Localizzazione e tracking a tempo minimo mediante rete di sensori wireless WSN

Sartore Filippo (566717-IAM) Sassaro Alex (567095-IAM) Vettori Davide (566721-IAM)

Abstract— **Il progetto si pone come collegamento tra due progetti precedentemente sviluppati riguardanti la localizzazione attraverso rete di sensori wireless (WSN) e il tracking per un veicolo (monociclo) mobile su gomma. La prima parte dell'articolo sarà dedicata alla localizzazione tramite WSN e ai metodi di taratura di una rete di sensori riportando, attraverso alcune sperimentazioni, i risultati ottenuti; in particolar modo si farà riferimento alla piattaforma utilizzata nel laboratorio NAVLAB presso il Dipartimento di Ingegneria dell'Informazione dell' Università di Padova. Nella seconda parte, saranno illustrati i risultati simulativi ottenuti combinando il software progettato per la rilevazione di posizione attraverso WSN con il software per il tracking a tempo minimo con discretizzazione della traiettoria, interponendo tra i due l'implementazione di un filtro di Kalman esteso (EKF) con la finalità di ridurre il grado di incertezza che si riscontra nella localizzazione della posizione sull'ambiente di azione dell'e-puck.**

I. Introduzione

Lo sviluppo del progetto è partito dall'analisi di due software di localizzazione presentati nell'anno accademico 2006-2007 per il corso di Progettazione di sistemi di controllo [7], [8], e da una tesi di laurea [6] riguardante il tracking per veicoli mobili anolonomi.
Per quanto riguarda la localizzazione, il primo passo è stato l'analisi approfondita dei progetti sopra citati, nei quali si affrontava il problema della realizzazione di un 'tesbed' per lo sviluppo di una rete wireless all'interno dei laboratori dell'Istituto Nazionale di Fisica Nucleare (INFN) di Legnaro (PD). In queste relazioni si erano implementati due algoritmi per la localizzazione di un nodo mobile in una rete di nodi fissi in modalità wireless. Il primo algoritmo si basa sull'utilizzo di un filtro particellare, il quale stima la posizione mediante una mappa di potenza del segnale nell'ambiente, tale mappa viene ricavata a priori tramite un elevato numero di misure rilevate e memorizzate in un computer centrale. Il secondo progetto, al contrario, si basa sulla misura diretta del segnale di una cella, cioè un numero limitato di nodi fissi dopo aver utilizzato metodi di identificazione per determinare il valore dei parametri del canale di comunicazione. Da questa informazione e dal fatto che i nodi àncora hanno posizione nota, si risale alla distanza del nodo mobile dai nodi àncora e alla sua localizzazione.
Nel progetto di tesi [6] si analizza la possibilità di pianificare e controllare il movimento di veicoli anolonomi tramite un controllore ibrido che utilizza ad anello chiuso l'informazione proveniente da sensori di bordo e riceve la comunicazione della posizione del veicolo da un sistema di localizzazione esterno (nel nostro caso una rete WSN) per correggerne la triettoria.
Saranno presentati nelle specifiche sezioni maggiori dettagli riguardanti lo stato dell'arte dei sistemi.

A. Obiettivi

La realizzazione della navigazione di un robot mobile attraverso la rilevazione della posizione tramite rete wireless risulta la composizione di più problematiche che possono essere studiate separatamente. Per questo il problema verrà scomposto in due sotto problemi ben distinti:

- identificazione della posizione del veicolo sul piano
- filtraggio della posizione acquisita tramite rete wireless e tracking a tempo minimo di traiettorie punto punto.

Come detto nello stato dell'arte, per l'identificazione della posizione del veicolo sul suo piano di movimento si è partiti dai progetti [7], [8], realizzati con la finalità di localizzare un sensore mobile, nel caso particolare posto sopra l'uniciclo, in un ambiente conosciuto, attraverso una rete WSN. Dopo uno studio approfondito, si è scelto di utilizzare il software [8], in quanto più consono all'attività che si dovrà svolgere. La prima parte dell'articolo tratterà lo studio e le problematiche affrontate per l'adattamento del software di localizzazione utilizzato dalla rete di sensori wireless presente sotto la piattaforma di movimento. Sarà analizzata, attraverso diverse sperimentazioni, la migliore configurazione delle rete. In aggiunta, verrà presentato il software implementato per il controllo dei veicoli mobili (e-puck) disponibili presso il laboratorio di ricerca Navlab.
La seconda parte dell'articolo tratterà invece l'altro problema menzionato. Vista l'elevata varianza d'errore

di stima della posizione, ottenuta dalla rete WSN, si è implementato un filtro di Kalman esteso (EKF), adatto ad essere inserito in sistemi fortemente non lineari quali il nostro. Il filtro di Kalman avrà lo scopo di stimare la posizione del veicolo nell'ambiante nel modo più preciso possibile. Si presenteranno i risultati soddisfacenti ottenuti dal filtraggio di misure corrotte da rumore, applicato a diverse tipologie di traiettorie. Seguirà un analisi di alcuni esempi di pianificazione di traiettoria punto punto, con filtraggio delle misure, evidenziando l'inefficacia di tale algoritmo al caso pratico in esame.

B. Risultati

La modifica del software di localizzazione e la taratura della rete wireless hanno portato, come si vedrà nella sezione dedicata, a dei risultati, con particolari ipotesi, soddisfacenti.

La pianificazione di traiettoria con filtraggio non risulta robusta. Si verificano casi in cui l'obiettivo viene raggiunto ma con percorsi sicuramente non ottimi, mentre in altri casi addirittura non è possibile raggiungere il punto stabilito. Dall'analisi dei risultati non è possibile stabilire un trend di prestazione costante per tutti i casi studiati. Comunque sembra che l'interazione tra pianificazione e filtraggio non sia performante.

L'interazione tra la parte relativa alla localizzazione e la parte riguardante il tracking ottimo non è stata possibile a causa di alcuni problemi pratici riguardanti l'impossibilità di fusione dei due software. Infatti, l'algoritmo di pianificazione ottima prevede il controllo dell'uniciclo alla massima velocità non lasciando il tempo opportuno alla rete wireless di effettuare una localizzazione affidabile. Quest'ultimo fatto si ripercuote sulla generazione della traiettoria ottima a tempo minimo.

II. Architettura Del Sistema

La definizione del tipo di architettura risulta fondamentale per lo sviluppo del sistema e per il raggiungimento dell'obiettivo finale. Lo scopo principale del progetto riguarda la localizzazione e il tracking a tempo minimo, tramite rete di sensori wireless, di un oggetto mobile presente in un definito ambiente di lavoro. Si vuole cioè realizzare un sistema, in cui un utente, da una centrale di comando, sia in grado di visualizzare in tempo reale la posizione dell'oggetto nell'ambiente, e, sempre dalla propria postazione, selezionare la posizione finale che esso dovrà raggiungere nel tempo più breve possibile. La soluzione adottata a tale proposito è riportata in *figura 1*.

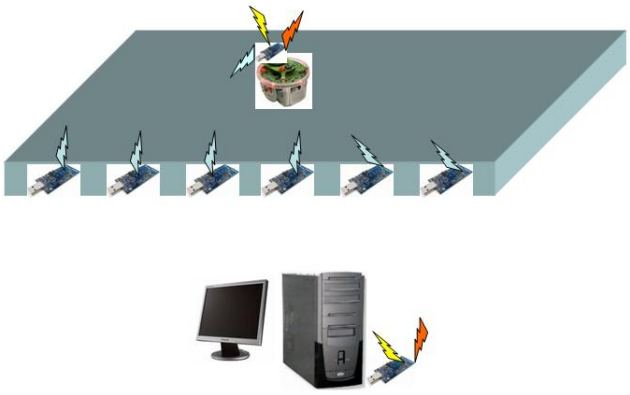


Fig. 1. Schema generale dell'applicazione

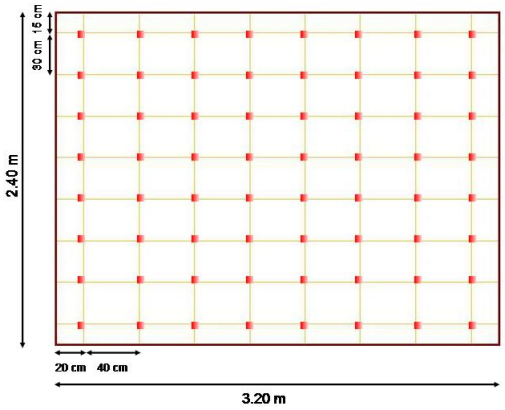


Fig. 2. Dimensioni della piattaforma e posizione dei sensori WSN

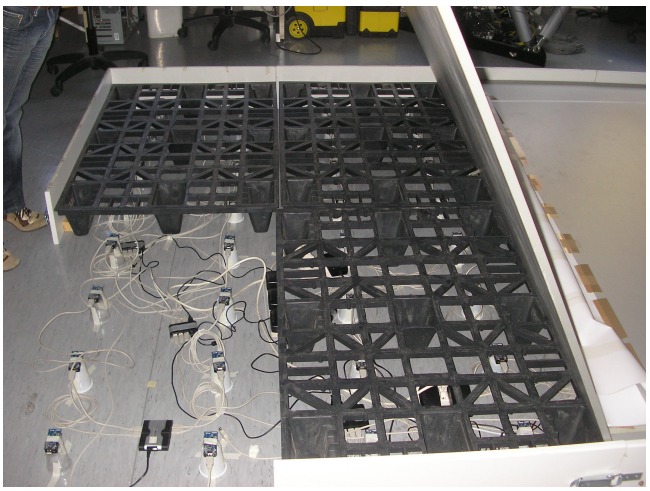


Fig. 3. Particolare della rete di sensori

L'ambiente di lavoro utilizzato per la sperimentazione risulta ridotto ad una piattaforma di dimensioni 3.20 m per 2.40 metri, in cui sono stati disposti, in maniera uniforme, 64 sensori WSN, secondo la griglia riportata in *figura 2*, dove i sensori sono indicati con dei quadratini rossi. In *figura 3* viene riportato un particolare della piattaforma realizzata.

Ai nodi presenti nella rete vanno poi aggiunti due ulteriori sensori: il primo collegato, tramite porta USB, al PC di controllo, il secondo collegato al robot mobile, cioè all'uniciclo *e-puck* utilizzato per la sperimentazione.
Prima di passare alla descrizione del principio di funzionamento del sistema ideato, si riporterà una breve panoramica dei componenti fino ad ora accennati.

A. *Sensori WSN* TmoteSky

I nodi wireless utilizzati nella sperimentazione sono i sensori TmoteSky prodotti dalla *MoteIV*, figura *4*, cui dati tecnici sono riportati in [1]. I nodi sono equipaggiati con un microcontrollore MSP430, [2], prodotto dalla Texas Instruments. Esso ha a disposizione 48 Kbyte di memoria Flash dedicata per il software e 10 Kbyte di SRAM. Ha un oscillatore DCO che opera sopra gli 8 Mhz e un oscillatore al quarzo utile alla calibrazione del primo. *È* dotato di 8 porte ADC interne e altrettante esterne, necessarie per leggere i valori dei sensori o dei livelli di batterie.
Il mote è dotato anche di una flash RAM STM25P80, [4], che opera a 40 MHz e fornisce 1024 Kbyte per memorizzare dati divisi in 16 segmenti di 64 Kbyte l'uno e collegata al microcontrollore tramite il bus SPI.
Il chip radio montato è il CC2420 prodotto dalla Chipcon ed è controllato dall'MSP430 tramite il bus SPI. I dati tecnici del chip sono riportati in [3] assieme allo standard IEEE 802.15.4 descritto in [5].

Attraverso appositi comandi è possibile definire tutti i parametri fondamentali che riguardano la trasmissione radio dei mote, come la potenza, la frequenza e il

Fig. 4. Nodo TmoteSky

gruppo. 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. Per quanto riguarda la potenza in trasmissione, sono disponibili 32 diversi livelli, quantizzati in 8 possibili valori elencati nella *tabella I* dal più elevato, 31, al più debole, 3.

TABLE I
LIVELLI DI POTENZA DI TRASMISSIONE CON RELATIVI CONSUMI

LIVELLO	Pot. di trasmissione [dBm]	Consumo [mA]
31	0	17.4
27	-1	16.5
23	-3	15.2
19	-5	13.9
15	-7	12.5
11	-10	11.2
7	-15	9.9
3	-25	8.5

Attraverso tali sensori è possibile ricavare l'indice di potenza del segnale ricevuto, RSSI, e l'indicatore di qualità della connessione, LQI.
I nodi sono programmati in NESC: un linguaggio di alto livello molto simile al C/C++. Tutti gli algoritmi di gestione delle diverse funzionalità del chip sono gestite attraverso il sistema operativo TINYOS. Maggiori dettagli a proposito saranno forniti nel capitolo riguardante il software utilizzato per la localizzazione. Per un eventuale approfondimento sul linguaggio di programmazione dei sensori si consiglia la lettura di [22] e [23].

B. Uniciclo e-puck

L'uniciclo utilizzato nella sperimentazione del sistema è il robot *e-puck* (*figura 5*), un progetto creato dall'*Ecole Polytechnique Fédérale de Lausanne* attraverso la collaborazione dell'*Autonomous Systems Lab*, la *Swarm-Intelligent Systems Group* e il *Laboratory of Intelligent System*. In questa sezione si forniranno delle informazioni generali circa l'hardware e il software del robot mobile.

Fig. 5. Uniciclo e-puck

Come si nota dalla figura 5, l'e-puck ha una struttura molto semplice e presenta un diametro di 70 mm. *È* alimentato grazie ad una batteria allocata nella parte inferiore della struttura e collegata grazie a due contatti verticali. La batteria può inoltre essere estratta per ricaricarla esternamente. L'uniciclo presenta due piccoli motori passo-passo con riduttori di marcia nelle ruote. In un giro il motore fa 20 passi, mentre le ruote hanno un fattore di riduzione di 50:1; questo significa che la velocità massima dell'e-puck è di 1000 passi al secondo, che in un secondo corrisponde ad un giro completo delle ruote. Le due ruote hanno un diametro di 41 mm e sono disposte ad una distanza di circa 53 mm.
L'elettronica dell'e-puck è stata costruita per poter supportare un elevato numero di usi a livello didattico universitario; si ha infatti a diposizione: un interfaccia bluetooth e RS232 per la comunicazione con un PC, otto sensori di prossimità posti intorno al robot, tre microfoni e un autoparlante per permettere la registrazione e la generazione di suoni, una piccola videocamera con una risoluzione di 640x480 pixel e 8 led rossi posti intorno al robot. Per poter aver una buona potenza computazionale e per usare i compilatori standard (GNU) vi è un microprocessore *dsPIC* della *Microchip*. La comunicazione tra PC e robot è permessa grazie al microntrollore dsPIC che mette a disposizione due UART: la prima collegata al chip bluetooth *LMX9820A* ([24]) e la seconda per la comunicazione seriale. Maggiori informazioni possono essere ricercate in [15], sito ufficiale dell'e-puck.

C. Principio di funzionamento del sistema

Con riferimento alla *figura 1*, si presenterà l'idea da cui si è partiti per lo sviluppo del progetto. L'inizializzazione del sistema avviene grazie allo start dell'utente posto in sala di controllo. Il mote collegato al PC di comando, invierà un segnale di avvio al nodo mobile posto nel robot, il quale, a sua volta, attiverà tutti

i nodi àncora presenti nell'ambiente. Una volta attivati i nodi sarà possibile effettuare una prima localizzazione dell'uniciclo. Il nodo mobile inizierà a comunicare con i nodi fissi inviando una serie di ping necessari ai nodi àncora per stimare la sua posizione. Passato il tempo necessario al calcolo della stima, i nodi àncora invieranno al mote mobile il dato elaborato che a sua volta verrà inviato al nodo collegato all'USB del PC utente. Il software elaborerà il dato ricevuto via seriale e visualizzerà a video, nella mappa dell'ambiente precedentemente disegnata, la posizione iniziale del robot. L'utente potrà ora comandare la posizione finale da far raggiungere all'uniciclo, che dovrà essere raggiunta nel minor tempo possibile. La posizione iniziale trovata dalla localizzazione e il punto finale inserito dall'utente verranno utilizzati come ingressi per il software di tracking ottimo il cui compito sarà di generare delle traiettorie per il comando dell'uniciclo: il robot inizierà dunque a muoversi per raggiungere la meta prefissata. I nodi àncora, con lo stesso procedimento visto per l'inizializzazione del sistema, continueranno a stimare la nuova posizione del mote mobile, che sarà inviata al PC: questa volta il dato ricevuto oltre ad essere utilizzato dal software di localizzazione dovrà essere elaborato dal software di comando del robot. Infatti, le misure ricevute dal campo saranno prima filtrate grazie al filtro di Kalman implementato e poi elaborate per il calcolo della nuova traiettoria ottima e quindi per la generazione dei comandi di movimento dell'e-puck. Durante tutte queste operazioni l'utente sarà in grado di tener sotto controllo la posizione del robot in tempo reale grazie al software di localizzazione.

D. Osservazioni

Quanto spiegato nel paragrafo precedente, in fase di sperimentazione, non è stato del tutto realizzabile per alcuni motivi che si andranno ora a spiegare.
Per prima cosa non è stato possibile realizzare il ponte radio tra PC di controllo e mote posto nell'uniciclo, in quanto il software di localizzazione, fornito da [8], non prevede la possibilità di decentralizzare il comando di avvio del sistema ad un nodo che non fosse il nodo mobile. Dopo svariati tentativi di modifica del codice NESC dei Tmote, ci si è accorti che il problema risiedeva nel codice implementato per l'interfaccia java di localizzazione, che assegnava come nodo mobile il nodo collegato alla porta USB del PC. Un'altra motivazione è data dal fatto che la stima della posizione necessita di uno scambio continuo e in tempo reale delle misure acquisite dai nodi àncora che trasmettono le informazioni in broadcast; questo continuo scambio avrebbe interferito con il ponte radio da implementare. Non è stato quindi possibile modificare tali sorgenti in quanto strutturate in un modo ben preciso e non adattabili per la realizzazione dell'idea iniziale del progetto. Il secondo punto riguarda invece il software utilizzato per il tracking e motion planning del robot mobile. Il codice di partenza è stato fornito da [6], a cui, come verrà spiegato nei capitoli successivi, è stato introdotto il filtro di Kalman e ulteriori modifiche. Il software, testato con esito positivo in fase di simulazione, non è stato ancora provato in fase di sperimentazione. Tale complicazione è dovuta soprattutto a motivi pratici. Il software infatti, dopo le proprie elaborazioni, invia all'uniciclo dei comandi di movimento impostati per il funzionamento alla sua massima velocità creando un serio problema alla rete di sensori che dovrà stimare la sua posizione. Infatti, come si vedrà successivamente nei grafici del capitolo III, le rete invierà pochissime misure al software che a sua volta fornirà dei comandi sbagliati all'e-puck impedendo così il raggiungimento del punto finale. La cosa risulta ancora più complicata se si considera le ridotte dimensioni dell'ambiente di lavoro (figura 2).
I capitoli successivi spiegheranno i maniera dettagliata le routine di funzionamento dei software utilizzati e presenteranno i risultati ottenuti durante le simulazioni.

III. Software Di Localizzazione

Questa sezione è dedicata ai metodi utilizzati per l'identificazione del canale di trasmissione, utile per ottenere una buona taratura della rete wireless di lavoro, dove, attraverso diverse sperimentazioni, saranno riportati i risultati conseguiti. Verrà inoltre illustrato il software di localizzazione utilizzato, con le opportune modifiche, e l'interfaccia Matlab e-puck creata per il controllo dell'uniciclo.

A. Stato dell'arte

Per quanto riguarda la localizzazione all'interno di ambienti chiusi, in letteratura sono presenti svariati studi e progetti affrontati in tutte le più importanti università del mondo. Le prime ricerche, sorvolando quelle per scopo militare, vennero effettuate negli Stati Uniti in cui lavorarono tra le più importanti menti del pianeta poichè per affrontare queste applicazioni sono necessarie conoscenze che spaziano in tutti i settori dell'Ingegneria dell'Informazione. Le tecniche di localizzazione che vennero ideate sono molteplici e si possono suddividere in base alle grandezze fisiche utilizzate per questo scopo: dall'analisi delle onde radio emesse (come in questo progetto), agli impulsi sonori o agli infrarossi. Un ulteriore classificazione si può trovare nel modo in cui vengono analizzate le informazioni: dalla misura dell'angolo di arrivo (*AOA - Angle Of Arrival*), in cui l'informazione della distanza è ottenuta stimando gli angoli relativi del segnale trasmesso tra nodi adiacenti, o dalla misura del tempo di volo della grandezza (*TDOA - Time Difference Of Arrival*), nella quale viene analizzato il tempo di propagazione del segnale dalla sorgente alla destinazione, da cui si basa il *Global Positioning System* o più comunemente conosciuto come *GPS*, fino alla misura della potenza del segnale ricevuto definito come *Receive Signal Strength Indicator* o *RSSI*. Quest'ultima si è rilevata la tecnologia principe per questa applicazione e da qui sono partite svariate linee di ricerca utilizzando questa idea. La problematica principale di questa tecnica è lo studio approfondito del modello di propagazione delle onde elettromagnetiche che lega la potenza del segnale ricevuta alla distanza tra i 2 nodi; essa si differenzia molto dall'architettura utilizzata, dalla tipologia di ambiente e da una grande quantità di fattori difficilmente prevedibili e modellizzabili a priori. Questo verrà affrontato nel successivo capitolo di questo articolo poichè punto fondamentale per la buona riuscita della localizzazione. Per un ottimo articolo che descrive esaurientemente tutte le tecniche finora conosciute per la localizzazione tramite una rete di sensori Wireless che per motivi di chiarezza espositiva è [28]. Un'ulteriore classificazione può riguardare la tipologia dell'algoritmo utilizzato per questo scopo: di tipo *centralizzato*, se è un unico nodo che elabora e gestisce i dati da misurare provenienti dai nodi àncora; l'altro è di tipo *decentralizzato*, se l'elaborazione viene fatta da ciascun nodo della rete.
Tra i principali progetti che si sono presi in esame per entrare nell'ottica di questo lavoro e da cui si sono comprese le difficoltà da affrontare e i limiti delle prestazioni che si sarebbero riscontrati nel corso di questo progetto, si riportano i seguenti:

- *Progetto Cricket*: è un lavoro cominciato nel 2000 dalla Massachusetts Institute of Technology (MIT); lo scopo del progetto era la realizzazione di un sistema di localizzazione di tipo decentralizzato per ambienti interni che mantenesse un basso costo di implementazione. La sua peculiarità era l'utilizzo di due tipologie di segnali:
  1) un segnale radio di tipo elettromagnetico;
  2) un segnale a ultrasuoni.

  Attraverso le differenti velocità di propagazione (la prima pari alla velocità della luce - circa $3 * 10^5$ [km/s] -, la seconda a quella del suono - circa 340 [m/s]), si misura la distanza tra il nodo emettitore e il ricevitore. Sebbene la precisione di questa tecnica sia molto elevata e non ci sia bisogno di uno studio approfondito e particolareggiato per la stima di parametri per il corretto funzionamento, dal 2005 è stata tralasciata poichè i componenti per l'invio e la ricezione dei segnali ultrasonici sono risultati molto sensibili agli agenti esterni (come le vibrazioni meccaniche) e quindi a rotture continue dei nodi rendendo inutilizzabile tale soluzione. Nella figura 5 sono riportati i nodi utilizzati in cui sono facilmente riconoscibili i componenti appena descritti.

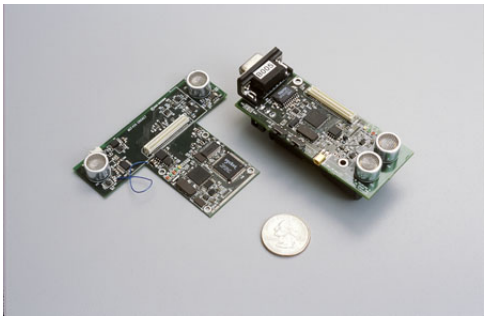


Fig. 6. Fotografia dei sensori utilizzati nel progetto Cricket

- *Progetto Radar* [27]: è un progetto di tesi, sponsorizzato dalla Microsoft, con lo scopo di studio e analisi per la realizzazione di un sistema di localizzazione, per rilevare la posizione di un oggetto all'interno di un edificio, ed il tracking, per guidarlo ad una meta ben precisa. Questo progetto si basa sulle misure della potenza *(RSS - Received Signal Strength)* del segnale emesso in radiofrequenza e sul rapporto tra segnale-rumore ricevuto *(SNR - Signal to Noise Ratio)*. Le operazioni eseguite dall'algoritmo di localizzazione si possono suddividere in 2 fasi: la prima serve per l'inizializzazione della rete di sensori in cui si effettua una raccolta di dati per descrivere com'è distribuita nell'edificio la potenza del segnale emesso dai nodi àncora, attraverso varie misurazioni sul campo a distanze conosciute; la seconda fase in cui vengono effettuate le misure in tempo reale dell'RSS e confrontate con quelle raccolte a priori ed elaborate. Come si può facilmente comprendere, la fase principale e più delicata è la prima, in cui avviene la raccolta dei dati necessari per la localizzazione; per questo, sono stati implementati 2 algoritmi: il primo attraverso le misurazioni costruisce sperimentalmente una mappa di valori che lega il valore misurato di RSS proveniente dai vari nodi àncora ad una particolare posizione all'interno dell'edificio; nel secondo si è utilizzata l'equazione che lega la distanza da un nodo trasmettitore all'attenuazione della potenza da parte del canale trasmissivo. I due metodi costituiscono differenti difficoltà: sebbene il primo non abbia bisogno di uno studio accurato a priori per identificare il modello di propagazione nell'ambiente, comporta una spesa in termini di tempo molto elevata poichè maggiori rilevazioni a priori vengono effettuate, maggiore sarà la precisione della localizzazione. Il secondo metodo verrà descritto esaurientemente nei capitoli successivi poichè è stato scelto per la realizzazione del progetto effettuato e descritto in questo articolo.

- *Progetto MoteTrack*: è un progetto sviluppato all'Università di Harvard dal 2004 che utilizza la misura della potenza del segnale elettromagnetico trasmesso dai nodi àncora per determinare la posizione di un nodo mobile. Come sistema di localizzazione di ambienti interni, esso si basa sull'utilizzo di una mappa costruita tramite svariate misure svolte a priori sul campo; in questo modo si è potuto associare ad ogni punto del piano una n-upla di valori di potenza rilevato dal nodo mobile. Come si comprende facilmente, questo algoritmo è figlio del primo metodo utilizzato nel progetto RADAR, ma attraverso uno studio molto accurato, questa soluzione ha trovato uno sviluppo adeguato per dare dei risultati molto affidabili e robusti. L'insieme dei valori dell'RSS rilevati dal nodo mobile, trasmessi assieme all'ID che identifica ciascuno dei nodi àncora trasmettitori, viene denominata *signature*. Durante la fase di localizzazione, il nodo mobile riceve una *signature* e attraverso la mappa costruita nella fase di inizializzazione e una media pesata delle distanze ricevute, riesce ad associare le coordinate della posizione in cui si trova in quel dato momento. A differenza del progetto RADAR, però, si sono attuate molte migliorie che danno all'algoritmo Motetrack una robustezza e delle prestazioni molto migliori che il precedente progetto non aveva. Innanzitutto il fatto che quest'ultimo è un sistema di localizzazione decentralizzato, evitando che l'elaborazione sia compito di un solo nodo; in secondo luogo la mappa, in cui ad ogni valore di potenza è associata una coordinata nello spazio, viene memorizzata fra i nodi della rete in modo da minimizzare sia la memoria occupata sia le ripetizioni dei dati raccolti, in modo che l'algoritmo riesca a funzionare anche nel caso di rottura di un nodo o perdita di dati in ricezione. Rimane però il limite di questa tecnica dato che necessita di una lunga fase di inizializzazione della rete di nodi che deve essere rieffettuata ad ogni modifica o cambiamento dell'allocazione dei nodi àncora.

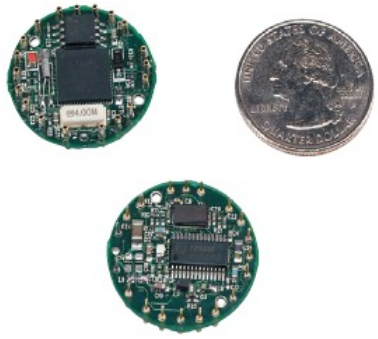


Fig. 7. Un esempio dei sensori utilizzati nel progetto MoteTrack

- *Progetto RIPS (Radio Interferometric Positioning System)* [32]: è un progetto portato avanti dai ricercatori dell'università di Vanderbilt (Nashville ,USA) che si basa su un'idea molto diversa dai precedenti lavori; come grandezza fisica utilizzata per localizzare la posizione di un nodo mobile

si è pensato di utilizzare la fase di un segnale elettromagnetico generato dall'interferenza che si crea tra 2 segnali sinusoidali emessi dai nodi àncora. Sebbene questo algoritmo di localizzazione porti ad ottenere un errore molto inferiore rispetto ai precedenti metodi, due sono le problematiche principali che si riscontrano in questo progetto e che non hanno permesso ad esso una maggiore diffusione: il *primo* è che non devono nè essere presenti ostacoli tra i nodi nè può essere utilizzato in ambienti chiusi a causa del fenomeno delle riflessioni delle onde che causa incertezze nella misura della fase; il *secondo* è dovuto proprio a quest'ultimo fattore cioè per ottenere un errore di localizzazione molto piccolo, è necessaria una misura della differenza di fase molto accurata e ciò è possibile solo se i nodi sono sincronizzati a livello del microsecondo (problematica molto simile a quella del *GPS*). Benchè queste questioni siano molto complesse da affrontare, questo metodo potrebbe risultare molto interessante in un prossimo futuro dato che si stanno studiando algoritmi innovativi e robusti per ottenere la sincronizzazione di più dispositivi in tempi molto brevi e con errori sempre più trascurabili.

Dopo questa carrellata di progetti che hanno in passato cercato di affrontare e risolvere il problema di localizzare un oggetto nello spazio, si è passati ad analizzare i due algoritmi che sono stati sviluppati lo scorso anno all'Università di Padova per il corso di Progettazione di Sistemi di Controllo con lo scopo di determinare la posizione di un operatore all'interno dell'INFN (Istituto Nazionale di Fisica Nucleare) di Legnaro (PD). Questi due progetti furono compiuti per poter permettere, in caso di incendio o di qualsiasi emergenza, di localizzare in tempo reale le persone all'interno dei laboratori e le zone di pericolo, in modo da facilitare il compito degli operatori di soccorso e di guidarli dall'esterno in modo da rendere il loro intervento più rapido e preciso.
L'approccio che è stato usato per ottenere questo fine è molto differente e si sono utilizzate tecniche molto diverse per la localizzazione del nodo mobile, sebbene l'architettura in uso fosse la stessa. Successivamente si riportano la tecnica di localizzazione su cui si basano questi due algoritmi soffermandosi brevemente sui vantaggi e svantaggi che ciascun metodo porta ad avere, allo scopo di scegliere il progetto più adatto da cui partire per l'applicazione a noi assegnata.

1) Progetti sviluppati a Padova nel 2007:

1) *BOZ Algorithm* [7]: questo algoritmo si pone come sviluppo del primo metodo utilizzato nel progetto RADAR e successivamente sviluppato in molti studi teorico-simulativi effettuati in molte facoltà tra cui quella di Milano [9]. Esso si basa su molteplici misure eseguite sul campo e sulle informazioni a priori relative alla topologia della rete WSN. L'idea innovativa è quella di modellizzare il movimento x_t effettuato dal nodo mobile come un processo markoviano del primo ordine del tipo:

$$x_t = x_{t-1} + v_t,$$

in cui v_t rappresenta il processo di guida. In questo modo si può utilizzare un filtraggio bayesiano per evitare che la variabilità della misura della potenza e i molteplici rumori presenti nel campo portino a localizzare il nodo in punti dello spazio drasticamente lontani (*false localization*), cosa che accade molto spesso quando si stima la posizione solamente minimizzando in tempo reale i dati misurati e i dati memorizzati che istante per istante vengono acquisiti dal sensore. Il procedimento per realizzare questo metodo di localizzazione si può brevemente suddividere in due fasi:

   a) *Fase di misurazione off-line*: qui avviene l'inizializzazione della rete WSN che necessita di un lungo lavoro di calibrazione per la creazione della mappa di valori di tutto il campo da localizzare; per fare ciò, devono essere effettuate svariate misurazioni in cui il nodo mobile viene posto in N differenti posizioni ed in ognuna delle quali viene calcolata media e varianza della potenza del segnale ricevuto (*RSSI*) da ciascun nodo àncora che è stato precedentemente posizionato nello spazio. A questa si è aggiunta la probabilità che il pacchetto venga perso nella trasmissione e ciò aumenta ancora di più la varianza dell'errore di localizzazione. Fatto ciò, per aumentare la distribuzione dei punti della mappa e quindi la risoluzione della stessa, si interpolano i dati ottenuti ricavando la mappa necessaria per associare ad ogni insieme di M misure ricevute dal sensore nella fase on-line di tracking, la posizione del nodo mobile nel campo.
   b) *Fase di Tracking on-line*: questo stadio riguarda la fase real-time di localizzazione in cui ad ogni istante di tempo *t*, ogni nodo àncora misura il valore dell'*RSSI* ricevuto dal nodo mobile. Attraverso un metodo che utilizza un Filtro Particellare (*PF - Particle*

Filter), vengono confrontate le misure ricevute con le mappe a priori costruite nella prima fase e questo permette di stimare la posizione del nodo mobile e localizzarlo nella pianta dell'edificio precedentemente disegnata. Il comportamento dell'operatore, cioè del nodo mobile, è stato modellizzato come una passeggiata aleatoria in modo da rimuovere le *false localization* attraverso un filtraggio di tipo particellare in cui il peso di ogni particella è costituito non solo dalla distanza fra la misura e il valore atteso dal modello statistico, ma anche dall'affidabilità dei pacchetti ricevuti.

Questo progetto ha portato ad ottimi risultati in campo della localizzazione in spazi molto ampi e con un numero di sensori abbastanza esiguo, ma attraverso alcune analisi si è rilevato poco consono alla nostra applicazione. Essendo queste ipotesi non provate sperimentalmente, questo metodo di localizzazione, con gli adattamenti e modifiche opportune, potrebbe risultare una valida alternativa alla tecnica scelta e discussa in questo articolo e portare a risultati interessanti.

2) *Teseo* [8]: questo algoritmo di localizzazione si basa sul calcolo della distanza del nodo mobile da ciascun nodo àncora utilizzando solamente l'informazione dell'*RSSI*, cioè la potenza del segnale radio trasmesso dal nodo fisso e ricevuto dal mobile; questo procedimento è possibile dato il fatto che nella propagazione di un segnale, l'energia di questo diminuisce in relazione alla distanza che ha percorso con legge logaritmica. Sebbene questa relazione sia affetta da moltissimi rumori, che si notano soprattutto in ambienti interni, tra cui la *diffrazione*, la *riflessione*, lo *scattering* e le *interferenze* delle onde elettromagnetiche (si veda [30] per maggiori approfondimenti), la non idealità del canale trasmissivo e gli svariati disturbi dovuti alla vicinanza di apparecchiature elettriche e altro, in generale è valido il seguente modello log-normale del canale che è comunemente accettato sulla base delle evidenze empiriche e sperimentali:

$$P_r(d)[dBm] = P_0(d_0)[dBm] - 10 \cdot n_p \cdot log_{10}\left(\frac{d}{d_0}\right) + \chi_\sigma, \quad (1)$$

in cui:

a) $P_r(d)$: potenza ricevuta dal nodo mobile in *[dBm]*: questa notazione sarà mantenuta per tutto questo articolo e denota la potenza in decibel di milliwatt;

b) $P_0(d_0)$: potenza di riferimento ad una data distanza d_0 dal trasmettitore sempre espressa in *[dBm]*;

c) n_p: fattore di decrescenza che misura il rate con cui decresce l'*RSS* con la distanza e questo valore dipende dall'ambiente e dall'applicazione specifica ed è affetto da variazioni molto importanti;

d) χ_σ: variabile aleatoria gaussiana di media nulla e deviazione standard σ che tiene conto di tutti quegli effetti stocastici e difficilmente prevedibili presenti nel campo.

L'identificazione di questi parametri è fondamentale per la buona riuscita del progetto e per questo motivo è stato speso diverso tempo per il suo studio e l'analisi, soprattutto per via sperimentale essendo questi coefficienti molto variabili tra diverse applicazioni e ambienti. Si rimanda alla sezione dell'identificazione del canale per la descrizione più dettagliata del lavoro affrontato.

Rimane un'ultima questione da chiarire brevemente: determinate le distanze del nodo mobile da ciascun nodo àncora, la tecnica che si effettua per localizzare in un punto della mappa la sua posizione è denominata *triangolazione* (in questo caso particolare si chiama *trilaterazione* poichè intersezione di più circonferenze), utilizzata per qualsiasi procedimento in cui si riesce a ricostruire la distanza d_{ij} che intercorre tra i 2 nodi *i* e *j*. Partendo dalla conoscenza della posizione di ciascun nodo àncora, che denominiamo (x_i, y_i), $i = 1, ..., n$ supposto n il numero di nodi fissi presenti nel campo, attraverso il metodo del calcolo dell'RSS descritto precedentemente si ricavano le misure, mediate opportunamente, della distanza d_i di ciascun nodo i-esimo dal nodo mobile di posizione incognita (P_x, P_y). Ora, calcolando il seguente semplice sistema:

$$\begin{cases} (x_1 - P_x)^2 + (y_1 - P_y)^2 = d_1^2 \\ (x_2 - P_x)^2 + (y_2 - P_y)^2 = d_2^2 \\ \vdots \\ (x_n - P_x)^2 + (y_n - P_y)^2 = d_n^2 \end{cases}$$

si può ricavare le coordinate del nodo mobile. Come si può notare facilmente, il sistema è decisamente sovraparametrizzato visto che basterebbero solo 3 equazioni per trovare il valore delle incognite; essendo però le misure delle distanze affette da rumore, facendo così si ha una misura più robusta e meno variabile a seconda dei nodi scelti, pagando però in una complessità di calcolo molto maggiore ma che può essere facilmente superata date le prestazioni degli attuali microcontrollori di cui sono dotati i sensori. Se però il numero n di sensori è molto elevato (nel nostro caso consta in $n = 64$), si è preferito calcolare la posizione del nodo mobile acquisendo solamente le misure dei nodi più

vicini, date le loro minori fluttuazioni di potenza, formando così una *cella* di nodi àncora da cui calcolare P_x e P_y. Si veda il capitolo del software per maggiori chiarimenti. Ora, per implementare questo calcolo al meglio in forma digitale, e attuare una stima dalle misure affette da rumore sovrapposto, ipotizzando che quest'ultimo sia gaussiano, si utilizza spesso la tecnica di *Minimi Quadrati*: dato che il sistema descritto si può riportare nella forma:

$$\mathbf{AP} = \mathbf{b} \text{ con } \mathbf{P} = [P_x \ P_y]^T .$$

Facilmente si arriva al seguente risultato (per i calcoli si rimanda a [11]):

$$\mathbf{P} = \arg \min_{(P_x,P_y)} \|\mathbf{AT} - \mathbf{b}\| = (\mathbf{A^T A})^{-1}\mathbf{A^T b}$$

. Questa tecnica è stata descritta poichè è la più semplice che si può utilizzare per questo scopo, ma se la rumorosità del canale e le fluttuazioni delle misure dell'RSS risultano molto marcate, le prestazioni di questo metodo degradono molto velocemente.

Per l'algoritmo di localizzazione si è scelto di attuare una stima del *Massima Verosimiglianza (M.V.)* sia per la distanza stimata da inviare al nodo centrale per l'elaborazione, sia per la stima della posizione poichè i risultati in simulazione davano una maggior precisione rispetto al metodo precedente, sempre ipotizzando che le misure siano affette da rumore gaussiano. Dal metodo descritto teoricamente in [11] e sviluppato per questa applicazione in [8], la distanza stimata si calcola come:

$$\hat{d}^N_{i,j} = d_0 \cdot 10^{\frac{P(d_0) - \hat{P}_{i,j}(N)}{10 \cdot n_p}}$$

in cui:

1) $P(d_0)$: potenza ricevuta alla distanza di riferimento d_0;
2) $\hat{P}_{i,j}(N)$: media campionaria calcolata su N che è il numero di pacchetti ricevuti prima di effettuare la stima;
3) n_p: fattore di decrescenza del modello del canale.

Dalle stime delle distanze che vengono ricevute dal nodo mobile, si calcola la stima della posizione sempre utilizzando la stessa tecnica di M.V.:

$$\hat{\mathbf{P}}_i = (\hat{P}_{x_i}, \hat{P}_{y_i}, \hat{P}_{z_i}) = \arg \min_{(P_{x_i},P_{y_i},P_{z_i})} \sum_{j \in C_i} \left(\ln \frac{\hat{d}^2_{i,j}}{d^2_{i,j}} \right)^2$$

in cui:

1) $\hat{d}^2_{i,j}$: è la stima della distanza tra i nodi i e j calcolata con il metodo precedente; il nodo i è il nodo mobile da localizzare mentre i nodi j sono tutti i nodi àncora appartenenti alla cella;
2) C_i: è l'insieme dei nodi che creano la cella del nodo i-esimo; essa è scelta in relazione alla potenza (*RSS*) e alla qualità (*LQI - Level Quality Indicator*) del segnale ricevuto.

Con questa espressione, si è calcolata la posizione in real-time del nodo mobile nella piattaforma utilizzata in questo progetto. Si va ora ad analizzare più in dettaglio il software utilizzato descrivendo le analisi e gli studi effettuati in questi mesi soffermandosi soprattutto sull'identificazione del modello del canale radio più realistico possibile e sulle modifiche effettuate in modo da ottenere un algoritmo più robusto ed efficiente possibile per le future applicazioni in cui si renda necessaria l'acquisizione della posizione di un oggetto all'interno dello spazio.

B. Software di localizzazione

Nel seguente paragrafo si presenteranno i codici sorgenti, con i relativi software di implementazione, utilizzati per il funzionamento dell'algoritmo di localizzazione fornito da [8], si consiglia di far riferimento a quest'ultimo per un accurato approfondimento.
La prima parte descriverà il codice NESC opportunamente modificato ed installato nei nodi TMOTESKY presenti nell'ambiente di lavoro. Si passerà poi ad una breve descrizione del client JAVA, fornito da [8], utilizzato per la visualizzazione a PC della posizione del nodo mobile, posto sopra l'e-puck. Infine, si descriverà il codice utilizzato per interfacciare l'e-puck con il programma MATLAB, utilizzato per l'invio dei comandi di movimento all'uniciclo.
Inizialmente, per l'installazione del codice sui mote e per le prove generali dell'algoritmo di localizzazione, si è utilizzato, su sistema operativo WINDOWS XP, il *VMware Player*, un programma free facilmente scaricabile ed installabile, presente nel sito della TINYOS [16]. Il *VMware Player* altro non è che una virtual machine del sistema operativo XUBUNTOS, corredata di tutti i pacchetti del TINYOS 2.0.1, utilizzati nel corso del progetto. Tale sistema si è verificato leggero ed efficace solo nella fase di simulazione dove, per le prove, non venivano utilizzati più di 5 mote come nodi àncora, pari al limite fisico dell'emulatore. In fase di sperimentazione, dove la rete presentava 64 sensori, si è dovuti migrare ad una postazione fissa presente nel laboratorio NAVLAB, in cui si è utilizzata la versione di XUBUNTOS 2.0.1, un vero è proprio sistema operativo. Lo spostamento, ha portato a svariati problemi di configurazione del PC e della rete di sensori, che una volta risolti, si è comunque

dimostrata un ottima soluzione di lavoro.

1) Introduzione al NesC: Mentre il TINYOS costituisce l'insieme delle regole e l'architettura su cui si basa il sistema operativo, il linguaggio utilizzato per la scrittura dei componenti è il NESC, un estensione del C. Il NESC è un linguaggio ad eventi progettato per la programmazione component-based principalmente per sistemi embedded come reti di sensori.
In generale, un'applicazione NESC è un insieme di componenti collegati tramite interfacce, dichiarazioni di servizi che un componente fornisce o richiede ad altri componenti, in questo modo si separa la progettazione logica dalla specifica realizzazione dei componenti stessi. Inoltre, esistono due tipi di implementazione di componenti: i *moduli* e le *configurazioni*. I moduli implementano le funzionalità delle interfacce (comandi ed eventi), mentre le configurazioni implementano componenti dichiarando una serie di sotto componenti e definendo i collegamenti tra le loro interfacce. Durante la compilazione di un'applicazione, i componenti del TinyOS e quelli dell'applicazione, vengono assemblati insieme, il risultato ottenuto costituisce l'intero software del nodo.
Il programma di localizzazione, utilizzato nel progetto, presenta due tipi di applicazione: il *Mobile Node*, relativo al nodo mobile posizionato sopra all'uniciclo, e l'*Anchor Node* da installare nei 64 nodi fissi posizionati nella rete descritta nel capitolo dedicato all'architettura del sistema. Le costanti e le tipologie dei messaggi utilizzati nelle due applicazioni, sono state definite in un *header file* denominato *localization.h*. La loro definizione può essere interpretata utilizzando l'ottima descrizione presente nel sorgente *.h* oppure grazie a [8].

2) NesC del nodo mobile: L'applicazione installata nel nodo mobile, come spiegato, è divisa in due sorgenti NESC: *MobileNodeC.nc*, relativo alla configurazione, e *MobileNodeP.nc*, relativo al modulo. Visto che lo scopo del file di configurazione è di delineare, sia al programmatore che al compilatore, come le varie componenti siano connesse tra di loro, si è deciso di trattare il solo file del modulo. Nella descrizione, si farà riferimento ad una simulazione di una normale routine di funzionamento.
Il codice sorgente del nodo mobile prevede, per prima cosa, ad inizializzare le periferiche e l'ambiente attraverso la funzione *booted()*, portando il nodo mobile in attesa di ricevere il comando di start dall'utente. Inoltre, viene impostata la frequenza di trasmissione, se tale operazione è andata a buon fine, vengono successivamente avviate la comunicazione radio e seriale. Il mote, per segnalare che l'avvio ha avuto successo, si presenterà con i tre led accesi, in caso contrario vi potrà essere un problema di inizializzazione delle periferiche. Se le notifiche risultato positive, viene settata la potenza di trasmissione dei vari *ping messages*. Appena arriva il segnale di avvio dall'utente, inizia l'algoritmo di localizzazione, informando il PC attraverso dei messaggi seriali. Il nodo mobile invia in broadcast ai nodi àncora della rete diversi ping, una volta terminati si porta in stato di ascolto. Dopo un tempo necessario ai nodi fissi per stimare la distanza dal mote e formare una cella preliminare, il nodo mobile comincia a ricevere i *data messages* salvandoli in una coda di tipo FIFO. Attraverso un apposito task, i *data messages*, vengono prelevati dalla coda e inoltrati al client tramite seriale. Terminate tali operazioni, sincronizzate da un definito timer, il nodo mobile ritorna alle condizioni iniziali, pronto per iniziare un nuovo processo di localizzazione.

3) NesC dei nodi àncora: Come per il mote posto sopra l'uniciclo, anche l'applicazione dei nodi àncora è divisa in due sorgenti: *AnchorNodeC.nc*, per quanto concerne la configurazione, e *AnchorNodeP.nc* riguardante il modulo e di cui si descriveranno le operazioni svolte durante un normale funzionamento.
All'accensione del nodo fisso, viene avviata la sequenza di *boot*, portando il nodo nello stato di ascolto, cioè in attesa di ricevere dei *ping messages* dal nodo mobile, tramite contatto radio. Viene impostata la frequenza di trasmissione e avviate in successione la radio e la comunicazione seriale. Se le notifiche hanno dato esito positive viene settata la potenza con cui il nodo àncora trasmetterà i *data messages* al nodo mobile. Quando il mote riceve i *ping messages* dal mobile, ricava da essi l'RSSI e l'LQI mediante due task presenti nell'interfaccia radio. I messaggi che hanno valori di RSSI e LQI superiori ad una certa soglia, impostata precedentemente, vengono salvati in una coda. Dopo un prefissato tempo, impostato secondo una precisa logica (si veda [8]), viene avviata la stima della distanza. Dopo di che il nodo fisso passa allo stato *estimation*, calcolando il valore medio di tutti gli RSS salvati nella coda. Tale valore viene poi passato ad un task che calcola, con il metodo di stima di *Massima Verosimiglianza*, la distanza. Disponendo della distanza il mote passa allo stato *send*, e chiama un altro task che ha il compito di avviare il sending dei *data messages* al nodo mobile. Per ridurre la possibilità di collisione dei *data messages* inviati dai nodi fissi appartenenti ad una stessa cella, l'istante in cui avviene la spedizione viene determinata in maniera aleatoria attraverso l'interfaccia

Random. Il nodo àncora invia dunque i *data messages* al nodo mobile, controllando che i messaggi inviati accettati sia superiore ad una certa soglia. In caso affermativo ritorna in stato di ascolto, altrimenti, prima di ritornare pronto per successivi broadcast, si esclude dalla cella.

4) Script per l'installazione della rete di sensori: Visto l'elevato numero di nodi àncora presenti nella rete di sensori, è stato creato uno script in linguaggio di programmazione della shell LINUX, per facilitarne l'installazione. Sulla shell di XUBUNTOS, grazie al comando *motelist*, è possibile visualizzare i mote presenti nella rete. Ogni nodo della rete viene identificato dal suo ID e dal numero di USB a cui si trova collegato, variabile ad ogni accensione della rete. Per l'utilizzo dello script, è necessario per prima cosa salvare la lista dei mote comparsa nella shell in un file *.txt*, il quale verrà poi passato come ingresso al programma. Lo script, leggerà dal file *.txt* i due dati identificativi di ogni mote, facendo partire sequenzialmente nella shell le riga di comando di installazione di ciascun TMOTE prevista dal TINYOS.

5) L'interfaccia Java Teseo: Per la visualizzazione in tempo reale della posizione dell'e-puck all'interno della piattaforma, è stato utilizzato il frame Java TESEO. La descrizione dettagliata del codice sorgente si può trovare in [8]. Prima dell'avvio del frame, è necessario creare la mappa di rappresentazione dell'ambiente di lavoro, indicando la posizione dei nodi àncora al suo interno. La mappa dovrà essere salvata nel formato *.jpg* con una risoluzione di 640x480. In aggiunta all'immagine, è necessaria la creazione di un file di testo, salvato con estensione *.map*. Quest'ultimo, viene utilizzato da TESEO per la visualizzazione dei nodi àncora nella mappa. In esso si dovrà quindi riportare, come da esempio presente, la posizione, in metri, dei mote àncora disposti nella griglia, ed il centro della mappa, questa volta indicato in pixel. Per il corretto funzionamento, i due file dovranno essere salvati nelle cartelle *maps* presenti in *classes* e *src* (sources), dell'applicazione di TESEO.
Dall'apposita shell di comando XUBUNTOS, si procede prima alla compilazione del software, attraverso il comando $./make.sh$, per poi lanciarlo, grazie al comando $./run.sh$. All'avvio, il TESEO si presenterà come in *figura 8*.

Attraverso il tasto *start*, presente nella consolle di comando, si invia il comando di inizializzazione dell'algoritmo di localizzazione al nodo mobile, che

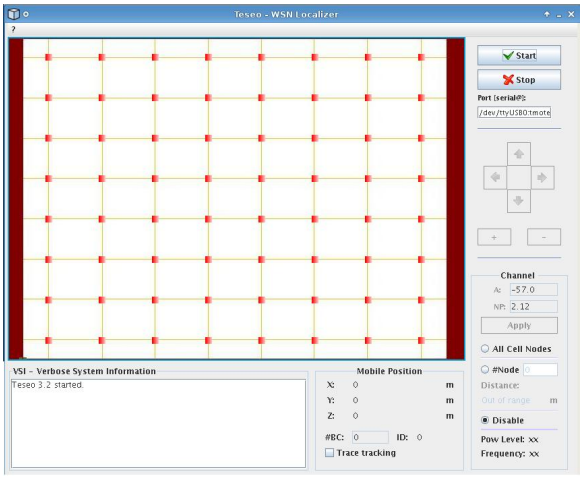


Fig. 8. Schermata iniziale di Teseo

di default è collegato alla porta USB0, ma che può essere facilmente cambiata nel caso la porta utilizzata sia un'altra. Appena il nodo dell'uniciclo non si trova più nello stato di quiete, il frame diviene sensibile alla ricezione di messaggi trasmessi via seriale dal mote stesso, visualizzando a PC la posizione stimata dai nodi àncora. In TESEO vi è inoltre un pannello molto utile per il debug del sistema, in quanto consente di trasferire a Java il compito di calcolare la distanza dei nodi fissi dal nodo mobile, mediante lo stimatore di Massima Verosimiglianza. In particolar modo permette di impostare i valori di A e n_p della potenza di trasmissione (si rimanda al parte dedicata all'identificazione). In questo modo è possibile ricavare i valori opportuni del canale, senza dover riprogrammare ad uno a uno i nodi, risparmiando così un notevole quantitativo di tempo. A disposizione dell'utente vi è un ulteriore pannello che visualizza le coordinate della posizione corrente dell'e-puck, con la possibilità di visualizzare la traccia del percorso fatto fino a quell'istante.

C. Interfaccia E-Puck Matlab

Per riuscire a comandare l'uniciclo attraverso dei semplici comandi Matlab è stata implementata un'apposita interfaccia grazie al software *MpLab 8.0*, disponibile in [15]. L'idea principale è di sfruttare la comunicazione *bluetooth*, disponibile nell'e-puck, con il PC. Questo ha permesso di inviare i comandi all'uniciclo in modo molto rapido, e riducendo la possibilità di perdita di pacchetti durante la trasmissione. L'e-puck può rispondere all'aggiornamento dei comandi inviati tramite *bluetooth* con un tempo minimo di 0,05 secondi, nel caso specifico si è scelto di aggiornare i comandi Matlab spediti all'uniciclo dopo 0,1 secondi.
L'interfaccia utilizzata a tale scopo è disponibile in [17],

chiamata *BTcom_default.hex*, contenente tutti i pacchetti utili al corretto funzionamento dell'applicazione. Una volta installata l'applicazione nell'e-puck, è stato possibile interagire con esso attraverso delle funzioni Matlab, fornite dal pacchetto *e-pic*, disponibile nella sezione software di [15].
Per il funzionamento, la prima funzione che deve essere richiamata in Matlab è la seguente: *epic=ePicKernel*; la quale crea una classe contenente tutte le variabili necessarie per comandare l'e-puck. Fatto questo, si deve collegare l'e-puck al PC tramite *bluetooth*; a tale scopo si utilizza la funzione *epic=connect(epic,'COMXX')*, dove *XX* indica il numero della porta seriale del PC con cui ci si collega all'uniciclo. Per interrompere la connessione si utilizza invece *epic=disconnect(epic)*. Una volta che l'uniciclo è connesso a Matlab, si può iniziare a comandarlo. In particolare nel progetto sono stati utilizzati i comandi di marcia e di arresto dell'e-puck. Il comando *write(epic,'D,left,right')*, consente di settare la velocità del motore delle ruote con un valore compreso tra 0 e 1000 passi al secondo. Attraverso degli opportuni valori, imposti alle due ruote, è possibile far eseguire all'e-puck dei percorsi rettilinei oppure delle circonferenze. Il secondo comando utile riguarda l'arresto dell'uniciclo, possibile attraverso la funzione *write(epic,'R')*. Per un maggior approfondimento delle funzioni elencate, per la ricerca di ulteriori funzioni disponibili dall'*e-pic* e per delle linee guida su come implementare codici sorgente con *MpLab 8.0*, si rimanda rispettivamente a: [18], [19], [20] e [21].
Questa interfaccia è stata utilizzata sopratutto in fase di test della rete di sensori, cui risultati verranno presentanti successivamente.

D. Identificazione Del Canale Di Trasmissione

La parte più importante per il buon funzionamento di questa tecnica di localizzazione sta nella misura della distanza tra il nodo mobile e ciascun nodo àncora. Da una prima analisi si può subito rendersi conto come le misure dell'RSS del segnale inviato da un mote all'altro sono affette da una somma di rumori e disturbi molto vari che causano degli errori di misura di valore molto elevato e lontano da quello "reale". Questi disturbi sono impossibili da prevedere e da modellizzare e quindi è necessaria un'analisi statistica dei dati acquisiti che cerchi di mediare correttamente le misure pervenute al nodo mobile.

1) Il programma Connectivity: Il software utilizzato per queste prove è stato pensato per la trasmissione in *Broadcast* di un pacchetto via radio frequenza; come tutti i programmi scritti in *Nes-C* è costituito da due file da installare in ciascun nodo: *MobileNodeC* e *MobileNodeP* per il nodo mobile che invia i pacchetti radio ad una potenza variabile modificabile nel file *Header.h*; *AnchorNodeC* e *AnchorNodeP* per il nodo fisso che diversamente dall'algoritmo di localizzazione è stato pensato essere il ricevente dei messaggi radio; in questo modo è stato possibile effettuare le misure della potenza direttamente sulla piattaforma e attraverso un'unica procedura acquisire le potenze dei pacchetti inviati dal nodo mobile a tutti i nodi ancora presenti senza ripetere l'invio per ogni misura e con la possibilità di effettuarle anche muovendo a piacere il nodo trasmettitore. L'inizializzazione del programma di misura è costituito dai nodi àncora in ascolto, mentre il nodo mobile aspetta il segnale dall'utente per iniziare a trasmettere i pacchetti via radio. Per avviare la procedura di invio in broadcast dei messaggi è necessario premere il tasto *USER* presente nel sensore e il nodo mobile invia sequenzialmente, uno ogni 0.5 secondi, un numero di pacchetti via radio che abbiamo scelto essere pari a 120; in questo modo i nodi àncora memorizzano nella loro *flash*, di cui è munito ogni mote, la potenza RSS di ogni pacchetto radio che ricevono fino a che non termina la trasmissione. Premendo ora lo *USER-botton* per ogni nodo àncora, si trasmettono via Seriale i valori dell'RSS ricevuti e il computer a cui è connessa l'USB genera un file di testo da cui è possibile leggere le misure della potenza del segnale ricevuto dal nodo in questione. Grazie a questo, è stato possibile effettuare svariate misurazioni per comprendere l'andamento della potenza in relazione della distanza dei 2 mote. Le misure acquisite sono state poi plottate attraverso vari grafici utilizzando MATLAB.

2) Prove tra 2 sensori: Sono state effettuate, per uno studio iniziale, diverse misurazioni tra due mote singoli: un trasmettitore e un ricevente. Avendo la possibilità nei sensori di scegliere il livello di potenza del segnale di trasmissione, in dBm, sono stati utilizzati 4 differenti livelli che spaziano dal più basso al massimo raggiungibile con questi dispositivi. Nella tabella II si riportano i 4 valori scelti per le prove.

TABLE II
LIVELLI DI POTENZA DI TRASMISSIONE SCELTI PER LE PROVE INIZIALI

LIVELLO	P_{TX} $[dBm]$
31	0
23	-3
11	-10
3	-25

Dalle prove fatte in questa parte è risultato come per ogni livello di potenza esistano dei range di valori in cui le misure sono sufficientemente stabili per una stima di posizione corretta; al di fuori di questa fascia di valori, che è molto variabile sia da luogo a luogo, ma anche da mote a mote, la misura comincia ad avere fluttuazioni molto varie e da questa si ricaverebbe una stima della distanza con una varianza d'errore molto elevata. Si riportano in figura 9 l'andamento dell'RSS di 2 mote appoggiati su un tavolo senza ostacoli e posti a 5 differenti lunghezze che si riportano in legenda con i corrispondenti colori. Il livello di potenza scelto, il 23 che corrisponde ad una potenza di trasmissione pari a $-3\ [dBm]$, risulta in questo caso il più esemplificativo; i grafici relativi agli altri livelli si riportano in appendice.

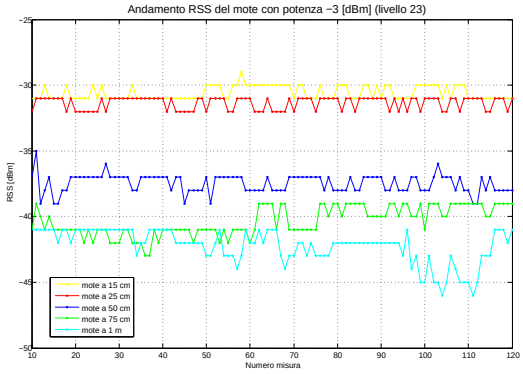


Fig. 9. Grafico dell'andamento dell'RSS misurato tra 2 mote

Osservando attentamente la figura 9 si può notare come in queste misurazioni i valori di ciascuna potenza ricevuta stiano tutti all'interno di un range massimo di 5 dBm che si riduce ancora di più nel caso che i 2 mote fossero posti ad una distanza minore di 50 cm. Le misure di 15 e 25 cm per la maggior parte risultano sovrapposte ma se andiamo ad analizzare la media sui 120 pacchetti ricevuti risulta che:

$$\overline{RSS}_{15cm} = -30.55[dBm]$$

$$\overline{RSS}_{25cm} = -31.35[dBm]$$

e i due valori si differiscono di $1\ [dBm]$. Un comportamento molto meno stabile risultano avere le misure con i 2 mote posti a $75cm$ e $1m$ tra loro: si può osservare come nei primi 50 pacchetti siano quasi coincidenti per poi separarsi in due range di valori molto differenti. Questo non è da incriminare ad ostacoli temporanei o a misure scorrette durante l'acquisizione, ma alla non idealità del canale trasmissivo e al tempo necessario che spesso richiedono i sensori affinchè le misurazioni si stabilizzino nella misura corretta. Sono stati plottati solo i pacchetti ricevuti successivamente al decimo, dato che le misure risultano affette da errori maggiori causati molto probabilmente dalla presenza dell'ostacolo fisico creato dal premere il tasto nel mote per iniziare la procedura di misurazione; in ogni caso però la potenza risultava all'interno della fascia di 10 $[dBm]$ dal valore medio e le misure scartate non sono state riportate solamente per un fatto di chiarezza e per avere una scala maggiore della figura.

3) Prove effettuate direttamente sul campo: Successivamente si è passati ad eseguire il maggior numero possibile di prove direttamente sui mote montati nella piattaforma. Superate le varie problematiche inerenti all'implementazione e al corretto funzionamento delle interconnessioni tra i mote stessi e tra i sensori e il sistema di elaborazione centrale, si è deciso di effettuare le misurazioni in una zona rettangolare del piano di misura che comprendeva 9 sensori e si è posto il nodo mobile in 13 punti differenti come spiegato nella figura 11.

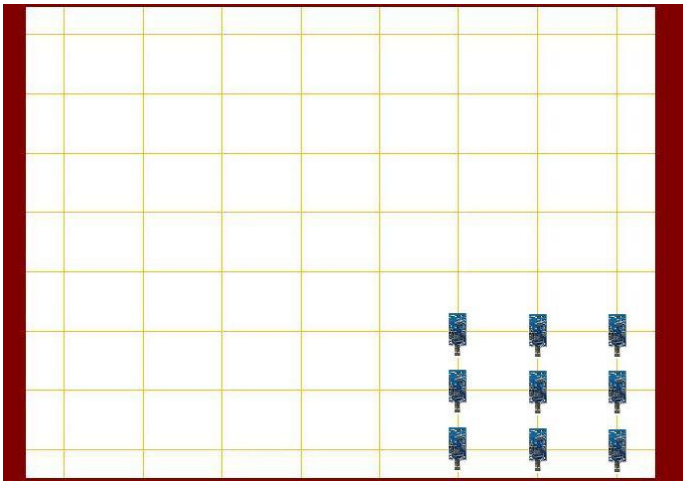


Fig. 10. Mote àncora attivi per le misurazioni

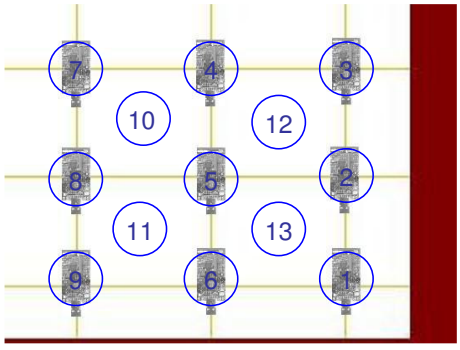


Fig. 11. Disposizione del mote mobile nelle 13 differenti posizioni da cui si sono compiute le misurazioni dell'RSS

Per ogni posizione numerata si sono inviati 120 pacchetti in broadcast dal nodo mobile ai nodi àncora riceventi, ottenendo in questo modo 1080 misurazioni direttamente sul campo per ciascuna delle 13 posizioni per ogni livello di potenza scelto. Si è deciso di scartare il livello 3 poichè analizzando i risultati della prima prova si è osservata una fluttuazione delle misure dell'RSS molto maggiori rispetto agli altri livelli di potenza (si vedano le figure riportate alla fine del capitolo), sebbene all'inizio sembrasse la scelta più indicata per questa applicazione vista la vicinanza e il numero dei sensori presenti. Si riportano di seguito gli andamenti delle misurazioni con il nodo mobile posto nella posizione (1) confrontato con l'RSS acquisito dal mote sovrapposto, dal mote rispettivamente in (2), a distanza di 30 cm, in (3), a distanza di 60 cm, in (4) e in (7), a distanza 75 cm e 1 m dal sensore trasmettente.

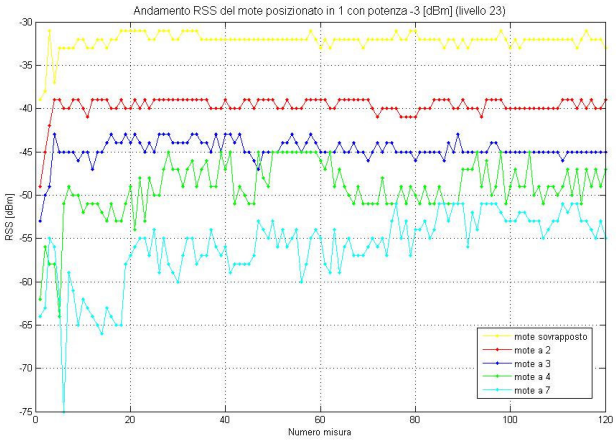


Fig. 12. Andamento dell'RSS del nodo mobile posizionato in (1) con -3 $[dBm]$ (livello 23)

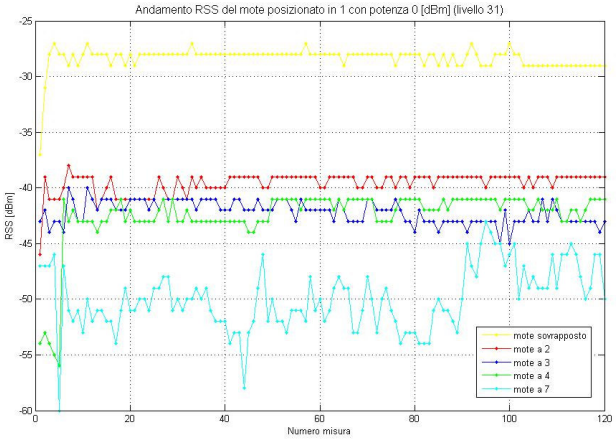


Fig. 13. Andamento dell'RSS del nodo mobile posizionato in (1) con potenza 0 $[dBm]$ (livello 31)

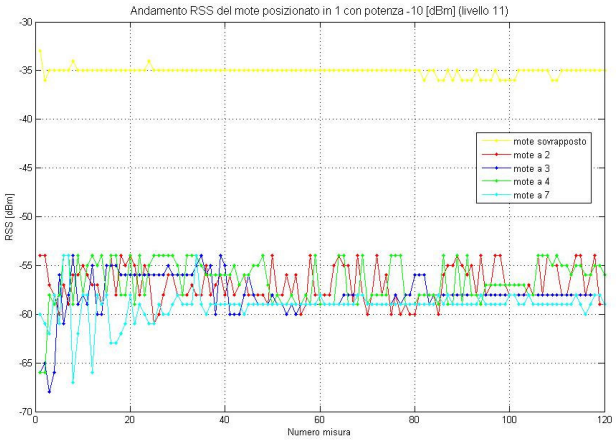


Fig. 14. Andamento dell'RSS del nodo mobile posizionato in (1) con -10 $[dBm]$ (livello 11)

Dalla figura 12 si denota come le misure di potenza abbiano un andamento leggermente peggiore di quello trovato nella prima prova dato che la presenza di altri sensori nelle vicinanze, dei cavi per le connessioni USB e degli *Hub* sotto la piattaforma aumentano i disturbi all'interno del canale di trasmissione aumentando l'incertezza nelle misurazioni della distanza. La situazione è ben peggiore se si analizzano le figure 13 e 14 che sono basate sulla stessa procedura di misura, ma con un livello di potenza diverso dal precedente.

In questi grafici si può osservare come sia utilizzando la potenza massima di trasmissione che il livello 11, la misura posizionando i 2 mote sovrapposti, separati solamente dalla piattaforma di compensato, è stabile e ben divisa e individuabile rispetto agli altri andamenti. Quello che si nota, soprattutto nella seconda figura, è come le misurazioni si sovrappongano e abbiano delle fluttuazioni a volte troppo ampie (si veda ad esempio l'andamento delle misure a livello 31 dei mote a distanza di 1 metro). Questo è dovuto al fatto che la presenza di sensori vicini aumenti l'incertezza della misura di potenza causando così la difficoltà di associare ad essa una distanza ben precisa e questo sarà il fatto principale di una varianza di misura della posizione molto elevata. Se andiamo però ad analizzare le medie delle misurazioni effettuate, si può osservare come i valori mediati abbiano un andamento lineare rispetto alla distanza; l'unica possibilità quindi di ottenere una localizzazione più precisa e robusta sarà quella di acquisire un numero maggiore di misurazioni e inviare al nodo centrale la distanza ricavata dalla media calcolata delle potenze. Questo però avrà l'effetto di un ritardo e un tempo per individuare la

posizione del nodo mobile molto più elevato. Si riporta in tabella III i valori mediati delle potenze misurate.

TABLE III
VALORI DELLE MISURE DELL'RSS MEDIATI SU 120 PACCHETTI UTILIZZANDO IL LIVELLO 31 E 11 DI POTENZA

	LIVELLO 31	LIVELLO 11
POS.1	-28.3 $[dBm]$	-35.1 $[dBm]$
POS.2	-39.55	-56.6
POS.3	-42.1	-56.9
POS.4	-42.5	-58
POS.7	-49.97	-59.14

E. Modello del Canale di Trasmissione

Fino ad ora ci si è limitati ad analizzare gli andamenti delle potenze senza però far riferimento alla distanza a cui ogni misura verrà associata. Si riporta quindi una veloce analisi del modello del canale trasmissivo più opportuno già accennato precedentemente da cui ricavare la distanza del mote ricevente (nel nostro caso il nodo mobile) da ciascun nodo àncora presente sotto la piattaforma costruita e posizionato in coordinate ben precise e conosciute dall'utente. Come già accennato precedentemente, 2 sono i parametri fondamentali che si utilizzano per la localizzazione e misurati dai sensori tmote utilizzati:

1) *RSS*: potenza del segnale radio trasmesso o ricevuto dai sensori legato all'*RSSI - Received Signal Strength Indicator* dall'equazione:
$$RSS = RSSI - 45[dBm]$$
2) *LQI - Link Quality Indicator*: è un indicatore della stabilità e quantità di rumore presente nel canale nella banda in cui avviene la trasmissione.

Sebbene l'LQI si utilizzi solamente come soglia di controllo per accettare o trascurare la misura ricevuta da un nodo àncora, il valore dell'*RSS* risulta fondamentale per la stima della distanza poichè si conosce il modello che lega la potenza del segnale radio ricevuto con la distanza di trasmissione, formula alternativa a (1):

$$P_r(d)[dBm] = P_{TX} + A - 10 \cdot n_p \cdot log_{10}(d) + \chi_\sigma,$$

in cui, oltre ai valori definiti precedentemente, compaiono:

1) P_{TX}: potenza di trasmissione scelta in inizializzazione tramite il livello compreso tra 3 e 31 (si veda la tabella II);
2) $\mathcal{A}$: fattore di attenuazione del canale di trasmissione.

Da questa relazione si comprende che aumentando la distanza d tra i due sensori, si ha una diminuzione della potenza P_r ricevuta dal nodo fisso con legge logaritmica. A parte la presenza della variabile aleatoria χ_σ di media nulla e deviazione standard σ incognita ma derivabile empiricamente, gli unici termini incogniti sono l'attenuazione $\mathcal{A}$ e il fattore di decrescenza n_p poichè sono variabili il cui valore si differenzia anche di molto da applicazione ad applicazione.

1) Identificazione dei parametri: Dalle considerazioni fatte nel paragrafo precedente si comprende come per stimare i valori di $\mathcal{A}$ e n_p basti tracciare la retta interpolatrice $y = mx + q$ di tutte le misure effettuate ed estrapolare il coefficiente angolare m e l'intercetta q con l'asse delle ordinate. Attraverso semplici calcoli, si ricava che:

$$\begin{aligned} m &= -10 \cdot n_p \\ q &= P_{TX} + \mathcal{A} \end{aligned} \tag{2}$$

attraverso cui ricavo la relazione tra la potenza P in funzione dell'ascissa $log(d)$. Per questo calcolo si è utilizzato uno stimatore di Minimi Quadrati (M.Q.) in cui si è definito un modello a regressione lineare:

$$\mathbf{y} = \mathcal{S} \cdot \theta + \mathbf{w}$$

in cui $\mathbf{y} = [y_1 \dots y_N]^T$ è il vettore della potenza RSS misurata da ciascun sensore alla distanza $x_i = log(d_i)$ dal nodo mobile, $\theta = [m\ q]^T$ da stimare e definite nell'equazione (2); la matrice S è definita come:

$$\mathcal{S} = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix}$$

Come si può osservare nella figura 15, ciascun rumore di misura w_i può essere pensato come una variabile aleatoria gaussiana di media nulla e varianza σ_i^2. Questo fatto può essere visto anche come ulteriore validazione della valenza del modello del canale trasmissivo utilizzato anche da misure ricavate direttamente sul campo.

Dati questi fatti, lo stimatore M.Q. si trova risolvendo il seguente calcolo:

$$\hat{\theta} = (\mathcal{S}^T \cdot \mathcal{S})^{-1} \mathcal{S}^T \mathbf{y}.$$

La varianza corrispondente dello stimatore si calcola con:

$$\hat{\sigma}^2 = \frac{1}{N} \left\| \mathbf{y} - \mathcal{S}\hat{\theta} \right\|^2.$$

Le misure necessarie per l'identificazione dei parametri incogniti sono state effettuate con le stesse modalità descritte nel paragrafo precedente, in cui per ogni posizione

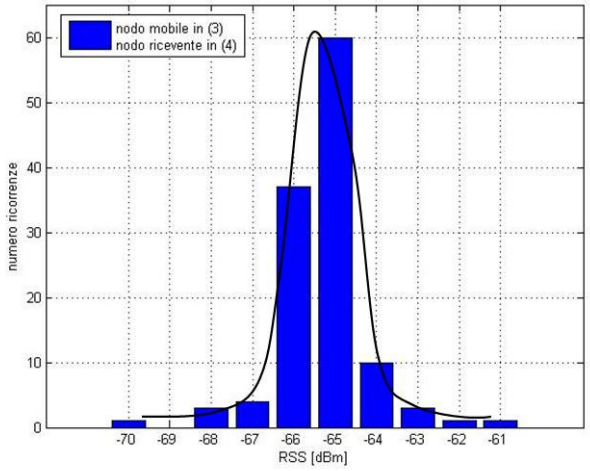


Fig. 15. Andamento delle ricorrenze dell'RSS di una misurazione su 120 pacchetti

dalla (1) alla (13) si sono acquisite le misure di ciascun mote posto nel rettangolo di figura 11; la distanza tra ogni sensore dal nodo mobile è stata facilmente ricavata sperimentalmente e spaziava dai $5\ cm$ che intercorrono tra i due mote sovrapposti, a $1\ m$ tra i mote diagonalmente opposti. Questo procedimento si è effettuato per i 3 livelli di potenza scelti e utilizzati nelle prove sul campo.

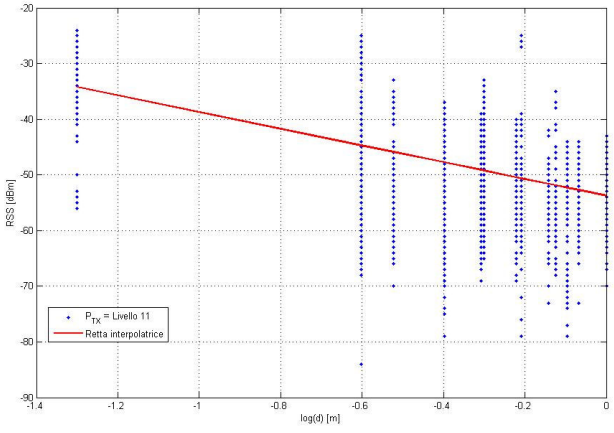


Fig. 16. Andamento del modello del canale trasmissivo con $P_{TX} = -10\ [dBm]$ (livello 11)

I risultati ottenuti dalle misurazioni e riportate nelle figure 16, 17 e 18 denotano degli andamenti molto vari ma le rette risultanti dal metodo ai Minimi Quadrati danno dei risultati finali molto vicini sebbene ci si aspettasse che l'utilizzo di potenze di trasmissione di minore entità generassero minor disturbi e interferenze rispetto a quella massima. Se però andiamo ad analizzare la varianza dello stimatore e la corrispondente deviazione standard, si può osservare che il livello 11 abbia un errore del 10% minore rispetto al 31 e del 5% rispetto al 23, rimanendo però troppo elevato per ottenere una localizzazione accurata. Le tabelle IV e V riportano i risultati numerici ottenuti da questo studio.

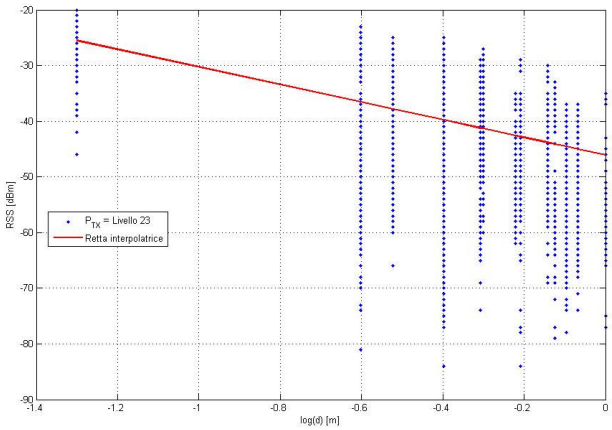


Fig. 17. Andamento del modello del canale trasmissivo con $P_{TX} = -3\ [dBm]$ (livello 23)

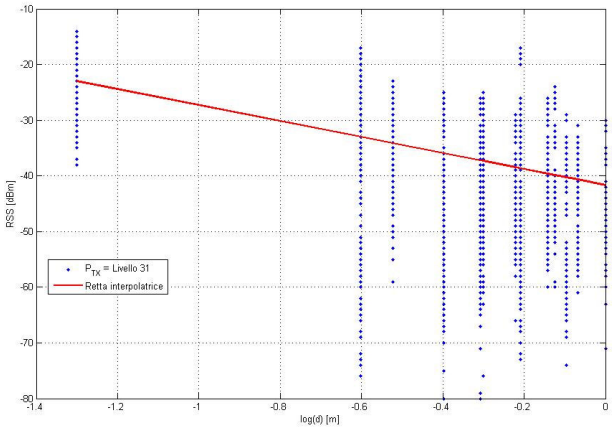


Fig. 18. Andamento del modello del canale trasmissivo con $P_{TX} = 0\ [dBm]$ (livello 31)

Dai risultati ottenuti, quindi, si è settato nel software il livello 11 di potenza poichè portava ad una varianza di errore di stima minore rispetto ai precedenti valori. Ciascun nodo àncora, poi, prima di inviare il pacchetto al nodo mobile per l'elaborazione, effettua una media pesata dei valori misurati su 40 pacchetti: questo stabilizza enormemente la misura dell'RSS, sebbene questo comporti un tempo di elaborazione maggiore dato che la stima della posizione avviene ogni 1.3 secondi. Da

TABLE IV

VALORI DELLA RETTA INTERPOLATRICE $y = mx + q$ CON LA STIMA M.Q. OTTENUTA PER I 3 DIFFERENTI LIVELLI DI POTENZA TRASMESSA

	m	q
Livello 31	-14.36	-41.65
Livello 23	-15.835	-46.09
Livello 11	-15.044	-53.756

TABLE V

VALORE DEI PARAMETRI RISULTANTI DALLE MISURE EFFETTUATE

	n_p	$\mathcal{A}$	$\hat{\sigma}^2$
Livello 31	1.463	-41.65	84.64
Livello 23	1.5835	-43.09	80.4
Livello 11	1.5044	-42.756	71.25

ulteriori prove effettuate, si è osservato come, se si utilizza un sensore differente come nodo mobile, le misure acquisite risultino leggermente differenti, ma al fine della localizzazione e dei parametri del modello del canale trasmissivo, i cambiamenti risultano solamente di qualche percentuale se permangono invariati i nodi àncora installati nella piattaforma. Se però si dovessero sostituire o costruire da zero una nuova pedana, risulta necessaria una nuova taratura dei valori di n_p e $\mathcal{A}$ poichè gli errori ricadrebbero sulla stima della posizione finale compromettendo il buon funzionamento dell'algoritmo e dando luogo a stime errate.

Un'ultima analisi che è risultata necessaria per il software di localizzazione utilizzato, è stata la taratura dei *bounds* sull'RSS e sull'LQI per la creazione della cella; questo serve per inviare al nodo mobile solo i pacchetti di misure robuste e stabili in modo da elaborare solamente le informazioni più utili per la localizzazione. Non avendo in questo caso problematiche di perdita di pacchetti dovute alla scarsa connettività, ma solamente disturbi dovuti alle interferenze e ai rumori del canale di trasmissione impossibili da modellizzare, questi parametri sono stati tarati per via sperimentale tramite varie prove direttamente sul campo. È risultato che l'informazione data dall'LQI non è utilizzabile come soglia di bontà della misura poichè risulta sempre superiore a 105 e né la distanza né le interferenze generate attivando tutti i nodi àncora influissero sul suo valore. Per questi motivi non si è modificato l'$LQI_{lim} = 103$ già impostato nel software per la precedente applicazione. Il *bound* quindi che si è tarato con più difficoltà è stato quello relativo all'RSS, poichè si sono ottenuti comportamenti non facilmente interpretabili né i cambiamenti seguivano una legge ben precisa. Si è osservato che scegliendo un RSS_{lim} di valore molto alto e quindi molto selettivo sui nodi àncora da attivare, la localizzazione spesso perdeva la posizione del nodo mobile e in quei frangenti il Teseo visualizzava nelle coordinate il valore *Nan - Not a number* che si è constatato un baco del software da migliorare in futuro. Se invece si sceglie un RSS_{lim} molto basso e quindi accettando la misurazione dalla maggior parte dei nodi àncora, la stima della posizione, forse per l'esagerata quantità di dati da elaborare e per i valori di potenza poco corretti, risultava sballata con posizioni esageratamente esterni rispetto alla piattaforma. Quindi la difficoltà maggiore è stata trovare un compromesso tra queste due problematiche per ottenere le prestazioni migliori e un tempo di risposta più breve possibile.

Per completezza si riportano nella figura 19 gli ID di ciascun tmote installato nella piattaforma, necessari per associare ad ogni nodo àncora posizionato nella mappa il corretto identificativo di ciascun sensore che viene visualizzato al momento della connessione dell'USB nella porta del sistema di elaborazione utilizzato.

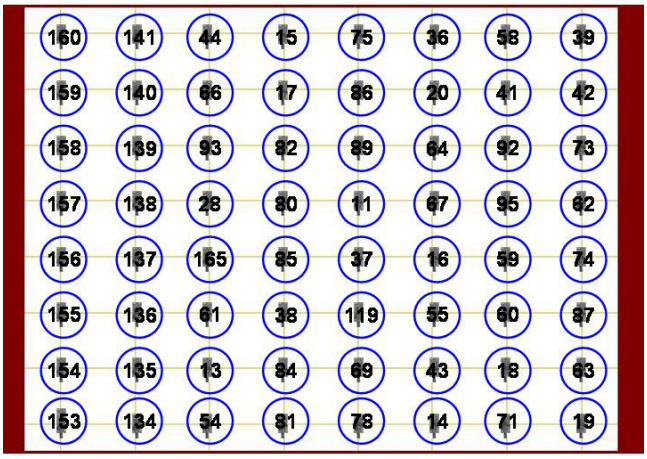


Fig. 19. Distribuzione dei sensori nella mappa e relativo ID di ciascun tmote

F. Prove sperimentali

Infine, per testare i risultati ottenuti da questo studio finora affrontato, si sono effettuate le prove sperimentali installando il tmote, che funge da nodo mobile, sopra l'uniciclo, utilizzato nei test, tramite una basetta montata ad hoc per questa applicazione come mostrato in figura 20.

Per movimentare l'uniciclo si è utilizzata l'interfaccia Epuck-Matlab spiegata precedentemente nella sezione dedicata al software in questo progetto.

1) Come primo test, si è posizionato l'uniciclo in una punto del campo e si è lasciato che Teseo

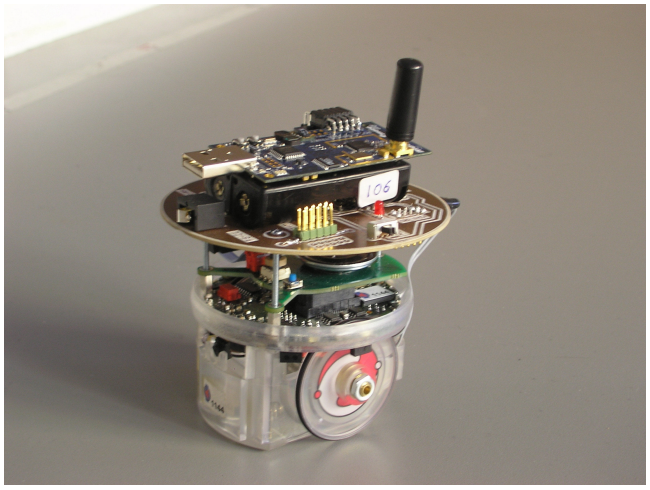


Fig. 20. Fotografia dell'epuck utilizzato per le prove sperimentali

visualizzasse le stime delle posizioni e le salvasse in un file di testo. Questo è stato importato in Matlab e si sono costruiti i plot con i dati ricevuti dal software. I risultati sono riportati in figura 21. Verranno mantenute per tutte le prove le seguenti peculiarità: in *blu* si rappresenta la traiettoria eseguita dall'Epuck, in *rosso* le misure acquisite dall'algoritmo di localizzazione. A causa delle problematiche descritte precedentemente, non è stato possibile ottenere una temporizzazione delle misurazioni ottenute poichè non vengono salvati nel file i valori Nan a volte visualizzati durante l'elaborazione in tempo reale.

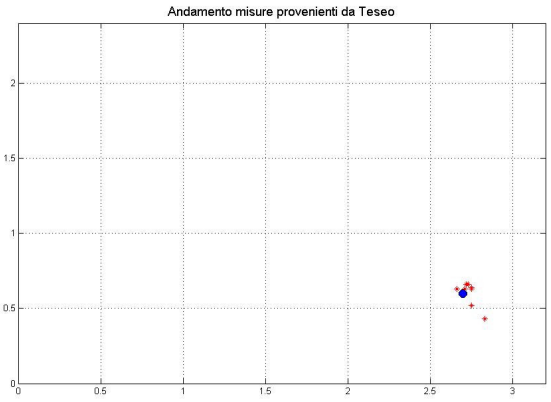


Fig. 21. Prima prova: uniciclo posto in una posizione della piattaforma FERMO

2) Come seconda prova si è guidato l'Epuck facendolo partire da una posizione angolare della piattaforma e attraversandola diagonalmente utilizzando la stessa procedura utilizzata precedentemente. La figura 22 mostra i risultati acquisiti.

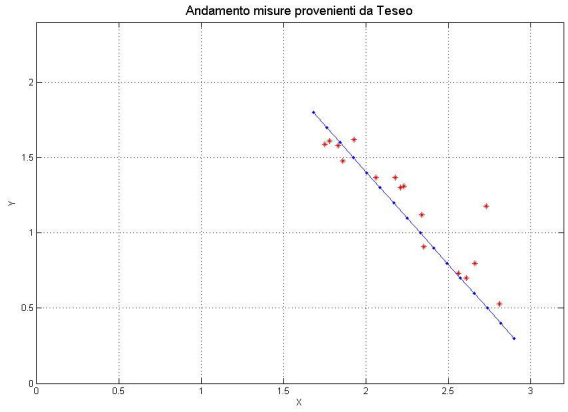


Fig. 22. Seconda prova: Traiettoria lineare

3) Nella terza prova, si è pilotato l'Epuck a eseguire un percorso a U in cui, dopo un primo tratto rettilineo, si è compiuta una semicirconferenza di raggio ampio e si è concluso con un ultimo tratto in linea retta. La localizzazione è stata erroneamente bloccata prima che l'uniciclo arrivasse nel punto di arrivo causando quindi l'assenza di misurazioni nel tratto finale. Si riportano i risultati ottenuti nella figura 23.

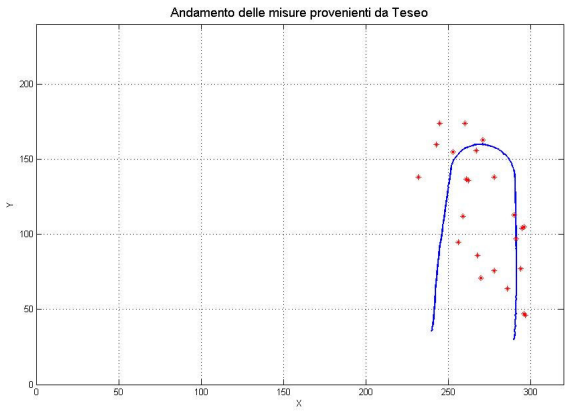


Fig. 23. Terza prova: Traiettoria a U

4) Come ultima prova, si sono eseguiti dei semicerchi consecutivi per testare il comportamento della localizzazione nell'ambito di traiettorie curvilinee mantenendo la stessa procedura di misura e le problematiche delle precedenti prove. Nella figura 24 sono rappresentati i risultati ottenuti in questo test.

Come si osserva facilmente dalle prove effettuate, l'algoritmo di localizzazione stima lo posizione dell'uniciclo in modo molto accurato se si mantiene

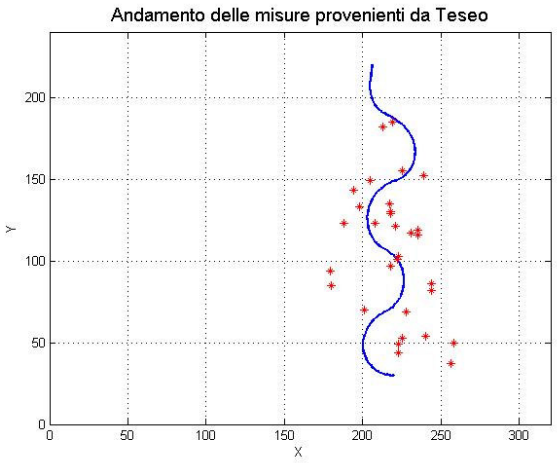


Fig. 24. Quarta prova: Traiettoria curvilinea

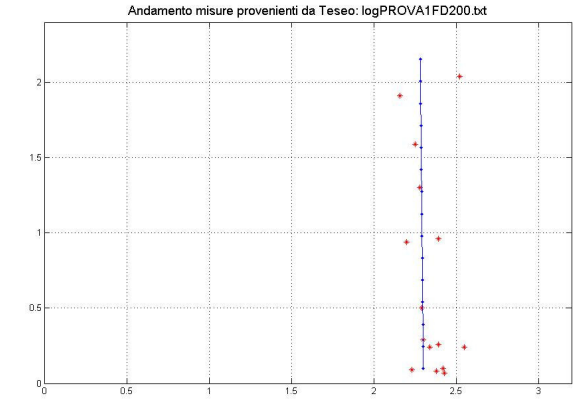


Fig. 25. Traiettoria lineare a velocità di 200 passi al secondo

nella stessa posizione per un arco di tempo abbastanza lungo oppure se le variazioni della traiettoria si effettuano molto lentamente. Si denota come nei percorsi rettilinei l'errore di misura risulta di poche decine di cm e molto minore rispetto ad andamenti curvilinei in cui le stime della posizione peggiorano decisamente. Questa problematica è dovuta al metodo di stima utilizzato dal software di localizzazione per la stima della posizione, poiché è stato pensato per una diversa applicazione basata su percorsi rettilinei e cambiamenti di direzione istantanei e non per la localizzazione di un uniciclo.

1) Prove sperimentali a differenti velocità: Come ultime prove sperimentali si è provato il software di localizzazione su tratti rettilinei ma a differenti velocità; lo scopo di questi test è stato quello di decidere quale fosse le velocità massima con cui poter muovere l'uniciclo affinché si riuscisse ad ottenere una stima della posizione stabile e utilizzabile nelle applicazioni future. Si riportano nelle figure 25, 26 e 27 le prove effettuate in cui in *blu* è rappresentata la traiettoria fatta eseguire dall'Epuck e in *rosso* le misure ottenute con il software di localizzazione.

Dall'analisi delle misure acquisite dalle traiettorie, si può osservare che fino a 600 passi al secondo, cioè il comando dato ai motori passo-passo delle ruote dell'uniciclo, che equivalgono a circa 7.7 [cm/s] (si veda la sezione relativa all'architettura del sistema), si ottengono degli errori di misura accettabili; se a queste misurazioni ottenute si aggiunge successivamente anche una stima tramite un filtro di Kalman Esteso (*EKF - Extended Kalman Filter*) che si è studiato in questo progetto solamente per via simulativa, si potrebbero ulteriormente migliorare in modo da ottenere un algoritmo

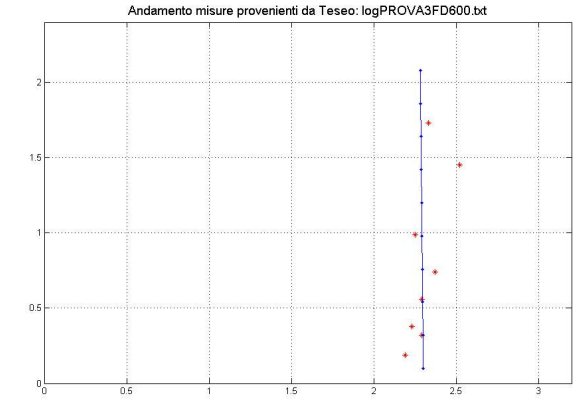


Fig. 26. Traiettoria lineare a velocità di 600 passi al secondo

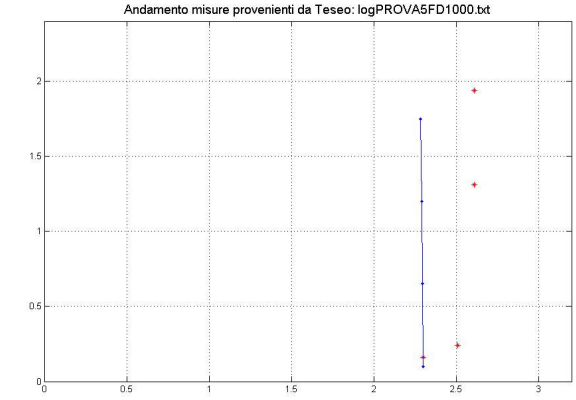


Fig. 27. Traiettoria lineare a velocità di 1000 passi al secondo

di localizzazione robusto e molto accurato. La traiettoria a 1000 passi al secondo è stata riportata come prova per l'analisi effettuata nei capitoli successivi, dato il fatto che il software utilizzato per la generazione della

traiettoria a tempo minimo lavora con tale velocità non modificabile; si osserva quindi facilmente già da ora che non potrà essere provato sperimentalmente poichè le misure provenienti dal Teseo sono troppo rade per ottenere una misura accettabile per la localizzazione neppure se successivamente filtrata dall'EKF.

IV. Filtro Di Kalman Esteso E Motion Planning

Quanto verrà presentato in questa sezione è la descrizione della parte prettamente di controllo del progetto, implementata attraverso l'utilizzo dei software *Matlab* e *Simulink*, necessari in fase di simulazione delle soluzioni proposte.

A. Stato dell'arte

Come già accennato nell'introduzione, l'obiettivo che ci si propone di raggiungere per quanto riguarda lo sviluppo del software di controllo ad alto livello dell'e-puck, è l'implementazione di una tecnica di motion planning punto punto a tempo minimo definibile altresì come tracking ottimo, intendendo con ottimo la minimizzazione del tempo di percorrenza della distanza tra punto di partenza e punto di destinazione. Lo stato dell'arte dal quale partiamo con il nostro sviluppo è costituito dal progetto [6], sviluppato come progetto di tesi nel corso dell'anno accademico $2006/07$. Tale progetto si basa sulla metodologia proposta negli articoli di *Frazzoli*, e considera il problema della pianificazione e del controllo del movimento di un singolo veicolo mobile in ambienti privi di ostacoli. La soluzione proposta in [6] prevede l'impiego di un controllore ibrido, che utilizzi ad anello chiuso l'informazione proveniente dai sensori propriocettivi di bordo e abbia la possibilità di ricevere la comunicazione della propria posizione da un sistema di localizzazione globale (GPS) esterno, eventualmente con frequenza di campionamento bassa e non uniforme, per correggere la propria traiettoria. Le traiettorie generate dal controllore costituiscono una famiglia finita di curve, ma sufficiente a garantire l'ottimalità nel senso della minima lunghezza. La tecnica proposta non prevede alcuna onerosa operazione di minimizzazione numerica, come accade invece per i metodi di controllo ottimo classici, ed è quindi applicabile a sistemi con capacità computazionali limitate come gli e-puck in dotazione. Il metodo, più generale e precisamente formulato per affrontare problemi di pianificazione del movimento, consiste nell'impiego del controllo ottimo. Tuttavia nella maggior parte dei problemi di motion-planning, le tecniche di controllo ottimo soffrono di un elevato carico computazionale, che le rendono inadatte per molte applicazioni in tempo reale, tanto più se i robot mobili hanno capacità computazionali ridotte. L'approccio considerato consiste nella cosiddetta *quantizzazione del controllo*. Anzichè quantizzare il tempo, lo stato del sistema o i valori di ingressi e uscite, si scelgono un numero finito di traiettorie di controllo, dette *primitive di movimento*. Tali primitive sono tra loro combinate per generare traiettorie ammissibili per il sistema. Le regole di concatenazione sono espresse dal linguaggio generato da un automa.
Lo studio del sistema e-puck viene associato allo studio di un particolare sistema: *la macchina di Dubins* poichè ne è un'ottima approssimazione ed essendo largamente studiato in letteratura. Il veicolo è modellato come una regione in movimento rigido sul piano, alla quale è solidale un vettore unitario **v**: l'orientazione. Trattandosi di movimento rigido, il veicolo ha tre gradi di libertà: x, y, θ; le coordinate x e y danno la posizione sul piano del centro di massa della regione e θ denota l'angolo che il vettore di orientazione v forma con l'asse x. Relativamente alle coordinate il vettore v è dato da $\mathbf{v} = (cos\theta, sen\theta)$. Si assume che il veicolo possa muoversi unicamente nel verso di **v**, con velocità $u_1 v$ con u_1 pari a zero o alla velocità massima. In altri termini il veicolo può andare avanti ma non indietro. La motivazione di questa scelta è di natura pratica: la maggior parte dei veicoli terrestri ed acquatici, pur avendo la possibilità di retrocedere, ha un verso di moto preferenziale fissato sia dalla locomozione sia dal campo visivo del conducente. In particolare gli e-puck, che costituiscono il modello fisico di riferimento del progetto, sono dotati di una telecamera fissa, quindi per sviluppi futuri è indicato che l'algoritmo di controllo li faccia avanzare in quella direzione. La limitazione del moto ad un unico verso consente, poi, di semplificare la caratterizzazione della famiglia parametrica di curve contenente le traiettorie di lunghezza minima sul piano. Il veicolo può anche cambiare direzione, si indica a tal fine, con u_2 la velocità angolare: $u_2 = \dot{\theta}$. Lo stato del sistema è dunque la terna (x, y, θ). In questo caso l'equazione generale della dinamica ha la forma:

$$\begin{cases} \dot{x} = u_1 cos(\theta) \\ \dot{y} = u_1 sen(\theta) \\ \dot{\theta} = u_2 \end{cases} \tag{3}$$

Si impone inoltre che il raggio di curvatura sia limitato inferiormente; il vincolo che si impone è quindi:

$$\frac{u_1(t)}{u_2(t)} \geq r_{min} \ \forall t \tag{4}$$

Il modello appena descritto è noto in letteratura come *veicolo di Dubins*, risulta essere un veicolo anolonomo non localmente controllabile in tempo breve, cioè non tutti i punti di un intorno di una configurazione p sono raggiungibili in un tempo non superiore ad un tempo fissato, questo per aver imposto un raggio di curvatura minimo e l'impossibilità di invertire il senso di marcia

se non compiendo una semicirconferenza di raggio minimo. Assegnati un punto iniziale e un punto finale, l'indice di costo che si vuole minimizzare è il tempo t_f richiesto per compiere la traiettoria; quindi l'indice è:

$$J = \int_0^{t_f} dt = t_f \tag{5}$$

Il metodo di pianificazione di traiettorie utilizzato si basa sulla selezione di una famiglia di primitive di movimento, che vengono combinate per formare delle traiettorie (coppie stato e ingresso di controllo), che soddisfino i vincoli imposti dal problema di guida. L'operazione di combinazione di due primitive per formarne una terza viene identificata con il termine *concatenazione*. Al fine di mantenere la validità delle traiettorie, le primitive non possono essere concatenate arbitrariamente, ma si deve accertare che siano verificate delle condizioni di compatibilità. Nello specifico, si deve verificare che lo stato finale della primitiva coincida con lo stato iniziale della seconda. Per mantenere una descrizione finita della libreria di pianificazione, l'algoritmo prende in considerazione una particolare classe parametrica di primitive di movimento, quella dei movimenti in stato stazionario, detti anche *equilibri relativi* o *traiettorie in assetto*. Durante l'esecuzione di tali primitive gli ingressi di controllo sono mantenuti costanti. Per il *veicolo di Dubins* sono previsti i seguenti equilibri relativi:

- I : stato di quiete
- S : moto rettilineo uniforme con velocità massima
- L : moto circolare uniforme in senso antiorario con raggio di curvatura minimo
- R : moto circolare uniforme in senso orario con raggio di curvatura minimo.

Una proprietà rilevante del linguaggio utilizzato per la pianificazione è la possibilità di determinare completamente lo stato del sistema in qualsiasi istante, senza ricorrere all'integrazione numerica delle equazioni differenziali che descrivono il sistema.

Il modello di Dubins prevede che i veicoli si muovano a velocità costante nella direzione del moto. Si è pertanto assunto che i segnali di controllo siano funzioni del tempo continue a tratti, ammettendo cioè che il comando possa avere delle discontinuità di prima specie in corrispondenza dell'istante in cui due successive primitive di movimento vengono concatenate. La banda dei segnali di controllo si è pertanto ipotizzata infinita. questa assunzione è lecita nel caso degli e-puck, in quanto, essendo dotati di due motori passo passo, le ruote sono in grado di muoversi compiendo un assegnato numero di passi e quindi di bloccarsi quasi istantaneamente. Anche l'accelerazione del veicolo fino alla velocità massima richiede un transitorio inferiore ad un passo di campionamento e può quindi pensarsi di durata nulla.

Il cammino minimo per il veicolo esiste ed appartiene ad una famiglia di soli sei cammini-tipo, composti al più da tre parti, ciascuna delle quali può essere o un arco di circonferenza di raggio minimo o un segmento di linea retta, ottenendo un sistema controllabile. Per un veicolo di Dubins si può ottenere dunque la soluzione del problema ed è inoltre possibile garantire che essa coincida con la soluzione ottima. Finora è stato presentato solo l'anello di regolazione più basso. Introducendo ora l'anello di regolazione di alto livello, ci si chiede se esistono leggi di controllo regolari stabilizzanti. Il controllo ottimo per veicoli mobili con raggio di curvatura inferiormente limitato si dimostra essere di tipo *bang-bang*, cioè con escursioni istantanee, in istanti detti di commutazione, dei segnali di comando dal valore massimo a quello minimo. Qualitativamente dunque i controllori per veicoli mobili devono avere una banda molto ampia, all'ottimo infinita. L'algoritmo di regolazione introdotto è in grado di operare con informazione trasmessa da sensori off-board con frequenza di campionamento bassa, in pratica informazione di posizione ed angolazione proveniente da un sistema GPS, il regolatore è inoltre in grado di gestire l'eventualità di assenza di comunicazione producendo le opportune decisioni. Tra due istanti successivi in cui si ha la comunicazione da parte del sistema GPS, il controllo ad alto livello del veicolo non può che operare ad anello aperto. L'idea di base è assimilabile a quella di un controllo *model predictive*. Quando vengono ricevute le misure di posizione ed orientazione, il controllore pianifica la sequenza di comandi per guidare il veicolo fino al punto specificato. Negli istanti in cui non c'è comunicazione da parte del GPS il regolatore esegue l'ultimo piano di movimento calcolato. Una volta che l'informazione sarà nuovamente disponibile, un nuovo piano di movimento viene calcolato e quello precedente risulta sovrascritto. In assenza di alcun tipo di disturbo si ha che la distanza di Dubins[1] dalla configurazione finale decresce monotonamente. Tuttavia questa versione dell'algoritmo è estremamente sensibile ad eventuali disturbi sulla posizione o sull'orientazione del veicolo. Persino i rumori di quantizzazione dovuti alla precisione numerica finita del processore su cui è implementato l'algoritmo possono dar luogo a comportamenti indesiderati, assimilabili a dei cicli limite. La ragione di questo problema risiede nel fatto

[1]La distanza di Dubins tra due configurazioni è definita come la lunghezza del cammino di Dubins che le congiunge.

che, come accennato precedentemente, il veicolo di Dubins non è controllabile in tempo breve. In un intorno del punto finale è quindi sufficiente una piccola deriva, un errore di modellizzazione o di misura per portare il sistema su una traiettoria molto più lunga del necessario. Se la ri-pianificazione del movimento avviene pochissimi istanti prima che venga raggiunto un punto della traiettoria stabilità, c'è il rischio che l'algoritmo faccia eseguire al veicolo un giro completo per correggere un errore possibilmente infinitesimale sulla posizione. Più la frequenza media di ricezione del segnale da GPS è elevata, più è probabile che al termine della manovra correttiva si ripeta lo stesso fenomeno. Una soluzione del problema è scartare il nuovo piano, calcolato all'arrivo di una nuova misura, se questo comporta un cammino molto più lungo della distanza residua di quello precedente. Questo accorgimento consente di eliminare il comportamento anomalo sopra descritto.

L'applicazione di questo algoritmo di pianificazione della traiettoria al caso pratico dell'e-puck viene implementato secondo i seguenti criteri. L'e-puck, fornito di svariati sensori a bordo, non è dotato di odometro poichè il controllo di posizione delle ruote è garantito da due motori a passo. Il controllo a basso livello sulle ruote dunque è già implementato. In questa applicazione si utilizzano gli encoder dei motori a passo come trasduttori propriocettivi della posizione angolare delle ruote. Test effettuati in laboratorio sul sistema reale hanno evidenziato assenza di deriva come parametro di buona qualità del controllore a basso livello. Come sensore esterocettivo si assume un sistema di telecamere fisse, collegate ad un computer per l'elaborazione delle immagini, facente funzione di GPS. Si ipotizza che l'elaboratore ricavi la posizione e l'orientazione del veicolo dalle immagini e comunichi con l'e-puck via radio. Il periodo che intercorre tra due successive comunicazioni è una variabile aleatoria di media $T_{GPS} \gg 1ms$. L'uniciclo si muove su una superficie orizzontale liscia. Le sue due ruote hanno un diametro di 41 mm e sono distanziate di 53 mm circa. Sono mosse da due motori a passo, che compiono 20 passi per giro, dotati di riduttore con rapporto di riduzione 50:1, che porta a 1000 il numero di passi che il motore deve compiere per effettuare un giro completo della ruota. La massima velocità angolare è di 1000 passi per secondo, che corrisponde a un giro al secondo e a una velocità traslazionale di 12.88 cm/sec. Il tempo di campionamento del controllore di basso livello è per quanto detto di 1 ms. L'uniciclo e-puck non è vincolato ad un limite inferiore sul raggio di curvatura, bensì è in grado di ruotare su sè stesso facendo girare le due ruote in versi opposti. Si impone però all'uniciclo di muoversi in accordo con le equazioni del veicolo di Dubins, fissando un unico verso di rotazione per entrambe le ruote. Con questa scelta si ha che una curva di raggio minimo ad esempio a destra, si ottiene facendo girare solamente la ruota sinistra. Il raggio minimo di curvatura ha quindi valore pari al semiasse delle ruote cioè 26.5 mm.

Per le simulazioni al computer sono stati utilizzati i software *Matlab* e *Simulink*. Sono state realizzate due S-function rispettivamente per l'implementazione delle equazioni dinamiche dell'uniciclo e dell'algoritmo di controllo. Le equazioni dell'uniciclo sono state modificate aggiungendo degli ingressi esterni di disturbo, d_1, d_2 e d_3, che provochino la deriva di ciascuno dei tre stati del sistema nonostante nel sistema fisico non sia stata evidenziata. Il codice necessario alla pianificazione della traiettoria di lunghezza minima è stato raccolto in una function esterna, richiamata dalla S-function del controllore.

B. Obiettivi del progetto

Dall'analisi dello stato dell'arte, in sede di discussione degli obiettivi, è risultato evidente che nessun sistema di rilevazione di posizione è in grado di fornire informazioni con grado di precisione assoluto, cioè senza introdurre un grado di incertezza sulle misure. Si è deciso perciò di sviluppare uno studio di fattibilità della tecnica di pianificazione di traiettoria presentata in [6] applicata ad un caso reale, quale la realtà fisica presente nel nostro laboratorio di visione computazionale. Si tratta in sostanza di considerare come fonte di informazione esterocettiva la rete di sensori wireless anzichè il sistema di telecamere menzionato nello stato dell'arte e di analizzare se l'incertezza delle misure fornite consente il buon funzionamento del sistema o se risulta necessario introdurre sistemi di correzione in tempo reale alle rilevazioni, in grado di migliorare le prestazioni della rete e possibilmente della pianificazione. In sostanza, cronologicamente, si prevede l'analisi dei dati forniti dall'algoritmo in [6] con i parametri della rete fisica da noi utilizzata, ricavati nelle sezioni precedenti, ed in caso di funzionamento non sufficientemente preciso, prevedere soluzioni software per il miglioramento delle prestazioni ed ipotizzare possibili sviluppi anche futuri.

C. Soluzioni proposte

Una caratteristica fondamentale del progetto presentato nello stato dell'arte è l'assunzione di

base di una perfetta conoscenza della posizione dell'e-puck proveniente dal sistema di rilevazione considerato per un corretto funzionamento della pianificazione. Come evidenziato nelle sezioni precedenti, la rete wireless da noi implementata non consente di ottenere una informazione di posizione priva di incertezza, anzi, l'accuratezza della rilevazione risulta molto approssimativa rispetto alle ridotte dimensioni della pedana su cui gli e-puck possono muoversi. Si vengono a considerare di conseguenza due obiettivi primari da conseguire in cascata, per analizzare le prestazioni del sistema complessivo. Dato che, come si è visto, l'informazione proveniente dalla rete wireless di localizzazione risulta pessima, è necessario prevedere un sistema software con la funzione di migliorare la qualità di tali misure. Vista inoltre la non linearità del sistema e-puck, dalla dinamica assimilabile a quella di un uniciclo, si applica un filtro di Kalman non lineare, denominato filtro di Kalman esteso. La struttura ricorsiva di tale sistema fa uso della linearizzazione ma in maniera intelligente. Linearizzare in modo ingenuo un modello può significare produrre degli errori di predizione grandi perchè le misure sono descritte troppo approssimativamente.

Il filtro di Kalman esteso *EKF* è un algoritmo di impiego molto generale che risolve in modo approssimato il problema di stima dello stato di un modello non lineare. Dato che è praticamente impossibile risolvere esattamente in termini di algoritmi a memoria finita un problema di filtraggio non lineare, la tecnica che inevitabilmente si deve utilizzare per derivare anche le equazioni dell'EKF è quella della linearizzazione. Si tratta però di un procedimento intelligente. Il principio fondamentale su cui è basato l'EKF è la linearizzazione ad ogni istante di campionamento delle equazioni della dinamica attorno alla miglior stima del vettore di stato disponibile in quel momento. Questo principio permette, in linea generale, di mantenere gli errori di linearizzazione ad un livello accettabile, ed è la chiave per spiegare il buon comportamento di questo algoritmo in molti casi pratici.

In un secondo momento risulta necessario integrare il sistema di filtraggio delle misure con il software di pianificazione e analizzare l'interazione dei due componenti nonchè apportare le eventuali modifiche alla struttura complessiva.

Innanzitutto per poter creare un filtro di Kalman è necessario disporre di un modello matematico del sistema fisico che si vuole controllare. Nel nostro caso, volendo predisporre un controllore discreto esatto, si orienta la modellizzazione all'ottenimento di un modello discreto del sistema fisico, su cui costruire su misura un controllore anch'esso discreto, evitando in tal modo errori provenienti da discretizzazioni di sistemi creati direttamente nel continuo.

1) Creazione di un modello matematico discreto del sistema fisico: per quanto riguarda il nostro caso specifico di uniciclo, si scelgono i seguenti parametri per l'identificazione di un modello matematico: $q = [x\ y\ \theta]^T$ e $u = [v\ \omega]^T$. Il modello cinematico stocastico non lineare dal quale parte l'analisi è:

$$\begin{cases} \dot{x} = f(x, u, N) \\ y = h(x, W) \end{cases} \tag{6}$$

dove N e W si ipotizzano essere dei rumori bianchi gaussiani ed indipendenti descritti da:

$$\begin{array}{l} N \sim N(0, Q) \\ W \sim N(0, R) \end{array} \tag{7}$$

per considerare eventuali imprecisioni di modellizzazione e ed errori di misura. Nella nostra applicazione questi parametri sono considerati costanti nel tempo, dato che le condizioni fisiche del sistema sono caratteristiche invarianti alla posizione. Il sistema matematico continuo, quindi, caratterizzato anche dall'aleatorietà introdotta dai rumori di misura e modello risulta:

$$\begin{cases} \dot{x} = v \cdot cos(\theta) + N_x \\ \dot{y} = v \cdot sen(\theta) + N_y \\ \dot{\theta} = \omega + N_\theta \\ y = \begin{bmatrix} x + W_x \\ y + W_y \end{bmatrix} \end{cases} \tag{8}$$

Per integrazione diretta delle equazioni, in modo approssimato per la posizione ed in modo esatto per l'orientamento, applicando il metodo di discretizzazione di *Runge-Kutta del 2° ordine* [25] si ottiene il sistema discreto:

$$\begin{cases} x(k+1) = x(k) + v(k)\ T_s\ cos(\theta(k) + \frac{\omega(k) \cdot T_s}{2}) + N_x \\ y(k+1) = y(k) + v(k)\ T_s\ sen(\theta(k) + \frac{\omega(k) \cdot T_s}{2}) + N_y \\ \theta(k+1) = \theta(k) + \omega(k)\ T_s + N_\theta \\ y(k) = \begin{bmatrix} x(k) + W_x \\ y(k) + W_y \end{bmatrix} \end{cases} \tag{9}$$

sul quale si fonda la costruzione del filtro di Kalman. Alle grandezze di velocità, si sostituiscono le letture degli encoder della ruota destra Δ_{ψ_R} e ruota sinistra Δ_{ψ_L}, che corrispondono agli incrementi angolari delle ruote dal tempo iniziale.

$$\begin{array}{rcl} v(k) \cdot T_s & = & r \cdot \frac{\Delta_{\psi_R} + \Delta_{\psi_L}}{2} \\ \omega(k) \cdot T_s & = & r \cdot \frac{\Delta_{\psi_R} - \Delta_{\psi_L}}{2d} \end{array} \tag{10}$$

dove r è il raggio delle ruote e d è la lunghezza del semiasse tra le ruote.

2) Filtro di Kalman Esteso: per l'implementazione dell'EKF si sono seguiti i testi [11] e [12]. Come riportato nei testi sopra citati il filtro di Kalman esteso è un algoritmo di impiego molto generale che risolve in modo approssimato il problema di stima dello stato di un modello non lineare. La tecnica che inevitabilmente si deve usare per derivare le equazioni dell'EKF è quella della linearizzazione. Il principio fondamentale su cui è basato questo filtro è di linearizzare ad ogni istante di campionamento le equazioni della dinamica attorno alla miglior stima del vettore di stato disponibile al momento. Le ipotesi di base per applicare questa metodologia sono: la presenza di rumori di modello bianco a media nulla con matrice semidefinita positiva; rumore di misura bianco con matrice strettamente definita positiva. Si suppongono inoltre i processi di rumore di modello e misura scorrelati fra loro e dallo stato iniziale del sistema. Per costruzione il filtro risulta tempo variante, quindi ad ogni passo di campionamento risulta necessario il calcolo ricorsivo dei coefficienti e delle matrici di linearizzazione. L'algoritmo, applicato al generico sistema non lineare:

$$\begin{cases} x_{k+1} = f_k(x_k) + H_k(x_k)\xi_k \\ v_k = g_k(x_k) + \eta_k \end{cases} \tag{11}$$

con *x* vettore di stato e *v* vettore delle uscite, è riportato di seguito per completezza:

- condizioni iniziali

$$\begin{aligned} P_{0|0} &= Var(x_0) \\ \hat{x}_0 &= E(x_0) \end{aligned} \tag{12}$$

- passo di predizione

$$\begin{aligned} P_{k|k-1} &= \left[\frac{\partial f_{k-1}}{\partial x_{k-1}}(\hat{x}_{k-1})\right] P_{k-1|k-1} \left[\frac{\partial f_{k-1}}{\partial x_{k-1}}(\hat{x}_{k-1})\right]^T + \\ &\quad + H_{k-1}(\hat{x}_{k-1}) Q_{k-1} H_{k-1}^T(\hat{x}_{k-1}) \\ \hat{x}_{k|k-1} &= f_{k-1}(\hat{x}_{k-1}) \end{aligned} \tag{13}$$

- guadagno del filtro

$$\begin{aligned} G_k = P_{k|k-1} &\left[\frac{\partial g_k}{\partial x_k}(\hat{x}_{k|k-1})\right]^T \cdot \\ &\left[\left[\frac{\partial g_k}{\partial x_k}(\hat{x}_{k|k-1})\right] P_{k|k-1} \left[\frac{\partial g_k}{\partial x_k}(\hat{x}_{k|k-1})\right]^T + R\right]^{-1} \end{aligned} \tag{14}$$

- passo di aggiornamento

$$\begin{aligned} P_{k|k} &= \left[I - G_k \left[\frac{\partial g_k}{\partial x_k}(\hat{x}_{k|k-1})\right]\right] P_{k|k-1} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + G_k \left(v_k - g_k(\hat{x}_{k|k-1})\right) \end{aligned} \tag{15}$$

dove per il sistema lineare:

$$\begin{cases} x_{k+1} = A_k x_k + u_k + \Gamma_k \xi_k \\ w_k = C_k x_k + \eta_k \end{cases} \tag{16}$$

le variabili A_k, u_k, w_k, Γ_k e C_k sono determinate in tempo reale attraverso approssimazione lineare di Taylor delle funzioni di stato e di uscita non lineari, cioè:

$$\begin{cases} f_k(x_k) \simeq f_k(\hat{x}_k) + A_k(x_k - \hat{x}_k) \\ g_k(x_k) \simeq g_k(\hat{x}_{k|k-1}) + C_k(x_k - \hat{x}_{k|k-1}) \end{cases} \tag{17}$$

dove:

$$A_k = \left[\frac{\partial f_k}{\partial x_k}(\hat{x}_k)\right] \qquad C_k = \left[\frac{\partial g_k}{\partial x_k}(\hat{x}_{k|k-1})\right] \tag{18}$$

inoltre:

$$\begin{cases} u_k = f_k(\hat{x_k} - A_k \hat{x}_k) \\ \Gamma_k = H_k(\hat{x}_k) \\ w_k = v_k - g_k(\hat{x}_{k|k-1}) + C_k \hat{x}_{k|k-1}. \end{cases} \tag{19}$$

3) Realizzazione dell'EKF: il sistema in variabili di stato, su cui sono stati applicati i passaggi per il calcolo del filtro di Kalman nel caso specifico, è il seguente:

$$\begin{cases} \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \\ \qquad + \begin{bmatrix} T_s cos(\theta_k + \frac{\omega_k T_s}{2}) v_k \\ T_s sen(\theta_k + \frac{\omega_k T_s}{2}) v_k \\ T_s \omega_k \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_\theta \end{bmatrix} \\ \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \\ v_\theta \end{bmatrix} \end{cases} \tag{20}$$

Le matrici che si ottengono per linearizzazione sono:

$$A_k = \begin{bmatrix} 1 & 0 & -v_k T_s sen\left(\hat{\theta_k} + \frac{\omega_k T_s}{2}\right) \\ 0 & 1 & v_k T_s cos\left(\hat{\theta_k} + \frac{\omega_k T_s}{2}\right) \\ 0 & 0 & 1 \end{bmatrix}$$

$$C_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \Gamma_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Le matrici dei rumori di misura e di modello, entrambe supposte diagonali, vengono tarate utilizzando valori provenienti da prove sperimentali e da simulazioni successive del filtro su diverse tipologie di traiettorie. La matrice di rumore di misura viene ricostruita da dati medi ricavati da numerose rilevazioni sulla rete WSN. La prestazione media della rete ha portato a riscontrare un errore di rilevazione contenuto in un cerchio di raggio $25cm$ per le posizioni sugli assi x e y, mentre si è supposto di conoscere perfettamente l'orientazione del veicolo, poichè il calcolo di tale grandezza è ottenuto per integrazione esatta delle equazioni dinamiche e dato che l'elevata imprecisione dell' informazione impedisce una stima dell'orientazione tra due rilevazioni successive senza portare a incertezze accettabili. Poichè inoltre tale

matrice deve risultare definita positiva per questo ultimo parametro si sceglie un valore irrisorio ma pur sempre maggiore di zero.
Per quanto riguarda, invece, il settaggio della matrice di rumore di modello, la scelta dei parametri avviene per via sperimentale su simulazioni consecutive e su campioni di traiettorie diverse. Da simulazioni effettuate in fase di taratura del filtro si sono scelti valori bassi per i rumori di modello dato che le misure risultano poco affidabili, mentre il modello matematico ricavato ha un grado di precisione sufficientemente accurato. Con i valori prescelti, in sostanza, il filtro considera solo marginalmente le misure dando maggior peso alla dinamica prodotta dal modello; questo permette in linea di principio di sopperire, almeno in parte, all'abbondante imprecisione delle rilevazioni.
Lo studio delle prestazioni del filtro viene illustrato nella sezione *Risultati raggiunti* dove si riportano le considerazioni fatte in sede di progetto ed i risultati ottenuti applicando quanto trovato in diversi casi sperimentali al fine di ottenere una sorta di analisi delle casistiche.

4) Integrazione EKF-Motion planning: una volta creato il filtro di Kalman esteso, lo si è integrato con il software di pianificazione di traiettoria pre-esistente per poterne verificare gli effetti nella realizzazione di un obiettivo pratico quale la pianificazione di traiettoria punto punto a tempo minimo di percorrenza. Data la struttura software dei file in [6], è stato di fondamentale importanza analizzare attentamente il punto in cui inserire l'elaborazione alla Kalman delle misure per poter rispettare le temporizzazioni di tutta la struttura e non andare così a pregiudicare i risultati. Avendo utilizzato *S-function* per implementare la teoria di pianificazione, si è deciso di mantenere tale struttura simulativa ed inserire nella *S-functioncontroller-e-puck* lo sviluppo dell'*EKF*. Si sono incontrate notevoli difficoltà nella gestione di tutte le temporizzazioni dei segnali, risolte aumentando a dovere le dimensioni del vettore di stato e dei vettori delle uscite e degli ingressi delle funzione considerata.

D. Risultati raggiunti

Per la complessità del sistema in esame, in sede di sviluppo del progetto, si è deciso di perseguire degli obiettivi in cascata per riuscire a comprendere meglio fino a che punto è possibile migliorare le prestazioni di ogni singolo componente software. Per tale motivo si è innanzitutto sviluppato un filtro di Kalman esteso con ingressi indipendente dalla pianificazione, con l'intento di valutarne le prestazioni su diverse casistiche di traiettoria. In un secondo momento si sono uniti i sistemi *EKF* e *Pianificazione di traiettoria* per poter analizzare le prestazioni del sistema complessivo.

1) EKF messo in pratica: in questa sezione si analizzeranno i risultati ottenuti nella simulazione del filtro di Kalman applicato a traiettorie prefissate, scelte per valutare le risposte del filtro in differenti occasioni applicative. Per rendere l'analisi del tutto svincolata dal nostro caso pratico si è scelto di parametrizzare la lunghezza dei percorsi simulati alla distanza percorsa in un giro di ruota dell'e-puck in moto rettilineo. Si sono assunte come ipotesi iniziali l'avere delle misure di posizione provenienti dalla rete di rilevazione con incertezza di $\pm 25cm$ dalla posizione reale e di conoscere esattamente l'orientazione dell'uniciclo. In particolar modo se non si pone questa ultima ipotesi, il filtro non riesce a recuperare l'informazione in termini di orientazione e propone un risultato insoddisfacente anche con traiettorie di lunghezza elevata dove il filtro può raggiungere la condizione di regime. La suddetta ipotesi è resa accettabile se si posiziona l'e-puck sul piano in posizione casuale ma con orientazione nota a priori. Inoltre, poichè il sistema di rilevazione non è in grado di fornire indicazioni di orientazione, se non per interpolazione tra misure consecutive con mancanza assoluta di precisione, vista la varianza di errore elevata delle misure stesse, si lascia tale informazione come conosciuta a priori. Grazie al fatto che la dinamica dell'angolo di orientazione è calcolata per integrazione esatta e sul sistema reale si è riscontrata assenza di deriva nelle rotazioni, si ha una perfetta conoscenza della direzione assunta in ogni istante di simulazione.
Un'ulteriore ipotesi della quale si è dovuto tener conto in fase di progetto, è stata la frequenza di alimentazione dei motori a passo dell'e-puck. Il software di elaborazione del filtro è stato creato con periodo di campionamento pari a un millesimo di secondo, per rispettare il periodo scelto nel software di pianificazione di traiettoria e creare così due sistemi che potessero lavorare in simbiosi. In fase di prova pratica tale frequenza di alimentazione dei motori dell'e-puck è risultata troppo elevata. Si è deciso di conseguenza di discretizzare le funzioni di alimentazione dei motori con un periodo più elevato; per tale parametro si è identificato come bound inferiore un valore di almeno cinque millesimi di secondo, ma si è scelto di lavorare a dieci millisecondi per valutare un caso leggermente peggiore della condizione migliore possibile, avendo però la certezza di non incappare in problemi di natura fisica che possono occorrere se il sistema è chiamato a lavorare nelle sue condizioni limite. In sostanza gli ingressi da fornire ai motori vengono

calcolati ogni millisecondo mentre gli ingressi effettivamente considerati sono funzioni continue a tratti dove i valori sono scelti campionando e trattenendo i valori delle funzioni originarie. Il periodo di acquisizione delle misure è fissato a un secondo e tre decimi, dato che è stato ricavato in fase di test della pedana in laboratorio. Anche in questo caso si è fatta un'ipotesi semplificativa, si è stabilito infatti che il tempo di interarrivo delle misure non fosse una variabile aleatoria descritta da media e varianza ma fosse deterministica e costante. Questo permette il calcolo degli ingressi da fornire al filtro di Kalman all'arrivo di ogni nuova misura, senza incorrere in problemi di calcolo corrispondenti all'aleatorietà dell'intervallo, in cui occorre sommare il numero di passi effettuati dall'e-puck. Infatti se si introducesse aleatorietà in tale intervallo temporale, nel caso in cui l'intervallo non risultasse esattamente multiplo del periodo di discretizzazione del comando ai motori, la somma del segnale a intervalli non costanti sarebbe affetta da un errore pari al numero di passi compiuti in un periodo di discretizzazione, introducendo così delle imprecisioni che in prima analisi non si vorrebbero considerare per consolidare i risultati su di una base il più semplice possibile, base che potrebbe essere arricchita in un secondo momento.
Si procede ora all'analisi dei risultati ottenuti in simulazione grazie al software appena presentato. Il set di curve considerato per estrarre le considerazioni è il medesimo previsto dal software di pianificazione di traiettoria, cioè la concatenazione di tratti rettilinei e curve di raggio minimo. La prima rilevazione della rete di misura viene considerata come punto di inizializzazione del filtro e la varianza di errore di misura viene associata alla varianza P_0 del filtro.
Per tutte le simulazioni che seguiranno il tratto nero costituisce la traiettoria realmente percorsa dell'e-puck, i triangolo blu sono le rilevazioni effettuate dalla rete WSN con il rispettivo errore di misura (cerchio azzurro), mentre i triangoli rossi sono le stime trovate applicando, in tempo reale alle misure, il filtro di Kalman esteso, con il relativo errore di predizione (cerchio viola).
Le casistiche selezionate per l'analisi delle prestazioni del filtro sono:

- Tratto rettilineo percorso a velocità massima.
  Come si nota dall'immagine 28, il filtro espleta in modo soddisfacente la sua funzione di riduzione della varianza di errore e la stima di posizione migliora costantemente ad ogni rilevazione di posizione fino al raggiungimento della condizione di regime. L'applicazione del filtro in questo caso permette di far scendere l'incertezza nel piano per le coordinate x e y da $25cm$ a soli $9.6cm$ raggiungendo la condizione di regime dopo una decina di rilevazioni. Come ci si poteva aspettare su tratti rettilinei con l'ipotesi di conoscere perfettamente l'orientazione del veicolo il filtro di Kalman riesce ad inseguire quanto voluto. Per realizzare l'immagine si sono ipotizzati inoltre un periodo di rilevazione pari a $1.3sec$ ed una lunghezza del tracciato pari a quindici giri completi di ruota con una discretizzazione delle variabili di comando ai motori al decimo di secondo.

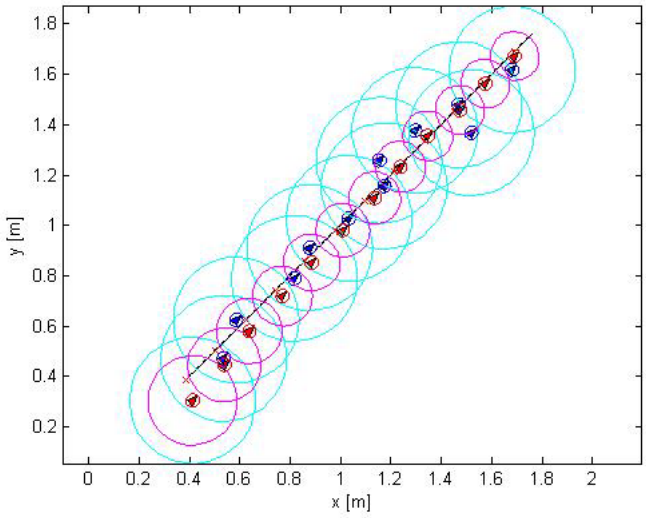


Fig. 28. Traiettoria rettilinea

- Curva a raggio minimo - tratto rettilineo - curva a raggio minimo
  In questo caso si sono voluti mettere in evidenza alcuni aspetti legati alla stima di percorsi contenenti curve di raggio minimo come previsto dalla pianificazione di traiettoria presentata in [6]. La prima curva prevista, ovviamente, non viene percorsa correttamente poichè il filtro di Kalman non è ancora arrivato a regime, quindi per le prime rilevazioni provenienti dalla rete, essendo molto incerte, non vi è un miglioramento efficace durante la stima. Dopo la curva viene fatto percorrere un lungo tratto rettilineo per permettere al filtro di arrivare a regime e all'e-puck di avvicinare sufficientemente la traiettoria corretta. A questo punto viene fatta compiere un'altra curva di raggio minimo e la risposta che si ottiene dal filtro evidenzia i primi problemi. Infatti, sebbene il filtro stia già lavorando a regime, non riesce a far percorrere correttamente all'e-puck la traiettoria

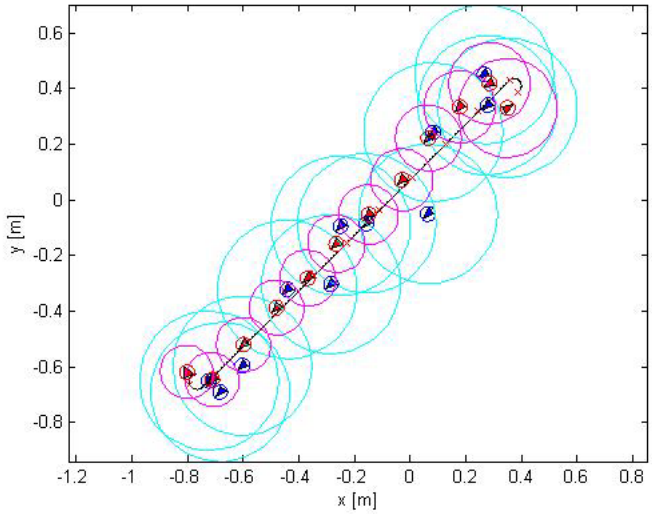


Fig. 29. Curve di raggio minimo e tratto rettilineo

prefissata. Con l'immagine 30 si vuole evidenziare proprio quest'ultimo comportamento.

La prima parte di tracciato è la stessa

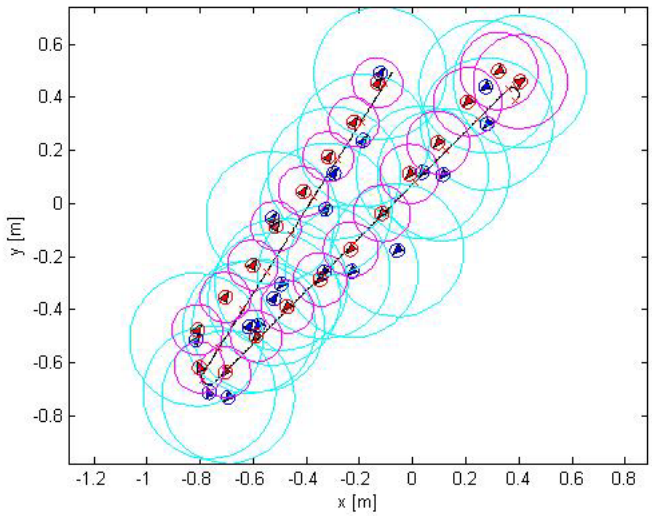


Fig. 30.

dell'immagine 29 ma dopo l'ultima curva è stato aggiunto un lungo percorso rettilineo per osservare la reazione del filtro. Si nota che la stima rientra lentamente in traiettoria, sovraelongando, se così si può dire, verso l'esterno della curva come se fosse affetta da una sorta di forza centrifuga, e solo dopo parecchie ulteriori acquisizioni raggiunge nuovamente la traiettoria effettivamente percorsa.
Quanto finora riportato è indicativo del funzionamento del filtro, una volta raggiunte le condizioni di regime il filtro risponde in maniera ottimale se l'e-puck deve percorrere traiettorie rettilinee mentre se è chiamato a compiere rotazioni di raggio minimo non reagisce in tempo breve e riesce a recuperare gli errori di stima solo sfruttando ulteriori tratti rettilinei successivi alla curva.

- Sequenza di curve a raggio minimo
  Questo caso è stato riportato per evidenziare

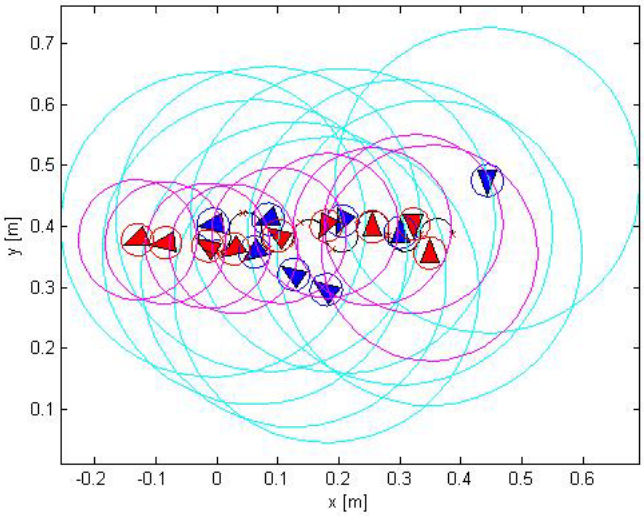


Fig. 31. Concatenazione di curve a raggio minimo

come il filtro non riesca a stimare correttamente la traiettoria nel caso l'e-puck sia chiamato a ruotare con aperture di raggio minimo in concatenazione, come potrebbe accadere in prossimità del raggiungimento dell'obiettivo. Come ben si vede dall'immagine 31 per effetto della grande incertezza delle misure la stima del filtro non risulta per niente corrispondente alla traiettoria effettivamente percorsa dal robot. Questo può essere l'effetto della quantizzazione del comando ai motori data la rapidità di rotazione richiesta in questo caso, ma sicuramente riveste un peso notevole l'elevata varianza d'errore delle misure.

- Cerchio di raggio non minimo
  Per valutare appieno le prestazioni del filtro si è deciso di riportare anche uno studio di traiettorie con curve di raggio non minimo, per poter sviluppare considerazioni sull'interazione tra filtraggio alla Kalman e pianificazione a tempo minimo. Infatti nell'immagine 32 si riporta il funzionamento del filtro su un cerchio di raggio non minimo sempre percorso alla massima velocità.

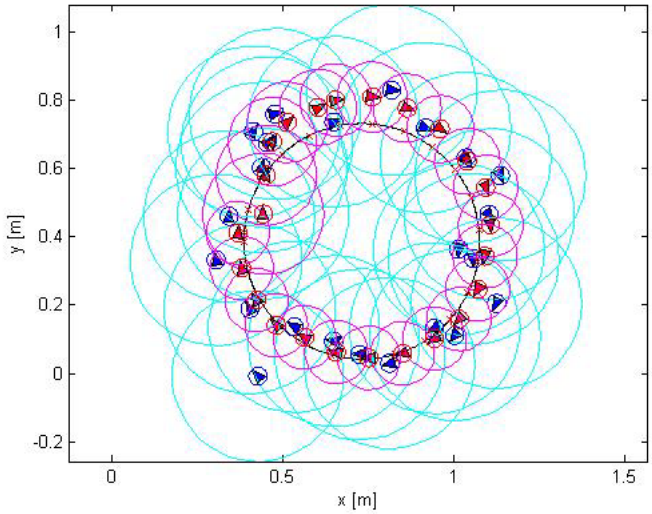


Fig. 32. Traiettoria circolare

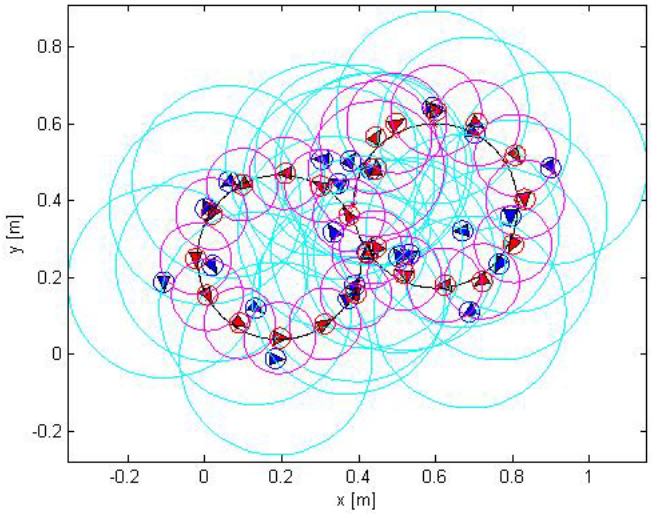


Fig. 33. Traiettoria a forma di otto

Si nota da questo, come ci si aspettava, che passate le prime misure incerte, al raggiungimento delle condizioni di regime del filtro, la stima insegue bene la traiettoria reale anche in presenza di misure che si discostano di molto da essa. Questo può essere spiegato con il fatto che, nel compiere una curva di raggio ampio, intervengono molte misure nella descrizione della traiettoria, mentre questo non accade per curve di raggio minimo dove le misure che possono arrivare durante la loro percorrenza sono al massimo due, visto che il robot sta viaggiando alla sua massima velocità. Un'ipotesi che si può fin d'ora avanzare, è che i risultati ottenibili dal filtro potrebbero essere molto migliori anche nel compiere curve di raggio minimo se si riducesse la velocità di movimento dell'e-puck. In questo progetto questa ipotesi non verrà trattata in quanto la pianificazione di traiettoria ottima, presentata nella sezione seguente, prevede che l'indice da ottimizzare sia proprio il tempo di percorrenza di tale traiettoria, perciò bisognerebbe assumere come condizione iniziale una velocità ridotta come massimale accettabile dal sistema.

- Curva a 8 a raggio non minimo
  Anche in questo caso, come in figura 33, cambi di direzione in raggi di rotazione di valore elevato non danno particolari problemi al filtro che è in grado di assecondare la traiettoria realmente percorsa.

- Curva a raggio non minimo - tratto rettilineo - curva a raggio minimo - tratto rettilineo
  In questo ultimo caso si sono volute mettere

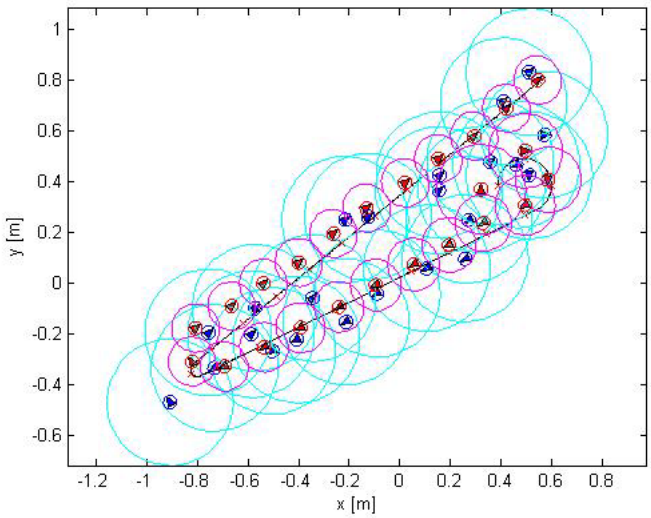


Fig. 34. Concatenazione di curve di raggio diverso

a confronto le due tipologie di curve finora analizzate, il tracciato prevede una curva di raggio ampio, all'inizio, e una curva di raggio minimo, separate da un tratto rettilineo. Come si può vedere dall'immagine 34, la curva iniziale, sebbene la varianza d'errore del filtro sia ancora elevata, viene seguita abbastanza bene nonostante le misure risultino poco performanti. Nel compiere, invece, la curva di raggio minimo, le prestazioni del filtro, nonostante abbia già ormai raggiunto le condizioni di regime, si discostano abbondantemente dalla

traiettoria reale, per poi recuperare in precisione nel tratto rettilineo che segue. Quanto appena evidenziato ci porta a pensare che le scarse prestazioni del filtro nell'inseguire curve di raggio minimo siano causate dall'alta velocità di percorrenza della traiettoria stessa in concomitanza alla bassa frequenza di arrivo delle misure del sistema di rilevazione. Si sono analizzati in simulazione le incidenze di tali fattori sulla risposta del filtro. Il risultato è stato quello di riscontrare una loro inscindibilità come causa di una stima difettosa durante la percorrenza di curve di raggio minimo.

Si riportano, come esempio, in 35 e 36, i

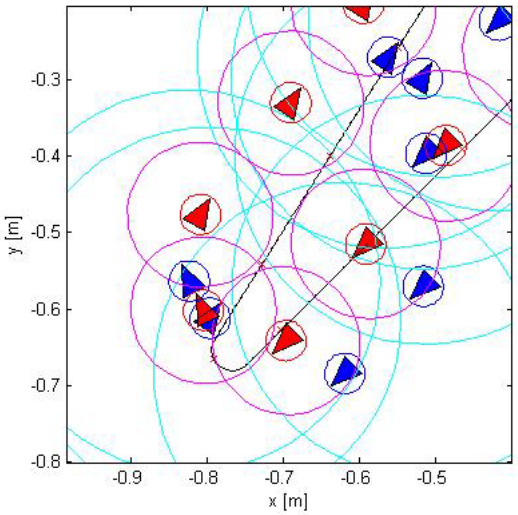


Fig. 35. Curva a raggio minimo con T_{acq} = 1.3 sec

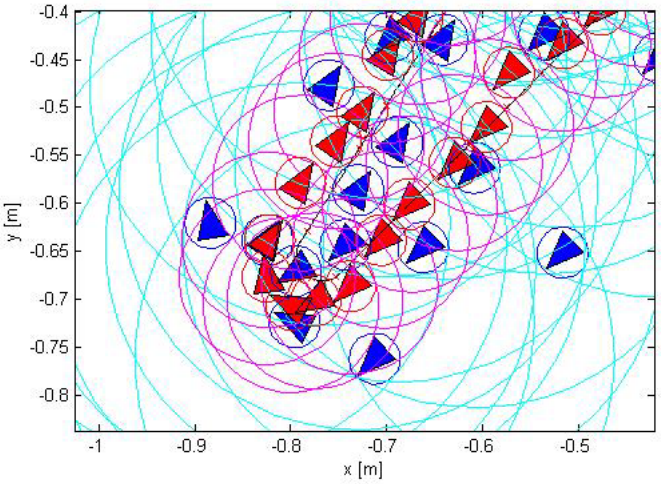


Fig. 36. Curva a raggio minimo con T_{acq} = 0.4 sec

particolari della stessa curva a raggio minimo percorsa a velocità massima ma con tempo di interarrivo delle misure fissati rispettivamente a $1.3sec$ e $0.4sec$, con filtro già in condizioni di regime. Si nota come la curva venga percorsa in maniera molto più precisa nel secondo caso.

Per l'obiettivo che ci siamo posti, velocità massima, tempo di interarrivo delle misure e curve di raggio minimo, sono tre caratteristiche peculiari delle ipotesi per ottenere ottimizzazione delle traiettorie secondo un indice di costo temporale. Si sono verificate prestazioni molto migliori all'abbassarsi del tempo di interarrivo delle misure, ma imporre questo parametro ad un valore inferiore a quello utilizzato nelle simulazioni comporterebbe dover utilizzare una rete diversa da quella effettivamente a nostra disposizione in laboratorio che offre come bound inferiore proprio il valore da noi scelto. Anche le caratteristiche geometriche e cinematiche delle traiettorie non possono venir meno senza compromettere l'ottimalità temporale del tracciato. Si è ipotizzato anche di far eseguire rotazioni sul posto all'e-puck anzichè fargli compiere curve di raggio minimo, ma tale ipotesi contravverebbe all'ipotesi iniziale di imporre un senso di marcia preferenziale e impedire il doppio senso di rotazione alle ruote.

Riassumendo quindi, il filtraggio alla Kalman esteso permette di ridurre l'imprecisione delle misure provenienti dalla rete di rilevazione wireless, ma non in tutte le situazioni che si possono presentare in sede di pianificazione di traiettoria. Innanzitutto l'insieme di ipotesi fatte in sede di progetto costituiscono una scelta troppo restrittiva per la funzionalità del filtro nell'inseguimento della traiettoria quando vengono previste curve a raggio minimo, in maggior modo se queste curve risultano concatenate in sequenza e non sono previsti lunghi tratti rettilinei in cui il filtro riesce a recuperare in prestazione. Ovviamente con questo studio non si riescono a coprire tutte le evenienze, occorre perciò, applicare il filtraggio alla pianificazione di traiettoria e verificarne l'effettiva funzionalità.

2) Pianificazione di traiettoria con EKF: in questa sezione si andranno ad analizzare gli effettivi comportamenti del software di pianificazione creato in [6] filtrato in tempo reale attraverso l'EKF fino ad ora analizzato.

Si premette che i risultati ottenuti non sono robusti, molto dipende dalla posizione che si impone di raggiungere e dal punto da cui si vuole partire. Innanzitutto

è risultato necessario modificare alcune caratteristiche del software in [6], in quanto impedivano l'esecuzione corretta dell'implementazione contenente anche il filtro di Kalman. In particolar modo questo problema è riferito alla parte di software che analizza la lunghezza della ri-pianificazione all'arrivo di una nuova misura. In [6], tale accorgimento permetteva di scartare eventuali piani di movimento troppo lunghi rispetto al piano precedentemente formulato, evitando così inconvenienti come anelli circolari o il ripetersi di traiettorie attorno al punto di arrivo, il tutto scelto attraverso un parametro. Nel caso in esame, invece, tale struttura impediva, nel caso la misura fosse troppo lontana dalla traiettoria reale, di prendere in considerazione il filtraggio di kalman e la ri-pianificazione, mantenendo sempre la prima pianificazione come ottima non potendo giovare così delle misure lungo il percorso e soprattutto dell'azione filtrante del filtro introdotto. Per questo motivo è stato aumentato tale parametro, rendendo così possibile la pianificazione anche in caso di arrivo di misure molto imprecise. La conseguenza di questa scelta si è rivelata subito, però, nelle simulazioni effettuate per testare la funzionalità dell'insieme. Avendo aumentato tale parametro, se una nuova misura giunge al software poco prima di giungere in prossimità del punto scelto come obiettivo, possono accadere dinamiche di movimento che allontanano l'e-puck dal punto di destinazione e lo portano a compiere lunghissimi percorsi prima di tornare a puntare verso l'obiettivo. Per tentare di arginare questo problema, almeno in parte, è stato aggiunto via *Simulink* un congegno software che rilassa le coordinate dell'obiettivo da raggiungere. Più in dettaglio con tale stratagemma si considera raggiunto l'obiettivo con un grado di precisione scelto a priori, questo permette di evitare ri-pianificazioni inutili quando l'e-puck si trova in prossimità del punto stabilito, il grado di precisione si fissa a priori attraverso parametri, nelle seguenti simulazioni si accetta un errore di posizionamento di raggio $2,5cm$.

Si riportano nelle pagine a seguire alcune prove di pianificazione, ogni prima immagine riporta la pianificazione in assenza di rumore di misura, le seconde immagini riportano le pianificazioni in presenza di rumore di misura con i parametri trovati per la rete di rilevazione WSN a disposizione in laboratorio, mentre le ultime riportano gli effetti della pianificazione con l'azione dell'EKF presentato nelle sezioni precedenti nelle medesime configurazioni.

Si premette che dall'analisi dei casi seguenti non si riesce a ricavare un risultato complessivo del problema, in quanto le prestazioni sono fortemente influenzate dalla lunghezza del percorso e dall'obiettivo finale scelto. Perciò si andranno ad analizzare le singole prestazioni tentando di identificare un trend di risultati, cercando di trovare delle regole di funzionamento globali del sistema complessivo.

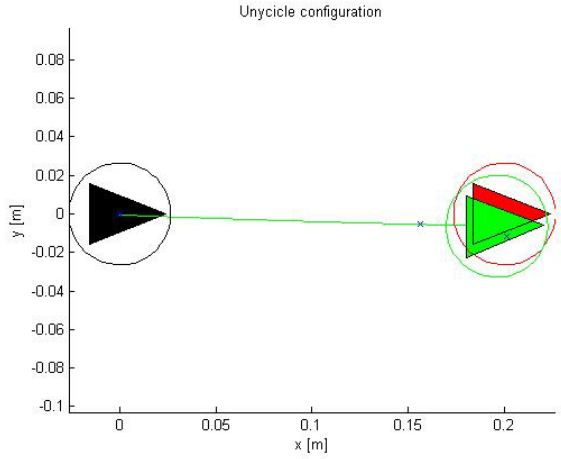


Fig. 37. Senza EKF e senza rumore di misura

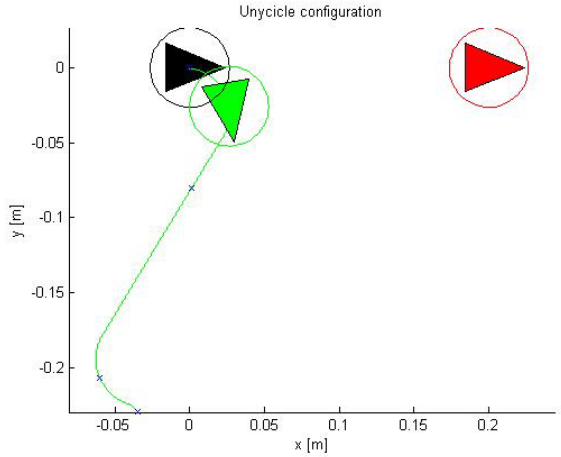


Fig. 38. Senza EKF con rumore di misura

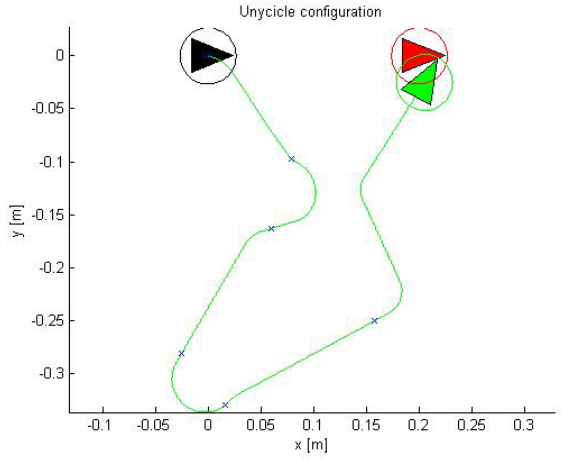


Fig. 39. Con EKF e rumore di misura

Questa prima prova prevede un tracciato semplice, breve, ciononostante risulta interessante per l'analisi delle prestazioni del filtro. In figura 37 si evidenzia il buon funzionamento dell'algoritmo di pianificazione, si osserva solo un leggero errore dovuto alla deriva aggiunta in simulazione per ricreare le peggiori condizioni possibili di funzionamento. In figura 38 invece, in presenza del rumore di misura con il quale lavora la rete WSN, come ci si aspettava la pianificazione non funziona correttamente portando l'e-puck totalmente in direzione opposta all'obiettivo impostato. In figura 39 si può constatare il corretto funzionamento dell'azione combinata tra filtro di Kalman e pianificazione, questo risultato però, data la lunghezza del percorso effettuato, non è performante in termini di tempo minimo di percorrenza. Il comportamento evidenziato in figura può essere il risultato di una stima non precisa dato che nel tragitto sono state rilevate solo sei misure, quindi il filtro, al raggiungimento dell'obiettivo, non ha ancora raggiunto le condizioni di regime. Un'altro particolare da far notare è la posizione finale del'e-puck, in termini assoluti l'obiettivo non sarebbe raggiunto correttamente, ma grazie alla variazione apportata nel file *Simulink* tale obiettivo viene rilassato per permettere di terminare la simulazione senza ri-pianificazioni lunghissime.

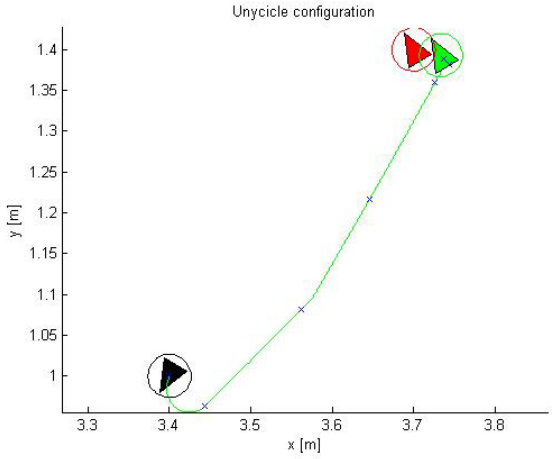


Fig. 40. Senza EKF e senza rumore di misura

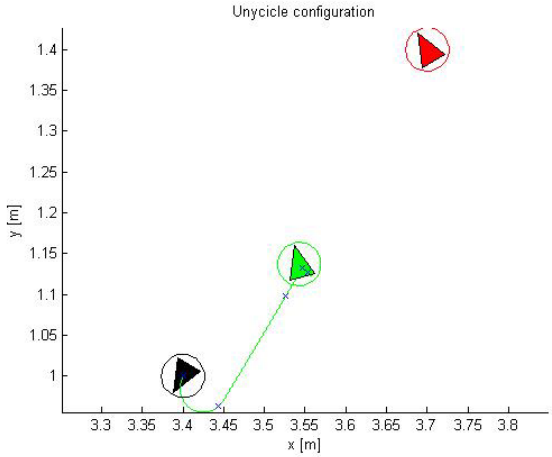


Fig. 41. Senza EKF con rumore di misura

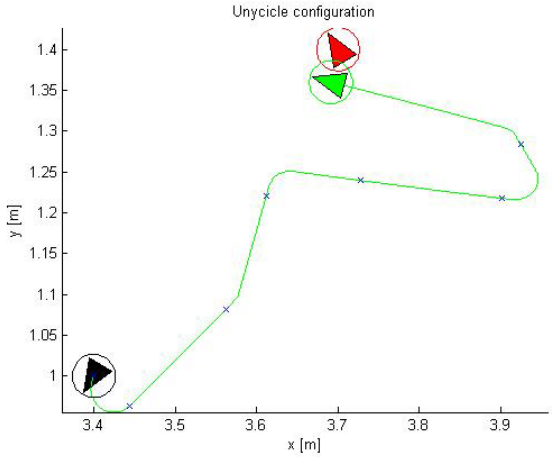


Fig. 42. Con EKF e rumore di misura

Quanto riportato nelle figure 40, 41, 42 è un altro esempio di tragitto breve con angoli di orientazione del punto iniziale e del punto finale scelti in modo da far compiere al robot una rotazione iniziale. Il comportamento senza rumore di misura e senza filtro, tranne per la presenza della deriva, risulta ottimo, mentre introducendo le prestazioni della WSN l'obiettivo è considerato raggiunto molto prima del previsto, con elevato errore di localizzazione. Nell'applicare il filtraggio di Kalman si hanno dei sensibili miglioramenti rispetto al caso precedente, tuttavia anche qui la pianificazione porta fuori strada l'e-puck per poi riuscire a riportarlo in traiettoria. Anche in questo caso non viene raggiunto precisamente l'obiettivo ma il robot viene fermato quando risultano soddisfatte le ipotesi di rilassamento dell'obiettivo.

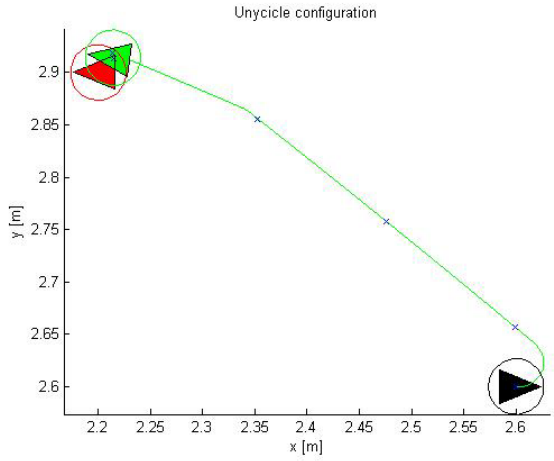


Fig. 43. Senza EKF e senza rumore di misura

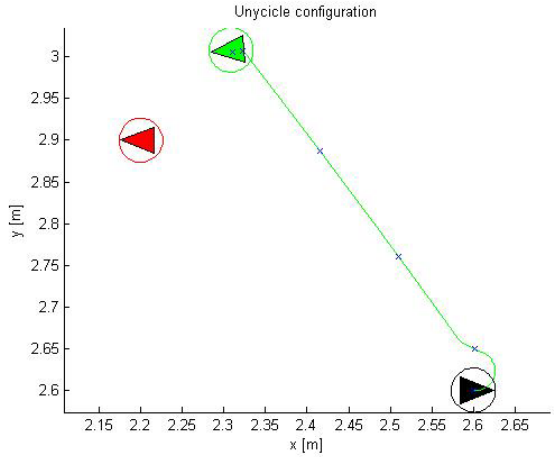


Fig. 44. Senza EKF con rumore di misura

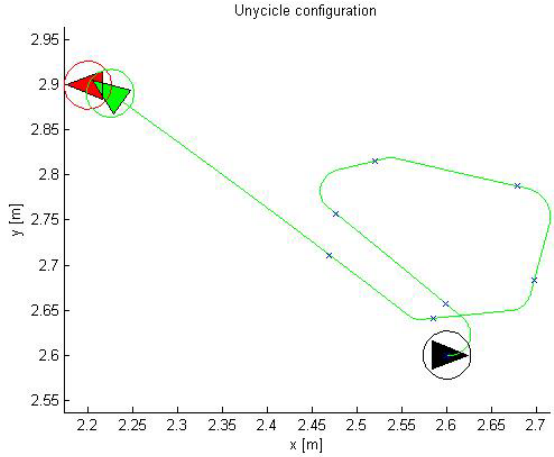


Fig. 45. Con EKF e rumore di misura

Il caso riportato in figura 45 è esemplificativo dell'eventualità discussa nelle sezioni precedenti. All'arrivo di una misura, la ri-pianificazione impone un piano di movimento più lungo di quello creato al passo precedente ma, essendo stata tolta la parte di software che gestiva questi inconvenienti, la traiettoria più lunga non viene scartata. In questo modo l'e-puck è costretto a girare su se stesso per poi tornare a puntare l'obiettivo e successivamente raggiungerlo. Anche in questo caso il risultato risulta scadente. Sebbene l'obiettivo venga raggiunto, sicuramente il percorso attraversato non risulta essere ottimizzato quindi sicuramente non corrisponde a quanto cercato.

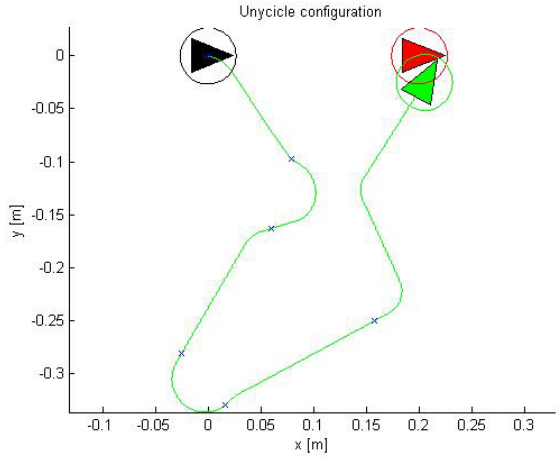


Fig. 46. Con EKF a tempo di acquisizione $1.3sec$

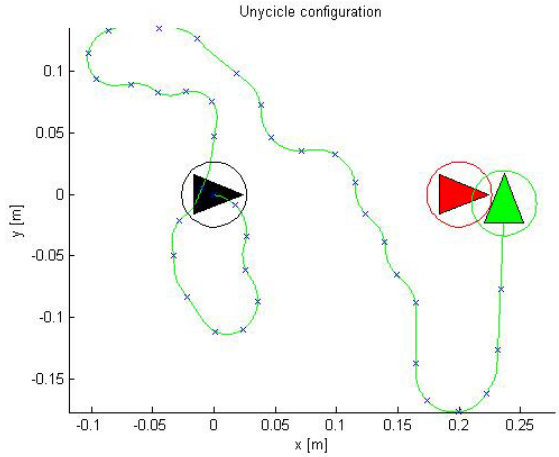


Fig. 47. Con EKF a tempo di acquisizione $0.4sec$

Con queste due immagini si sono volute mettere a confronto le prestazioni di due pianificazioni con medesimi punti iniziali e finali, ma con tempo di acquisizione delle misure diverso. In linea di principio se il tempo di interarrivo delle misure è basso, il filtro di Kalman, avendo misure più frequenti, lavora meglio. Dalle immagini si riscontra invece un pessimo comportamento per entrambi i sistemi, anche se tutti e due puntano comunque verso il punto stabilito. In questo caso si è testato che anche con tempi di acquisizione breve, il software di pianificazione soffre l'incertezza delle misure.

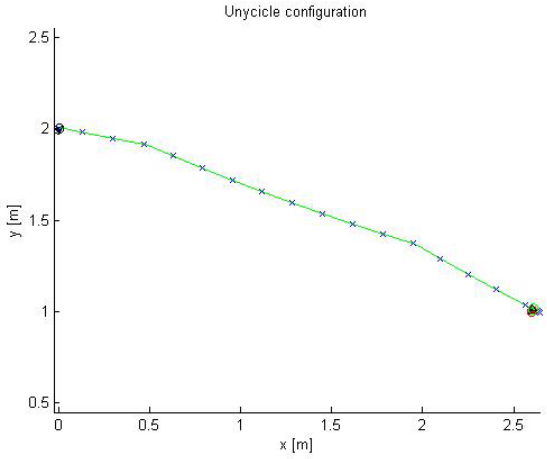


Fig. 48. Senza EKF e senza rumore di misura

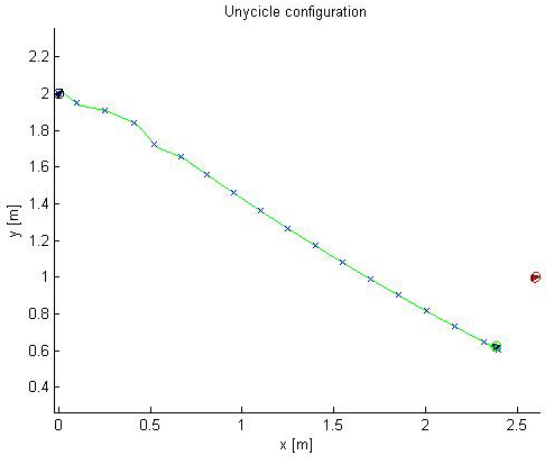


Fig. 49. Senza EKF con rumore di misura

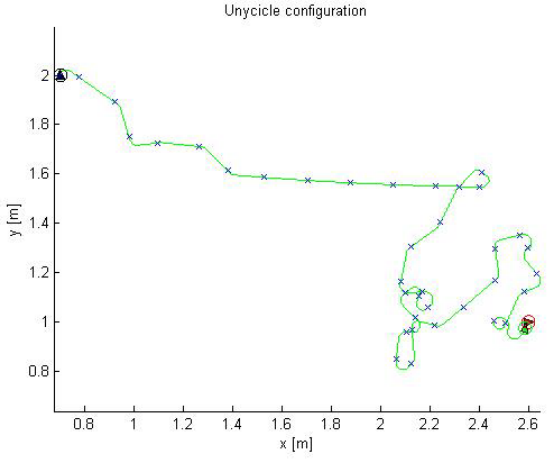


Fig. 50. Con EKF e rumore di misura

Con le immagini 48, 49, 50 si vogliono analizzare le prestazioni del sistema complessivo nell'esecuzione di un tracciato piuttosto lungo, dando così tempo al filtro di andare a regime e poter così lavorare nel migliore dei modi. Anche in questo caso, si evidenzia un funzionamento altalenante del software. Le manovre convergono al punto prefissato, ma come già fatto notare in precedenza, non vengono scartati piani di movimento troppo lunghi rispetto alla distanza residua; si vengono a formare così anelli e percorsi lunghissimi prima di giungere all'obiettivo. Secondo quanto evidenziato in precedenza, coprendo la distanza prevista, il filtro raggiunge la condizione ottima di esercizio perciò la strana traiettoria percorsa può essere imputata ad errori cumulativi di arrotondamento o ad errori numerici che si sommano nel tempo.

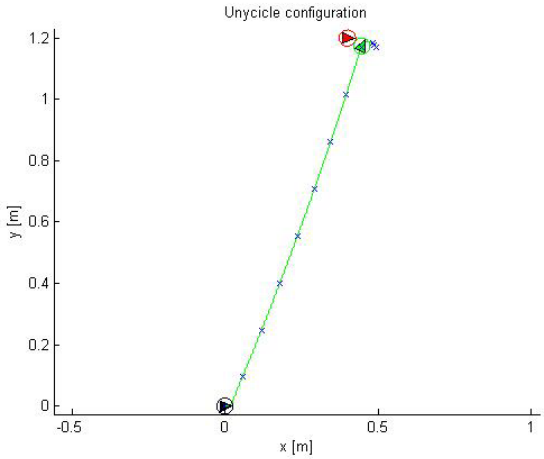


Fig. 51. Senza EKF e senza rumore di misura

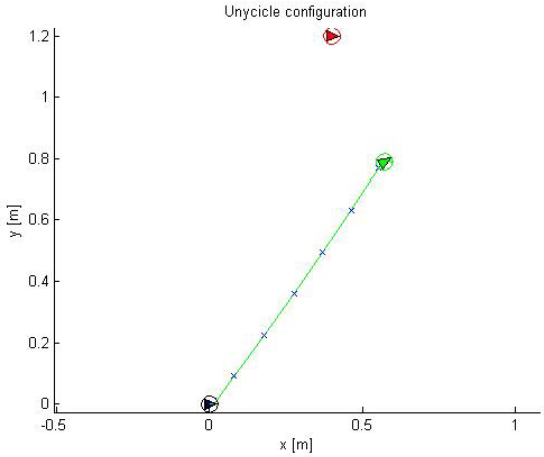


Fig. 52. Senza EKF con rumore di misura

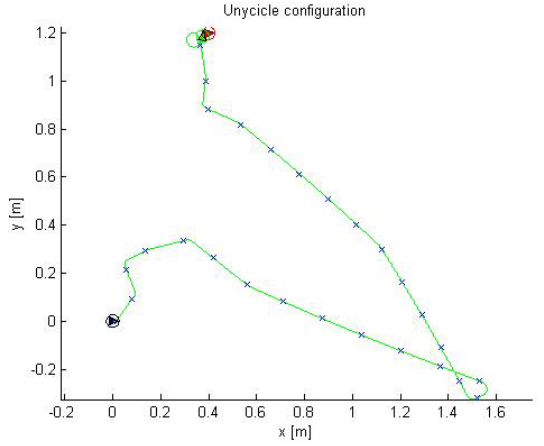


Fig. 53. Con EKF e rumore di misura

I casi riportati in questa pagina e nella seguente permettono di affermare che il software di pianificazione accoppiato al filtro di Kalman soffre di problemi numerici. Risultano infatti inspiegabili le deviazioni compiute nelle figure 53, 56, 59. Ciò nonostante, in momenti successivi, viene recuperata la direzione corretta e viene raggiunto l'obiettivo.

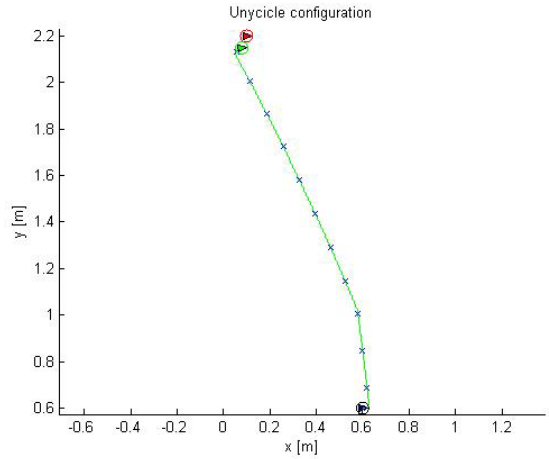


Fig. 54. Senza EKF e senza rumore di misura

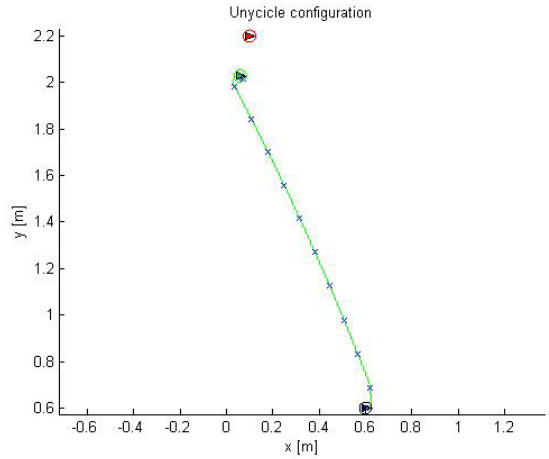


Fig. 55. Senza EKF con rumore di misura

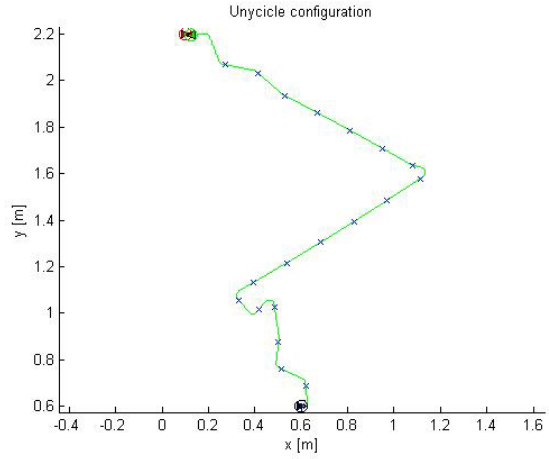


Fig. 56. Con EKF e rumore di misura

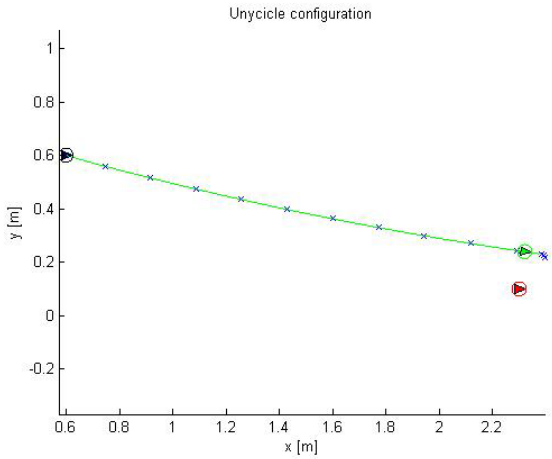


Fig. 57. Senza EKF e senza rumore di misura

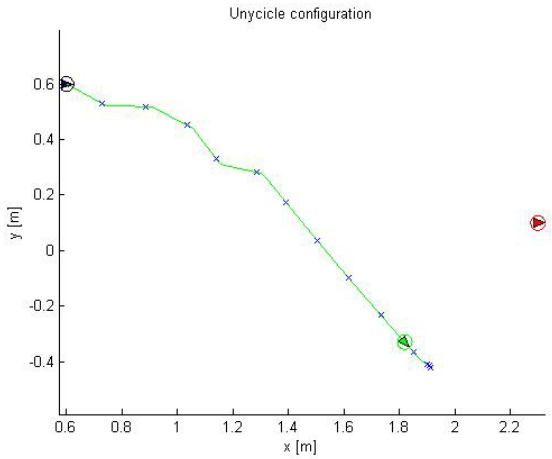


Fig. 58. Senza EKF con rumore di misura

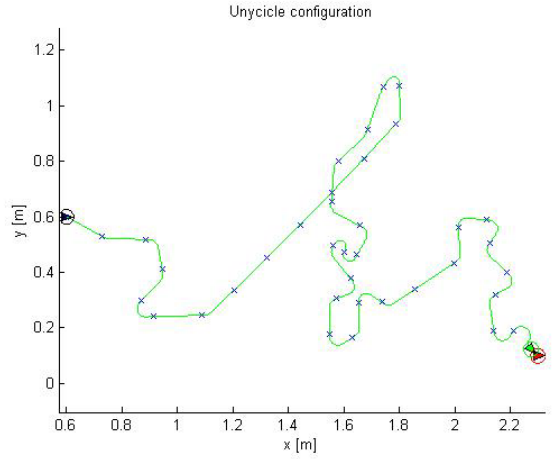


Fig. 59. Con EKF e rumore di misura

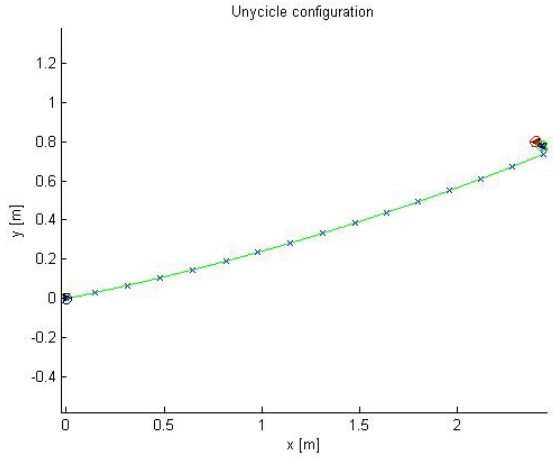


Fig. 60. Senza EKF e senza rumore di misura

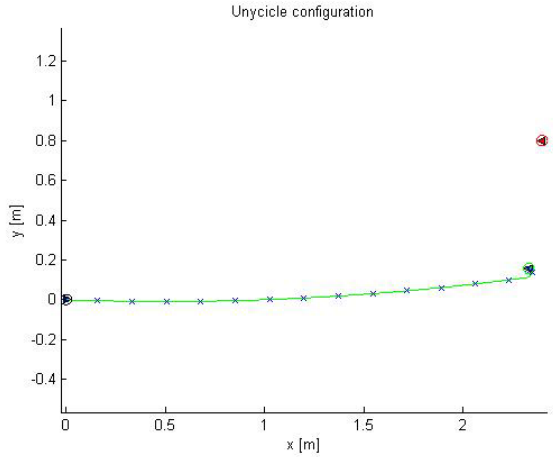


Fig. 61. Senza EKF con rumore di misura

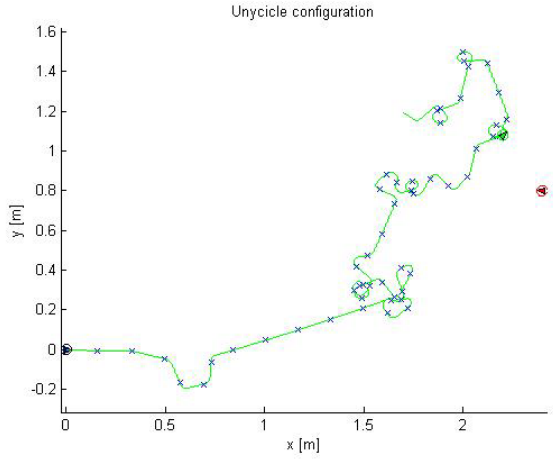


Fig. 62. Con EKF e rumore di misura

Le traiettorie presentate in questa pagina e nella seguente consentono di affermare che il sistema complessivo non è robusto. Si riportano due casi in cui la pianificazione non raggiunge l'obiettivo, anzi, il piano di movimento diverge allontanandosi dal punto finale dopo essersi avvicinato a breve distanza. Tali comportamenti sono difficilmente imputabili ad un cattivo funzionamento del filtro di Kalman, viste le prestazioni offerte nelle simulazioni che lo riguardano, tuttavia è possibile che si creino errori numerici cumulativi che a lungo andare rendano impossibile il funzionamento complessivo. Una ulteriore ipotesi di malfunzionamento potrebbe risiedere nella pesantezza del software complessivo e nell'elevata frequenza cui si chiede di lavorare, questo potrebbe portare ad errori non visibili a livello numerico, ma solo in sede di visualizzazione dei risultati.

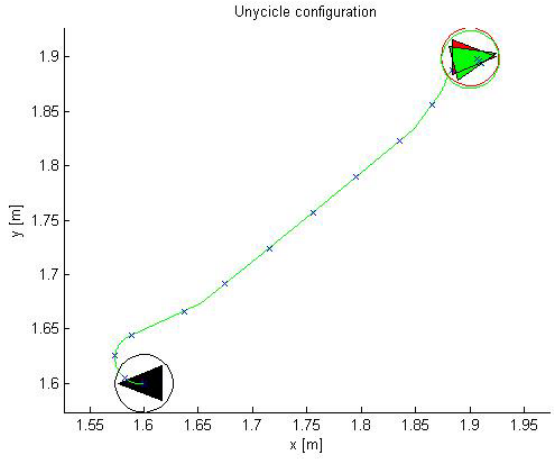


Fig. 63. Senza EKF e senza rumore di misura

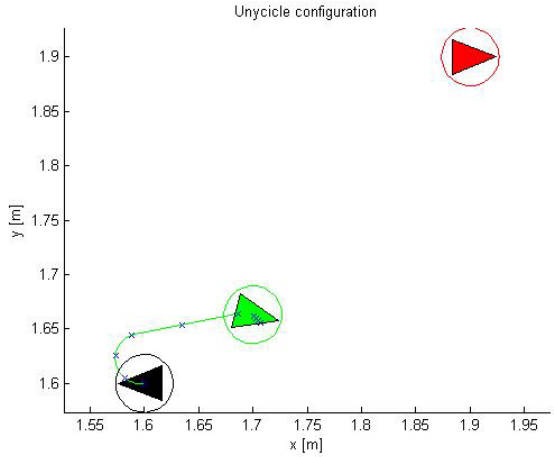


Fig. 64. Senza EKF con rumore di misura

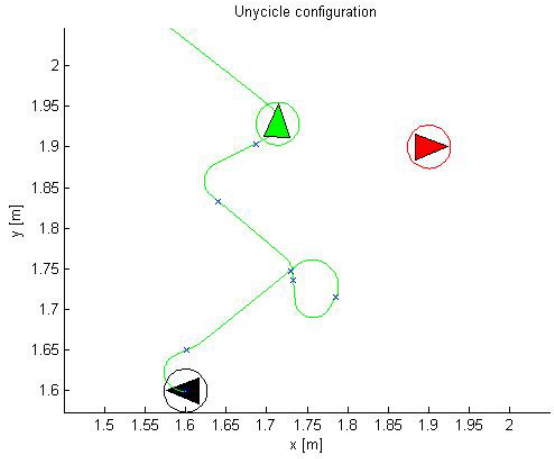


Fig. 65. Con EKF e rumore di misura

E. Conclusioni

Si riporta di seguito, per completezza, un diagramma di flusso del principio di funzionamento del software complessivo:

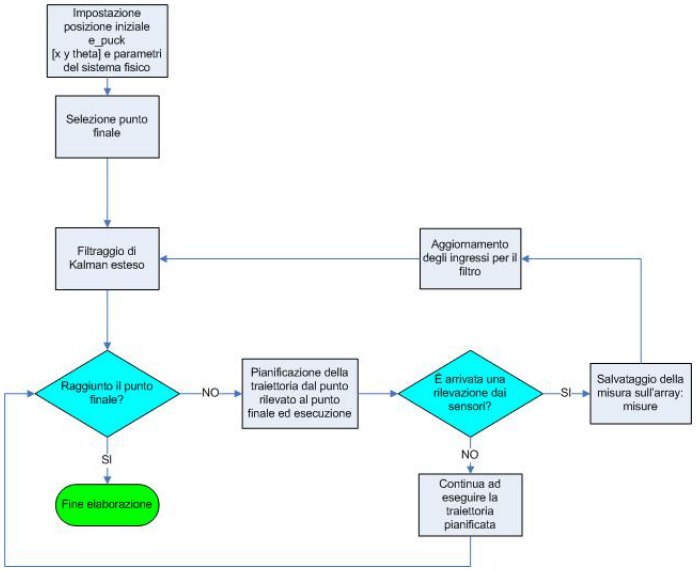


Fig. 66. Flusso decisionale

Secondo quanto si è potuto analizzare nel progetto, si possono evidenziare vari punti a sfavore del sistema pianificazione + EKF:

- l'EKF funziona correttamente se applicato a sottoinsiemi di curve anche se si evidenziano prestazioni meno performanti in concomitanza a curve di raggio minimo.
- la pianificazione di traiettoria in accoppiata al filtraggio con EKF non è robusta. Il sistema non reagisce positivamente in modo uniforme, anzi alterna prestazioni mediocri ad altre pessime, sia con tracciati brevi che lunghi. Non si è in grado perciò di garantire il raggiungimento dell'obiettivo in qualsiasi caso.
- i criteri di pianificazione ottima (curve di raggio minimo, velocità massima di movimento) risultano incompatibili con l'elevata imprecisione della rete di misura prevedendo anche sistemi di correzione come l'EKF

Risultati positivi si potrebbero ottenere rilassando alcune ipotesi tra quelle appena citate ed eseguendo uno studio di caso per la validazione del metodo, ma questo esula dagli obiettivi del progetto.

V. CONCLUSIONI E SVILUPPI FUTURI

Visto che per la localizzazione è stato preferito l'utilizzo del software [8] si potrebbe, come prossimo sviluppo, studiare e valutare più approfonditamente il programma realizzato in [7]. In questo caso, si dovrebbero rivalutare le prestazioni della rete attraverso nuove sperimentazioni pratiche e metterle a confronto con quelle ottenute durante il presente progetto. Infine, per quanto riguarda la parte di localizzazione, si dovrebbe trovare una soluzione per creare un ponte radio tra PC di controllo e robot mobile nell'ambiente di lavoro, in modo da evitare il collegamento via cavo USB dei due dispositivi, soluzione che allo stato attuale causa non poche noie ai movimenti dell'e-puck. Per far questo si potrebbe pensare alla creazione di un nuovo software di localizzazione realizzato ad hoc per l'architettura descritta nel capitolo II, o alla modifica dei sorgenti del TESEO.

Per quanto riguarda i software di filtraggio delle misure e pianificazione di traiettoria, si possono prevedere alcuni miglioramenti futuri. Il filtro di Kalman può essere implementato affinchè possa tener conto anche di perdita di pacchetti e ritardi aleatori di comunicazione, caratteristiche molto frequenti in trasmissioni radio. Inoltre, una volta realizzata la comunicazione tra tutti i componenti, potranno essere sviluppati software per tracking in ambiente con ostacoli di forma e posizione conosciuta o meno. Si potranno anche testare altre tecniche di ottimizzazione del percorso e possibilmente creare un algoritmo nel quale l'e-puck non sia costretto a viaggiare alla velocità massima.

Un ulteriore passo si potrebbe realizzare, riscrivendo il software di simulazione Matlab del filtro di Kalman e il traking ottimo in un linguaggio di programmazione come JAVA o C++ per poi, in un secondo momento, fonderlo al software di localizzazione; in questo modo si creerebbe una versione completa dal punto di vista del prodotto finale.

APPENDIX

Grafici dell'andamento dell'RSS

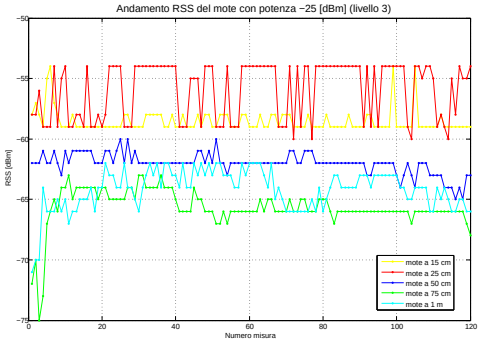


Fig. 67. Grafico dell'andamento dell'RSS misurato tra 2 mote: livello 3

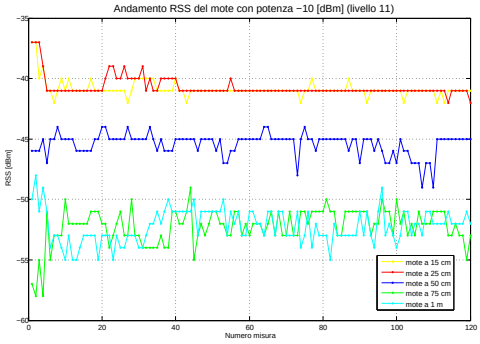


Fig. 68. Grafico dell'andamento dell'RSS misurato tra 2 mote: livello 11

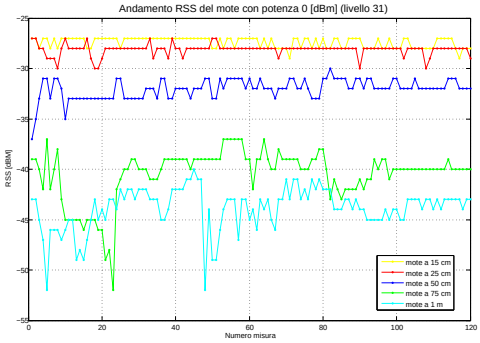


Fig. 69. Grafico dell'andamento dell'RSS misurato tra 2 mote: livello 31

REFERENZE

[1] Moteiv, *T-mote Sky: Ultra low power IEEE 802.15.4 compliant wireless sensor module - Product information*, 2006

[2] Texas Instruments, *Datasheet MSP43X1XX Family - User's Guide*, 2006

[3] Chipcon, *Datasheet CC2420 2.4 GHz IEEE 802.15.4 ZigBee-ready RF Transceiver*, 2005

[4] ST Microelectronics, *Datasheet STM25P80*, 2007

[5] IEEE Standard, *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*, IEEE Standard for Information technology, 2003

[6] Agnoli A., *Controllo di veicoli anolonomi su ruota*, 2007

[7] Bertinato M., Ortolan G., Zambotto P., *Localizzazione e tracking di agenti mobili mediante una rete di sensori wireless*, 2007

[8] Marcassa A., Marcon R., Maran F., Zanella F., *Localizzazione e tracking distribuito tramite rete di sensori wireless*, 2007

[9] Morelli C., Nicoli M., Rampa V., Spagnolini U., Alippi C., *Particle filters for RSS-Based localization in WSN: an esxperimental study*, 2006 IEEE [957-960]

[10] Kumar P. R., web: `http://black.csl.uiuc.edu/~prkumar`

[11] Picci G., *Filtraggio statistico (Wiener, Levinson, Kalman) e applicazioni*, 2006, Edizioni Progetto Padova

[12] Chui C.K., Chen G., *Kalman filtering*, Edizione Springer-Verlag (Cap.8 Extended Kalman filter)

[13] Tipaldi G. D., *Modelli aprossimati per localizzazione e mapping mediante robot mobili*, 2005

[14] Schenato L., *Optimal estimation in networked control systems subject to random delay and packet drop*

[15] web: `http://www.e-puck.org` (sito dell'uniciclo e-puck)

[16] web: `http://www.tinyos.net` (sito dei TMOTESKY)

[17] web: `http://svn.gna.org/viewcvs/e-puck/trunk` (possibili interfaccie dell'e-puck)

[18] *E-Puck Reference Manual 1.0*, 2007

[19] Hubert J., Stirling T. *Introduction to Matlab and the E-Puck*, 2007

[20] Hubert J., *ePic2 Documentation*, 2007

[21] Hubert J., *Software Development for the e-Puck Educational Robot: Coders Guidelines*, 2006

[22] Gay D., Levis P., Culler D., Brewer E., *nesC 1.1 Language Reference Manual*, Maggio 2003

[23] Levis P., *TinyOS Programming*, 2006

[24] National Semiconductor, *LMX9820A Bluetooth Serial Port Module*, 2007

[25] Prof. Alessandro De Luca, Sapienza Università di Roma, *Corso di Robotica 1: Robot Mobili su Ruote*, 2007

[26] Ing. Elisabetta Fabrizi, Università Degli Studi Roma Tre, *Filtro di Kalman esteso applicato al problema della localizzazione nella robotica mobile*, 1998

[27] P. Bahl, V. N. Padmanabhan*RADAR: An In-Building RF-based User Location and Tracking System*, 2000

[28] G. Mao, B. Fidan, B. Anderson, *Wireless Sensor Network Localization Techniques*

[29] R. Crepaldi *Algoritmi di localizzazione per reti di sensori: progettazione e realizzazione di una piattaforma sperimentale*, 2006

[30] L. Parolini *Metodi di localizzazione per reti di sensori wireless*, 2006

[31] M. Andretto *Studio e sperimentazione di algoritmi di localizzazione per reti di sensori*, 2006

[32] M. Maroti, B. Kusy, G. Balogh et al. *Radio Interferometric Positioning*, 2005

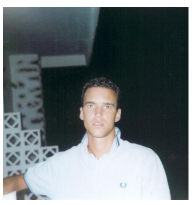
SARTORE FILIPPO

Nato a Camposampiero (PD) il 16/09/83, tuttora vive a Cittadella (PD). Ha conseguito la Laurea Triennale in Ingegneria dell'Automazione presso l'Università di Padova, dove attualmente sta proseguendo gli studi.

SASSARO ALEX

Nato a Malo (VI) il 21/01/83. Ha conseguito la Laurea Triennale in Ingegneria dell'Automazione presso l'Università di Padova nell'a.a. 2005/06.

VETTORI DAVIDE



Nato a Camposampiero (PD) il 28/07/83, residente a Vigodarzere, provincia di Padova. Ha conseguito, presso l'Università di Padova, la Laurea Triennale in Ingegneria dell'Automazione nell'a.a. 2005/06.