvisarlo in caso di allarme.

A2.4 Vincoli

Tutti i dati provenienti dai nodi devono essere salvati in locale nel database.
I dati che escono dal database devono essere in formato XML.
Il web server e il database server devono utilizzare poche risorse hardware. Le macchine su cui gireranno sono infatti dotate di scarse capacità computazionali, quindi, i software installati non devono fare un uso intensivo della CPU.
Il sistema deve essere in grado di aggiornare i dati con una frequenza che sia almeno pari all'intervallo di trasmissione dati impostato dall'utente (minimo 1 minuto).
Il web server e il database server non devono essere accessibili da utenti non autorizzati

A2.5 Assunzioni e dipendenze

La rete WSN deve essere già installata e un addetto deve già aver inserito la posizione (latitudine, longitudine) di ogni nodo nel database.
L'utente finale deve avere una connessione TCP/IP verso il server.
Il traffico HTTP non deve essere bloccato nella connessione tra l'utente e il server.

A2.6 Divisione dei requisiti

Tutte le caratteristiche finora illustrate, verranno implementate.
Se le prestazioni del sistema dovessero risultare insufficienti, verrà valutata l'installazione di un altro web server e/o database server.

A3 Requisiti specifici

A3.1 Interfacce esterne

A3.1.1 Interfacce utente

L'utente interagisce col sistema tramite quattro schermate, corrispondenti alle quattro funzionalità principali. Di seguito si riportano alcuni mock-up dell'interfaccia.
1) Monitoraggio ambientale (Figura A1.2).
2) Analisi dati (Figura A1.3).
3) Gestione allarmi (Figura A1.4).
4) WSN (Figura A1.5).

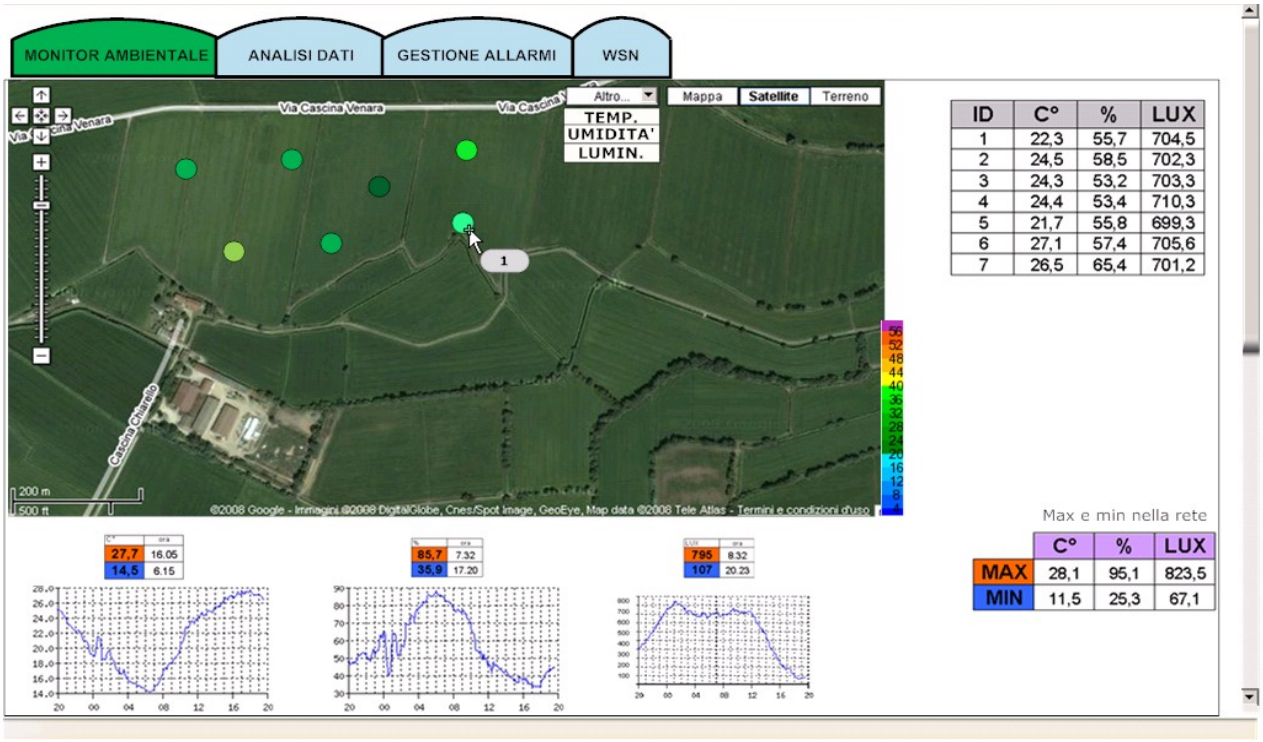
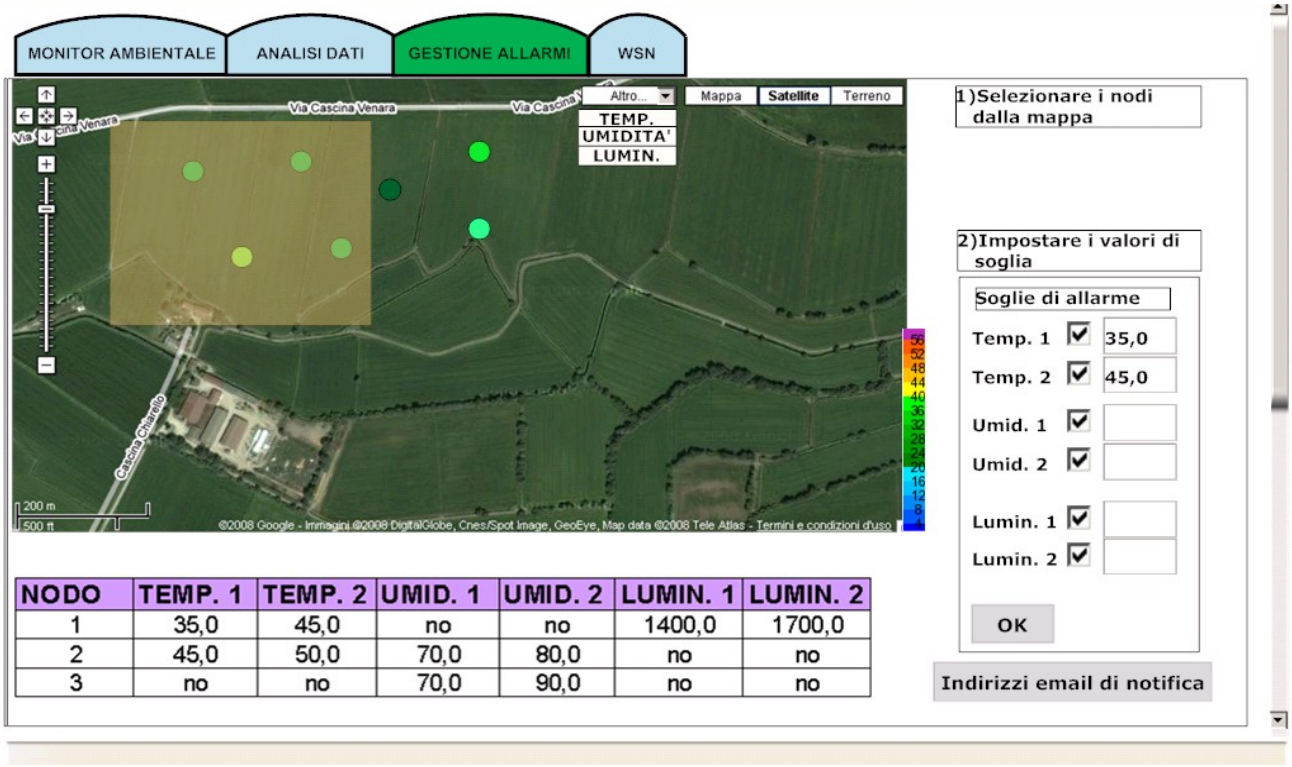


Figura A1.2: Mockup della funzionalità di monitoraggio

Figura A1.3: Mockup della funzionalità di analisi dati



NODO	TEMP. 1	TEMP. 2	UMID. 1	UMID. 2	LUMIN. 1	LUMIN. 2
1	35,0	45,0	no	no	1400,0	1700,0
2	45,0	50,0	70,0	80,0	no	no
3	no	no	70,0	90,0	no	no

Figura A1.4: Mockup della funzionalità di gestione degli allarmi

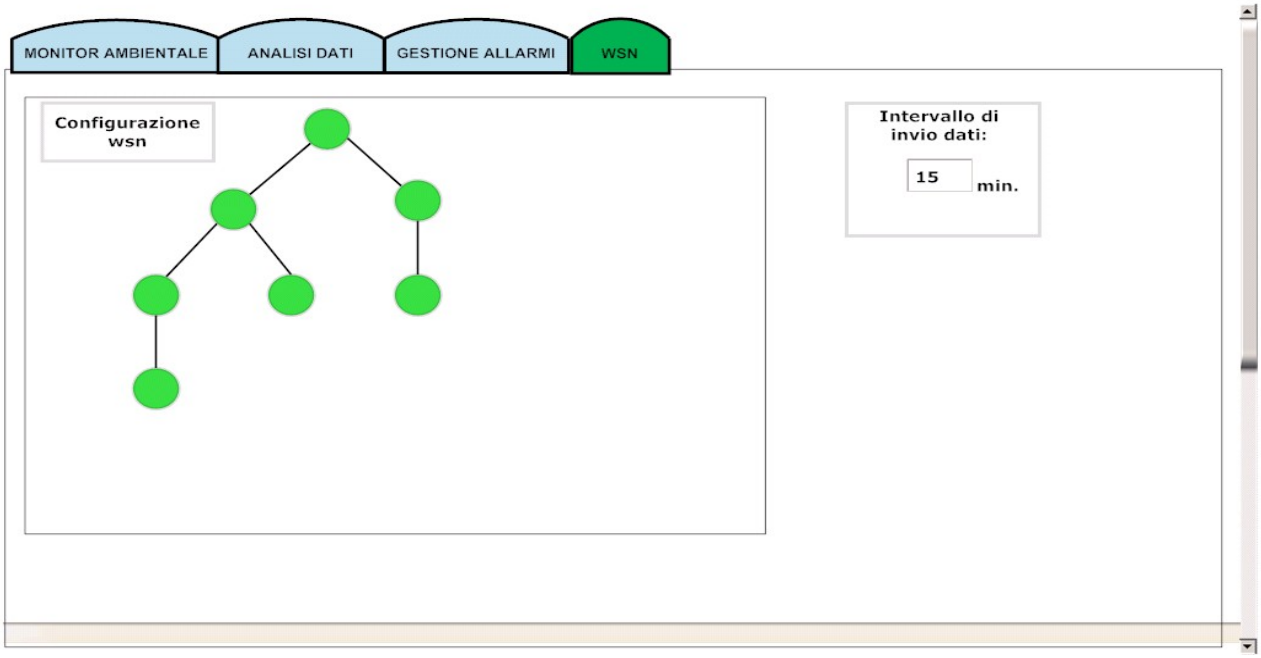


Figura A1.5: Mockup della funzionalità di gestione della WSN

A3.1.2 Interfacce software

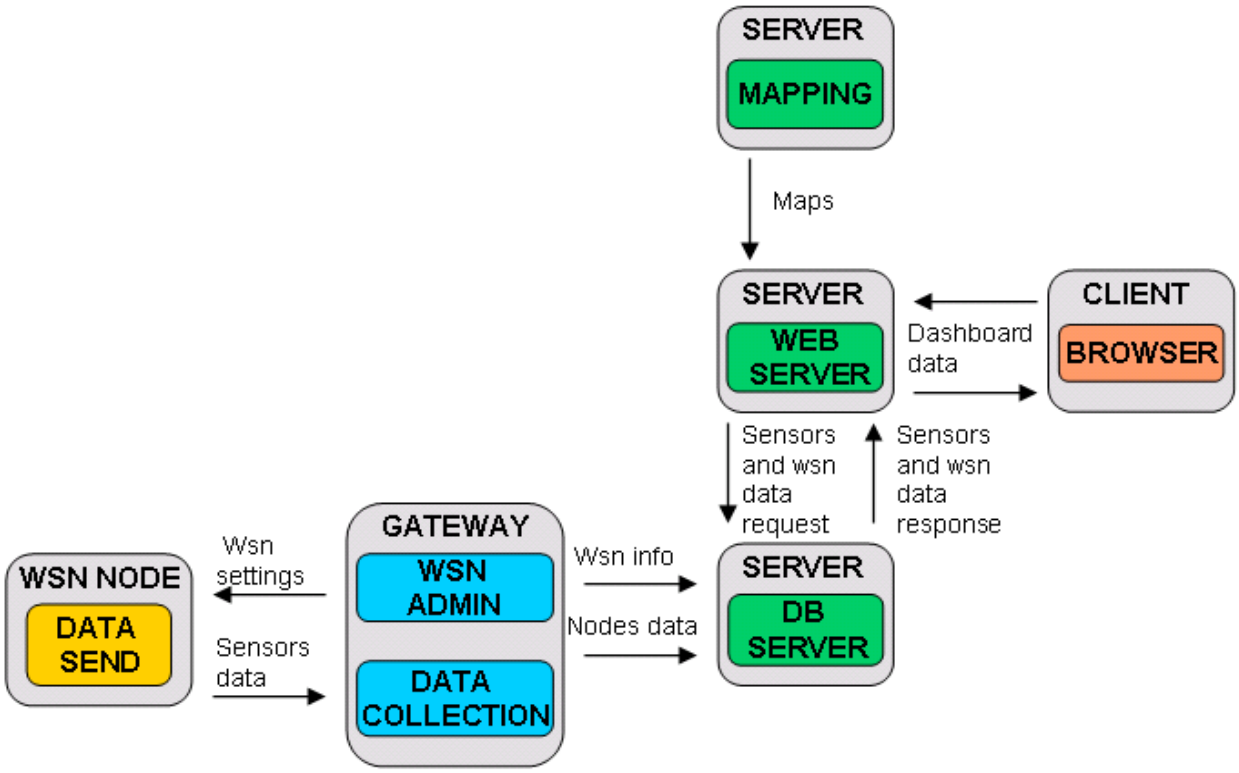


Figura A1.6: Architettura software

Il sistema è caratterizzato da una struttura multi-tier (Figura A1.6), costituita da:

- Insieme dei nodi (rete WSN)
- Gateway della WSN
- DBMS
- Web server
- Mapping server

- Client

I *nodi* della WSN si occupano di leggere periodicamente i dati dei sensori e di spedire questi dati ad un punto di raccolta, detto gateway. Il *gateway* è un nodo della WSN avente compiti speciali:

- Amministrazione della WSN:
  - gestisce l'ingresso di nuovi nodi della rete, assegnando loro un indirizzo identificativo
  - schedula l'invio dei dati da parte dei nodi.

- Raccolta dati:
  - Riceve tutti i dati inviati dai nodi e li salva nel database a cui è collegato (i dati memorizzati comprendono sia dati registrati dai sensori dei nodi, sia i dati di configurazione della WSN)

Tutti i pacchetti scambiati tra i nodi della Sensor Network, gateway compreso, sono in formato *plain-ASCII*.
Il *DBMS* si occupa di memorizzare i dati provenienti dalla WSN e di rendere accessibili questi dati al web server a cui è collegato.
Il *web server* gestisce le richieste provenienti dall'utente, carica dal database le informazioni richieste e le rispedisce al client. Per poter fornire l'accesso alle mappe, deve interrogare un altro server dedicato a questo compito di *mapping.*
Infine il *client*, ovvero il browser dell'utente, costituisce l'interfaccia tra utente stesso e tutto il sistema descritto, attraverso la web dashboard.

A3.2 Funzioni

A3.2.1 Attivazione monitoraggio

- Attori: Utente

- Descrizione: Permette di avviare il processo di monitoraggio ambientale e visualizzare i dati rilevati dai sensori sullo schermo

- Flusso base:
  - Il sistema carica i dati disponibili automaticamente e rappresenta i nodi della rete con dei simboli colorati sulla mappa. Di default, vengono caricati i dati attuali di temperatura, codificati tramite il colore dei simboli dei nodi. Nella griglia vengono invece caricati tutti i dati dei sensori per ogni nodo.

- Flusso alternativo: Nessuno

- Pre Condizioni: L'utente si trova nella pagina principale della dashboard

- Post Condizioni: I dati vengono caricati sulla mappa e nella griglia; la modalità di notifica in caso di allarme è attiva

- Eccezioni: Non pervengono dati aggiornati dalla rete

- Frequenza Stimata di utilizzo: Alta

- Criticità: Alta

A3.2.2 Scelta dati raffigurati sulla mappa

- Attori: Utente

- Descrizione: Permette di scegliere quali dati (temperatura, umidità, luminosità) visualizzare sulla mappa, tramite codifica di colore dei simboli dei nodi

- Flusso base:
    - L'utente seleziona, tramite un pulsante, quale dati visualizzare sulla mappa interattiva (temperatura, umidità o luminosità)
    - Il sistema carica i dati scelti nella mappa, rappresentando con un simbolo la posizione del nodo da cui proviene quel dato e con il colore il valore del dato stesso. Una legenda mette in corrispondenza il colore con il valore approssimato

- Flusso alternativo:
    - L'utente non seleziona alcun tipo di dato

- Pre Condizioni: L'utente si trova nella pagina principale della dashboard

- Post Condizioni: I simboli dei nodi riportano, tramite codifica di colore, il valore della tipologia di dato scelto

- Eccezioni: Non pervengono dati dalla rete

- Frequenza Stimata di utilizzo: Alta

- Criticità: Alta

A3.2.3 Visualizzazione dati in griglia

- Attori: Utente

- Descrizione: Permette di visualizzare tutti i dati attuali di tutti i nodi in una griglia

- Flusso base:
    - Il sistema carica nella griglia gli ultimi dati ricevuti da tutti i nodi (temperatura, umidità, luminosità).

- Flusso alternativo:
    - Se un nodo comunica un valore che supera la soglia di allarme impostata, il sistema evidenzia con un colore diverso la riga della tabella di quel nodo
    - L'utente decide di ordinare la tabella in maniera crescente o decrescente rispetto al numero del nodo, al valore attuale di temperatura, di umidità o di luminosità

- Pre Condizioni: L'utente si trova nella pagina principale della dashboard

- Post Condizioni: I dati vengono caricati nella griglia

- Eccezioni: Non pervengono dati dalla rete

- Frequenza Stimata di utilizzo: Alta

- Criticità: Alta

A3.2.4 Visualizzazione dell'andamento dei valori riferiti ad un nodo

- Attori: Utente
- Descrizione: Permette di visualizzare l'andamento nel tempo di un valore monitorato su un nodo
- Flusso base:
  - L'utente seleziona un nodo dalla mappa cliccando sul suo simbolo
  - Il sistema crea tre grafici raffiguranti l'andamento nel tempo di temperatura, umidità e luminosità per quel nodo
- Flusso alternativo: Nessuno
- Pre Condizioni: L'utente si trova nella pagina principale della dashboard
- Post Condizioni: Vengono creati tre grafici dal sistema
- Eccezioni: Il nodo selezionato non ha ancora fornito nessun dato
- Frequenza Stimata di utilizzo: Alta
- Criticità: Media

A3.2.5 Visualizzazione valori massimi e minimi di un nodo

- Attori: Utente
- Descrizione: Permette di visualizzare il valore massimo e minimo della giornata registrato dal nodo selezionato, con rispettiva ora
- Flusso base:
  - L'utente seleziona un nodo dalla mappa cliccando sul suo simbolo
  - Il sistema riporta in caselle di testo i valori di massimo e minimo di temperatura, umidità e luminosità. Oltre a ciò, riporta anche l'ora in cui è stato registrato il massimo/ minimo.
- Flusso alternativo: Nessuno
- Pre Condizioni: L'utente si trova nella pagina principale della dashboard
- Post Condizioni: Vengono visualizzati i valori massimi e minimi
- Eccezioni: Il nodo scelto non ha ancora fornito dati
- Frequenza Stimata di utilizzo: Media
- Criticità: Bassa

A3.2.6 Visualizzazione del valore di massimo e minimo in tutta la rete

- Attori: Utente

- Descrizione: Permette di visualizzare il valore massimo e minimo della giornata, con rispettivo ora, in riferimento a tutti i nodi presenti nella rete

- Flusso base:
    - All'arrivo di un nuovo dato, il sistema verifica se si tratta di un nuovo valore di massimo o minimo
    - Il sistema visualizza in una casella di testo il massimo e il minimo per i tre valori monitorati (temperatura, umidità, luminosità). I valori vengono calcolati a partire dalle ore 24 e azzerati ogni giorno

- Flusso alternativo: Nessuno

- Pre Condizioni: L'utente si trova nella pagina principale della dashboard

- Post Condizioni: Vengono visualizzati il valore di minimo e massimo della giornata

- Eccezioni: Non pervengono dati aggiornati dalla rete

- Frequenza Stimata di utilizzo: Alta

- Criticità: Media

A3.2.7 Creazione grafici personalizzati

- Attori: Utente

- Descrizione: Permette di creare fino a tre grafici con i valori scelti in base a tipologia dato, periodo e nodo

- Flusso base:
    - L'utente accede alla schermata "analisi dati"
    - L'utente seleziona quale valore gli interessa, di quale periodo, da quali nodi, tramite un form
    - L'utente sceglie quanti grafici (da uno a tre) visualizzare
    - Il sistema crea il grafico e lo visualizza

- Flusso alternativo:
    - L'utente crea un grafico raffigurante l'andamento delle tre variabili, riferite ad un solo nodo
    - L'utente crea un grafico raffigurante l'andamento di una variabile riferita a uno o più nodi

- Pre Condizioni: L'utente si trova nella schermata di analisi dati

- Post Condizioni: Vengono visualizzati i grafici richiesti

- Eccezioni: L'utente seleziona un periodo che non contiene registrazioni

- Frequenza Stimata di utilizzo: Alta

- Criticità: Alta

A3.2.8 Esportazione su foglio di calcolo

- Attori: Utente
- Descrizione: Permette di esportare i dati selezionati su un foglio di calcolo
- Flusso base:
  - L' utente accede alla schermata "analisi dati"
  - L'utente seleziona quale valore gli interessa, di quale periodo, da quali nodi
  - L'utente avvia, tramite pulsante, l'esportazione
  - Il sistema crea e invia il foglio di calcolo
  - L'utente salva o apre il foglio di calcolo ricevuto
- Flusso alternativo: Nessuno
- Pre Condizioni: L'utente si trova nella schermata di analisi dati
- Post Condizioni: Viene creato un foglio di calcolo
- Eccezioni: Non vi sono dati registrati che soddisfano i parametri selezionati
- Frequenza Stimata di utilizzo: Bassa
- Criticità: Bassa

A3.2.9 Impostazione soglia di allarme

- Attori: Utente
- Descrizione: Permette di impostare due soglie d'allarme; non appena vengono superate, il sistema avvisa l'utente tramite notifiche visive e via e-mail
- Flusso base:
  - L'utente inserisce uno o più indirizzi e-mail per l'invio della notifica d'allarme
  - L'utente seleziona dalla mappa il nodo o i nodi su cui attivare gli allarmi
  - L'utente sceglie quali valori monitorare (temperatura, umidità, luminosità)
  - L'utente imposta il valore di soglia
  - Il sistema riassume gli allarmi impostati in una tabella
- Flusso alternativo:
  - L'utente non imposta alcuna soglia d'allarme
- Pre Condizioni: L'utente si trova nella schermata di gestione allarmi
- Post Condizioni: In caso di superamento delle soglie, il sistema evidenzia nella pagina di "monitoraggio ambientale" il nodo che ha causato l'allarme sulla mappa e sulla griglia e invia la notifica via mail. Inoltre vengono salvati tutti i dati relativi all'allarme (nodo, valore e data) nel database
- Eccezioni: Indirizzo mail inserito non valido
- Frequenza Stimata di utilizzo: Bassa

- Criticità: Alta

A3.2.10 Visualizzazione della configurazione della rete WSN

- Attori: Utente
- Descrizione: Permette di visualizzare l'attuale configurazione topologica della rete WSN
- Flusso base:
    - Il sistema raffigura automaticamente la topologia della rete. Non appena un nodo si aggiunge o si disconnette dalla rete, il sistema aggiorna automaticamente la topologia della WSN
- Flusso alternativo: Nessuno
- Pre Condizioni: L'utente si trova nella schermata "WSN"
- Post Condizioni: Viene creato un'immagine che raffigura la configurazione
- Eccezioni: Un nodo che non ha più inviato le ultime cinque rilevazioni
- Frequenza Stimata di utilizzo: Bassa
- Criticità: Bassa

A3.2.11 Impostazione intervallo di trasmissione dati

- Attori: Utente
- Descrizione: Permette di fissare l'intervallo di tempo di invio dei dati dai nodi al gateway della WSN. Maggiore è l'intervallo, maggiore è il risparmio energetico dei nodi e quindi la loro autonomia nel tempo. Un intervallo troppo basso causa aggiornamenti dei dati sulla dashboard e sul database molto tardivi.
- Flusso base:
    - L'utente inserisce il valore dell'intervallo nelle casella di testo apposita
    - Il sistema applica la modifica a tutta la rete
- Flusso alternativo: L'utente non modifica il valore di default
- Pre Condizioni: L'utente si trova nella schermata "WSN"
- Post Condizioni: I nodi inviano i dati seguendo il nuovo intervallo di tempo stabilito
- Eccezioni: Viene inserito un valore troppo basso (< 1 minuto)
- Frequenza Stimata di utilizzo: Bassa
- Criticità: Bassa

A3.3 Requisiti prestazionali

Il web server e il database devono occupare meno risorse hardware possibile. Le macchine su cui

funzioneranno i servizi avranno basse capacità computazionali, perciò non ci devono essere applicazioni che fanno uso intensivo della CPU.
Il sistema deve avere un tempo di risposta massimo inferiore ai 10 secondi.
I nodi devono garantire l'invio dei dati al tempo prestabilito.
Qualsiasi operazione richiesta ai nodi, deve essere eseguita entro il prossimo intervallo di invio dei dati.

A3.4 Requisiti logici del database

Tutti i dati provenienti dai nodi (temperature, umidità, luminosità, numero nodo e dati della connessione) devono essere memorizzati, insieme all'ora di ricezione da parte del gateway della WSN.
Il database deve essere di tipo relazionale e le transazioni di tipo ACID.

A3.5 Vincoli progettuali

Il database deve fornire i dati in XML.
Tutto il software adoperato e sviluppato deve essere open-source.

A3.6 Attributi del sistema software

Il prodotto consiste di diversi moduli collegati tra loro tramite interfacce ben definite.
Gli elementi principali del sistema sono:

- il web server
- il DBMS
- la dashboard.

A3.6.1 Affidabilità

L'affidabilità del database è critica: deve garantire che ogni dato arrivato della rete sia correttamente memorizzato e accessibile per la consultazione.
Il web server deve essere altrettanto affidabile. In ogni caso, non verrà sottoposto a grandi carichi di lavoro, dato che si presuppone che il numero di accessi contemporanei sia basso, raramente sopra 100.
Se il browser dell'utente va in crash, interrompe l'applicazione dashboard; tuttavia ciò non dovrebbe verificarsi e, comunque, una riapertura del browser permette un rapido riavvio dell'applicazione.

A3.6.2 Disponibilità

Il sistema deve essere disponibile 24 ore su 24 e deve essere accessibile da qualsiasi luogo sia possibile connettersi al web server. In caso di problemi software o hardware, il web server e il database devono riavviarsi da soli laddove possibile.

A3.6.3 Sicurezza

Il web server e il database non devono essere modificabili o cancellabili dagli utenti non autorizzati. I dati inseriti dall'utente (e-mail per le notifiche d'allarme) non devono essere accessibili da utenti non autorizzati.
Il server deve essere protetto da attacchi di qualsiasi tipo provenienti da Internet.

A3.6.4 Manutenibilità

Il sistema non necessita di alcuna attenzione da parte dell'utente.

A3.6.5 Portabilità

Il sistema web server e database è basato su Linux, ed è possibile farlo funzionare senza molte modifiche sotto sistemi Unix e derivati.
La parte utente del sistema, ovvero la dashboard, è utilizzabile su qualsiasi sistema operativo con un browser recente (Opera 6+, Firefox 2+, Safari, IE6+) che supporti Javascript.

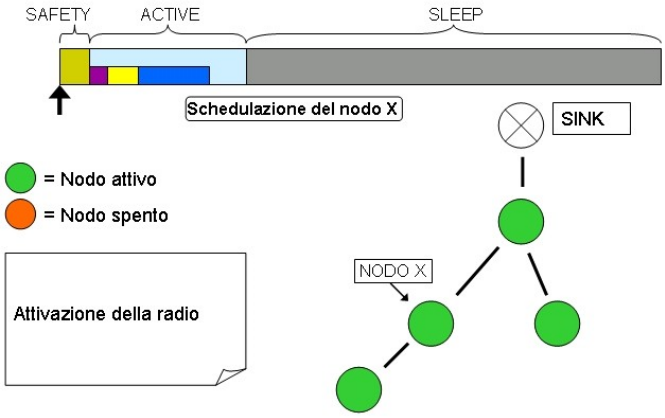
A3.6.6 Ulteriori commenti

Dato che i nodi sono privi di GPS, le loro coordinate vanno registrate manualmente in fase di preparazione della Wireless Sensor Network. Quindi, l'installatore della rete dovrà inserire manualmente nel database longitudine e latitudine di ogni nodo.

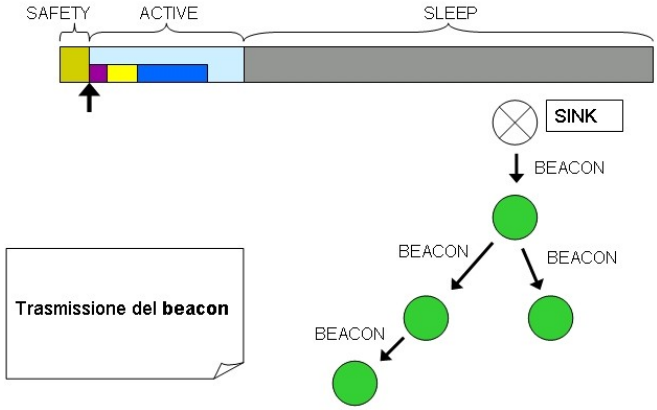
APPENDICE B

SCHEMA DI FUNZIONAMENTO DELLA WSN

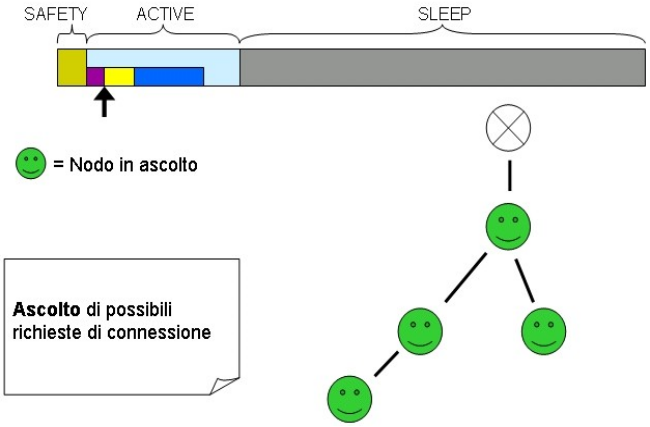
Di seguito, vengono schematizzate la varie fasi attraversate dal “nodo X”, evidenziato nella prima figura, già connesso e operativo nella rete:



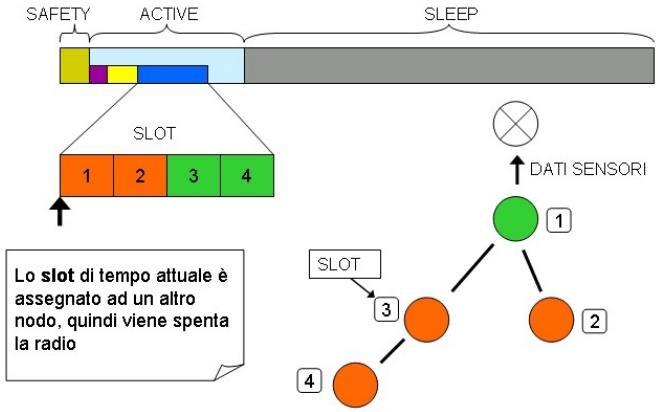
1) Inizio fase Safety: attivazione della radio



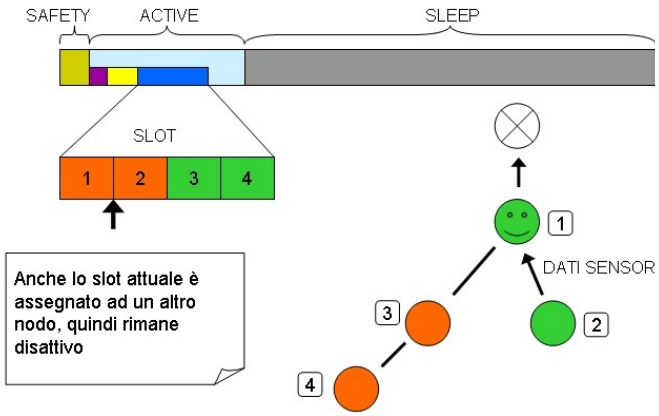
2) Fase Beacon: trasmissione dei beacon



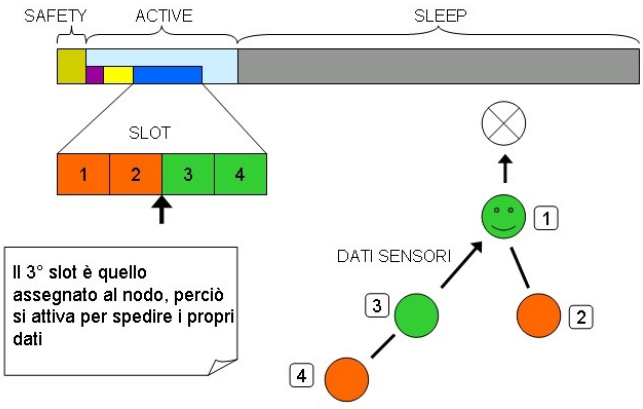
3) Fase Connection: il nodo rimane in ascolto di eventuali messaggi di connessione



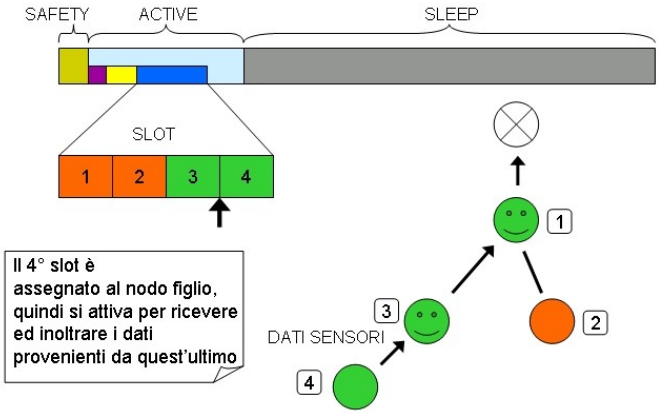
4) Inizio fase Data: nel primo slot, il nodo rimane spento



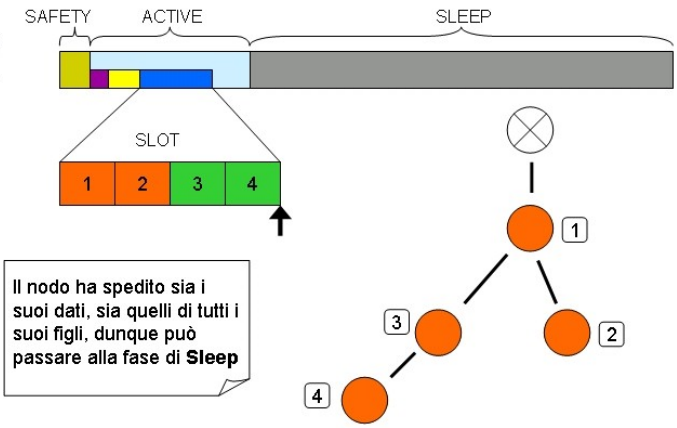
5) Secondo slot: il nodo continua a dormire



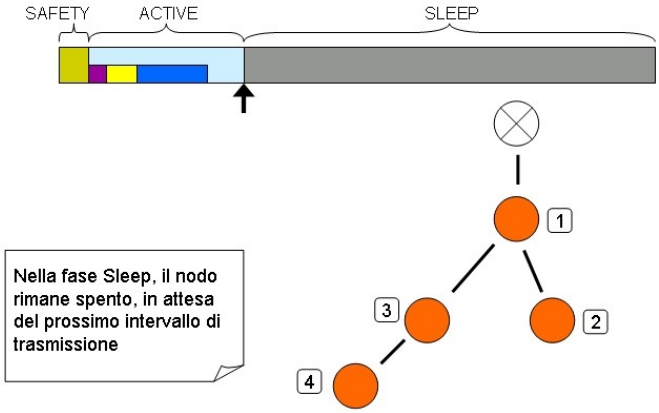
6) Terzo slot: è lo slot assegnato al nodo, perciò trasmette i propri dati verso il sink



7) Nel quarto slot il nodo riceve e inoltra i dati del nodo figlio

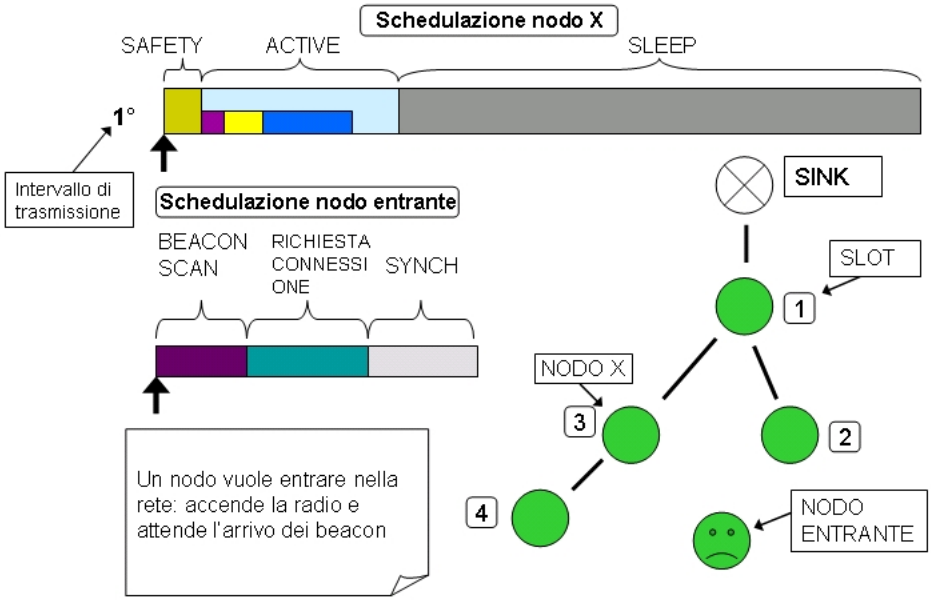


8) Fine degli slot: il nodo torna a dormire

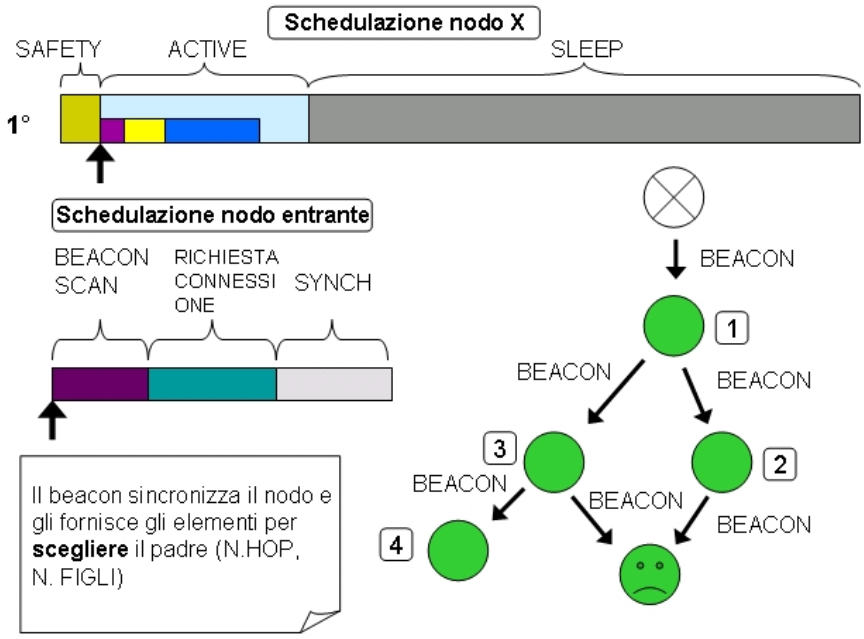


9) Inizio della fase Sleep: il nodo è spento

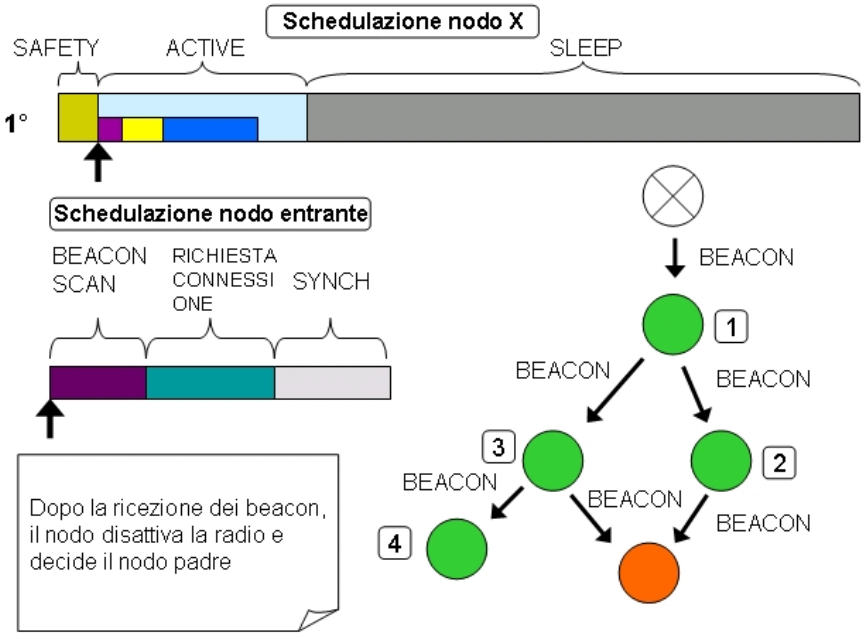
L'ingresso di un nodo nella rete viene riassunto nel seguente schema:



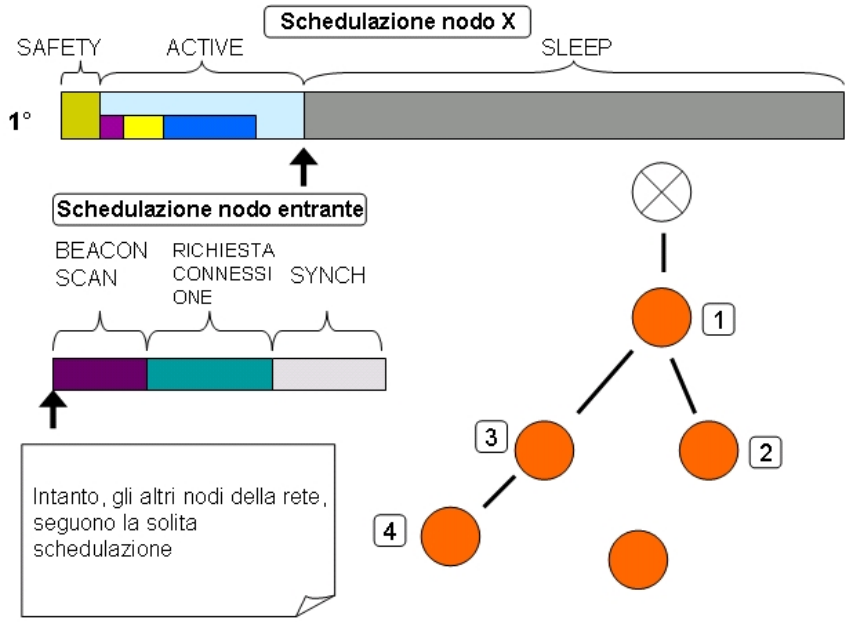
1) Il nodo entrante attiva la radio ed esegue il Beacon Scan



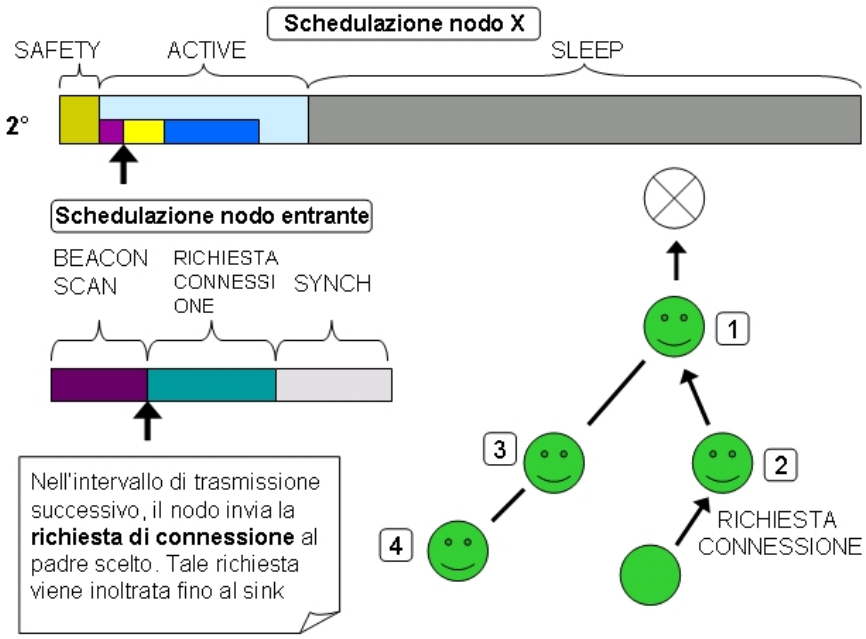
2) Ricezione dei beacon



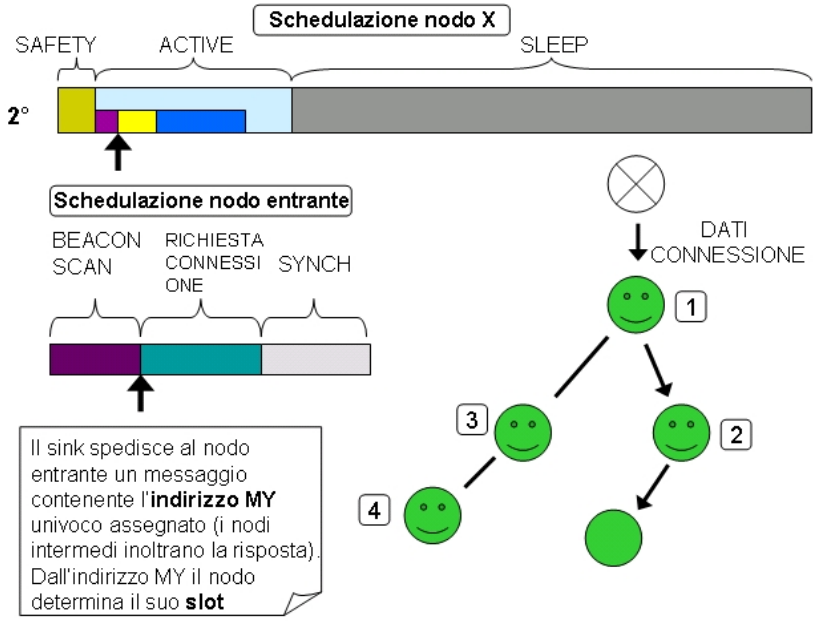
3) Il nodo entrante decide il nodo padre e spegne la radio



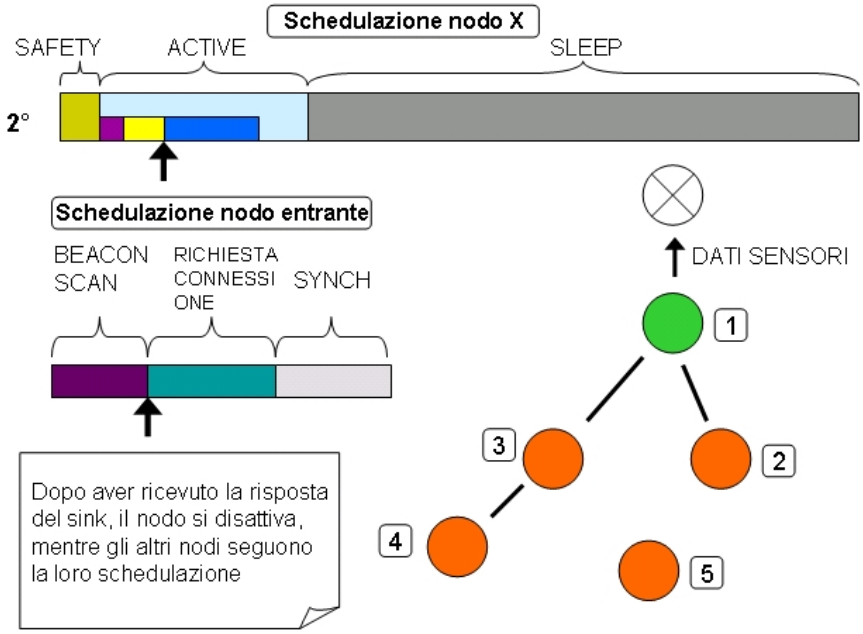
4) Gli altri nodi continuano con la loro schedulazione



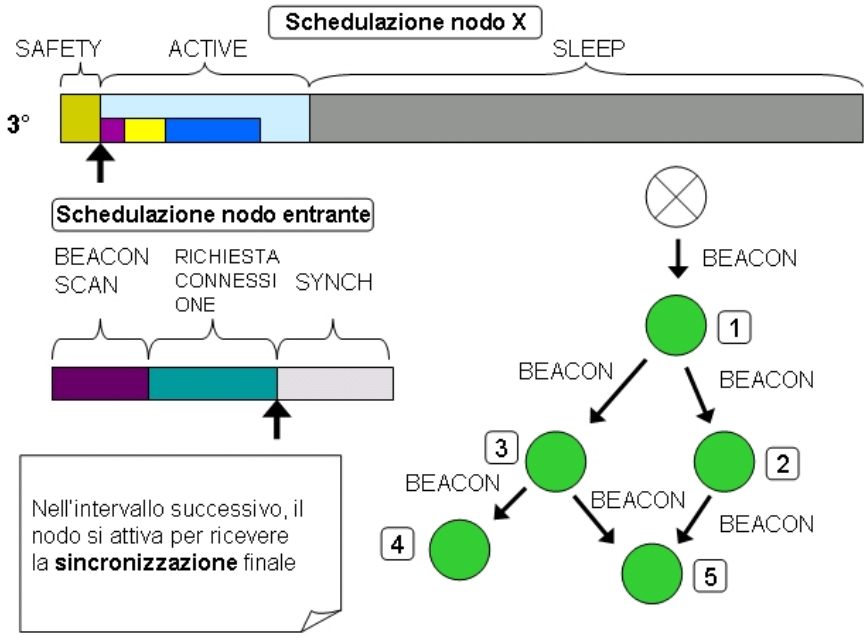
5) Nell'intervallo di trasmissione successivo, il nodo invia la richiesta di connessione



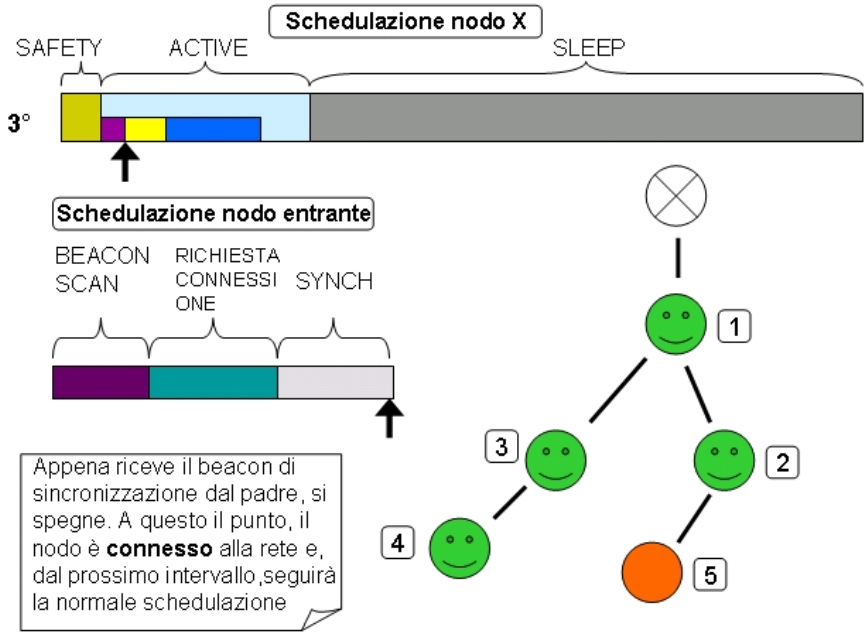
6) Il sink invia la risposta di connessione al nodo



7) Il nodo si spegne e attende il prossimo intervallo



8) Nel terzo intervallo, il nodo attende il beacon di sincronizzazione



9) Dopo aver ricevuto il beacon, il nodo si spegne e dal prossimo intervallo è di fatto un nuovo nodo della rete

APPENDICE C

FORMATO DEI MESSAGGI DELLA WSN

Nella Tabella C.1 viene descritto il formato dei messaggi che i nodi si scambiano.

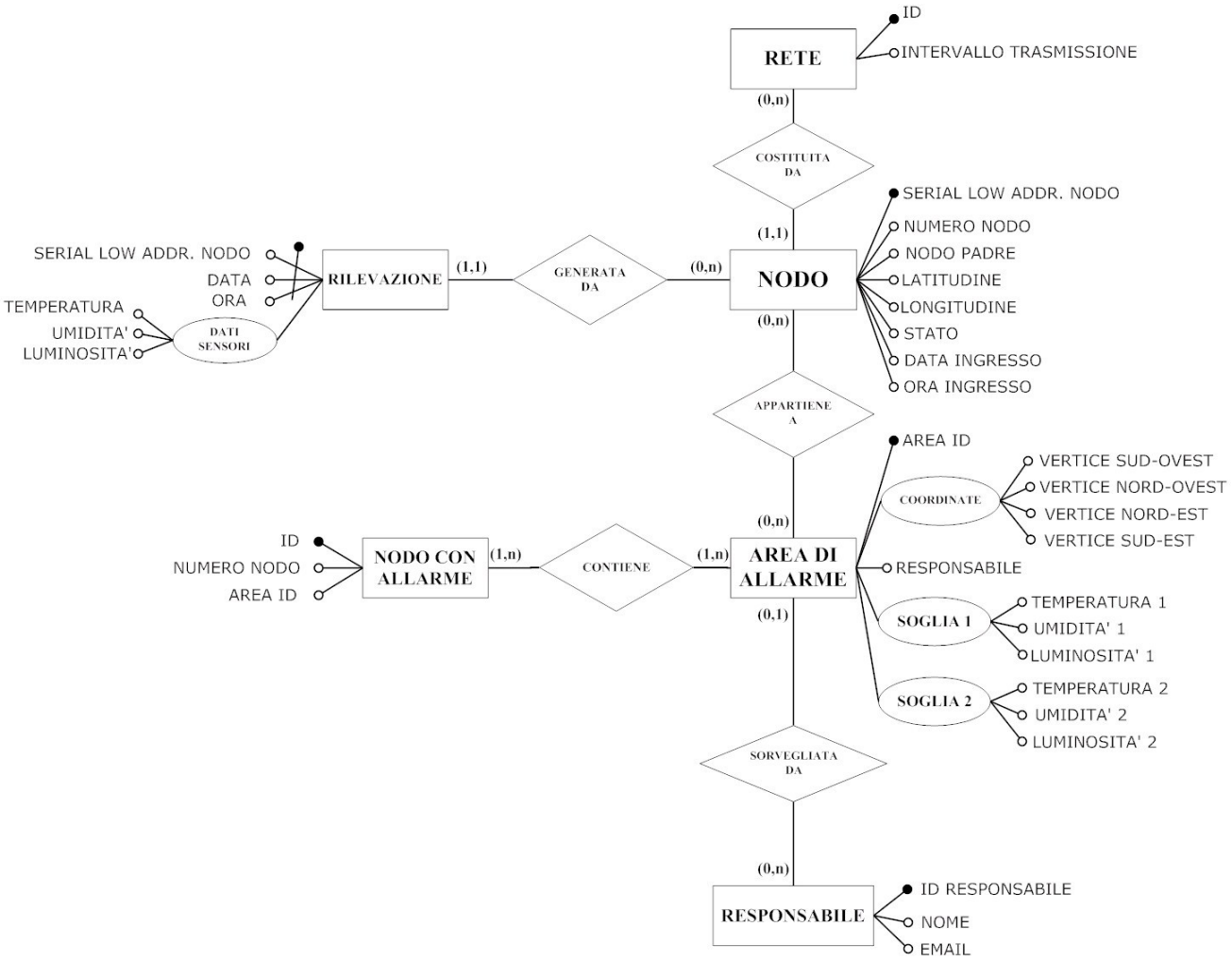
TIPO MSG	DESCRIZIONE	ID	CAMPO 1	CAMPO 2	CAMPO 3	CAMPO 4	CAMPO 5
ask association	nodo orfano richiede al nodo padre di diventare un suo figlio	**a**	Serial Low di nodo entrante	MY del padre scelto dal nodo entrante	Last Forwarder = ultimo nodo per cui è transitato questo msg	contatore n. hop	
	MESSAGGIO ("#" = carattere di spazio)						
	#a#SL_ORFANO#MY_PADRE#LAST_FORWARDER#N_HOP#@						
send data	contiene i dati rilevati dai sensori	**b**	MY del nodo che ha rilevato i dati	numero di sequenza	luminosità (valore campionato)	umidità (valore campionato)	temperatura (valore campionato)
	MESSAGGIO ("#" = carattere di spazio)						
	#b#MY#N_SEQ#LUM#HUM#TEMP#@						
gateway response	risposta del gateway alla richiesta di ingresso nella rete di un nodo orfano	**c**	MY del nodo padre	Serial Low del nodo orfano entrante	MY assegnato al nodo orfano	n. hop dal nodo orfano al gateway	
	MESSAGGIO ("#" = carattere di spazio)						
	#c#MY_PADRE#SL_ORFANO#MY_ASSEGNATO#N_HOP#@						
answer orphan	msg destinato al nodo orfano, contenente dati per l'ingresso nella rete	**d**	Serial Low di nodo entrante	MY assegnato al nodo entrante	n. hop da nodo entrante a gateway		
	MESSAGGIO ("#" = carattere di spazio)						
	#d#SL_ORFANO#MY_ASSEGNATO#N_HOP#@						
beacon	messaggio beacon	**e**	MY del nodo che ha trasmesso beacon	n. hop dal gateway	n. figli	intervallo di trasmissione	
	MESSAGGIO ("#" = carattere di spazio)						
	#e#MY_PADRE#MY#N_HOP#N_FIGLI#INTERVALLO#@						

Tabella C.1: formato dei messaggi

APPENDICE D

SCHEMA DEL DATABASE

Lo schema del database del gateway è il seguente:



Lo schema logico è così definito:

```
1. CREATE TABLE controlled_areas (
        id INTEGER PRIMARY KEY,
        lat_so REAL,
        lng_so REAL,
        lat_no REAL,
        lng_no REAL,
        lat_ne REAL,
        lng_ne REAL,
        lat_se REAL,
        lng_se REAL,
        owner TEXT,
        al_t1 REAL,
```

```
        al_t2 REAL,
        al_h1 REAL,
        al_h2 REAL,
        al_l1 REAL,
        al_l2 REAL,
        t1_major INTEGER,
        t2_major INTEGER,
        h1_major INTEGER,
        h2_major INTEGER,
        l1_major INTEGER,
        l2_major INTEGER
  )

2. CREATE TABLE nodo (
        serial_low TEXT,
        node_number INTEGER,
        parent_node INTEGER,
        latitud REAL,
        longitud REAL,
        status TEXT,
        date CURRENT_DATE,
        time CURRENT_TIME
  )

3. CREATE TABLE nodes_areas (
        id INTEGER PRIMARY KEY,
        node_number INTEGER,
        area_id INTEGER
  )

4. CREATE TABLE owner (
        id INTEGER PRIMARY KEY,
        name TEXT,
        email TEXT
  )

5. CREATE TABLE network (
        id INTEGER PRIMARY KEY,
        interval INTEGER,
        date NOT NULL DEFAULT CURRENT_DATE,
        time NOT NULL DEFAULT CURRENT_TIME
  )

6. CREATE TABLE rilevazione (
        node_number INTEGER,
        date NOT NULL DEFAULT CURRENT_DATE,
        time NOT NULL DEFAULT CURRENT_TIME,
        temp INTEGER,
        hum INTEGER,
        lum INTEGER,
        UNIQUE(node_number, date, time)
  )
```

```
7. CREATE TRIGGER new_area
   AFTER INSERT ON controlled_areas
        WHEN (SELECT owner.name FROM controlled_areas, owner WHERE NEW.owner =
        owner.name ) IS NULL
                BEGIN
                        INSERT INTO owner (name) VALUES (NEW.owner);
                END;

8. CREATE TRIGGER new_interval
   BEFORE INSERT ON network
        BEGIN
                DELETE FROM network WHERE id = (SELECT id FROM network WHERE id =
                (SELECT MAX (id) FROM network));
        END;

9. CREATE TRIGGER update_area
   AFTER UPDATE OF owner ON controlled_areas
        WHEN (SELECT owner.name FROM controlled_areas, owner WHERE owner.name =
        NEW.owner) IS NULL
                BEGIN
                        INSERT INTO owner (name) VALUES (NEW.owner);
                END;

10. CREATE TRIGGER delete_area
    AFTER DELETE ON controlled_areas
        BEGIN
                DELETE FROM nodes_areas WHERE area_id = OLD.id;
        END;
```

GLOSSARIO

A/D
Analog/Digital. Sigla per indicare un convertitore analogico/digitale, che converte segnali analogici a variazione continua in codice binario.

AJAX
Asynchronous JavaScript and XML. È un insieme di tecnologie per lo sviluppo web, usate per la creazione di siti web interattivi e di Rich Internet Application.

ARDUINO
Piattaforma open-source di physical computing basata su microcontrollore RISC (vedasi *RISC*) Atmel ATmega.

CSMA/CA
Carrier Sense Multiple Access/ Collision Avoidance. Meccanismo di regolazione di accesso al mezzo trasmissivo, in cui tutte le stazioni possono trasmettere (Multiple Access), ma soltanto dopo aver verificato che il canale non sia occupato (Carrier Sense). Inoltre le stazioni devono evitare trasmissioni contemporanee che porterebbero ad eventuali collisioni (Collision Avoidance).

CTS
Clear To Send. Frame spedito in risposta ad un RTS. Garantisce la disponibilità del canale per un tempo specificato.

DBMS
Database Management System. È un insieme di sistemi software che consente la creazione e la manipolazione efficiente di banche dati.

DSSS
Direct-Sequence Spread Spectrum. Un sistema di modulazione del segnale, utilizzato nelle prime specifiche dell'802.11 e nell'802.15.4.

EEML
Extended Environments Markup Language. È un protocollo per la condivisione di dati di sensori tra ambienti remoti, sia virtuali che fisici.

GATEWAY
Vedasi *sink.*

GPRS
General Packet Radio Service. Standard per la trasmissione dati nella rete telefonica cellulare.

GPS
Global Position System. Sistema di rilevamento della posizione tramite una serie di satelliti.

IEEE
Institute of Electrical and Electronics Engineers. Ente di standardizzazione.

ISM
Industrial, Scientific and Medical. Nome assegnato dall'Unione Internazionale delle

Telecomunicazioni (ITU) ad un insieme di porzioni dello spettro elettromagnetico riservate alle applicazioni radio non commerciali, per uso industriale, scientifico e medico.

MAC
Medium Access Control. Funzione delle reti IEEE che regolamenta l'accesso al mezzo di trasmissione e l'utilizzo del canale radio.

MANET
Mobile Ad-hoc NETwork. È definita come un sistema autonomo di router (detti anche nodi) mobili connessi mediante collegamenti wireless. Le reti Ad-Hoc vengono costruite all'occorrenza ed utilizzate in ambienti estremamente dinamici, non necessariamente con l'aiuto di una infrastruttura già esistente.

MEMS
Micro Electro-Mechanical Systems. Dispositivi miniaturizzati composti da parti meccaniche e circuiti elettronici, tipicamente su un chip semiconduttore, con dimensioni da decine fino a poche centinaia di micrometri. Applicazioni comuni per MEMS includono sensori, attuatori e unità di controllo di processo.

MOTE
Nome con cui è possibile identificare il nodo dotato di sensori di una rete WSN.

OSI
Open Systems Interconnection. Rappresentazione logica degli standard delle reti, i quali vengono divisi in livelli gerarchici: il primo livello contiene le regole di trasmissione dei bit direttamente sul canale fisico (Physical Layer), mentre l'ultimo gestisce i dati da passare ai processi del sistema operativo.

PHP
PHP: Hypertext Preprocessor. È un linguaggio di scripting interpretato, con licenza open source, originariamente concepito per la realizzazione di pagine web dinamiche.

PHY
Sigla usata per indicare il livello Fisico (Physical Layer) del modello OSI.

QoS
Quality of Service, cioè qualità di servizio. Indica il rispetto di determinati canoni di qualità in un servizio offerto, come banda garantita o latenza limitata.

RIA
Rich Internet Application. È un'applicazione web avente le caratteristiche e le funzionalità di un programma desktop.

RISC
Reduced Instruction Set Computer. Indica una filosofia di progettazione di architetture per microprocessori formate da un set di istruzioni contenente istruzioni in grado di eseguire operazioni semplici che possono essere eseguite in tempi simili.

RSSI
Received Signal Strength Indication. È una misura della potenza presente in un segnale radio ricevuto.

RTS
Request To Send. Frame spedito da una stazione per prenotare l'accesso al canale di trasmissione. Si applica solo per frame sopra una certa dimensione, specificata nell'RTS Threshold.

RTT
Round Trip Time. È una misura del tempo impiegato da un pacchetto di dimensione trascurabile per viaggiare da un nodo della rete ad un altro e tornare indietro.

SINK
Nodo della WSN in cui confluiscono i dati dei mote. Solitamente ha anche compiti di gestione della rete e di inoltro dei dati verso un sistema remoto. In quest'ultimo caso viene definito anche gateway o base station.

SOS
Sensor Operating System. Acronimo usato per indicare il sistema operativo di un nodo di una wsn.

SQUIDBEE
Nodo wireless open-hardware e open-source, compatibile con l'IEEE 802.15.4 e dotato di sensori; è basato sulla scheda Arduino (vedasi *Arduino)* e utilizza il modulo radio XBee (vedasi *XBee*).

TDMA
Time Division Multiple Access. È un metodo di accesso al mezzo di trasmissione: consente a più utenti di condividere lo stesso canale attraverso una suddivisione del segnale in diversi slot di tempo, ciascuno assegnati ad un utente.

TTL
Time To Live. Parametro di un pacchetto della rete che definisce un tempo limite alla sua validità.

UWB
Ultra-WideBand. Tecnologia per trasmettere e ricevere segnali mediante l'utilizzo di impulsi di energia in radiofrequenza di durata estremamente ridotta.

WI-FI
Abbreviazione di Wireless e Fidelity. Indica i dispositivi che possono collegarsi a reti locali senza fili basate sulle specifiche IEEE 802.11.

WSN
Wireless Sensor Network. È una rete wireless costituita da nodi autonomi dotati di sensori per il monitoraggio di ambienti o sistemi.

XBEE
Modulo radio compatibile con lo standard IEEE 802.15.4.

XML
eXtensible Markup Language. È un metalinguaggio utilizzato per creare nuovi linguaggi, atti a descrivere documenti strutturati.

ZIGBEE
Nome di una specifica per un insieme di protocolli di comunicazione ad alto livello che utilizzano antenne a bassa potenza e basato sullo standard IEEE 802.15.4 per Wireless Personal Area Network (WPAN).

BIBLIOGRAFIA

[1] A. Kansal and M.B. Srivastava, “An Environmental Energy Harvesting Framework for Sensor Networks”. In Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), Seoul, Korea, August 2003.

[2] Mauricio Castillo-Effen, Daniel H. Quintela, Ramiro Jordan, ” Wayne Westhoff and Wilfrido Moreno. Wireless Sensor Networks for Flash-Flood Alerting”. In Proceedings of the Fifth IEEE International Caracas Conference on Devices, Circuits and Systems, Dominican Republic, Nov.3-5, 2004.

[3] M. Youssef, A. Yousif, N. El-Sheimy, A. Noureldin, “A Novel Earthquake Warning System based on Virtual MIMO-Wireless Sensor Networks”. In Electrical and Computer Engineering, 2007. CCECE 2007. Canadian Conference, 22-26 April 2007 Page(s):932 – 935.

[4] Elizabeth Basha and Daniela Rus, “Design of Early Warning Flood Detection Systems for Developing Countries”. In Proceedings of the Conference on Information and Communication Technologies and Development, Bangalore, India, December, 2007.

[5] H. G. Goh, M. L. Sim, and H. T. Ewe, "Agricultural Monitoring using Wireless Sensor Network and Mobile Internet Application". In Sensor Network and Configuration: Fundamentals, Techniques, Platforms, and Experiments, N. P. Mahalik, Ed. Germany: Springer-Verlag, 2006.

[6] Seong-eun Yoo, Jae-eon Kim, Taehong Kim, Sungjin Ahn, Jongwoo Sung, Daeyoung Kim, “A2S: Automated Agriculture System based on WSN”. In ISCE2007, Dallas, Texas, 20~23 June, 2007.

[7] Hui Liu, Zhijun Meng, Shuanghu Cui, “A Wireless Sensor Network Prototype for Environmental Monitoring in Greenhouses”. In Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference, 21-25 Sept. 2007. Page(s):2344 – 2347.

[8] Aline Baggio, “Wireless sensor networks in precision agriculture”. In Proc. ACM Workshop Real-World Wireless Sensor Networks, 2005.

[9] Daan Goense, John Thelen, “Wireless Sensor Networks for Precise Phytophthora Decision Support ”. In Paper number 053099, 2005 ASAE Annual Meeting.

[10] Thelen J., Goense D., Langendoen K., “Radio Wave Propagation in Potato Fields”. In 1st workshop on wireless network measurement, Riva del Garda, Italy, April 2005.

[11] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, John Anderson, “Wireless Sensor Networks for Habitat Monitoring”. In Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, 2002.

[12] Wang, Yang Huang, Liusheng Wu, Junmin Xu, Hongli, “ Wireless Sensor Networks for intensive irrigated agriculture”. In Consumer Communications and Networking Conference, 2007. CCNC 2007. 4th IEEE.

[13] R. Cardell-Oliver, K. Smettem, M. Kranz, and K. Mayer, “A Reactive Soil Moisture Sensor Network: Design and Field Evaluation,” In Distributed Sensor Networks, vol. 1, no. 2, pp.149-162, 2005.

[14] M. Minami, T. Morito, H. Morikawa, and T. Aoyama, “Solar biscuit: A battery-less Wireless Sensor Network system for environmental monitoring applications”. In The 2nd International Workshop on Networked Sensing Systems, 2005.

[15] Barrenetxea G., Ingelrest F., Schaefer G., Vetterli M., Couach O., Parlange M., “SensorScope: Out-of-the-Box Environmental Monitoring”. In Information Processing in Sensor Networks, 2008. IPSN '08. Publication Date: 22-24 April 2008 On page(s): 332-343. ISBN: 978-0-7695-3157-1.

[16] Thomas Schmid, Henri Dubois-Ferriere, Martin Vetterli, “SensorScope: Experiences with a Wireless Building Monitoring Sensor Network”. In Workshop on Real-World Wireless Sensor Networks (REALWSN'05), 2005.

[17] “SensorScope”, 2008, http://sensorscope.epfl.ch
[18]“Camalie Net Wireless Sensing”, 2008, http://camalie.com/WirelessSensing/WirelessSensors.htm
[19] Mark Holler. High Density, Multiple Depth, “Wireless Soil Moisture Tension Measurements for Irrigation Management”. Crossbow Technologies. 2008.
[20] B. Peters, ‘‘Sensing Without wires: Wireless Sensing Solves Many Problems, But Introduces a Few of Its Own,’’ Machine Design, Penton Media, Cleveland, OH, http://www.machinedesign.com/ASP/viewSelectedArticle.asp?strArticleId=57795&str-&strSite=MDSite&Screen = & CURRENTISSUE&CatID=3
[21] M. C. Vuran, I. F. Akyildiz, “Spatial Correlation-based Collaborative Medium Access Control in Wireless Sensor Networks”. In IEEE/ACM Transactions on Networking, June 2006.
[22] D. Ferrara, et. al., “MACRO: An Integrated MAC/Routing Protocol for Geographical Forwarding in Wireless Sensor Networks,”. In Proc. IEEE Infocom ’05, vol. 3, pp. 1770 - 1781, March 2005.
[23] J. Yuan, Z. Li, W. Yu, B. Li, “A Cross-Layer optimization framework for multicast in multi-hop wireless networks wireless internet”. In Proc. WICON ’05, pp. 47 - 54, July 2005.
[24] M. Chiang, “Balancing transport and physical Layers in wireless multihop networks: jointly optimal congestion control and power control,” IEEE JSAC, vol. 23, no. 1, pp. 104-116, Jan. 2005.
[25] R. Madan, S. Cui; S. Lall, A. Goldsmith, “Cross-layer design for lifetime maximization in interference-limited wireless sensor networks”. In Proc. IEEE INFOCOM ’05, vol. 3, pp. 1964 - 1975, March 2005.
[26] T. Melodia, M. C. Vuran, D. Pompili, "The State of the Art in Cross-layer Design for Wireless Sensor Networks". In Proc. of EuroNGI Workshops on Wireless and Mobility. Springer Lecture Notes in Computer Science 3883, Como, Italy, July 2005.
[27] Y. Xu, J. Heidemann, D. Estrin, “Geography-informed Energy Conservation for Ad Hoc”. In Proc. ACM MobiCom 2001, pp. 70 – 84. Rome, 2001.
[28] B. Chen, K. Jamieson, H. Balakrishnan, R. Morris. “Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks”. In ACM Wireless Networks, Vol. 8, N. 5, September 2002.
[29] A. Cerpa, D. Estrin, “Ascent: Adaptive Self-Configuring Sensor Network Topologies”. In Proc. IEEE INFOCOM 2002.
[30] C. Schurgers, V. Tsiatsis, M. B. Srivastava, “STEM: Topology Management for Energy Efficient Sensor Networks”. In IEEE Aerospace Conference '02, Big Sky, MT, March 10-15, 2002.
[31] X. Yang, N. Vaidya, “A Wakeup Scheme for Sensor Networks: Achieving Balance between Energy Saving and End-to-end Delay ”. In Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2004), pp. 19-26, 2004.
[32] J. Rabaey, J. Ammer, T. Karalar, S. Li, B. Otis, M. Sheets, T. Tuan, “Picoradios for wireless sensor networks: the next challenge in ultra-low-power design”. In Proc. International Solid-State Circuits Conference, San Francisco, CA, February 3-7, 2002.
[33] L. Gu, J. Stankovic, “Radio-Triggered Wake-up for Wireless Sensor Networks”. In Real-Time Systems Journal, Vol. 29, pp. 157-182, 2005.
[34] A.G. Ruzzelli, R. Jurdak, and G.M.P. O'Hare. "On the RFID Wake-up Impulse for Multi-hop Sensor Networks". In proceedings of 1st ACM Workshop on Convergence of RFID and Wireless Sensor Networks and their Applications (SenseID) at the Fifth ACM Conference on Embedded Networked Sensor Systems (ACM SenSys 2007), Sydney, Australia. November, 2007.
[35] R. Jurdak, A.G. Ruzzelli, and G.M.P. O'Hare, "Multi-hop RFID Wake-up Radio: Design, Evaluation and Energy Tradeoffs". In Proceedings of the 17TH International Conference on Computer Communications AND Networks (ICCCN), August, 2008.
[36] P. Buonadonna, D. Gay, J. Hellerstein, W. Hong, S. Madden, “TASK: Sensor Network in a Box”. In Proc. European Workshop on Sensor Networks (EWSN 2005), January 2005.

[37] A. Keshavarzian, H. Lee, L. Venkatraman, ”Wakeup Scheduling in Wireless Sensor Networks”. In Proc. ACM MobiHoc 2006, pp. 322-333, Florence (Italy), May 2006.
[38] G. Anastasi, M. Conti, M. Di Francesco and A. Passarella, “An Adaptive and Low-latency Power Management Protocol for Wireless Sensor Networks”. In Proc. of the 4-th ACM International Workshop on Mobility Management and Wireless Access (MobiWac 2006), Torremolinos (Spain), October 2, 2006.
[39] B. Hohlt, L. Doherty, E. Brewer, “Flexible Power Scheduling for Sensor Networks“. In Proc. ACM Workshop on Information Processing in Sensor Networks (ISPN 2004), Berkeley (USA), April 26-27, 2004.
[40] R. Zheng, J. Hou, L. Sha, “Asynchronous Wakeup for Ad Hoc Networks”. In Proc. ACM MobiHoc 2003, pp 35- 45, Annapolis (USA), June 1-3, 2003.
[41] V. Paruchuri, S. Basavaraju, R. Kannan, S. Iyengar, “Random Asynchronous Wakeup Protocol for Sensor Networks”. In Proc. IEEE Int’l Conf. On Broadband Networks (BROADNETS 2004), 2004.
[42] V. Rajendran, K. Obracza, J. J. Garcia-Luna Aceves, “Energy-efficient, Collision-free Medium AccessControl for Wireless Sensor Networks”. In Proc. ACM SenSys 2003, Los Angeles (USA), November 2003.
[43] J. Polastre, J. Hill and D. Culler, “Versatile Low Power Media Access for Sensor Networks”. In Proc. of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys), November 3-5, 2004.
[44] W. Ye, J. Heidemann and D. Estrin, “Medium Access Control With Coordinated Adaptive Sleeping for Wireless Sensor Networks”. In IEEE/ACM Transactions on Networking, Volume: 12, Issue: 3, Pages: 493-506, June 2004.
[45] G. Lu, B. Krishnamachari and C.S. Raghavendra, “An Adaptive Energy-efficient and Low-latency Mac for Data Gathering in Wireless Sensor Networks”. In Proc. of 18th International Parallel and Distributed Processing Symposium, Pages: 224, 26-30 April 2004.
[46] D. Mirza, M. Owrang, C. Schurgers, "Energy-efficient Wakeup Scheduling for Maximizing Lifetime of IEEE 802.15.4 Networks". In Proc. International Conference on Wireless Internet (WICON’05), Budapest (Hungary), pp. 130 - 137, July 2005.
[47] I. Rhee, A. Warrier, M Aia, J. Min, “Z-MAC: a Hybrid MAC for Wireless Sensor Networks“. In Proc. ACM SenSys 2005, S. Diego (USA), November 2005.
[48] Michael Buettner, Gary Yee, Eric Anderson, Richard Han, “X-MAC: A Short Preamble MAC Protocol For Duty-Cycled Wireless Sensor Networks”. In Proceedings of the 4th international conference on Embedded networked sensor systems. Pages: 307 - 320. Year of Publication: 2006. ISBN:1-59593-343-3.
[49] Y. Yu, D. Estrin, and R. Govindan, “Geographical and Energy-Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks”. In UCLA Computer Science Department Technical Report, UCLA-CSD TR-01-0023, May 2001.
[50] V. Rodoplu and T.H. Ming, “Minimum energy mobile wireless networks”. In IEEE Journal of Selected Areas in Communications, Vol. 17, No. 8, pp. 1333-1344, 1999.
[51] L. Li, J. Y. Halpern, “Minimum-energy mobile wireless networks revisited”. In IEEE International Conference on Communications (ICC’01), Helsinki, Finland, June 2001.
[52] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, “Energy-efficient Communication Protocol for Wireless Microsensor Networks”. In Proc. Hawaii International Conference on System Sciences (HICSS-34), January 2000.
[53] S. Lindsey, C. Raghavendra, “PEGASIS: Power- Efficient Gathering in Sensor Information Systems”. In IEEE Aerospace Conference Proceedings, 2002, Vol. 3, 9-16 pp. 1125-1130.
[54] A. Manjeshwar, D. P. Agrawal, “TEEN: A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks”. In Proc. International Parallel and Distributed Processing Symposium (IPDPS'01) Workshops, 2001.
[55] A. Manjeshwar and D. P. Agrawal, “APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks”. In Proc. Int’l. Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing,

April 2002.
[56] C. Intanagonwiwat, R. Govindan and D. Estrin,“Directed diffusion: A scalable and robust communication paradigm for sensor networks”. In Proc. ACM MobiCom 2000, August 2000.
[57] C. Schurgers and M.B. Srivastava, “Energy efficient routing in wireless sensor networks”. In MILCOM Proceedings on Communications for Network-Centric Operations: Creating the Information Force, McLean, VA, 2001.
[58] C. Rahul, J. Rabaey, “Energy Aware Routing for Low Energy Ad Hoc Sensor Networks”. In Proc. of IEEE Wireless Communications and Networking Conference (WCNC), Orlando, FL, 2002.
[59] W. Heinzelman, J. Kulik, and H. Balakrishnan, “Adaptive Protocols for Information Dissemination in Wireless Sensor Networks”. In Proc. ACM/IEEE MobiCom '99, pp. 174-85, Seattle (USA), August, 1999.
[60] Y. Yao, J. Gehrke, ‘‘The Cougar Approach to In-Network Query Processing in SensorNetworks,’’. In SIGMOD Record, Vol. 31, No. 1, Mar. 2002.
[61] Narayanan Sadagopan, Bhaskar Krishnamachari, Ahmed Helmy. “Active query forwarding in sensor networks”. In AdHoc Networks Journal, 3(1): 91–113, 2005.
[62] J.-H. Chang and L. Tassiulas, “Maximum Lifetime Routing in Wireless Sensor Networks”. In Proc. of the Advanced Telecommunications and Information Distribution Research Program (ATIRP'2000), College Park, MD, March 2000.
[63] K. Kalpakis, K. Dasgupta and P. Namjoshi, “Maximum lifetime data gathering and aggregation in wireless sensor networks”. In Proc. of IEEE International Conference on Networking (NETWORKS '02), Atlanta, GA, August 2002.
[64] A. C. F. Ye, S. Lu and L. Zhang. “A Scalable Solution to Minimum Cost Forwarding in Large Sensor Networks”. In Proceedings of the IEEE International Conference on Computer Communication and Networks (ICCCN), pp. 304–309. 15–17 October 2001.
[65] K. Sohrabi, J. Pottie, “Protocols for self-organization of a wireless sensor network”. In IEEE Personal Communications, Volume 7, Issue 5, pp 16-27, 2000.
[66] Kemal Akkaya, Mohamed Younis, “An Energy-Aware QoS Routing Protocol for Wireless Sensor Networks”. In Proceedings of the 23rd International Conference on Distributed Computing Systems table of contents. Page: 710. Year of Publication: 2003.
[67] T. He, J. A Stankovic, C. Lu, T. Abdelzaher, “SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks”. In Proc. of the 23rd IEEE International Conference on Distributed Computing Systems (ICDCS'03), Providence, RI, May 2003.
[68] Tijs van Dam, and Koen Langendoen, “An adaptive-efficient MAC protocal for wireless sensor networks”. ACM, 2003.
[69] “Lighttpd vs nginx”,2008, http://www.wikivs.com/wiki/Lighttpd_vs_nginx
[70] “nGinx vs Lighttpd - web server of choice”, 2007, http://blog.grik.net/2007/08/nginx-vs-lighttpd-web-server-of-choice.html
[71] “Benching Lighttpd vs Nginx (static files)”, 2007, http://superjared.com/entry/benching-lighttpd-vs-nginx-static-files/
[72] “Lighttpd - the good and the bad”, 2007, http://spencer.ir/management/lighttpd-the-good-and-the-bad
[73] “JavaScript Toolkit Comparison”, 2006, http://www.jasig.org/wiki/display/UP3/Javascript+Toolkit+Comparison
[74] Martin A Hebel, Ralph F Tate, Dennis G Watson. “Results of Wireless Sensor Network Transceiver Testing for Agricultural Applications”. In 2007 ASAE Annual Meeting 073077.
[75] Kazem Sohraby, Daniel Minoli, Taieb Znati, “Wireless Sensor Networks: Technology, Protocols, and Applications”, Wiley-Interscience, 2007. ISBN 0471743003, 9780471743002.
[76] De Nardis, L. Di Benedetto., “Overview of the IEEE 802.15.4/4a standards for low data rate Wireless Personal Data Networks”. In Positioning, Navigation and Communication, 2007. WPNC '07. 4th Workshop. Publication Date: 22-22 March 2007. On page(s): 285-289.
[77] G. Anastasi, M. Conti, M. Di Francesco, A. Passarella, “ How to Prolong the Lifetime of

Wireless Sensor Networks". In Mobile Ad Hoc and Pervasive Communications. American Scientific Publishers, 2006.
[78] "Extended Environments Markup Language (EEML)", 2008, http://www.eeml.org/
[79] "Mica Motes", 2008, http://www.xbow.com
[80] "Spec chip", 2007, http://www.jlhlabs.com/jhill_cs/spec/
[81] "BTnodes - A Distributed Environment for Prototyping Ad Hoc Networks", 2007, http://www.btnode.ethz.ch/
[82] "Energy Efficient Sensor Networks", 2007, http://www.eyes.eu.org
[83] "Scatterweb", 2008, http://cst.mi.fu-berlin.de/projects/ScatterWeb/index.html
[84] "Real-Time Wireless Sensor Network Platforms", 2007, http://www.ece.cmu.edu/firefly/
[85] "TelosB", 2008, http://www.xbow.com/Products/productdetails.aspx?sid=252
[86] "Millennial Net: Wireless Sensor Mesh Networking", 2008, http://www.millennial.net/
[87] "ZigBee Wireless Networking Systems", 2008, http://www.ember.com
[88] "TinyOS – An open-source OS for the networked sensor regime", 2008, http://www.tinyos.net
[89] "The Contiki Operating System", 2008, http://www.sics.se/contiki
[90] "Nano-RK", 2007, http://www.nanork.org
[91] "MANTIS", 2008, http://mantis.cs.colorado.edu/tikiwiki/tiki-index.php
[92] W. B. Heinzelman, A. L. Murphy, H. S. Carvalho, M. A. Perillo, ''Middleware to Support Sensor Network Applications''. In IEEE Network, Jan.–Feb. 2004, pp. 6–14.
[93] X. Yu, K. Niyogi, S. Mehrotra, N. Venkatasubramanian, ''Adaptive Middleware for Distributed Sensor Environments''. In IEEE Distributed Systems Online, May 2003.
[94] P. B. Gibbons, B. Karp, Y. Ke, S. Nath, S. Seshan, ''IrisNet: An Architecture for a Worldwide Sensor Web''. In IEEE Pervasive Computing, Oct.–Dec. 2003, pp. 22–33.
[95] R. Kumar, M. Wolenetz, B. Agarwalla, J. K. Shin, ''DFuse: A Framework for Distributed Data Fusion''. In Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys'03), Los Angeles, Nov. 2003, pp. 114–125.
[96] S. Li, S. H. Son, J. Stankovic, ''Event Detection Services Using Data Services Middleware in Distributed Sensor Networks''. In Proceedings of the Workshop on Information Processing in Sensor Networks (IPSN'03), Palo Alto, CA, Apr. 2003.
[97] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, D. Estrin, ''Em*: A Software Environment for Developing and Deploying Wireless Sensor Networks''. In Proceedings of the USENIX 2004 Annual Technical Conference, Boston, MA, June– July 2004, pp. 283–296.
[98] A. Boulis, C. C. Han, M. B. Srivastava, "Design and implementation of a framework for efficient and programmable sensor networks". In Proceedings of ACM International conference on Mobile Systems, Applications and Services (MobySys '03), San Franciso, CA, May 2003, pp. 187-200.
[99] D. Ganesan, A. Cerpa, W. Ye, Y. Yu, J. Zhao, D. Estrin, "Networking Issues in Wireless Sensor Networks". In Journal of Parallel and Distributed Computing, Vol. 64 (2004) , pp. 799-814.
[100] Akyildiz, I.F. , Weilian Su Sankarasubramaniam, Y. Cayirci, E. Georgia, "A survey on sensor networks". In Communications Magazine, IEEE, Aug 2002,Volume: 40, Issue: 8. On page(s): 102- 114. ISSN: 0163-6804.
[101] B. Chen, K. Jamieson, H. Balakrishnan, R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks". In ACM Wireless Networks, Vol. 8, N. 5, September 2002.
[102] Bob Heile, "Wireless Sensors and Control Networks: Enabling New Opportunities". ZigBee Alliance April 2007. http://www.zigbee.org
[103] Bob Heile, " ZigBee Smart Energy: The Evolution of Smart Metering". ZigBee Alliance. September 2008. http://www.zigbee.org
[104] XBee datasheet, 2008, http://www.digi.com/products/wireless/point-multipoint/xbee-series1-modulespecs.jsp
[105] SQLite, 2008, http://www.sqlite.org/

[106] USQLiteServer, 2007, http://users.libero.it/irwin/
[107] TechFell protocol, 2007, http://users.libero.it/irwin/TechFell_protocol.html
[108] PHP, 2008, http://www.php.net
[109] Dojo Toolkit, 2008, http://dojotoolkit.org
[110] XML, 2008, http://www.w3.org/XML/
[111] Prototype JavaScript framework, 2008, http://www.prototypejs.org/
[112] JQuery, 2008, http://jquery.com
[113] Google Web Toolkit, 2008, http://code.google.com/webtoolkit/
[114] Google Maps API, 2008, http://code.google.com/apis/maps/
[115] Lighttpd, 2008, http://www.lighttpd.net/
[116] Nginx, 2008, http://nginx.net/