Localizzazione e SLAM con Wireless Sensors Network (WSN)

D. Hoxhaj , N. Mazzucato, M. Montis , M. Sommacal .

Marzo 28, 2008

1 Abstract

Lo scopo di questo lavoro è trattare il complesso problema della simultanea localizzazione e costruzione di una mappa di sensori wireless e della ricostruzione della posizione del veicolo mobile che naviga all'interno della mappa stessa. La ricostruzione della posizione dei nodi ancora viene effettuata impiegando un sistema di tipo *range-only*, ovvero un sistema nel quale si conosce solamente la distanza relativa tra veicolo mobile e nodo fisso (tale informazione viene ricavata dal segnale RSSI tra nodo fisso e nodo mobile).
Partendo dalle informazioni recuperate nella letteratura, il lavoro si sviluppa attraverso la creazione di un modello sufficientemente preciso (ma anche relativamente semplice da manipolare) dell'area di lavoro, l'implementazione di una soluzione impiegando il filtro di Kalman Esteso (*EKF*) e l'apporto di alcuni accorgimenti all'algoritmo risolutivo per migliorare la precisione della soluzione trovata.
Tutto questo è stato inizialmente implementato in ambiente *Matlab* per poter determinare, attraverso una serie di simulazioni, sia il comportamento del sistema in diverse situazioni sia individuare possibili punti critici e di forza; successivamente si è cercato di valutare tali risultati ottenuti in simulazione nei laboratori del *NAVLAB*, cercando di risolvere tutte le problematiche a livello pratico sorte.

2 Introduzione

Nel mondo dell'automazione e della robotica uno degli aspetti attualmente più complessi è cercare di rendere i robot più autonomi nelle loro decisioni e tale ricerca è molto attiva.
Nel problema della simultanea localizzazione e mappatura (*Slam*) si chiede se è possibile per un robot mobile posto in un ambiente totalmente sconosciuto e privo di conoscenze sulla propria posizione e orientamento ricostrutire in modo incrementale una mappa verosimile di tale ambiente e

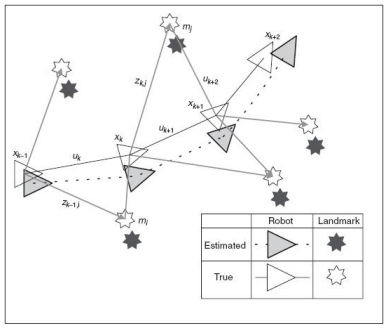


Figura 1: Esempio grafico del problema dello Slam.

al contempo di determinare la propria posizione all'interno della mappa stessa. Una soluzione al problema di *Slam* è vista come un *santo graal* per la comunicazione robotica mobile poichè ciò significherebbe la possibilità di rendere i robot *veramente* autonomi.
La soluzione del problema dello *Slam* è stato uno dei più grandi successi degli anni novanta per la comunità di studiosi e ricercatori di questo campo. Tale problema è stato formulato e risolto come un problema teorico in numerose differenti forme; inoltre lo *Slam* è stato implementato in vari ambiti: robot che lavorano in ambienti indoor, outdoor, sott'acqua e perfino per sistemi aerei. Il problema principale della simultanea localizzazione e della costruzione della mappa dell'ambiente risiede nel seguente paradosso: per localizzare se stesso, il robot ha bisogno di una mappa accurata dell'ambiente ma al contempo per costruire una mappa dettagliata dell'ambiente circostante il robot deve conoscere precisamente la propria posizione. Ad ogni modo a livello teorico e concettuale, tale problema può essere considerato un problema risolto. Rimangono però ancora alcuni quesiti riguardanti la realizzazione delle soluzioni del problema di *Slam* ed in particolare sulla creazione e sull'impiego di mappe precise e ricche di dettagli all'interno dell'algoritmo risolutivo stesso.
In generale si può dire che la scelta dipende fortemente dal

tipo di ambiente in cui si va ad operare, nonchè dalle assunzioni che vengono fatte. Ambienti strutturati possono essere ben rappresentati da mappe basate su landmark, o mappe topologiche. Ambienti non strutturati, in cui sono presenti molte irregolarità vengono rappresentati meglio con mappe a griglia. I principali approcci al problema possono essere suddivisi in due classi:

- sistemi che sfruttano il GPS o sensori che forniscono la distanza e la direzione;
- sistemi che sfruttano solo la distanza o la direzione dell'oggetto rilevato.

Poichè per il primo approccio sono state fatte molte ricerche a riguardo, il nostro lavoro sarà rivolto allo studio del secondo metodo indicato.
Come fin quì detto, i problemi che siamo chiamati ad affrontare e risolvere sono sostanzialmente due:

- la costruzione di una mappa più precisa possibile dell'ambiente a partire dai dati raccolti;
- la stima della posizione del robot all'interno di questa mappa.

Le informazioni disponibili per l'elaborazione sono le distanze relative robot-nodo ancora ottenute tramite segnali RSSI, ovvero segnali in cui intercorre una relazione tra potenza del segnale e distanza tra sorgente e destinazione. Data la forte non linearità di tale relazione e la rumorosità del canale trasmissivo, la risoluzione del problema di *Slam* prevederà delle scelte e degli accorgimenti *ad hoc* in fase di simulazione e sperimentazione.
Il progetto e la relazione si articolano sui seguenti punti:

1. *acquisizione del materiale e studio delle metodologie risolutive*: dalle ricerche fatte si è osservato che il problema di *Slam* è un problema sì definito e risolto, ma è anche un problema molto generico. Attraverso il materiale reperito si è cercato di individuare e definire:
   - i diversi approcci risolutivi impiegabili (lo *stato dell'arte* del problema di *Slam*);
   - il metodo migliore da implementare per risolvere il nostro problema specifico;
   - eventuali vincoli e condizioni per semplificare ed ottimizzare l'algoritmo risolutivo scelto.
2. *studio del sistema e modellizzazione*: si è cercato di costuire un modello matematico del sistema il più semplice e flessibile possibile, che si adattasse all'approccio risolutivo scelto. Ciò ha comportato lo studio delle grandezze fisiche agenti, in particolar modo le caratteristiche cinemantiche del veicolo mobile all'interno dell'area di lavoro con conseguente determinazione dei profili di velocità e accelerazione da adottare, lo studio del modello fisico del canale individuando le principali problematiche legate al filtraggio del rumore;
3. *studio simulativo dell'algoritmo risolutivo adottato*: lo studio effettuato sfruttando i risultati del punto precedente si suddividono in due parti:

   studio della mappatura di un'area : conoscendo in modo esatto la posizione e l'orinetamento del nodo mobile in ogni punto dell'ambiente si cerca di stimare nel modo più preciso possibile il numero e la posizione dei nodi ancora presenti;

   studio del problema di *Slam* : sfruttando lo studio effettuato per la mappatura dell'ambiente si cerca, diminuendo le conoscenze a priori del veicolo mobile, di determinarne con buon precizione la posizione in ogni punto della mappa.

   Per affrontare i problemi simulativi sono state create mediate il software *Matlab* delle routine ad hoc per la rappresentazione dell'area di lavoro, per la rappresentazione dei nodi ancora, del nodo mobile e del canale trasmissivo disponibile nei laboratori *Navlab*.
4. *verifica sperimentale*: nell'ultima parte del lavoro si è cercato di implementare in laboratorio la soluzione al problema di *Slam* trovata in fase simulativa e di verificare o confutare i risultati ottenuti nella fase precedente.

3 Storia dello Slam

Le origini del problema dello *Slam* risalgono a 1986 nel corso di una conferenza dell'IEEE chiamata *"Robotica e Automatica"*(IRAC) tenutasi a San Francisco. In quegli anni, i metodi probabilistici iniziavano a venire applicati alla robotica e all'intelligenza artificiale.
Nel corso della conferenza si prestó particolare attenzione alla discussione sulla mappatura, poiché da parte di diversi ricercatori erano stati applicati metodi di stima per mappatura e localizzazione. Inoltre ulteriori argomenti di conversazione quali questioni computazionali e concettuali furono sollevati da Raja Chatila, Oliver Faugeras e Randal Smith.
Dopo tale conferenza l'interesse per il problema accrebbe, con il risultato che negli anni successivi furono prodotti due importanti documenti:

- il lavoro di Smith, Cheesman e Durrant-Whyte che stabilì le basi statistiche per la descrizione delle relazioni tra caratteristiche dell'ambiente e incertezze sulla manipolazione geometrica. Il punto cruciale del lavoro

é stato il dimostrare che vi deve essere un alto grado di correlazione tra le stime della posizione dei diversi oggetti dell'ambiente, ed esso deve crescere con il numero di osservazioni;

- allo stesso tempo Ayache e Faugeras pensarono alla navigazione mediante visione;
- Crowley, Chatila e Laumond lavorarono alla navigazione di Robot mobili con sonar utilizzando il filtro di Kalman.

Questi due argomenti di ricerca avevano molto in comune e ció si comprese dalla pubblicazione di Smith: la ricerca spiegava che quando un robot mobile si spostava in un ambiente sconosciuto prendendo le relative osservazioni, le stime venivano tutte necessariamente correlate poichè era comune l'errore di stima nella localizzazione del veicolo. Tale assunzione implicava che una soluzione completa del problema di mappatura e localizzazione richiedeva uno stato composto da posizione del veicolo mobile e tutta la posizione dell'ambiente circostante per aggiornare le osservazioni. A sua volta, questo richiedeva per lo stimatore un grande vettore di stato con un onere di calcolo pari al quadrato dei punti di riferimento fissi.

Fondamentalmente questa idea non teneva conto delle proprietá di convergenza della mappa e del suo comportamento allo stato stazionario. Le suddette ipotesi sono state fatte dal momento che gli errori sulla mappa stimata non convergevano, ma si vedeva al contrario un aumento illimitato dell'errore. Cosí, vista la complessitá computazionale del problema di mappatura e vista l'assenza della convergenza della mappa, i ricercatori decisero di effettuare delle approssimazioni: minimizzare o eliminare le correlazioni tra riferimenti, riducendo cosí il filtro completo ad una serie di filtri disaccoppiati tra riferimenti e veicolo. Per questi motivi il lavoro teorico contemporaneo di mappatura e localizzazione subì una battuta d'arresto e i due problemi vennero presi in considerazione separatamente.

La svolta concettuale venne quando si realizzò che il problema combinato, formulato come un unico problema di stima, convergeva. Di notevole importanza fu anche il riconoscimento che la correlazione tra riferimenti, che molti ricercatori avevano cercato di minimizzare, era diventata la parte critica del problema poiché al contrario piú questa cresceva migliore era la soluzione. Fu così che la struttura del problema, il risultato di convergenza e la coniatura dell'acronimo *Slam* vennero presentati in un documento di robotica mobile durante l'*"International Symposium on Robotics Research"(IROS)* tenutosi nel 1995.

La teoria essenziale sulla convergenza e molti risultati iniziali vennero sviluppati da Csorba. Altri gruppi lavorarono su mappatura e localizzazione, tra cui Zaragoza al MIT e l'ACFR a Sydney. Da qua in poi il lavoro si é concentrato sul miglioramento dell'efficenza computazionale, sulla risoluzione di questioni di associazione dei dati e sui loop di chiusura.

L'*"International Symposium on Robotics Research"* del 1999 fu un importante punto d'incontro poiché si trovò un'intesa tra la teoria di $Slam$ basata sul Filtro di Kalman e i metodi probabilistici di localizzazione e mappatura introdotti da Thrun. Nel 2000 alla conferenza sulla *"Robotica e Automatica"* dell'IEEE l'interesse fu rivolto su temi quali la complessitá algoritmica, l'associazione dei dati e la realizzazione di sfide.
Nella successiva conferenza del 2002 sullo *Slam* ospitata da Henrik Christiansen al KTH di Stoccolma furono presentati alcuni dei principali ricercatori provenienti da tutto il mondo; il risultato alla fine della conferenza fu un enorme successo per quanto riguarda la costruzione dell'ambiente. Negli anni a venire l'interesse per lo *Slam* crebbe esponenzialmente e si sono continuate a tenersi le conferenze ICRA e IROS. Nel 2004 a Tolouse nacque la prima scuola estiva sullo *Slam* e successivamente si svolse ad Oxford nel 2006. Tutt'oggi lo *Slam* continua ad essere un argomento gettonato in ricerca.

4 Stato dell'arte

4.1 Slam Probabilistico

Nella forma probabilistica, il problema di simultanea localizzazione e costruzione della mappa richiede che la distribuzione di probabilità

$$P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) \tag{1}$$

venga calcolata per tutti i k passi. Questa distribuzione di probabilità descrive la *densità congiunta a posteriori* (*joint posterior density*) dei nodi ancora e dello stato del veicolo (al tempo k) ed è data dalle osservazioni precedenti, dall'ingresso di controllo e dallo stato iniziale del robot mobile.
In generale, si cerca di determinare una soluzione *ricorsiva* al problema di $Slam$. Partendo da una stima per la distribuzione al tempo $k-1$ del tipo:

$$P(\mathbf{x}_{k-1}, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k-1}) \tag{2}$$

la *joint posterior density*, a seguito di un ingresso di controllo u_k e all'osservazione z_k, può essere facilmente calcolata impiegando il *teorema di Bayes*. Questo calcolo richiede che il modello dello stato di transizione e il modello di osservazione siano definiti descrivendo rispettivamente l'effetto dell'input e dell'osservazione.

Il modello di osservazione descrive la probabilità di effettuare un'osservazione z_k quando il veicolo mobile e la posizione del nodo ancora sono conosciute; generalmente tale probabilità viene indicata nella forma:

$$P(\mathbf{z}_k|\mathbf{x}_k, m) \tag{3}$$

E' ragionevole supporre che, una volta che il veicolo e la mappa siano definiti, le osservazioni successive possano essere supposte *condizionatamente indipendenti* dalla mappa e dall'attuale stato del robot.
Il modello di movimento del veicolo può essere descritto in termini di probabilità su transizioni di stato della forma:

$$P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) \tag{4}$$

In questo modo lo stato di transizione può essere descritto attraverso un *processo di Markov* nel quale il prossimo stato x_k dipende solo dallo stato immediatamente precedente x_{k-1} mentre l'ingresso u_k applicato è indipendente ad entrambe le osservazioni e alla mappa.

L'algoritmo di $Slam$ viene così implementato in una forma *ricorsiva e sequeziale* composta dai seguenti due passi:

passo 1 : *Time update*

$$P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0) = \int P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)$$

$$\times P(\mathbf{x}_{k-1}, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k-1}, \mathbf{x}_0) d\mathbf{x}_{k-1} \tag{5}$$

passo 2 : *Measurement update*

$$P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)$$

$$= \frac{P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m}) P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0)}{P(\mathbf{z}_k|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k})} \tag{6}$$

Tali equazioni forniscono una procedura ricorsiva per determinare la distribuzione

$$P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) \tag{7}$$

per il robot e per la mappa al k-esimo passo basato su tutte le osservazioni fatte ($\mathbf{Z}_{0:k}$) e su tutti gli ingressi impiegati ($\mathbf{U}_{0:k}$). La ricorsione è perciò funzione di un modello del veicolo $P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)$ e di un modello di osservaizione $P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})$.
Vale la pena notare che il problema di costruzione della mappa può essere formulato come il calcolo della densità condizionale

$$P(\mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{X}_{0:k}) \tag{8}$$

Questo presuppone che la posizione del veicolo mobile sia nota (o almeno deterministica) in ogni momento e soggetta alla conoscenza della condizione iniziale. Una mappa **m** viene poi costruita dalla fusione di osservazioni dell'ambiente in posizioni differenti.

Viceversa il problema di localizzazione del veicolo può essere formulato come il calcolo della distribuzione di probabilità

$$P(\mathbf{x}_k|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{m}) \tag{9}$$

Ciò presuppone che le posizioni dei nodi ancora siano note con certezza, e l'obiettivo è quello di calcolare una stima della posizione del veicolo rispetto a tali nodi fissi.

4.2 Struttura dello Slam Probabilistico

Il modello di osservazione $P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})$ rende esplicita la dipendenza delle osservazioni sia sul veicolo sia sui sensori fissi. Ne consegue che la *joint posterior density* non può essere partizionata come $P(\mathbf{x}_k, \mathbf{m}|\mathbf{z}_k) = P(\mathbf{x}_k|\mathbf{z}_k)P(\mathbf{m}|\mathbf{z}_k)$ poichè in caso contrario si avrebbero stime inconsistenti. Tuttavia il problema di $Slam$ ha una struttura molto più profonda rispetto a quello che si può evincere da queste equazioni.
In generale si può dire che l'errore tra valori stimati e valori reali delle posizioni dei sensori hanno un andamento comune e omogeneo: ciò è dovuto al fatto che tutti i nodi fissi fanno riferimento ad una stessa sorgente, ovvero si confrontano con il medesimo nodo mobile. Vengono poi introdotti errori sulla conoscenza sulla posizione del veicolo a causa delle osservazioni fatte sui sensori fissi. Tutto questo comporta che gli errori nelle stime delle posizioni dei sensori siano *fortemente correlati*: in pratica la posizione relativa tra due nodi fissi m_i e m_j può essere conosciuta con buona precisione, anche quando la posizione assoluta è abbastanza incerta. Dal punto di vista probabilistico questo significa che la densità di probabilità congiunta per la coppia di sensori $P(m_i, m_j)$ ha un valore elevato (presenta un picco) anche quando le densità marginali $P(m_i)$ sono molto dispersive.

La più grande intuizione nel problema di $Slam$ è stato quello di capire che la correlazione tra le stime dei sensori ha un andamento monotono crescente in funzione del numero di osservazioni fatte[1]. Ne consegue che la conoscenza della posizione relativa dei nodi ancora migliora sempre e non diverge mai, indipendentemente dal movimento del robot; in termini probabilistici la densità congiunta su tutti i sensori $P(m_i)$ tende a crescere velocemente all'aumentare delle osservazioni. Tale convergenza si verifica perchè le osservazioni fatte dal robot possono essere considerate

[1] questi risultati sono stati provati solo nel caso Gaussiano. La prova di formule per casi probabilistici più generali sono ancora un problema aperto.

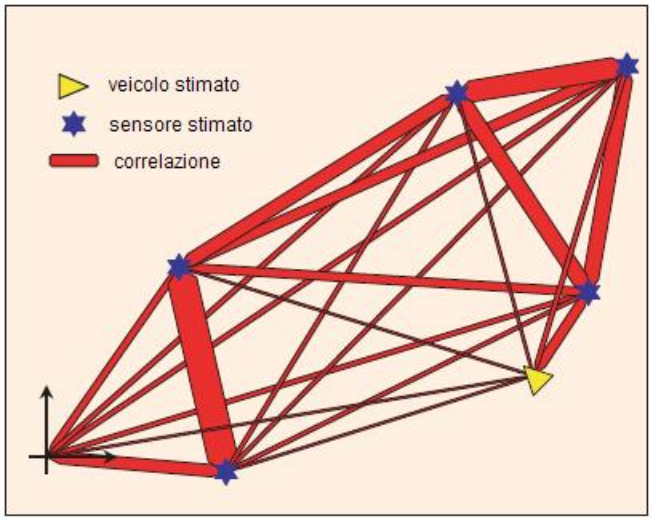


Figura 2: Esempio di correlazione tra nodi fissi e veicolo mobile.

come *"quasi indipendenti"*.

Si può allora pensare di creare un grafo tra i vari nodi ancora e il nodo mobile la cui informazione in ogni arco sia la correlazione (figura 2): tale informazione risulta divenire sempre più precisa ogni volta che viene eseguita una nuova osservazione.
Definendo un intorno arbitrario di un punto della mappa (tale punto viene definito *sistema sorgente*), risulta che localmente il suo effetto è legato alla proprietà di *rigidità* (*correlazione*) e questa *rigidità* diminuisce all'aumentare della distanza dagli altri sensori. In particolare ogni volta che il veicolo mobile si muove nell'ambiente e prende delle osservazioni dei nodi fissi, il sistema sorgente diventa sempre più rigido.

Ne consegue che al limite si riesce ad ottenere:

- una mappa di riferimento rigida

oppure

- una mappa relativa discretamente accurata

Quando la mappa è disponibile, l'accuratezza della stima della posizione del veicolo mobile (attraverso le misure *relative* disponibili) è limitata solo dalla qualità della mappa stessa. Teoricamente risulta che la precisione della posizione del robot diviene uguale alla precisione di localizzazione ottenibile con una mappa deterministica.

4.3 Slam e stima attraverso il filtro di Kalman Esteso

Tra le varie pubblicazioni trovate inerenti al problema di $Slam$ ne è stata individuata una molto interesante, nella quale dà una possibile soluzione al problema di localizzazione e mappatura in una *DTN* (*delay-tolerant sensor network*) attraverso il filtro di Kalman (in una sua particolare versione) e l'impiego di segnali RSSI.
Scritto da Pubudu N. Pathirana, Nirupama Bulusu, Andrey V. Savkin e Sanja Jha, la pubblicazione descrive una rivisitazione del problema della localizzazione dei nodi attraverso la stima della loro posizione nella DTN. Un ambiente di questa tipologia ha numerose caratteristiche che permettono approcci alternativi alla risoluzione del problema.
Il loro lavoro si articola nei seguenti punti:

- propongono un nuovo approccio al problema che permette, impiegando uno o più veicoli mobili, di localizzare i sensori in una DTN, eliminando l'obbligo di elaborare un numero piccolo di dispositivi. Può anche essere sfruttata la mobilità dei sensori per ridurre gli errori di localizzazione e il numero di riferimenti statici richesti per localizzare univocamente una rete di sensori;
- sviluppano un nuovo tipo di *Filtro di Kalman Esteso Robusto* (REKF) basato sull'algoritmo di stima per la localizzazione di nodi nella DTN. La localizzazione basata sulla misura della potenza di segnale è risolta trattandola come una stima in linea in un sistema dinamico non lineare. Il modello matematico che propongono ingloba errori di misura e di incertezza e risulta essere numericamente più efficiente e robusto se confrontato con implementazioni che prevedono l'utilizzo del classico Filtro di Kalman (EKF).

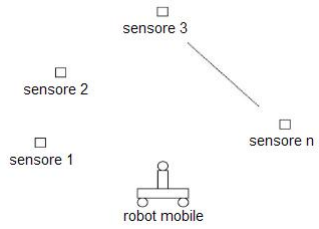


Figura 3: Schema di una rete di sensori.

4.4 Modello dinamico del sistema proposto dalla pubblicazione

Lavorando con un sistema in cui anche i nodi da localizzare possono essere in movimento, definiscono come robot mo-

bile il veicolo mobile connesso alla *wireless base station*. Inoltre i sensori da localizzare sono distribuiti in modo casuale nell'ambiente.
Il modello dinamico per $n+1$ elementi (n sensori ed il robot mobile) può essere descritto in un sistema cartesiano a due coordinate come segue:

$$\dot{x}(t) = Ax(t) + B_1 u(t) + B_2 w(t) \tag{10}$$

avendo posto:

$$A = \begin{bmatrix} \Theta & & 0 \\ & \ddots & \\ 0 & & \Theta \end{bmatrix} \tag{11}$$

$$-B_1 = \begin{bmatrix} \Phi \\ \vdots \\ \Phi \end{bmatrix} \tag{12}$$

$$B_2 = \begin{bmatrix} \Phi & & 0 \\ & \ddots & \\ 0 & & \Phi \end{bmatrix} \tag{13}$$

Le matrici Θ e Φ sono definite come segue:

$$\Theta = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{14}$$

$$\Phi = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \tag{15}$$

Il vettore di stato $x(t) = [x_1(t)...x_i(t)...x_n(t)]'$ è composto da

$$x_i(t) = [X_i(t)Y_i(t)\dot{X}_i(t)\dot{Y}_i(t)]' \tag{16}$$

dove con $X_i(t)$ e $Y_i(t)$ si rappresenta la posizione dell'i-esimo sensore rispetto al robot mobile all'istante t e con la loro derivata ($\dot{X}_i(t)$ e $\dot{Y}_i(t)$) le velocità relative lungo le direzioni X e Y.
In altre parole se si rappresenta con il vettore

$$x_c(t) = [x_c(t)y_c(t)\dot{x}_c(t)\dot{y}_c(t)]' \tag{17}$$

lo stato assoluto del robot mobile (ovvero posizioni e velocità assolute lungo i due assi cartesiani) e con i vettori

$$x^i(t) = [x^i(t)y^i(t)\dot{x}^i(t)\dot{y}^i(t)]' \tag{18}$$

lo stato assoluto del sensore i-esimo, l'i-esimo stato del modello dinamico (legato all'i-esimo sensore) è rappresentato dalla seguente relazione:

$$x_i(t) \equiv x_c(t) - x_s^i(t) \tag{19}$$

Il vettore $u(t)$ individua invece il comando di accelerazione del robot mobile lungo i due assi cartesiani (i valori sono ricavati da appositi trasduttori) mentre $w(t)$ denota il comando (sconosciuto) di accelerazione nelle due direzioni dei rimanenti n sensori. E' possibile considerare anche il caso in cui i sensori siano in una posizione fissa, ed in questo caso risulta $w(t) = 0$.

Attraverso questa rappresentazione in spazi di stato del modello dinamico si osserva che è possibile determinare la posizione dei sensori attraverso la stima dello stato x impiegando le misure y. Nel problema di localizzazione inoltre tutte le posizioni dei sensori sono sconosciute a priori; si assume allora che tutti i nodi siano posizionati nella locazione $(0, 0)$.
La ricerca fatta ha dimostrato che con queste assunzioni iniziali lo stato converge verso la configurazione reale, e che quindi tutte le posizioni di ogni sensore sconosciute a priori possono essere stimate (ciò vale anche per la posizione del robot mobile).

4.5 Modello di misura del segnale RSSI proposto

In una rete wireless, la distanza tra due dispositivi posti in comunicazione può essere osservata usando l'*RSSI* (*received signal strength indication*) del ricevitore. Quando sono presenti più trasmissioni, ogni informazione trasmessa è univocamente determinata attraverso il pacchetto dati inviato.
Rimandando la descrizione del canale di comunicazione alla sezione 5, si individua che il segnale RSSI $p_i(t)$ associato all'i-esimo sensore può essere formulato come segue:

$$p_i(t) = p_{oi} - 10\epsilon \log d_i(t) + v_i(t) \tag{20}$$

dove:

- $p_{oi}(t)$ è una costante dipendente dalla potenza di segnale trasmessa, dalla lunghezza d'onda e dal guadagno dell'antenna del robot mobile;
- ϵ è un indice di decrescenza del segnale;
- $v_i(t)$è il logaritmo dell'attenuazione dovuta allo *shadowing*.

Così facendo si ha che il termine $d_i(t)$ rappresenta la distanza relativa tra il robot mobile e l'i-esimo sensore e tale

distanza può essere espressa in funzione delle componenti X_i e Y_i :

$$d_i(t) = \sqrt{X_i(t)^2 + Y_i(t)^2} \tag{21}$$

Il vettore di osservazione che ne deriva è quindi

$$y(t) = \begin{bmatrix} p_1(t) \\ . \\ . \\ . \\ p_n(t) \end{bmatrix} \tag{22}$$

L'equazione di misura effettuata dal robot mobile per gli n sensori è definita come

$$y(t) = C(x(t)) + v(t) \tag{23}$$

dove

$$v(t) = \begin{bmatrix} v_1(t) \\ . \\ . \\ . \\ v_n(t) \end{bmatrix} \tag{24}$$

$$C(x(t)) = \begin{bmatrix} p_{oi} - 10\epsilon \log\left(X_1(t)^2 + Y_1(t)^2\right) \\ . \\ . \\ . \\ p_{oi} - 10\epsilon \log\left(X_n(t)^2 + Y_n(t)^2\right) \end{bmatrix} \tag{25}$$

4.6 Filtro di Kalman proposto

Il sistema dinamico in spazi di stato fin quì descritto considera in ingresso due rumori:

1. il rumore di misura v nell'equasione di uscita del sistema $y(t) = Cx(t) + v(t)$;
2. il rumore sull'accelerazione w indicato nell'equazione di stato $\dot{x}(t) = Ax(t) + B_1u(t) + B_2w(t)$.

In questa applicazione le condizioni iniziali sugli errori sono molto importanti tanto quanto la completa ignoranza delle posizioni iniziali dei sensori poichè tale risultato è direttamente collegato all'algoritmo risolutivo che si vuole proporre.
I due rumori di ingresso e gli errori di stima iniziali (di ogni sensore) devono soddisfare l'equazione ICQ.s Se esiste una soluzione per l'equazione di Riccati allora l'equazione ICQ risulta soddisfatta e il vettore di stato può essere stimato dalle misure dei segnali RSSI attraverso l'impiego di una versione *robusta* del *filtro esteso di Kalman*. ($REKF$).

Utilizzando tale filtro in una DTN l'i-esimo sistema (ovvero robot mobile e i-esimo sensore), durante un intervallo di campionamento, può essere rappresentato da un sistema non lineare del tipo:

$$\begin{aligned} \dot{x} &= A(x,u) + B_2w \\ z &= K(x,u) \\ y &= C(x) + v \end{aligned} \tag{26}$$

e dalla relazione IQC:

$$(x(0) - x_0)' N_i (x(0) - x_0) + \alpha(w,v,t) \leq d + \beta(z,t) \tag{27}$$

avendo posto:

$$\alpha = \frac{1}{2} \int_0^s \left(w(t)' Q_i(t) w(t)\right) + v(t)' R_i(t) v(t) dt \tag{28}$$

$$\beta = \frac{1}{2} \int_0^s z(t)' z(t) dt \tag{29}$$

con $Q_i > 0$, $R_i > 0$ e $N_i > 0$.

Lo stato iniziale x_0 è conseguentemente lo stato stimato del rispettivo sistema allo startup. In una relazione incerta come lo è la IQC, il rumore di misura, l'accelerazione sconosciuta del robot mobile e l'incertezza delle condizioni iniziali vengono considerate come ingressi (incerti) deterministici limitati. In particolare l'equazione di misura può essere riscritta come:

$$y = C(x) + \delta C(x) + v_0 \tag{30}$$

dove $|\delta| \leq \epsilon$ avendo indicato con ϵ il limite superiore del modulo della parte limitata del rumore.
Scegliendo $z = \xi C(x)$ e $v = \delta C(x)$ si osserva:

$$\int_0^T |v| dt \leq \int_0^T z' z dt \tag{31}$$

Considerando v_0 e la corrispondente incertezza in w (w_0) soddisfacente il limite

$$\Phi(x(0)) + \int_0^T \left[w_0(t)' Q w_0(t) + v_0(t)' R v_0(t)\right] dt \leq d \tag{32}$$

si può notare che il sistema così definito viene a soddisfare la condizione di IQC.

Volendo essere questa soltanto un'indicazione delle informazioni da cui siamo partiti per sviluppare il nostro approccio risolutivo al problema di *Slam*, si rimanda a [1] per approfondimenti e ulteriori chiarimenti.

5 Modello del canale

Una parte importante della modellizzazione del sistema in cui si è operato risiede nella definizione e nella determinazione delle caratteristiche fisiche e tecniche del canale trasmissivo impiegato. Tale modello risulta però molto difficile da ottenere in ambienti indoor poichè per averene uno sufficientemente accurato si rende necessario conoscere con precisione l'area in cui avvengono le comunicazioni in modo da poter calcolare gli effetti delle riflessioni agenti sui segnali.
In genere per i problemi di localizzazone vengono impiegati chip radio che utilizzano come parametri:

LQI : il *Level Quality Indicator* indica la facilità di associare la sequenza di simboli ricevuti ad una parola di quattro bit. Quindi tale valore si lega alla misura della quantità di rumore presente nella banda in cui avviene la trasmisione;

RSS : l'RSS (*Received Signal Strength*) è un valore determinato a partire dal valore di RSSI (*Received Signal Strength Indication*), e i due parametri sono legati dall'equazione:

$$RSS = RSSI - 45[dBm] \tag{33}$$

L'RSSI è un segnale la cui potenza decresce con legge logaritmica all'aumentare della distanza tra sorgente e ricevente.

Per misurare la distanza relativa tra veicolo mobile e nodi ancora e per costruire il modello matematico è stato allora impiegato il segnale RSSI vista la semplicità della relazione che lega la distanza di trasmissione con la potenza del campo elettromagnetico:

$$P(d) = P(d_0) - 10n_p \log\left(\frac{d}{d_0}\right) + \chi_\sigma \tag{34}$$

Nella formula appena indicata si definiscono:

- la potenza $P(d)$ del campo elettromagnetico ricevuto espressa in $[dBm]$; è quindi la potenza che il ricerivore del veicolo mobile capta e invia in elaborazione;
- la potenza $P(d_0)$ del campo elettromagnetico ad una distanza d_0 fisata dal trasmettitore (espressa anch'essa in $[dBm]$);
- il fattore n_p di decrescenza della potenza del campo elettromagnetico in funzione del logatitmo della distanza del trasmettitore;
- la variabile aleatoria gausiana χ_σ di media nulla e varianza σ^2 (espressa in $[dBm]$);
- la distanza d (espressa in $[m]$) dal trasmettitore.

Ne consegne che $P(d)$ è una variabile aleatoria gaussiana di media

$$\overline{P}(d) = P(d_0) - 10n_p \log\left(\frac{d}{d_0}\right) \tag{35}$$

e varianza σ in quanto $P(d)$ è esprimibile come trasformazione lineare affine di una variabile aleatoria gaussiana.
La stessa relazione tra distanza e potenza del campo elettromagnetico può essere espressa con una formulazione alternativa, ovvero:

$$P(d) = P_{TX} + A - 10n_p \log(d) + \chi_\sigma \tag{36}$$

dove

- P_{TX} indica la potenza di trasmissione nota;
- A è il fattore di attenuazione.

Questa seconda formulazione deriva proprio dalla relazione che lega distanza e intensità di campo elettromagnetico: infatti all'aumentare della distanza d tra trasmettitore e ricevitore la potenza del campo elettromagnetico decresce con legge:

$$\begin{aligned} P(d) &= \frac{P(d_0)}{\left(\frac{d}{d_0}\right)^{n_p}} \\ &\downarrow \\ &= 10\log\left(P(d_0)\right) + 10n_p \log\left(P(d_0)\right) - 10n_p \log\left(P(d)\right) \end{aligned} \tag{37}$$

$$\begin{aligned} P(d)_{[dBm]} &= 10\log\left(\frac{P(d)}{1[mW]}\right) \\ &\downarrow \\ &= P_{TX} + A - 10n_p \log\left(\frac{d}{d_0}\right) \end{aligned} \tag{38}$$

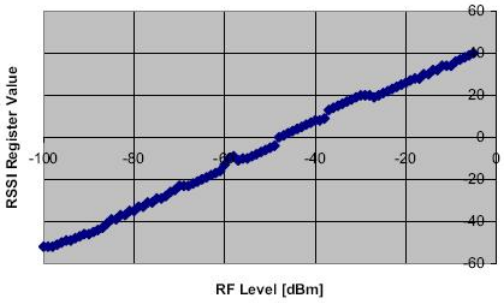


Figura 4: Andamento dell'RSSI in funzione della potenza del segnale ricevuto.

Avendo a disposizione il valore della potenza del campo elettromagnetico al ricevitore è possibile quindi, rovesciando ad esempio la seconda relazione, determinare la distanza a cui è posta la trasmittente; risulta quindi:

$$d = e^{\frac{\gamma}{2}} 10^{\frac{A - P_{TX}}{10 n_p}} \tag{39}$$

avendo posto:

$$\gamma = \left(\frac{\sigma \ln 10}{10 n_p} \right)^2 \tag{40}$$

Questa relazione non lineare sarà quindi quella che verrà impiegata all'interno del modello matematico.

Per la scelta dei parametri sono stati impiegati quelli trovati sperimentalmente nei laboratori del $NAVLAB$ e riportati in [2]; vengono inoltre prese in considerazione tutte le informazioni ed i risultati sperimentali ottenuti dai ricercatori:

1. durante le misure svolte di è stimato come fattore di attenuazione $A = -18.2[dBm]$ e come fattore di decrescenza $n_p = 2.18$;
2. i valori delle *potenze di comunicazione minime* fra i nodi della rete sono dei buoni indici per determinare dei limiti alle *massime distanze* comprese fra gli stessi;
3. le potenze di trasmizzione maggiormente utili al processo di localizzazione sono per lo più quelle minime. In particolare sono state utilizzate anche delle potenze di trasmissione non dichiarate nei dati dei data-sheet del chip EM2420, ma che i dati sperimentale hanno indicato essere efficientemente utilizzabile;
4. il valore del parametro LQI è utile per decidere se accettare o scrtare i pacchetti ricevuti.

6 Il Filtro di Kalman

Il filtro di Kalman è un insieme di equazioni matematiche che forniscono un metodo computazionale efficiente per stimare li stato di un processo, in modo da minimizzare l'errore quadratico medio. A differenza di altre tipologie di filtro (come per esempio il filtro di Wiener o di Levinson) la soluzione che adotta Kalman si presta in maniera eccezionale ad essere impiegata in problemi definiti in spazi di stato; inoltre tra i suoi punti di forza si osserva la possibilità di stimare stati passati, presenti e futuri e di poterlo fare anche quando il modello esatto del sistema è sconosciuto.
Il filtro nella sua versione originale era pensato per stimare lo stato di un processo a tempo discreto governato da un'equazione differenziale stocastica lineare (per i processi non lineari si rimanda al *filtro di Kalman esteso*).

6.1 Descrizione del filtro di Kalman

Il filtro formisce una stima di un processo utilizzando una forma di controllo a *feedback*: il filtro stima il processo ad un certo istante e ottiene un feedback sotto forma di misurazione affetta da rumore.
Le equazioni del filtro si distinguono in:

- *time update*: sono le equazioni responsabili della previsione dello stato attuale e della covarianza dell'errore e permettono di ottenere una stima *a priori* dello stato del sistema;
- *measurement update*: sono invece le equazioni che governano il feedback e vengono impiegate per correggere con una nuova misurazione la stima a priori fatta al passo precedente, in modo tale da ottenere una stima *a posteriori* migliore.

Il filtro cerca di stimare lo stato $x \in \mathbf{R}^n$ di un processo regolato dall'equazione a tempo discreto:

$$x(k+1) = Ax(k) + Bu(k) + w(k) \tag{41}$$

con

- $u(k)$ ingresso del processo al k-esimo istante;
- $w(k)$ rumore di processo modellato come una variabile gaussiana a media nulla e varianza Q:

$$p(w) \sim N(0, Q) \tag{42}$$

La stima viene fatta attraverso le misurazioni $y \in \mathbf{R}^m$ ottenute come uscita dell'equazione:

$$y(k) = Cx(k) + v(k) \tag{43}$$

dove in questo caso $v(k)$ rappresenta il rumore di misura ed è anch'esso modellabile come una variabile aleatoria gaussiana di media nulla e varianza R:

$$p(v) \sim N(0, R) \tag{44}$$

Per descrivere il comportamento della stima del filtro di Kalman si introducono ora le definizioni di *stato "a priori"* e di *stima "a posteriori"*:

stato a priori : è il vettore $\hat{x}(k+1|k) \in \mathbf{R}^n$ ottenuto al k-esimo passo data la conoscenza del passo k;

stima a posteriori : è il vettore $\hat{x}(k+1|k+1) \in \mathbf{R}^n$ ottenuto come stima a posteriori data la misura $y(k+1)$.

Si definiscono a questo punto come *errori di stima* (a *priori* e a *posteriori*) le quantità:

$$e(k|k) = x(k+1) - \hat{x}(k|k) \tag{45}$$

$$e(k+1|k) = x(k+1) - \hat{x}(k+1|k) \tag{46}$$

a cui si associano le covarianze:

$$P(k|k) = E\left[e_{k|k} e_{k|k}^T\right] \tag{47}$$

$$P(k+1|k) = E\left[e_{k+1|k} e_{k+1|k}^T\right] \tag{48}$$

Gli stimatori lineari a minima varianza $\hat{x}(k+1|k)$ e $\hat{x}(k|k)$ dello stato del modello lineare al passo $k+1$ e k in base alle osservazioni $\{y(s) : k_0 \leq s \leq k\}$ sono calcolabili mediante il seguente *algoritmo ricorsivo*:

1. *Stime a Priori*:
   è l'aggiornamento temporale dello stato e della varianza d'errore

$$\hat{x}(k+1|k) = F\hat{x}(k|k) + SR^{-1} \tag{49}$$

$$P(k+1|k) = FP(k|k)F^T + \tilde{Q} \tag{50}$$

2. *Stime a Posteriori*:
   è l'aggiornamento rispetto alle misure

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + L(k+1)[y(k+1) - C\hat{x}(k+1|k)] \tag{51}$$

$$P(k+1|k+1) = P(k+1|k) - P(k+1|k)C^T\Lambda(k+1)^{-1}CP(k+1|k) \tag{52}$$

3. *Condizioni Iniziali*:

$$\hat{x}(k_0|k_0 - 1) = \mu_0 \tag{53}$$

$$P(k_0|k_0 - 1) = P_0 \tag{54}$$

avendo posto nelle relazioni appena indicate:

- la matrice S indica la correlazione tra i rumori di processo e di misura. Essa deriva dalla matrice di covarianza:

$$E\left\{\begin{bmatrix} \mathbf{v}(t) \\ \mathbf{w}(t) \end{bmatrix} \begin{bmatrix} \mathbf{v}(s)^T & \mathbf{w}(s)^T \end{bmatrix}\right\} = \begin{bmatrix} Q & S \\ S & R \end{bmatrix} \delta(t-s) \tag{55}$$

- la matrice F è definita dall'equazione

$$F = A - SR^1C \tag{56}$$

- la matrice $\tilde{Q}$ è la varianza del rumore bianco $\tilde{v}(k)$ ($\tilde{v}(k) = v(k) - SR^{-1}w(k)$) ed è definita come

$$\tilde{Q} = Q - SR^{-1}S^T \tag{57}$$

- la matrice $\Lambda(k)$ è la varianza del *processo di innovazione* $e(k) = y(k) - C\hat{x}(k|k-1)$

$$\Lambda(k) = CP(k|k-1)C^T + R \tag{58}$$

- la matrice $L(k)$ è il guadagno del filtro di Kalman ed è definito dalla relazione

$$L(k) = P(k|k-1)C^T\Lambda^{-1}(k) \tag{59}$$

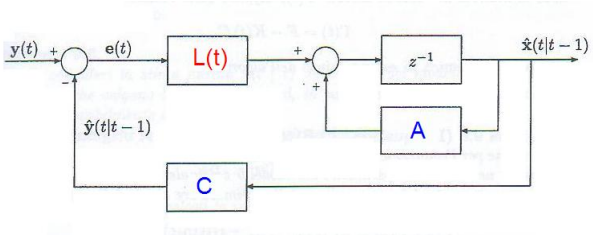


Figura 5: Schema a blocchi del filtro di kalman.

6.2 Il filtro di Kalman esteso [EKF]

Come si è visto precedentemente, il filtro di Kalman risolve il problema della stima di uno stato di un processo governato da equazioni lineari. Quando invece il processo da stimare e/o la relazione tra la misura e il processo non è lineare è necessario ricorrere ad una variante di tale filtro, denominata *filtro di Kalman Esteso (EKF)*.
L'EKF è un algoritmo di impiego molto generale che risolve in modo approssimato il problema di stima dello stato in un modello non lineare attraverso una *linearizzazione intelligente*[2], ovvero una linearizzazione fatta, ad ogni istante di campionamento, attorno alla migliore stima di $x(k)$ disponibile a quel momento. In questo modo si riesce a mantenere gli errori di linearizzazione a un livello accettabile.

[2] poichè nei problemi di misura conta molto la *precisione* dei risultati, una linearizzazione non appropriata può portare a produrre degli errori irrealisticamente grandi causati da misure descritte troppo approssimativamente

Si considera allora un processo avente lo stato $x \in \mathbf{R}^n$ governato da un'equazione non lineare del tipo:

$$x(k+1) = f(x(k), u(k), w(k)) \tag{60}$$

Similmente si può avere che anche l'equazione di osservazione sia non lineare:

$$y(k) = h(x(k), v(k)) \tag{61}$$

Come accadeva nel processo lineare, i termini $w(k)$ e $v(k)$ indicano rispettivamente i rumori di processo e di misura; non conoscendo i valori in ogni mommento è lecito approssimare le due equazioni ignorando il rumore ottenendo così:

$$\tilde{x}(k+1) = f(\hat{x}(k), u(k), 0) \tag{62}$$

$$\tilde{y}(k) = h(\hat{x}(k), 0) \tag{63}$$

Si osserva che dopo la linearizzazione eseguita dal filtro EKF la distribuzione delle variabili casuali non sono più lineari dopo la loro linearizzazione.
A questo punto di possono scrivere le nuove equazioni che linearizzano una stima delle nuove relazioni:

$$x(k+1) \approx \tilde{x}(k+1) + A(x(k) - \hat{x}(k)) + Ww(k) \tag{64}$$

$$y(k) \approx \tilde{y} + H(x(k) - \tilde{x}(k)) + Vv(k) \tag{65}$$

avendo posto:

- $x(k)$ e $y(k)$ lo stato e l'osservazione al passo attuale;
- $\tilde{x}(k)$ e $\tilde{y}(k)$ lo stato e l'osservazione approssimata tramite le equazioni (62) e (63);
- $\hat{x}$ come stima *a posteriori* dello stato al passo k;
- $w(k)$ e $v(k)$ sono il rumore del processo e il rumore della misura;
- A risulta essere la matrice Jacobiana delle derivate parziali della funzione f rispetto allo stato x:

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}(k), u(k), 0) \tag{66}$$

- W è la matrice Jacobiana delle derivate parziali di f rispetto al rumore di processo w:

$$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{x}(k), u(k), 0) \tag{67}$$

- H è la matrice Jacobiana delle derivate parziali di h rispetto allo stato x:

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\tilde{x}(k), 0) \tag{68}$$

- V è la matrice Jacobiana delle derivate parziali di h rispetto al rumore di misura v:

$$V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}}(\tilde{x}(k), 0) \tag{69}$$

Anche se non è stato esplicitamente indicato, le matrici A, W, H e V dipendono dal passo k in cui vengono calcolate.
Si possono quindi scrivere le equazioni relative alla fase di *time update* per il filtro di Kalman Esteso:

$$\hat{x}(k+1|k) = f(\hat{x}(k|k), u(k), 0) \tag{70}$$

$$P(k+1|k) = A_{k+1}P(k|k)A_{k+1}^T + W_{k+1}Q(k)W_{k+1}^T \tag{71}$$

Le equazioni di *measurement update* risultano invece:

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K_k(y(k) - h(\hat{x}(k|k-1), 0)) \tag{72}$$

$$P(k|k) = P(k|k-1) - K_k H_k P(k|k-1) \tag{73}$$

avendo posto come *guadagno del Filtro di Kalman Esteso* la matrice K definita come segue:

$$K_k = P(k|k-1)H_k^T(H_k P(k|k-1)H_k^T + V_k R_k V_k^T)^{-1} \tag{74}$$

7 Modello dinamico del sistema

La costruzione del modello del sistema attraverso una descrizione in spazi di stato è stata fatta prendendo spunto dalle informazioni indicate in [1] poichè che le caratteristiche delle due situazioni di lavoro erano molto simili.
Per il generico sistema a tempo continuo

$$\dot{x}(t) = f(x(t), u(t), w(t)) \tag{75}$$

$$y(t) = h(x(t), u(t), v(t)) \tag{76}$$

si è assunto che:

stato x :
il vettore x rappresentante lo stato del modello risulta essere composto da $2n + 4$ elementi avendo indicato con n il numero di nodi ancora presenti nell'area di lavoro.

Si ha quindi:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ . \\ . \\ . \\ x_{4+2n-1} \\ x_{4+2n} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_v \\ \mathbf{x}_s \end{bmatrix} \tag{77}$$

Il vettore $\mathbf{x}_v$ definito dai primi 4 elementi del vettore di stato (x_1,x_2,x_3,x_4) rappresenta le caratteristiche cinematiche di posizione e velocità *assolute* lungo gli assi cartesiani X e Y del veicolo mobile:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_v \\ \dot{x}_v \\ y_v \\ \dot{y}_v \end{bmatrix} \tag{78}$$

In generale per definire la posizione del robot nel piano di lavoro occorre conoscere posizione e orientamento di quest'ultimo (l'angolo θ di figura (6)); introducendo però la descrizione della velocità sui due assi si è potuto fare a meno di tener conto dell'angolo θ ad ogni posizione del veicolo.

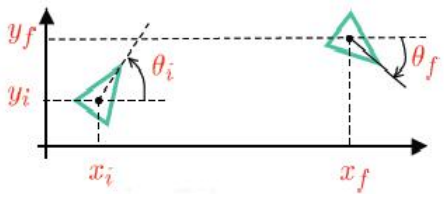


Figura 6: Descrizione del veicolo in funzione di posizione e orientamento.

Le rimanenti $2n$ variabili di stato rappresentano invece le coordinate lungo gli assi cartesiani dei nodi ancora, ovvero:

$$\begin{bmatrix} x_5 \\ x_6 \\ . \\ . \\ x_j \\ x_{j+1} \\ . \\ . \\ x_{4+2n-1} \\ x_{4+2n} \end{bmatrix} = \begin{bmatrix} x_{s_1} \\ y_{s_1} \\ . \\ . \\ x_{s_i} \\ y_{s_i} \\ . \\ . \\ x_{s_n} \\ y_{s_n} \end{bmatrix} \quad \forall\, 1 \leq i \leq n \tag{79}$$

Poichè i sensori da localizzare nell'ambiente non sono in movimento la loro velocità è nulla; pertanto l'introduzione di variabili di stato legate a tali grandezze è totalmente superfluo.

La scelta di queste variabili di stato per la descrizione del modello sono state effettuate imponendo delle *condizioni a priori* sul problema di $Slam$: nel problema generale le conscenze sull'ambiente sono praticamente nulle, quindi non si ha neanche la conoscenza del numero di sensori presenti nell'area. Ciò comporta che:

- la grandezza del vettore di stato x non è nota a priori;
- ogni volta che il robot mobile individua un segnale relativo ad un nuovo nodo fisso, è necessario modificare il vettore di stato aggiungendo 2 nuove variabili (ovvero le componenti x_{new} e y_{new} del nuovo sensore). Ne consegue che la complessità computazionale aumenta.

Alla luce di tali osservazioni si è assunto per lo sviluppo dell'algoritmo risolutivo che **il numero dei nodi ancora presenti nell'ambiente sono noti priori**.

ingresso u :

come ingressi che compongono il vettore **u** sono state scelte le componenti di accelerazione sugli assi X e Y del veicolo mobile:

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \ddot{x}_v \\ \ddot{y}_v \end{bmatrix} \tag{80}$$

I valori che vengono passati in ingresso derivano dalle caratteristiche cinematiche imposte al robot; si è assunto infatti che la traiettoria percorsa dal veicolo sia definita a priori: il robot precorre una traiettoria a spirale partendo dall'angolo in basso a sinistra dell'area di movimento e termina il movimento quando raggiunge il centro del piano.

Il veicolo è stato programmato per effettuare uno spostamento *per passi* ovvero, partendo dalla posizione iniziale, si sposta di una distanza predeterminata, si ferma, si mette in ascolto ed acquisisce i segnali RSSI che sente, dopodichè si muove verso il passo successivo fino a quando non completa il percorso. Questa scelta è stata presa per semplificare l'operazione di ricezione e di elaborazione dei segnali provenienti dai nodi ancora.

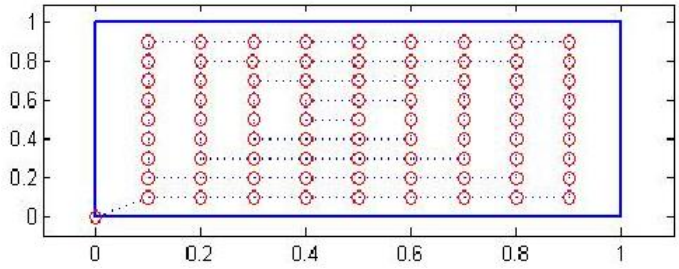


Figura 7: Traiettoria del veicolo mobile.

Per far ciò si è reso necessario definire i profili di velocità e accelerazione ad ogni passo del movimento del veicolo; si è optato allora:

profilo di accelerazione : assumendo che il veicolo compia ad ogni passo uno spostamento $s = 0.1\ m$ in un tempo[3] $T = 0.1\ sec$ e imponendo che il profilo di accelerazione sia ad accelerazione costante come indicato in figura 8, attraverso la formula del moto retilineo uniformemente accelerato:

$$s(t) = s_0 + v_0 t + \frac{1}{2} a t^2 \tag{81}$$

è possibile calcolare il valore di a.
Partendo da posizione iniziale $s_0 = 0$ e con il veicolo fermo (quindi $v_0 = 0$), l'equazione si semplifica ed è possibile determinare il valore dell'accelerazione del robot in fase di accelerazione e decelerazione (per simmetria del profilo adottato i due valori coincidono):

$$a = \frac{\frac{s}{2}}{\frac{T^2}{4}} = 40\ ms^{-2} \tag{82}$$

Avendo costruito il movimento come successione di spostamenti lungo l'asse X o lungo l'asse Y,

[3]tale tempo è legato al *passo di campionamento* imposto nella *fase di discretizzazione* del modello dinamico continuo

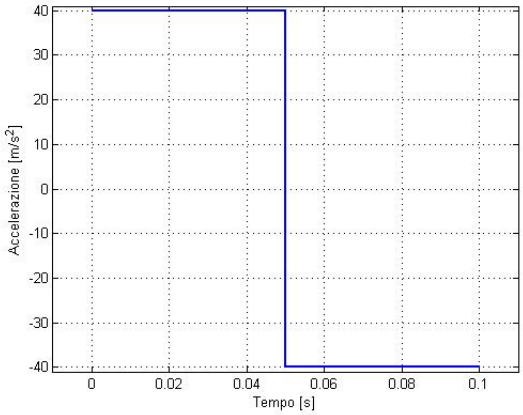


Figura 8: Profilo di accelerazione.

questo valore è quindi il termine numerico direttamente impiegato come ingresso al sistema dinamico.

profilo di velocità : al profilo di accelerazione scelto corrisponde un profilo di velocità di tipo *triangolare* (figura 9). Dall'equazione che lega la velocotà e l'accelerazione in un moto retilineo uniformemente accelerato:

$$v = v_0 + at \tag{83}$$

e sapendo che la condizione iniziale sulla velocità è $v_0 = 0\ ms^{-1}$, il tempo totale su cui avviene l'accelerazione e la decelerazione è $T = 0.1\ sec$ e che dallo studio fatto al punto precedente $a = 40\ ms^{-2}$ si ottiene che la velocità massima acquisita dal veicolo mobile è:

$$V_{max} = 2\ ms^{-1} \tag{84}$$

uscita y :
l'uscita del modello dinamico del sistema è rappresentata dai segnali RSSI generati in funzione delle distanze relative tra veicolo mobile e nodi ancora. Si ha quindi che, impiegando n sensori fissi, il vettore **y** avrà una dimensione pari ad n:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_n \end{bmatrix} = \begin{bmatrix} RSSI_1 \\ RSSI_2 \\ . \\ . \\ RSSI_n \end{bmatrix} \tag{85}$$

rumori v e w :
i termini v e w rappresentano i rumori di misura e di

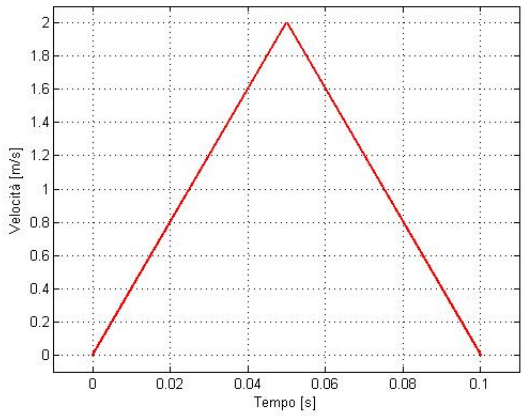


Figura 9: Profilo di velocità.

processo presenti all'interno del modello. Essi vengono modellati come variabili aleatorie gaussiane a media nulla e varianza rispettivamente Q e R:

$$p(w) \sim N(0, Q) \tag{86}$$

$$p(v) \sim N(0, R) \tag{87}$$

Ora è possibile costruire le relazioni tra stato, ingresso e uscita:

- la relazione di stato è descrivibile come

$$\dot{x}(t) = Ax(t) + B_1 u(t)$$

dove è stato posto:

$$A = \begin{bmatrix} \Theta & 0 & . & . & 0 \\ 0 & 0 & & & \\ . & & . & & \\ . & & & . & \\ 0 & & & & 0 \end{bmatrix} \tag{88}$$

$$B_1 = \begin{bmatrix} \phi \\ 0 \\ . \\ . \\ 0 \end{bmatrix} \tag{89}$$

Le matrici Θ e ϕ rappresentano le relazioni tra le caratteristiche del veicolo mobile (posizione, velocità e accelerazione) rispetto alle coordinate delle posizioni degli n sensori; risulta quindi che:

$$\Theta = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{90}$$

$$\phi = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{91}$$

- impiegando le equazioni indicate nella sezione 5 relative alle caratteristiche del canale trasmissivo, risulta che la relazione di uscita è non lineare e dipendente solamente dalle variabili di stato e dal rumore di misura: $y(t) = h(x(t), v(t))$. In particolare avendo posto come stato le posizioni assolute dei nodi ancora e del veicolo mobile (in questa relazione non interessano le componenti di velocità del robot), la distanza relativa tra i-esimo sensore fisso e sensore mobile è data da:

$$d_i^2 = (x_v - x_{s_i})^2 + (y_v - y_{s_i})^2 \tag{92}$$

per cui la funzione di uscita diventa:

$$\mathbf{y} = \begin{bmatrix} RSSI_1 \\ RSSI_2 \\ . \\ . \\ RSSI_n \end{bmatrix}$$

ovvero

$$\mathbf{y} = \begin{bmatrix} P_{TX} + A - 5n_p \log(d_1^2) + \chi_\sigma \\ P_{TX} + A - 5n_p \log(d_2^2) + \chi_\sigma \\ . \\ . \\ P_{TX} + A - 5n_p \log(d_n^2) + \chi_\sigma \end{bmatrix} \tag{93}$$

Attraverso il materiale raccolto si è determinato che le costanti entranti in gioco sono:

- $P_{TX} = 0$;
- $A = 18.2$;
- $n_p = 2.18$;
- $var(\chi_\sigma) = 6.03$.

8 Discretizzazione

Dato il seguente sistema lineare a tempo continuo $G(s)$:

$$\begin{cases} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{cases} \quad (94)$$

attraverso l'inserimento di un filtro *zero-order holder* $H_0(t)$ ed un *campionatore ideale* con *periodo di sampling* T rispettivamente a monte e a valle del sistema continuo si ottiene il sistema a tempo discreto mostrato in figura 11.

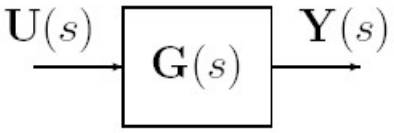


Figura 10: Sistema a tempo continuo.

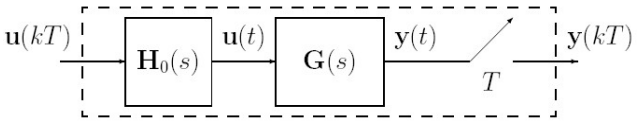


Figura 11: Discretizzazione del sistema a tempo continuo: si può osservare il blocco *campionatore* T ed il *filtro interpolatore* $H_0(t)$.

I due nuovi blocchi elaborano i segnali di ingresso e di uscita del sistema $G(s)$ come segue:

zero-order holder $H_0(t)$:

l'interpolatore *"raccorda"* i valori discreti che osserva dal segnale $u(kT)$ in ingresso; questo significa in un certo senso che il filtro riesce a "inventare" i valori non presenti nel segnale di ingresso.
Dalla teoria dei segnali è noto che sotto la *condizione di corretta interpolazione*, ponendo $u(kT)$ come segnale originario e $\tilde{u}(t)$ ($t \in$ **R**) come segnale interpolato valutato negli istanti kT:

$$\tilde{u}(kT) = u(kT) \quad (95)$$

il segnale in uscita dal $H_0(t)$ risulta *raccordato* in maniera opportuna (figura 12). La tipologia di interpolazione dipende dalla risposta impulsiva del fitro, e per filtri interpolatori ZOH questa è:

$$g_0(t) = rect(\frac{t - \frac{1}{2}T}{T}) \quad (96)$$

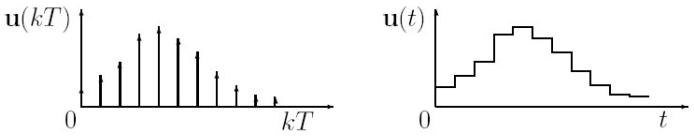


Figura 12: Segnale di ingresso (segnale a sinistra) e segnale di uscita (a destra) del filtro interpolatore ZOH.

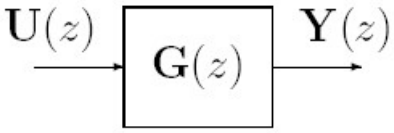
sampler T :

l'operazione di campionamento permette di ottenere un segnale discreto $y(kT)$ partendo da un segnale continuo $y(t)$ ($t \in$ **R**) secondo la relazione:

$$y_{campionato}(kT) = y_{continuo}(kT) \quad (97)$$

Il comportamento ingresso-uscita del sistema complessivo è quello di un sistema tempo discreto $G(z)$ avente come relazioni:

$$\begin{cases} x((k+1)T) &= Fx(kT) + Gu(kT) \\ y(kT) &= Hx(kT) \end{cases} \quad (98)$$



Le matrici F, G e H vengono determinate in funzione delle matrici del sistema continuo $G(s)$ di partenza:

$$F = e^{AT} \quad (99)$$

$$G = \int_0^T e^{A\sigma} B d\sigma \quad (100)$$

$$H = C \quad (101)$$

9 Parte sperimentale - e-puck

Per provare quanto sviluppato in parte simiulativa con il software $Matlab$, é stato impiegato un robot mobile come nodo in movimento e alcuni sensori di distanza come nodi ancora.
Per quanto concerne il veicolo mobile, è stato utilizzato un *"e-puck"*, distribuito da Epfl (École polytecnique federale de Lausanne).

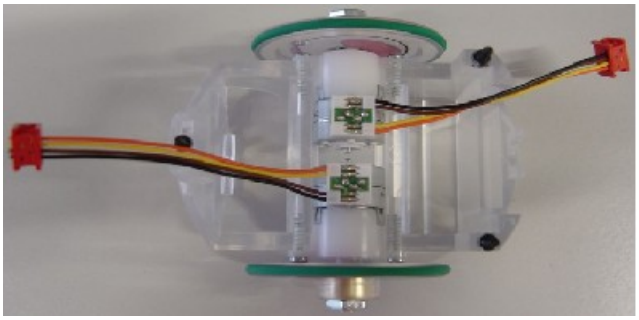


Figura 13: Telaio e batteria dell'e-puck.

Sviluppato su un diametro di 70mm, é costituito di un telaio plastico che sostiene due motori, il circuito stampato e la batteria (vedi figura 13).
Esso monta motori passo-passo: con ingranaggi di riduzione, ciascun motore dispone di 20 passi per la rivoluzione mentre per la marcia vi é predisposta una riduzione pari a 50:1. L'ultimo asse sostiene direttamente la ruota.
Le ruote aventi un diametro di 41mm ciascuna e distanti tra loro 53mm si fissano all'asse mediante un mandrino a quattro denti. Esse sono poi avvolte da pneumatici costituiti da anelli in gomma. La massima velocitá delle ruote é di circa 1000 passi/s, che corrisponde ad una rivoluzione della ruota al secondo.

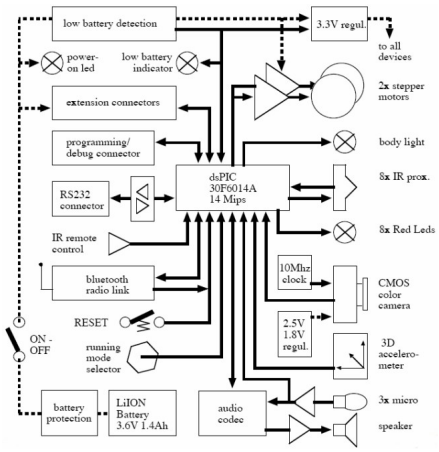


Figura 14: Schema elettrico del veicolo mobile.

L'elettronica come si puó vedere dallo schema elettrico (figura 14) é costituita dalle seguenti componenti:

- Un unico alimentatore, tutto funziona a 3,3V eccetto la macchina fotografica che necessita di altri 1,8V.
- Un connettore di estensione per inserire nuove funzionalitá.
- Una porta seriale RS232 e un interfaccia bluetooth per comunicare con un computer host.
- La batteria ricaricabile LiION, di capacitá 5Wh.
- 8 sensori di prossimitá a raggi infrarossi posizionati attorno al robot.
- Un accelerometro.
- Tre microfoni e un altoparlante per la cattura e la generazione del suono.
- Una fotocamera con risoluzione 640x480 pixel a colori.
- 8 led rossi posizionati attorno al robot.
- Un processore dsPIC che usa compilatori standard quali C, C++ con una buona potenza di calcolo; con molte interfacce e un nucleo dsp molto efficiente che consente l'elaborazione dei dati.

Figura 15: Vista dell'e-puck.

Per quanto riguarda la comunicazione il microcontrollore dsPIC é dotato di due porte. La prima collegata ad un chip bluetooth e la seconda fisicamente visibile grazie ad un connettore micromatch.
Per non rendere troppo lenta la parte sperimentale e visto che muovendosi in ambienti "troppo insidiosi" il robot non funzionava, si é deciso di lavorare su un perimetro di 50x50cm, si é fatta inseguire una traiettoria a spirale verso l'interno di 10cm per passo ad una velocitá tale da permettere ai mote la comunicazione.
Sotto l'ipotesi di comunicazione adottata (ovvero quando l'e-puck é fermo), si sono fatti percorrere i 10cm in 1s, con

una sosta di 7s ogni passo.
Tutto questo é stato realizzato nei seguenti passi:

- Programmazione in C: grazie al software presente nei computer di Navlab "MPLAB IDE v8.00" e ad alcune librerie messe "Open Source" da Francesco Sardara sono stati creati alcuni cicli di codice per determinare la traiettoria.

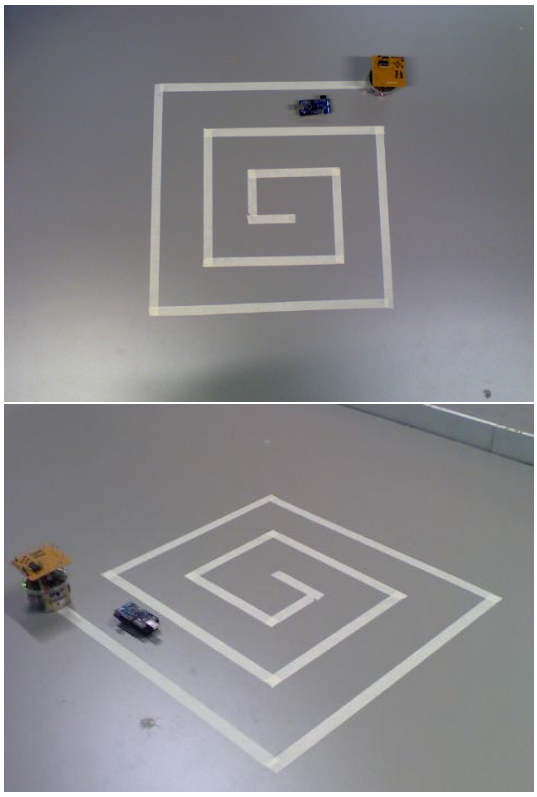


Figura 16: Traiettoria impostata per il veicolo mobile.

Questi sono nel dettaglio undici cicli for, che indicano il numero di segmenti di cui é composta la traiettoria.
Una volta determinate all'inizio del programma le costanti:

  - $wait = 7000$: indica il tempo in cui sta fermo il robot dopo aver percorso un segmento di 10cm.
  - $angle = 90$: indica di quanti gradi deve girare il robot una volta compiuto un intero segmento.

Si procede poi con la determinazione delle operazioni legate alla programmazione e temporizzazione dei Tmote:

1. si indicizza una variabile i da incrementare di una unitá ogni ciclo; tale indice ha un range che va da 0 alla lunghezza del segmento (espressa in cm e divisa per 10 poiché il robot deve compiere passi di 10cm);
2. si setta la velocitá a $10cm/s$ attraverso il comando $setV(10)$;
3. si imposta lo spazio si che deve compiere ad ogni passo ($10cm$);
4. si annida un ciclo while nel ciclo for, attraverso il quale é possibile fermare il robot per 7s, tempo necessario ai Tmote per effettuare le misure. Questo é stato realizzato impostando la velocitá a $0.1cm/s$ e attraverso un ulteriore comando presente nelle librerie, settando il tempo di percorrenza in ms (costante $wait$).

Con queste operazioni il robot compie un intero segmento, alla fine del quale gira a sinistra di $90°$ (comando $turn('left', angle)$). Introducendo ricorsivamente i cicli ed i comandi suddetti, viene percorsa tutta la traiettoria.

- Connessione dell'e-puck via bluetooth al computer.
- Con il software "tinybldWin" scrittura nella memoria dell'e-puck del codice costruito con "MPLAB IDE v8.00".

A questo punto sull'e-puck é stata montata una struttura in grado di trasportare il t-mote.

10 Parte sperimentale - motes

10.1 Analisi sperimentale del problema

Il problema dello SLAM viene affrontato sperimentalmente con il movimento del nodo mobile che traccia la posizione di alcuni nodi ancora posizionati all'interno di uno spazio definito.

I nodi ancora hanno una posizione fissa nota all'inizio dell'esperimento. Il numero di nodi ancora è noto e rientra all'interno di un valore massimo specificato nella variabile `uint16_t numnode`.
Ogni nodo ancora inoltre ha un unico ID differente da tutti gli altri nodi ancora. Al fine di coprire tutto lo spazio il nodo mobile deve eseguire un opportuna e predefinita traiettoria.

Determinata la posizione dei vari nodi il robo sarà quindi in grado di determinare la propria posizione rispetto agli stessi. Per ogni punto della traiettoria il nodo mobile determina la propria distanza dai vari nodi ancora tracciando la potenza di una trasmissione wireless con ogni nodo.
La potenza della radio CC2420 è stata impostata al suo livello massimo corrispondente a $0dBm$; questo ci permette di avere un segnale di intensità più forte, meno effettuo da

rumore e inoltre permette di avere le comunicazioni a una distanza maggiore.

Nello svolgimento del progetto per la comunicazione fra i *nodi ancora* e *robot mobile*, si è scelto di utilizzare una comunicazione in broadcast poichè, pur essendo noto a priori il numero dei sensori fissi, potrebbero essercene in numero superiore.

Comunicazione Broadcast Per broadcasting si intende la trasmissione di informazioni da un sistema trasmittente ad un insieme di sistemi riceventi non definito a priori.

L'esempio più classico è costituito da un trasmettitore radio di grande potenza e da un gran numero di ricevitori montati nelle automobili o nelle case. In questo caso, tutti i ricevitori situati nell'area di copertura del trasmettitore riceveranno il segnale, e il trasmettitore non potrà sapere esattamente con chi ha comunicato.
La trasmissione broadcasting è unidirezionale. Le informazioni sono inviate dal trasmettitore ai ricevitori, senza canale di ritorno e senza sicurezza che le stesse riescano ad essere consegnate. In opposizione alle comunicazioni broadcasting, ci sono le comunicazioni punto-punto o bidirezionali.
Nel mondo della radio e della televisione, il termine broadcasting indica anche il livello di qualità richiesto per trasmissioni commerciali a grande diffusione, e gli strumenti professionali utilizzati per ottenerlo.
Nelle reti di calcolatori, il termine ha un significato simile: un pacchetto inviato ad un indirizzo di tipo broadcast verrà consegnato a tutti i computer collegati alla rete (ad esempio, tutti quelli su un segmento di rete ethernet, o tutti quelli di una sottorete IP).

10.2 Componenti Hardware

Negli ultimi anni le reti di sensori wireless si stanno diffondendo in maniera sempre maggiore date le molteplici funzioni per cui possono essere usate.
Nel progetto i nodi sono *T-moteSky* prodotti dalla Moteiv Corporation, messi a disposizione dalla Facoltà di Ingegneria, che offrono buone prestazioni. Questa tipologia di sensori, a cui fanno parte i *T-moteSky* della Moteiv permettono il rilevamento di posizione dei nodi *ancora* e del *nodo mobile*. Un grande vantaggio che offre la rete di sensori wireless è quello di causare il minimo ingombro in termini di spazio in quanto non cablata.

I sensori sono alimentati da batterie stilo, quindi con un uso ottimale della rete prevede che esse si scarichino il più lentamente possibile. I sensori sono per la maggior parte del tempo spenti e si accendono solo per rilevare le misure e scambiare i dati.

La piattaforma è dotata di una flash RAM che opera a 40 MHz e fornisce 1024 kB per memorizzare dati divisi in 16 segmenti di 64 kB.
I nodi *T-moteSky* sono equipaggiati con un microcontrollore MSP430 prodotto dalla Texas Instruments il quale ha a disposizione 48 Kbyte di memoria Flash dedicata per il software e 10 kB di SRAM.
Ha inoltre:

- un oscillatore (DCO) che opera sopra gli 8 MHz
- e un oscillatore al quarzo utile alla calibrazione del primo

É dotato di 8 porte ADC interne e altrettante esterne, necessarie per leggere i valori dei sensori o dei livelli di batterie.
I *T-moteSky* sono mote a bassissimo consumo sui quali oltre al microcontrollore MSP430 prodotto dalla Texas Instruments (8 MHz, 10 kB di RAM, 48 kB di memoria Flash) è montato anche un chip radio CC2420 prodotto dalla Chipcon.
I componenti più rilevanti per l'applicazione:

- **Chip radio**: Le comunicazioni radio sono rese possibili dal chip radio CC2420 della Chipcon ed è controlato dall'MSP430 tramite il bus SPI.
  Il bus SPI è un sistema di comunicazione tra un microcontrollore e altri circuiti integrati o tra più microcontrollori.
  La radio lavora nell'intorno dei 2.4 GHz con una modulazione O-QPSK, ma è possibile selezionare tra 16 canali differenti numerati da 11 a 26, con i quali ci si può spostare dalla frequenza base con step di 5 MHz per canale.

  Ad ogni segnale ricevuto è possibile associare un valore, l'RSSI, legato alla potenza del segnale ricevuto (Received Signal Strength).
  L'intero algoritmo è basato sul calcolo dell'RSSI; il range di valori dell'RSSI restituiti dal chip è [-60, 40] dBm.
  L'RSSI può essere ottenuto dal ricevente da un qualsiasi pacchetto trasmesso dai nodi fissi. In figura (**??**) si rappresenta la mappa tipica tra la potenza del segnale ricevuto e l'RSSI (vedi figura 4).

- **Antenna radio**: I *T-moteSky* offrono un'antenna integrata di tipo inverted-F microstrip design (antenna a banda stretta a forma di F rovesciata), composta da una pista di rame posta su un lato della scheda coperta da uno strato di vernice che funge da dielettrico, mentre sul lato opposto della scheda un substrato di rame ha potenziale di terra.
  L'antenna offre un campo di irradiamento pressochè circolare con poche irregolarità.

Nel progetto non è stato necessario ricorrere all'antenna opzionale esterna prevista per i mote.

Del nodo *T-moteSky* è noto anche il diagramma di radiazione dell'antenna che viene riportato in figura (17). Dalla figura si osserva come il campo prodotto dall'antenna utilizzata nel nodo possa essere considerato sostanzialmente isotropo.

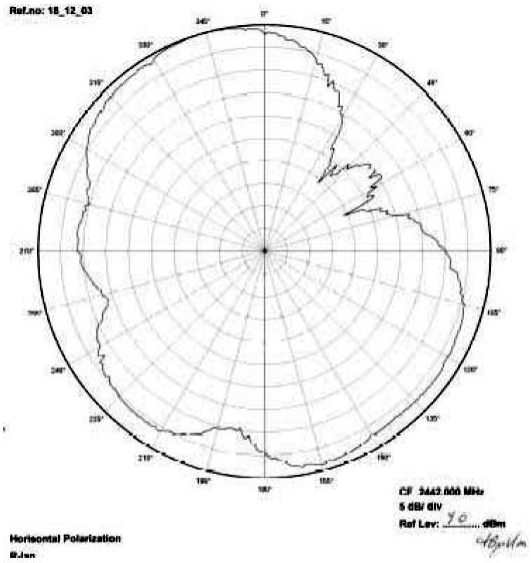


Figura 17: Diagramma di radiazione del nodo *T-moteSky*, piano verticale

- **Alimentazione**: L'alimentazione può essere fornita da due batterie di tipo AA oppure tramite il collegamento USB.
  Per un funzionamento conforme alle direttive di programmazione, è necessario che la tensione di alimentazione sia compresa tra i 2.1 V ed i 3.6 V, e non sia inferiore ai 2.7 V in fase di riprogrammazione.
  Il collegamento alla porta USB di un PC alimenta il mote a 3 V. É fondamentale che sia sempre superato il limite di alimentazione minima per consentire una corretta interpretazione dei dati forniti da ogni nodo.

In figura (18) viene rappresentato il nodo *T-moteSky*, invece nelle figure (19) e (20) si mostra parte frontale e retro del modulo *T-moteSky*.

Figura 18: *T-moteSky*

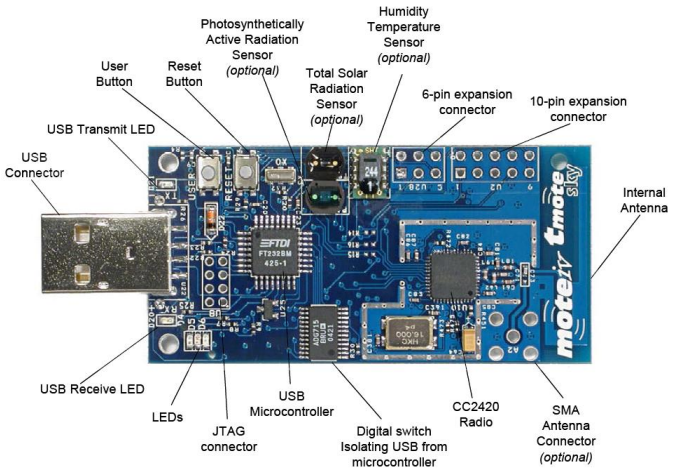


Figura 19: Parte frontale del *T-moteSky*

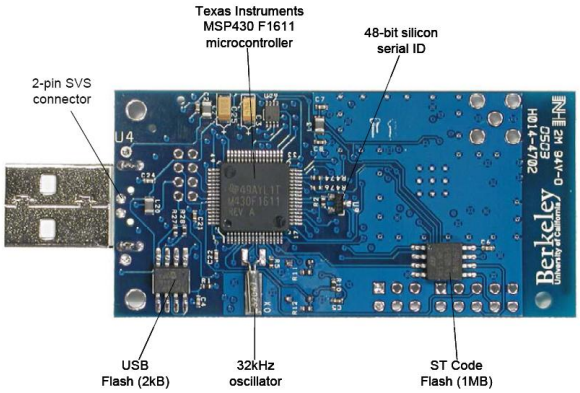


Figura 20: Retro del *T-moteSky*

Per quanto riguarda la potenza in trasmissione, sono disponibili 32 diversi livelli, quantizzati in 8 possibili valori. Questi valori vengono elencati in tabella dal più basso (3) al più elevato (31).

Livello	Consumo [mA]	Pot.di trasmissione[dbm]
3	8.5	-25
7	9.9	-15
11	11.2	-10
15	12.5	-7
19	13.9	-5
23	15.2	-3
27	16.5	-1
31	17.5	0

10.3 TinyOS

TinyOS è un sistema operativo open-source svilupatto dall'università di Berkeley in California. In sostanza, *TinyOS* consiste in alcuni moduli software che realizzano le funzionalità di base per il controllo del nodo sensore come:

- l'accesso ai registri del micro-controllore;
- la lettura e la scrittura della memoria EPROM;
- la gestione dell'interfaccia radio per la trasmissione e la ricezione dei dati;
- la generazione dei numeri pseudo-casuali e nodo ancora.

Non esiste un *kernel* che gestisce le risorse disponibili, dividendole tra più applicazioni, ma solo un *scheduler* che si limita a esegurie in sequenza i task secondo una politica *First In First Out (FIFO).*
In altri termini i vari task sono eseguiti nell'ordine con cui sono generati, consentendo allo *scheduler* di interrompere l'esecuzione di un task nel caso si verifichi un evento con priorità maggiore. Quando non vi sono più funzioni in attesa di esecuzione lo *scheduler* porta il processore a riposo e attende la segnalazione di un nuovo evento per riprendere l'esecuzione.
I moduli elementari forniti dal sistema operativo possono quindi venire utilizzati dai programmatori per sviluppare applicazioni o funzionalità avanzate.
Il codice di *TinyOS* è scritto in *NesC*, una variante del linguaggio C concepita per i *sistemi embeddeded* e ottimizzata per la programazzione dei sensori.
NesC è anche utilizzato per lo sviluppo delle applicazioni, rendendo cosi possibile un'integrazione molto spinta con il sistema operativo.
Nel codice eseguibile che viene generato dal compilatore, inoltre vengono inclusi solo i moduli del sistema operativo effettivamente utilizzati dall'applicazione.
Ciò consente di ridurre al minimo l'ingombro del programma che dovra essere traferito ed eseguito dal nodo sensore, condizione indispensabile in sistemi con uno spazio di memoria limitato a qualche decina di kB.

Pertanto *TinyOS* si differenzia in modo sostanziale dai sistemi operativi tradizionali (general purpose), integrandosi direttamente con le applicazioni sviluppate.
Esso semplifica notevolmente lo sviluppo di applicazioni per WSN, mascherando le funzionalità di più basso livello della piattaforma per consentire al programmatore di concentrarsi sugli aspetti d'interesse. Tuttavia, per raggiungere prestazioni elevate è necessaria un'analisi minuziosa delle operazioni elementari eseguite dal programma al fine di individuare le criticità.
Nonostante *TinyOS* sia il più popolare e diffuso sistema operativo per WSN, non è di certo l'unico, a dire di molti, neppure il migliore.
Si è deciso di utilizzare la versione più recente del software TINYOS, la 2.0.1, a fronte dei numerosi perfezionamenti apportati e dei bug opportunamente corretti, rispetto alla versione 1.0.

11 Implementazione algoritmo sui *T-mote*

Allo scopo di testare le simulazioni viste in precedenza con *Matlab*, i dispositivi *T-mote* sono stati programmati in linguaggio *NesC*.
Per questo vengono realizzati tre moduli con le rispettive configurazioni:

- Anchor.nc $\Rightarrow$ per i nodi fissi;
- Mobile.nc $\Rightarrow$ per il nodo mobile;
- BaseSt.nc $\Rightarrow$ per il node BaseStation collegato tramite USB al PC.

11.1 Algoritmo ad alto livello

La simulazione è basata sul movimento del nodo mobile, montato sul robot e-Puck.
Il programma del robot stabilisce una traiettoria, definita nella sezione teorica del progetto, che viene percorsa con spostamenti lineari di 10 cm/s.
L'algoritmo esegue la procedura nel seguente ordine:

1. Al termine di ogni tratto della traiettoria percorso dal robot, il nodo mobile interroga in modo sequenziale tutti i nodi ancora.

2. Nel caso in cui un nodo ancora *N* risponda all'interrogazione il nodo mobile richiede ad esso l'invio di un determinato numero di messaggi, fino ad un predefinito messaggio di STOP, necessari per effettuare la media della potenza di trasmissione del nodo *N*. Questo è necessario per ottenere una buona stima della reale potenza trasmessa.

3. Il nodo mobile procede quindi all'invio alla BS del dato (indice di posizione, indice del nodo ancora interrogato, potenza).

4. Il nodo BS ricevuto il pacchetto lo ritrasmette tramite comunicazione seriale al PC, che procede quindi al salvataggio di tutti i dati sperimentali.

5. Il nodo mobile ripete quindi l'interrogazione sul nodo ancora *N+1* fino ad aver interrogato, indipendetemente dalla risposta, tutti i nodi.

6. Il nodo mobile procede quindi al successivo tratto di traiettoria e ripete l'algoritmo fino al completamento della stessa.

11.2 Mobile.nc

Questa configurazione implementa l'algoritmo per il nodo mobile. Di particolare importanza sono le variabili denominate:

- `uint16_t counter` che identifica il nodo ancora sul quale effettua l'interrogazione, è di fatto un contatore che può assumere valori compresi tra 0 e `numnode`;
- `uint32_t posIndex` che identifica con un numero intero crescente la posizione attuale del nodo mobile lungo i vari tratti della traiettoria ;
- `int16_t rssmean` che contiene la media del RSSI(su un numero predefinito di messaggi ricevuti) ricevuto dal nodo su cui si stà effettuando l'interrogazione.

La comunicazione tra nodo mobile e nodo ancora e tra nodo mobile e BS avviene sfruttando dei pacchetti dati specifici, rispettivamente:

- `message_t pkt` che contiene:
  - `nx_uint16_t nodeid` ID del nodo trasmittente;
  - `nx_uint16_t counter` ID del nodo ricevente;
  - `nx_bool START` una variabile booleana di stato che abilita la trasmissione.
- `message_t pktAnchor` che contiene:
  - `nx_uint16_t nodeId` ID del nodo ancora sul quale è stata effettuata la rilevazione della potenza trasmessa;
  - `nx_uint32_t posIndex` indice della posizione del nodo mobile nella è stata effetuata l'interrogazione;
  - `nx_uint32_t power` il valore medio della potenza ricevuta dal nodo sul quale è stata effetuata l'interrogazione.

L'ID del nodo mobile è unico e impostato a 0x00.
Il nodo mobile è regolato da due timer:

- `Timer0` (di periodo `TIMER_PERIOD_MILLI` = 800ms) che regola l'invio dei pacchetti con un singolo nodo ancora, espletando inoltre la funzione di WatchDog timer quando un nodo ancora non può essere raggiunto.
- `Timer1` (di periodo `TIMER_PERIOD_MOVEMENT` = 4000ms) che viene avviato con funzionamento periodico di periodo superiore al tempo necessario ad inviare tutti pacchetti con tutti i nodi ancora, questo ha lo scopo di sincronizzare il funzionamento con il movimento del robot e-puck.

I seguenti eventi regolano il funzionamento del codice sul nodo mobile:

- `Boot.booted()` viene eseguito all'accensione del T-mote e richiama l'accensione del modulo radio;
- `AMControl.startDone(error_t err)` segnala che l'operazione è avvenuta con successo, invocando il comando start() viene eseguita l'accesione della radio. Inoltre vengono avviati i due timer `Timer0` e `Timer1`.
- `Timer0.fired()`: Quando questo evento viene invocato aggiorna l'ID del nodo ancora su cui effettuare l'interrogazione ed invia a questo un messaggio di tipo `pkt` dove la variabile booleana `START` è settata per imporre l'avvio della trasmissione dal nodo ancora indirizzato da `counter`, mentre il `nodeid` è 0x00 (ID del nodo mobile). Il nodo mobile dopo aver inviato il pacchetto attende la ricezione di una trasmissione dal nodo ancora indirizzato in precedenza tramite un evento `receive`.
- `Receive.receive` Il codice contenuto in questo evento viene eseguito quando la radio del nodo mobile riceve un pacchetto di tipo `pkt`. In questo caso determina il valore del RSSI del pacchetto ricevuto e lo somma ad una variabile temporanea `rsssum`.

Nel caso in cui sia stato ricevuto il numero di pacchetti necessario per effettuare la media (nel caso in esame il numero è stato posto uguale a 40) del RSSI vengono eseguite le seguenti operazioni:

1. la variabile `rsssum` viene divisa per il numero di pacchetti e salvata in `rssmean`;
2. `rsssum` viene reinizializzata a 0;
3. viene preparato il pacchetto `pktAnchor` per la trasmissione del dato ricevuto al nodo BS;
4. viene inviato un pacchetto `pkt` al nodo ancora, di cui è appena stata calcolata la potenza media, con variabile di stato `START` resettata per terminare la trasmissione di pacchetti verso il nodo mobile;
5. infine nel caso in cui le procedure precedenti abbiano avuto successo il pacchetto `pktAnchor` viene trasferito alla BS attraverso un opportuno TASK.

Quando è stata effettuata (o tentata) la comunicazione con tutti i nodi ancora, il `Timer0` viene fermato.
Nel periodo successivo del `Timer1` il nodo mobile non effettua alcuna trasmissione, in quanto il robot e-puck è in movimento. Il `Timer0` viene nuovamente inizializzato al successivo evento generato dal `Timer1`. Il funzionamento del codice procede quindi in modo ciclico e deve essere terminato dall'utente.

11.3 Anchor.nc

La seguente configurazione implementa l'algoritmo per il nodo ancora. L' algoritmo di comunicazione è basato sullo stesso tipo di messaggio `pkt`. Quando il nodo ancora viene avviato tramite l'evento `Boot.booted()`, il modulo radio del T-mote viene avviato.

Il sistema rimane in attesa di un evento di tipo `Receive.receive`. Quando tale evento si verifica con un pacchetto trasmesso dal nodo mobile (`nodeid`=0x00), con il valore della variabile `counter` corrispondente all'ID del nodo ancora e con `START` settata viene avviato un timer `Timer2` con funzionamento periodico e periodo `PERIOD_MILLI_ANCHOR` del valore di 15ms. Quando il timer `Timer2` scatta viene inviato al nodo mobile un messaggio contenente il proprio `nodeid`; essendo questo messaggio necessario alla sola identificazione della potenza di ricezione del messaggio (calcolo dell'RSSI) gli altri campi non sono specificati e rimangono vuoti.

L'invio di messaggi da parte del nodo ancora viene infine interrotto con lo stop del timer alla ricezione di un messaggio dal nodo mobile con variabile `START` resettata. Il nodo ancora rimane quindi in attesa di un nuovo messaggio ad esso indirizzato.

11.4 BaseSt.nc

All'avvio del nodo BS viene innanzitutto abilitata la comunicazione seriale `SerialCtrl.start`. Viene quindi inizializzato anche il modulo radio. Alla ricezione di un messaggio di tipo `pktAnchor` il nodo BS aggiunge i dati ricevuti alla propria coda `Fifo` e lancia un task `sendDataMsg()`.

Il task `sendDataMsg()` invia attraverso la comunicazione seriale un pacchetto `serialpkt` con il contenuto del primo elemento nella coda. L'adozione della coda non è necessaria, ma permette di rendere non sequenziale la ricezione del pacchetto via radio e l'invio del dato tramite porta seriale al PC.

La comunicazione tra BaseStation (BS) e computer avviene tramite l'interfaccia seriale della porta USB.
Per la ricezione dei dati esadecimali trasmessi dalla BS è stato realizzato un *microprogramma* in C++ che permette di estrapolare la parte contenente i dati di interesse:

- indice di posizione;
- identificativo del nodo ancora;
- potenza dall'intero pacchetto trasmesso

I dati inviati tramite comunicazione seriale ad una velocità di 115200 bps vengono quindi separati dai dati tag di apertura e chiusura del pacchetto seriale e salvati su di un file di testo *.mat* da un apposito microprogramma scritto in C++. Il file verrà elaborato dal codice realizzato in Matlab. I dati vengono organizzati in una tabella nella forma:

indice di posizione(posIndex)	**power**	**nodeId ID**

11.5 Slam.h

Entrambi i moduli fanno riferimento ad un C++ Header file comune, `Slam.h`, al fine di condividere costanti e tipi di messaggi.

Nelle figure (21) e (22) si mostra il funzionamento dei vari timer e lo schema completo della struttura hardware con i protocolli di comunicazioni tra i nodi ancora, mobile, base-station e PC.

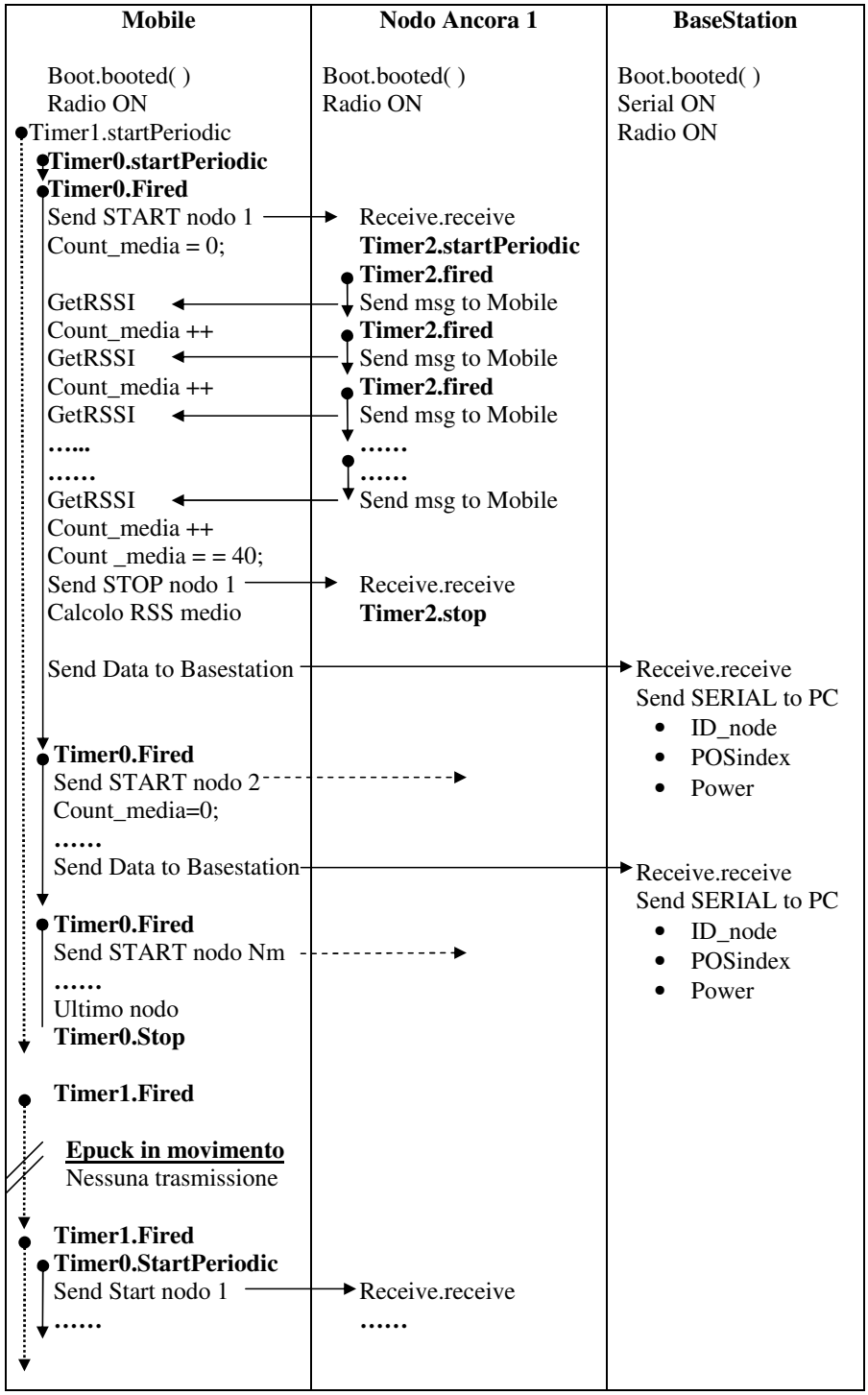


Figura 21: Timers

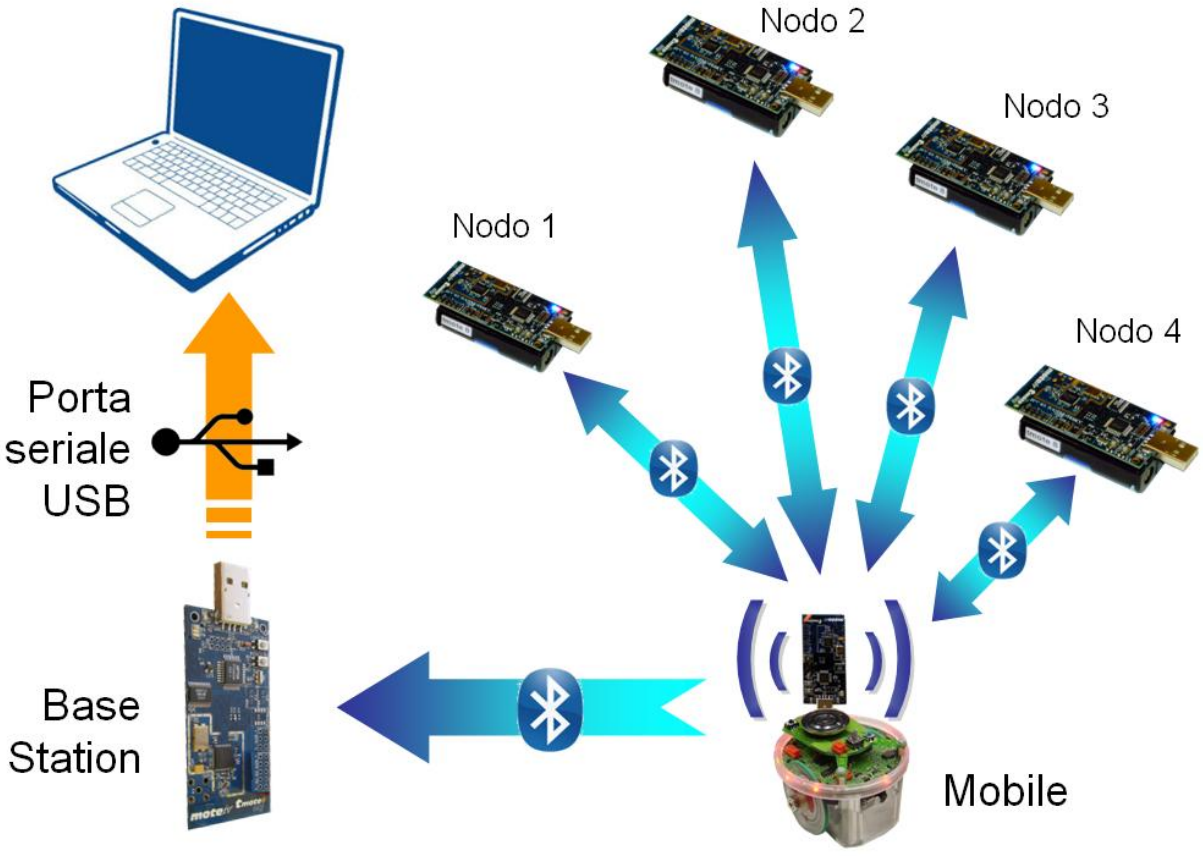


Figura 22: I protocolli di base tra nodi ancora, mobile, base-station e PC

11.6 Metodologia delle prove sperimentali

Visto il materiale a disposizione in laboratorio si è deciso che le prove dovevano essere effettuate secondo le seguenti caratteristiche:

- A causa del dislivello presente nel piano di lavoro si è dovuto limitare la dimensione del piano di lavoro, inoltre vista la tempistica per le misure e per il movimento sì è dovuto decrementare il numero di passi totali compiuto dal veivolo mobile. Per garantire tutto ciò si è ridotto il piano di lavoro a un quadrato di dimensione 50x50 cm in cui il veicolo può assumere fino a 36 posizioni differenti.
- Si è pensato di rilassare il problema e non richiedere la ricostruzione tempo reale di ciò che accadeva sul piano di lavoro. Quindi è stato possibile suddividere il problema in due parti, una di raccolta dei dati e una di ricostruzione attraverso Matlab. Questa supposizione ha semplificato notevolmente il lavoro e ha richiesto in pratica un solo applicativo che salvava i dati all'interno di un file testo.
- Il veicolo mobile come già accennato si muove per $1s$ percorrendo $10cm$ e poi attende $7s$ affinchè per ogni nodo vengano effettuate 40 misure, che poi verranno mediate in modo da ridurre i disturbi dovuti alla riflessione del segnale. A questo punto il mote a bordo del veicolo mobile trasmette i dati rilevati alla Base Station che li scrive nel file di testo con il formato precedentemente illustrato. I motes trasmetto con potenza di trasmissione di $0dBm$

12 Algoritmi Proposti e Simulazioni del Modello

Prima di iniziare con la trattazione degli algorimi sviluppati, è bene tener presente alcune considerazioni iniziali, che verranno *sempre* ritenute valide.
In primis il veicolo mobile è abilitato alla comunicazione via radio solo quando è fermo. Inoltre la traiettoria del veicolo mobile deve essere sempre preassegnata e immutabile.
Il primo algoritmo, vista la sua inefficacia, non è stato adattato alla situazione del laboratorio. Nei rimanenti si è preferito adattare l'ambiente di lavoro alla situazione presente nel laboratorio.

12.1 Localizzazione con Kalman: *Run.m*

Inizialmente si è trattato il problema di localizzare n nodi fissi usando solo la conoscienza della posizione di un nodo mobile in grado di ricevere il valore dell'RSSI dei nodi ancora posti nell'ambiente.

Si è proceduto con la creazione di un ambiente di lavoro (file *mappa.m*); esso genera un vettore contenente numeri complessi che rappresentano la posizione in coordinate cartesiane dei nodi fissi.
Un altro elemento importante per simulare l'algoritmo di risoluzione del problema è il movimento del veicolo (file *movimentoveicolo.m*); questo genera un percorso a spirale rettangolare (rappresentato da un vettore di punti) per il nodo mobile all'interno di un quadrato di lato unitario.
Si poi proceduto col costruire una funzione per riprodurre il comportamento del canale (file *modellomisura.m*), che genera un vettore di lunghezza opportuna di misure di RSSI in base alla distanza tra il nodo mobile ed un nodo ancora e un valor medio necessario come valore iniziale per il filtro di Kalman successivo.
Una volta creato l'ambiente si procede con l'algoritmo risolutivo vero e proprio. Esso prevede che iterativamente il veicolo mobile esegua un passo, si fermi ed effettui una misura generando un vettore di valori RSSI su tutti i nodi. Solo se il nodo fisso è entro un una certa distanza dal nodo mobile la misura viene utilizzata per i passi successivi altrimenti viene scartata.
A misura effettuata, per ogni nodo viene costruita in base ai parametri del canale la matrice di varianza della misura R (ponendo Q=**0**). A questo punto si applica il filtro di Kalman esteso (file ekfcao.m), che esegue la predizione ad un passo e la stima della distanza, cioè lo stato, ricavata dalla formula inversa del modello del canale. Questa funzione deve essere ripetuta ricorsivamente un numero di volte pari alla lunghezza del vettore di misura dell'RSSI fornito dalla funzione che modellizza il canale.
Si dispone ora di un valore sufficientemente accurato della distanza, si sa quindi che il nodo in analisi è posto in una circonferenza centrata nel nodo mobile di raggio pari alla distanza appena rilevata. Una volta ottenute almeno tre di queste circonferenze centrate in tre posizioni diverse del veicolo mobile, non allineate tra loro (altrimenti non si ha unicità nella localizzazione), per triangolazione si può determinare la posizione del nodo fisso. Tante più misure si hanno a disposizione tanto più precisa è la stima della posizone del nodo ancora.
Tuttavia ci si è resi conto che a causa del rumore e del metodo risolutivo si determina una nuvola di punti, all'interno della quale c'è anche la posizione "reale", ma tale soluzione non è minimamente accettabile. Per questo motivo l'algoritmo è stato ritenuto inefficiente e scartato.

L'algoritmo Run per funzionare, ha bisogno delle seguenti variabili

- *dimpiano*: è la dimensione del pinao di lavoro;
- *n*: è il numero di nodi fissi da localizzare;
- *z*: può assumere due valori: '*c*' e '*d*', che

rispettivamente indicano una distribuzione causale o deterministica dei nodi fissi;

- *r*: è il raggio entro il quale la misura viene utilizzata;
- *nodilato*: è il numero di nodi massinmo che può essere dislocato sul perimetro del rettangolo interno;

Di seguito viene presentato un esempio dei risultati ottenibili, in particolare nella prima figura viene visualizzato l'ambiente di lavoro, mentre nella seconda figura viene presentato l'esito dell'algoritmo, i punti verdi rappresentano la posizione "reale", mentre le croci nere rappresentano i punti di intersezione delle circonferenze.
Questo è quanto ottenuto ponendo:

- *n* = 20;
- *dimpiato* = 1;
- *z* = 'd';
- *r* = 0.15;
- *nodilato* = 4.

Si nota come alcuni punti sono fuorvianti, così come alcuni nodi sono individuati da un numero esiguo di intersezioni.

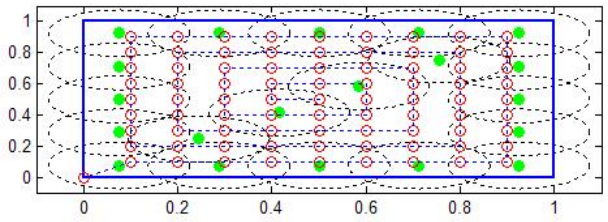


Figura 23: Mappa generata dal file *Run.m*.

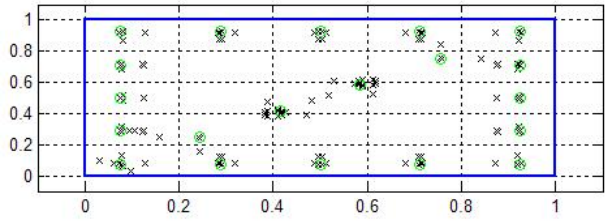


Figura 24: Localizzazione generata dal file *Run.m*.

12.2 Localizzazione tramite Minimi Quadrati Pesati: *Localizzazione.m*

Visti gli scarsi risultati ottenuti con il metodo precedente si è deciso di ricorrere ad uno stimatore ai minimi quadrati corretto. Per avere quest'ultima condizione si è dovuto correggere la formula di inversione per determinare la distanza in funzione della potenza come segue:

$$d_C(p) = e^{-\frac{\gamma}{2}} 10^{\frac{A-p}{10n_p}} \; dove \; \gamma = \left(\frac{\sigma ln10}{10n_p}\right)^2 \qquad (102)$$

in questo modo si ha che :

$$E_d\left[d_C(p)\right] = d \qquad \forall p \in R \qquad (103)$$

con varianza:

$$var_{d_C} = E_d\left[(d_C - E_d\left[d_C\right])^2\right] = d^2(e^\gamma - 1). \qquad (104)$$

Indicando con T_{x_i} e T_{y_i} le coordinate dell'*i*-esimo nodo ancora, con x_j e y_j la *j*-esima posizione del nodo mobile ed infine con r_j la stima della distanza tra il nodo fisso *i* e il nodo mobile al passo *j*, il problema della localizzazione del nodo *i*-esimo si può scrivere come:

$$\begin{cases} (x_1 - T_{x_i})^2 + (y_1 - T_{y_i})^2 = r_1^2 \\ (x_2 - T_{x_i})^2 + (y_2 - T_{y_i})^2 = r_2^2 \\ \vdots \\ (x_n - T_{x_i})^2 + (y_n - T_{y_i})^2 = r_n^2 \end{cases} \qquad (105)$$

che può essere risolto scrivendolo nella forma:

$$AT = b \qquad (106)$$

dove:

$$A = \begin{bmatrix} x_1 - x_2 & y_1 - y_2 \\ x_1 - x_3 & y_1 - y_3 \\ \vdots & \\ x_1 - x_n & y_1 - y_n \end{bmatrix} \qquad (107)$$

$$T = \begin{bmatrix} T_{x_i} \\ T_{y_i} \end{bmatrix} \qquad (108)$$

$$b = 0.5 \begin{bmatrix} r_2^2 - r_1^2 - x_2^2 + x_1^2 - y_2^2 + y_1^2 \\ r_3^2 - r_1^2 - x_3^2 + x_1^2 - y_3^2 + y_1^2 \\ \vdots \\ r_n^2 - r_1^2 - x_n^2 + x_1^2 - y_n^2 + y_1^2 \end{bmatrix}. \qquad (109)$$

Tale sistema si risolve come:

$$T = argmin_T \left\| AT - b \right\|. \qquad (110)$$

In questa risoluzione non si è tenuto conto che le distanze maggiori hanno un incertezza maggiore, si è pensato allora di introdurre una matrice di peso ottima per la stima di T. Per individuare questa matrice si deve ripensare il problema come uno di stima parametrica, attraverso un vettore aleatorio *w*, con varianza R. Seguendo quanto riportato nella tesi proposta da Luca Parolini [4] si è scelto w come:

$$w = 0.5 \begin{bmatrix} d_{d_2}^{C\ 2} - d_{d_1}^{C\ 2} - (d_2^2 - d_1^2) \\ d_{d_3}^{C\ 2} - d_{d_1}^{C\ 2} - (d_3^2 - d_1^2) \\ \cdot \\ \cdot \\ \cdot \\ d_{d_n}^{C\ 2} - d_{d_1}^{C\ 2} - (d_n^2 - d_1^2) \end{bmatrix} \quad (111)$$

a cui corrisponde una matrice di varianza R i cui elementi sono:

$$r_{i-1,j-1} = \begin{cases} d_1^4 e^{2\gamma}(e^{4\gamma} - 1) & i \neq j \\ d_1^4 e^{2\gamma}(e^{4\gamma} - 1) + d_i^4 e^{2\gamma}(e^{4\gamma} - 1) & i = j \end{cases} \quad (112)$$

Si può quindi ottenere la stima di T come:

$$T = (A^T R^{-1} A)^{-1} A^T R^{-1} b. \quad (113)$$

Si è osservato inoltre che non è necessario tener conto delle misure inferiori ad un determinato valore, poichè il loro contributo è nullo, inoltre tener conto di tutte le misure rende l'algoritmo computazionalmente inefficiente. Si può migliorare questa fatto introducendo l'algoritmo iteraticvo che però è meno efficiente.
Questi calcoli vengono eseguiti dalla funzione Matlab Localizzazione.m che richiede in ingresso le seguenti variabili:

- *lim*: valore minimo per cui le misure sono ritenute valide ai fini del calcolo;
- *n*: numero di nodi ancora;
- *z*: può assumere valore '*d*', che indica una disposizione dei nodi ancora sulla diagonale del piano di lavoro e sul perimetro di un rettangolo incluso nel piano di lavoro; può anche assumere valore '*c*', che indica una disposizione randomizzata dei nodi ancora;
- *nodilato*: numero di nodi che può stare sul lato del rettangolo interno;
- *Ts*: tempo di campionamento.

In figura 25 vengono presentati alcuni risultati ottenuti per mezzo dei seguenti valori:

- *lim* = 15;
- *n* = 10;
- *z* = 'd';
- *Ts* = 0.1;
- *nodilato* = 2.

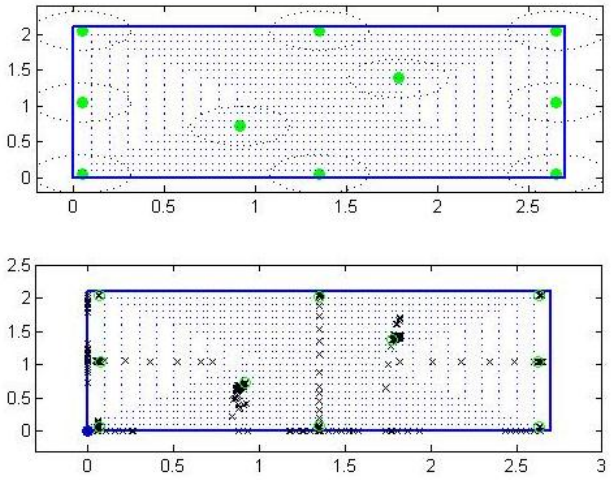


Figura 25: Localizzazione generata dal file *Localizzazione.m*.

Nel prima parte viene presentato il comportamento "vero" del sistema, mentre nella seconda parte è presentata l'evoluzione del sistema stimato, le croci nere rappresentano la stima del nodo, mentre i cerchietti verdi rappresentano le stime all'istante finale.
Nelle figure 26 e 27 è illustrato l'andamento della stima di due nodi, in particolare il numero 7 e il numero 10:

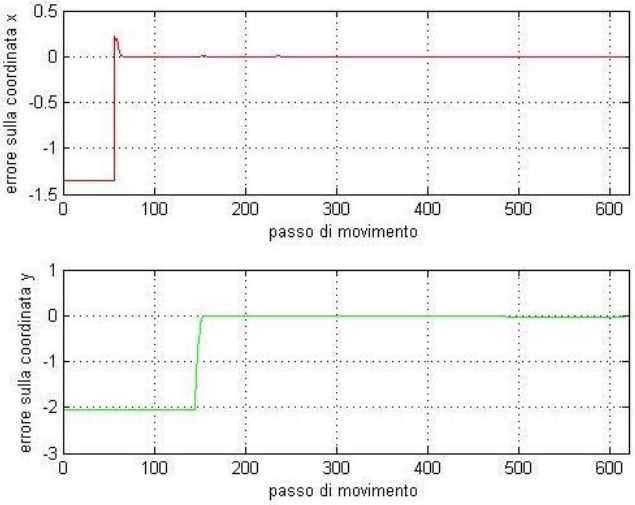


Figura 26: Errore del nodo 7.

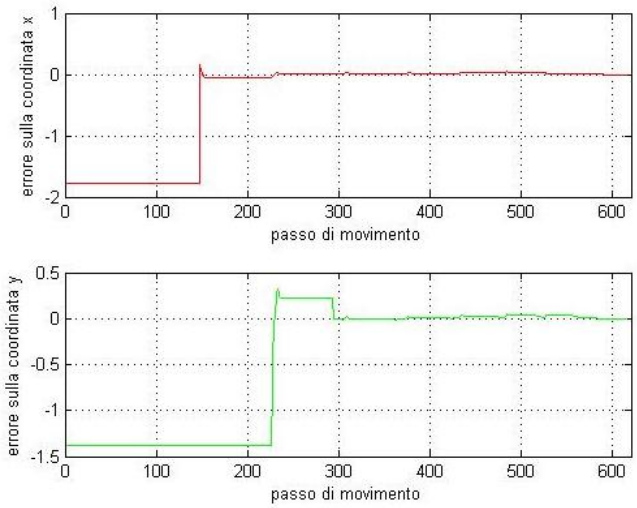


Figura 27: errore nodo 10.

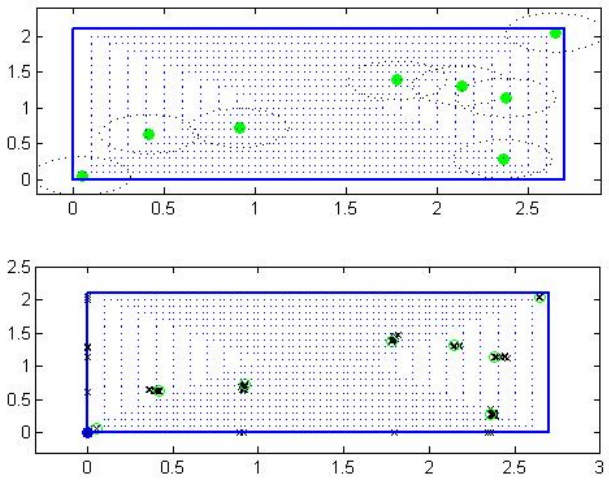


Figura 28: Localizzazione generata dal file *Localizzazione.m*.

Come si vede la precisione è notevole, l'errore è dell'ordine dei millimetri. Inoltre anche la varianza è molto piccola, la traccia della matrice della varianza raggiunge inoltre a stento lo 0.01.
Anche cambiando i parametri i risultati restano più che soddisfacenti; in figura 28 e 29 si riportano i risultati ottenuti con:

- *lim* = 5;
- *n* = 4;
- *z* = 'c';
- *Ts* = 0.1;
- *nodilato* = 2.

Usando però le misure reali si ottengono risultati molto meno efficaci, come si vede dalla figura 30, tenendo lim=0:

A prima vista la stima è molto peggiore, a conferma di ciò si presenta l'andamento degli errori dei nodi in coordinate x e y, rispettivamente figura 31 e 32.
Si ritiene che tale differenza di prestazioni sia dovuta alla non perfetta scelta dei parametri del canale e/o delle caratteristiche del rumore utilizzato per la costruzione della matrice peso. A seguito però di alcuni tentavi, si osserva che con ogni probabilità soprattutto le non perfette caratteristiche del rumore inficiano le prestazioni

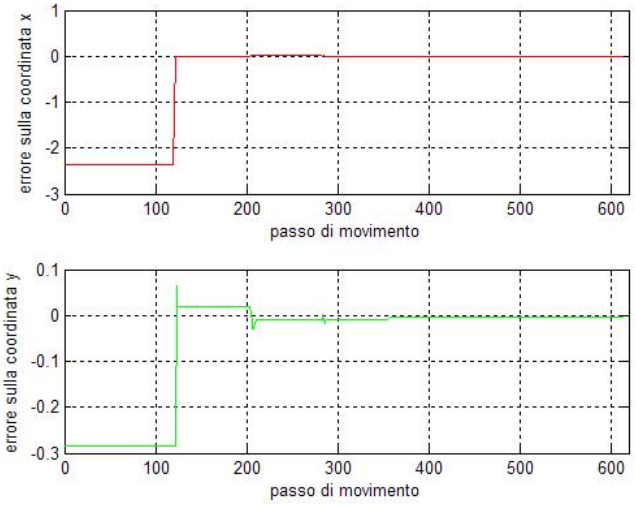


Figura 29: errore nodo 2.

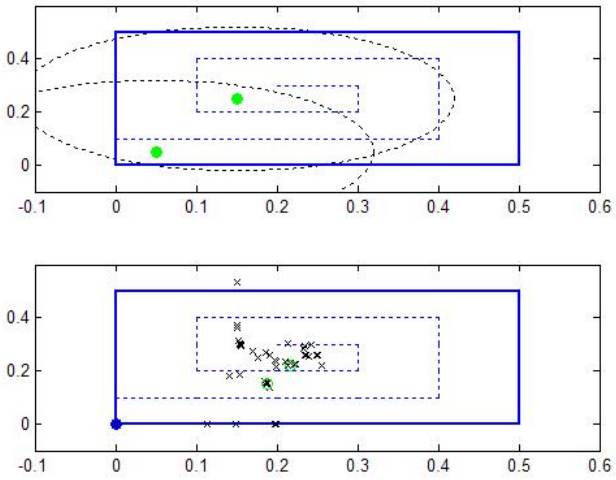


Figura 30: Localizzazione generata dal file *Localizzazione.m* con le misure reali.

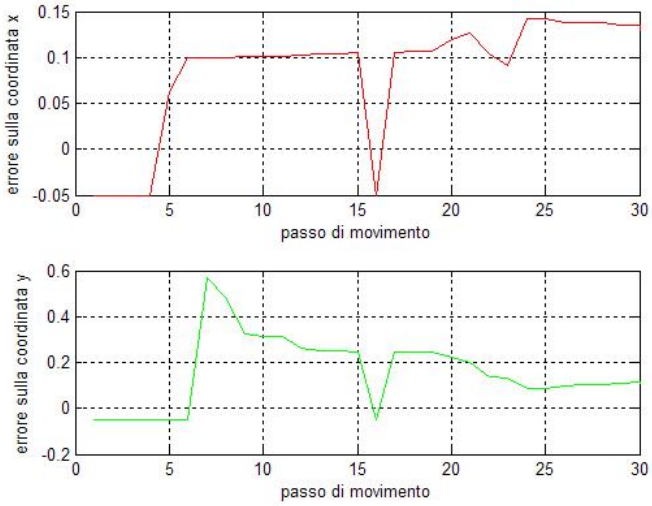


Figura 31: Errore nodo 1 in base alle misure vere.

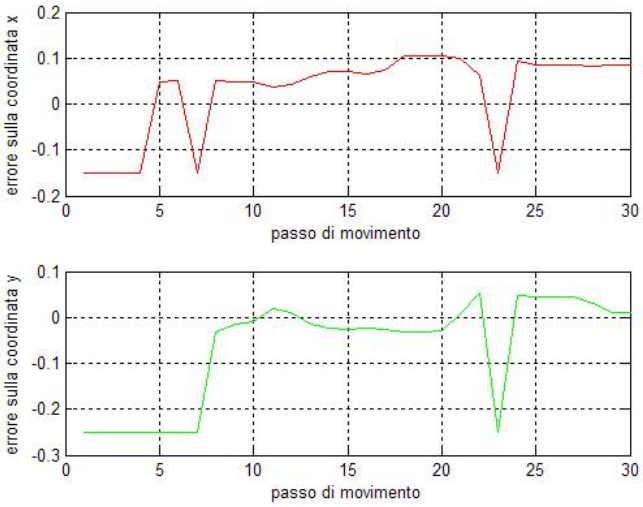


Figura 32: Errore nodo 2 in base alle misure vere.

12.3 SLAM con Kalman: *SLAM.m*

Come già precedentemente introdotto, nel problema dello Slam si vuole contemporaneamente individuare la posizione dei nodi ancora e del nodo mobile. Ovviamente questo complica notevolmente il problema rispetto a quello di localizzazione.
In una prima fase, visto anche l'approcio presente nella documentazione reperita si è pensato di usare unicamente il filtro di Kalman esteso introducendo però degli accorgimenti che tenessero conto della bonta delle misure.
Si è introdotto quindi un primo parametro γ che varia tra 0 e 0.5 che tiene conto di quanto è forte il segnale RSSI e quindi della bontà della misura. Esso agisce solo sul primo range di misure, quelle invece che superano un determinato valore hanno automaticamente γ pari a 0.5 Oltre a questo primo parametro si è introdotto un secondo parametro γ_1 che tiene conto, oltre al caso in cui ci sono valori dell'RSSI sufficientemente accurati, anche di situazioni in cui non ci sono misure utili, seppure vi siano informazioni nel calcolo di $C\hat{x}(t|t)$. Anch'esso può variare fra 0 e 0.5. Entrambi i parametri si è visto che danno buoni risultati usando una legge cubica dei valori fra 0 e 0.5 al variare dell'Rssi all'interno di un range prefissato. Questi parametri moltiplicano i fattori correttivi del Filtro di Kalman, in particolare entrano in gioco nell'aggiornamento.

$$\hat{x}(t|t) = \hat{x}(t|t-1) + (\gamma + \gamma_1)K(Rssi - Rssi_{ric}). \quad (114)$$

Inoltre si è osservato un problema nel caso che la posizione dei nodi sia totalmente incognita, in particolare non avendo un valore iniziale di partenza in prossimità del punto in cui si avevano misure il nodo fisso non veniva individuato, quindi si è dovuto ricorrere ad una modifica che prevede che alla prima misura del nodo la sua posizione iniziale viene modificata e portata nei pressi del nodo mobile a una distanza pari all'inverso del segnale RSSI e su un punto qualsiasi della circonferenza corrispondente. In realtà si può migliorare tale riposizionamento sfruttando la tipologia di percorso preassegnata.
Purtroppo a causa della divergenza del filtro di Kalman esteso questo approccio risulta inefficace poichè le correzioni rispetto alle condizioni iniziali sono insufficienti e trascurabili.
Il tutto viene eseguito dalla funzione Matlab *SLAM.m* che richiede in ingresso le seguenti variabili:

- *AA*: valore nominale di potenza dei nodi;
- *Ts*: tempo di campionamento;
- *n*: numero di nodi ancora;
- *nodilato*: numero di nodi che può stare sul lato del rettangolo interno;
- *z*: può assumere valore '*d*', che indica una disposizione dei nodi ancora sulla diagonale del piano di lavoro e sul perimetro di un rettangolo incluso nel piano di lavoro; può anche assumere valore '*c*', che indica una disposizione randomizzata dei nodi ancora;
- *eps*: valore minimo per cui le misure sono ritenute valide ai fini del calcolo;
- *eps1*: intervallo entro il quale determinare il gamma, se una misura supera *eps*+*eps1* automaticamente essa avrà gamma 0.5;
- *metodo_stima*: può assumere valore '*n*', che indica una conoscenza a priori nulla, tutte le componenti dello stato vengono messe a zero con una determinata varianza; può anche assumere valore '*a*', che indica una conoscenza a priori migliore, i nodi vengono inizializzati all'interno di una circonferenza di raggio pari a 10cm rispetto alla posizione vera con una variuanza inferiore, pari allo 0.1 della varianza precedente;
- *p*: valore della varianza dei nodi.

Inoltre la funzione richiama iterativamente il file Stima.m in cui è presente il filtro di Kalman vero e proprio, che richiede i seguenti parametri:

- *A*: la matrice del modello discreto *F*;
- *B*: la matrice del modello discreto *G*u(t)*;
- *x*: lo stato presente al tempo t;
- *P*: la matrice di varianza presente al tempo t;
- *z*: il vettore delle misure;
- *gamma*: è il valore γ determinato;
- *n*: è il numero di nodi presenti;
- *eps*: stesso paramentro del file SLAM.m;
- *Start_stima*: è il vettore delle condizioni iniziali
- *eps1*: stesso paramentro del file SLAM.m;
- *Metodo_stima*: stesso paramentro del file SLAM.m.

Per maggiori dettagli si rimanda al codice Matlab.

In figura 33 viene riportato l'esito di alcune sperimentazioni, in particolare si sono usati come parametri:

- *eps* = -10;
- *eps1* = 20;
- *n* = 10;
- *z* = 'd';
- *Ts* = 0.1;
- *nodilato* = 2.
- *metodo_stima* = 'n';
- *p* = 10;

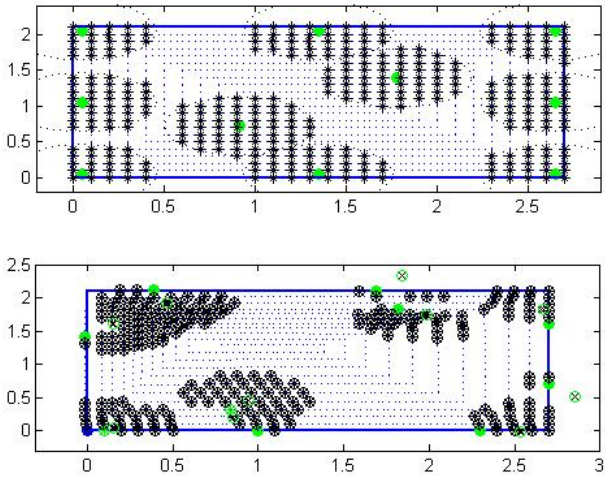


Figura 33: SLAM con Kalman.

Si nota (figure 34, 35, 36) che molto del risultato dipende da dove il sistema sposta il nodo al momento della prima misura. Inoltre si osserva che una volta effettuata la correzione iniziali, le modifiche introdotte dal filtro di Kalman sono irrilevanti e quindi è evidente che il sistema non funziona in modo efficiente. Inoltre anche la varianza cresce a dismisura ottenendo valori privi di significato. Si vede inoltre che non vi è convergenza del sistema.
Anche cambiando i parametri la situazione non migliora (figure 37, 38,39). Viene riportato l'esito di un esperimento in cui si sono usati come parametri:

- *eps* = -10;
- *eps1* = 25;
- *n* = 5;
- *z* = 'c';

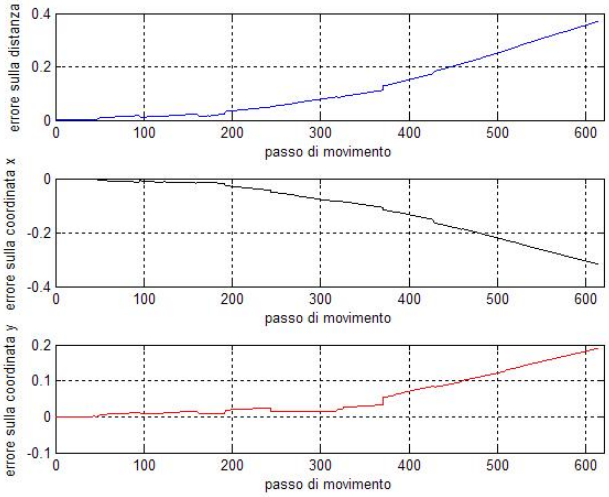


Figura 34: SLAM con Kalman:errore sul veicolo.

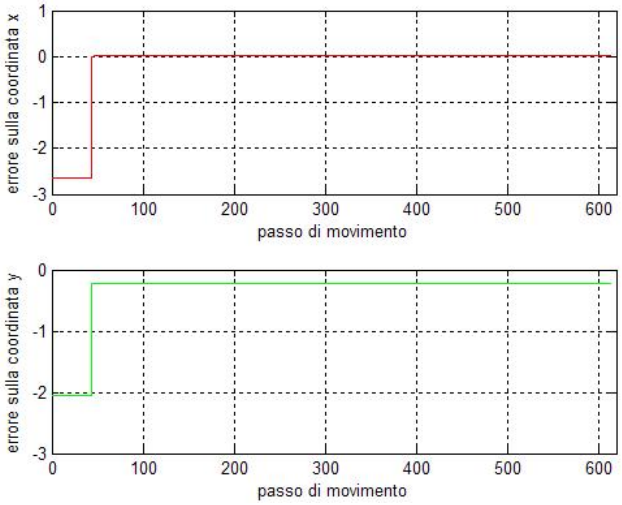


Figura 35: SLAM con Kalman:errore sul nodo fisso n°6.

- *Ts* = 0.1;
- *nodilato* = 2.
- *metodo_stima* = 'a';
- *p* = 15;

Come si vede i risultati sono insoddisfacenti e quindi anche questo algoritmo non viene testato con i valori sperimentali.

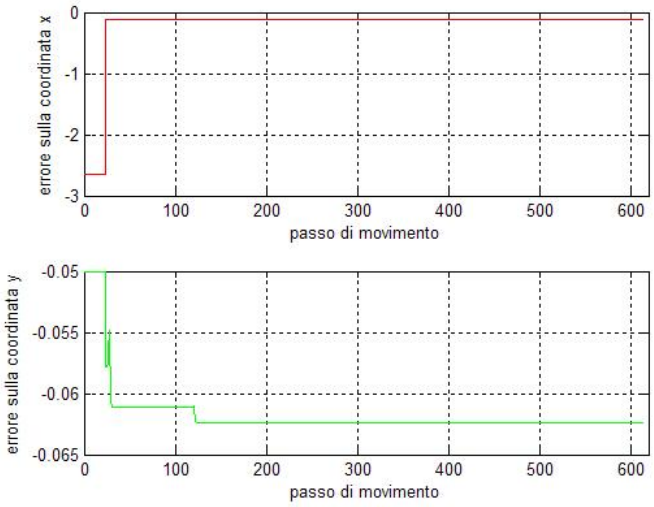


Figura 36: SLAM con Kalman:errore sul nodo fisso n°2.

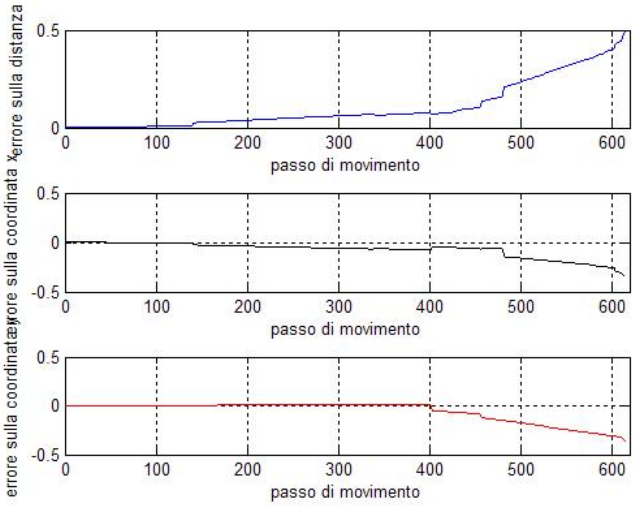


Figura 38: SLAM con Kalman:errore sul veicolo.

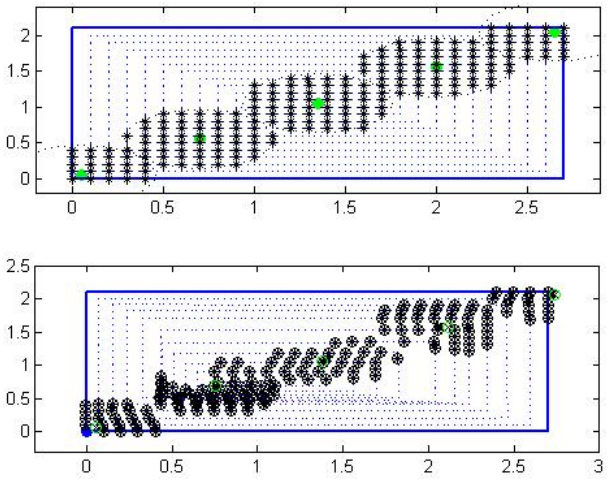


Figura 37: SLAM con Kalman.

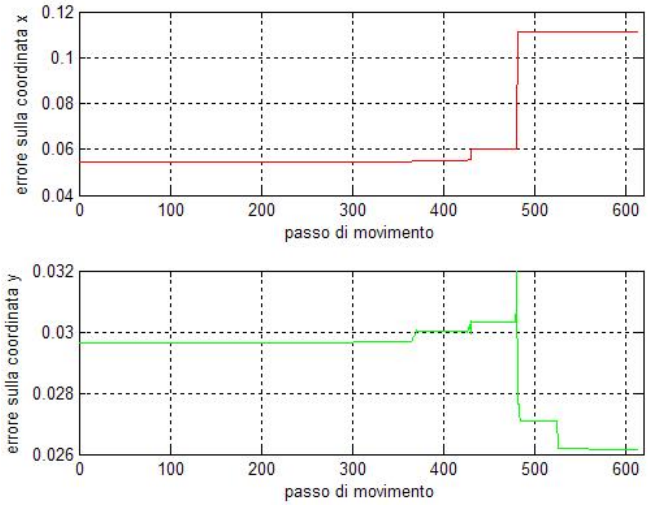


Figura 39: SLAM con Kalman:errore su un nodo fisso.

12.4 SLAM con Kalman e Least Squares : *SLAM.m*

Per migliorare alcune delle lacune presenti nel file per lo Slam precedente si è pensato di introdurre il metodo least square per la stima della posizione dei nodi fissi senza però tener conto della bontà della stima della posizione corrente del nodo mobile. È evidente che questo non è l'approccio ottimo, tuttavia dà già buoni risultati, anche in considerazione del fatto che comunque le condizioni di inizializzazione del nodo mobile non sono totalmente incognite.
Tuttavia si sono presentate alcune problematiche, in particolare nei primi 5 passi, quando cioé non é ancora disponibile una stima accurata dei nodi: in questo caso si utilizza ancora il filtro di Kalman esteso (nella sua versione integrale e semplificata) che riesce a corregere in parte la posizione iniziale del nodo.
In ogni caso anche in questo uso del filtro di Kalman si è introdotto un fattore correttivo, che agisce, similmente a quanto già introdotto nell'aggiornamento, in modo che correzioni evidentemente fuorvianti, dovute a misure inaffidabili, non si presentino.
Ad ogni modo se la condizione iniziale è evidentemente errata non si può correggere in modo utile la traiettoria. In secondo luogo ci si è resi conto che solo quando la stima ai minimi quadrati è sufficientemente accurrata si ottengono dei buoni risultati altrimenti è meglio lasciar procedere l'algoritmo in catena aperta.
In questo caso il file *Slam.m* richiede i seguenti parametri in ingresso:

- *Ts*: tempo di campionamento;
- *n*: numero di nodi ancora;
- *nodilato*: numero di nodi che può stare sul lato del rettangolo interno;
- *z*: può assumere valore '*d*', che indica una disposizione dei nodi ancora sulla diagonale del piano di lavoro e sul perimetro di un rettangolo incluso nel piano di lavoro; può anche assumere valore '*c*', che indica una disposizione randomizzata dei nodi ancora;
- *p*: valore della varianza iniziale dei nodi.
- *metodo_stima*: può assumere valore '*n*', che indica una conoscenza a priori nulla, tutte le componenti dello stato vengono messe a zero con una determinata varianza; può anche assumere valore '*a*', che indica una conoscenza a priori migliore, i nodi vengono inizializzati all'interno di una circonferenza di raggio pari a 10cm rispetto alla posizione vera con una variuanza inferiore, pari allo 0.1 della varianza precedente;
- *lim*: valore minimo per cui le misure sono ritenute valide ai fini del calcolo;
- *q*: valore della varianza del rumore di processo dei nodi fissi;
- *q1*:valore della varianza del rumore di processo del nodo mobile.

Di seguti vengono proposti i risultati ottenuti attraverso simulazione e attraverso le prove in laboratorio.
In particolare le prime figure (n. 40, 41, 42) rappresentano i valori ottenuti immettendo come parametri i seguenti valori:

- *lim* = 15;
- *q* = 1;
- *q1* = 10;
- *n* = 5;
- *z* = 'd';
- *Ts* = 0.1;
- *nodilato* = 2.
- *metodo_stima* = 'a';
- *p* = 10;

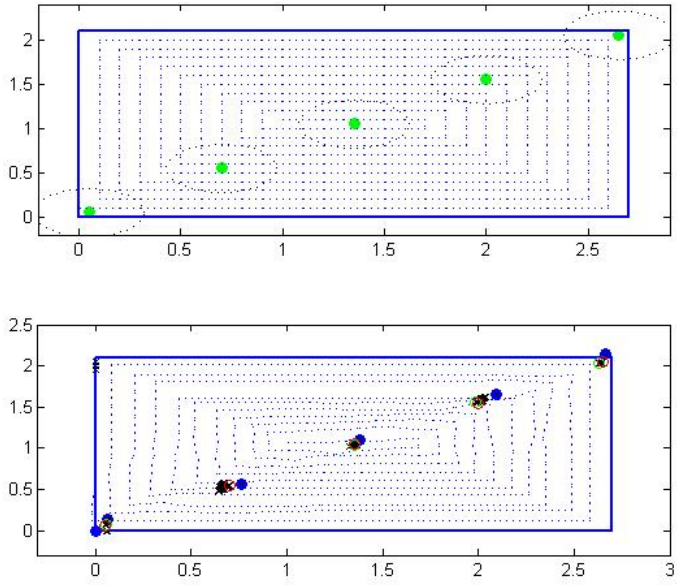


Figura 40: SLAM con Kalman e Least Squares.

Anche cambiando i parametri le prestazioni restano soddisfacenti. Ad esempio nelle figure 43, 44, 45 è riportato quanto si ottiene ponendo:

- *lim* = 15;

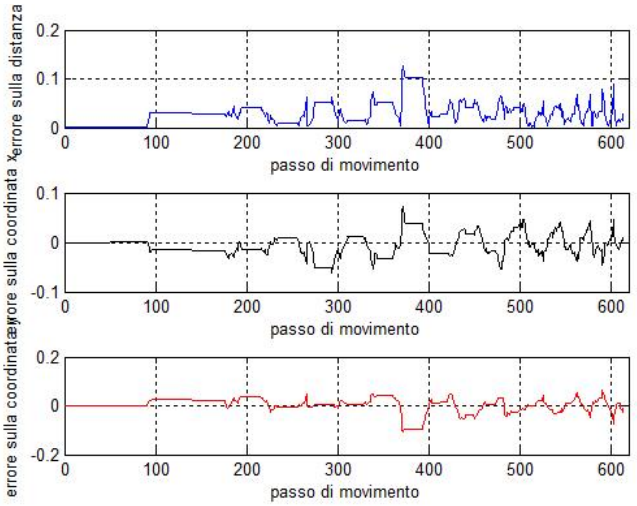


Figura 41: SLAM con Kalman e Least Squares:errore sul veicolo.

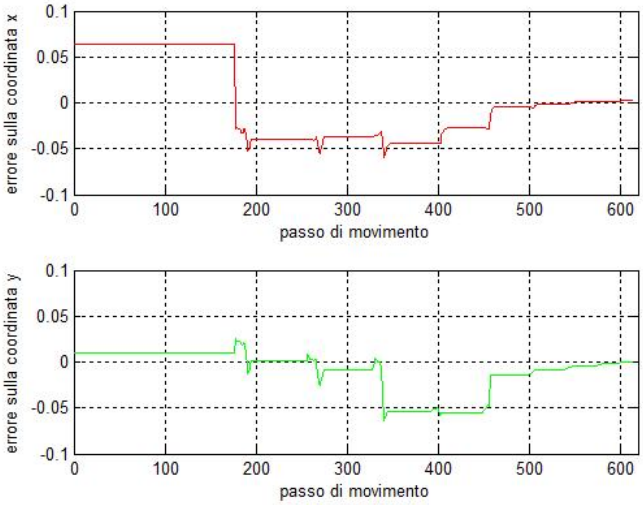


Figura 42: SLAM con Kalman e Least Squares:errore sul nodo fisso n 2.

- q = 10;
- *q1* = 10;
- n = 5;
- z = ’d’;
- *Ts* = 0.1;
- *nodilato* = 2.
- *metodo_stima* = ’n’;
- p = 100;

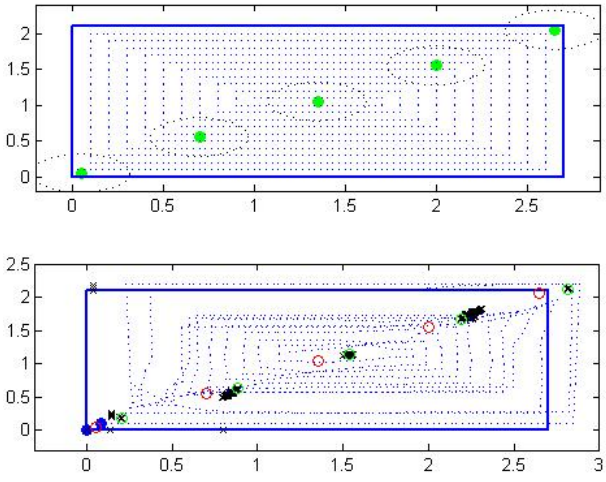


Figura 43: SLAM con Kalman e Least Squares.

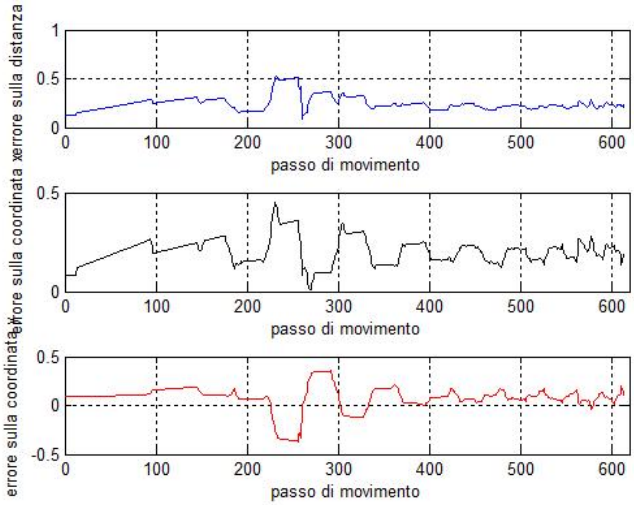


Figura 44: SLAM con Kalman e Least Squares: errore sul veicolo.

Ora si vuole provare questo algoritmo con i dati rilevati in laboratorio. Si osserva che le prestazioni peggiorano in modo rilevante, tuttavia ponendo i seguenti parametri:

- *lim* = 3;
- q = 0.6;
- *q1* = 0.5;
- n = 2;
- z = ’d’;
- *Ts* = 1;
- *nodilato* = 2.

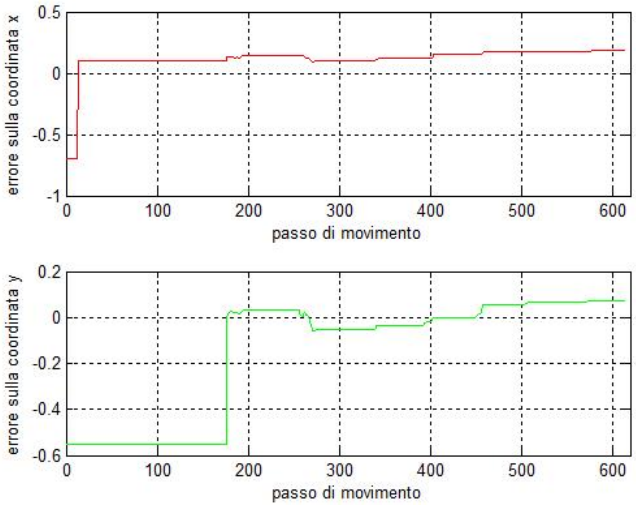


Figura 45: SLAM con Kalman e Least Squares: errore sul nodo fisso n 2.

- *metodo_stima* = 'n';
- p = 10;

si ottengono i risultati presentati nelle figure che vanno dal n. 46 al 49:

Come si vede i risultati non sono ottimali, questo è dovu-

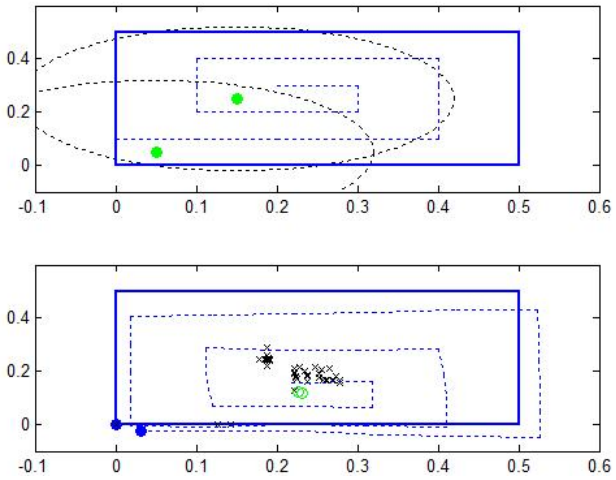


Figura 46: SLAM con Kalman e Least Squares.

to al fatto che nel calcolo della posizione dei nodi fissi non si tiene conto dell'incertezza sulla posizione del nodo mobile. Inoltre osservando i valori misurati dall'Rssi ci si accorge che il comportamento dell'Rssi è differente da quello aspettato.

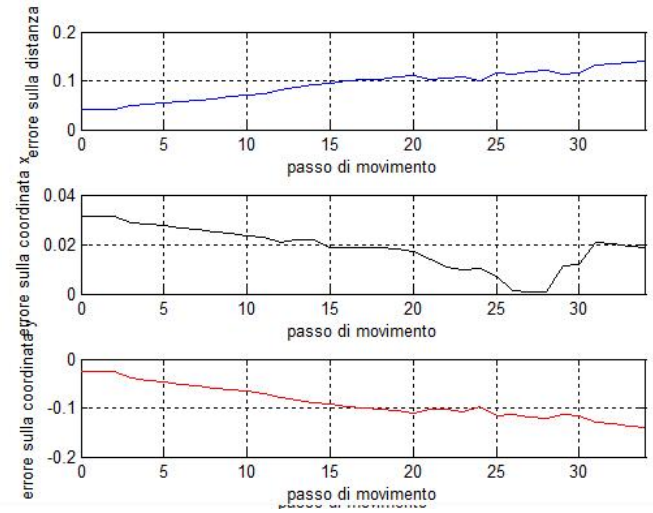


Figura 47: SLAM con Kalman e Least Squares: errore sul veicolo.

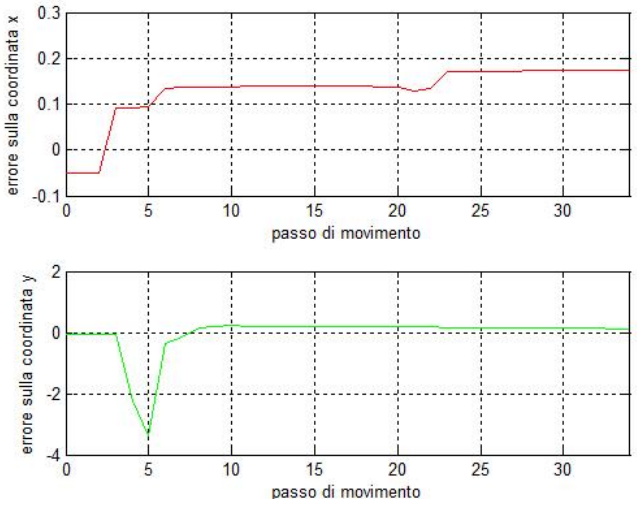


Figura 48: SLAM con Kalman e Least Squares: errore sul nodo fisso n 1.

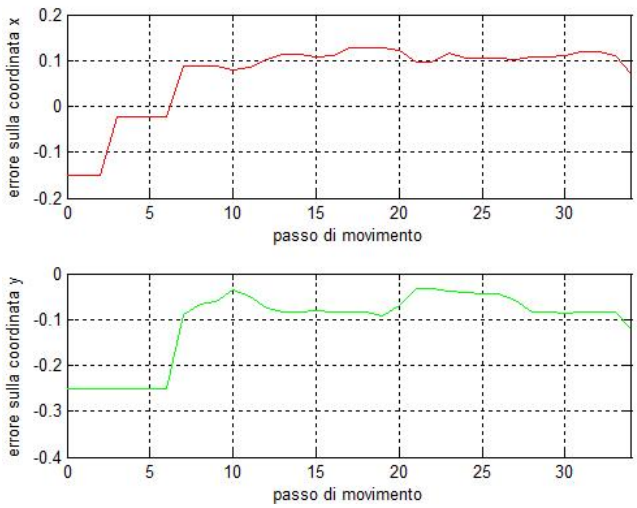


Figura 49: SLAM con Kalman e Least Squares: errore sul nodo fisso n 2.

13 Conclusioni e Sviluppi Futuri

In questo relazione sono stati presentati due algoritmi risolutivi per ciascun problema di localizzazione e di Slam. In entrambi i casi il primo algoritmo si è rilevato inefficace e in particolar modo nella localizzazione anche fuorviante.
Per quanto concerne la localizzazione l'algoritmo che utilizza i *Weighted Least Squares* è risultato molto efficiente e preciso, per lo meno nella simulazione. Si è riscontrato invece, al momento della prova sperimentale, che le prestazioni sono calate. Questo è da imputarsi al fatto che per funzionare in modo corretto esso richiede una notevole precisione nella modellizzazione del canale e del rumore *w* utilizzato per determinare i pesi della matrice *R*.

Questi buoni risultati non si sono ripresentati nello Slam, in parte a causa della approssimazione fatta nella ricostruzione della posizione dei nodi fissi, in parte a causa della maggiore difficoltà del problema. Si è però riusciti a limitare tali errori correggendo in modo opportuno i vari parametri, in particolare le soglie sui valori dell'Rssi.
Durante le simulazioni ci si è resi conto che pur non ottenendo buoni risultati in assoluto, si è sempre verificata la struttura geometrica dei nodi fissi. Questo era prevedibile in base a quanto reperito in letteratura.
Si è inoltre osservato che il filtro di Kalman esteso privo di ausili non offre buone prestazioni. Anche questa caratteristica é ampiamente nota in letteratura. Si è in parte risolto tale problema introducendo i minimi quadrati che hanno dato maggiore affidabilità.

L'algoritmo dello *Slam* con i minimi quadrati è certamente migliorabile introducendo modifiche al metodo di calcolo della posizione dei nodi fissi, tenendo conto della varianza del nodo mobile. Inoltre si potrebbe aggiungere un algoritmo più accurato che tiene conto della bontà delle misure Rssi come proposto anche in [1].

Riferimenti bibliografici

[1] Pubudu N. Pathirana, Nirupama Bulusu, Andrey V. Savkin, Sanjay Jha, "Node Localization Using Mobile Robots in Delay-Tolerant Sensor Networks," *IEEE TRANSACTIONS ON MOBILE COMPUTING*, vol. 4, no. 3, may/june 2005 pp. 285-296

[2] R. Frezza, S. Dazzo, L. Parolini, R. Sala, "Il problema della Localizzazione e dell'Autolocalizzazione in una Rete di Sensori Wireless," *UNIVERITA' DEGLI STUDI DI PADOVA - DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE*, 2006

[3] S. Zilli, "Range-only Slam in ambienteindoor per unità robotica mobile e rete di sensori wireless," *UNIVERITA' DEGLI STUDI DI PADOVA - DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE*, 2006-2007

[4] L. Parolini, "Metodi di localizzazione per reti di sensori wireless," *UNIVERITA' DEGLI STUDI DI PADOVA - DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE*, 2005-2006

[5] H. Durrant-Whyte, T. Bailey, "Slam tutorial: part 1," *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, june 2006

[6] G. Zunino, H. I Christensen, "Navigation in realistic environment," 2006

[7] P. Robertson, "Introduction to Slam," 2005

[8] G. Picci, "Filtraggio statistico (Wiener, Levinson, Kalman) e applicazioni, terza edizione" *Libreria Progetto*, 2007.

[9] G. Cariolaro, G. Pierobon, G. Calvagno "Segnali e sistemi," *McGraw-Hill*, 2003.