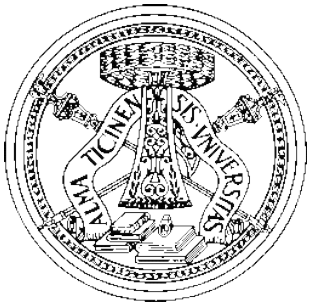


Università degli Studi di Pavia
Facoltà di Ingegneria

Corso di Laurea Specialistica in Ingegneria Informatica

Sviluppo di una rete wireless di sensori per il monitoraggio di strutture in tempo reale

Relatore:
prof. Paolo E. Gamba

Correlatore:
dott. Emanuele Goldoni

Tesi di laurea di:
Marco Beltrame
Matricola: 358880/27

Anno Accademico 2007/08

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Lei mi conosceva e io conoscevo lei, l'avevo sempre conosciuta sin dal primo istante. Tutto il resto non aveva importanza apparteneva al passato: era come se non ci fosse mai stato. Esisteva solo nel mio ricordo.

dal film *Paradiso Perduto (Great Expectations, 1998)*

La massiccia diffusione di computer portatili, cellulari, palmari, navigatori, sistemi radio di identificazione e, più in generale, di tutti i dispositivi elettronici "intelligenti" ha reso la tecnologia ed i computer parte della quotidianità. Questo sviluppo è stato favorito anche dai prezzi sempre più contenuti e dall'inarrestabile processo di miniaturizzazione dei componenti elettronici. Occorre inoltre aggiungere che lo sviluppo di protocolli e sistemi di telecomunicazione wireless hanno consentito di creare architetture di comunicazione a costi contenuti, raggiungere luoghi impervi e slegare il concetto di terminale da quello di posizione fisica, introducendo così nelle reti la mobilità.

In questa prospettiva le reti wireless di sensori, o Wireless Sensor Network (WSN) [31], possono essere considerate come l'ultima fase del processo di miniaturizzazione e di diffusione pervasiva, quasi integrata con l'ambiente circostante, dei dispositivi elettronici. Una WSN [51] può essere brevemente descritta come una rete costituita da un insieme di sensori distribuiti nell'ambiente e che comunicano e tra loro allo scopo di rilevare, condividere ed elaborare i dati acquisiti dall'ambiente fisico. Tipicamente queste reti sono costituite da un certo numero di elementi in grado di comunicare tra loro, detti nodi, aventi dimensioni estremamente ridotte e un costo contenuto. Un nodo di una sensor network funziona solitamente grazie ad una batteria ed è dotato di un numero variabile di sensori e, eventualmente, attuatori. Una rete di questo tipo può contenere dalle decine alle migliaia di nodi, che comunicano tra loro senza fili e che sono in grado di auto-organizzarsi e collaborare per elaborare o trasmettere le informazioni da un capo all'altro del sistema. L'obiettivo primario di ogni nodo è comunque quello di inviare i propri dati verso un centro di raccolta interno alla Wireless Sensor Network detto *gateway*: quest'ultimo ha il compito di fungere da ponte tra la rete di sensori ed il sistema di raccolta ed elaborazione dati, inoltrando tutti i dati pervenuti attraverso un canale di comunicazione differente quali Wi-Fi, GPRS [61] o Ethernet.

Le reti di sensori, a differenza delle più diffuse reti Wi-Fi utilizzate per l'accesso dei terminali ad una rete locale, sono programmate per svolgere un compito preciso e specifico. I fenomeni monitorati da una WSN [64] possono essere di tipo strettamente ambientale, come l'andamento nel tempo di temperature, umidità o quantità di illuminazione, anche se non è da escludere l'impiego di una rete di sensori con funzioni di controllo su apparecchiature industriali o per la sorveglianza remota.

Sebbene i possibili campi di applicazione siano innumerevoli, le Wireless Sensor Network non sono ad oggi ancora molto diffuse. I problemi da affrontare sono infatti molteplici e riguardano la limitata capacità di banda, l'inaffidabilità della comunicazione via radio, la durata limitata delle batterie a bordo dei dispositivi e la necessità di disporre di una rete in grado di tollerare guasti e riorganizzarsi autonomamente. In particolar modo, dal momento che tutti gli aspetti di una rete di sensori sono fortemente dipendenti dalle specifiche esigenze degli utilizzatori e dai parametri ambientali monitorati, non esistono standard e protocolli in grado di funzionare adeguatamente in ogni condizione.

Il protocollo oggi più accreditato per divenire lo standard nelle reti wireless di sensori è IEEE 802.15.4 [19], insieme al suo successore 802.15.4a basato sulla tecnologia Ultra Wide Band (UWB) [63]. Se però il livello fisico proposto è stato adottato da molti produttori, la ricerca nell'ambito dei meccanismi di accesso al canale MAC (Media Access Control) è invece ancora attiva e sono state proposte numerose soluzioni alternative. Esistono inoltre alcuni protocolli "concorrenti", sviluppati in particolar modo da aziende privati o ristretti consorzi, che potrebbero trovare applicazione in particolari nicchie di mercato. Per quanto concerne i livelli superiori, il panorama è decisamente variegato e non ancora consolidato. Da un lato infatti le specifiche dello standard *ZigBee*, pur offrendo garanzie di interoperabilità tra i diversi costruttori e costi contenuti, possono risultare eccessivamente complesse e difficilmente integrabili in una rete IP esistente. D'altra parte lo standard 6LoWPAN [57], così come altri protocolli più semplici sviluppati *ad hoc*, risolve alcuni dei problemi di ZigBee ma ha una diffusione ancora limitata. Lo stesso discorso vale anche dal punto di vista applicativo: *TinyOS* [66] e *Contiki* [25] sono i sistemi oggi più diffusi a livello universitario, ma lo sviluppo di software, middleware e ambienti di simulazione dedicati è ad uno stadio embrionale e non sono ancora disponibili prodotti maturi. Per quanto riguarda infine le piattaforme hardware per le WSN oggi disponibili, il panorama è piuttosto variegato anche se, per lo più, si tratta di prototipi sviluppati da centri di ricerca o *spin-off* universitarie.

La diffusione di numerose soluzioni, seppur non standardizzate, ha consentito ai ricercatori di applicare sul campo le reti di sensori nelle più disparate realtà. Ad esempio, negli ultimi anni, è stata più volte proposto e sperimentato l'impiego delle reti wireless di sensori nel settore dell'Ingegneria Civile e, in particolar modo, per le operazioni di Structural Health Monitoring (SHM) [70], ovvero il monitoraggio dello stato di salute di ponti, palazzi, edifici e strutture in genere. In questo ambito le applicazioni riguardano sia il monitoraggio cosiddetto "reattivo", in risposta a fenomeni catastrofici (esplosioni, terremoti, inondazioni), che il monitoraggio preventivo e continuo di strutture soggette a vibrazioni ambientali (vento, traffico automobilistico o ferroviario etc...). Generalmente le condizioni di una struttura possono essere verificate attraverso due distinti approcci. Una prima soluzione, detta monitoraggio diretto, utilizza fotografie a raggi X della costruzione; l'alternativa è il monitoraggio

indiretto, effettuato catturando le vibrazioni attraverso una rete di sensori. Proprio in questo settore è attivamente impegnata Eucentre, una Fondazione senza scopo di lucro fondata dal Dipartimento della Protezione Civile, dall'Istituto Nazionale di Geofisica e Vulcanologia, dall'Università degli Studi di Pavia e dall'Istituto Universitario di Studi Superiori di Pavia, con il fine di promuovere, sostenere e curare la formazione e la ricerca nel campo della riduzione del rischio sismico. In particolare, grazie ad un laboratorio tra i più avanzati del mondo, dove è possibile svolgere prove quasi statiche, pseudo-dinamiche e dinamiche con tavola vibrante su prototipi di edifici e strutture, Eucentre sviluppa progetti per studiare e migliorare il comportamento sismico delle strutture e per investigare e implementare metodologie e tecniche innovative per il consolidamento antisismico.

Il sistema di acquisizione dati attualmente utilizzato dai ricercatori di Eucentre è realizzato con apparecchiature ottimizzate per questo tipo di applicazioni e prodotto da National Instrument. Il numero di sensori che possono essere collegati e la banda disponibile sono elevati, ma ogni dispositivo di misurazione deve essere collegato al sistema di acquizione dati con un cavo dedicato. Proprio per superare quest'ultimo limite si è deciso di sostituire l'attuale sistema di acquisizione con una WSN [60] appositamente sviluppata.

Tra le varie soluzioni esistenti, si è scelto di adottare la piattaforma *SquidBee*, realizzata da un gruppo di ricercatori dell'Università di Saragozza. Si è deciso di utilizzare questa soluzione, seppur meno consolidata e, talvolta, meno efficiente di altre alternative, per la sua semplicità e facilità di modifica e accesso ai vari componenti. La scheda principale dei nodi SquidBee [2] è basata sul progetto open source Arduino [1] e può essere programmata in linguaggio C utilizzando un ambiente libero e multipiattaforma. Inoltre, poiché lo stesso schema circuitale è liberamente disponibile, è stato possibile realizzare con facilità le interfacce hardware per i diversi sensori utilizzati durante le simulazioni. Poiché la piattaforma SquidBee non è stata pensata appositamente per questo tipo di applicazioni, si è resa necessaria l'implementazione di alcuni soluzioni innovative. In particolar modo, per poter ottenere misurazioni sincrone da sensori differenti, sono sono stati valutati alcuni algoritmi esistenti ed implementato, a parità di efficienza, quello meno complesso. A questo riguardo, un attento studio del comportamento dei *clock* fisici ha consentito di stimare con precisione i *drift* [42] e, di conseguenza, i possibili errori di temporizzazione delle misurazioni. Inoltre, al fine di ridurre le collisioni radio tra i vari nodi durante la trasmissione e limitare il consumo di banda, è stato effettuato uno studio teorico delle prestazioni del sistema. Questo ha consentito di ricavare una funzione in grado di calcolare la frequenza massima di campionamento dei dati partendo da variabili quali la risoluzione delle misurazioni, il numero di nodi presenti nella rete e la velocità di campionamento di un sensore. La soluzione trovata è stata anche implementata nel sistema, facendo sì che l'infrastruttura sia in grado di ottimizzare l'occupazione di banda al variare dei parametri di funzionamento. Il sistema sviluppato è stato infine provato nel laboratorio di Eucentre ed è stato possibile così valutare l'affidibilità, l'accuratezza e la scalabilità del nuovo sistema rispetto a quello esistente.

Per poter meglio approfondire quanto brevemente indicato in questa introduzione, la presente tesi è stata strutturata come segue. Dopo una breve introduzione alle Wireless Sensor Network, trattate nel Capitolo 2, verranno analizzati in dettaglio nel Capitolo 3 gli standard a livello hardware o software e i protocolli più diffusi

per la realizzazione di una rete di sensori. Nel Capitolo 4 sarà quindi introdotto il problema del monitoraggio di strutture ed edifici, presentando le soluzioni cablate tradizionalmente impiegate e lo stato dell'arte in materia di monitoraggio attraverso sensori radio. Il successivo Capitolo 5 conterrà poi una descrizione dettaglia delle soluzioni adottate nella rete di misurazione al fine di garantire il sincronismo delle acquisizioni effettuate. Il Capitolo 6 ospiterà poi la descrizione matematica dell'algoritmo di accesso al canale proposto, seguita infine da un confronto tra il modello teorico ed i risultati ottenuti in laboratorio nel Capitolo 7. Alcune riflessioni conclusive sul lavoro svolto e possibili sviluppi futuri verranno presentati nell'ultimo Capitolo.

Nella Figura 1.1 e Figura 1.2 vengono mostrate visivamente i due sistemi per il monitoraggio di strutture con il sistema cablato e con il sistema wireless.

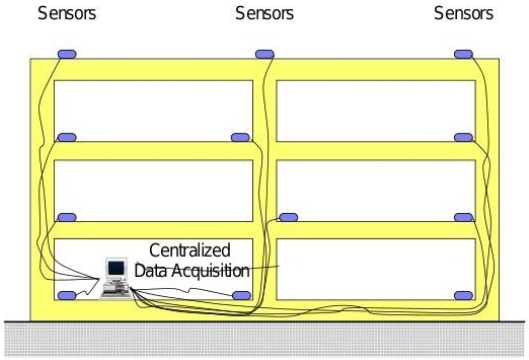


Figura 1.1: Monitoraggio attraverso un sistema wired.

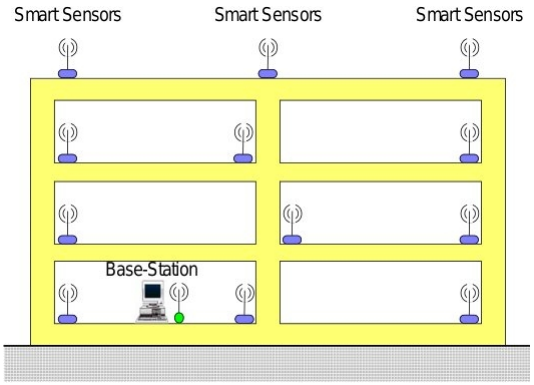


Figura 1.2: Monitoraggio attraverso un sistema wireless.

Capitolo 2

Wireless Sensor Network

Ciò che e vero nella nostra mente è vero, che certe persone lo sappiano o no. Mi sono reso conto di essere parte del problema, non per quello che io ti ricordo, ma perché non sono riuscito a starti vicino e... ti ho lasciata sola.

dal film *Al di là dei Sogni (What Dreams May Come, 1998)*

2.1 Introduzione alle Reti Wireless

Fino a pochi anni fa, qualsiasi tipo di connessione di rete era costruita con cavi. I vantaggi offerti dalle reti cablate sono ben noti: nessuna limitazione enegetica, poichè laddove è possibile portare una connessione è possibile prevedere anche una o più linee di alimentazione, buoni livelli di sicurezza, dato che per prelevare informazioni dalla rete si necessita di un accesso fisico al canale, e prestazioni elevate. Nel contempo però le reti *wired* soffrono di gravi limitazioni, tra cui l'evidente difficoltà di realizzazione in ambienti inospitali e gli elevati costi che un cablaggio strutturato comporta. Inoltre una infrastruttura di collegamento cablata viene detta "rigida", in quanto risulta difficile aggiungere nuovi nodi alla rete o modificare a piacimento la posizione di sensori preesistenti.

Le reti wireless, invece, permettono ai terminali di usufruire dei servizi senza che richiedere una connessione diretta via cavo (infatti il termine wireless significa "senza fili"). In questi ultimi anni le reti wireless hanno aperto scenari decisamenti innovativi, nei quali gli utenti possono, integrandosi ai classici servizi di telefonia, sfruttare reti wireless dispiegate nel territorio per essere connessi ad Internet. Le reti di questo tipo si sono rapidamente diffuse in quanto offrono una serie di innegabili vantaggi: la mobilità, che permette ai terminali di potersi muovere, la flessibilità ed i bassi costi di realizzazione. Tuttavia anche le reti wireless devono affrontare alcune problematiche. Una di queste è indubbiamente la capacità del mezzo trasmissivo, l'etere, che è unico e condiviso da tutti i nodi connessi. L'esistenza di un unico canale limita necessariamente il numero massimo di utenti che possono usufruire del servizio contemporaneamente; allo stesso modo, la presenza di più utenti determina una riduzione della velocità di trasmissione, in quanto la capacità del canale trasmissivo

deve essere suddivisa tra tutti coloro che ne stanno facendo uso. Non da meno è il problema della sicurezza: in assenza di specifici controlli, risulta facile per un attaccante intercettare le informazioni che viaggiano nell'etere o riuscire ad accedere a servizi anche senza autorizzazione. Occorre inoltre considerare che la qualità della comunicazione può venir influenzata anche da fattori esterni, come interferenze elettromagnetiche e ostacoli in movimento. Infine il consumo energetico degli apparati di trasmissione radio è tipicamente più elevato di quelli per la comunicazione via cavo.

In Tabella 2.1 vengono messe a confronto le caratteristiche principali del sistema wired e wireless. Per molti di questi problemi esistono soluzioni efficaci, ma per ogni soluzione adottata si deve sempre prendere in considerazione l'aumento della complessità progettuale e il relativo aumento dei costi di realizzazione.

Wired	Caratteristica	*Wireless*
⇓	Costo	⇑
⇑	Consumi	⇓
⇑	Affidabilità	⇓
⇓	Mobilità	⇑
⇓	Flessibilità	⇑
⇑	Sicurezza	⇓

Tabella 2.1: Confronto tra le caratteristiche dei sistemi wired e wireless.

2.2 Tipologie di Reti Wireless

Come mostrato in Figura 2.1, una classificazione delle reti wireless può essere fatta in base all'area coperta dal segnale trasmesso dai dispositivi [47; 64].

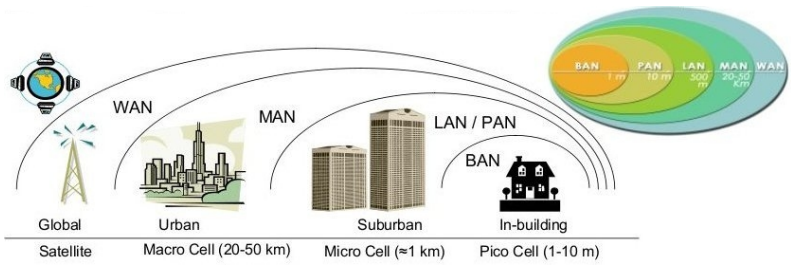


Figura 2.1: Classificazione delle reti wireless in base all'area di copertura.

Le reti wireless più piccole vengono chiamate Wireless Body Area Network (WBAN). Una Body Area Network serve per connettere dispositivi "indossabili" sul corpo umano come ad esempio possono essere i componenti di un computer (auricolari); idealmente una rete di questo tipo supporta l'auto-configurazione, facendo così in modo che l'inserimento e la rimozione di un dispositivo dalla BAN risultino essere trasparenti all'utente.

Aumentando il raggio d'azione di una rete si passa alle le Wireless Personal Area Network (WPAN). Il raggio di comunicazione delle PAN si aggira ai 10 metri

e l'obiettivo di queste reti è quello di consentire a dispositivi vicini di condividere dinamicamente informazioni. Pertanto, mentre una BAN si dedica all'interconnessione dei dispositivi indossabili da una persona, una PAN è operativa nella immediate vicinanze degli utenti. In una Personal Area Networl e possibile perciò connettere dispositivi portatili tra loro oppure con stazioni fisse, ad esempio per accedere ad Internet, sincronizzare o scambiari dati e contenuti multimediali su computer portatili, desktop e palmari, etc. Lo standard più utilizzato per la realizzazione delle reti Body/Personal Area Network è Bluetooth.

Un raggio d'azione maggiore hanno invece le Wireless Local Area Network (WLAN), che consentono la comunicazione tra dispositivi distanti tra loro anche alcune centinaia di metri e risultando così adeguate per la realizzazione soluzioni di interconnessione in ambienti chiusi o tra edifici adiacenti. La famiglia degli standard più diffusi per la realizzazione di questo genere di interconnessioni è IEEE 802.11x [17; 33; 6] e può essere considerata l'alternativa wireless alle tradizionali LAN basate su IEEE 802.3/Ethernet.

Le Wireless Metropolitan Area Network (WMAN) hanno invece un raggio d'azione di circa 10 Km, fornendo così un accesso a banda larga ad aree residenziali anche piuttosto estese. La tecnologia più accreditata per la realizzazione di reti Wireless MAN è WiMAX, basata sullo standard IEEE802.16.

Quello sopra proposto non è però l'unico modo possibile per classificare le reti wireless. Ad esempio è possibile confrontare le tecnologie in base al consumo e alla velociatà di trasmissione (*data-rate*). Come è possibile vedere in fig:protocolli il rapporto tra consumo e velocità di trasmissione risulta essere proporzionale: in generale, alti data-rate implicano consumi elevati e viceversa.

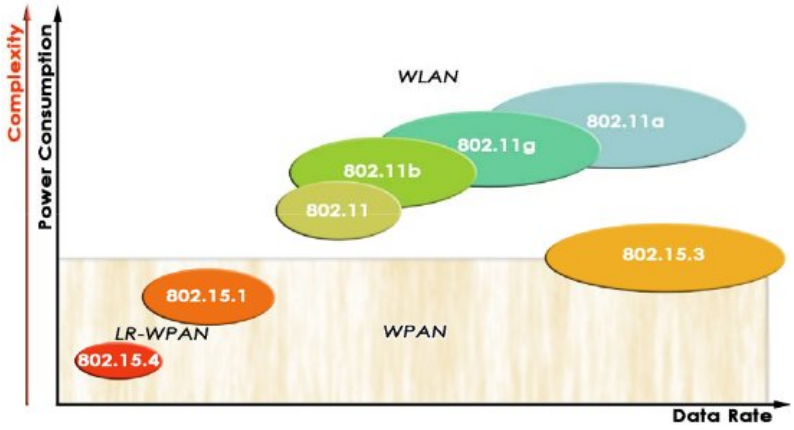


Figura 2.2: Rapporto tra velocità e consumi dei protocolli wireless.

2.3 Personal Area Network e Sensor Network

All'interno della categoria delle Wireless PAN, è possibile considerare anche la categoria delle Wireless Sensor Network (WSN). Questo tipo di reti hanno il compito di interconnettere apparati di misura o attuatori al fine di svolgere compiti specifici, come rilevare temperature, umidità, luminosità, oscillazioni, etc. Ogni nodo apparte-

nente alla rete ha la capacità di eseguire calcoli, di catturare dati e di comunicare e cooperare con gli altri nodi della rete.

Nel panorama delle Personal Arean Network sono diffuse numerose tecnologie wireless, prima tra tutti lo standard Bluetooth, anche noto come IEEE 802.15.1. Bluetooth permette l'interconnessione di piccoli dispositivi e per trasmissioni di breve durata e prevede due diversi tipi di dispositivi: i nodi *master*, aventi funzioni di coordinamento, e i nodi finali, detti *slave*. In una rete Bluetooth, detta *piconet*, insieme al master possono essere contemporaneamente attivi non più di 7 nodi slave.

Lo standard IEEE 802.15.4 è invece specifico per le WSN. Questo tipo particolare di protocollo è stato progettato per offrire, rispetto agli altri della stessa famiglia IEEE 802.15, la possibilità di poter realizzare reti costituite da un gran numero di nodi e, quindi, con una densità spaziale anche molto elevata. Inoltre nelle reti si sensori i limiti in termini di consumo energetico sono fondamentali e la quantità di banda richiesta è piuttosto limitata; non a caso il protocollo 802.15.4, specifico per le Wireless Sensor Network, è un protocollo *low-power* e *low-data-rate*.

I dettagli dello standard IEEE 802.15.4 verranno approfonditi successivamente nel Capitolo 3.

2.4 Strutture delle Wireless Sensor Network

Una rete di sensori può essere realizzata in tre differenti topologie [48; 51], dove con il termine topologia si indica la modalità in cui i diversi dispositivi della rete vengono disposti e, in particolare, i collegamenti fisici e logici che li intereconnettono. Nelle reti wireless esistono tre principali strutture, riassunte anche in Figura 2.3 : stella (*star*), *mesh* e albero (*tree*).

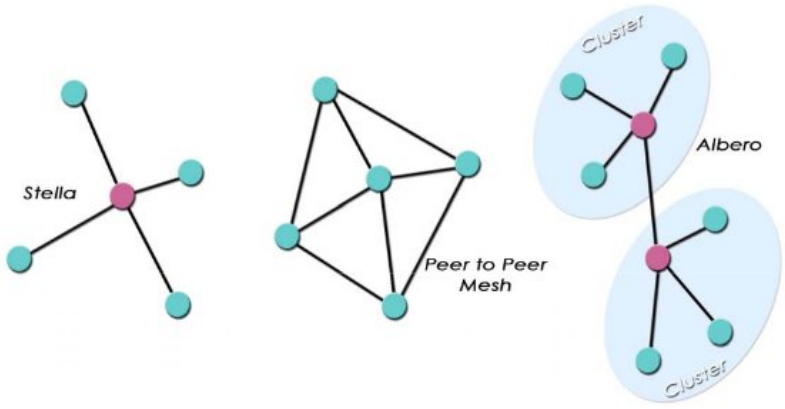


Figura 2.3: Possibili topologie di rete nelle Wireless Sensor Network.

La topologia di rete a stella prevede che un unico nodo si comporti da coordinatore e gli altri nodi sono connessi direttamente al coordinatore tramite sono link di comunicazione uno-ad-uno.

La struttura a mesh è formata da nodi sensori interconnessi tra loro attraverso uno o più link. Questa rete può essere chiamata anche a maglia. In questa struttura è necessario definire chiare regole di routing per lo scambio di messaggi tra i vari nodi.

Può esistere anche un coordinatore e trovarsi al di fuori della struttura mesh. In questo caso sarà collegato ad uno solo dei nodi ed avrà funzioni di controllo limitate. Se non esiste un coordinatore tutti i nodi avranno funzioni di coordinamento e raccolta dati in modo paritetico.

Infine, la struttura ad albero è una topologia di rete che combina la struttura a stella e quella a mesh [9]. Più nodi, che fanno a capo ad un unico nodo, formano un gruppo chiamato *cluster*. I nodi-coordinatori possono essere a loro volta far riferimento ad un nodo coordinatore di livello superiore e costruendo, così, vari livelli di gerarchia.

2.5 Impiego delle Wireless Sensor Network

Le reti di sensori possono essere utilizzate in svariati campi [21]. Viene di seguito presentato un elenco dei settori dove vengono tipicamente impiegate le Wireless Sensor Network.

- Monitoraggio ambientale (*Enviromental and habitat monitoring*) In campo ambientale le applicazioni possono essere molteplici: è possibile monitorare movimenti di animali oppure studiare particolari habitat, come il mare, il terreno o l'aria impiegando, ad esempio, sistemi di monitoraggio di agenti inquinanti. Appartengono a questo settore anche lo studio di eventi naturali catastrofici quali incendi, tornado, terremoti ed eruzioni vulcaniche.

- Monitoraggio di strutture (*Structural Health Monitoring*) Le reti di sensori posizionate sulle strutture rilevano lo stato di salute di edifici, ponti, case sottoposte a sollecitazioni esterne; in alternativa, potrebbero essere utilizzate anche per misurare difetti strutturali di componenti.

- Controllo del traffico veicolare (*Traffic control*) Un sistema di sensori finalizzato al monitoraggio del traffico controlla il passaggio di automobili, analizzare la velocità ed l'intensità del traffico ed individuando eventuali blocchi o situazioni anomale.

- Sorveglianza di edifici (*Infrastructure control*) Questo tipo di reti può essere utilizzato come ausilio per la sorveglianza di centri commerciali o di luoghi a rischio, come stazioni ferroviarie o aereoporti.

- Sorveglianza militare (*Militar control*) Le reti di sensori sono state utilizzate in principio per questo scopo. Le loro applicazioni spaziando dal monitoraggio sullo stato o la posizione delle forze sul campo di battaglia alla sorveglianza di luoghi strategici. Possono inoltre essere dispiegate in luoghi ostili al fine di raccogliere informazioni sugli spostamenti del nemico.

- Monitoraggio di apparecchiature industriali (*Industrial sensing*) I sensori wireless possono essere applicati a macchinari industriali per analizzarne il comportamento dei componenti sottoposti a stress meccanico, migliorarne le performance o prevenire rotture e guasti.

- Monitoraggio Agricolo (*Agriculture sensing*) Nel settore agricolo le reti WSN vengono utilizzate per il monitoraggio di particolari situazioni ambientali; in particolar modo la cosiddetta "agricoltura di precisione" utilizza le reti di sensori per rilevare in tempo reale il livello di pesticidi nell'acqua o nei terreni agricoli.

- Applicazioni Personali (*Personal sensoring*) Nel settore della domotica le reti di sensori riescono a fornire servizi all'utente all'interno della propria abitazione, ad esempio informandolo tempestivamente di eventuali guasti. I sensori possono essere introdotti negli apparecchi elettrici e questi, interagendo tra loro e con una rete esterna o Internet stesso, permettono all'utente di comandare facilmente gli elettrodomestici a distanza.

- Applicazioni Mediche (*Personal Health Care*) Le reti wireless sono oggi impiegate anche per monitorare pazienti, eseguire valutazioni diagnostiche, amministrare farmaci in ospedale e monitorare a distanza dati fisiologici quali la frequenza cardiaca o la pressione sanguigna.

Capitolo 3

Standard, Hardware e Software di trasmissione per le WSN

...ogni anno durante la luna delle foglie cadenti
sognavo che la voce dell'orso dentro di lui si fosse
fatta silenziosa e che Tristan sarebbe potuto tornare a
vivere in questo mondo, ma poi giungeva l'inverno
infine un'altra primavera ed era ancora lontano...

dal film *Vento di passioni (Legends of the Fall, 1994)*

Nel capitolo verranno illustrati inizialmente i tipi di hardware e i sistemi operativi utilizzati per la realizzazione di Wireless Sensor Network. Successivamente verrà presentato lo standard di comunicazione IEEE 802.15.4 e le sue evoluzioni; infine verrà fornita una panoramica sui protocolli di rete e applicativi più diffusi in questo ambito.

3.1 Hardware

Esistono in commercio molte soluzioni per la costruzione di WSN, ognuna delle quali presenta delle proprie caratteristiche e funzionalità. Vengono di seguito descritte le principali piattaforme hardware oggi disponibili in commercio.

3.1.1 UC Berkeley

L'università di Berkeley è stata una delle prime a costruire dei nodi sensori, proponendo negli anni una serie di modelli. Una famiglia di piattaforme molto conosciuta è quella dei *motes* MICA, evolutosi dalla versione MICA (Figura 3.1) a MICA2 (Figura 3.2) e MICA-Z [56]. Tutte e tre le versioni montano un processore ATMega128L, non hanno sensori integrati nella piattaforma hardware di base e sono impiegabili per il rilevamento di temperatura, umidità, accelerazioni, pressione o campi magnetici. Ciò che differenzia le varie versione dei motes è il componente radio: nei MICA è utilizzato il chip TR1000, che offre un data-rate massimo di 40 Kbits/s, nei MICA2 è installata la radio CC1000, con un data rate massimo di 38.4 Kbits/s, mentre nei MICA-Z è presente il più recente integrato CC2420, che può raggiungere velocità

trasmissive di 250 Kbits/s. Evoluzione della famiglia MICA è il nodo TELOS, di cui sono disponibili le versioni Telos, TelosB e TmoteSKY. A differenza dei MICA, i TELOS montano un processore TI MSP430 e per le schede radio adottano il modulo CC2420. La particolarità di questi nodi sta nel fatto che dispongono di sensori di temperatura, umidità e luminosità già integrati sulla scheda.

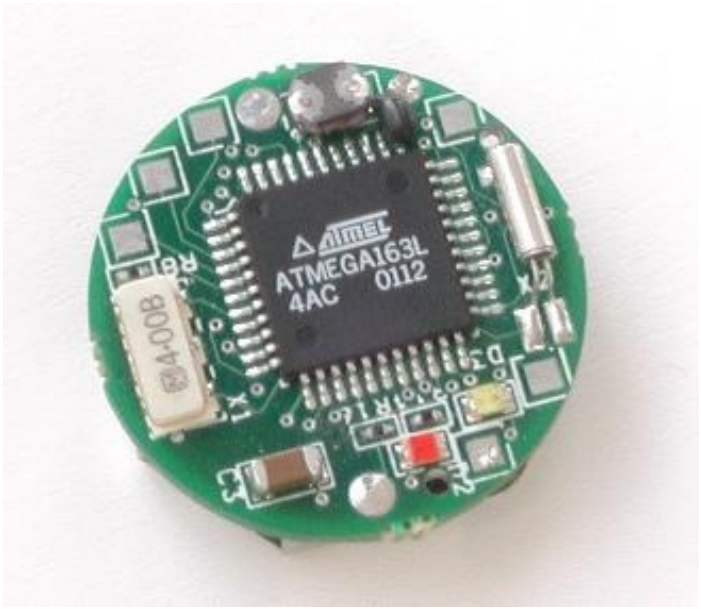


Figura 3.1: Il sensore MICA.

3.1.2 BTnode

I BTnode (Figura 3.3) [16] sono sensori che utilizzano un processore ATMega128L e una scheda radio CC1000, che opera nella banda ISM 433-915 MHz. Progettato principalmentee dall'ETH di Zurigo, è stato realizzato in tre versioni: la più recente, chiamata semplicemente BTnode rev3, è la piattaforma che ha avuto più successo. La particolarità del di questa piattaforma consiste nel possibilità di usufruire di un doppio sistema di trasmissione radio, disponendo anche di una radio Bluetooth. Il sistema Bluetooth integrato può supportare 4 piconet, ognuna delle quali può comandare al massimo 7 slave. Il BTnode non presenta alcun sensore integrate ma la scheda supporta i sensori di luminosità, temperatura, accelerazione e suono costruiti dalla TAOS (Texas Advanced Optoelectronic Solutions).

3.1.3 Squidbee

I nodi Squidbee si basano su un processore Atmega168 e il modulo radio è l'XBee prodotto dalla DiGi. La piattaforma Squidbee è progettata per la costruzione di WSN che rilevano temperature, luminosità e umidità grazie a sensori esterni, non integrati nella scheda principale (Figura 3.4). Il nodo sensore è formato da una board, costruita dal progetto Arduino, e un circuito di interfacciamento tra il modulo radio XBee e la board stessa, progettato dalla Libelium. Il modulo radio XBee può operare

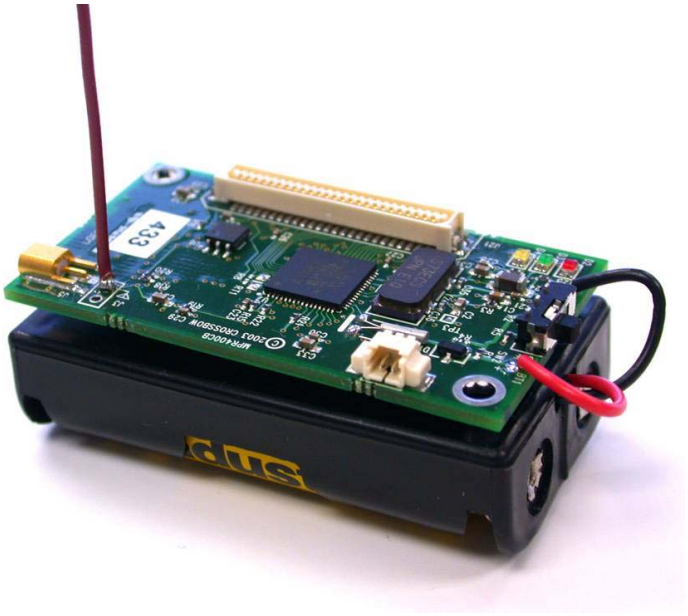


Figura 3.2: Il sensore MICA 2.

Figura 3.3: Il sensore BTnode.

nelle bande di frequenza 2.4 GHz, 915 MHz e 868 MHz e ha un data-rate variabile da 20 Kbits/s ad un massimo di 250 Kbits/s. Esiste anche la versione del modulo radio XBee-Pro, che estende il raggio d'azione della radio ad 1 Km. I progetto dei nodi è *open*, ovvero ogni parte del nodo è accessibile e può essere studiata, cambiata e personalizzata utilizzando gli schemi circuitali e il codice sorgente. Questo tipo di piattaforma non usa alcun tipo di sistema operativo: il codice viene compilato e caricato direttamente all'interno della EEPROM per essere caricato all'avvio direttamente dal *bootloader*.

Figura 3.4: Il sensore SquidBee.

3.1.4 Eyes

Il modello Eyes [21], come si può vedere in Figura 3.5, ha dimensioni ridottissime. Il nodo monta un processore MSP430 e un modulo radio TDA 5250, che opera nella banda degli 868 MHz e offre una velocità massima di trasmissione di 64 Kbit/s. La piattaforma ha integrato i sensori di luminosità e di temperatura. Il sensore di temperatura lavora in un range molto ampio, che va da -30°C a $+100$°C con un errore di circa ± 2°C. L'hardware è stato progettato e costruito per offrire una alta efficienza energetica; alcuni esperimenti effettuati hanno dato ottimi risultati, riuscendo a preservare il consumo di energia ma raggiungendo però un data-rate massimo di circa 20 Kbits/s, decisamente al di sotto del valore teorico atteso.

Figura 3.5: Il sensore Eyes.

3.1.5 Firefly

Il nodo sensore Firefly, visibile in Figura 3.6, è dotato di un processore Atmega128L e un modulo radio CC2420. Al suo interno sono integrati in una scheda i sensori

di luminosità, temperatura, audio e accelerazione. I consumi del nodo sono estremamente ridotti: la durata prevista delle batterie è di quasi due anni. Questo è reso possibile da complessi meccanismi di sincronizzazione tra i vari nodi dell'intera rete, che permettono di poter mandare i sensori in fase di sleep molto rapidamente e minimizzando i periodi di *idle*. Il raggio di azione di questi dispositivi è di circa 50-100 metri e il data-rate massimo è pari a 250 Kbps.

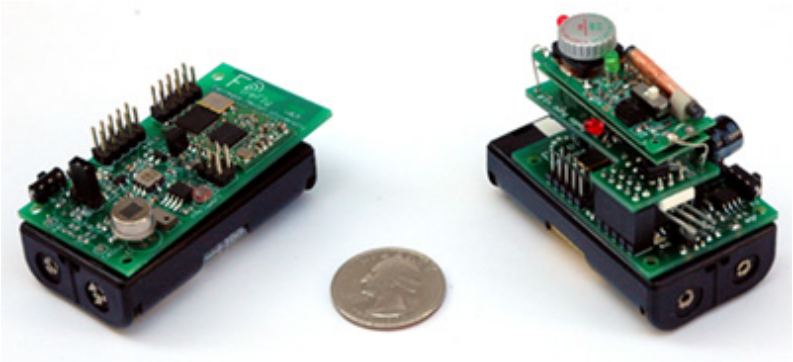


Figura 3.6: Il sensore FireFly.

3.2 Sistemi Operativi

Il sistema operativo per i sensori deve avere bassa occupazione in memoria, supportare un modello operativo di tipo essenzialmente reattivo ed infine adattarsi a diversi scenari con la minima occupazione di memoria. La tradizionale distinzione tra i diversi livelli è spesso sfumata per ottimizzare l'occupazione di memoria. Lo strato di rete è di norma fortemente integrato al resto e il SO viene spesso fornito come libreria da collegare all'applicazione. I più utilizzati nell'ambito della ricerca sono: TinyOS, Contiki e NANORk.

3.2.1 TinyOS

TinyOS [38] è un sistema operativo totalmente non bloccante, è basato su un kernel minimale e gestisce le operazioni di I/O. L'occupazione in memoria di TinyOS è di circa una decina di Kbyte e non è previsto il supporto per i thread. Il sistema è scritto in NesC, un linguaggio ad eventi progettato per programmare applicazioni component-based principalmente per sistemi embedded, e anche le varie applicazioni che girano all'interno di TinyOS devono essere scritte in NesC. E' il più diffuso e il più utilizzato in quasi tutti i tipi di sensori.

3.2.2 Contiki

Contiki [25] è un sistema operativo multitasking progettato per i dispositivi embedded e le vecchie architetture a 8 e 16 bit incluso il processore 6510 della Commodore 64. E' sviluppato per sistemi di rete con limitata capacità di memoria. La sua occupazione è di 2 KByte di memoria RAM e 40 di ROM. Contiki offre un kernel che permette di poter applodare dinamicamente i programmi. Per eseguire dei processi utilizza thread

molto leggeri e usa messaggi per la gestione degli eventi. Questo sistema operativo può funzionare microprocessori TI MSP430 e i primi Atmel AVR. Opzionalmente Contiki offre una GUI grafica per la connessione con i terminali della rete. Il sensore che utilizza questo SO è il TmoteSKY.

3.2.3 Nano-RK

Nano-RK, come TinyOS, è un sistema operativo della Carnegie Mellon University specifico per le Wireless Sensor Networks molto compatto e ricco di funzionalità. I tasks, come nei tradizionali sistemi *real-time*, fanno richiesta di risorse e Nano-RK garantisce e controlla gli accessi alla CPU, controllando anche i consumi di ogni operazione. Questo sistema operativo supporta nativamente il meccanismo di accesso al canale B-MAC, insieme al classico CSMA/CA. La richiesta di memoria è più grande rispetto a quella fatta per il sistema operativo TinyOS. Nano-RK ottimizza prima la memoria RAM e successivamente la ROM. Elaborato esclusivamente per le Sensor Network che hanno caratteristiche di memoria di 32-64 KB di ROM e 4-8 KB di RAM. Supporta tecniche di power-management e provvede automaticamente al risparmio di energia. Agevola le funzioni di routing occupandosi dell'incapsulatura dei pacchetti e della trasmissione [32]. FireFly è il sensore che supporta il SO Nano-RK.

3.3 Protocolli di comunicazione

Nell'ambito delle comunicazioni wireless e più precisamente nelle wireless a corto raggio è stato definito uno standard definito dall'IEEE chiamato 802.15.4 Questo standard è pensato per comunicazioni wireless di basso costo, bassa velocità e basso consumo [69] energetico. E adatto a reti W-PAN a basso bit rate costituite da dispositivi alimentati tramite batterie che non possono essere sostituite frequentemente, come, ad esempio, i sensori. Le sue principali caratteristiche sono la capacità di coprire un raggio di azione di poche decine di metri offrendo fino ad un data-rate massimo di 250 Kbps. Offre una bassa potenza in trasmissione implicando un basso consumo energetico [68]. Permette ai livelli superiori la presenza di livelli di rete che possono effettuare routing. Offre altresí la possibilità di poter utilizzare ACK e quindi di poter effettuare ritrasmissioni dei dati in caso di mancato ricevimento o di errore di trasmissione.

3.3.1 Standard IEEE 802.15.4

Lo standard 802.15.4 definisce il livello 1 (livello fisico) e il livello 2 (livello MAC). I dispositivi, dettate dalle specifiche IEEE 802.15.4, possono avere due diverse modalità: Reduced Function Device (RFD) e Full Function Device (FFD). FFD ha funzioni da coordinatore e può funzionare con qualsiasi tipologia di rete si fa carico delle connessioni con gli altri dispositivi. L'RFD è utilizzato unicamente nelle tipologie a stella, non può diventare coordinatore della rete e può colloquiare solamente con esso. La possibilità offerta dal protocollo ai dispositivi di poter creare un collegamento con altri dispositivi adiacenti dà la possibilità di poter costruire reti di grandi dimensione e molto complesse passando da strutture a stella a strutture formate da cluster di dispositivi.

Livello fisico

I servizi, che il livello fisico offre, intervengono sull'accensione e sullo spegnimento del modulo radio, questa opzione permette al sistema un il risparmio d'energia, effettua, inoltre, la scelta del canale di comunicazione calcolando una stima sull'occupazione del canale ed analizza e calcola il rapporto segnale/rumore di un canale per scegliere il migliore diminuendo la probabilità di errori in trasmissione. Ovviamente modula e demodula i segnali in ricezione e in trasmissione. Il livello fisico opera nella banda ISM utilizzando tre possibili bande libere (Figura 3.7):

- 868-868.6 Mhz: banda utilizzata nella maggior parte dei paesi europei con un data-rate di 20kbps disponendo di un solo canale a disposizione;
- 902-928 Mhz : banda utilizzata nel continente oceanico e nel continente americano, offrendo un data-rate di 40Kbps e 10 canali disponibili;
- 2400-2483.5 : banda utilizzabile in quasi tutto il globo con un data-rate massimo di 250 Kbps e 16 canali a disposizione.

PHY (MHz)	Frequency band (MHz)	Spreading parameters		Data parameters		
		Chip rate (kchip/s)	Modulation	Bit rate (kb/s)	Symbol rate (ksymbol/s)	Symbols
868/915	868–868.6	300	BPSK	20	20	Binary
	902–928	600	BPSK	40	40	Binary
868/915 (optional)	868–868.6	400	ASK	250	12.5	20-bit PSSS
	902–928	1600	ASK	250	50	5-bit PSSS
868/915 (optional)	868–868.6	400	O-QPSK	100	25	16-ary Orthogonal
	902–928	1000	O-QPSK	250	62.5	16-ary Orthogonal
2450	2400–2483.5	2000	O-QPSK	250	62.5	16-ary Orthogonal

Figura 3.7: Bande di Frequenza per le WSN.

La modulazione del segnale più diffusa è il DSSS utilizzando tecniche BPSK, O-QPSK, anche se recentemente sono state introdotte nuove modulazioni in aggiunta al DSSS come PSSS.

Il datagram a livello fisico, come mostrato Figura 3.8, è formato da una serie di campi: all'inizio si trova il preambolo, formato da 32 bit, con finalità di sincronizzazione tra i nodi. Successivamente viene utilizzato un otteto (11100101) prefissato che funge da indicatore di inizio pacchetto. Segue un campo physical header che indica la lunghezza del payload. Il payload è chiamato Phisical Service Data Units (PSDU) che può avere una lunghezza massima di 127 bytes.

Livello MAC

Il secondo livello (MAC) offre servizi quali la possibilità di creare una rete PAN, la trasmissione dei beacon e l'accesso al canale tramite il protocollo CSMA/CA. Questo livello supporta algoritmi di cifratura basata su AES-128 per la sicurezza dei dati, gestisce, inoltre, l'handshake, cioè gli Acknowledge per la ritrasmissione dei dati in caso di mancata o erronea ricezione. Calcola e verifica l'integrità della PDU. Può

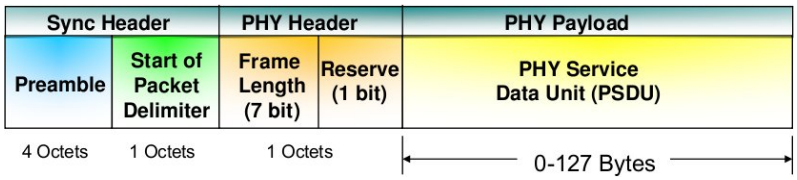


Figura 3.8: Datagram a livello Fisico.

supportare reti fino ad un massimo di 65536 nodi poiché utilizza un indirizzamento fino a 16 bit.

Il livello MAC prevede una struttura chiamata superframe. Questa frame è costruita dal coordinatore della rete ed è contenuta tra due messaggi chiamati beacon. I beacon contengono informazioni che possono essere utilizzate per la sincronizzazione dei dispositivi, per l'identificazione della rete, per descrivere la struttura della superframe stessa e la periodicità di spedizione dei beacon. La superframe è divisa in 16 slot temporali di uguale grandezza, dove il beacon frame è trasmesso nel primo e nell'ultimo slot di ogni superframe. Si possono avere due tipi di superframe senza GTS (Guaranteed Time Slot) e con GTS. Quando viene inviata una superframe senza GTS, l'accesso al canale è regolarizzato dal protocollo CSMA/CA dove ogni dispositivo deve competere con gli altri per assicurarsi l'accesso ad uno slot. Questo periodo è chiamato CAP (Contention Access Period), visibile nella Figura 3.9.

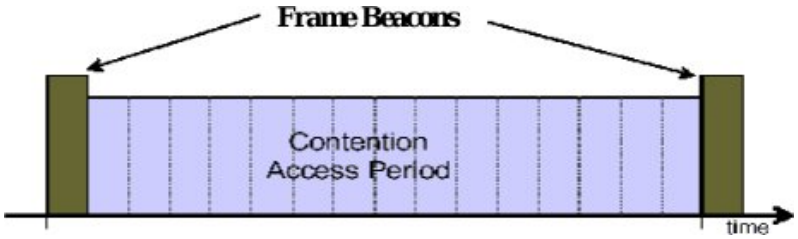


Figura 3.9: Superframe senza GTS.

In altri tipi di applicazioni, invece, si necessita di costruire reti tali per cui garantire a tutti i nodi di poter trasmettere. E' il caso di superframe con GTS, la quale dedica fino ad un massimo di sette slot temporali, detti CFP (Contention-Free Period), a determinati nodi. In questo periodo possono comunicare solamente i nodi prestabiliti e l'accesso al canale non è più CSMA/CA (Figura 3.10).

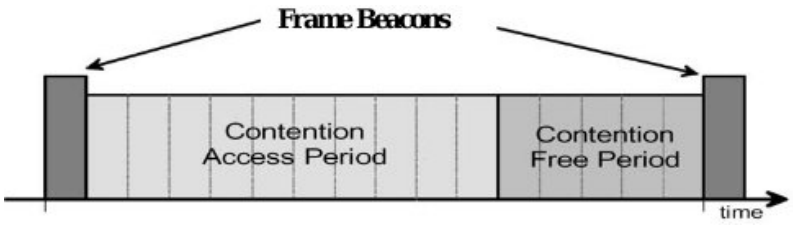


Figura 3.10: Superframe con GTS.

Modalità di trasferimento

La trasmissione dei dati può avvenire in tre modi differenti a seconda di chi spedisce i dati. La prima modalità si riferisce al trasferimento dati dal nodo al coordinatore, la seconda da coordinatore a nodo, la terza tra due nodi. Le prime due modalità possono essere utilizzate in tipologia di reti a stella mentre per le reti mesh possono essere utulizzate tutte e tre le modalità.
E' possibile utilizzare due meccanismi differenti per la trsmissione dei dati: trasmissione senza l'utilizzo di beacon oppure trasmissioni basate sui beacon.
Con una rete beacon-enabled, quando un nodo deve trasferire i dati ad un coordinatore, ascolta il beacon e si sincronizza alla superframe. Il nodo trasmette il relativo pacchetto al coordinatore il quale, opzionalmente, dopo aver ricevuto i dati, trasmette un pacchetto di conferma dell'avvenuta ricezione al dispositivo (ACK). Il colloquio tra i due terminali è visibile nella Figura 3.11

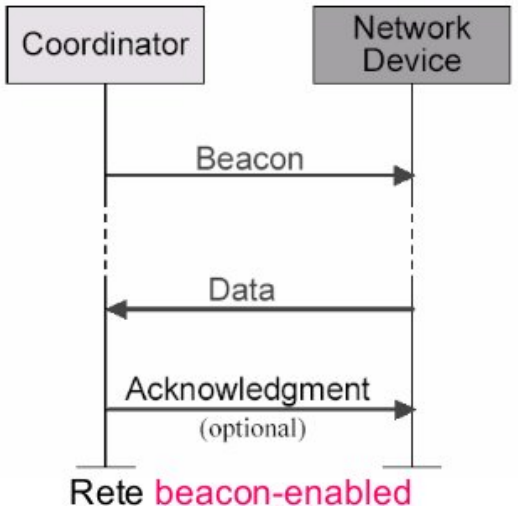


Figura 3.11: Trasmissione nodo-coordinatore Beacon-Enabled.

In una rete beacon-disable, quando un dispositivo desidera trasferire dei dati al coordinatore, trasmette semplicemente utilizzando il protocollo di accesso al canale CSMA/CA. Anche in questo caso il coordinatore dopo la ricezione dei dati trasmette un ACK opzionale (Figura 3.12).

La comunicazione diventa più articolata quando è il coordinatore che necessita di comunicare con un nodo. Se la rete è beacon-enabled, il coordinatore indica nel beacon la necessità di trasferire dati ad un dispositivo, il quale risponde attraverso una PDU chiamata MAC command. La MAC command ha il significato di conferma all'invio dei dati. Il coordinatore, dopo aver ricevuto la MAC command, spedisce in successione una PDU ACK di conferma e successivamente i dati. Il nodo a trasferimento completato invia un ACK (Figura 3.13). Nella rete beacon-disabled, nel trasferimento dati da coordinatore a nodo avviene allo stesso modo con la sola differenza che il coordinatore memorizza i dati finchè non è il nodo stesso a stabilire la connessione. In questa modalità si ricorda che ogni accesso è fatto attrverso il CSMA/CA.(Figura 3.14)

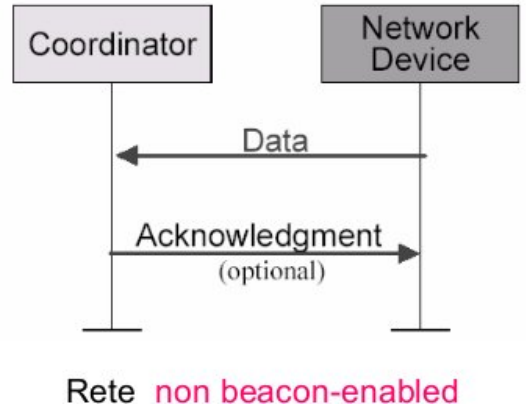


Figura 3.12: Trasmissione nodo-coordinatore Beacon-Disabled.

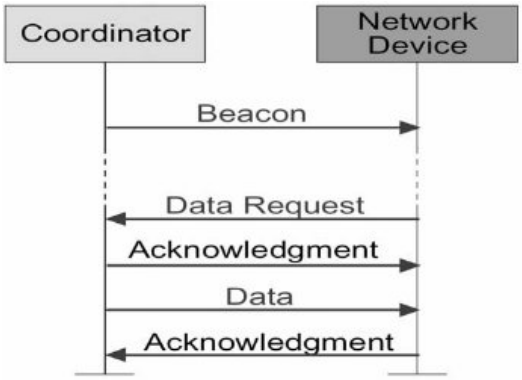


Figura 3.13: Trasmissione coordinatore-nodo Beacon-Enabled.

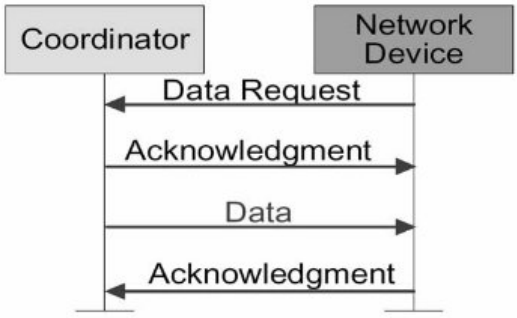


Figura 3.14: Trasmissione coordinatore-nodo Beacon-Disabled.

Tipi di frame a livello MAC

I tipi di frame definite dall'IEEE 802.15.4 sono di quattro tipi:

- Beacon Frame. Sono utilizzati , come detto, per delimitare una superframe e per coordinare la sincronizzazione tra nodo e coordinatore.(Figura 3.18)
- Data Frame. Questa frame ha un payload massimo di 104 byte. Il suo compito principale è la trsmissione dei dati ma fornisce altri servizi come assicurarsi di una tracciatura dei pacchetti e di eseguire controlli sull'integrità della PDU.(Figura 3.15)
- ACK Frame. Il compito dell'ACK frame è quello di fornire un feedback attivo dal ricevitore della corretta ricezione di un pacchetto.(Figura 3.16)
- MAC Command Frame. Essa consente meccanismi per il controllo e la configurazione dei vari nodi come visto per le comunicazioni tra coordinatore e nodo.(Figura 3.17)

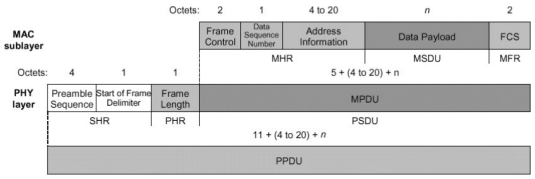


Figura 3.15: Formato Data Frame.

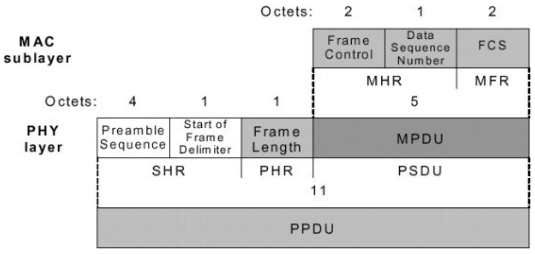


Figura 3.16: Formato dell'ACK Frame.

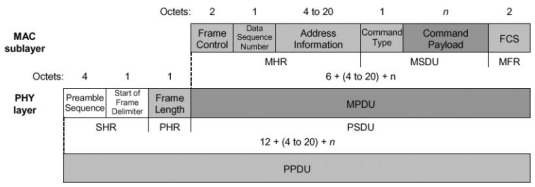


Figura 3.17: Formato della MAC command Frame.

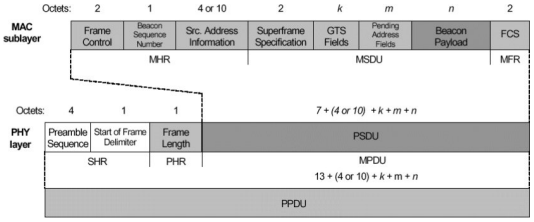


Figura 3.18: Formato della Beacon frame.

3.3.2 Standard IEEE 802.15.4a

Lo standard 802.15.4a [22; 7] è l'evoluzione dello standard 802.15.4 [3; 5]. La differenza sta nella composizione del livello fisico che può avere due diverse modalità: la prima, chiamata Chirp Spread Spectrum (CSS), la quale offre un data-rate massimo di 2 Mbps e più robustezza alle interferenze, la seconda è basata su Ultra Wide Band (UWB). Quest'ultima tecnologia è sviluppata per trasmettere e ricevere segnali mediante l'utilizzo di impulsi in radiofrequenza con durata estremamente ridotta. Offre la possibilità di avere un data-rate massimo di 10 Mbps ma con un raggio d'azione molto basso inferiore ai 10 metri. Entrambe le modalità non sono compatibili con lo standard 802.15.4.

3.3.3 Standard Bluetooth

Bluetooth fornisce un metodo standard (802.15.1) [4; 62], economico e sicuro per scambiare informazioni tra dispositivi diversi attraverso una frequenza radio sicura a corto raggio. La tecnologia Bluetooth opera anch'essa nella banda di frequenze tra 2,4 e 2,5 GHz (ISM), usando una modulazione FSK con un data-rate di 720Kbps. Le reti topologicamente più semplici che possono essere formate usando dispositivi Bluetooth vengono dette *piconet*. Esse sono composte al massimo da otto elementi. In ogni piconet si distingue un nodo, detto master, mentre gli altri sono detti slave. Il master funge da coordinatore e concorda la sincronizzazione. Lo slave funge da nodo passivo, accetta le condizioni dettate dal master. Un dispositivo può trovarsi in stadi intermedi, meno reattivi, ed è utilizzata la trasmissione adattiva per il risparmio delle batterie.

3.4 Protocolli di rete e applicativi

Basandosi sui primi due livelli definiti dagli standard sono stati sviluppati diversi protocolli di livello tre (network) e di livello quattro (applicativi). In commercio sono proposte varie soluzioni costruite da diverse case produttici.

3.4.1 ZigBee

Basato sulle specifiche dello standard 802.15.4, la tecnologia ZigBee [35] che implementa protocolli fino a livello applicativo.[10; 3; 5] Nella Figura 3.19 viene proposto un parallelo con la classica architettura ISO/OSI. Viene definita a basso costo perchè può essere utilizzata in svariate applicazioni e a basso consumo perchè offre alte

prestazioni sulla conservazione delle batterie. ZigBee Alliance è l'organismo che definisce i profili delle applicazioni rendendonli accessibili e gratuitamente.

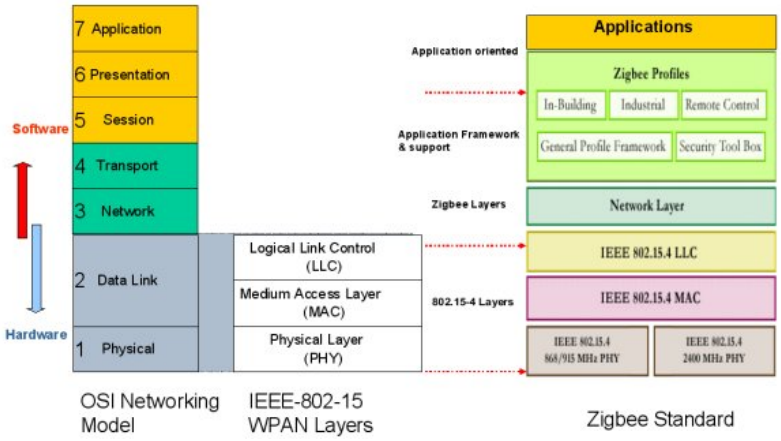


Figura 3.19: Protocolli implementati nello ZigBee.

Il livello network, dello ZigBee [49; 8], ha il compito di assegnare gli indirizzi ai nodi che compongono la rete, di sincronizzazione globale supportando regole di routing. I nodi devono essere pre-impostati con la funzione che essi devono assumere nella rete: nodo sensore, nodo coordinatore o entrambe le funzionalità. La fase di routing viene gestita in modi differenti a seconda della tipologia di struttura. Nelle reti a stella i collegamenti sono bi-direzionali nel senso che la comunicazione può avvenire da coordinatore a nodo e viceversa attraverso un unico link. Nelle reti mesh, invece, è possibile configurare più percorsi tra due nodi.
Le finalità, invece, implementate nel livello applicativo sono: scoprire se, tra dispositivi adiacenti entro il raggio d'ascolto, uno di loro sta operando, mantenere una relazione fra uno o più dispositivi a loro associati potendo colloquiare con messaggi di controllo oppure di esecuzione comandi.

3.4.2 6LoWPAN

Di ultimissima generazione è lo sviluppo di 6LoWPAN [54; 57]. Basato anch'esso sullo standard 802.15.4 implementano due tipologie di indirizzi: indirizzi estesi IEEE EUI-64 bit, univoci globalmente, o indirizzi a 16 bit, univoci soltanto all'interno della PAN. La possibilità di estendere il campo d'indirizzamento ha dato la possibilità di poter configurare reti di sensori che possono essere gestite direttamente dalla rete Internet ad indirizzamento IPv6. La tipologia pacchetto 6LoWPAN [45] offre il supporto per la comunicazione di reti mesh, per il broadcasting e multicasting e per la gestione della frammentazione dei pacchetti. Eredita dal protocollo IPv6 anche la funzionalità di autoconfigurazione dei nodi. I livelli implementati nella 6LoWPAN, per trasferire le informazioni dai livelli superiori ai livelli inferiori, prevedono, ovviamente, un adattamento del pacchetto non avendo a disposizione la possibilità di costruire frame di grandi dimensioni; infatti come visto in precedenza la massima grandezza consentita del protocollo 802.15.4 è 127 bytes contro i 1260 bytes dell'IPv6.

3.5 Conclusioni

L'802.15.4 è uno standard completo e piuttosto diffuso anche se alcune delle sue funzionalità più avanzate come, per esempio, la modalità Beacon-Enabled, difficilmente viene adottata all'interno delle WSN. Sebbene sia attualmente oggetto di numerose ricerche, è invece ancora poco diffuso lo standard 6LoWPAN che, grazie al supporto di IP, rende possibile l'interfacciamento di una WSN con la rete Internet, consentendo ad esempio ad un utente di comandare una sensore da remoto. Allo stesso modo, numerose ricerche più recenti riguardano il comportamento e l'ottimizzazione dei processi nei sistemi operativi real time per le Wireless Sensor Network.

Anche se la piattaforma SquidBee non supporta attualmente 6LoWPAN e nemmeno i sistemi operativi TinyOS o Contiki, è stata individuata come la piattaforma più adeguata per lo sviluppo del sistema W-TREMORS. Infatti la soluzione basata sulla combinazione hardware Arduino/XBee risulta essere la più intuitiva ed *user-friendly*; inoltre può essere programmata sia nel linguaggio standard C che in Wiring, semplice e facilmente comprensibile anche dai "non-addetti ai lavori". Lo schema circuitale della board Arduino permette poi di interfacciare molti sensori esistenti con uno sforzo minimo e, analogamente, il modulo radio XBee permette di impostare con estrema facilità numerosi parametri di funzionamento di livello 1 e 2.

Capitolo 4

Structural Health Monitoring (SHM)

Primavera, 1877. E il periodo più lungo che abbia mai passato in un solo posto da quando ho lasciato la fattoria a 17 anni. Ci sono tante cose qui che non capirò mai. Non sono mai stato un frequentatore di chiese e quello che ho visto sui campi di battaglia mi ha spinto ad interrogarmi sui disegni di Dio. Ma c'è indubbiamente qualcosa di spirituale in questo luogo e sebbene possa rimanere eternamente oscuro per me non posso che essere consapevole del suo potere.

dal film *L'Ultimo Samurai (The Last Samurai, 2003)*

Il settore dello Structural Health Monitoring (SHM) si occupa di analizzare strutture sottoposte a sollecitazioni, siano esse di tipo naturale (vulcani, terremoti, vento) [44] o provocate da attività umane. Con il termine strutture si vuole intendere non solo palazzi, case, ponti [26] ma anche componenti aereo-spaziali o parti di navi. Il compito principale di queste attività è fornire indicazioni sullo stato di integrità delle strutture stesse.

Un esempio pratico è quello del monitoraggio di una torre campanaria, provvista di un orologio collegato ad una campana. In questo tipo di struttura la campana, segnando le ore attraverso rintocchi, provoca vibrazioni nella struttura e, a lungo andare, può comportare danni alla struttura e provocare la caduta di frammenti di laterizio ed intonaco. In casi come questo vengono vengono dislocati dei sensori di accelerazione o inclinazione sulla torre a differenti altezze e si studia quindi in diversi punti la risposta della struttura alle vibrazioni. Lo scopo finale è quello di identificare la frequenza e l'intensità massima delle sollecitazioni che la torre può sopportare senza che esista il pericolo di danneggiamento o crollo.

Di seguito verranno presentati alcuni esperimenti di monitoraggio di strutture tramite WSN realizzate in passato, inoltre verrà presentata la piattaforma realizzata nell'ambito del progetto di tesi.

4.1 Esempi di utilizzo dello SHM

Alcuni esperimenti sull'utilizzo delle reti wireless di sensori nel campo dello Structural Health Monitoring sono già stati eseguiti, fornendo peraltro ottimi riscontri. Verranno illustrate di seguito alcune applicazioni sperimentate dai ricercatori in questo settore.

4.1.1 Vincent Thomas Bridge

Un primo esperimento è stato eseguito per analizzare le condizioni del ponte di Vincent Thomas Bridge a Long Beach, in California. La finalità principale di tale misurazione era quella di comparare le nuove acquisizioni con vecchie misurazioni, effettuate antecedentemente. La topologia adottata per la rete wireless di misurazione è del tipo a stella, in cui ogni nodo sensore trasferiva i propri dati ad un unico nodo centrale. Le posizioni dei sensori sul ponte hanno rispettato quelle delle simulazioni fatte precedentemente: due sensori sono stati posizionati orizzontalmente (laterale e longitudinale) ed uno verticalmente. La raccolta dei dati è stata effettuata ad una frequenza di 512 Hz per nodo, facendo durare l'applicazione per 5 minuti e ripetendo l'esperimento per tre volte.

Figura 4.1: Il Vincent Thomas Bridge.

4.1.2 Ponte Pedonale a Berkeley

Un secondo esperimento ha riguardato l'analisi di un ponte pedonale a Berkeley e, in questo caso, i ricercatori hanno affrontato il problema in modo differente. La rete disposta sul ponte era costituita da 13 nodi, aventi però compiti differenti: 10 di essi sono stati destinati alla cattura delle oscillazioni, mentre i rimanenti 3 sono stati utilizzati per il controllo della calibratura e della variazione degli assi. Al conteggio dei nodi citati occorre aggiungere una *base station*, avente il compito di raccogliere e archiviare i dati. Poiché i nodi sono stati disposti in fila lungo tutto il ponto,

è stata costruita una rete di tipo *multi-hop*. La prova eseguita è durata per circa quattro minuti, campionando ad 200 Hz. In realtà ogni sensore ha effettuato un campionamento ad una frequenza pari ad 1 KHz, calcolando poi la media ogni 5 acquisizioni e trasmettendo solo quest'ultimo valore.

Visti i buoni risultati ottenuti sempre su questa linea, i ricercatori stanno preparando un nuovo esperimento sul ponte Golden Gate Bridge [41]. Per questa futura prova i ricercatori prevedono di utilizzare una rete composta da ben 59 nodi, dislocati in diversi luoghi del ponte: 51 lungo un lato e 8 posizionati sui piloni che lo sostengono. Questo esperimento servirà per testare questa tipologia di rete in realtà più complesse e, naturalmente, fornirà dati importanti agli ingegneri civili.

4.1.3 Geumdang Bridge

Un esperimento simile ai precedenti è stato eseguito anche in Corea del Sud e, più precisamente, sul Geumdang Bridge [50] nella regione di Icheon. L'obiettivo primario dello studio è stato quello di accertare, usufruendo di una rete composta da 14 nodi, l'attendibilità dei dati già in possesso degli ingegneri sul comportamento del ponte. I nodi sono stati posizionati negli spazi all'interno delle travi maestre; i test hanno previsto acquisizioni dati sia in assenza di traffico che in presenza di attraversamento di auto-articolati a diverse velocità e con pesi differenti. Accanto di questa rete di sensori è stata posizionata una rete cablata, impiegata per confrontare successivamente le misurazioni catturate. Il campionamento eseguito è stato di 200 Hz per la rete cablata mentre per la rete wireless sia a 200 che a 70 Hz. La tipologia di rete, per i sensori, dispiegata lungo il ponte, era di tipo a stella. Dopo aver effettuato le prove, i campioni, acquisiti con i due sistemi, sono stati comparati: il segnale acquisiti a 200 Hz sono risultati identici, dimostrando una ottima affidabilità del sistema wireless. Peraltro, anche diminuendo la frequenza di campionamento del sistema wireless a 70 Hz, è stato osservato che, calcolando lo spettro del segnale e sovrapponendo poi i grafici risultanti, la correlazione tra i due segnali rimane pressoché identica.

4.1.4 SHM in altre strutture

Un differente esperimento [11] è stato eseguito su un palazzo di Los Angeles, dichiarato inagibile dopo un terremoto nel 1994. Per realizzare la rete di misurazione sono stati utilizzati dei nodi Mica-2 e Mica-Z. Questo tipo di sensori offre un rate di campionamento a 500 Hz: le prove sul campo sono state tuttavia effettuate a frequenze di campionamento inferiori, ottenendo buoni risultati, in termini di perdita di pacchetti, campionando fino a 370 Hz. La topologia della rete, visibile nella Figura 4.2, di misura realizzata era di tipo *mesh*, con un'unica base station per la raccolta dei campioni; alcuni nodi all'interno della rete avevano anche la possibilità di utilizzare percorsi alternativi i caso di sovraccarico dei nodi principali. Questa analisi è servita anche per testare l'affidabilità di una rete mesh per questo tipo di applicazioni. Infatti, in parallelo alla rete di sensori wireless, è stata posizionata anche una rete cablata. Sono state quindi eseguite prove per identificare danneggiamenti, sia a livello globale di struttura che a livello localizzato. Attraverso netSHM, un programma formato da applicazioni in linguaggio C e in Matlab e appositamente

scritto per questo esperimento, è stato possibile valutare la robustezza dei diversi piani della struttura, identificando anche quale era quello maggiormente danneggiato.

E' interessante notare come entrambe le reti di misurazione, quella cablata e quella wireless, abbiano fornito dati sostanzialmente equivalente. Tuttavia, per ogni esperimento eseguito, i tecnici della rete wired hanno necessitato di trenta minuti di *set-up* per il posizionamento e l'avvio; al contrario, il sistema wireless ha mostrato grande flessibilità e facilità di riposizionamento dei sensori.

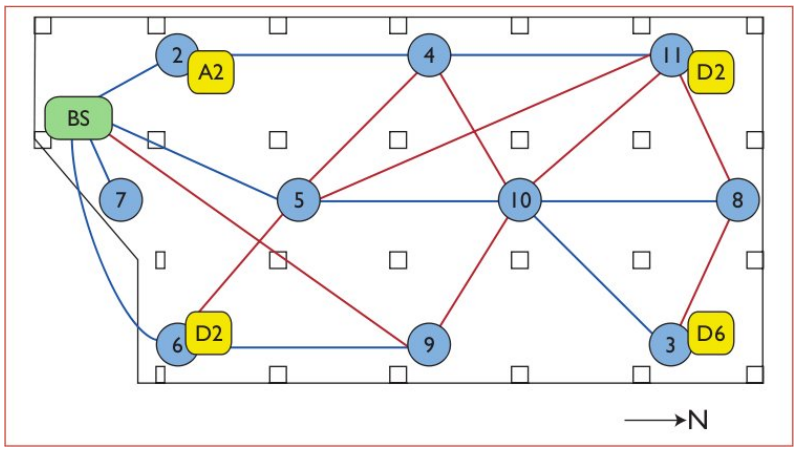


Figura 4.2: Schema di rete utilizzata per il monitoraggio dei piani.

4.1.5 IVHM

Un ramo della SHM, chiamato IVHM (Integrate Vehicle Health Monitoring), si occupa di sistemi di monitoraggio sullo stato di strutture su veicoli in speciale modo su aerei [14]. Questi sistemi di sensori servono per rilevare e tenere traccia di eventuali danneggiamenti alle strutture dei componenti. Sistemi di IVHM cominciano ad essere sempre più spesso inseriti negli aerei di linea commerciali. Questo tipo di sistemi di rilevazione riesce a produrre utilissime informazioni sullo stato dell'aereo, segnalando eventuali rotture e agevolando le operazioni di manutenzione. In diversi casi l'introduzione di questi sistemi nell'aviazione ha permesso alle officine addette alla manutenzione di poter intervenire in modo preciso e in tempo utile, prima che gli eventuali guasti diventassero pericolosi. Le misurazioni possono essere eseguite attraverso vari tipi di sensori come fibre ottiche, MEMS o attraverso l'ausilio di accelerometri piezoelettrici. Lo stato di danneggiamento è stimato catturando una serie di dati dai sensori e applicandoli ad un modello strutturale caratteristico della struttura.

L'IVHM è stato sperimentato *in primis* dalla Agenzia Spaziale Americana sugli Shuttle, fornendo ai piloti un valido aiuto ad identificare rotture o probabilità di falle. Tra i vari esperimenti si ricorda l'utilizzo di sensori di vibrazione sugli scudi termici del vettore; i testi hanno permesso di disporre di numerosi dati in tempo reale, incrementando notevolmente il livello di sicurezza degli astronauti a bordo.

In questo settore i sistemi di monitoraggio wireless non sono ancora diffusi:

data la criticità di molti applicazioni, l'inaffidabilità del mezzo trasmissivo è oggi il principale ostacolo alla diffusione di Wireless Sensor Network per l'IHVM.

4.2 W-TREMORS

In questo settore è attivamente impegnata Eucentre, una Fondazione senza scopo di lucro fondata dal Dipartimento della Protezione Civile, dall'Istituto Nazionale di Geofisica e Vulcanologia, dall'Università degli Studi di Pavia e dall'Istituto Universitario di Studi Superiori di Pavia, con il fine di promuovere, sostenere e curare la formazione e la ricerca nel campo della riduzione del rischio sismico. Il sistema di acquisizione dati attualmente utilizzato dai ricercatori di Eucentre è realizzato con apparecchiature ottimizzate per questo tipo di applicazioni e prodotto da National Instrument. Il numero di sensori che possono essere collegati e la banda disponibile sono elevati, ma ogni dispositivo di misurazione deve essere collegato al sistema di acquisizione dati con un cavo dedicato. Proprio per superare quest'ultimo limite si è deciso di sostituire l'attuale sistema di acquisizione con una WSN appositamente sviluppata. I vantaggi di un monitoraggio wireless sono evidenti, dal momento che la soluzione radio è in grado di semplificare notevolmente le operazioni di installazione dei sensori. La piattaforma del progetto è stata chiamata W-TREMORS (Wireless Tremors, vibRations and Earthquakes MOnitoRing System). Gli hardware utilizzati sono: modulo radio XBee mentre per il microprocessore è stato utilizzato un ATMega128 della Arduino visibili nella Figura 4.3. Come verrà spiegato successivamente nei capitoli Capitolo 5 e Capitolo 6 sono stati affrontati problemi di sincronizzazione e di accesso al canale implementando nuovi algoritmi sulla base di quelli disponibili in letteratura. I risultati delle prove sono riportati, successivamente, nel Capitolo 7.

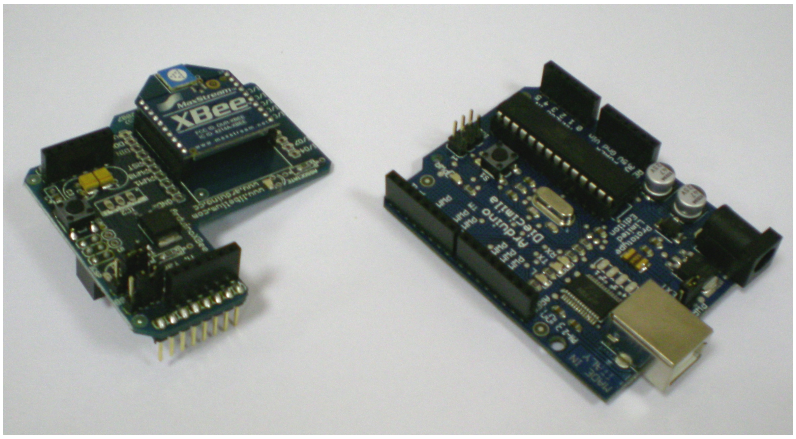


Figura 4.3: Hardaware utilizzati nella piattaforma W-TREMORS.

Capitolo 5

Sincronizzazione

...Fiducia, responsabilità assumerti il peso delle scelte
e dei sentimenti passare il resto della vita tenendo
fede a questi e soprattutto non ferire l'oggetto del tuo
amore.
E' questo l'amore...?
...Moltiplicalo all'infinito, portalo negli abissi
dell'eternità e comunque vedrai appena uno spiraglio
di quello di cui parlo.

dal film *Vi presento Joe Black (Meet Joe Black, 1998)*

Il problema della sincronizzazione [27] di host in una rete è un problema molto sentito, specialmente nei sistemi distribuiti. E' impossibile garantire che i clock di ogni singolo dispositivo avanzino tutti alla stessa ed esatta frequenza. I problemi di clock possono essere di natura fisica o di implementazione software. Il primo problema può essere spiegato facendo riferimento all'oscillatore per la scansione del tempo all'interno di ogni dispositivo. Per quanto la costruzione industriale di tali apparecchi possa essere precisa, ognuno si differenzia dall'altro scandendo il passare del tempo in maniera più veloce o più lenta. Il secondo problema, invece, può derivare dalla scrittura di codice poco ottimizzato e non sincronizzato.

La time-synchronization è stata soggetta a lunghi studi e sono stati sviluppati molti algoritmi per diverse applicazioni.

5.1 Algoritmi tradizionali per la sincronizzazione

In questa sezione vengono descritti alcuni algoritmi di sincronizzazione classici [43; 39; 51] che fanno da riferimento a quelli implementati, con opportuni accorgimenti, per le WSN.

5.1.1 Algoritmo di Lamport

Lamport definisce [46], per la sincronizzazione dei clock, la relazione:

Happens →Before

indicata con il simbolo →

Definendo con a e b due eventi, la relazione può avere i seguenti significati:

- a →b significa che l'evento a avviene prima dell'evento b;
- se a e b sono due eventi di uno stesso processo e a avviene prima di b, allora a →b è vera;
- se in due processi a è l'evento di invio di un messaggio m e b è l'evento di ricezione del messaggio m, allora a →b è vera;
- se a →b e b →c allora a →c.

Per rappresentare ogni evento è necessario definire una misura globale del tempo valida per tutti i processi. Questa viene definita come C(evento). Tramite questa definizione si può dire che se a →b allora C(a) < C(b); in termini pratici, significa che il tempo misurato all'accadere dell'evento a è minore del tempo dell'evento b perché avviene prima. I tipi di processo a cui Lamport si riferisce sono non concorrenti nel senso che non devono accedere alle stesse risorse, ma sono processi interagenti. Infatti si dice che: se due processi non interagiscono non è necessario sincronizzare i loro clock; ciò che è importante per due o più processi interagenti è rispettare l'ordine corretto in cui gli eventi avvengono.

L'algoritmo di Lamport può essere definito come segue:

- Ogni messaggio contiene il tempo del proprio invio sul mittente (basato sul clock del nodo mittente).
- Quando un messaggio arriva il clock del ricevente deve essere maggiore di almeno un tick del tempo del mittente (indicato nel messaggio).
- Tra due eventi il clock deve avanzare almeno di un tick; infatti non si possono avere due eventi che accadono nello stesso istante di tempo.

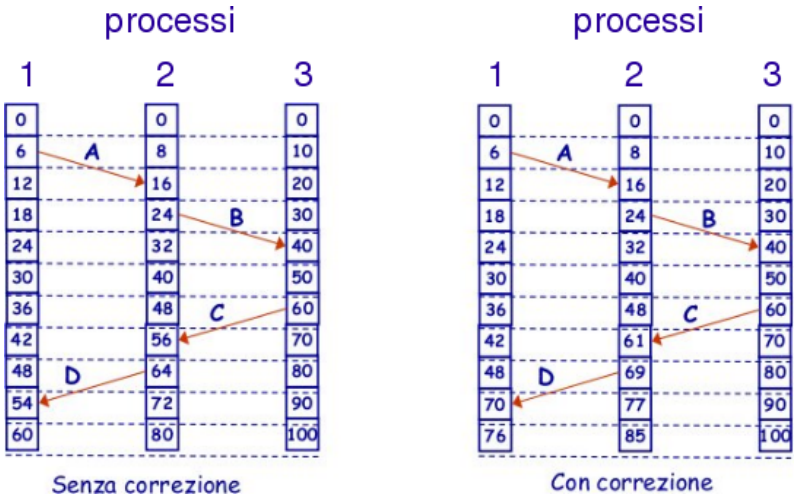


Figura 5.1: Esempio di funzionamento dell'algoritmo di Lamport.

L'algoritmo di Lamport è spesso utilizzato in database replicati e distribuiti. Nell'esempio riportato in Figura 5.1 si hanno tre dispositivi che si scambiano informazioni. I tre processi, avendo dei clock fra loro sfasati, per eseguire correttamente

gli eventi nell'ordine A-B-C-D necessitano di correzioni in modo che i clock dei processi successivi siano di almeno un tick superiore al precedente, per eseguire la sequenza in modo corretto e non trovarsi difronte ad incongruenze.

5.1.2 Algoritmo di Cristian

L'algoritmo di sincronizzazione di Cristian [15; 67] viene detto algoritmo di sincronizzazione esterna. Con il termine "esterna" ci si riferisce al fatto che ogni processo chiede il tempo ad un dispositivo esterno che ha la funzione di Time-Server. Il meccanismo di funzionamento è molto semplice: un processo fa richiesta di sincronizzazione mandando un messaggio al Time-Server, attendendo una risposta contenente il timestamp del server. Il processo a questo punto imposta il proprio clock al tempo inviato dal server aggiungendo la media del Round Trip Time (RTT). RTT è il tempo che intercorre tra l'invio di richiesta di sincronizzazione e la risposta del server ed è registrato e calcolato dal processo stesso. Lo schema temporale è visibile nella Figura 5.2.

$$T_{corrente} = T_{server} + \frac{RTT}{2}$$

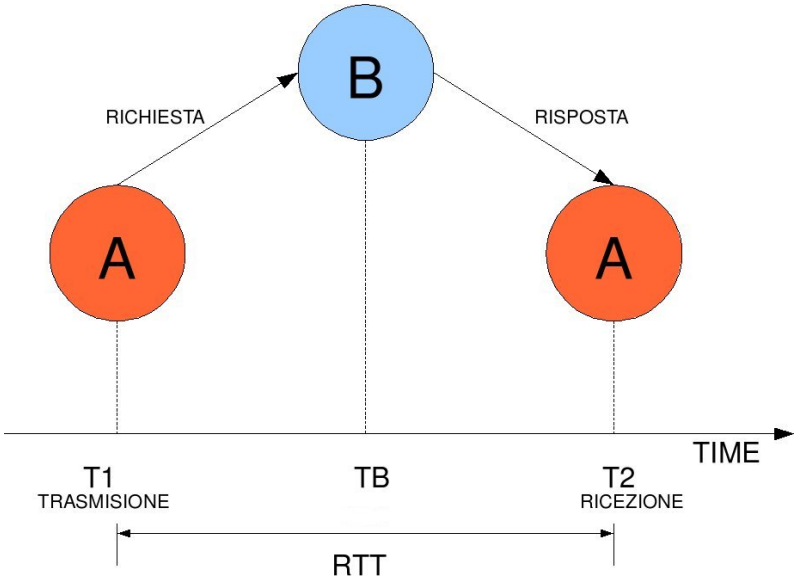


Figura 5.2: Esempio di funzionamento dell'algoritmo di Cristian.

Questo tipo di algoritmo è implementato nel Network Time Protocol (NTP) [59] ed è usato molto nelle applicazioni in Internet.

5.1.3 Algoritmo di Berkeley

Anche per questo algoritmo [37] ci si appoggia ad un server ma, in questo caso, il server non fornisce un valore esatto del tempo come invece accade per l'algoritmo di Cristian. Al contrario, funge da master e chiede agli altri nodi, che compongono la rete, di fornire il loro clock. Collezionate tutte le differenze dei clock ricevuti, ne

costruisce un valore medio. Il server, dopo aver eseguito il calcolo, spedirà a tutti il nuovo clock aggiornando anche se stesso.

Per meglio illustrare questo algoritmo verrà di seguito presentato un semplice esempio reale di funzionamento. Al primo step (Figura 5.3) il master interroga tutti mandando il proprio tempo.

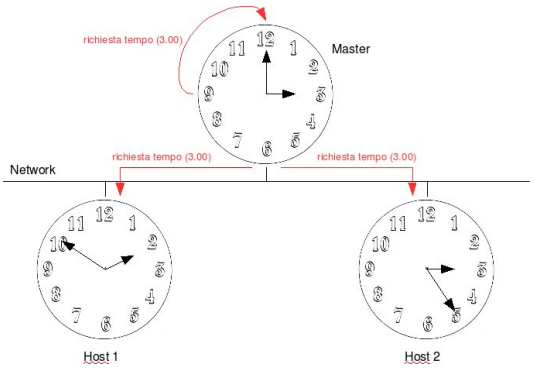


Figura 5.3: Il master interroga gli host.

Al secondo step (Figura 5.4) gli host rispondono con la differenza tra il loro clock e quello spedito dal master.

$$Differenza_{Host1} = 2.50 - 3.00 = -10$$

$$Differenza_{Host2} = 3.25 - 3.00 = +25$$

$$Differenza_{Master} = 3.00 - 3.00 = 0$$

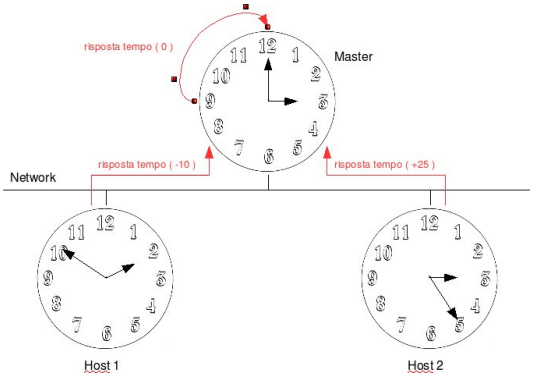


Figura 5.4: Risposta degli host.

All'ultimo step (Figura 5.5) calcola la media delle differenze e invia gli aggiornamenti, in modo che i vari host possano spostare i clock in base al valore indicato.

$$Media = \frac{Differenza_{Host1} + Differenza_{Host2} + Differenza_{Master}}{3} = -10+25+0 = +5.$$

$$NewClock_{Master} = 3.00 + 5 = 3.05$$

$$\Delta_{Host1} = 3.05 - 2.50 = +15$$
$$\Delta_{Host2} = 3.05 - 3.25 = -20$$

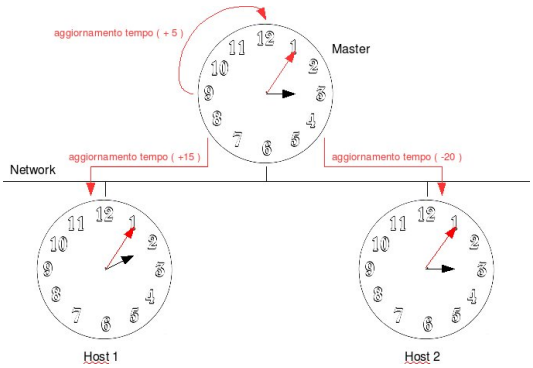


Figura 5.5: Aggiornamento della rete.

5.2 Algoritmi di Sincronizzazione per le Wireless Sensor Network

In letteratura sono stati proposti per le WSN una serie di algoritmi [39; 43] che, spesso, ricalcano gli algoritmi classici. In ogni caso tutti questi algoritmi utilizzano una corrispondenza attraverso messaggi fra nodi al fine di aggiornarsi su un clock unico.

5.2.1 Reference Broadcast Synchronization (RBS)

Questo protocollo utilizza un segnale spedito dal master in broadcasting verso tutti i nodi per avviare il processo di sincronizzazione [28]. Si consideri una rete composta da tre nodi A,B,C, come è possibile vedere nella Figura 5.6, dove B è il nodo-master della rete. Il nodo-master diffonde in broadcast un segnale (messaggio), chiamato *reference*, verso A e C simultaneamente il quale non contiene alcun tipo di informazione sul tempo (in questo tipo di algoritmo vengono trascurati eventuali ritardi di trasmissione). Al momento della ricezione i due nodi fissano il loro tempo locale. Il nodo A e C poi cambieranno il loro tempo locale attraverso messaggi separati tra loro escludendo il nodo master. Questo è sufficiente, ai due ricevitori, per determinare il loro relativo offset al tempo della ricezione del reference inviato da B. Questo protocollo può essere esteso ad un numero elevato di nodi. Lo schema base è simile al meccanismo chiamato CesiumSpray utilizzato dai nodi sensori equipaggiati da un GPS usato per la localizzazione. Un aspetto chiave del protocollo RBS è che elimina completamente la parte di settaggio del tempo da parte del master. Tuttavia, poichè non vengono considerate in alcun modo le tempistiche di propagazione, è fondamentale che il segnale di *reference* giunga contemporaneamente su tutti i nodi coinvolti nel processo. RBS può essere esteso anche a reti multi-hop. Infatti, i nodi, posizionati in regioni dove riescono a sentire più reference spedite dai nodi master, possono determinare le relazioni sia tra clock locali sia tra le reference spedite dai master. Questi nodi vengono chiamati nodi “ponte”.

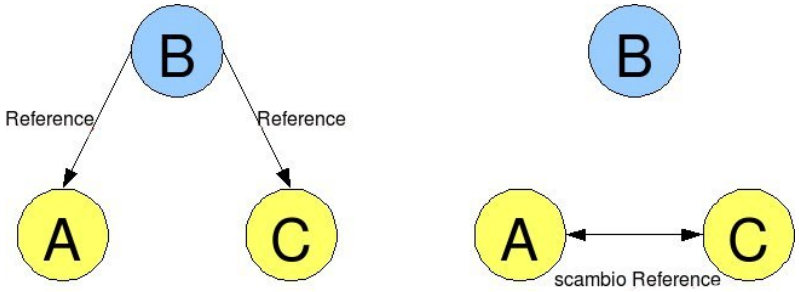


Figura 5.6: Algoritmo RBS.

5.2.2 Timing-Sync Protocol for Sensor Network (TPSN)

L'algoritmo TPSN [36] esegue una sincronizzazione classica tra mittente e destinatario molto simile all'algoritmo di Cristian.

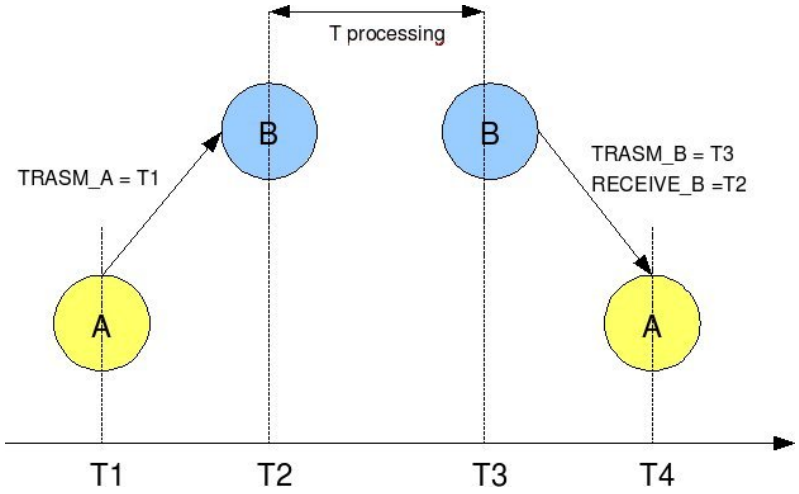


Figura 5.7: Algoritmo TPSN.

Come mostrato in Figura 5.7, il nodo A trasmette un messaggio inserendo il proprio valore di clock al tempo T1. B riceve il pacchetto e lo marca con il tempo di ricezione T2. In T3, B spedisce un messaggio ad A con due informazioni: il proprio clock e il tempo T2. A, infine, marca con il proprio tempo T4 l'arrivo del pacchetto. In questo modo A riesce a costruirsi, avendo tutti i tempi intermedi, sia l'offset di clock sia i tempi di propagazione dei pacchetti. Questo algoritmo permette di conoscere anche il T di processing di un nodo dalla ricezione di un pacchetto alla spedizione. TPNS è un algoritmo molto utilizzato nelle reti con struttura ad albero. I nodi che stanno al primo livello gerarchico dell'albero si sincronizzano con la root. Ricorsivamente i livelli sottostanti con quelli superiori fino ad arrivare ai nodi foglia. L'intera rete sarà sincronizzata con il clock del nodo radice.

5.2.3 Flooding Time-Synchronization Protocol (FTSP)

Il protocollo FTSP [52] è utilizzato per sincronizzare una rete di grandi dimensioni. Un nodo viene eletto come master e il suo clock sarà quello di riferimento per tutti i nodi appartenenti alla rete. Il nodo eletto master (di norma viene eletto quello con ID minore) manda in broadcasting un messaggio di sincronizzazione, inserendo il proprio clock, a tutti i nodi che riescono a sentirlo. I nodi che ricevono il messaggio a loro volta lo propagano agli altri nodi spedendo il loro offset e il tempo della ricezione. I nodi successivi correggono l'errore di propagazione costruendo una regressione lineare dei valori ricevuti. Confrontando il proprio tempo riescono a recuperare l'offset a cui si devono allineare. Affinchè la regressione lineare abbia una validità si consiglia che ogni nodo riceva almeno otto coppie di *(offset, clock)* da altri otto nodi. Questa tecnica viene implementata nelle reti multi-hop ed è più performante se collocata al livello 2 dello stack protocollare.

5.3 Sorgenti d'errore

In generale i messaggi per la sincronizzazione possono avere degli errori di latenza, spesso trascurabili [29]. Questi errori possono essere comunque divisi in quattro tipologie.

5.3.1 Tipologie di errore

- Tempo di spedizione *(Send-time)*: è l'intervallo di tempo che intercorre tra la costruzione del messaggio e la sua spedizione.
- Tempo di accesso *(Access-time)*: si riferisce a quanto tempo il messaggio aspetta nel buffer prima di essere spedito. Questo tempo è puramente casuale: infatti è possibile che ci siano nodi con il buffer vuoto i quali possono spedire il messaggio immediatamente oppure nodi che accodano il messaggio perché devono spedire altre informazioni.
- Tempo di propagazione *(Propagation-time)*: indica il tempo di "viaggio" di un messaggio tra due nodi. E' irrilevante per reti punto-punto con nodi vicini.
- Tempo di processamento del messaggio *(Receive-time)*: coincide con il tempo di risalita dello stack architetturale del messaggio, dall'arrivo nel livello fisico al livello più alto.

5.3.2 Drift

Particolare attenzione deve essere infine posta su una diversa e tipica sorgente d'errore: i drift. Come ricordato all'inizio del capitolo, ogni componente calcola lo scorrere del tempo in modo differente: questo fenomeno, dovuto alle variazioni fisiche dell'orologio, fa divergere il clock nel tempo rispetto al tempo reale. In letteratura viene questo tipo di problema viene chiamato drift. Il drift-rate è quindi il disallineamento graduale di un clock nel tempo. Per esempio, orologi al quarzo ad alta precisione hanno un tasso di disallineamento di $10^{-7} - 10^{-8}$. Per determinare se il drift-rate hardware di un componente è trascurabile, questo si deve mantenere

all'interno di una soglia. Per le applicazioni in oggetto del nostro lavoro, il drift-rate massimo tollerato è pari a 10^{-6} [34].

5.4 Soluzioni adottate in W-TREMORS

La tipologia di rete scelta per la costruzione della WSN nel settore dell'SHM è a stella. In questa tipologia un nodo avrà i compiti di coordinare e di raccogliere i dati mentre i nodi-sensori avranno i compiti di catturare i dati stessi e di trasmetterli. Questa scelta è basata sul tipo di acquisizione che viene fatta: infatti l'acquisizione attraverso accelerometri è una applicazione altamente dinamica dove sono necessarie tempestività e velocità di cattura dei dati. Una rete di tipo mesh avrebbe comportato uno spreco di risorse soprattutto perchè ogni sensore non avrebbe dovuto solo preoccuparsi di catturare i dati ma anche di eseguire funzioni di routing e forwarding, il che, con le capacità limitate dei processori, avrebbero comportato un abbassamento delle prestazioni di ogni singolo nodo.
Per quanto riguarda il problema dei drift sono state eseguite alcune prove che hanno calcolato la cadenza temporale del micro-processore collocato sul nodo. La risoluzione massima possibile per la segnalazione del tempo sugli AVR è il milli secondo. Le prime prove effettuate sono state nel conteggiare i clock di un micro-processore rispetto al clock del PC. I risultati ottenuti hanno messo in evidenza un forte drift da parte degli AVR [13] guadagnando in un'ora tra i 0,7 e 0,9 secondi. In un'analisi più approfondita dei dati si è trovato che il calcolo del tempo fornito dal PC è un calcolo software e quindi non affidabile. L'inaffidabilità si è potuta notare anche dalla scansione dei tempi degli AVR i quali tra loro erano vicinissimi se non nulli. La prova di verifica effettuata è stata quella di confrontare l'avanzamento del tempo del micro-processore paragonandolo con un un cronometro. Il risultato ottenuto è stato un disallineamento a favore dell'Arduino [12] di circa 0,300 secondi dopo un'ora, come è possibile vedere nella Figura 5.8 .

Il suo drift-rate è quindi presto calcolato:

$$\frac{0,300s}{3600s} \approx 83 \cdot 10^{-6}$$

il che significa che lavorando con i millisecondi ogni micro-processore si sfasa di 1 millisecondo circa ogni dodici secondi dal tempo reale:

$$\frac{10^{-3}}{83 \cdot 10^{-6}} \approx 12 \qquad s$$

Come si può notare dalla Figura 5.9, fra stessi processori invece il drift è pressochè nullo. Esiste, infatti, un disallinamento tra un AVR e l'altro di circa 6 millisecondi dopo un'ora il che ci porta a calcolare un drift-rate di:

$$6 \cdot \frac{10^{-3}}{3600} \approx 1,67 \cdot 10^{-6}$$

trovando uno sfasamento di un millisecondo ogni dieci minuti:

$$\frac{10^{-3}}{1,67 \cdot 10^{-6}} \approx 10 \qquad minuti$$

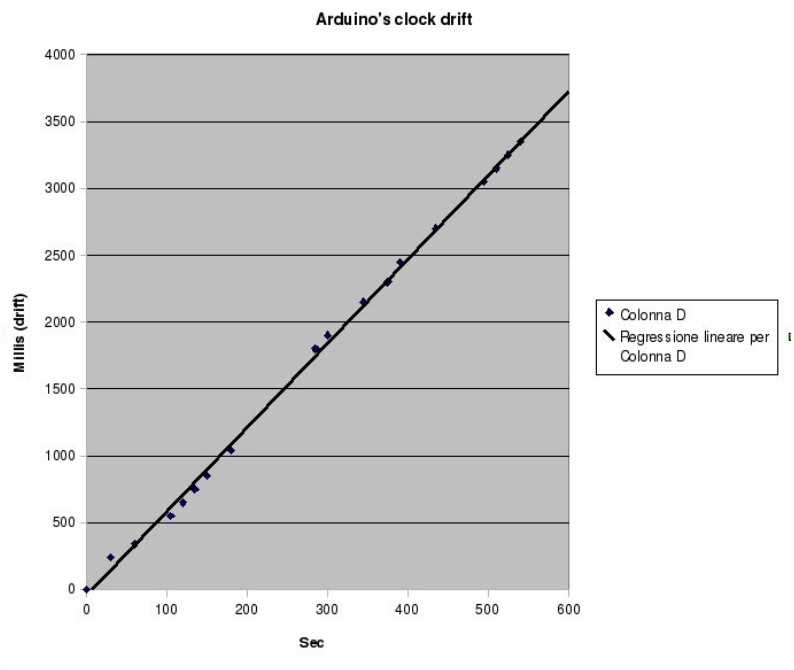


Figura 5.8: Drift Arduino con il tempo.

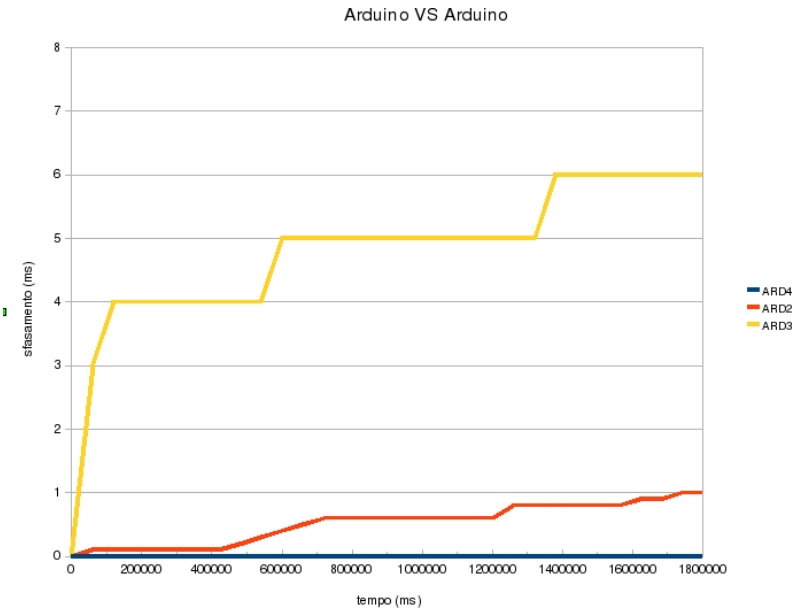


Figura 5.9: Confronto drift tra micro-processori.

Per la costruzione dell'applicazione ciò che più importa non è il raffronto con un tempo assoluto bensì con un tempo relativo segnalato ad un preciso istante. Ciò che si vuole ottenere è che ad un preciso istante di tempo tutti i componenti della rete partano contemporaneamente a contare il tempo e che, ad un preciso istante di tempo, ogni sensore attribuisca il valore catturato. I drift calcolati non influenzano pesantemente la piattaforma che verrà costruita per due motivi: il primo motivo è che le prove non si protraggono per periodi di lunga durata, il secondo che la perdita di un millisecondo è per tutti i componenti AVR che fanno parte della rete e non solamente da parte di un singolo.
Il tempo relativo per l'applicazione sarà costruito similmente rifacendosi al protocollo RBS. La scelta di costruire una tipologia a stella ha agevolato l'implementazione di tale algoritmo. Nella struttura costruita, è presente un nodo sensore, che verrà denominato d'ora in poi gateway, con funzioni da master: quando l'applicazione dovrà attivarsi per la cattura dei valori, il gateway invierà un messaggio in broadcasting a tutti i sensori dando lo start al conteggio del tempo. In questo modo tutti i sensori partiranno nello stesso istante. Altre fonti di errore non intervengono a modificare in modo significativo il tempo. I motivi sono abbastanza semplici: su ogni processore è montato lo stesso identico programma, sicchè gli errori che possono essere commessi da un nodo sono uguali a tutti quelli commessi da parte degli altri. Una considerazione particolare si vuole fare invece sul tempo di propagazione del segnale. Questo tempo, che può portare a una sorgente di errore, interviene se due enti comunicanti si trovano ad una distanza rilevante (problema non di facile soluzione e che si evidenzia nelle reti wired); tuttavia per le caratteristiche intrinseche del protocollo 802.15.4 i nodi delle reti non possono distanziare più di una decina di metri tra di loro altrimenti si troverebbero isolati e non potrebbero comunicare con gli altri host della rete trovandosi fuori dal raggio di comunicazione. Restando in un raggio di circa 10 metri il tempo di propagazione diventa ininfluente, nell'ordine dei microsecondi:

$$T_{propag} = \frac{\text{dist}}{\frac{1}{3} \cdot c} = \frac{10}{99930819} \approx 0,1 \quad \mu s$$

Nelle WSN può nascere questo tipo di problema se vengono costruite reti di tipo multi-hop dove i nodi sensori commutano i pacchetti verso altri componenti della rete o verso una stazione destinataria. In questo tipo specifico di rete, soprattutto se molto grande con tanti hop da eseguire, il tempo di propagazione deve essere tenuto in considerazione. Nel modello costruito non è stata implementata alcuna correzione in real-time per il fatto che si potrebbe inficiare le prestazioni di ogni singolo nodo: bisogna infatti sempre ricordare che il micro-processore dei nodi ha capacità ridotte e ogni operazione che gli si chiede di svolgere costa nel calcolo computazionale. E' possibile comunque correggere tale errore una volta ottenuto un file di output dei dati acquisiti.

Capitolo 6

Trasmissione delle acquisizioni

Grazie per avermi fatto sentire sempre orgoglioso di te
per la tua forza, per la tua dolcezza, per come eri e
come sei, per come ho sempre desiderato toccarti.
Oddio eri tutta la mia vita e ti chiedo scusa per tutte
le volte che ho fallito con te. Specialmente questa.

dal film *Al di là dei Sogni (What Dreams May Come, 1998)*

La realizzazione di una rete di sensori wireless richiede l'utilizzo di tecniche di rete specifiche; anche se molti protocolli e algoritmi sono stati proposti in letteratura per la realizzazione di reti wireless, questi non sono completamente adattabili a livello universale, a causa delle specifiche necessità e delle particolarità delle WSN. La piattaforma W-TREMORS necessita di sincronizzazione, è in real-time e, infine, deve essere regolarizzata nelle trasmissioni. Per avere questi requisiti si devono affrontare e risolvere problematiche quali l'utilizzo del canale, implementando protocolli per l'accesso multiplo.
In questo capitolo verranno presentate di volta in volta le soluzioni adottate per riuscire a soddisfare tutti i requisiti necessari affinchè si possa ottenere un'acquisizione corretta ed affidabile.

6.1 Vincoli e requisiti

Come visto nel Capitolo 5, per fare in modo che tutti i nodi inizino ad acquisire nello stesso istante, si è deciso di utilizzare l'algoritmo RBS per la sincronizzazione. Tale algoritmo permette a tutti i sensori di partire a conteggiare il tempo nell'istante in cui il gateway segnala in broadcast l'inizio della applicazione.

L'applicazione implementata è mono-task, ciò implica che i nodi eseguano sequenzialmente una operazione alla volta; nel momento in cui un nodo spedisce esclude, automaticamente, la possibilità di acquisire e viceversa. Questo ha due implicazioni molto forti. In primo luogo occorre considerare che il protocollo standard di accesso al canale per le reti wireless è il CSMA/CA. All'applicazione costruita viene chiesto di essere tempestiva nelle operazioni di acquisizione delle misure e nella spedizione dei pacchetti: il protocollo CSMA/CA, però, una volta che il nodo cattura i dati e

prepara il pacchetto per la spedizione, se il canale è occupato, attende che questo si liberi, non permettendo alla CPU del micro-processore di acquisire nuove misure, tenendola "appesa" alla trasmissione. Inoltre l'impiego di un unico processo per l'acquisizione e l'invio dei dati impone un limite massimo ai tempi di trasmissione pari al periodo di campionamento.

Indipendentemente dal tipo di accesso al canale utilizzato, occorre poi considerare che la precisione massima con cui i microprocessori conteggiano lo scorrere del tempo, pari ad un millisecondo, è limitante. Questo non offre infatti piena affidabilità sul fatto di quando un singolo nodo acquisisca e spedisca i propri pacchetti. Non fornisce inoltre adeguate garanzie sulla spedizione dei pacchetti, poiché è sufficiente che un nodo sfasi leggermente rispetto agli altri per compromettere la sua trasmissione.

6.2 Soluzioni adottate per la piattaforma W-TREMORS

Per assicurare che i nodi non sprechino tempo rimandendo inattivi nella fase di spedizione perchè trovano il canale occupato, si è implementato una sorta di TDMA (Time Division Multiple Access) [21]. Questo algoritmo di accesso al canale, implementato soprattutto in reti a lungo raggio, è stato costruito sui nodi a livello applicativo della struttura architetturale dei protocolli. Infatti non è possibile intervenire a livello MAC, dove sono soliti essere implementati i protocolli di accesso al canale, perchè sono già stati definiti dallo standard 802.15.4.
Il protocollo TDMA, visibile nella Figura 6.1, suddivide l'asse temporale in frame di durata fissa che, a sua volta, è frazionato in un numero fisso di intervalli più piccoli chiamati slot. Ogni slot occupa una precisa posizione all'interno del frame. Un nodo che vuole trasmettere può farlo solamente per un tempo pari ad uno slot ad intervalli pari alla durata di un frame. Tuttavia nel tempo che ha a disposizione per inviare i propri messaggi, la stazione può sfruttare tutta la banda.

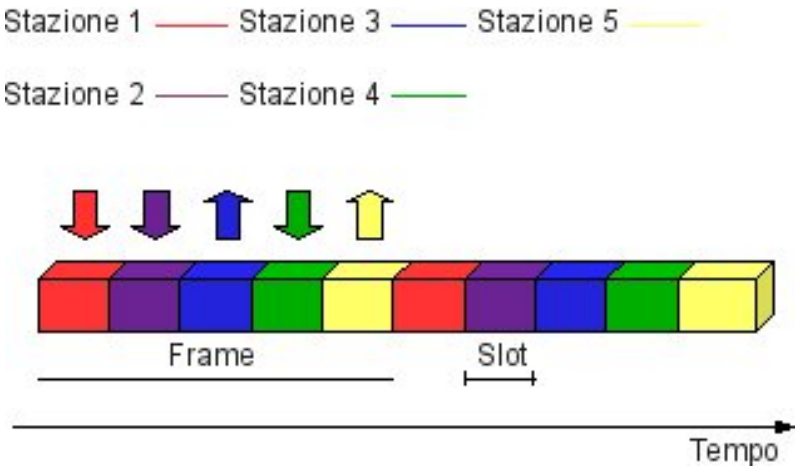


Figura 6.1: Funzionamento del protocollo TDMA.

Il TDMA risolve il problema dell'utilizzo del canale andando a stabilire, per ogni singolo nodo, il preciso slot di trasmissione: questa soluzione comporta due aspetti positivi: il nodo non trova mai il canale occupato e le collisioni tra i vari pacchetti

spediti dai nodi diminuiscono. Teoricamente il protocollo TDMA permette intrinsecamente di identificare il mittente esclusivamente sulla base dello slot temporale in cui il pacchetto è stato trasmesso. Tuttavia i vincoli, dettati precedentemente, sul fatto che i microprocessori non sono precisi al microsecondo e possono avere anche dei drift, portano ad una sfasatura degli slot temporali, provocando talvolta spostamenti nella trasmissione. Queste sfasature causano l'irriconoscibilità sull'appartenenza dei pacchetti. Infatti, se la trasmissione di un pacchetto da parte di un nodo supera lo slot designato per la sua trasmissione, questo va a collidere con lo slot temporale destinato al nodo successivo. In questa situazione, aiutati dal CSMA/CA, i due nodi potrebbero riuscire a risolvere positivamente il conflitto; in nessun caso però questo meccanismo fornisce garanzie circa la consegna, così come nemmeno sull'ordine di arrivo dei pacchetti. Per ovviare a questa incongruenza si è deciso di costruire una PDU a livello applicativo che contenesse al suo interno, oltre al time-stamp in cui ha eseguito la misurazione, anche l'identificativo di chi l'ha trasmessa. Con questa tecnica la piattaforma aumenta notevolmente la propria affidabilità. Il protocollo TDMA si preoccupa di stabilire gli slot di spedizione sfuttando tutta la banda a disposizione e diminuendo le collisioni e, quindi, la perdita dei pacchetti, mentre la PDU offre certezza sull'appartenenza delle misure. Infatti, anche se i nodi utilizzano un diverso slot di spedizione, ad esempio a causa di uno sfasamento, è possibile sempre ricondurre un pacchetto al mittente.

6.2.1 PDU single-data

Al fine di rispettare tutti i vincoli imposti dall'applicazione e, nel contempo, superare i problemi visti poco sopra, è necessario costruire una PDU di livello applicativo *ad-hoc*. La PDU più semplice in grado di consentire la trasmissione di tutte le informazioni necessarie, il riconoscimento del mittente e l'istante di cattura dei valori da parte dell'accelerometro, visibile in Figura 6.2, è formata dai seguenti campi:

- `START OF PACKET [0x40]`: indica l'inizio della PDU di dimensione un byte; corrisponde al carattere ASCII `@`.
- `ID [0x00 - 0xFF]`: indica il nominativo del sensore di dimensione un byte (solitamente con il valore 0 si indica il gateway).
- `TIME [0x0000 - 0xFFFF]`: composto da due byte che identificano la parte più significativa e parte meno significativa del *timestamp*.
- `MEASURE`: Misura catturata, dimensione pari a un byte.
- `END OF PDU [0x0A0D]`: formata da due byte che indicano la chiusura della PDU; corrisponono ai valori ASCII "\n" (*newline*) e "\r" (*carriage return*).

Ogni PDU, costruita a livello applicativo con all'interno solamente una singola misurazione, occupa in totale:

$$PDU_{single} = P_{header} + P_{misura} + P_{tailer} = 4 + 1 + 2 = 7byte$$

Questa soluzione tuttavia non è l'ideale perchè ha lo svantaggio che per trasferire una sola misurazione, la PDU utilizza ben 6 byte per sopperire alle problematiche viste precedentemente.

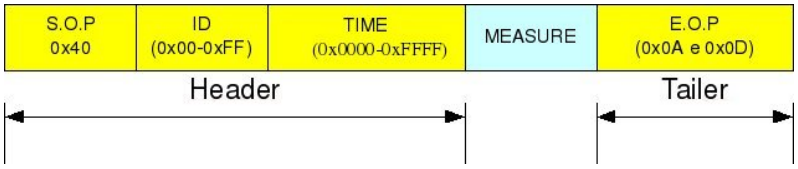


Figura 6.2: Tracciato PDU single-data.

A titolo di esempio, si supponga che un nodo debba trasferire tre letture. Il nodo dovrebbe trasmettere, in questo caso, tre pacchetti differenti contenenti ognuno la misura rilevata. La dimensione totale P_{tot} delle tre spedizioni risulterebbe pari a:

$$P_{tot} = 3 \cdot PDU_{single} = 21byte$$

Come è facile da intuire, per tre singole misure si ha una ridondanza di informazioni troppo elevata andando ad immettere sul canale trasmissivo per ben tre volte una header ed una tailer. Questa ridondanza provoca due limitazioni: la prima, una forte occupazione del canale rischiando collisioni; la seconda, l'impossibilità di costruire reti con un gran numero di nodi.

6.2.2 PDU variable-data

Questa forte limitazione sul numero massimo di nodi possibili per la costruzione di una rete, soprattutto nel caso in cui il dato richieda frequenze di campionamento elevate, ha portato a valutare la possibilità di effettuare trasmissioni con PDU di dimensione più ampia, ottimizzando il numero di spedizioni e contenendo più misure acquisite. Questa forma di PDU, chiamata burst, consente ad ogni nodo di effettuare meno trasmissioni. Con una PDU single-data, ad ogni valore catturato segue immediatamente una trasmissione. Il compito del burst, invece, è quello di eseguire più di una lettura e, successivamente, spedirle insieme in un unica trasmissione.

A questo punto la PDU della piattaforma W-TREMORS, visibile nel tracciato della Figura 6.3, è uguale a quello della PDU single-data con l'aggiunta di due campi:

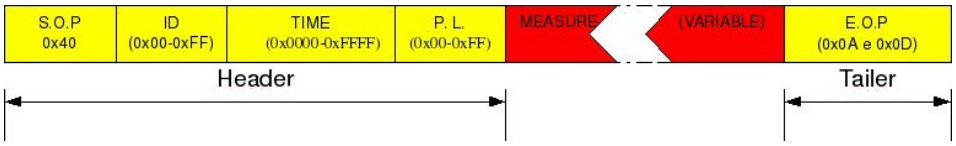


Figura 6.3: Datagram della PDU di W-TREMORS.

- `PAYLOAD LENGTH [0x00 - 0xFF]`: indica il numero di misure presenti nel pacchetto (burst).
- `MEASURE(S)`: sono i valori delle misure catturate dal sensore. Ogni misura è composta da un byte; la dimensione totale del campo varia a seconda del numero di misure trasmessi.

I primi cinque byte (HEADER) e gli ultimi due (TAILER) sono fissi per tutti i pacchetti spediti dai sensori. La dimensione totale della PDU dipende esclusivamente dalla quantità S di misurazioni presenti nella stessa:

$$PDU_{variable}(S) = P_{header} + (S \cdot P_{misura}) + P_{tailer} = 5 + (S \cdot 1) + 2 = (S + 7)byte$$

Confrontando con l'esempio visto precedentemente per il sigle-data, dove un nodo doveva trasmettere tre ($S = 3$) misurazioni, l'utilizzo di un burst consente di realizzare un'unica PDU variable-data di dimensione:

$$P_{tot} = 5 + (3 \cdot 1) + 2 = 10byte$$

Il risparmio può essere visto sia in termini di numero trasmissioni (due in meno) o di byte totali trasferiti (11 byte in meno). I vantaggi che si possono ottenere da questo tipo di soluzione sono due: tenere meno occupato il canale e, come verrà mostrato successivamente, costruire reti con una quantità maggiore di nodi.

6.2.3 Confronto tra single e variable data

In questa sezione vengono formalizzate analiticamente le equazioni utilizzate precedentemente per la scelta di come costruire le PDU a livello applicativo. Noti C_{canale} la capacità del canale (ovvero la velocità massima di trasmissione) e la frequenza di campionamento F_c richiesta ai sensori, il numero massimo di nodi N_{max} può essere facilmente calcolato. Si ricorda che la capacità massima del canale è fornita dalle specifiche di costruzione del modulo radio, mentre la campionatura con cui si vuole monitorare un evento dipende dalla scelta fatta dall'utente.

Nel caso della PDU single-data, descritta nella Sezione 6.2.1, è possibile sapere quanti nodi single-task possono essere impiegati in una rete TDMA-based. Infatti in un timeslot, avente durata $\frac{1}{F_c}$, tutti i nodi devono poter trasmettere la propria PDU contentente il valore acquisito:

$$T_{timeslot} = \frac{1}{F_c} \geq N \cdot T_{PDU_{single}} = N \cdot \frac{PDU_{single}}{C_{canale}}$$

In questo è possibile ricavare il numero N massimo di nodi che si possono avere nella rete:

$$N_{max} = \left\lfloor \frac{C_{canale}}{PDU_{single} \cdot F_c} \right\rfloor = \left\lfloor \frac{C_{canale}}{F_c \cdot 7byte} \right\rfloor$$

Nel caso invece della PDU variable-data, i limiti sul numero massimo di nodi impiegabili nella rete di accelerometri sono doppiamente legati alla frequenza con cui si vuole monitorare il fenomeno. Come detto in precedenza, ogni PDU contiene S misurazioni aventi dimensione P_{misura} pari ad $1byte$. Tuttavia, poiché i nodi sono mono-task, è necessario che la tramissione della PDU impieghi un tempo non superiore al periodo di campionamento; in caso contrario, il nodo non potrebbe riuscire ad acquisire con una frequenza adeguata. Pertanto è possibile scrivere:

$$\frac{1}{F_c} \geq T_{PDU(S)} = \frac{PDU_{variable}(S)}{C_{canale}} = \frac{S \cdot P_{misura} + 7byte}{C_{canale}}$$

Da ciò è possibile il numero massimo di campioni S raggruppabili in un burst:

$$S_{max} = \left\lfloor \frac{C_{canale}}{F_c} - 7 \right\rfloor$$

Ovviamente, nel caso in cui S valga 1, si ricade nel caso della PDU single data.

Inoltre occorre considerare che, effettuando trasmissioni a burst, ogni nodo effettua una trasmissione ogni S timeslot. Questo equivale a dire che ogni S timeslot, ovvero in ogni periodo di trasmissione di un burst, possono trasmettere N nodi in sequenza:

$$S \cdot \frac{C_{canale}}{T_{timeslot}} = \frac{C_{canale}}{\frac{F_c}{S}} \geq N \cdot PDU_{variable}(S)$$

Imponendo la dimensione del burst al massimo valore consentito S_{max}, è possibile riscrivere:

$$N \leq \frac{C_{canale}}{PDU_{variable}(S_{max}) \cdot \frac{F_c}{S_{max}}} = \ldots = S_{max}$$

Di conseguenza il numero massimo N_{max} di nodi della rete è esattamente uguale a S_{max}.

Quindi, dato il numero N di nodi nella rete, è possibile ricavare la frequenza massima possibile $F_{c_{max}}$, imponendo poi $S = N$ in fase di schedulazione:

$$F_{c_{max}} = \frac{C_{canale}}{(P_{header} + N + P_{tailer})} = \frac{14400}{7 + N} Hz$$

La massima velocità di trasmissione consentita dai moduli a RF impiegati in W-TREMORS è pari a $14400 \frac{byte}{s}$. Confrontando i risultati ottenuti dalle due PDU è possibile notare come l'utilizzo dei burst sia notevolmente migliore nel numero di nodi rispetto ad una PDU fissa (Figura 6.4).

Frequenza	N_{max} *senza burst*	N_{max} *con burst*
50	41	281
100	20	137
200	10	65
250	8	50
500	4	21
1000	2	7

Tabella 6.1: Numero massimo di nodi con e senza burst.

6.2.4 Trasmissione delle PDU

Riassumendo, le scelte fatte per la schedulazione della piattaforma W-TREMORS sono: 1) implementazione a livello applicativo del protocollo di accesso al canale TDMA allo scopo di assegnare slot di tempo ad ogni singolo nodo per la trasmissione diminuendo le collisioni 2) utilizzo di burst con PDU-variable, che offrono la possibilità di avere reti con maggior quantità di nodi ottimizzando l'accesso al canale. Il tracciato della PDU permette altresì di sopperire eventuali problematiche

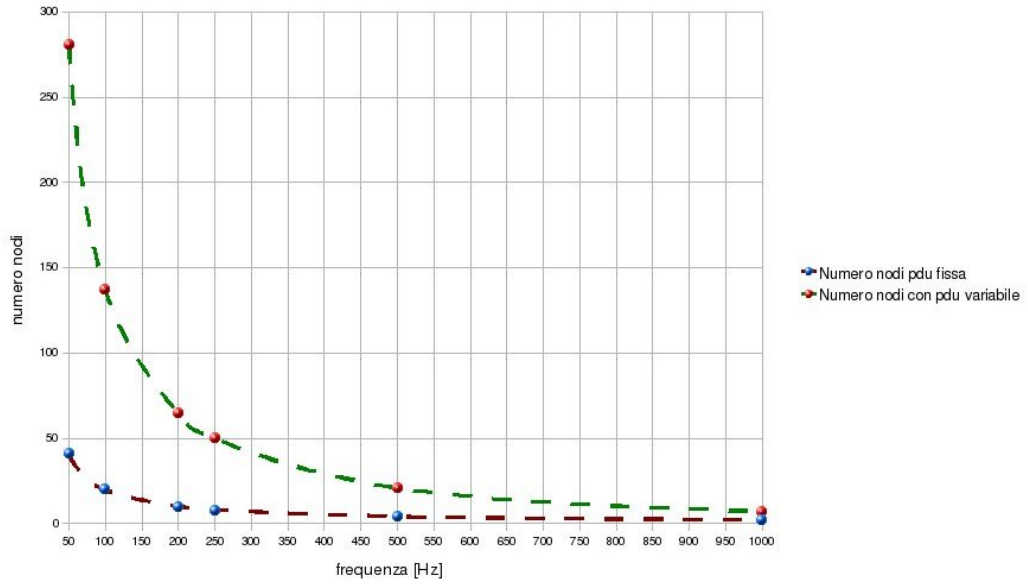


Figura 6.4: Numero di nodi con e senza l'ausilio dei burst.

riguardo l'appartenenza di misure effettuate da un nodo risolvendo l'imprecisione del timestamp di ogni micro-processore. Il TDMA, per assegnare gli slot temporali ai vari nodi, esegue algebricamente una sfasatura iniziale che viene chiamata "offset dello scheduler". Per esempio, avendo una rete formata da tre nodi, come in Figura 6.5, l'offset del TDMA setta ad ogni nodo il numero di letture da eseguire all'inizio precisando così lo slot di tempo destinato alla trasmissione.

Sia l'offset che il burst sono calcolati attraverso un algoritmo software il quale calcola ed assegna dinamicamente i valori da utilizzare ad ogni nodo facente parte della rete.
Al burst viene sempre dato il valore di numero totale di nodi presenti ed utilizzati nella rete ed è uguale per tutti. L'offset, invece, assegna, specificando ad ogni singolo nodo, lo slot di tempo che dovrà utilizzare per la spedizione. Ciò che si può notare anche nella Figura 6.5 è che l'ultimo nodo a cui si assegna la schedulazione l'offset risulta uguale al burst, cioè parte già a regime.

Ricapitolando, i nodi eseguiranno la prima spedizione con una PDU pari a quella indicata dall'offset mentre le successive, fino al termine, con quelle indicata dal burst. Nella Sezione C.4.1 è possibile vedere il diagramma di flusso dell'algoritmo implementato nella piattaforma.

6.3 Problemi ancora aperti

Le prove effettuate per validare la teoria hanno comportato, soprattutto a campionature elevate, problemi in trasmissione [65; 58]. La causa risiede nello slot di tempo destinato alla trasmissione calcolato con il TDMA, il quale risulta essere, in alcuni casi, troppo breve. Infatti la funzione *millis()* [24; 18], utilizzata all'interno dei nodi

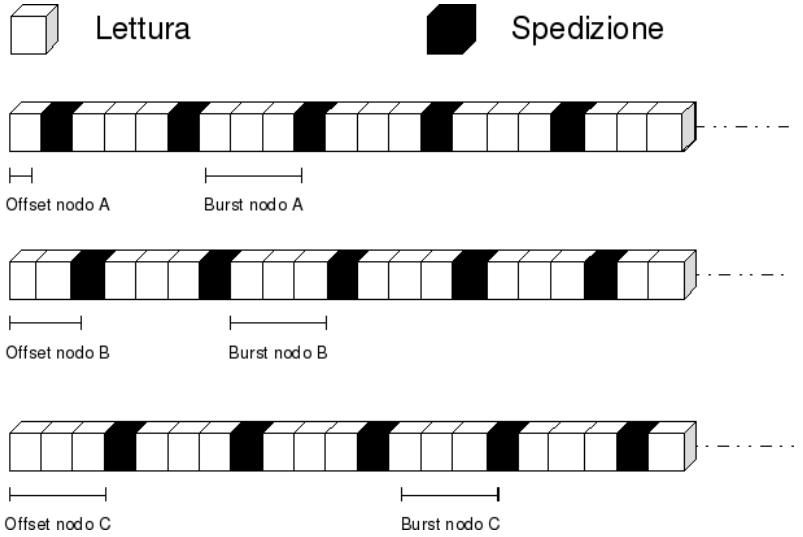


Figura 6.5: Esempio di funzionamento dell'offset.

per gestire la temporizzazione, è piuttosto imprecisa: la risoluzione massima è pari a un millisecondo e l'errore, nel caso peggiore, può arrivare a $\pm 1ms$. Questo può comportare un disallineamento della sincronizzazione tra i vari nodi, provocando collisioni tra i pacchetti e conseguente la perdita di alcuni dati. La perdita dei dati è definitiva poiché il modulo radio è stato utilizzato nella configurazione standard del protocollo 802.15.4 senza l'ausilio degli ACK. Infatti, utilizzare il protocollo 802.15.4 [30] confermato provoca un abuso del canale dando luogo a una maggiore perdita di pacchetti oppure una sovra-abbondanza di pacchetti duplicati (i settaggi del modulo radio XBee possono essere visti nell'Appendice B).
Per ovviare a questo problema si è pensato di aggiungere una banda di guardia (BdG) tra una trasmissione e la successiva. La banda di guardia fornisce uno slot di tempo maggiore per la trasmissione. Per aver una maggior affidabilità del sistema, questa aggiunta riduce il numero massimo possibile di nodi. Infatti, se si vuole inserire una banda di guardia di uno slot è necessario moltiplicare per $\frac{1}{2}$ il numero massimo di nodi. Se invece si vogliono mettere due slot tra una trasmissione e l'altra, è necessario moltiplicare il numero massimo di nodi per $\frac{1}{3}$, come mostrato nel grafico della Figura 6.7. Tuttavia, si può vedere che, anche diminuendo il numero dei nodi massimo per la rete, questo valore resta, grazie all'utilizzo di burst, più alto che nel caso di PDU single-data.

Un altro possibile problema sono le prestazioni al variare delle frequenze di campionamento. A questo proposito sono state eseguite delle prove sull'affidabilità del sistema con una rete formata da due nodi facendo variare la frequenza di campionamento. La verifica è stata fatta contando il numero totale di pacchetti giunti a destinazione in percentuale. Come risulta evidente dal grafico nella Figura 6.6, senza banda di guardia a velocità di campionamento elevate la perdita di pacchetti risulta pesante, andando a compromettere la ricostruzione del segnale acquisito.

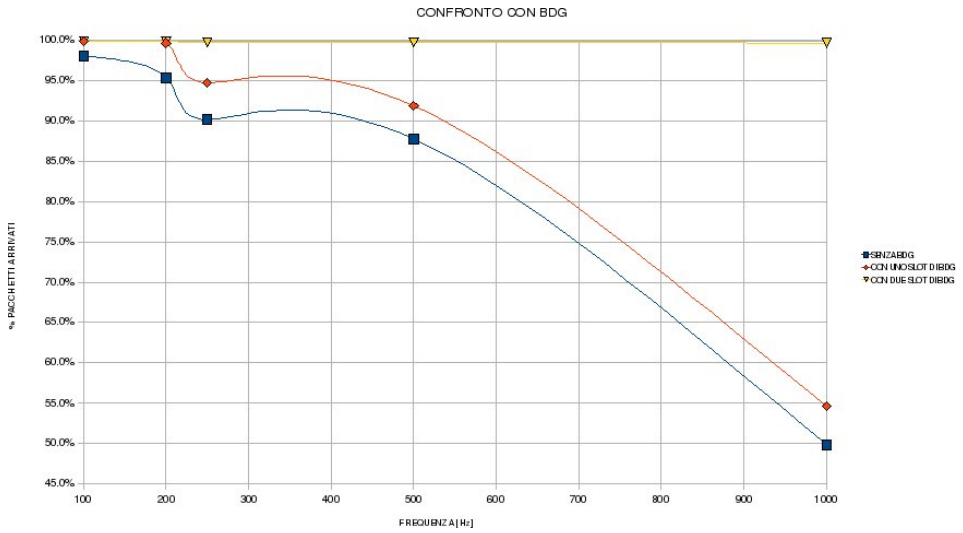


Figura 6.6: Tasso di perdita dei paccheti con e senza BdG.

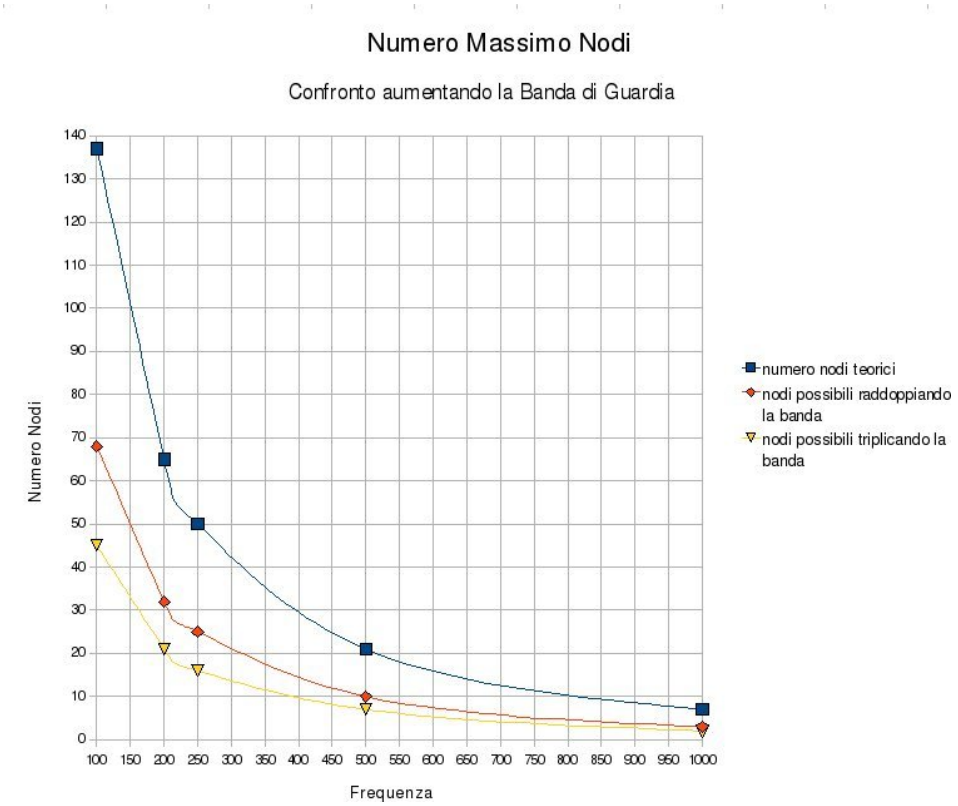


Figura 6.7: Variazione del numero massimo di nodi aumentando la BdG.

Frequenza Hz	*Numero nodi massimo*	*Numero nodi raddoppiando la banda*	*Numero nodi triplicando la banda*
1	14393	7196	4797
5	2873	1436	957
10	1433	716	477
20	713	356	237
25	569	284	189
50	281	140	93
100	137	68	45
200	65	32	21
250	50	25	16
500	21	10	7
1000	7	3	2

Tabella 6.2: Numero massimo di nodi con e senza BdG.

Frequenza Hz	*Senza BDG*	*Con 1 slot di BDG*	*Con 2 slot di BDG*
100	98.1%	99.9%	99.9%
200	95.4%	99.7%	99.9%
250	90.2%	94.8%	99.8%
500	87.8%	91.9%	99.8%
1000	49.8%	54,6%	99.7%

Tabella 6.3: Tasso di perdita dei pacchetti con e senza BdG.

6.4 Conclusioni

La capacità del canale, in una wireless sensor network basata su IEEE 802.15.4, è necessariamente limitata; questo implica che si deve cercare di ottimizzare il trasferimento dei dati da parte di un nodo senza che la sua trasmissione interferisca sulle altre oppure che monopolizzi l'intero canale.
Le imprecisioni intrinseche nella piattaforma SquidBee riducono ulteriormente l'effettiva banda disponibile per le trasmissioni. L'introduzione di una trasmissione a burst permette, almeno in parte, di sopperire a questa forte limitazione.
Ulteriori indagini sulle cause dell'imprecisione, presente nell'implementazione della funzione *millis()*, potrebbero sicuramente migliorare le prestazioni complessive del sistema.

Capitolo 7

Prove pratiche

Adesso credo, guardandomi indietro, che non combattevamo il nemico ma noi stessi, ed il nemico era in noi.

dal film *Platoon (Platoon, 1986)*

Alla realizzazione dell'intera piattaforma WSN sono seguiti dei test. I test sono serviti per evidenziare la bontà dei dati catturati dai sensori e per verificare la corretta funzionalità della struttura. I primi test si sono svolti in EUCENTRE a Pavia, mentre i successivi nel Laboratorio Telecomunicazioni di Mantova.

7.1 Test in EUCENTRE

Il giorno venerdì 19 Settembre 2008 in EUCENTRE è stato effettuato il primo test. Sono state fornite tre aste in ferro fissate a terra con un piedistallo. Una di queste è visibile nella Figura 7.1. Di ognuna delle tre aste è stato fornito il periodo (T) e la frequenza (f).

Asta	*Frequenza f [Hz]*	*Periodo T [s]*
1	1.62	0.61
2	2.06	0.48
3	4.50	0.22

Tabella 7.1: Tabella frequenze e periodo delle aste.

La rete impostata era costituita da un nodo-sensore e del gateway. La finalità principe è stata quella di catturare i valori forniti dall'accelerometro, ricostruire il segnale e comparare quest'ultimo con i valori di frequenza e periodo forniti.
Il campionamento con cui sono state eseguite le prove per la cattura dell'oscillazione è stato di 10 Hz per tutte le aste. Per l'asta a 2.06 Hz sono state provati anche campionamenti a 5, 20, 25 e 50 Hz. I valori delle prove sono stati inseriti automaticamente in un file. Con l'ausilio di un semplice script in Matlab [20], partendo dal file come input, si è ricostruito il segnale, fornendo in uscita il plottaggio dello stesso.
Vengono mostrati in figura i segnali raccolti nel dominio del tempo dal sensore

Figura 7.1: Asta da 2.06 Hz.

posizionato sulle tre aste con frequenza di oscillazione 1.62 Hz (Figura 7.2a), 2.06 Hz (Figura 7.2c) e 4.50 Hz (Figura 7.2e).

Già da questi tre grafici è possibile notare la correttezza delle misurazioni attraverso il numero delle oscillazioni per periodo. Applicando poi al segnale la trasformata di Fourier, la frequenza di oscillazione dell'asta può essere ottenuta come il picco della trasformata. I valori sono in Tabella 7.2.

Asta	*Frequenza misurata [Hz]*	*Frequenza reale [Hz]*
1	1.69	1.62
2	2.08	2.06
3	4.52	4.50

Tabella 7.2: Confronto tra le frequenze trovate e date.

A questo punto si è scelta una delle tre aste ed è stata provata la piattaforma aumentando la frequenza di campionamento. L'asta scelta è stata la 2.06 Hz. Di seguito viene illustrato nella tabella i valori trovati applicando la trasformata di Fourier ai segnali campionati con l'accelerometro.

Come è possibile notare, anche aumentando il campionamento, l'imprecisione rispetto alla frequenza di riferimento iniziale risulta essere 1/100. Questo errore, tuttavia, può essere causato dalla somma di tanti piccoli errori dovuti all'acquisizione attraverso componenti hardware e software della piattaforma.
Si noti che la campionatura a 5 Hz denota un errore maggiore; questo perché, come noto dalla teoria, al di sotto dei 5 Hz ci si avvicina al limite teorico del campionamento e si rischia di sottocampionare.
I risultati ottenuti sono stati molto soddisfacenti sia per quanto riguarda il funzionamento della piattaforma sia per l'affidabilità dei dati acquisiti dal nodo-sensore.

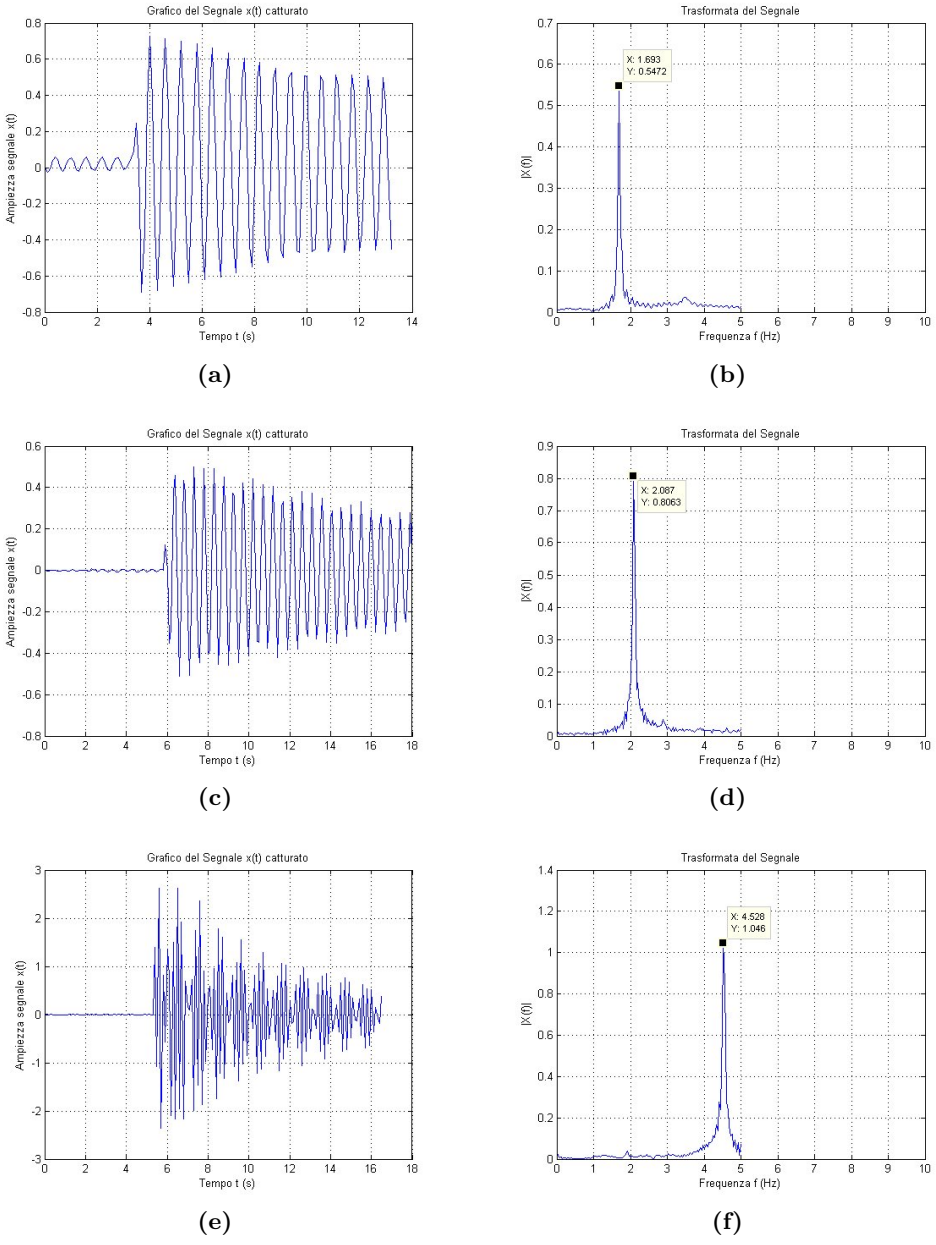


Figura 7.2: Segnale campionato nel tempo e analisi in frequenza con l'asta oscillante rispettivamente a 1.62 Hz ((a), (b)), 2.06 Hz ((c), (d)) e 4.50 Hz ((e), (f)).

Frequenza di Campionamento	*Frequenza principale trovata*
5	2.14
10	2.08
20	2.07
25	2.07
50	2.07

Tabella 7.3: Valori delle frequenze catturate trovate a diverse frequenze di campionamento.

L'unico problema riscontrato è stato l'elevato numero di errori in ricezione provando a campionare ad alte frequenze. Questo problema, si è scoperto successivamente, era dovuto ad un errore di implementazione del firmware della scheda radio. Infatti, una volta aggiornato il firmware dell'XBee alla versione 10A5, problema non si è più evidenziato.
E' possibile che qualche pacchetto con i dati nella prove effettuate possa essere andato perso, tuttavia nell'implementazione della piattaforma i valori persi vengono ricostruiti per interpolazione lineare.

7.2 Test con due nodi

Successivamente, sono state eseguite delle prove costruendo una rete con due nodi. Queste prove sono state necessarie per validare la sincronizzazione degli accelerometri. Il primo test è stato effettuato campionando a 100 Hz e posizionando i due accelerometri uno sotto l'altro su un'asta identica a quella utilizzata nel primo test di EUCENTRE (Figura 7.3). L'asta a disposizione ha un periodo noto di circa 0,27 secondi, che corrispondono alla frequenza di 3.7 Hz. La durata del test è stata pressapoco di 12 secondi.

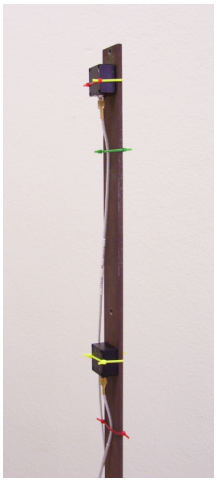
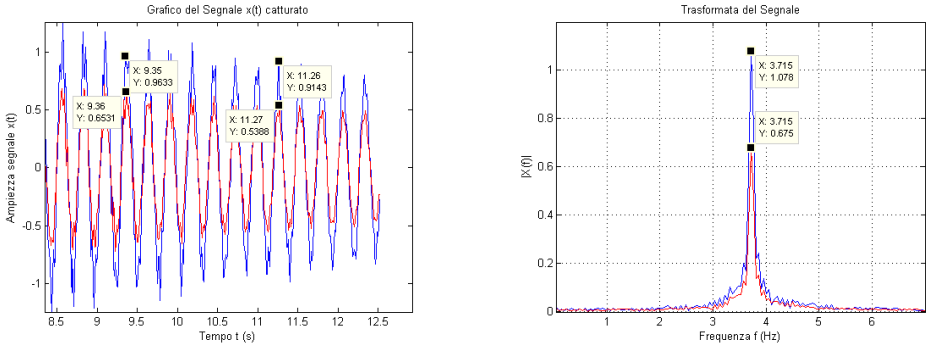


Figura 7.3: Asta del laboratorio TLC.

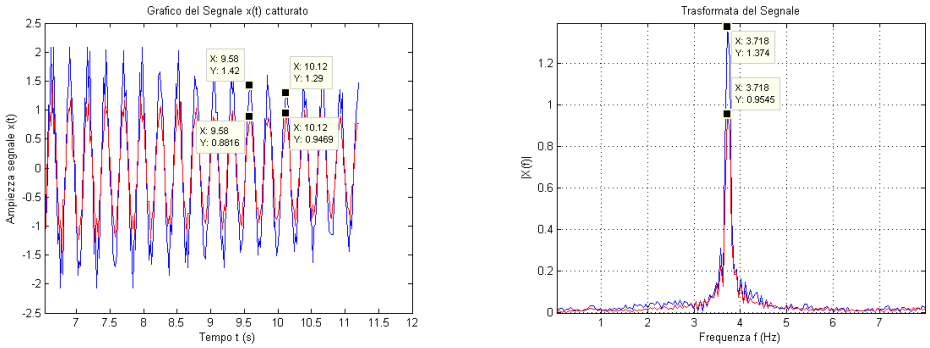
Come è possibile notare dai grafici i segnali sono fra loro sincronizzati. Ovviamente



(a) **(b)**

Figura 7.4: Acquisizione sincrona di due nodi con frequenza di campionamento pari a 100 Hz.

il nodo posizionato più in basso sull'asta ha un'ampiezza minore, ma entrambi seguono l'oscillazione dell'asta in modo sincrono. Nel dominio del tempo si può osservare (Figura 7.4a) come, nel momento di massima ampiezza di un'onda del segnale, entrambi i nodi si trovino nello stesso istante di tempo con un errore pari a 0.01 secondi, dimostrando un'alta sincronizzazione.
Passando invece nel dominio delle frequenze, si può osservare che entrambi i nodi rilevano la stessa frequenza di oscillazione, come visibile nel grafico della Figura 7.4b.



(a) **(b)**

Figura 7.5: Acquisizione sincrona di due nodi con frequenza di campionamento pari a 100 Hz.

Si è provato ad abbassare la frequenza di campionamento per verificare che i nodi riuscissero comunque a ricostruire il segnale. Anche in questo caso il campionamento è risultato preciso e sincrono, come visibile in Figura 7.5a e in Figura 7.5b.

Capitolo 8

Conclusioni

25 maggio 1877. Questa sarà l'ultima pagina del mio diario. Ho cercato di dare un resoconto fedele di ciò che ho visto e che ho fatto. Non ho la presunzione di comprendere il corso della mia vita, so solo che sono grato di aver partecipato a tutto questo anche se solo per un momento..

dal film *L'Ultimo Samurai (The Last Samurai, 2003)*

Questa tesi ha voluto aprire nuovi orizzonti sulle altre applicazioni possibili dell'abbinata Arduino-XBee, originariamente non progettato per lo Structural Health Monitoring. In questo capitolo verranno descritti i vantaggi che offre la piattaforma W-TREMORS e i limiti hardware dei componenti utilizzati; infine verranno delineate alcune possibili sviluppi futuri.

Come si è potuto vedere dallo studio effettuato per l'implementazione dell'intera piattaforma, si è dovuto talvolta cercare di ottimizzare al meglio sia il codice che gli algoritmi di implementazione: questo è stato necessario per risparmiare memoria sull'AVR, non costringendo il micro-processore ad eseguire un codice prolisso, oltre che per limitare la quantità di dati trasmessi. Infatti 802.15.4 è uno standard "low-data rate" e, quindi, non offre alte velocità di trasmissione. Come si è potuto notare dalle prove pratiche, ad alte velocità di campionamento la struttura non risulta essere molto efficiente, soprattutto sulla quantità di nodi associabili alla rete. Per ovviare a questo limite sarebbe pensabile una re-ingegnerizzazione dell'hardware con sviluppi alternativi usufruendo, ad esempio, di moduli radio ad alto data-rate basati su Ultra Wide Band (UWB).

Il tempo, nell'applicazione, è contato attraverso una variabile intera ($2^{16} = 65536$); questo implica una durata massima di circa 65 secondi per ogni prova, prima che il contatore si azzeri. Per avere tempistiche più lunghe, una soluzione potrebbe consistere nell'utilizzare un contatore di giri che segnali il traboccamento ogni qualvolta il tempo superi il valore massimo. Tuttavia, modificare l'intera applicazione, passando da una variabile intera a una variabile di tipo long ($2^{32} = 4294967296$), verrebbe a costare sia dal lato acquisizione sia, soprattutto, dal lato sensore. Quest'ultimo dovrebbe infatti sempre verificare la condizione di superamento, rallentando così l'applicazione stessa, e comportando, una volta ogni 65 secondi, un byte in più nella spedizione del pacchetto. E questo introdurrebbe ulteriori

problemi: come si è potuto notare da alcune prove eseguite, spedire un carattere o far spedire una stringa influenza il tempo di processing dell'AVR e introduce ritardi non facilmente determinabili a priori.

Un tema legato alla durata degli esperimenti è il reset. In questo momento, una volta eseguita una prova, per resettare la rete serve premere un pulsante posizionato sul nodo stesso. E' possibile intervenire via software, tuttavia quando l'applicazione parte e viene interrotta attraverso un segnale, ciò che si viene a bloccare è il gateway non i nodi-sensori, i quali continuano ad inviare i propri dati. Le soluzioni dovrebbero stare nel dare un limite massimo alla prova oppure nell'implementare un segnale apposito, che in broadcasting riavii tutti i nodi-sensori. Purtroppo, resettare un nodo non è così banale come può apparire poiché si deve gestire la problematica degli interrupt ed implementare tutte le varie eccezioni; inoltre sarebbe necessario individuare un messaggio apposito, non ambiguo e chiaramente distinguibile dagli altri dati.

Un'altro punto interessante dell'applicazione è l'utilizzo di un database, che ha lo scopo di costruire un file di output coerente con quello richiesto da EUCENTRE. Questo database potrebbe essere ulteriormente ampliato, usufruendo di tutte le potenzialità offerte dai DBMS relazionali come, ad esempio, l'archiviazione di tutti gli esperimenti eseguiti oppure il confronto fra risposte di stessi sensori su esperimenti diversi.

Nello stile dell'Open Source e per perseguirne le finalità, l'intera struttura è stata implementata e costruita in modo tale che sia lasciata libera scelta a qualsiasi programmatore di poter modificare anche solamente alcune parti della piattaforma in base alle esigenze. I sorgenti software, implementati in C e visibili nell'Appendice C, possono essere facilmente adattati per qualsiasi altro tipo di hardware mentre la parte teorica rimane valida per qualsiasi modulo radio low data-rate vista nel Capitolo 6.

Appendice A

Le primitive termios

Le termios sono funzioni che permettono di gestire le caratteristiche di un terminale tramite una porta seriale. Viene proposto in questa appendice il funzionamento di tali funzioni che permettono di colloquiare con un dispositivo esterno. Le termios sono una struttura in c dove è possibile identificare le proprietà di un terminale. Le caratteristiche sono divise in diversi campi:

- `tcflag_t c_iflag`
- `tcflag_t c_oflag`
- `tcflag_t c_cflag`
- `tcflag_t c_lflag`

Questi primi quattro sono flag che controllano il comportamento del terminale.

- `cc_t c_cc[NCCS]`
- `cc_t c_line`

I c_cc flag vengono usati per impostare i caratteri speciali associati alle varie funzioni di controllo.

- `speed_t c_ispeed`
- `speed_t c_ospeed`

I due flag settano le velocità di input ed output della seriale.

A.1 I Parametri IFLAG

- `INPCK`: abilita il controllo di parità in input.
- `IGNPAR`: ignora gli errori di parità.
- `PARMKR`: controlla controlla come vengono riportati gli errori di parità.

- `ISTRIP`: se impostato i caratteri di input vengono tagliati a 7 bit mettendo a 0 il più significativo altrimenti
  vengono passati tutti e 8.
- `IGNBRK`: ignora condizioni di BREAK in input.
- `BRKINT`: controlla la reazione ad un BREAK.
- `IGNCR`: se impostato il carattere di RETURN CARRELL viene scartato dall'input.
- `ICRNL`: se impostato un char "\r" viene trasformato in "\n" sulla coda dell'input.
- `ICLCR`: se impostato un char "\n" viene trasformato in "\r" sulla coda dell'input.
- `IUCLC`: se impostato trasforma i caratteri maiuscoli in minuscoli.
- `IXON`: attiva il controllo di flusso in uscita con char di stop e start.
- `IXANY`: se attivato con il controllo di flusso permette a qualunque char di far ripartire l'output bloccato da un char di stop.
- `IXOFF`: se impostato abilita il controllo di flusso in ingresso.
- `IMAXBEL`: allarme sonoro se si riempie la coda in ingresso.

A.2 I Parametri OFLAG

- `OPOST`: se impostato i char vengono convertiti per la visualizzazione sul terminale.
- `OCRNL`: se impostato converte il carattere "\n" nella coppia "\r""\n".
- `OLCUC`: se impostato trasforma char minuscoli in ingresso in char maiuscoli all'uscita.
- `ONLRC`: se impostato converte automaticamente il char "\n" in "\r".
- `ONOCR`: se impostato converte il carattere "\r" nella coppia "\r""\n".
- `ONLRET`: rimuove dall'output il carattere "\r".
- `OFILL`: se impostato in caso di ritardo sulla linea invia dei caratteri di riempimento invece di attendere.
- `OFDEL`: se impostato il carattere di riempimento è DEL (0x3F) invece di NULL (0x00).
- `NLDLY`: maschera per i bit che indicano il ritardo per il carattere "\n".
- `CRDLY`: maschera per i bit che indicano il ritardo per il carattere "\r".

- `TABDLY`: maschera per i bit che indicano il ritardo per il carattere TAB
- `BSDLY`: maschera per i bit che indicano il ritardo per il carattere BACKSPACE
- `VTDLY`: maschera per i bit che indicano il ritardo per il carattere TAB verticale
- `FFDLY`: maschera per i bit che indicano il ritardo per il carattere NEW PAGE

A.3 I Parametri CFLAG

- `CLOCAL`: se impostato indica che il terminale è connesso in locale.
- `HUPCL`: se impostato viene staccata la connessione con il modem.
- `CREAD`: se impostato si può leggere l'input del terminale.
- `CSTOPB`: se impostato vengono usati due bit di stop altrimenti uno.
- `PARENB`: se impostato abilita la generazione del controllo sul bit di parità.
- `PARODD`: controllo con il bit di parità associato con PARENB.
- `CSIZE`: maschera per i bit usati per specificare la dimensione del char.
- `CRTSCTS`: Abilita il controllo di flusso hardware sulla seriale.

A.4 I Parametri LFLAG

- `ICANON`: se impostato il terminale opera in modo canonico.
- `ECHO`: se impostato viene attivato echo dei caratteri di input sull'output del terminale.
- `ECHOE`: se impostato mostra la cancellazione di un char di input cancellando l'ultimo carattere della riga corrente.
- `ECHOPRT`: se impostato abilita la visualizzazione del carattere cancellato con uscita su stampante.
- `ECHOK`: se impostato abilita il trattamento della visualizzazione del carattere KILL
- `ECHOKE`: se impostato abilita il trattamento della visualizzazione del carattere KILL cancellando i caratteri precedenti nella linea.
- `ECHONL`: se impostato viene effettuato echo di un "\r".
- `ECHOCTL`: se impostato insieme ad ECHO i caratteri di controllo ASCII (tranne TAB, NL, START, e STOP) sono mostrati nella forma che prepende un ^ alla lettera ottenuta sommando 0x40 al valore del carattere.
- `ISIG`: se impostato abilita il riconoscimento dei char INTR QUIT SUSP.

- `NOFLSH`: se impostato disabilita lo scarico delle code in ingresso e in uscita quando vengono emessi i segnali SIGINT, SIGQUIT,SIGSUSP.
- `TOSTOP`: se impostato genera il segnale SIGTTOU per un processo in background.
- `XCASE`: se impostato il terminale funziona solo con le maiuscole.
- `DEFECHO`: se impostato effettua l'echo solo se c'è un processo in lettura.

A.5 I Parametri Speciali

- `VINTR`: carattere di interrupt provoca l'emissione di SIGINT.
- `VQUIT`: carattere di uscita provoca l'emissione di SIGQUIT.
- `VERASE`: cancella l'ultimo carattere precedente nella linea.
- `VKILL`: cancella l'intera riga.
- `VEOF`: carattere di END OF FILE.
- `VTIME`: timeout in dec/secondo per la lettura in modo non-canonico.
- `VMIN`: numero minimo di caratteri per una lettura in modo non-canonico.
- `VSTART`: carattere di start per riavviare l'output.
- `VSTOP`: carattere di stop che blocca l'output fino a start.
- `VSUSP`: carattere di sospensione.
- `VEOL`: carattere di fine riga.
- `VREPRINT`: stampa dei caratteri non ancora letti.
- `VWERASE`: cancella una parola.

A.6 Abilitare e disabilitare i FLAG

- &= disabilito i flag. Esempio : opciones.c_cflag &= (PARENB | PARODD);//DISABILITO
- = imposto il valore. Esempio: opciones.c_oflag = 0; //DISABILITO TUTTO
- |= setto e abilito i flag. Esempio: opciones.c_cflag |= CS8; //IMPOSTO 8 BIT

A.7 Le Funzioni

Le funzioni delle termios sono:

- `int tcsetattr(int fildes, int optional_actions, const struct termios *termios_p);` setta le nuove impostazioni della seriale.
- `int tcgetattr(int fildes, struct termios *termios_p);` legge i valori delle impostazioni attuali della seriale.
- `speed_t cfgetispeed(const struct termios *termios_p);` legge la velocità in ingresso della seriale
- `speed_t cfgetospeed(const struct termios *termios_p);` legge la velocità in uscita della seriale.
- `int cfsetispeed(struct termios *termios_p, speed_t speed);` imposta la velocità in ingresso della seriale.
- `int cfsetospeed(struct termios *termios_p, speed_t speed);` imposta la velocità in uscita della seriale.

  L'unità di misura delle costanti, che specificano la velocità della seriale, sono in bit/s:

Velocità della seriale			
B0	B50	B75	B110
B134	B150	B200	B300
B600	B1200	B1800	B2400
B4800	B9600	B19200	B38400
B57600	B115200	B230400	B460800

Tabella A.1: Tabella velocità possibili della seriale.

  Con bit-rate 0,B0, si identifica la terminazione di una connessione.

- `int tcdrain(int fildes);` attende lo svuotamento della coda di output. La funzione blocca il processo fine a che tutto l'output presente sulla coda d'uscita non è stato trasmesso.
- `int tcflow(int fildes, int action);` sospende e riavvia il flusso dei dati sul terminale.

  Possibili action:

- `int tcflush(int fildes, int queue_selector);` svuota immediatamente le code cancellando tutti i dati presenti.

  Possibili queue_selector:

	Action di Flusso
TCOOFF	sospende l'output
TCOON	riavvia l'output
TCIOFF	sospende l'input.
TCION	riavvia l'input

Tabella A.2: Tabella action della seriale.

	Action su code I/O
TCIFLUSH	svuota la coda in ingresso.
TCOFLUSH	svuota la coda in uscita.
TCIOFLUSH	svuota entrambe le code.

Tabella A.3: Tabella queue della seriale.

A.8 La modalità canonica e non-canonica

Operare con un terminale in modo canonico è relativamente semplice. I caratteri d'ingresso restano in coda fin tanto che non viene ricevuto un "\n". La dimensione massima definita per il buffer di input è data attraverso il parametro MAX_INPUT mentre con MAX_CANON si designa la dimensione massima di una riga. La gestione dell'input nella forma canonica tuttavia è di norma eseguita direttamente dal driver del terminale, che si incarica di cancellare i caratteri, bloccare e riavviare il flusso dei dati, terminare la linea quando viene ricevuti uno dei vari caratteri di terminazione. Operare nella forma non-canonica bisogna gestire tutte le eccezioni. Esistono però due specifiche principali VMIN e VTIME in c_cc che informano il sistema di ritornare da una read quando è stata letta una determinata quantità di dati o quando è passato un certo tempo. Il comportamento del sistema con i due parametri è suddiviso in quattro casi:

- `MIN > 0, TIME > 0` In questo caso MIN stabilisce il numero minimo di caratteri desiderati e TIME un tempo di attesa, in decimi di secondo, fra un carattere e l'altro. Una read ritorna se vengono ricevuti almeno MIN caratteri prima della scadenza di TIME (MIN è solo un limite inferiore, se la funzione ha richiesto un numero maggiore di caratteri ne possono essere restituiti di più); se invece TIME scade vengono restituiti i byte ricevuti fino ad allora (un carattere viene sempre letto, dato che il timer inizia a scorrere solo dopo la ricezione del primo carattere).

- `MIN > 0, TIME = 0` Una read ritorna solo dopo che sono stati ricevuti almeno MIN caratteri. Questo significa che una read può bloccarsi indefinitamente.

- `MIN = 0, TIME > 0` In questo caso TIME indica un tempo di attesa dalla chiamata di read, la funzione ritorna non appena viene ricevuto un carattere o scade il tempo. Si noti che è possibile che read ritorni con un valore nullo.

- `MIN = 0, TIME = 0` In questo caso una read ritorna immediatamente resti-

tuendo tutti i caratteri ricevuti. Anche in questo caso può ritornare con un valore nullo.

Appendice B

Il settagio del modulo radio XBee

Il modulo radio XBee [53] possiede una serie di comandi con cui è possibile effettuare regolazioni. Dagli studi e dalle prove effettuate si è trovata una configurazione finale ottima all'applicazione. In questa appendice verranno mostrati quali comandi sono stati utilizzati e il loro significato. I comandi possibili sono all'incirca, in totale, una settantina divisi in gruppi:

- Speciali: resetta, scrive, ripristina i valori settati al modulo con i vari comandi.
- Rete e sicurezza:offrono la possibilità di assegnare indirizzi ai moduli, di eseguire tabelle d'instradamento e di poter criptare i pacchetti attraverso l'algoritmo AES.
- Interfaccia: offre la possibilità di regolare la potenza dei moduli radio.
- Basso consumo: serie di comandi per il risparmio di energia (fase di sleep).
- Seriale in I/O: opzioni per la seriale come settare la velocità e tempo di pacchettizzazione.
- Diagnostica: versione del firmware, hardware, energia.
- Opzioni: entrata/uscita nel modulo radio per il settaggio dei comandi.

Per parlare con il modulo radio si necessita di terminale di seriale (gtkterm in linux). Una volta aperto si digita +++ il modulo radio risponde con OK. A questo punto è possibile effettuare i vari settaggi. Tutti i comandi sono preceduti dalle lettere AT seguite dal comando e dal parametro. Ad esempio, il comando ATMY 3 assegna al modulo radio l'indirizzo con valore 3.

Alla fine bisogna confermare tutte le modifiche attraverso il comando ATWR.

I moduli radio per l'applicazione W-TREMORS sono stati regolati con i seguenti parametri:

- `ATBD`: setta il data-rate della seriale può avere valori tra 1 a 7. Ad ogni numero corrisponde una velocità. L'opzione scelta è la 7 a cui corrisponde la velocità di 115200b/s per tutti i nodi.

- `ATMY`: setta il proprio indirizzo. I valori dati ai moduli radio per l'applicazione sono contenuti nel range 0-256. Questo si è fatto per contenere il consumo di byte nella trasmissione. Il valore 0 è stato assegnato al gateway mentre gli altri possono essere destinati ai nodi che compongono la rete.

- `ATDH`: indirizzo di destinazione (parte alta) a cui si destinano i pacchetti. In W-TREMORS è settato a 0 per tutti i componenti della rete.

- `ATDL`: la destinazione (parte bassa), vista la tipologia, è stata settata con due indirizzi diversi uno riferito al gateway e uno riferito ad ogni nodo. Il gateway è settato con 0xFFFF che indica il broadcasting. Infatti le impostazioni iniziali devono essere fornite a tutti i nodi. Per quanto riguarda i nodi hanno l'indirizzo per destinare i propri pacchetti con 0x00 (indirizzo del gateway).

- `ATRN`: abilita e disabilita la possibilità di usare il protocollo CSMA-CA. I possibili valori sono tra 0 e 3. impostando il valore 0 si disabilita il protocollo mentre con gli altri valori è possibile abilitare il CSMA-CA restando all'ascolto del canale. Se il canale è libero il pacchetto viene spedito, se il canale è occupato il modulo attende per un tempo random, poi verifica successivamente se il canale è libero. Dopo il tempo di attesa per la riprova di trasmissione il processo di ascolto termina e i dati vengono persi. Il numero delle volte con cui il protocollo ascolta il canale si riferisce al valore impostato. Per l'applicazione è stata impostato il valore 1.

- `ATRO`: questo comando attende un tempo-carattere prima di effettuare la trasmissione. Se in questo tempo il modulo non sente nulla in arrivo sul buffer d'uscita costruisce il pacchetto e lo spedisce. La massima dimensione della PDU è di 100 byte. Il valore impostato sui nodi è 0x08 il che significa che se all'interno del buffer d'invio non viene passato alcun byte dopo 0,55 millisecondi viene costruito il pacchetto e quindi spedito.

$$T_{char} = \frac{8}{115200} = 0,069 \quad ms$$

impostando a 8 il valore di RO risulta:

$$RO(8) = 8 \cdot 1000 \cdot 0,07 \cdot 10^{-}3 = 0,55 \quad ms$$

- `ATMM`: il comando si riferisce alla possibilità di utilizzare diverse configurazioni del protocollo 802.15.4. Il range di possibili valori sono tra 0-2."\n" Con il valore 0: viene utilizzato il protocollo 802.15.4 con la header MaxStream la quale inserisce una serie di campi aggiuntivi quali controllo CRC e serie di bit che informano l'inizio della PDU. Nelle modalità 1 e 2 utilizza il sistema 802.15.4 base: con 1 non si usufruisce degli ACK con 2 si attivano gli ACK. Nell'applicazione costruita si è scelto la modalità ATMM 1.

- `ATWR`: scrive definitivamente sul modulo radio i settaggi cambiati.

- `ATCN`: si esce dalla modalità di settaggio.

In alternativa è possibile usufruire del programma software X-CTU che permette attraverso una interfaccia grafica di poter settare il modulo radio in modo semplice e veloce.

B.1 Prestazioni del Modulo con diverse configurazioni

Di seguito vengono riportati alcuni grafici sulle prestazioni del nodo con settaggi diversi. Si dimostra visivamente attraverso grafici come la configurazione scelta sia la migliore ottenuta avendo le prestazioni più elevate.

B.1.1 Il comando ATMM

Configurazione iniziale due nodi con schedulazione con doppia cioè uno slot di BDG campionatura ad 1 Khz. Numero di campioni inviati 10000 per ogni nodo. Nodi utilizzati 35 e 123. In questa prova si evidenzia perché si è scelto di inviare i pacchetti in modalità base 802.15.4 senza ACK. La configurazione dei parametri è ATBD 7, ATRO 8, ATRN 0.

ATMM 0	*ATMM 1*	*ATMM 2*
66.5%	99.0%	123.0%

Tabella B.1: Tasso ricezione dei pacchetti al variare del MAC (comando ATMM).

La Tabella B.1 mostra il comportamento della rete sul totale dei pacchetti inviati dai due nodi. Come si può notare la scelta migliore è la soluzione senza ACK in quanto la più costante nei risultati e molto vicini al raggiungimento del 100 per cento dei pacchetti arrivati. Le considerazioni invece sull'opzione 0 del comando ATMM che si possono dare sono sulle dimensione finale della PDU. L'aggiunta di informazioni provoca una PDU molto grande che necessita di maggior tempo per la trasmissione andando a consumare gli slot destinati all'invio destinati ad altri nodi. Risulta poco affidabile invece dando risultati inverosimili il comando ATMM 2. Infatti arrivano sul destinatario molti più pacchetti di quanti non ne siano stati inviati. Questo è da ricondursi al fatto che vengono persi gli ACK. Analizzanddownloado il file di uscita si può notare come in realtà esistano molte copie di pacchetti duplicati.

B.1.2 Il comando ATRN

Ora si analizza il comportamento della rete al variare di ATRN. Questa prova è stata effettuata con tre nodi sempre ad una frequenza di 1 KHz con ATRO 0x08 e ATMM 1 e velocità posta sempre a 115200 b/s. Ogni nodo spedisce 20000 pacchetti.

ATRN 0	*ATRN 1*	*ATRN 2*	*ATRN 3*
53.6%	95.6%	40.7%	72.2%

Tabella B.2: Tasso ricezione dei pacchetti al variare del MAC (comando ATRN).

Come si può vedere dalla Tabella B.2 il sistema migliore è quello con ATRN 1 dove il totale dei pacchetti giunti sul destinatario è vicina al 100 per 100 ed è la più costante. Con questa scelta viene attivato il protocollo CSMA-CA con un solo tentativo di spedizione in caso di occupazione del canale.

B.1.3 Il comando ATRO

Per quanto riguarda il comando ATRO sono state eseguite due serie di prove settando il comando ATRO a valore 0x04 e una con ATRO con valore 0x08. Questi valori sono stati scelti in quanto aumentando o diminuendo tale valore il numero di pacchetti persi peggiora drasticamente. Con questi due valori invece è possibile notare che con l'aumentare delle frequenze ATRO 0X04 peggiora mentre 0x08 si mantiene costante nelle prove e soprattutto vicino alla soglia massima. I valori sono riportati in Tabella B.3. Per frequenze basse sono entrambe validi. La media totale per ATRO 8 è di 99,14% mentre quella di ATRO 4 è di 80,19%

frequenza Hz	*ATR0 8*	*ATRO 4*
50	98,84%	99,52%
100	99,99%	93,75%
200	97,96%	95,14%
250	99,91%	60,23%
500	98,90%	71,95%
1000	99,27%	60,55%

Tabella B.3: Tasso ricezione dei pacchetti al variare del MAC (comando ATRO).

Appendice C

Codici e diagrammi di flusso

Questa appendice raccoglie i codici sorgenti realizzati per il funzionamento del gateway e dei nodi sensore della piattaforma W-TREMORS. Inoltre è illustrato anche il principio di funzionamento del sistema costruito al fine di interfacciare la rete con i programmi di analisi strutturale attualmente impiegati da EUCENTRE. Quasi tutto il codice è stato scritto in linguaggio C ([40],[23]) ed è rilasciato sotto licenza Open Source.

C.1 Gateway

```
1  /*
2   * SquidBee Gateway - Retreiving the data - V.0.0
3   * 05/12/2008
4   * Author: Marco Beltrame & Emanuele Goldoni
5   * The project: http://www.squidbee.com
6   * The company: hhtp://www.libelium.com
7   * Platform : W-TREMORS
8   */
9
10 #include <stdio.h>
11 #include <string.h>
12 #include <unistd.h>
13 #include <fcntl.h>
14 #include <errno.h>
15 #include <stdlib.h>
16 #include <termios.h> /* Terminal control library (POSIX) */
17 #include <signal.h>
18
19 #define MAX 100
20 #define SIMPLEDEBUG
21 #define SIGNALGW '@'
22 #define STARTCHAR 's'
23 #define MAX_NUMNODI 99
24 #define HEADER_BYTE 7
25 #define THROUGHPUT_BYTE 14400
26
27 //global variables
28 unsigned int frequenza;
29 int numnodi = 0;
30 int nodi[MAX_NUMNODI];
```

```
31 FILE *outStream;
32
33
34 //declaration of methods that use in programm
35 void init_termios(int);
36 unsigned char readSerial(int);
37 void write_Output(int);
38 unsigned char id();
39 int sendPing(int, unsigned char);
40 int sendBurstOffset(int, unsigned char, unsigned char, unsigned char);
41 void sendStart(int);
42 int checkParametri(int , int );
43 void readConfig( char[] );
44 int sendFrequency(int ,unsigned char ,int );
45
46 /*
47 method that set statements of serial port
48 in input and in output
49 */
50
51 void init_termios(int sd) {
52         struct termios opciones;
53         memset(&opciones, 0, sizeof(struct termios));//erase all the
             statements previouses
54         tcgetattr(sd, &opciones);
55         //Set the speed od serial
56         cfsetispeed(&opciones, B115200);
57         cfsetospeed(&opciones, B115200);
58         //Set the C_FLAGS
59         opciones.c_cflag |= ( CLOCAL | CSTOPB | CREAD ); // enable
60         opciones.c_cflag &= ~( PARENB | PARODD);//disable
61         opciones.c_cflag &= ~CSIZE;//erase the mask of CSIZE
62         opciones.c_cflag |= CS8;//set the CSIZE on 8 bit
63         //Set the I_FLAG
64         opciones.c_iflag |= ~(INPCK | PARMRK | ISTRIP); //disable the
             parity control
65         opciones.c_iflag |= ( IGNPAR | IGNBRK | BRKINT ); //ignore
             error of parity
66         opciones.c_iflag &= ~(IXON | IXOFF | IXANY | IGNCR | ICRNL |
            INLCR );//disable control flows
67         opciones.c_iflag |= IMAXBEL;//enable a bell when the tail in
             input is full
68         //Set L_FLAG
69         opciones.c_lflag =0;
70         opciones.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);//disable
             the canonical form
71         //Set O_FLAG
72         opciones.c_oflag = 0; //disable all
73         opciones.c_oflag &= OPOST;
74         //opciones.c_oflag |= OFILL;
75         //others statements
76         opciones.c_cc[VMIN] = 1; //must be read at least a char without
              timeout
77         opciones.c_cc[VTIME] = 0;
78         tcsetattr(sd, TCSANOW, &opciones); //set the new options of the
              serial port
79         tcflush(sd,TCOFLUSH);
80         tcflush(sd,TCIFLUSH);
```

```
81  }
82
83  /*
84  method that read the serial: if a char belong to range of values the
        progamm send informations to the device
85  */
86
87  unsigned char readSerial(int sd) {
88
89          unsigned char valor[MAX];
90          int j=0;
91          unsigned char d;
92          memset(valor, 0, sizeof(valor));
93
94          read(sd , &d , 1);
95                  valor[j] = d;
96                  j++;
97
98                  if((j==(MAX-1)) || (d=='/') || (d=='!') || (d=='&') ||
                        (d>48)) {
99
100                         int x;
101                         for(x=0;x<j;x++) {
102                                 valor[x]='\0';
103
104                         }
105                         j=0;
106                 }
107         return d;
108 }
109
110 /*
111 method that write the values, that the device send, in a file
112 */
113
114 void write_Output(int sd) {
115
116         unsigned char vector[MAX];
117         unsigned char c;
118         int z = 0;
119         int x = 0;
120         memset(vector, 0, sizeof(vector));
121
122         while(1) {
123
124                 read(sd,&c,1);
125                 vector[z] = c;
126                 z++;
127
128                 if( (c == '\n') || (z == (MAX-1)) ) {
129
130                         for(x = 0; x < z; x++) {
131                                 fprintf(outStream,"%c",vector[x]);
132                                 vector[x]=0;
133                         }
134                         z=0;
135                         fflush(outStream);
136                 }
```

```
137
138         }
139 }
140
141 /*
142 method that check if a device is attainable (ping)
143 */
144
145 int sendPing(int sd, unsigned char destination) {
146
147         unsigned char specialchar = SIGNALGW;
148         unsigned char pong;
149
150         printf("Pinging node %i...\n", destination);
151
152         //spedisco @
153         write(sd, &specialchar, 1);
154         tcflush(sd, TCIOFLUSH);
155         //spedisco il my del nodo
156         write(sd, &destination, 1);
157         tcflush(sd, TCIOFLUSH);
158
159         pong = readSerial(sd);
160         if (pong == '!') {
161                 printf("Pong :)\n");
162                 return 0;
163         } else {
164                 printf("Buuhh :(\n");
165                 return -1;
166         }
167 }
168
169 /*
170 method that set to all devices of the network the scheduler that they'
        ll use.
171 -burst=final dimension of the packet
172 -offset=dimension of the packet at the start
173 */
174 int sendBurstOffset(int sd, unsigned char destination, unsigned char
        burst, unsigned char offset) {
175
176         unsigned char specialchar = SIGNALGW;
177         unsigned char pong;
178
179         printf("Sending scheduling to node %i... \n", destination);
180
181         //contol var "@"
182         write(sd, &specialchar, 1);
183         tcflush(sd, TCIOFLUSH);
184         //control var  "name of motes"
185         write(sd, &destination, 1);
186         tcflush(sd, TCIOFLUSH);
187         //send the scheduler
188         printf("burst %c - ", burst);
189         write(sd,&burst,1);
190         printf("offset %c\n", offset);
191         write(sd,&offset,1);
192         tcflush(sd, TCIOFLUSH);
```

```
193         //verify if the motes learn
194         pong = readSerial(sd);
195         if (pong == '&') {
196                 printf("Scheduled :)\n");
197                 return 0;
198         } else {
199                 printf("Buuhh :(\n");
200                 return -1;
201         }
202
203 }
204
205 /*
206 method that send the start to all of motes to capture the values (
        synchronization)
207 */
208 void sendStart(int sd) {
209
210         unsigned char start = STARTCHAR;
211         unsigned char specialchar = SIGNALGW;
212         char inutile;
213         printf("Press INVIO to start acquisition.");
214         scanf("%c", &inutile);
215         printf("Broacasting start signal...\nStarted!\n");
216         write(sd,&specialchar,1);
217         write(sd,&start,1);
218
219 }
220
221 /*
222 method that check the statements at the biginning on the file where
        there are write frequequency that use, number of motes and
223 the names of motes
224 */
225
226 int checkParametri(int n, int f) {
227
228         int s_max = ((THROUGHPUT_BYTE / f) - HEADER_BYTE);
229         int f_max = THROUGHPUT_BYTE / (HEADER_BYTE + (1 * n));
230
231         if (n > s_max || f > f_max) {
232                 printf("Errore nell'impostazione dei parametri di
                        sistema:\n ridurre il numero di nodi a %i o ridurre
                        la frequenza a %i Hz\n", s_max, f_max);
233                 return -1;
234         } else {
235                 return 1;
236
237         }
238
239 }
240
241 /*
242 Method that read the file with the statements at the biginning.
243 Check first control
244 */
245
246 void readConfig(char filename[]) {
```

```
247
248         char parametro;
249         int valore;
250         FILE *lettura;
251
252         if(( lettura = fopen(filename, "r")) == NULL) {
253                 printf("Errore: non riesco a leggere il file %s\n",
                        filename);
254
255         } else {
256
257                 fscanf(lettura,"%c %i\n", &parametro, &valore);
258                 if ( parametro == 'f') {
259                         frequenza = valore;
260                         printf("Frequenza di campionamento: %i Hz\n",
                                frequenza);
261                         printf("Delay da impostare sui nodi: %i millis\
                                n", 1000/frequenza);
262                 } else {
263                         printf("Errore nel file di configurazione:\n
                                impostare correttamente la frequenza di
                                campionamento\n");
264                         exit(-1);
265                 }
266
267                 fscanf(lettura,"%c %i\n", &parametro, &valore);
268
269                 if ( parametro == 'n') {
270                         numnodi = valore;
271                         printf("Numero di nodi: %i\n", numnodi);
272                 } else {
273                         printf("Errore nel file di configurazione:\n
                                impostare correttamente la frequenza di
                                campionamento\n");
274                         exit(-1);
275                 }
276
277                 int i = 0;
278
279                 while(!feof(lettura)) {
280
281                         fscanf(lettura,"%i\n", &valore);
282                         nodi[i] = valore;
283                         printf("ID nodo %i: %i\n", i, nodi[i]);
284                         i++;
285                 }
286
287                 //check number of motes with the sum of names of motes
288
289                 if (i < numnodi) {
290                         printf("Errore nel file di configurazione:\n
                                numero di nodi errato o %i identificativi
                                di nodi mancanti\n", numnodi-i);
291                         exit(-1);
292                 } else if (i > numnodi) {
293                         printf("Errore nel file di configurazione:\n
                                numero di nodi errato o %i identificativi
                                di nodi in eccessi\n", i-numnodi);
```

```
294                         exit(-1);
295                 }
296
297                 fclose(lettura);
298
299                 if (checkParametri(numnodi, frequenza)) {
300                         // call checkParametri
301                         // verify the function with tie frequency-
                                numbers of motes
302                 } else {
303                         exit(-1);
304                 }
305
306         }
307
308 }
309
310 /*
311 method that send in broadcasting to motes of network frequency to use
        write in the config file
312 */
313
314 int sendFrequency(int sd,unsigned char destination, int frequency) {
315
316         unsigned char speciachar = SIGNALGW;
317         unsigned char freq;
318         //control var "@"
319         write(sd, &speciachar, 1);
320         tcflush(sd, TCIOFLUSH);
321         //control var  "name of motes"
322         write(sd, &destination, 1);
323         tcflush(sd, TCIOFLUSH);
324         //send frequency
325         unsigned char high;
326         unsigned char low;
327         high = (frequency >> 8) & 0x00FF;
328         low = frequency & 0x00FF;
329         write(sd,&high,1);
330         write(sd,&low,1);
331         tcflush(sd, TCIOFLUSH);
332
333         /*
334         Call readSerial to verify that all of motes receives the
                correct frequency
335         */
336
337         freq = readSerial(sd);
338
339         if (freq == '*') {
340                 printf("Frequency :)\n");
341                 return 1;
342         } else {
343                 printf("Buuhh :(\n");
344                 return -1;
345         }
346
347
348 }
```

```
349
350 /*
351 main of the application
352 */
353
354 int main(int argc, char** argv) {
355
356         int sd;
357         char *serialPort="/dev/ttyUSB0";
358         unsigned char dest;  //identifier;
359         int return_code = 1;
360
361         /*
362         the programm have in input the OUTPUT_FILENAME
363         */
364
365         if( argc!=2 ){
366                 fprintf(stderr, "Usage: %s OUTPUT_FILENAME \n", argv
                        [0]);
367                 exit(1);
368         }
369
370         /*
371         Open the file in input
372         */
373
374         char *filename = argv[1];
375
376         if ( !( outStream = fopen(filename, "w") ) ) {
377                 printf("Errore: can't open specified ouput file %s\n",
                        filename);
378                 exit(2);
379         }
380
381         /*
382         verify if the serial port is open
383         */
384
385         if ((sd = open(serialPort, O_RDWR | O_NOCTTY | O_NDELAY)) ==
                -1) {
386                 fprintf(stderr,"Unable to open the serial port %s - \n"
                        , serialPort);
387                 exit(-1);
388         } else {
389                 fprintf(stderr,"Serial Port open at: %i\n", sd);
390                 fcntl(sd, F_SETFL, 0);
391         }
392
393         //call the statements of serial
394         init_termios(sd);
395         tcflush(sd, TCIOFLUSH);
396
397         //read the statements in file of configuration
398         readConfig("./cfg/global.cfg");
399         sleep(3);
400
401         /*
402         the algoritm makes two operation:
```

```
403         1-send a ping to all the motes
404         2-calculate automatically the burst and l'offset and send both
                to each mote
405         */
406
407         int l;
408         for(l = 0; l < numnodi; l++) {
409                 dest = nodi[l];
410                 return_code = sendPing(sd, dest);
411                 return_code = sendBurstOffset(sd, dest, numnodi+'0', l+
                        '1');
412                 unsigned int time;
413                 time = (1000/frequenza);
414                 return_code = sendFrequency(sd, dest ,time);
415         }
416
417         sleep(2);
418
419         /*
420         Call the method sendStart(int)
421         */
422         sendStart(sd);
423         /*
424         Write on file the output values
425         */
426         write_Output(sd);
427         /*
428         Close the serial port
429         */
430         close(sd);
431         /*
432         Clear the streaming of the file
433         */
434         fflush(outStream);
435         /*
436         Close the output file
437         */
438         fclose(outStream);
439
440         return 0;
441
442
443 }//END APPLICATION
```

C.2 Mote

```
1  /*
2   * SquidBee Motes - Retreiving the data - V.0.0
3   * 05/12/2008
4   * Author:Emanuele Goldoni & Marco Beltrame
5   * The project: http://www.squidbee.com
6   * The company: hhtp://www.libelium.com
7   * Platform : W-TREMORS
8   */
9  #define MAX_LETTURE 99
10 /*
11 Global var
```

```
12  */
13  int ADDRESS_SIZE = 3; //8 bit
14  char SIGNALGW = '@';
15  unsigned char nodeID = 0;
16  int burst = 0;
17  int offset = 0;
18  int i;
19  unsigned char frequencyH;
20  unsigned char frequencyL;
21  unsigned int frequency;
22  unsigned int currentTime;
23  unsigned int startTime;
24  int contatore = 0;
25  unsigned char letture[MAX_LETTURE];
26
27  /*
28  set the pin where connect the motes
29  */
30
31  int pin = 2;
32  int value = 0;
33
34  /*
35  Declerate methods use in this programm
36  */
37
38  char readByte();
39  char returnedOK();
40  unsigned char getMYID();
41
42  /* START APPLICATION */
43
44  /*
45     Method VOID set the initial statement. It read only one.
46     Set the speed of serial port 115200 b/s
47  */
48
49  void setup() {
50    Serial.begin(115200);
51  }
52
53  /*
54    Method recurrence
55  */
56
57  void loop() {
58
59    /*
60      Set the variable to prepare motes.
61    */
62    int step0 = 0;
63    int step1 = 0;
64    int step2 = 0;
65    int step2a = 0;
66    int step3 = 0;
67
68    //The mote learns the name of radio.
69    while(step0 == 0) {
```

```
70      nodeID = getMYID();
71      if (nodeID > 0) {
72        step0 = 1;
73      } else {
74        while(1) {
75          //do nothing
76        }
77      }
78    }
79
80    //Il gateway send a ping to radio.
81    while(step1 == 0) {
82      if (readByte() == SIGNALGW) {
83        if (readByte() == nodeID) {
84          step1 = 1;
85          Serial.print("!");
86        }
87      }
88    }
89
90   // Gateway informs motes of burst and offset to use.
91   while ( step2 == 0 ) {
92      if (readByte() == SIGNALGW) {
93        if (readByte() == nodeID) {
94            burst =(int) (readByte() - '0');
95            offset =(int) (readByte() - '0');
96            step2 = 1;
97            Serial.print("&");
98        }
99      }
100   }
101
102   // Gateway informs motes of frequency to use.
103   while ( step2a == 0) {
104     if (readByte() == SIGNALGW) {
105       if (readByte() == nodeID) {
106           frequencyH = Serial.read();
107           frequencyL = Serial.read();
108           frequency =(int) (frequencyH*256 + frequencyL);
109           step2a = 1;
110           Serial.print("*");
111       }
112     }
113   }
114
115   //Gateway send all motes start of application.
116   while ( step3 == 0 ) {
117        if (readByte()==SIGNALGW) {
118          if (readByte()=='s') {
119            step3 = 1;
120          }
121        }
122   }
123
124   //Set variables to count time
125   startTime=millis();
126   currentTime = 0;
127
```

```
128   //Statement del burst
129   if (offset < burst) {
130     //read values of sensor
131     for (contatore = 0; contatore < offset; contatore++) {
132       letture[contatore] = ((analogRead(pin) >> 2) & 0x00FF);
133       //Time to sampling
134       delay(frequency);
135     }
136     //built packet to send
137     //HEADER
138     Serial.print("@");
139     Serial.print(nodeID, BYTE);
140     Serial.print((currentTime>>8) & 0x00FF, BYTE );
141     Serial.print(currentTime & 0x00FF, BYTE);
142     Serial.print(contatore, BYTE);
143     //values of motes
144     for (i = 0; i < offset; i++) {
145       Serial.print(letture[i], BYTE);
146     }
147     //TAILER
148     Serial.print('\n', BYTE);
149     Serial.print('\r', BYTE);
150   }
151   //Start the burst
152   while(1) {
153     //calculate time
154     currentTime = millis() - startTime;
155     //motes make reads
156     for (contatore = 0; contatore < burst; contatore++) {
157       letture[contatore] = ((analogRead(pin) >> 2) & 0x00FF);
158       //time sampling
159       delay(frequency);
160     }
161     //Built packet
162     //HEADER
163     Serial.print("@");
164     Serial.print(nodeID, BYTE);
165     Serial.print((currentTime >> 8) & 0x00FF, BYTE );
166     Serial.print(currentTime & 0x00FF, BYTE);
167     Serial.print(contatore, BYTE);
168     //values of motes
169     for (i = 0; i < burst; i++) {
170       Serial.print(letture[i], BYTE);
171     }
172     //TAILER
173     Serial.print('\n', BYTE);
174     Serial.print('\r', BYTE);
175   }
176
177 }//END PROGRAMM
178
179 //Methods that acquire the id of the radio.
180
181 unsigned char getMYID() {
182   //Set variable
183   int l = 0;
184   char c = 48;
185   char nodeMyID[ADDRESS_SIZE];
```

```
186   memset(nodeMyID, '0', ADDRESS_SIZE);
187   unsigned char numericID = 0;
188   //delay
189   delay(10000);
190   Serial.print("+++");
191   delay(1100);
192
193   //Call the function returnedOK
194
195   if (returnedOK() == 'F') {
196     while(1) {
197       Serial.println("LED"); // inserire led 13 lampeggiante
198       delay(500);
199     }
200   } else {
201     Serial.print("ok␣+++");
202     Serial.println("ATMY");
203     delay(3500);
204     c = readByte();
205     //Algoritm that calculate the ID
206     while(c >= 48 && l < ADDRESS_SIZE) {
207       numericID = numericID * 10 + (c - '0');
208       //call the method readByte()
209       c = readByte();
210       l++;
211     }
212   }
213   Serial.println("ATCN");
214   return numericID;
215 }
216
217 //Method that reads values on serial port
218
219 char readByte() {
220     while (Serial.available() == 0) {
221           // do nothing
222           delay(10);
223     }
224     return (char) Serial.read();
225 }
226
227 //Method that verify the name of radio
228
229 char returnedOK() {
230   // this function checks the response on the serial port to see if it
        was an "OK" or not
231   char incomingChar[3];
232   char okString[] = "OK";
233   char result = 'n';
234   int startTime = millis();
235   while (millis() - startTime < 2000 && result == 'n') { // use a
        timeout of 10 seconds
236     if (Serial.available() > 1) {
237     // read three incoming bytes which should be "O", "K", and a
          linefeed:
238       for (int i=0; i<3; i++) {
239         incomingChar[i] = Serial.read();
240       }
```

```
241       if ( strstr(incomingChar, okString) != NULL ) { // check to see if
              the respose is "OK"
242       // if (incomingChar[0] == 'O' && incomingChar[1] == 'K') { // check
              to see if the first two characters are "OK"
243         result = 'T'; // return T if "OK" was the response
244       } else {
245         result = 'F'; // otherwise return F
246       }
247     }
248   }
249   return result;
250 }
```

C.3 Funzionamento del programma W-TREMORS

Dalla cartella "W-TREMORS" si deve far eseguire lo script "main.sh". Apparirà la finestra come in Figura C.1

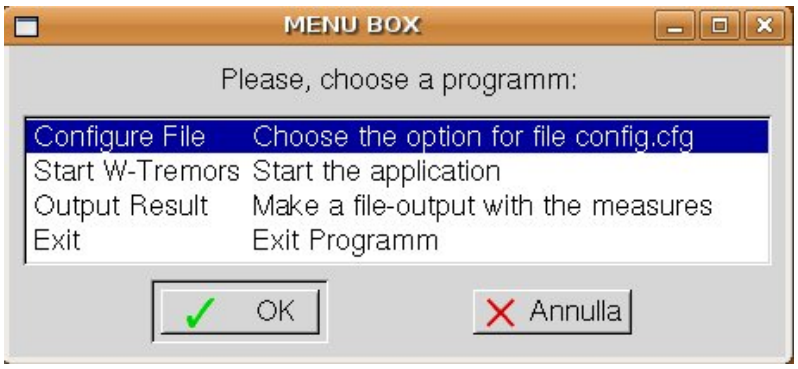


Figura C.1: Finestra di inizio programma.

E' possibile scegliere tra quattro opzioni:

- *Configure File*: selezionando questa scelta si settano le opzioni del file di configurazione (frequenza di campionatura, numero nodi, nome dei nodi).
- *Start W-Tremors*: mediante questa scelta si fa partire l'applicazione.
- *Output Result*: costruisce il file finale con i risultati.
- *Exit*: uscita dal programma.

C.3.1 Configure File

Con questa opzione si sceglie la configurazione del file. La prima scelta da operare è la frequenza di campionatura che si vuole applicare (Figura C.2).

Successivamente viene chiesto di inserire il numero totale di nodi che faranno parte della rete (Figura C.3).

Come ultimo passo viene richiesto il nome di ogni singolo nodo. Come nome si intende l'indirizzo di scheda scheda radio (Figura C.4).

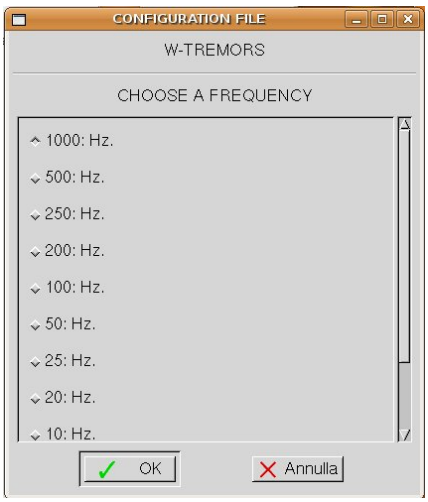


Figura C.2: Finestra per la scelta della frequenza di campionatura.

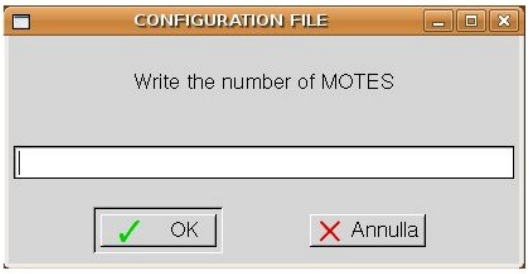


Figura C.3: Finestra per l'inserimento del numero di nodi.

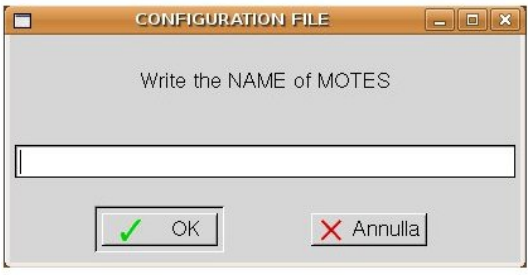


Figura C.4: Finestra per l'inserimento dei nomi dei nodi.

C.3.2 Start W-Tremors

Questa opzione fa partire l'applicazione seguendo le indicazioni inserite nel file di configurazione. All'inizio viene chiesto il nome del file su cui l'applicazione andrà ad inserire i valori catturati dai nodi (Figura C.5).

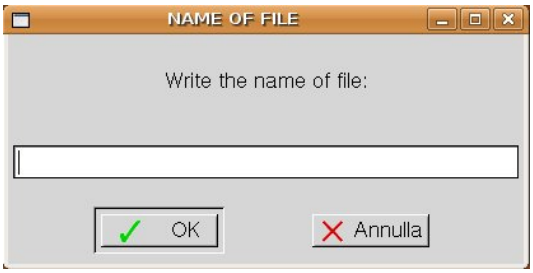


Figura C.5: Finestra per l'inserimento del nome del file dove verranno registrate le misure.

C.3.3 Output Result

Il nome del file richiesto dall'opzione *Start W-Tremors* produce un file di testo in ascii *(byte)* non leggibili. Questo è dovuto al fatto che l'applicazione per essere più performante non converte direttamente i valori catturati dai in un formato testuale leggibile ma li inserisce semplicemente convertendoli successivamente, una volta terminata l'applicazione. Questi file scritti in ascii possono essere convertiti in un secondo momento. Questa opzione permette di convertire tali file: per questo motivo facendo partire *Output Result* viene richiesto di inserire il nome del file in ascii inserito precedentemente per la cattura dei dati (Figura C.6). Questa procedura verrà, successivamente illustrata con uno schema a blocchi.

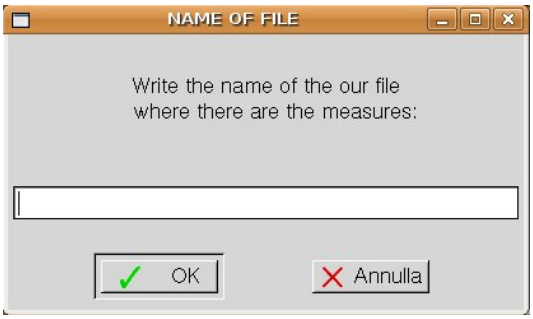


Figura C.6: Finestra per l'inserimento del nome del file per la conversione delle misure.

C.4 Diagramma di flusso del file di output

Come mostrato nel diagramma di flusso nella Figura C.7, una volta terminata la misurazione, il file creato necessita di una procedura. Questa procedura è stata

necessaria per avere alla fine un file testuale identico a quello che viene creato utilizzando il sistema wired di EUCENTRE. Tuttavia il file costruito dalla piattaforma W-TREMORS è nella forma:

$$\texttt{id}_{nodo} \quad \texttt{timestamp}_{burst} \quad \texttt{valore}_1 \quad \ldots \quad \texttt{valore}_n$$

per arrivare alla forma desiderata:

$$\texttt{tempo} \quad \texttt{valore}_{nodo1} \quad \ldots \quad \texttt{valore}_{nodoN}$$

con i valori espressi in g e salvati in un file testuale.

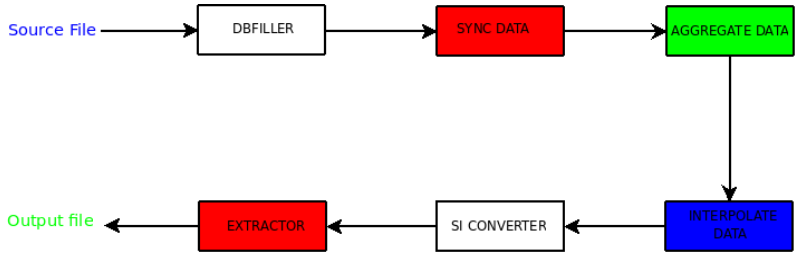


Figura C.7: Finestra per l'inserimento del nome del file per la conversione delle misure.

- *DBFILLER*: il file entra nel programma DBFILLER il quale trasforma i valori in byte inseriti nel file e li converte in formato leggibile. Durante la conversione questi valori vengono inseriti in una tabella del database nella forma:

$$\texttt{id}_{nodo} \quad \texttt{tempo} \quad \texttt{valore}$$

- *SYNC DATA*: dopo aver costruito la tabella i valori di tempo vengono sincronizzati. Come detto nel capitolo della sincronizzazione, esiste un problema di drift. Il tempo conteggiato dai nodi si sfasa di 1 o 2 microsecondi in questa applicazione i tempi vengono tutti riordinati togliendo le eventuali sfasature. Esempio il nodo X ha campionato al tempo 1000 mentre il nodo Y ha campionato al tempo 1001: per livellare tutti allo stesso tempo il tempo del nodo Y viene portato a 1000.
- *AGGREGATE DATA*: in questa sezione i dati nella tabella vengono riordinati secondo il tempo e sarà così formata:

$$\texttt{tempo} \quad \texttt{valore}_{nodo1} \quad \ldots \quad \texttt{valore}_{nodoN}$$

- *INTERPOLATE DATA*: se esistono dei valori NULL nelle tuple a causa di perdita di pacchetti, i valori vengono aggiunti facendo una media tra il il primo valore precedente disponibile e il primo valore successivo disponibile
- *SI CONVERTER*: i valori vengono convertiti nella misura del Sistema Intenazionale dando in uscita l'accelerazione in g.

- *EXTRACTOR*: dal database viene estratto e creato il file finale.

Il database utilizzato è SQLite. Esso è molto leggero e facilmente interfacciabile con il linguaggio di programmazione C, infatti possiede di default, alla sua installazione, metodi in C per eseguire qualsiasi query SQL possibile. E' un database relazionale con un uso di librerie molto limitato le quali lo rendono snello nelle operazioni di I/O. Il software è gratuito e utilizza il linguaggio SQL per interrogare il database. A differenza della maggior parte degli altri database SQL, SQLite non dispone di un server separato processo. Un completo database SQL con più tabelle, indici, trigger, e punti di vista, è contenuta in un singolo file su disco[55].

C.4.1 Diagramma di flusso per il calcolo del burst e dell'offset

In questa sezione viene illustato il diagramma di flusso, presentato nella Figura C.8, per il calcolo del burst e dell'offset nella piattaforma W-TREMORS e il diagramma temporale del colloquio tra il coordinatore della rete e i nodi nella fase si settaggio dei parametri nella Figura C.9.

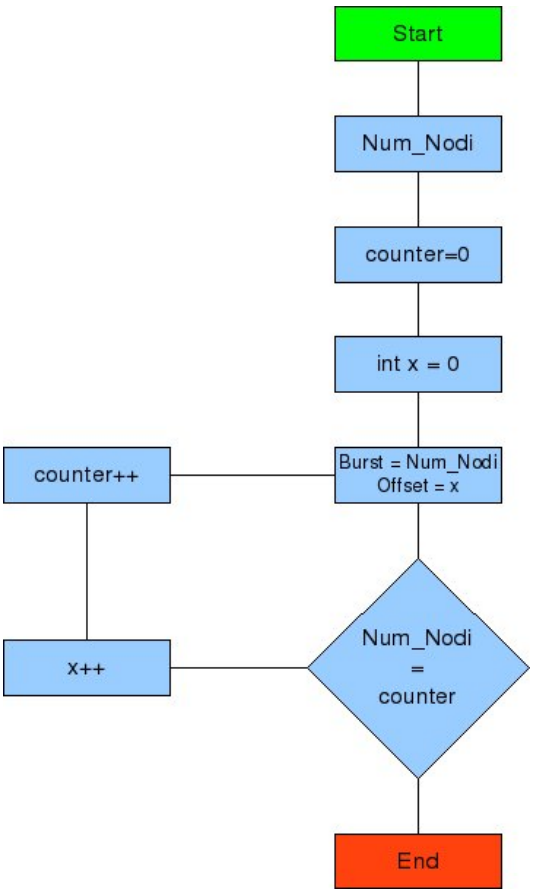


Figura C.8: Diagramma di flusso del calcolo del burst e dell'offset.

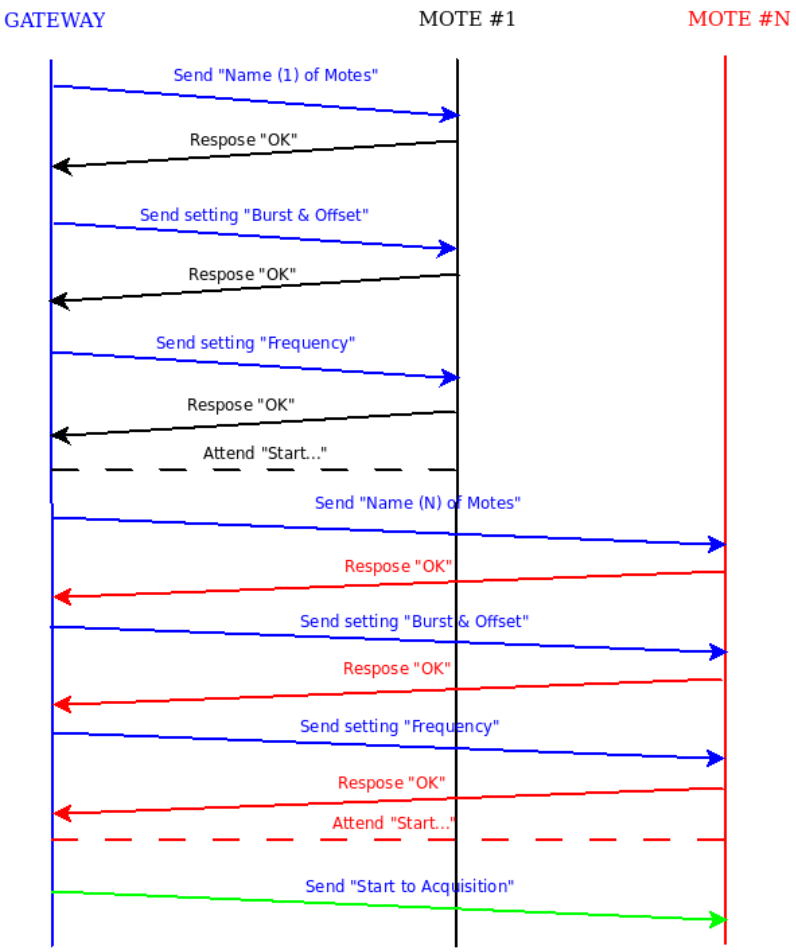


Figura C.9: Diagramma temporale del colloquio tra coordinatore e nodi.

Appendice D

Rete di condizionamento

L'accelerometro in dotazione in EUCENTRE sono i ServoK-Beam Accelerometer della Kistler del tipo 8330A3. Sono accelerometri a singolo asse per misurazioni a basso livello e a bassa frequenza. Dispone di una eccelente risposta in frequenza resistente anche alle escursioni termiche. La sua sensibilità è di 1200 mV/g ciò significa che ad una tensione di 1,2 Volt corrisponde ad una accelerazione di un g. La sua alimentazione è bipolare e compresa tra 6 V e 15 V in corrente continua. Il range di misurazione va da $\pm 3g$.

Per poter configurare gli accelerometri ai nodi si necessita di una costruzione di una rete di condizionamento, visibile nella Figura D.1 e nella fotografia Figura D.2, poichè il suo range di alimentazione è compreso tra 0 e 5 Volt. La rete avrà quindi la funzione di traslare il range da $\pm 2,4V$ dell'accelerometro all'intervallo 0-5 Volt utilizzato dall'ADC ingrato nell'AVR ATMega168. Questa traslazione circuitale deve far combaciare lo 0 dell'accelerometro esattamente con 2,5 Volt dell'alimentazione del processore. Gli amplificatori operazionali sono JFET TI TLE2071. Grazie all'aiuto del professor Lodovico Ratti è stato disegnato il circuito che effettua la traslazione ed inverte le accelerazioni acquisite: i valori compresi tra 0 e 2,5 V equivalgono ad accelerazioni positive mentre a valori di voltaggio maggiori di 2,5 Volt corrispondono accelerazioni negative. Si ringrazia per la costruzione dei prototipi delle reti di condizionamento il Sig. Goldoni Sergio.

Per quanto riguarda la conversione A/D effettuata dal microprocessore, l'ADC integrato nell'ATmega168 ha una risoluzione di 10 bit, con un range che varia quindi tra i valori 0 e 1024. Questo significa che variazioni di $1g$, corrispondenti ad una variazione di tensione pari a 1,2V, vengono rappresentate con 245 bit; in altre parole, la precisione massima di una misurazione ottenibile con questo sistema è pari a $0.004g$. Inoltre la conversione della misurazione fornita dall'ADC in valori di accelerazione in g è così descritta:

$$a_{SI} = \frac{512 - X_{ADC}}{245} g$$

Il tempo massimo di conversione stimato è di 260 μs; questo significa che a campionature superiori a 4 KHz l'ADC potrebbe non essere efficace.

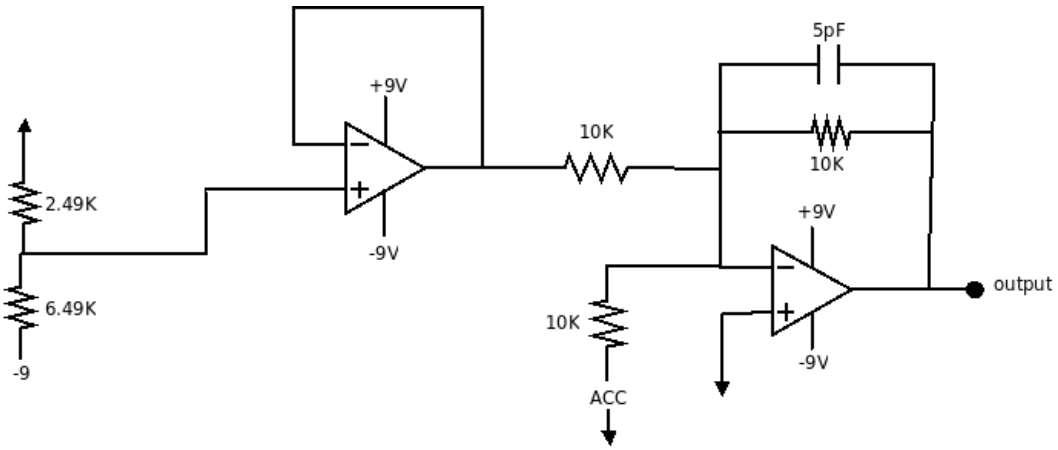


Figura D.1: Rete di condizionamento.

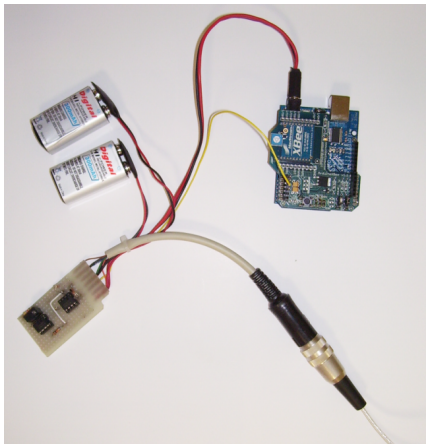


Figura D.2: Fotografia della Rete di condizionamento.

Bibliografia

[1] www.arduino.cc.

[2] www.libelium.com.

[3] IEEE Std 802.15.4-2003 - Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). 2003.

[4] IEEE Std 802.15.1-2005 (Revision of IEEE Std 802.15.1-2002) - Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs). 2005.

[5] IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003) - Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). 2006.

[6] IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-19999) - Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. 12 2007.

[7] IEEE Std 802.15.4a-2007 (Amendment to IEEE Std 802.15.4-2006) - Alternative Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). 2007.

[8] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol (EAP). RFC 3748 (Proposed Standard), June 2004. Updated by RFC 5247.

[9] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325–349, 2005.

[10] I.F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications Magazine, IEEE*, 40(8):102–114, Aug 2002.

[11] S.N. Applications. Monitoring Civil Structures with a Wireless Sensor Network. 2006.

[12] Arduino. Arduino board, 2008.

[13] ATMEL. *AVR datasheet ATmega*, 07 2008.

[14] G. Bartelds. Aircraft Structural Health Monitoring, Prospects for Smart Solutions from a European Viewpoint. *Structural Health Monitoring. Current Status and Perspectives*, pages 18–20, 1998.

[15] Christian Bettstetter and Roland Krausser. Scenario-based stability anlysis of the distributed mobility-adaptive clustering (dmac) algorithm. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 232–241, New York, NY, USA, 2001. ACM.

[16] J. Beutel. Fast-prototyping using the btnode platform. *Design, Automation and Test in Europe Conference and Exhibition*, 1:207, 2006.

[17] R. Bolla. Architetture e protocolli per reti dati Wireless. *Standard IEEE*, 802.

[18] brian w. evans. *Arduino programming notebook*, 08 2007.

[19] E. Callaway, P. Gorday, L. Hester, J.A. Gutierrez, M. Naeve, B. Heile, and V. Bahl. Home networking with ieee 802.15.4: a developing standard for low-rate wireless personal area networks. *Communications Magazine, IEEE*, 40(8):70–77, Aug 2002.

[20] G. Ciaburro. *Matlab. Versione 7. xe precedenti. Guida all'uso*. Edizioni FAG Srl, 2007.

[21] Riccardo Crepaldi. *Algoritmi di localizzazione per reti di sensori: progettazione e realizzazione di una piattaforma sperimentale*. PhD thesis, Università degli Studi di Padova Dipartimento di Ingegneria dell'Informazione, 2006.

[22] L. De Nardis and M.-G. Di Benedetto. Overview of the ieee 802.15.4/4a standards for low data rate wireless personal data networks. *Positioning, Navigation and Communication, 2007. WPNC '07. 4th Workshop on*, I:285–289, March 2007.

[23] H.M. Deitel and P.J. Deitel. *C. Corso completo di programmazione*. Apogeo Editore, 2007.

[24] Doxygen. *avr-libc Reference Manual 1.6.1*, 12 2007.

[25] Adam Dunkels, Bj? Gr?nvall, and Thiemo Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. *Local Computer Networks, Annual IEEE Conference on*, 0:455–462, 2004.

[26] A. Elgamal, J.P. Conte, L. Yan, M. Fraser, S.F. Masri, M. El Zarki, T. Fountain, and M. Trivedi. A Framework for Monitoring Bridges and Civil Infrastructure. In *Proceedings of 3rd China-Japan-US Symposium on Structural Health Monitoring and Control*, 2004.

[27] Jeremy Elson and Deborah Estrin. Time synchronization for wireless sensor networks. *Parallel and Distributed Processing Symposium, International*, 3:30186b, 2001.

[28] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163, 2002.

[29] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163, 2002.

[30] P. Eronen. IKEv2 Mobility and Multihoming Protocol (MOBIKE). RFC 4555 (Proposed Standard), June 2006.

[31] D. Estrin, A. Sayeed, and M. Srivastava. Mobicom 2002 tutorialwireless sensor networks. *Atlanta, Georgia, USA*, 2002.

[32] Anand Eswaran, Anthony Rowe, and Raj Rajkumar. Nano-rk: An energy-aware resource-centric rtos for sensor networks. *Real-Time Systems Symposium, IEEE International*, 0:256–265, 2005.

[33] P. Ferrari, A. Flammini, D. Marioli, and A. Taroni. Ieee802.11 sensor networking. *Instrumentation and Measurement, IEEE Transactions on*, 55(2):615–619, April 2006.

[34] C. Fetzer and F. Cristian. An optimal internal clock synchronization algorithm. *Computer Assurance, 1995. COMPASS '95. 'Systems Integrity, Software Safety and Process Security'. Proceedings of the Tenth Annual Conference on*, pages 187–196, Jun 1995.

[35] M. Galeev. Home Networking with ZigBee. *EMBEDDED SYSTEMS PROGRAMMING*, 17(5):26–31, 2004.

[36] S. Ganeriwal, R. Kumar, and M.B. Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 138–149. ACM New York, NY, USA, 2003.

[37] Riccardo Gusella and Stefano Zatti. An election algorithm for a distributed clock synchronization program. Technical Report UCB/CSD-86-275, EECS Department, University of California, Berkeley, Dec 1985.

[38] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. *SIGPLAN Not.*, 35(11):93–104, 2000.

[39] R. Jurdak and SpringerLink (Online service. *Wireless Ad Hoc and Sensor Networks: A Cross-layer Design Perspective*. Springer Science+ Business Media, 2007.

[40] B.W. Kernighan, V. Marra, and D.M. Ritchie. *Il linguaggio C*. Pearson, 2004.

[41] S. Kim. Structure Monitoring using Wireless Sensor Networks. *CS294-1 Deeply Embedded Network Systems class project*, 2003.

[42] H. Kopetz, A. Kruger, D. Millinger, and A. Schedl. A synchronization strategy for a time-triggered multicluster real-time system. *Reliable Distributed Systems, IEEE Symposium on*, 0:154, 1995.

[43] Bhaskar Krishnamachari. *Networking Wireless Sensors*. Cambridge University Press, 2006.

[44] N. Kurata, B.F. Spencer, and M. Ruiz-Sandoval. Risk monitoring of buildings with wireless sensor networks. *Structural Control and Health Monitoring*, 12(3):315–327, 2005.

[45] N. Kushalnagar, G. Montenegro, and C. Schumacher. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919 (Informational), August 2007.

[46] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[47] Stefano Landi. *Progetto di servizi resource-aware per reti ad-hoc*. PhD thesis, Università degli studi di Bologna FACOLTA' DI INGEGNERIA, 2004.

[48] F.L. Lewis. Wireless Sensor Networks. *Smart Environments: Technologies, Protocols, and Applications*, 2003.

[49] Libelium. Squidbee, 2008.

[50] J.P. Lynch, Y. Wang, K.H. Law, J.H. Yi, C.G. Lee, and C.B. Yun. Validation of a large-scale wireless structural monitoring system on the Geumdang Bridge. In *Proceedings of 9th International Conference on Structural Safety and Reliability*, 2005.

[51] Nitaigour P. Mahalik. *Sensor Networks and Configuration: Fundamentals, Standards, Platforms, and Applications*. Springer, 2006.

[52] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 39–49, New York, NY, USA, 2004. ACM.

[53] MaxStream, Inc. *XBeeTM Series 2 OEM RF Modules*, 06 2007.

[54] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944 (Proposed Standard), September 2007.

[55] M. Owens. *The Definitive Guide to SQLite*. Apress, 2006.

[56] H. Park, J. Friedman, M.B. Srivastava, and J. Burke. A new light sensing module for mica motes. *Sensors, 2005 IEEE*, pages 4 pp.–, Oct.-3 Nov. 2005.

[57] GIORGIO PORCU. *6LoWPAN: un livello adattivo per la convergenza di Wireless Sensor Network e IPv6. Teoria, analisi, sviluppo e confronto di soluzioni low-power*. PhD thesis, UNIVERSITA' DEGLI STUDI DI PISA Facoltà di Scienze Matematiche Fisiche e Naturali, 2008.

[58] Nithya Ramanathan, Kevin Chang, Rahul Kapur, Lewis Girod, Eddie Kohler, and Deborah Estrin. Sympathy for the sensor network debugger. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 255–267, New York, NY, USA, 2005. ACM.

[59] K. Romer. *Time Synchronization and Localization in Sensor Networks.* PhD thesis, Swiss Federal Institute of Technology Zurich, 2005.

[60] L.B. Ruiz, J.M. Nogueira, and A.A.F. Loureiro. Manna: a management architecture for wireless sensor networks. *Communications Magazine, IEEE*, 41(2):116–125, Feb 2003.

[61] A.K. Salkintzis, C. Fors, and R. Pazhyannur. Wlan-gprs integration for next-generation mobile data networks. *Wireless Communications, IEEE*, 9(5):112–124, Oct. 2002.

[62] W. Simpson. The Point-to-Point Protocol (PPP). RFC 1661 (Standard), July 1994. Updated by RFC 2153.

[63] K. Siwiak. Ultra-wide band radio: introducing a new technology. *Vehicular Technology Conference, 2001. VTC 2001 Spring. IEEE VTS 53rd*, 2:1088–1093 vol.2, 2001.

[64] J.A. Stankovic. Wireless Sensor Networks, 2006.

[65] G. Tolle and D. Culler. Design of an application-cooperative management system for wireless sensor networks. *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pages 121–132, Jan.-2 Feb. 2005.

[66] P. Volgyesi and A. Ledeczi. Component-based development of networked embedded applications. *Euromicro Conference, 2002. Proceedings. 28th*, pages 68–73, 2002.

[67] Christian Winkler and Hartmut M. Hofmann. Determination of bound-state wave functions by a genetic algorithm. *Phys. Rev. C*, 55(2):684–687, Feb 1997.

[68] Dawei Xia and Natalija Vlajic. Near-optimal node clustering in wireless sensor networks for environment monitoring. *Advanced Information Networking and Applications, International Conference on*, 0:632–641, 2007.

[69] Wei Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *Networking, IEEE/ACM Transactions on*, 12(3):493–506, June 2004.

[70] Zude Zhou and Sheng Wang. An approach to localized edge detection in shm. *Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on*, 2:1211–1213, Sept. 2005.