Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Localizzazione in reti di sensori wireless: un algoritmo per la stima delle distanze

I. Infantino, G. La Tona, R. Rizzo, P. Storniolo, A. Urso

Rapporto Tecnico N.:
RT-ICAR-PA-06-07 **ottobre 2006**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Cosenza, Via P. Bucci 41C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sede di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.na.icar.cnr.it*
– Sede di Palermo, Viale delle Scienze, 90128 Palermo, URL: *www.pa.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Localizzazione in reti di sensori wireless: un algoritmo per la stima delle distanze

I. Infantino[1], G. La Tona[2], R. Rizzo[1], P. Storniolo[1], A. Urso[1]

Rapporto Tecnico N.: **RT-ICAR-PA-06-07**

Data: **ottobre 2006**

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Palermo Viale delle Scienze edificio 11 90128 Palermo
[2] Università degli Studi di Palermo Dipartimento di Ingegneria Informatica Viale delle Scienze 90128 Palermo

***Abstract*—In questo Rapporto Tecnico viene presentato un algoritmo per la stima di distanze in reti di sensori wireless. A tale scopo viene misurata la differenza dei tempi di propagazione di un segnale acustico udibile e un segnale radio. Viene presentata l'implementazione di tale algoritmo per i sensori MicaZ e la piattaforma TinyOS. Si discutono sia le fonti di errore legate all'hardware utilizzato che quelle intrinseche al metodo. Vengono inoltre presentati i risultati sperimentali ottenuti con l'applicazione di tale algoritmo.**

***Parole chiave*—Reti di sensori, localizzazione, TinyOS, Micaz.**

I. Introduzione

La ricerca nel campo delle reti di sensori wireless ha subito una crescita esponenziale negli ultimi anni. Esse hanno trovato utilizzo in diverse applicazioni ad esempio nel monitoraggio ambientale o nel tracciamento di target all'interno di un ambiente.

Il vantaggio più rilevante delle reti di sensori wireless consiste nella possibilità di realizzare reti di centinaia di nodi in grado di connettersi e configurarsi in maniera autonoma. Rispetto alle reti cablate i costi di installazione sono notevolmente inferiori, inoltre la rete può essere estesa in qualsiasi momento semplicemente aggiungendo più dispositivi.

Oltre a ridurre drasticamente i costi di installazione le reti di sensori wireless presentano il vantaggio di adattarsi in maniera dinamica ai cambiamenti dell'ambiente. I meccanismi di adattamento possono ad esempio rispondere a cambiamenti della topologia della rete. A differenza di reti infrastrutturate, esse non necessitano di dispositivi di livello superiore che gestiscano la rete, né hanno la necessità di comunicare soltanto con un qualche punto di accesso. Una rete di questo tipo in cui ogni nodo può essere connesso con i vicini, consente l'implementazione di protocolli di routing in cui ogni nodo può comunicare con qualsiasi altro nodo della rete, realizzando una rete stabile, flessibile e tollerante ai guasti.

La possibilità di conoscere la posizione di determinati target fornisce un valore aggiunto a particolari applicazioni delle reti di sensori wireless, nelle quali la conoscenza di informazioni sulla posizione di ciascun nodo può essere utilizzata per il tracciamento di oggetti di interesse o per associare un riferimento di posizione alle informazioni sensoriali raccolte. Inoltre, le informazioni di posizione possono consentire di ottimizzare svariati aspetti dell'applicazione quali l'instradamento ottimale, la gestione delle congestioni, l'auto organizzazione della rete e la tolleranza ai guasti.

In questo lavoro viene presentato un algoritmo per la stima di distanze in reti di sensori wireless ai fini della localizzazione. Il metodo proposto usa la misura della differenza dei tempi di propagazione di un segnale radio e di un segnale acustico a frequenze udibili per stimare la distanza tra i nodi trasmettitore e ricevitore. L'algoritmo è implementato per sensori MicaZ utilizzando il sistema operativo TinyOS. Sono mostrati, infine, i risultati sperimentali ottenuti dall'applicazione di tale algoritmo.

La restante parte di questo documento è strutturata come segue. Nella sezione II sono presentate le principali tecniche di localizzazione conosciute, nella sezione III sono presentati i sensori MicaZ e il sistema TinyOS, nella sezione IV è descritto il metodo per la stima di distanze utilizzato dal sistema, ed infine nella sezione V è descritta l'implementazione del sistema e vengono esposti i risultati sperimentali ottenuti.

II. Tecniche di localizzazione

Le tecniche di localizzazione attualmente disponibili differiscono per accuratezza, area di copertura, costi di installazione e costi di gestione.

Una delle tecniche più conosciute e diffuse di localizzazione è quella che si basa sul segnale del sistema GPS(*Global Positioning System*)[1]. Il GPS usa la misura del tempo di propagazione di un segnale radio e permette di ottenere una stima precisa della posizione a partire dal confronto dei tempi di propagazione di segnali trasmessi da fonti diverse. Le fonti di trasmissione sono dei satelliti che orbitano attorno alla terra alla quota di circa 20.200 Km, dotati di orologi al cesio di grandissima precisione sincronizzati tra di loro, che continuamente trasmettono messaggi radio contenenti l'orario di trasmissione e la propria posizione. Un ricevitore che sia in grado di ricevere ad un dato istante i messaggi da almeno tre satelliti può risalire ai tempi di propagazione dei segnali, calcolare la distanza dai satelliti e ricavare la propria posizione relativa ai satelliti, quindi conoscendo anche la posizione dei satelliti può risalire alla posizione assoluta. Questa tecnica però è applicabile soltanto all'aperto, in quanto il segnale inviato dal satellite non riesce a penetrare ostacoli quali le pareti.

Altre tecniche di localizzazione quali RADAR(*Radio Detection And Ranging*), LASER(*Light Amplification by Simulated Emission of Radiation*) e SONAR(*Sound Navigation And Ranging*) sono largamente utilizzate in applicazioni commerciali e industriali, ma hanno bisogno di dispositivi specificamente progettati o si basano su specifiche caratteristiche dell'ambiente in cui il sistema viene installato. I sensori wireless per motivi di costo, potenza impegnata o semplicità non possono usare le tecniche di localizzazione appena descritte e devono quindi affidarsi ad altre soluzioni.

Le tecniche di localizzazione appositamente sviluppate per reti di sensori senza fili possono essere raggruppate in due categorie, possono essere: basate sulla stima delle distanze o meno.

Le tecniche che non si basano sulla stima delle distanze partono dall'assunzione che le misurazioni possono essere imprecise. Alcune di queste tecniche si basano sulla connettività radio dei nodi[3] e stimano la posizione basandosi implicitamente sul conteggio dei salti di un pacchetto radio[4][5]. Sempre basandosi sugli stessi principi, in [6] è presentato un metodo per migliorare la stima di posizione anche in reti in cui i nodi siano stati posizionati in maniera casuale attraverso la realizzazione di un accurato sistema di coordinate locali allineato con le coordinate globali.

Altre tecniche misurano la potenza del segnale ricevuto leggendo il valore dell'RSSI(*Received Signal Strength Indicator*), ma non cercano di risalire a misure di distanza,

bensì in una fase preliminare effettuano una serie di misure in posizioni differenti di tutto l'ambiente in cui il sistema di localizzazione verrà eseguito, con ricevitori e trasmettitori differenti. In questo modo viene costituita una cosiddetta *wireless map* costituita dalle associazioni tra le varie posizioni dell'ambiente e le letture dell'RSSI effettuate. Nella seconda fase il nodo da localizzare effettua una serie di misure di RSSI da diverse fonti, e a partire da queste misure e dalla wireless map risale alla propria posizione. I metodi utilizzati per risalire alla posizione possono essere deterministici o probabilistici. Un metodo di tipo deterministico viene implementato nel sistema RADAR[7] sviluppato dalla Microsoft, nel quale una volta ottenute delle letture di RSSI, queste vengono confrontate con i dati raccolti nella wireless map e si sceglie la posizione per cui le misurazioni precedentemente effettuate hanno la minima distanza dalle misurazioni attuali, avendo naturalmente definito una particolare metrica. Metodi che invece usano un approccio probabilistico sono descritti in [8][9][10][12], nei quali viene costruita una distribuzione di probabilità per la posizione del nodo da localizzare, cercando sempre di mantenere un bilanciamento tra precisione e sovraccarico computazionale.

Le tecniche che si basano sulla stima delle distanze in reti di sensori a partire appunto da questa, cercano di risalire alla posizione triangolando le distanze del nodo da localizzare da altri nodi di cui la posizione è conosciuta, quanto più è precisa la stima della distanza tanto più semplice è l'algoritmo di localizzazione.

Una tecnica particolarmente interessante è stata sviluppata in [11], in cui è presentata una implementazione semplificata di interferometria ottenendo elevate precisioni. Tale tecnica non richiede hardware aggiuntivo, ma non può essere realizzata con tutti i dispositivi di comunicazione in quanto dipende fortemente dai meccanismi di modulazione utilizzati e dalle libertà lasciate agli utilizzatori.

Tipicamente per stimare la distanza tra due nodi vengono utilizzati metodi basati sulla misura della potenza ricevuta(RSSI), sulla misura del tempo di propagazione di un segnale e sulla differenza di propagazione di segnali di natura differente. Di seguito vengono discusse le caratteristiche e le problematiche di questi metodi.

A. Misura dell'RSSI

L'RSSI(*Received Signal Strength Indicator*) è un indicatore della potenza del segnale ricevuto, la corrispondenza con questa dipende dall'implementazione ed è specifica di ogni dispositivo e di ogni protocollo. Nei sensori a nostra disposizione il valore dell'RSSI è quantizzato usando 256 livelli. Il vantaggio dell'utilizzare l'RSSI sta nel fatto che non è necessario nessun dispositivo aggiuntivo basta infatti il dispositivo ricetrasmittente.

Tuttavia estrarre una stima della distanza del trasmettitore dal ricevitore a partire dall'RSSI e quindi dalla potenza del segnale ricevuto è tuttaltro che semplice, poiché non esiste una relazione deterministica tra distanza e potenza del segnale. Un segnale elettromagnetico infatti subisce fenomeni che ne attenuano la potenza quali ad esempio riflessione, rifrazione, diffrazione. Soprattutto in ambienti chiusi, in cui la densità di ostacoli è elevata, un segnale percorre cammini multipli per arrivare a destinazione ed è impossibile calcolare a priori in maniera deterministica la potenza del segnale ricevuto. Studi statistici riguardo le proprietà dell'RSSI sono stati presentati in [13], mentre in [14] viene presentata un'analisi metrologica di un dispositivo particolare, il Mica2, utile a comprendere le possibili problematiche di un dispositivo reale.

E' possibile determinare modelli probabilistici di propagazione del segnale tramite i quali è possibile stimare la probabilità che ad una data distanza il segnale venga ricevuto con una data potenza; un semplice modello per ambienti esterni è il seguente:

$$P_r(d) = P_r(d_0) - \eta 10\log(\frac{d}{d_0}) \quad (1)$$

in cui $P_r(d)$ rappresenta la potenza ricevuta alla distanza d, $P_r()$ la potenza la potenza ricevuta alla distanza d_0, η è un parametro che dipende dall'ambiente, i cui valori tipici vanno da 2 a 7, con valori più grandi in aree urbane e negli interni.

Modelli empirici più sofisticati quali quello di Okumura-Hata[29] o di Lee[30] consentono maggiori precisioni ma che comunque non permettono di ottenere stime con errori inferiori al metro in ambienti chiusi.

Per questi motivi, benché la misura dell'RSSI non necessiti l'utilizzo di dispositivi aggiuntivi, l'RSSI non fornisce un metodo affidabile negli interni quando si è interessati a misure con errori dell'ordine dei centimetri.

B. Misura del tempo di propagazione

Stimare una distanza tramite la misura del tempo di propagazione di un segnale è un'operazione teoricamente semplice conoscendo la velocità con cui il segnale si propaga; purtroppo nella pratica ci sono problemi che rendono meno semplice l'utilizzo di questo metodo.

Innanzi tutto è necessario che i dispositivi coinvolti nella localizzazione siano sincronizzati tra loro per potere misurare il tempo di propagazione del segnale. Se si è interessati alla sincronizzazione dei nodi solo per la localizzazione, allora basta sincronizzare i nodi per il periodo in cui questa viene eseguita, altrimenti si può procedere a una procedura di sincronizzazione globale continuamente eseguita nella rete. Uno dei protocolli sviluppati per la sincronizzazioni in reti più diffuso è NTP(*Network Time Synchronization*)[23], sviluppato appositamente per la rete Internet, non adatto quindi per applicazioni in reti di sensori. Protocolli sviluppati per reti di sensori sono RBS(*Reference Broadcast Synchronization*)[24], TPSN(*Timing-sync Protocol for Sensor Networks*)[25] e FTSP(*Flooding Time Synchronization Protocol*)[26]. Tutti i protocolli elencati si basano sulla lettura dei tempi di invio e di ricezione registrati dal livello MAC e ignorano i ritardi dovuti ai tempi di propagazione dei segnali radio. A causa della imprecisione della frequenza dei clock dei vari nodi, non è sufficiente sincronizzare la rete una sola volta per mantenerla sincronizzata, bensì è necessario ripetere il procedimento con regolarità.

La sincronizzazione risulta critica per la stima dei tempi di propagazione, in quanto un errore nella sincronizzazione, ad esempio dell'ordine dei millisecondi, può vanificare la localizzazione stessa.

Tipicamente i segnali utilizzati sono o acustici o

elettromagnetici, nel secondo caso quando viene inviato un pacchetto viene aggiunta l'ora di invio, quando questo viene ricevuto poiché i nodi sono sincronizzati basta sottrarre l'orario di ricezione a quello di invio per ottenere il tempo di propagazione (il meccanismo utilizzato in GPS). Per quanto riguarda i segnali acustici questi possono essere segnali a frequenze udibili dall'orecchio umano o ultrasuoni, tuttavia il meccanismo di base è identico. Poiché il segnale audio non può trasportare informazioni è necessario inviare, contemporaneamente all'emissione di questo, un pacchetto via radio riportante l'orario di emissione, ricevuto il segnale audio basterà sottrarre l'orario di ricezione con quello di invio per ottenere il tempo di propagazione.

L'utilizzo degli ultrasuoni consente di ottenere precisioni elevate ma necessità di hardware dedicato per l'emissione e la ricezione di tali segnali, inoltre tali trasmettitori e ricevitori sono altamente direzionali, infatti i dispositivi hanno un cono di trasmissione di 120° al di fuori del quale non è affatto possibile ottenere una stima di distanza. A differenza di quanto detto, sopra per l'audio a frequenze udibili è sufficiente hardware economico e a bassi consumi energetici che non soffre così pesantemente dei problemi di direzionalità. Tali segnali però non consentono precisioni altrettanto elevate, si ottengono infatti errori anche del 30% sulla stima della distanza dovuti al rumore ambientale, alle riflessioni e al volume del segnale ricevuto, che si attenua in maniera proporzionale al logaritmo della distanza.

In [15] viene presentato *Calamari* un sistema di localizzazione appositamente pensato per reti di sensori che sfrutta le misure dei tempi di propagazione di segnali acustici unitamente a misure dell'RSSI.

C. Misura della differenza dei tempi di propagazione di segnali di natura differente

Un metodo per superare le problematiche legate alla sincronizzazione è quello di utilizzare contemporaneamente due segnali di natura differente, quindi con velocità di propagazione differenti, e misurare la differenza dei tempi di propagazione. Siano infatti v_1 e v_2 le velocità dei due segnali, supposto $v_1 > v_2$ il nodo ricevente percepito il primo segnale avvierà un timer che fermerà quando avrà ricevuto il secondo segnale, misurando in questo modo la differenza dei tempi di propagazione. Risalire alla distanza a partire da tale intervallo di tempo è teoricamente molto semplice, infatti:

$$d = \Delta t \left(\frac{v_1 v_2}{v_1 - v_2} \right) \qquad (2)$$

dove Δt è l'intervallo misurato.

L'hardware necessario per questo metodo dipende dai segnali scelti, se uno di questi è un segnale radio, non è richiesto alcun hardware aggiuntivo, per segnali acustici valgono naturalmente le stesse considerazioni fatte nel paragrafo precedente, e naturalmente anche le considerazioni relative agli errori di misura.

Lavori precedenti che utilizzino questo metodo sono ad esempio Cricket[2] sviluppato dal MIT che utilizza un segnale radio e un segnale ad ultrasuoni, con hardware appositamente progettato, tale sistema permette di ottenere errori di pochi centimetri.

III. TinyOS e MicaZ

Il sistema implementato è stato realizzato utilizzando come nodi della rete i sensori **MicaZ**(MPR2400) prodotti dalla Crossbow[17] e come punto di accesso uno di questi stessi sensori collegato alla scheda di programmazione **MIB600**, sempre prodotta dalla Crossbow, a sua volta connessa tramite LAN a un computer.

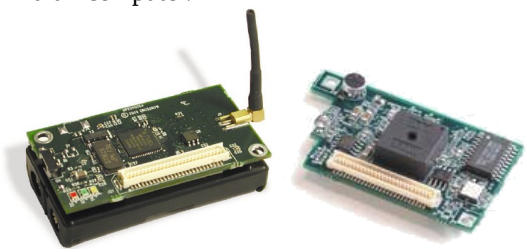

Figura 1. A sinistra MicaZ (MPR2400), a destra la sensorboard MTS310CA.

A. MicaZ

I sensori MicaZ (Figura 1) sono dotati di un microcontrollore a 8 bit, l'ATmega128 prodotto dalla Atmel, a questo viene connesso un chip ricetrasmettitore che rispetta lo standard IEEE 802.15.4. Attraverso un connettore a 51 pin è possibile utilizzare una sensorboard, in particolare è stata utilizzata la MTS310CA. La sensorboard MTS310CA prodotta dalla Crossbow è dotata di un rilevatore di luminosità e di temperatura, di un accelerometro, di un magnetometro. Inoltre la sensorboard è dotata di un *buzzer* Ario S14T40A che emette un suono a 4 KHz, di un microfono WM-62A della Panasonic la cui uscita viene amplificata e poi filtrata attraverso un filtro passa-banda a sua volta connesso con un decodificatore di toni a bassa potenza LM567 prodotto da National Semiconductor, il quale riconosce toni a 4KHz, cioè quelli che possono essere emessi dal buzzer.

Le limitazioni di questi sensori sono legate alla bassa potenza di calcolo e ad una bassa capacità di memorizzazione.

I sensori MicaZ sono dotati del chip di comunicazione CC2420 prodotto dalla Chipcon[18], questo implementa lo standard 802.15.4[16], un protocollo di comunicazione a bassi consumi energetici pensato per la creazione delle cosiddette **PAN**(*Personal Area Network*). Questo chip permette di realizzare comunicazioni in maniera robusta e affidabile utilizzando la banda ISM(*Industrial Scientific Medic*) intorno a 2.4 GHz per utilizzare la quale non è necessaria alcuna licenza. Inoltre il CC2420 fornisce supporto hardware per la gestione dei pacchetti, la memorizzazione temporanea dei dati, la misura dell'RSSI, la crittografia e l'autenticazione.

Lo standard 802.15.4 specifica quelli che sono lo strato fisico e lo strato MAC della pila ISO/OSI. Per quanto riguarda lo strato fisico viene impiegata la tecnica DSSS(*Direct Sequenze Spread Spectrum*) per evitare le interferenze tipiche di un canale condiviso e una velocità di trasmissione di 250 KBit/s. Viene anche definito il formato di un pacchetto per lo strato fisico (Figura 2), per ogni pacchetto viene trasmesso: un preambolo di 4 byte che serve per la sincronizzazione di trasmettitore e ricevitore, un byte chiamato SFD(*Start Frame Delimiter*) che indica l'inizio di un pacchetto, un campo di 7 bit che indica la lunghezza di tutto il pacchetto esclusi soltanto SFD e preambolo, ed infine un campo di lunghezza variabile contenente il pacchetto dello strato superiore.

Octets: 4	1	1		variable
Preamble	SFD	Frame length (7 bits)	Reserved (1 bit)	PSDU
SHR		PHR		PHY payload

Figura 2. Formato di un pacchetto per lo strato fisico.

Lo strato MAC definisce in particolare il protocollo di accesso al mezzo che è di tipo CSMA/CA(*Carrier Sense Multiple Access/Collision Avoidance*).

Il chip CC2420 implementa lo strato fisico del protocollo 802.15.4 e fornisce il supporto necessario allo sviluppo dello strato MAC, che deve essere implementato via software.

I sensori MicaZ sono inoltre di un'interfaccia UART(*Universal Asynchrous Receiver-Transmitter*) che consente la connessione seriale con dispositivi esterni. In particolare questa interfaccia viene utilizzata quando i sensori vengono collegati alla scheda di programmazione. La scheda MIB600 connette l'interfaccia UART del sensore ad un adattatore di rete ethernet, è quindi possibile connettersi con il sensore attraverso una LAN instaurando una connessione TCP. Esistono anche altri tipi di schede di programmazione che espongono interfacce di connessioni differenti, ad esempio MIB510(porta seriale) e MIB520 (USB). Utilizzando una scheda di programmazione è quindi possibile connettersi da un computer ad un sensore e quindi alla rete intera, il sensore montato sulla scheda di programmazione viene denominato *stazione base*.

Collegando un sensore alla scheda di programmazione si può trasferire e installare su questo un programma, questo processo viene chiamato programmazione del dispositivo.

B. TinyOS

Nel presente studio è stato usato il sistema operativo open source **TinyOS** [19][22], nella versione 1.1.10, ampiamente diffuso per scopi di ricerca nella comunità scentifica. Questo è caratterizzato da un'architettura basata su componenti in maniera da permettere una rapida innovazione e implementazione, mantenendo allo stesso tempo contenute le dimensioni del codice, così come richiesto dalle restrizioni di memoria tipiche delle reti di sensori. Il modello di esecuzione di TinyOS è basato sugli eventi, permettendo in questo modo un maggiore risparmio energetico, ma consente allo stesso tempo un'accodamento di operazioni chiamate **task,** che vengono eseguite in maniera asincrona.

Il sistema TinyOS, le librerie e le applicazioni sono scritte in nesC(*Network Embedded System C*)[21], versione 1.1, un linguaggio derivato dal C appositamente pensato per la programmazione di applicazioni strutturate basate su componenti. Il linguaggio nesC è progettato per sistemi *embedded*, quali reti di sensori. nesC ha una sintassi simile al C, ma supporta il modello di concorrenza di TinyOS, così come meccanismi per strutturare, nominare e collegare insieme componenti software per realizzare robusti sistemi di rete integrati. L'obbiettivo principale di nesC è quello di consentire ai progettisti di realizzare componenti che possano essere facilmente composti in sistemi completi e concorrenti. Una applicazione nesC consiste in uno o più componenti, un *componente* può *usare* e *definire* una o più *interfacce*. Un'interfaccia dichiara uno o più *comandi* e uno o più *eventi*. Le interfacce sono bidirezionali e sono l'unico punto di accesso ai componenti che le implementano. Un comando è una funzione che il componente che implementa l'interfaccia in cui è definito deve implementare. Un evento è una funzione che invece deve essere implementata dal componente che usa l'interfaccia in cui questi è definito. Quindi un componente che voglia chiamare un comando di un'interfaccia deve anche implementare gli eventi di tale interfaccia.

Ci sono due tipi di componenti: i *moduli* e le *configurazioni*. I moduli forniscono il codice applicativo implementando le interfacce. Le configurazioni vengono utilizzate per assemblare insieme altri componenti, collegando le interfacce usate da alcuni componenti alle interfacce fornite da altri componenti. Ogni applicazione scritta in nesC contiene una configurazione di alto livello che collega tutti i componenti dell'applicazione.

Il modello di concorrenza di TinyOS prevede due tipi di percorsi di esecuzione: i *task* e i *gestori degli eventi hardware*. I task sono funzioni la cui esecuzione viene differita, cioè non viene eseguita immediatamente come avviene con una normale chiamata a funzione, i task vengono chiamati attraverso la parola chiave **post**. Una volta schedulati i task vengono eseguiti per intero senza possibilità di prelazione da parte di altri task. Il codice eseguito all'interno di task viene definito sincrono. I gestori degli eventi hardware vengono eseguiti in maniera asincrona, possono cioè avere diritto di prelazione sui task e anche su altri gestori. I gestori di eventi hardware devono essere dichiarati con la parola chiave **async** e vengono richiamati direttamente dai gestori degli interrupt. La possibilità di eseguire codice in maniera asincrona può portare alle cosiddette *race conditions* cioè situazioni che portano ad un accesso incoerente ai dati condivisi, queste in nesC vengono evitate attraverso un esteso controllo a tempo di compilazione.

Caratteristica saliente di TinyOS è il modello di comunicazione impiegato basato su *Active Message*. Active Message[27] è un paradigma semplice ed estensibile per la comunicazione basato sui messaggi largamente utilizzato nei sistemi di elaborazione paralleli e distribuiti. Ogni messaggio contiene il nome di un gestore di livello applicazione da richiamare e un insieme di dati da passare a questo. Come discusso in [28], un paradigma di questo tipo è particolarmente adatto per le reti di sensori per la semplicità di gestione e per essere intrinsecamente asincrono e basato sugli eventi. Quando a basso livello viene ricevuto un pacchetto, viene letto il tipo di active message e viene quindi richiamato l'evento corretto per quel tipo di active message. Un tale modello di comunicazione è senza stato e senza connessione, cioè non è necessario instaurare una connessione con un nodo prima di poter comunicare con questo, semplicemente basta inviare, ogni pacchetto ricevuto viene confermato tramite il meccanismo di acknowledgment dello strato MAC.

Insieme a TinyOS viene fornita una serie di utili strumenti, tra i quali in particolare delle API java per interagire con la rete, ed uno strumento chiamato **MIG**(*Message Interface Generator*). MIG è uno strumento in grado di generare classi Java corrispondenti ai tipi di messaggio dichiarati all'interno di file di intestazione. MIG legge la definizione in nesC delle strutture(quelle del linguaggio C) che rappresentano un tipo

di messaggio e genera una classe Java per ogni tipo di messaggio, semplificando in questo modo la ricezione e l'invio di pacchetti all'interno di applicazioni Java.

Le API Java fornite sono contenute all'interno del package `net.tinyos`. In particolare il package `net.tinyos.message` contiene le classi che permettono l'invio e la ricezione di pacchetti dalla rete, cioè dalla stazione base, in maniera semplice. Fra tutte spicca la classe *Message* dalla quale ereditano tutte le classi generate da MIG e che rappresenta l'astrazione di un messaggio. La classe *MoteIF* fornisce le funzionalità di invio e ricezione di messaggi. Per l'invio di un pacchetto basta chiamare il metodo *send* di MoteIF passando come argomento un riferimento di tipo Message e l'indirizzo del destinatario. La ricezione è più complessa in quanto è necessario definire una classe che implementi l'interfaccia *MessageListener* e passare un riferimento a tale classe e un riferimento ad un oggetto che rappresenti il tipo di messaggio che si vuole ricevere al metodo *registerListener* di MoteIF. Alla ricezione di ogni messaggio del tipo richiesto verrà richiamato il metodo *messageReceived* di MessageListener.

La connessione con la stazione base viene astratta per rendere tutto il resto indipendente, in quanto a questa ci si può connettere tramite porta seriale o tramite ethernet, o ci si può anche connettere ad un simulatore. A questo scopo vengono utilizzate le classi del package `net.tinyos.packet` nel quale si astrae la connessione con la stazione base attraverso la classe *PacketSource* e le classi che ereditano da questa. Quando si vuole quindi instaurare una connessione viene passato un riferimento di tipo PacketSource al costruttore della classe MoteIF.

Infine viene fornita un'applicazione chiamata *SerialForwarder* contenuta nel package `net.tinyos.sf`. Si tratta di un server multi-thread che si connette ad una stazione base e si mette in ascolto accettando connessioni tcp, per i client connessi SerialForwarder si sostituisce alla stazione base, quindi i pacchetti che questo riceve dalla rete vengono reinoltrati a tutti i client ad esso connessi, allo stesso modo i client possono inviare pacchetti a SerialForwarder che verranno da questo reinoltrati verso la rete. L'utilità principale di questa applicazione è che accetta più connessioni contemporanee a differenza di quanto avviene quando ci si connette direttamente alla stazione base.

IV. Stima della differenza dei tempi di propagazione di un segnale radio e un segnale sonoro

Il metodo sviluppato in questo lavoro implementa la stima della distanza attraverso la misura della differenza dei tempi di propagazione di un segnale radio e un segnale acustico udibile.

Per stimare una distanza un nodo viene scelto per emettere i due segnali e altri nodi vengono scelti per ricevere i due segnali.

Teoricamente, poiché si vuole stimare i tempi di propagazione, bisognerebbe misurare l'intervallo di tempo che intercorre tra l'inizio della ricezione dei due segnali, ma avendo a che fare con dispositivi non realizzati appositamente per questo scopo ciò è impossibile.

L'emissione del suono viene effettuata tramite il *buzzer* che emette un suono a 4 KHz. Il suono viene ricevuto dal microfono la cui uscita viene amplificata e poi filtrata attraverso un filtro passa-banda a sua volta connesso con il decodificatore di toni. Quando il tone decoder riconosce un tono viene generato un interrupt del microcontrollore.

Per quanto riguarda il segnale radio questo viene generato trasmettendo un pacchetto attraverso il chip CC2420. Quando in ricezione il chip CC2420 riconosce l'SFD(*Start Frame Delimiter*) viene segnalato un interrupt del microcontrollore ad indicare l'inizio della ricezione di un pacchetto, in questo modo possiamo rilevare l'arrivo del segnale radio. Questo metodo porta a delle imprecisioni in quanto, dato il formato di un pacchetto dello strato fisico 802.15.4 [16], prima che venga trasmesso l'SFD viene trasmessa una sequenza di sincronizzazione necessaria per la demodulazione, quindi la ricezione dell'SFD non porta ad una corretta stima del tempo di propagazione del segnale radio a causa dei tempi di trasmissione della sequenza di sincronizzazione e dell'SFD stesso, bisogna però notare che questi ritardi sono costanti.

V. Implementazione e risultati sperimentali

A. Architettura del sistema

Il sistema implementato si divide tra il sottosistema in esecuzione sui sensori e il sottosistema in esecuzione sul computer che richiede le stime di distanza.

Il sottosistema in esecuzione sui sensori consiste principalmente nel modulo *LocalizationM* che implementa l'interfaccia *Localization*, una qualsiasi applicazione che intenda utilizzare questo sistema deve semplicemente usare questa interfaccia collegandola con il modulo definito. L'interfaccia *Localization* definisce il comando *startLocalization* e l'evento *responseDone*.

Altro componente importante dell'architettura è l'interfaccia *ProcessCmd* nella quale vengono definiti i comandi e gli eventi necessari per ricevere ed eseguire un comando remoto. Viene definito infatti il comando *execute* che accetta come parametro il messaggio ricevuto dalla rete, e l'evento *done* che segnala che l'azione è stata eseguita e indica se tale azione è andata a buon fine.

Quando in particolare viene riconosciuto un comando di localizzazione viene eseguito il comando *startLocalization* di *Localization* al quale vengono passati come argomenti i parametri che specificano le modalità di localizzazione richieste da remoto.

I comandi vengono inviati come pacchetti di tipo *CommandMsg*, il quale specifica l'azione da compiere e i parametri relativi a quella particolare azione(Figura 3).

Figura 3. Formato di un pacchetto di tipo CommandMsg, **src** indica il nodo sorgente, **action** indica l'azione da compiere, **args** indica parametri relativi all'azione indicata.

Ricevuto un pacchetto di comando che riguardi la localizzazione, questo viene passato come argomento al comando *startLocalization*. In particolare come argomento di

un pacchetto di comando di localizzazione viene specificato il nodo della rete designato per l'invio dei segnali.

Ogni sensore che misura la differenza dei tempi di propagazione dei segnali che riceve invia tale misura al computer attraverso un messaggio di tipo *LocalizationMsg* che indica la misura dell'intervallo di tempo misurato, ma anche l'RSSI misurato alla ricezione del segnale radio.

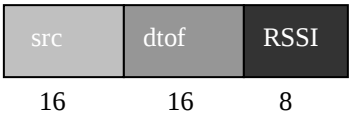

Figura 4. Formato di un pacchetto di tipo LocalizationMsg, **src** indica il nodo sorgente, **dtof** indica l'intervallo, **RSSI** l'Rssi misurato.

Per quanto riguarda il sottosistema che interroga la rete, questo è stato implementato in Java sfruttando le API messe a disposizione da TinyOS per l'interazione con la rete. E' stata implementata la classe *LocalizationMsgListener* la quale implementa l'interfaccia *MessageListener*. Tale classe è responsabile della ricezione e interpretazione dei messaggi ricevuti dalla rete.

B. Invio dei segnali

Un sensore, quando designato per inviare i due segnali, per prima cosa si occupa di bloccare tutti gli altri moduli in esecuzione. Vengono disattivati anche i componenti che gestiscono la comunicazione, per gestire ogni singolo dettaglio della trasmissione radio. Per potere trasmettere i due segnali in maniera sincrona è necessario bypassare il sistema operativo e accedere a basso livello al chip CC2420. Ciò è necessario per evitare l'uso del protocollo di accesso al mezzo CSMA/CA che comporta un ritardo impredicibile.

L'emissione del segnale audio è piuttosto semplice in quanto basta attivare l'alimentazione del buzzer.

Di seguito viene riportato lo pseudocodice della funzione responsabile dell'invio dei due segnali.

```
carica in memoria msg
carica msg sul buffer di
trasmissione del CC2420
alimenta buzzer
chiedi a CC2420 di iniziare a
trasmettere
aspetta che i dati siano stati
trasmessi
aspetta un tempo k necessario
perché il suono venga
riconosciuto
smetti di alimentare buzzer
```

Figura 5 Pseudo codice della funzione che invia i due segnali.

Naturalmente non è possibile eseguire le istruzioni per l'inizio dell'invio dei segnali in maniera assolutamente contemporanea, ma si può eseguire tali istruzioni in maniera sequenziale con un ritardo minimo.

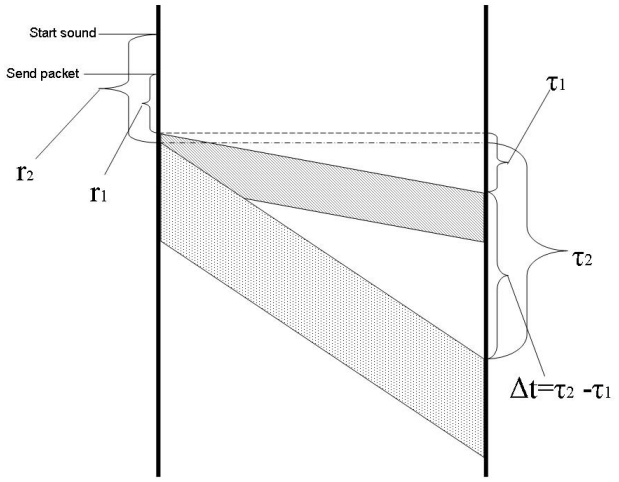

Figura 5. Diagramma temporale dell'invio dei segnali, si indicano graficamente oltre ai tempi di propagazione dei segnali anche il tempo di trasmissione del pacchetto radio e la durata dell'emissione del suono.

In Figura 5 viene riportato il diagramma temporale dell'invio dei due segnali, nella quale si indica con:

- τ_1 e τ_2 rispettivamente il tempo di propagazione del segnale radio e di quello acustico
- con r_1 il tempo che intercorre tra l'esecuzione del comando che avvia la trasmissione e l'inizio della trasmissione stessa, 196 μs
- con r_2 il tempo che intercorre tra l'esecuzione del comando che avvia l'emissione del suono e l'emissione stessa, tale intervallo non è conosciuto a priori

C. Ricezione dei segnali e stima della differenza dei tempi di propagazione

Quando un sensore è stato designato per ricevere i segnali, vengono fermati tutti i componenti non coinvolti nella procedura di localizzazione e in particolare i componenti responsabili della comunicazione, in questo modo così come per la trasmissione è possibile interfacciarsi a basso livello con l'hardware.

L'arrivo di un pacchetto, riconosciuto tramite la ricezione dell'SFD, viene segnalato tramite un interrupt che richiama in maniera asincrona l'evento che gestisce la ricezione di pacchetti.

La ricezione del tono audio viene riconosciuta tramite il decodificatore di toni, che genera un interrupt del microcontrollore.

Il microcontrollore ATmega128 [20] è dotato di 4 timer/contatori, i primi tre, da *Timer0* a *Timer2*, vengono utilizzati per vari scopi da TinyOS, mentre per *Timer3* non viene neanche implementato alcun componente di astrazione hardware. Il Timer3 è dotato di un unità di cattura dell'input la quale è collegata ad un pin esterno, quando su tale pin viene rilevata una transizione di stato viene conservato su un registro il valore attuale del contatore e viene generato un interrupt. In particolare sui sensori MicaZ a tale pin è connessa l'uscita del decodificatore di toni, che ha valore di uno logico(alto) quando il segnale in ingresso non è riconosciuto come tono a 4KHz e

```
azzera valore del contatore in
Timer3
avvia Timer3
imposta valore timeout
imposta transizione da catturare
in ingresso all'unità di cattura
di Timer3 come da alto a basso
abilita interrupt per la cattura
in input di Timer3
disabilita interrupt per la
ricezione di pacchetti
accendi led verde
```

Figura 7 Pseudo codice dell'evento richiamato all'arrivo del pacchetto radio.

va

```
leggi valore X del contatore
all'istante di rilevazione
della transizione
accendi led rosso
invia risultato a stazione base
```

Figura 8 Pseudo codice dell'evento eseguito alla ricezione del tono.

lore logico zero(basso) altrimenti. Quindi in questo modo quando un tono viene riconosciuto viene generato un interrupt.

La misura dell'intervallo di tempo che intercorre tra la ricezione dei due segnali viene effettuata azzerando il valore del contatore del Timer3, avviando tale timer e abilitando l'interrupt per la rilevazione di una variazione da alto a basso del valore di uscita del decodificatore di toni. Quando il tono viene decodificato viene letto il valore del contatore che quindi indica l'intervallo di tempo da misurare. La misurazione in questo modo si conclude e viene inviato un messaggio di tipo *LocalizationMsg* alla stazione base contenente la misura effettuata.

In Error: Reference source not found e Error: Reference source not found viene riportato lo pseudocodice degli eventi richiamati rispettivamente all'arrivo del pacchetto radio e del riconoscimento del tono.

D. Risultati

Prima di esporre i risultati sperimentali ottenuti, è necessario discutere le principali fonti di errore di questo metodo:

- poiché l'ambiente in cui si esegue l'applicazione è un interno, i segnali subiscono i fenomeni di rifrazione, riflessione e diffusione che soprattutto nel caso in cui i due nodi, trasmettitore e ricevitore, non abbiano visibilità diretta possono portare a ritardi consistenti non conoscibili a priori;
- per quanto riguarda il segnale acustico una rilevante fonte di ritardo è il rumore ambientale, considerando inoltre che all'aumentare della distanza tra i nodi la potenza con cui il segnale viene ricevuto diminuisce e quindi diminuisce anche il rapporto segnale rumore rendendo più difficile decodificare il tono in ingresso, tuttavia gli errori causati dal rumore ambientale possono essere modellati con una distribuzione gaussiana;
- il buzzer impiega un intervallo di tempo non conosciuto a priori prima di emettere il suono al massimo livello di potenza. Allo stesso modo il decodificatore di toni può non riconoscere il tono immediatamente, introducendo in questo modo un'altra fonte di ritardo non deterministica;
- dalle prove sperimentali effettuate è risultato che il decodificatore di toni integrato sulla sensorboard MTS310CA non sempre indica correttamente la presenza o l'assenza di toni alla frequenza indicata, portando quindi alla rilevazione di falsi positivi;
- buzzer e microfono possono avere leggere variazioni da dispositivo a dispositivo, portando quindi a intervalli misurati che variano tra le differenti coppie trasmettitore-ricevitore;
- la velocità del suono dipende dalla temperatura ambientale variando quindi nell'arco di funzionamento del sistema quindi porta a variazioni delle misure effettuate;

Gli esperimenti sono stati effettuati utilizzando due sensori, un trasmettitore ed un ricevitore, posizionati a diverse distanze gli uni dagli altri, all'interno di una stanza di dimensioni 7 m per 12 m. Al fine di calcolare l'andamento statistico delle variabili misurate sono state eseguite 60 misurazioni consecutive per ogni configurazione a partire da una distanza di 20 cm fino a 1 m con intervalli di 5 cm.

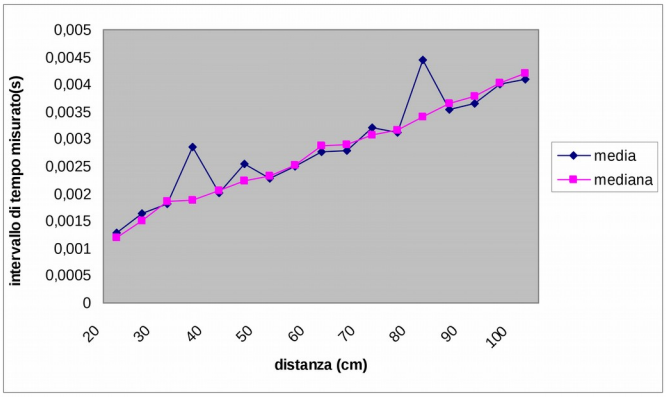

Figura 6 grafico della media e della mediana degli intervalli di tempo misurati per ciascuna configurazione.

In Figura 6 è riportato il grafico della media e della mediana degli intervalli di tempo misurati per ciascuna configurazione.

Dalle considerazioni effettuate riguardo le fonti di errore è possibile dedurre che l'intervallo di tempo misurato differisce dalla effettiva differenza dei tempi di propagazione di un fattore δ secondo la seguente:

$$\Lambda t' = \Delta t + \delta \qquad (3)$$

in cui si è indicato con $\Delta t'$ l'intervallo effettivamente misurato e con Δt l'intervallo previsto a partire dalla distanza utilizzando la (1). Il fattore delta per le considerazioni svolte non può essere stimato a priori, in quanto dipende principalmente dal ritardo di riconoscimento del

Tabella 1 Risultati delle prove effettuate.

dist effettiva (m)0,20,250,30,350,40,450,50,550,60,650,70,750,80,850,90,951**media**0,1980,3200,3750,7360,4440,6290,5400,6140,7010,7130,8520,8231,2740,9681,0051,1261,150**mediana**0,1050,5340,4090,4040,3440,2300,4580,6540,8280,6870,6870,5780,6211,0511,1331,2741,100**varianza**0,0010,0240,0571,0790,0370,0430,0460,0020,0150,0090,0120,0260,0530,0010,0060,0310,017**deviazione std**0,0230,1540,2381,0390,1920,2080,2150,0460,1230,0920,1080,1620,2310,0230,0770,1770,131

decodificatore di toni e dal ritardo di emissione del suono del buzzer.

Il parametro δ è stato stimato, a partire dai valori misurati, in funzione di $\Delta t'$ attraverso una regressione lineare con il metodo dei minimi quadrati. In particolare la regressione è stata effettuata a partire dai valori della mediana di $\delta = \Delta t' - \Delta t$ ad ogni distanza.

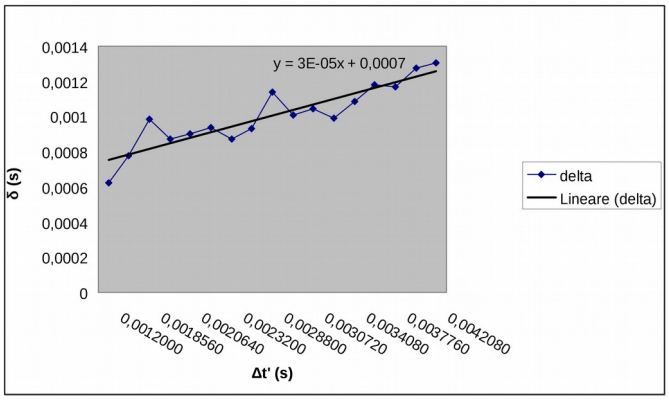

Figura 7 Grafico dei valori di δ in funzione di $\Delta t'$, si riporta inoltre la retta di regressione lineare di tali valori.

In Figura 7 è riportato il grafico dei valori di δ in funzione di $\Delta t'$, si riporta inoltre la retta di regressione lineare di tali valori.

L'algoritmo a partire dal valore $\Delta t'$ misurato e dal valore di δ corrispondente calcola la distanza come:

$$d = (\Delta t' - \delta)\frac{v_1 v_2}{v_1 - v_2} \qquad (4)$$

Nella Error: Reference source not found vengono riportati i risultati finali ottenuti applicando l'algoritmo. Dall'analisi dei risultati si può concludere che il metodo implementato risente delle limitazioni dell'hardware a disposizione, con errori non trascurabili dovuti alle caratteristiche del decodificatore di toni.

VI. Conclusioni

La tesi presenta un algoritmo per la stima di distanze in reti di sensori wireless che utilizza la misura della differenza dei tempi di propagazione di un segnale radio e di un segnale acustico. L'algoritmo è stato implementato per sensori MicaZ utilizzando TinyOS ed il linguaggio di programmazione nesC.

Tale algoritmo può essere utilizzato da un sistema di localizzazione che richieda come punto di partenza le stime di distanze tra i sensori.

I risultati sperimentali ottenuti hanno evidenziato le limitazioni dell'hardware dal punto di vista della localizzazione. Gli sviluppi futuri riguardano l'implementazione via software del decodificatore di toni, in maniera tale da ridurre gli errori legati ai ritardi di riconoscimento del decodificatore di toni hardware e poter aumentare il raggio di azione in cui è possibile effettuare le misure.

Riferimenti bibliografici

[1] Getting, "The Global Positioning System," *IEEE Spectrum*, December 1993, vol.30, pp. 36 – 47.

[2] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *Mobile Computing and Networking*, pages 32–43, 2000.

[3] James McLurkin. Algorithms for distributed sensor networks. Master's thesis, University of California at Berkeley, 1999.

[4] Dragos Niculescu and Badri Nath. Ad hoc positioning system (APS). In *GLOBECOM (1)*, pages 2926–2931, 2001.

[5] Chris Savarese. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. Master's thesis, University of California at Berkeley, 2002.

[6] Radhika Nagpal1. Organizing a Global Coordinate System from Local Information on an Ad-Hoc Sensor Network.

[7] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *INFOCOM (2)*, pages 775–784, March 2000.

[8] V. Seshadri, G. Záruba, M. Huber. A Bayesian Sampling Approach to In-door Localization of Wireless Devices Using Received Signal Strength Indication.

[9] M. Youssef, A. Agrawala, A. U. Shankar, and S. H. Noh, *A Probabilistic Clustering-Based Indoor Location Determination System*, Tech. Report, University of Maryland at College Park, CS-TR 4350, March 2002.

[10] M. Isard, and A. Blake, "Contour Tracking by Stochastic Propagation of Conditional Density," *European Conference on Computer Vision*, Cambridge, 1996, UK, pp. 343 – 356.

[11] M. Maroti, B. Kusy. Radio interferometric geolocation.

[12] G. Záruba, M. Huber. Monte Carlo Sampling Based In-Home Location Tracking With Minimal RF Infrastructure Requirements

[13] D. Lymberopoulos, Q. Lindsey, A. Savvides. An Empirical Analysis of Radio Signal Strength Variability in IEEE 802.15.4 Networks using Monopole Antennas.

[14] C. Alippi, G. Vannini. Wireless Sensor Networks and Radio Localization: a Metrological Analysis of the MICA2 received signal strength indicator.

[15] Cameron Dean Whitehouse. The Design of Calamari: an Ad-hoc Localization System for Sensor Networks.

[16] IEEE Std. 802.15.4 2003.

[17] Crossbow inc. www.xbow.com.

[18] Chipcon. www.chipcon.com.

[19] TinyOS. www.tinyos.net.

[20] Atmel. ATmega128 datasheet.

[21] D. Gay, P. Levis, et al. nesC 1.1 Language Reference Manual.

[22] P. Levis. TinyOS programming.

[23] Mills, D. L. Internet Time Synchronization: The Network Time Protocol. IEEE Transactions on Communications COM **39** no. 10, p. 1482–1493, October 1991.

[24] Elson, J. E., Girod, L., and Estrin, D. Fine-Grained Network Time Synchronization using Reference Broadcasts. The Fifth Symposium on Operating Systems Design and Implementation (OSDI), p. 147–163, December 2002.

[25] Ganeriwal, S., Kumar, R., and Srivastava, M. B. Timing-Sync Protocol for Sensor Networks. The First ACM Conference on Embedded Networked Sensor System (SenSys), p. 138–149, November 2003.

[26] M. Maròti, B. Kusy, G. Simon, A. Lédeczi. The Flooding Time Synchronization Protocol.

[27] von Eicken, T., et al. Active messages: a mechanism for integrated communication and computation. in 19th Annual International Symposium on Computer Architecture. 1992.
[28] P. Buonadonna, j. Hill, D. Culler. Active Message Communication for Tiny Networked Sensors.
[29] A. Medeisis, A. Kajackas. On the Use of the Universal Okumura-Hata Propagation Prediction Model in Rural Areas
[30] G. Evans, B. Joslin, L. Vinson, B. Foose. The optimization and application of the w.c.y. lee propagation model in the 1900 **mHZ** frequency band.