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

Bellinato Luca (586291) D'Agostino Massimo (586281) Roveron Francesco (586259)

***Abstract*—Il progetto si pone come obiettivo la localizzazione della posizione di un veicolo mobile e il tracking della sua traiettoria allo scopo di raggiungere una posizione prestabilita nel minor tempo possibile.**
Il primo aspetto del problema è dunque la localizzazione, che viene realizzata tramite una wireless sensor network. Attraverso l'elaborazione della stima della posizione, ottenuta implementando un filtro di Kalman esteso (*EKF*), si effettua poi il controllo del movimento del veicolo. Inizialmente si è proceduto per via simulativa modellizzando la rete di sensori, il rumore di misura, l'uniciclo e implementando il sistema di controllo dello stesso. Successivamente si è testato il tutto sperimentalmente nel laboratorio Navlab presso il Dipartimento di Ingegneria dell'Informazione dell' Università di Padova utilizzando la piattaforma presente e un veicolo mobile (*e-puck*).

I. Introduzione

Il tema della localizzazione dei nodi di una rete wireless è un problema che ormai da anni viene affrontato in letteratura: in modo piuttosto approfondito è preso in considerazione il caso di localizzazione di nodi mobili rispetto ad una serie di nodi fissi detti *àncora* e di coordinate note. Allo stesso modo, il problema del tracking e della pianificazione di traiettoria è un aspetto che è stato al centro delle ricerche e degli studi, con particolare attenzione al caso anolonomo: ogni tipo di veicolo infatti in prima approssimazione può essere rappresentato da un modello anolonomo caratterizzato da equazioni della cinematica non lineari, e la categoria di veicoli su ruote *WMR* (Wheeled Mobile Robot) risulta attualmente la più importante a livello accademico.
Lo sviluppo di questo progetto è partito dall'analisi di queste due tematiche ed in particolare dall'analisi di un progetto presentato nell'anno accademico 2007-2008 per il corso di *Progettazione di sistemi di controllo* [5] in cui si era cercato di implementare ed analizzare il software di pianificazione creato in [6]. Tale pianificazione di traiettoria in accoppiata al filtraggio *EKF* non risultava essere robusta ed i percorsi ottimi che venivano a crearsi risultavano incompatibili con l'elevata imprecisione della rete di misura. Per questo motivo si è scelto di portare avanti un tipo di controllo più semplice ma al tempo stesso più robusto in modo tale da poter essere implementato anche in situazioni in cui si presenta un rumore di misura decisamente elevato. Tutto questo è stato inizialmente implementato in ambiente Matlab per poter determinare, attraverso una serie di simulazioni, sia il comportamento del sistema in diverse situazioni sia individuare possibili punti critici e di forza; successivamente si è cercato di valutare tali risultati ottenuti in simulazione nei laboratori del Navlab, cercando di risolvere tutte le problematiche sorte a livello pratico.
Come punto di partenza si è deciso di effettuare alcune rilevazioni direttamente in laboratorio in modo tale da valutare le reali condizioni di lavoro; particolare attenzione è stata data al rumore di misura la cui modellizzazione verrà poi inserita nelle simulazioni per renderle più similari alla realtà. Si è cercato di costruire un modello matematico del sistema *e-puck* semplice e flessibile scegliendo un modello molto usato in letteratura ossia il modello dell'uniciclo (o macchina di Dubin) che è un tipico esempio di sistema non lineare. Vista l'elevata varianza d'errore di stima della posizione, ottenuta dalla rete *WSN*, si è deciso di implementare 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'ambiente di lavoro nel modo più preciso possibile. Verrà poi illustrata la tecnica di controllo adottata con le relative simulazioni Matlab, notando come l'obiettivo venga raggiunto in maniera soddisfacente. Nell'ultima parte del lavoro si è cercato di implementare in laboratorio la soluzione trovata in fase simulativa e di verificarne i risultati; per far ciò si è dovuto adattare tutto il codice in modo tale che funzionasse con l'architettura a nostra disposizione.

II. Stato dell'arte

In questi ultimi tempi le reti di sensori wireless di tipo distribuito, in cui il processo dei segnali viene distribuito tra i sensori stessi, stanno sempre più incrementando la loro popolarità per il grandissimo numero di applicazioni possibili nelle discipline scientifiche più diverse. Tali

applicazioni possono essere raggruppate come prima analisi in tre grandi categorie:

- *Monitoring*: tipo di rete utilizzato per tener traccia di grandezze relative ad una particolare area geografica (monitoraggio ambientale dell'acqua, dell'aria, chimica del suolo, monitoraggio di possibili incendi in zone boschive o a rischio ecc...)
- *Tracking e Localization*: vengono sfruttate le capacità di riorganizzazione della topologia della rete di sensori per identificare gli spostamenti di un oggetto in una determinata area geografica (per esempio in campo militare, in quello edile nelle zone di cantiere a rischio, negli ospedali per migliorare il controllo sulla degenza dei pazienti, ecc...)
- *Controlling*: tipo di rete finalizzato a riconoscere determinati eventi relativi alle misure rilevate da sensori o gruppi di sensori (nei campi dell'agricoltura di precisione, strumentazioni di fabbrica, videosorveglianza wireless, ecc..

La base di questo progetto sperimentale è la Wireless Sensor Network implementata in Navlab tramite i nodi mobili della *Moteiv*. Questa rete di nodi mobili è stata oggetto di numerosi progetti e ricerche in ambito accademico: dal monotoraggio ambientale in serra, alla localizzazione e tracking in un edificio, allo studio delle proprietà dei segnali trasmessi. Noi partiamo dal software di localizzazione TESEO funzionante a livello teorico su diverse reti di sensori, da quella dislocata sul piano del DEI a quella posta sotto la piastra. Tale strumento si è rivelato impreciso ma funzionante; nonostante ciò non lo si era mai provato a utilizzare per una gestione completa del movimento di alcun dispositivo affidandosi solo alle stime da esso fornito. In generale la localizzazione di un nodo mobile posto all'interno di un *WSN* avviene per via geometrica, sfruttando la triangolazione, che consiste nella raccolta di misure di un determinato parametro proveniente al sensore da almeno tre nodi àncora: il nodo mobile infatti spedisce all'elaboratore tutti i pacchetti che riceve dalla rete, contenenti l'identificativo del trasmittente e accorpandoci la misura di potenza del segnale ricevuto; conoscendo l'andamento della potenza nel segnale radio utilizzato, il computer provvede alla stima di posizione. Per maggiori approfondimenti si rimanda a [31].

Già in fase di ricerca delle informazioni ci si è imbattuti sul problema della modellizzazione del canale trasmissivo su cui si basa l'algoritmo di localizzazione appena spiegato. Durante la propagazione dell'onda elettromagnetica il campo è sottoposto ad attenuazione; questa tende ad aumentare con la distanza per fenomeni fisici di assorbimento ed in ambiente urbano è appesantita ulteriormente dai fenomeni di riflessione e diffrazione. L'entità dell'attenuazione subita dal segnale trasmesso è un parametro che deve essere attentamente valutato in fase di progetto di un sistema di comunicazione in quanto determina la massima distanza che può essere raggiunta con una data tecnica di trasmissione. L'ambiente in cui lavoreremo può essere definito ostile per via degli spigoli e ostacoli presenti in numero elevato nella stanza ma risulta necessario determinare la relazione tra RSSI e potenza ricevuta per poter avere un modello più verosimile possibile. Nella sezione dedicata al TESEO è descritto il modello utilizzato che dipende principalmente da tre parametri. Abbiamo deciso di considerare attendibili i valori della pedana utilizzati fino ad oggi relativi ai parametri caratteristici sebbene sarebbe necessario uno studio approfondito sulla modellizzazione del canale trasmissivo ipotizzando per esempio l'applicazione degli studi effettuati da alcuni dottorandi e a noi accennati da *Del Favero* in una esposizione in aula.

Il secondo caposaldo su cui è basato il progetto consiste nei robot mobili su ruota o *wheeled mobile robot* (*WMR*) che hanno conosciuto in anni recenti un notevole sviluppo grazie agli sforzi della ricerca nel campo dei sistemi non lineari e alle emergenti tecnologie a basso costo. Oltre al naturale interesse teorico, il problema della pianificazione del percorso e del controllo di *WMR* ha attratto grande interesse in vista delle possibili applicazioni nel settore dell'automotive, della sorveglianza e dell'esplorazione di ambienti sfavorevoli all'uomo. Nel nostro caso l'uniciclo è stato modellizzato come un veicolo di Dubins; tale scelta è motivata dalle seguenti ragioni:

- quasi tutti i veicoli, terrestri, marini ed anche aerei che volino a quota costante, possono essere modellati, almeno in prima approssimazione, come veicoli di Dubins sul piano;
- si tratta di un modello largamente studiato, poichè, da un lato, è descritto da equazioni semplici, dall'altro è fortemente non lineare; tuttavia quasi tutti gli studi condotti sinora presumono la regolazione a tempo continuo, mentre il controllo digitale di un veicolo di Dubins è stato scarsamente considerato;
- il modello di Dubins si può applicare con ottima approssimazione, in particolare, agli unicicli e-puck, in dotazione al laboratorio Navlab del dipartimento d'Ingegneria dell'Informazione dell'Università di Padova: il controllo del movimento degli e-puck è il primo passo per studi futuri di robotica coordinata.

Uno dei problemi principali che devono essere risolti per i robot mobili e i veicoli autonomi è la pianificazione del movimento per raggiungere un obiettivo da una data condizione iniziale. Ulteriori vincoli possono essere imposti dalla presenza di ostacoli nell'ambiente in cui il veicolo si trovi ad operare.

I classici problemi di controllo di veicoli su un piano per i quali sono già stati sviluppati numerosi algoritmi sono:

- *path following*: il robot deve raggiungere e seguire un determinato cammino geometrico nello spazio cartesiano, partendo da una generica posizione iniziale facente parte o meno del cammino geometrico, senza dover soddisfare determinate specifiche temporali. Viene dunque data una descrizione geometrica del cammino cartesiano assegnato: tale informazione è solitamente espressa in forma parametrica e quindi esprime il moto desiderato in funzione del parametro. Per questo particolare tipo di *task* la dipendenza dal tempo non è rilevante in quanto l'unico parametro da controllare è legato all'errore geometrico di piazzamento tra robot e cammino da seguire;
- *trajectory tracking*: con tale tecnica si impone che il robot debba raggiungere e seguire una determinata traiettoria geometrica nello spazio cartesiano, cioè un cammino assegnato ad una determinata legge temporale partendo da una generica posizione iniziale che può eventualmente far parte della traiettoria;
- *point-to-point motion o posture stabilization*: il robot deve raggiungere una determinata configurazione desiderata partendo da una generica configurazione iniziale

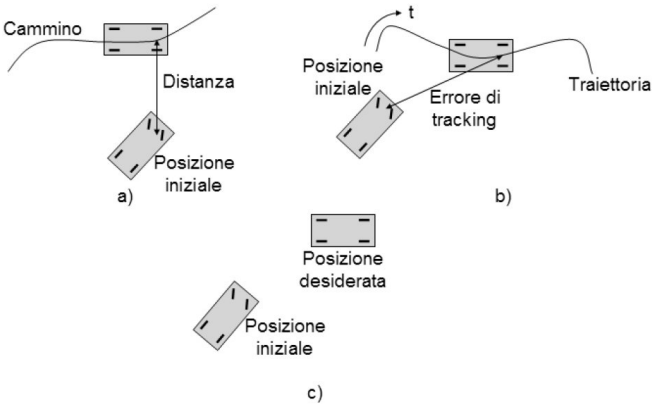

Fig. 1. *Problemi di controllo: a) Path following, b) Trajectory tracking, c) Point-to-point motion*

Per risolvere le problematiche sopra presentate si possono utilizzare svariati metodi; in ambiente accademico gli algoritmi che hanno riscosso più successo sono quelli basati sul controllo non lineare secondo Lyapunov [12], sul controllo mediante *feedback linearization* [10] e alcuni algoritmi basati sul controllo predittivo e sue varianti [11], [6]. Tutte queste tecniche di controllo sono molto efficaci e garantiscono buone prestazioni al costo di una puntuale conoscenza dello stato del sistema, condizioni queste non possibili per l'hardware a nostra disposizione. Ad esempio la tecnica di *feedback linearization* applicata nel progetto [13] funzionava correttamente grazie all'uso di una telecamera che garantiva un'alta precisione di misura; nel caso si fosse usata esclusivamente la rete di sensori *WSN*, tale algoritmo non si sarebbe potuto mettere in pratica. Nel progetto presentato nell'anno accademico 2007-2008 per il corso di Progettazione di sistemi di controllo [5] si era cercato di implementare ed analizzare il software di pianificazione creato in [6] basato sul controllo predittivo unicamente usando la rete di rilevazione wireless. Tale pianificazione di traiettoria, in accoppiata al filtraggio *EKF*, non risultava essere robusta ed i percorsi ottimi che venivano a crearsi risultavano incompatibili con l'elevata imprecisione della rete di misura. In questo progetto sperimentale dunque, non si è provato a mettere in pratica uno degli algoritmi citati in precedenza, bensì si è progettata una tecnica di controllo che potesse essere realmente messa in pratica usando come rilevazione delle misure solamente la *WSN* presente nel laboratorio Navlab. Come è intuibile, una delle difficoltà che si andrà ad affrontare in questo progetto è simile ad un problema di controllo punto-punto; una delle difficoltà maggiori presenti in questo tipo di controllo è principalmente legato all'anolomia del veicolo. In generale si ha a disposizione un numero di gradi di libertà inferiore al numero di variabili di stato del veicolo da controllare: nel caso dell'uniciclo si hanno due controlli (velocità lineare ed angolare) per uno spazio di stato di dimensione tre (posizione del punto centrale dell'interasse delle ruote ed orientamento del veicolo). L'elevata incertezza delle misure fornite dalla rete *WSN* non permette di conoscere con elevata precisione la posizione del robot da controllare quindi si considererà come target da raggiungere una circonferenza di raggio congruo con la varianza delle misure a disposizione. Un ulteriore semplificazione risiede nel fatto che l'obiettivo può essere raggiunto con qualsiasi angolo di orientazione, infatti essendo quest'ultimo il parametro soggetto ad una maggiore incertezza, specificare un angolo di arrivo sarebbe del tutto inapplicabile.

III. MODELLO DEL SISTEMA UNICICLO

La scelta del modello riveste un'importanza fondamentale, e influisce in modo determinante sulla progettazione del sistema di controllo; naturalmente tale scelta dipende sia dallo specifico sistema in esame, sia dal tipo di controllo che si vuole ottenere. Nel nostro caso, in considerazione della tipologia di robot che utilizzeremo, abbiamo scelto il modello cinematico dell'uniciclo, che governa il comportamento delle variabili di maggior interesse per i nostri obiettivi, ovvero posizione e velocità angolare e lineare dei robot.

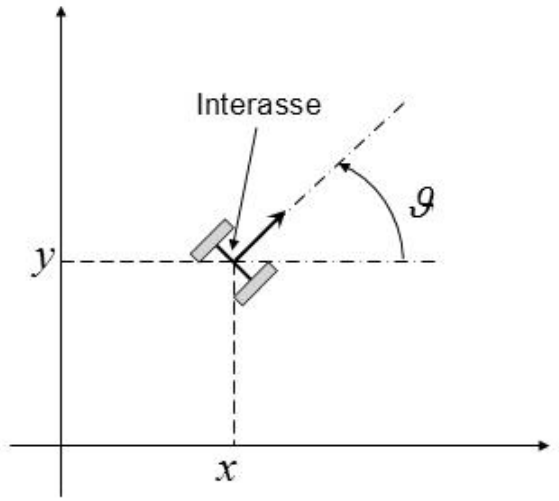

Fig. 2. *Schematizzazione dell'uniciclo: le variabili di principale interesse sono la posizione del baricentro del robot* (x, y) *e la posizione angolare* θ.

Lo studio del sistema *e-puck* viene associato dunque allo studio di un particolare sistema: *la macchina di Dubins* poiché ne è un'ottima approssimazione ed è 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.

Il modello a tempo continuo del sistema, senza considerare il rumore, è il seguente:

$$\begin{cases} \dot{x_1} = u_1 \cos(\theta) \\ \dot{x_2} = u_1 \sin(\theta) \\ \dot{\theta} = u_2 \end{cases}$$

in cui: $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \theta \end{bmatrix}$ è il vettore di stato, le cui componenti rappresentano rispettivamente la misura delle coordinate x,y e θ; mentre le componenti del vettore di ingresso $\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ rappresentano rispettivamente la velocità lineare ed angolare.
Nonostante il modello di Dubins imponga un raggio di curvatura limitato inferiormente, si è pensato di non cosiderare tale ipotesi e di realizzare la rotazione dell'*e-puck* ruotandolo su se stesso e quindi mantenendone inalterato il baricentro. Tale semplificazione è del tutto trascurabile data l'elevata rumorosità delle stime. La discretizzazione del modello è fatta con il metodo $Runge - Kutta\ del\ secondo\ ordine$.
Il modello a tempo continuo $\frac{dx_i}{dt} = f_i(t, x_1, \cdots, x_N)$ viene così discretizzato:

$$\begin{aligned} \mathbf{x}(k+1) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}(k) + \\ &+ \begin{bmatrix} T_c \cos(\theta(k) + \frac{u_2(k)T_c}{2}) & 0 \\ T_c \sin(\theta(k) + \frac{u_2(k)T_c}{2}) & 0 \\ 0 & T_c \end{bmatrix} \mathbf{u}(k) \end{aligned}$$

per la derivazione del modello discretizzato si veda l'appendice A.

IV. ARCHITETTURA DEL SISTEMA

Avere chiara la struttura del sistema e la sua realizzazione è un requisito fondamentale al raggiungimento degli obiettivi preposti. Lo scopo principale del progetto riguarda la localizzazione e il tracking a tempo minimo, tramite rete di sensori wireless, di un oggetto mobile posto in un punto casuale della piastra presente in laboratorio Navlab. Si vuole cioè realizzare un algoritmo che permetta al robot di stimare la propria posizione e muoversi di conseguenza per raggiungere, entro una certa tolleranza, un punto prefissato detto GOAL possibilmente minimizzandone il tempo necessario.

L'ambiente di lavoro utilizzato per la sperimentazione risulta ridotto ad una piattaforma di dimensioni 3.20 per 2.40 metri, in cui sono stati disposti, in maniera uniforme, 64 sensori *WSN* (3), secondo la griglia riportata in 4, dove i sensori sono indicati con dei quadratini

rossi. In 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. Descriveremo ora le principali caratteristiche dell'hardware utilizzato

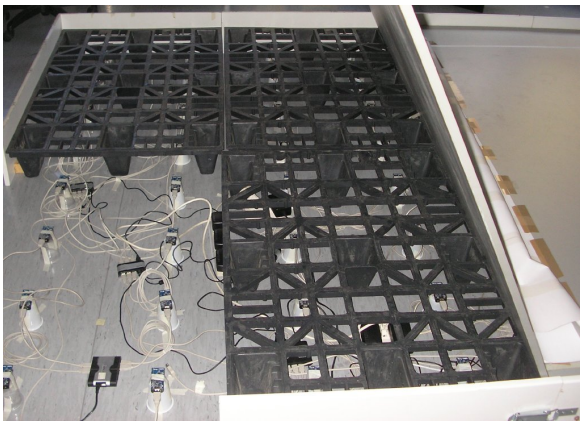

Fig. 3. *realizzazione della WSN*

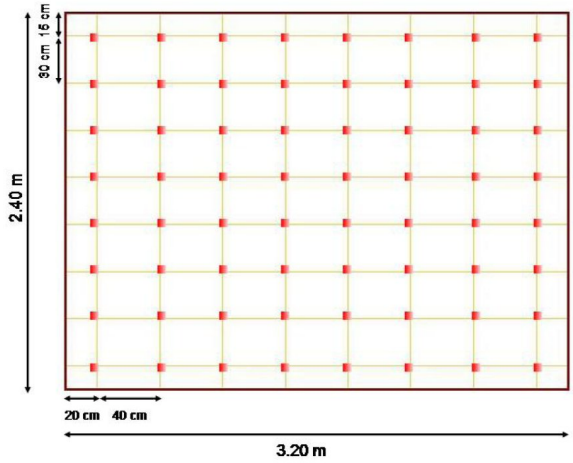

Fig. 4. *disposizione dei sensori nella piastra*

A. Wireless sensor network

Le reti che sfruttano la tecnologia wireless si suddividono in base alla potenza delle onde magnetiche utilizzate e, di conseguenza, in base alle distanze coperte. In questa trattazione si considera una PAM (personal area network): una rete a bassa potenza capace di coprire lunghezze non superiori ai 50 metri indoor e ai 150 metri outdoor. In generale, una *WSN* (wireless sensor network), è costituita da dispositivi programmabili, chiamati nodi, capaci di comunicare via radio tra loro e spesso dotati di sensori (come ad esempio per la temperatura) in grado di instaurare delle connessioni ben distribuite e capillari con l'obiettivo di monitorare attentamente un'area territoriale (industriale, faunistica, urbana..). Nel nostro caso è importante poter mettere in relazione la potenza dei segnali tra un dispositivo e l'altro con la distanza che li divide e sfruttare questo per l'algoritmo di localizzazione.

Il punto cruciale delle *WSN* è normalmente l'aspetto energetico; problema questo che non ci interessa dato che l'obiettivo preposto ci svincola dalla gestione di minima potenza: si farà in modo di lavorare utilizzando una buona fonte di energia preoccupandoci solo che la tensione fornita ai dispositivi dalla batteria sia sufficiente almeno all'interno dell'arco di simulazione; infatti, in caso contrario, i dispositivi non smetterebbero di funzionare completamente ma i singoli dispositivi integrati inizierebbero a funzionare in maniera anomala senza segnalazioni particolari; la comunicazione bluetooth e la precisione dei motori passo passo del robot sono i primi a risentire di una tensione insufficiente.

Per questo tipo di reti, diversamente che per gli standard utilizzati nell'informatica, i protocolli sono snelli perché pensati su misura per i dispositivi su cui si lavora. Questo comporta una completa gestione dell'hardware, fino ai livelli più bassi, con un incremento delle potenzialità e un miglior controllo.

B. TmoteSky

Per la realizzazione della rete di sensori con cui effettuare la localizzazione si sono utilizzati i dispoitivi già presenti in laboratorio: i *TmoteSky* prodotti dalla *Moteiv* Corporation raffigurati in 5. Ciascun dispositivo integra

- un microprocessore prodotto dalla Texas Instruments, l'MSP430 è caratterizzato da 10kB di RAM, 48kB di flash ROM e un'architettura 16bit RISC. Ha un oscillatore (DCO) che opera sopra gli 8 MHz e un oscillatore al quarzo utile alla calibrazione del primo. E' dotato ancora di 8 porte ADC interne e altrettante esterne, necessarie per leggere i valori dei sensori o dei livelli di batterie.
- una radio, nello specifico il Chipcon CC2420; questo fornisce tutto il necessario per comunicazioni wireless affidabili basate sullo standard IEEE 802.15.4. Viene controllata dall'MSP430 tramite il bus SPI e possono essere programmati i parametri fondamentali per la trasmissione: potenza, frequenza e gruppo. La radio lavora infatti attorno ai 2.4GHz con una modulazione O-QPSK, ma è possibile selezionare tra 16 canali differenti numerati

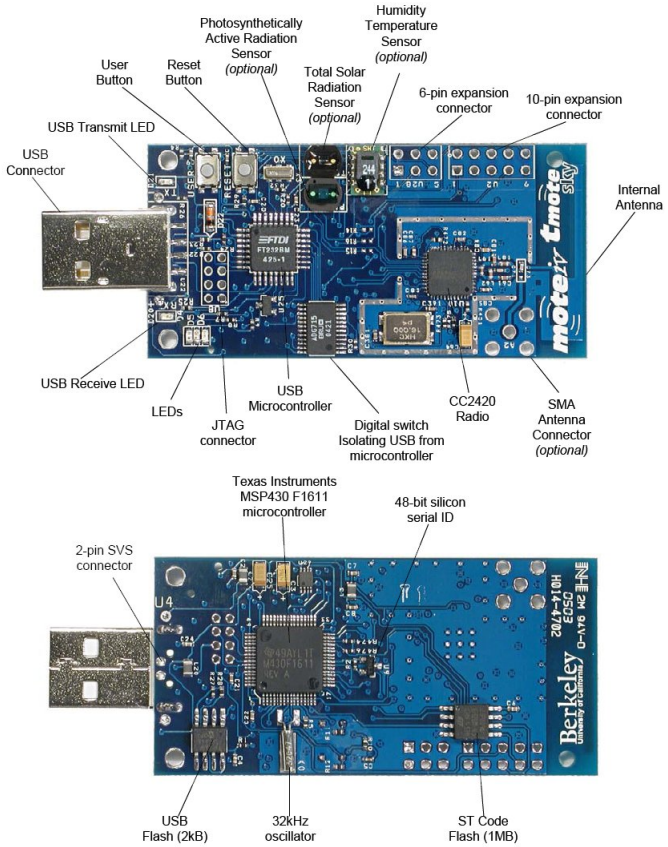

Fig. 5. *TmoteSky*

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 di seguito dal più elevato (31) al più debole (3). Questo integrato fornisce ancora un indice di robustezza del segnale ricevuto (RSSI) e un indicatore di qualità della connessione (LQI); è proprio sulla base di questi dati che il software TESEO elabora la stima della localizzazione.

- una interfaccia USB; infatti il *TmoteSky* utilizza un USB controller per gestire la comunicazione seriale col PC. Una volta inserito sulla relativa borchia verrà visualizzato come dispositivo nelle macchine Linux o OSX, e come COM in Windows; gli verrà inoltre assegnato un indirizzo identificativo univoco.
- un'antenna interna, posta parallelamente alla superficie della basetta; esiste la possibilità di montarne di esterne tramite il connettore apposito. Ha un range di copertura di 50 metri indoor e oltre 125 metri outdoor ma non è perfettamente omnidirezionale e l'irraggiamento può risentire della presenza o meno delle batterie e persino della posizione del *mote* (si ha un incremento delle prestazioni posizionandolo in verticale piuttosto che disteso come si può notare confrontando le figure 6 e 7 dei grafici dell'irraggiamento).

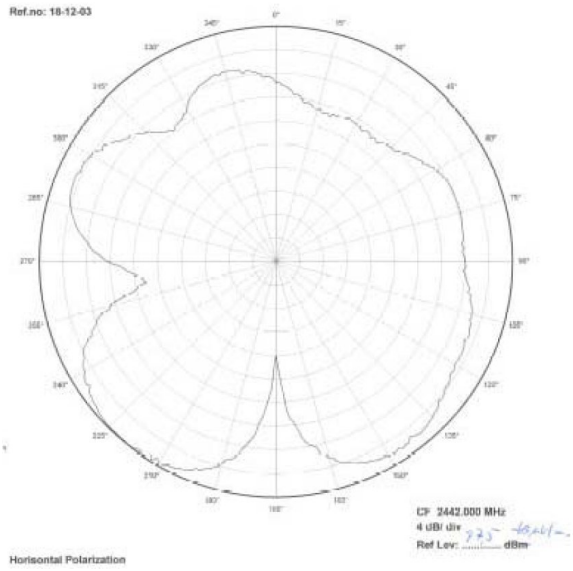

Fig. 6. *radiazione del nodo T-moteSky, posizione orizzontale*

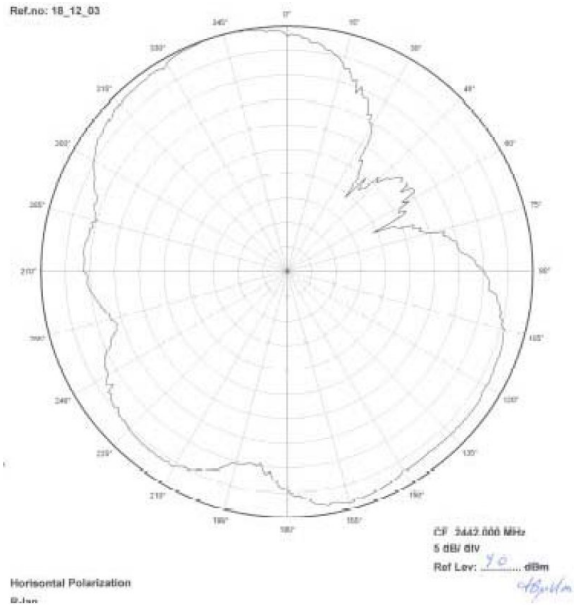

Fig. 7. *radiazione del nodo* TmoteSky*, posizione verticale*

- una flash esterna, più precisamente al ST M25P80 40MHz che rende disponibile 1024kB per memorizzare dati: è composta di 16 segmenti di 64kB l'uno ed è collegata al microcontrollore tramite l'SPI, lo stesso bus che collegava calcolatore e radio. Questa condivisione delle risorse richiede un minimo di gestione; infatti lo stesso mezzo può essere utilizzato da un unico componente alla volta

e non è possibile, per esempio, scrivere nella flash e ascoltare la radio contemporaneamente.
- sensori ambientali di temperatura e umidità oltre a un indicatore del livello delle batterie e uno delle temperatura interna del microcontrollore; entrambi non sono molto precisi e richiedono di essere calibrati.
- alcuni connettori per l'espansione utili per controllare, tramite *Tmote*, altri dispositivi (per esempio relè, display LCD e altre periferiche digitali) o per connettere, tramite apposita scheda, un *TmoteSky*; ce ne sono due, uno a 6 e uno a 10 pin.

C. Uniciclo e-puck

Per quanto concerne il veicolo mobile, è stato utilizzato un *e-puck* (*figura 8*), distribuito da *Epfl* (*École polytecnique federale de Lausanne*).

Fig. 8. *e-puck*

Sviluppato su un diametro di 70 mm, è costituito di un telaio plastico che sostiene due motori, il circuito stampato e la batteria allocata nella parte inferiore della struttura è collegata grazie a due contatti verticali. La batteria può inoltre essere estratta per ricaricarla esternamente. Esso monta motori passo-passo: con ingranaggi di riduzione, ciascun motore dispone di 20 passi per la rivoluzione mentre per la marcia vi è predisposta una riduzione pari a 50:1. L'ultimo asse sostiene direttamente la ruota.
Le ruote aventi un diametro di 41mm ciascuna e distanti tra loro 53mm si fissano all'asse mediante un mandrino a quattro denti. Esse sono poi avvolte da pneumatici

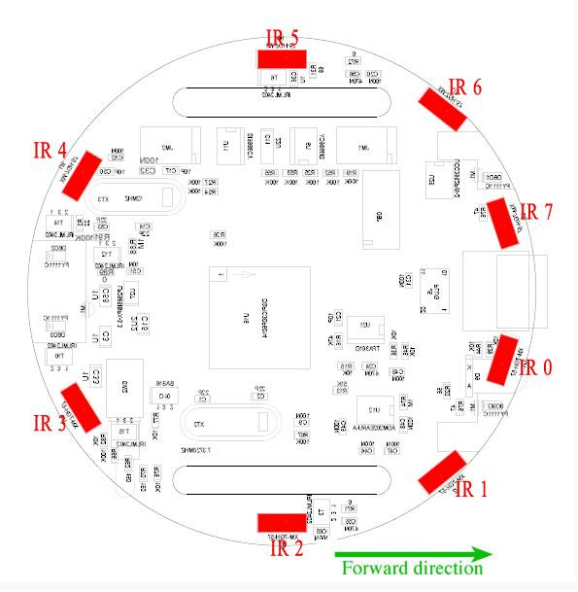

Fig. 9. *particolare dei sensori di prossimità*

costitutiti da anelli in gomma. La massima velocità delle ruote è di circa 1000 passi/s, che corrisponde ad una rivoluzione della ruota al secondo.
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 (vedi fig. 9) posti uniformemente 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* ([26]) e la seconda per la comunicazione seriale. Maggiori informazioni possono essere ricercate in [17], sito ufficiale dell'*e-puck*.

V. Software utilizzato

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. Infine si accenna all'espediente usato per utilizzare le stime realizzate dal TESEO, scritto in JAVA, dal software di controllo del robot, implementato invece in Matlab.

1) Introduzione al TINYOS *e al* NESC*:* I *Tmote* sono gestiti da un piccolo sistema operativo dedicato, il TINYOS. E' appositamente realizzato per le reti di sensori ed è caratterizzato da una architettura a moduli che sono caricabili singolarmente. In questo modo il programmatore utilizza solo quelli realmente necessari alla sua applicazione, come per esempio quello per la radio o per la scrittura in Flash, snellendo la dimensione dell'applicativo da memorizzare in RAM. Ogni modulo fornito implementa o fornisce interfacce software in grado anche di gestire la risposta agli eventi. Il linguaggio di programmazione utilizzato invece per programmare i *Tmote* si chiama NESC e di fatto è una estensione del C basato sulla guida ad eventi, appositamente pensato per realizzare i concetti e le strutture di esecuzione del TINYOS. La completa concatenazione di sistema operativo e applicazioni rende impossibile caricare più programmi sullo stesso *mote* tanto meno modificarli in maniera dinamica. Le caratteristiche principali di questo linguaggio sono:

- Separazione del costrutto NESC e dei moduli propri del sistema operativo: un generico programma è costituito di un file chiamato configurazione e di uno detto modulo; il primo richiama i componenti che forniscono le interfacce e definisce la connessione dei vari componenti (wiring) mentre il secondo contiene il codice da eseguire.
- Ciascun componente, attraverso delle interfacce, esercita le sue funzionalità mettendo a disposizione dei comandi a ciascuno dei quali fa corrispondere degli eventi. L'utilizzatore lavora attraverso i primi e gestisce il flusso del programma tramite il verificarsi degli eventi. La chiamata a queste azioni e la gestione degli avvenimenti viene implementata nel file di modulo. Un semplice esempio è quello del componente che gestisce i timer: mette a disposizione dei comandi, coi quali si sceglie di far partire il clock (in maniera ciclica o meno e impostandone i tempi), e un evento che ne segnala le scadenze.

Il programma di localizzazione, utilizzato nel progetto, presenta tre tipi di applicazione: il *Mobile Node*, relativo al nodo mobile posizionato sopra all'uniciclo, l'*Anchor Node* da installare nei 64 nodi fissi posizionati nella rete posta sotto la pedana, e la *Base Station* attaccata al calcolatore per ricevere i dati dal NODO MOBILE via WiFi.

2) Nodo Mobile: Il *Tmote* posto sopra il robot è ad esso collegato tramite apposita scheda e riceve alimentazione proprio dalla batteria del robot; questo rende superfluo il controllo dello stato della tensione al NODO MOBILE date le buone qualità della batteria al litio utilizzata.

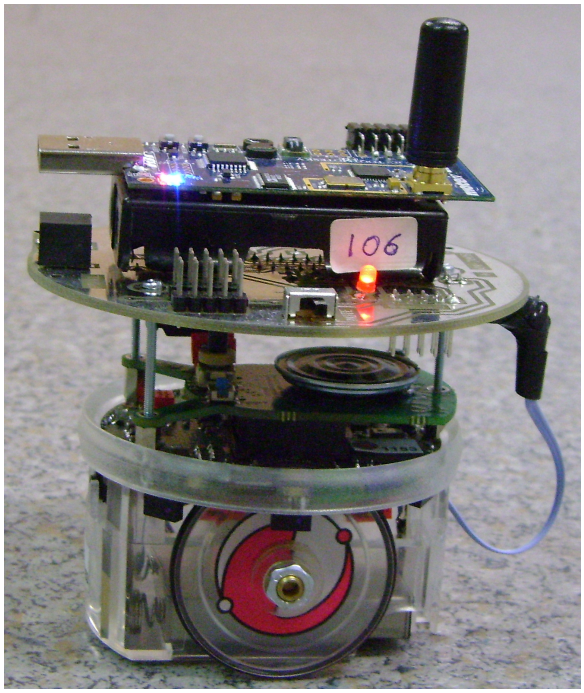

Fig. 10. e-puck *usato nelle simulazioni in laboratorio*

Il dispositivo è programmato per ricevere i pacchetti spediti dalla *WSN* e inoltrarli a sua volta tramite la radio e la BASE STATION all'algoritmo di TESEO che provvederà all'elaborazione dei dati. L'utilizzo del chip radio sia per ricevere che per inviare i dati prevede una temporizzazione degli eventi poiché, come già spiegato, le due operazioni non possono essere realizzate contemporaneamente. Questo significa quasi dimezzare il numero di informazioni passate a TESEO e quindi una dinamica più lenta dell'intero sistema di controllo; d'altronde era necessario svincolare il robot dall'elaboratore e renderlo il più possibile autonomo.

3) Nodi àncora: I NODI ÀNCORA sono programmati semplicemente per inviare ciclicamente un pacchetto contenente il loro ID. Sono connessi tramite cavi USB a diversi hub dai quali ricevono l'alimentazione di tensione, tutti orientati nello stesso modo. Disposti sotto la pedana in otto file di otto dispositivi ciascuna, realizzano la Wireless Sensor Network.

4) Base Station: La BASE STATION altro non fa che ascoltare il traffico radio e selezionare i soli pacchetti spediti dal NODO MOBILE ignorando quelli prodotti dalla *WSN*. In questo modo si collega direttamente il nodo, di cui si vuole fare la localizzazione, al computer su cui è installato il software.

5) Teseo: L'algoritmo utilizzato per la 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, 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:

1) $P_r(d)$: potenza ricevuta dal nodo mobile in *[dBm]*: questa notazione sarà mantenuta per tutto questo articolo e denota la potenza in decibel di milliwatt;
2) $P_0(d_0)$: potenza di riferimento ad una data distanza d_0 dal trasmettitore sempre espressa in *[dBm]*;
3) 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;
4) 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.

Il software TESEO si presenta con l'interfaccia grafica presentata in 11. A differenza del TESEO originale, in questa versione da noi ritoccata manca la grafica riguardante la mappa aggiornata in tempo reale del processo in corso; in effetti sono state apportate alcune modifiche al software originale, prima fra tutte la scrittura su file in tempo reale di ogni stima elaborata così da rendere accessibile a altri applicativi, nello specifico a Matlab, l'utilizzo immediato delle misure di posizione. La scelta di eliminare la grafica dell'interfaccia nasce dall'intento di alleggerire il più possibile il carico al processore per poter usufruire al contempo, sul medesimo computer, entrambi gli applicativi.

Fig. 11. *interfaccia* TESEO

A. Interconnessioni ed elaboratori

A differenza delle prove simulative in Matlab, passando alla prova in laboratorio si incontrano alcune difficoltà pratiche: la prima consiste nel trovare un unico sistema operativo che permetta l'utilizzo di tutti i software necessari; la scelta non è affatto semplice poiché, per quanto riguarda l'*e-puck*, le librerie utilizzate fino ad oggi sono tutte scritte per ambiente WINDOWS, il materiale riguardante il TINYOS presente in laboratorio è stato pensato per ambienti LINUX, mentre il TESEO sembra essere l'unico che, a meno di piccoli accorgimenti, sia compatibile con entrambe le piattaforme. Per limitare il tempo necessario al reperimento e apprendimento di nuovi applicativi si è scelto di utilizzare due elaboratori diversi con due sistemi operativi differenti e sfruttare un software disponibile sul sito del TINYOS, il SERIALFORWARDER, per instradare i dati presenti porta USB (ascoltati quindi dalla BASE STATION) su rete ethernet così da poter essere letti da un computer collegato alla stessa rete.

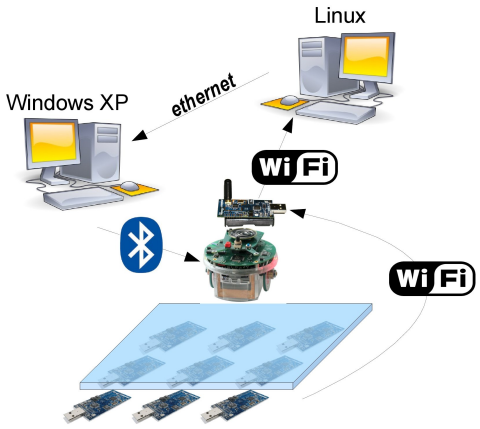

Fig. 12. *schema connessioni*

VI. Analisi dei dati forniti da Teseo

Prima di adattare passo passo il materiale delle simulazioni in Matlab per le prove in laboratorio, si è pensato di raccogliere una serie di dati sul funzionamento del Teseo per valutare l'attendibilità e la veridicità delle stime fornite. Una prima prova consiste nel dislocare il Nodo Mobile in alcune posizioni specifiche della piastra verificando le misure restituite dalla localizzazione; il grafico 13 mostra le misure relative agli angoli della piastra mentre il 14 rappresenta le stime del Nodo Mobile posizionato su coordinate precise, segnalate con un disco blu. In entrambi si rappresentano con colori differenti, serie due serie di misure relative alle stesse posizioni ma rilevate in giorni differenti.

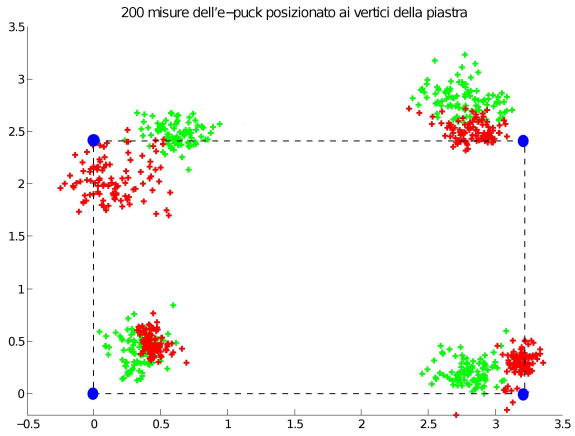

Fig. 13. *stime* Teseo *con* e-puck *posizionato negli angoli della piastra*

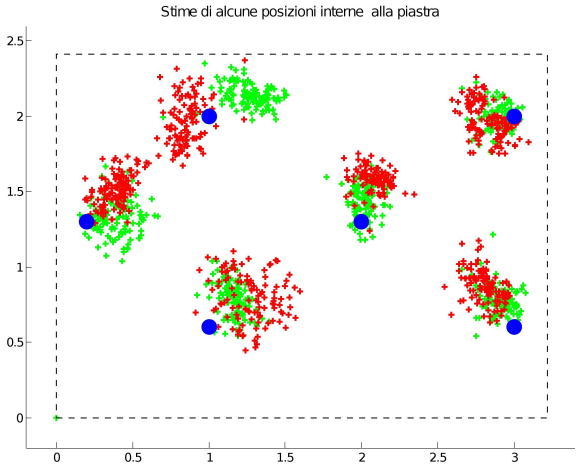

Fig. 14. *alcuni espempi di stime fornite da* Teseo

Una semplice raccolta dati come questa mette in luce chiaramente i primi lati deboli del sistema di localizzazione: le stime fornite da Teseo sono sempre affette da un rumore intrinseco e da un offset.

1) L'offset di misura: Si è cercato di capirne la natura ma non si è arrivati a grossi risultati: sembra dipendere principalmente dalla zona della piastra su cui si lavora e subire l'influenza di fattori casuali non ben conosciuti. Infatti, nelle varie prove, si è visto che alcuni settori di piastra, come la metà superiore, funzionano abbastanza correttamente e presentano offset solitamente trascurabili; altre sezioni, come la metà inferiore e in particolare la zona in basso a destra, presentano una maggiore aleatorietà nell'entità del gap di misura che arriva a massimi di 60 centimetri.

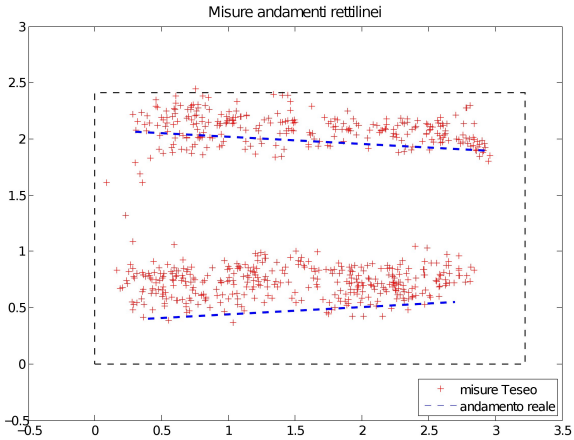

Fig. 15. *Misure fornite da* Teseo *di due tratti rettilinei percorsi dall'*e-puck *alla velocità di 200 tick/s*

Le stime non risultano ben distribuite sulla piastra ma, generalmente, traslate verso l'alto rispetto alle coordinate reali. Da un lato ci si aspettava che lungo i bordi la stima non fosse buona, questo a causa del minor numero di Nodi Àncora posti nelle vicinanze e quindi per una asimmetria intrinseca nella misura, ma non ci si spiega il motivo per cui l'errore non tende a centrare le misure sulla piastra ma piuttosto a traslarle verso l'alto. Questa serie di fenomeni citati fino a questo punto è ben visibile nelle immagini sopra riportate; infatti le due serie di misure sono state effettuate in giorni diversi e mostrano un discreto cambiamento nel comportamento della localizzazione; la piastra non si comporta in maniera uniforme e mentre in alcune zone la funzionalità si mantienee discreta, in alcuni punti si osservano variazioni apprezzabili nell'entità dell'offset; si osservi, invece, come la direzione di quest'ultimo rimanga normalmente fedele nel tempo. Tra le cause di questo fastidioso comportamento si esclude la conformazione del luogo in cui si effettuano i test visto che in questo caso, il laboratorio non ha modificato aspetto e mobilio nel corso delle prove e che il numero e posizione delle persone presenti all'interno della stanza erano circa le stesse. Anche i Nodi Ancora non sono stati mossi da sotto la piastra e si sono utilizzati sempre lo stesso *e-puck*, Nodo Mobile e Base Station. A questo punto restano in gioco le variabili ambientali come l'umidità, capace di incrementare l'influenza di scattering, diffrazione e riflessione, e quelle imputabili alla stabilità dell'algoritmo su cui si basa Teseo e dell'hardware utilizzato.

2) L'errore di misura: Per quanto riguarda il rumore che affligge le misure, se il campione considerato è sufficientemente consistente, almeno un centinaio di misure, lo si può facilmente schematizzare con una distribuzione gaussiana di

varianza $\sigma = 0.1$ circa (dipendente, per i motivi sopra spiegati, anche dal luogo della piastra in cui si esegue il test). In 16 e 17 vengono riportate le densità delle misure rispettivamente lungo le ascisse e le ordinate di 1300 stime registrate.

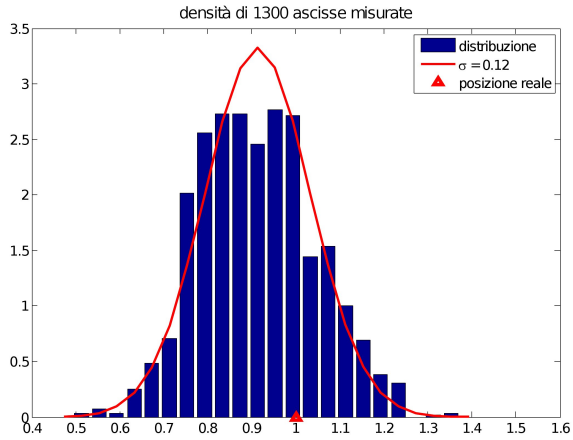

Fig. 16. *densità delle ascisse stimate del* e-puck *posizionato in x = 1 m*

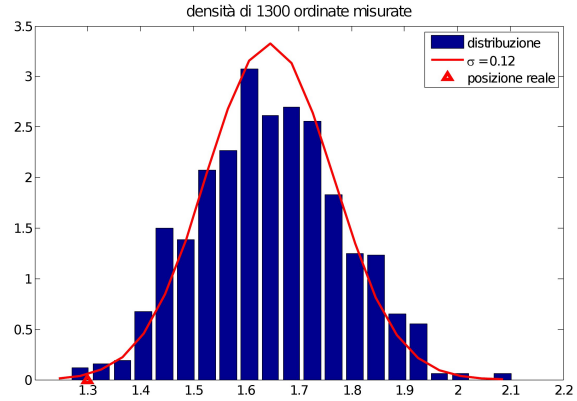

Fig. 17. *densità delle ordinate stimate del* e-puck *posizionato in y = 1.3 m*

Questi dati si riferiscono a misure completamente incorrelate dal modello dell'*e-puck*, o dal sistema in uso; sono ottenute semplicemmete lasciando fermo in una determinata posizione il NODO MOBILE sopra la piasra.

Se consideriamo un numero limitato di misure, come accade comunemente nel momento dell'inizializzazione, ove si considerano le ultime 15-20 misure fatte al robot fermo e se ne calcola la media, ci si troverebbe di fronte a una situazione come quella in figura 18 in cui è ben visibile la posizione reale del robot (rombo blu) e la media delle misure (rombo verde). L'errore in cui si incorre in questo caso è di circa 35 centimetri, oltre sette volte il diametro dell'*e-puck*.

Si è giunti alla conclusione che fosse il caso di simulare l'errore caratteristico delle misure fornite dal TESEO con una gaussiana di varianza maggiore rispetto a quella di figura 16 e 17; di qui la scelta descritta in `VII - Simulazione del filtro di Kalman Esteso` di utilizzare $\sigma = 0.3$ *cm*.

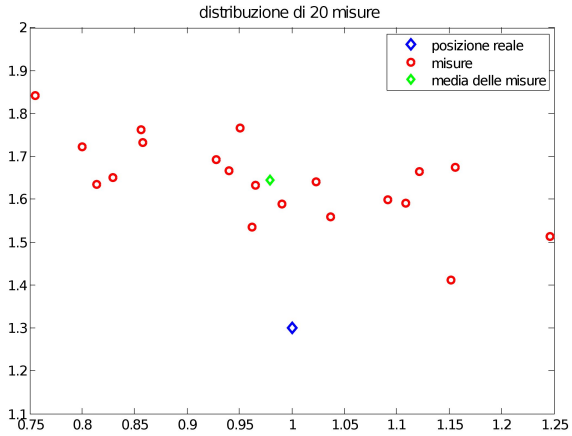

Fig. 18. distribuzione nel piano di alcune misure

VII. FILTRO DI KALMAN ESTESO

Per migliorare la precisione delle misure fornite dal TESEO si è utilizzato il filtro di Kalman esteso di cui si fa una dettagliata descrizione in appendice $B-C$; il principio fondamentale su cui si basa è di linearizzare ad ogni istante di campionamento le equazioni dinamiche attorno alla miglior stima di $x(t)$ disponibile a quell'istante. Tale scelta è giustificata dal verificarsi delle ipotesi fondamentali:

- il rumore di modello e di misura sono rumori bianchi di media nulla e varianza semidefinita e definita positiva rispettivamente;
- il rumore di modello e di misura sono scorrelati tra loro e dallo stato iniziale $x(0)$

Ci si occupa ora di mettere in pratica l'algoritmo del filtro in esame.

Le equazioni di aggiornamento, ottenute con la discretizzazione di *Runge-Kutta* del 2^o ordine sono le seguenti:

$$\begin{cases} \mathbf{x}(k+1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}(k) + \\ + \begin{bmatrix} T_c \cos(\theta(k) + \frac{u_2(k)T_c}{2}) & 0 \\ T_c \sin(\theta(k) + \frac{u_2(k)T_c}{2}) & 0 \\ 0 & T_c \end{bmatrix} \mathbf{u}(k) \\ + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{v}(k) \\ \mathbf{y}(k) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{w}(k) \end{cases}$$

in cui le componenti del vettore di stato

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \theta \end{bmatrix}$$

rappresentano le misure della coordinata x, della coordinata y e dell'orientazione dell'uniciclo; i vettori

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_\theta \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

rappresentano i rumori di modello e di misura e infine

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

rappresenta il vettore velocità, le cui componenti u_1 e u_2, rappresentano rispettivamente le velocità lineare ed angolare. Esse sono legate alla lettura degli encoder sulla ruota destra Δ_{ψ_R} e ruota sinistra Δ_{ψ_L} tramite la relazione

$$\begin{aligned} u_1(k) \cdot T_c &= r \cdot \frac{\Delta_{\psi_R} + \Delta_{\psi_L}}{2} \\ u_2(k) \cdot T_c &= r \cdot \frac{\Delta_{\psi_R} - \Delta_{\psi_L}}{2d} \end{aligned}$$

dove r è il raggio delle ruote e d è la lunghezza del semiasse tra le ruote.
Applicando le definizioni risulta facile a questo punto calcolare le matrici da utilizzare nell' *EKF*

$$\begin{aligned} \hat{\Phi}(k|k) &= \tfrac{\partial f}{\partial x}|_{x=\hat{x}(k|k)} \\ &= \begin{bmatrix} 1 & 0 & -u_1(k)T_c \sin(\hat{\theta}(k|k) + \frac{u_2(k)T_c}{2}) \\ 0 & 1 & u_1(k)T_c \cos(\hat{\theta}(k|k) + \frac{u_2(k)T_c}{2}) \\ 0 & 0 & 1 \end{bmatrix} \\ \hat{H}(k+1|k) &= \tfrac{\partial h}{\partial x}|_{x=\hat{x}(k+1|k)} = \begin{bmatrix} 1\ 0\ 0 \\ 0\ 1\ 0 \end{bmatrix} \end{aligned}$$

Questo modello, pur fedele alla realtà del laboratorio in cui si hanno a disposizione solo le misure delle coordinata x e y, in fase di implementazione presenta qualche complicazione.
Se si progetta il filtro di Kalman esteso a partire da questo, avendo cioè accesso solo ad alcune componenti dello stato, a causa dell'elevato errore di misura le stime delle coordinate cartesiane risultano inutilizzabili. In figura 19 si presenta il risultato della stima, su un tratto lungo un metro, del filtro costruito con il suddetto modello.

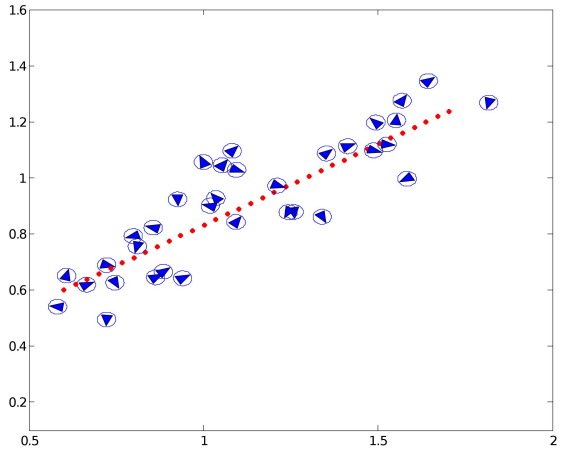

Fig. 19. *filtro di Kalman senza misura di θ*

Si rende necessaria quindi una modifica: la conoscenza dell'angolo è fondamentale per progettare un filtro accettabile, ma TESEO non ha accesso a questa misura che deve quindi essere trovata a partire dalla sola conoscenza delle coordinate x e y.
La soluzione che si è trovata è stata quella di far percorrere all'uniciclo un tratto rettilineo (che si suppone percorso ad angolo costante) e accettare come stima dell'orientamento del robot, l'angolo della retta di regressione sulle misure rilevate. Test di simulazione hanno dimostrato che è una scelta ragionevole in quanto si è rilevato un errore di circa 5 gradi rispetto all'angolo della retta effettivamente percorsa.
Si è quindi costruito un modello di questo tipo

$$\mathbf{y}(k) = \begin{bmatrix} 1\ 0\ 0 \\ 0\ 1\ 0 \\ 0\ 0\ 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \tilde{\theta} \end{bmatrix}$$

in cui x_1 e x_2 sono le misure delle coordinate spaziali rilevate dalla rete, mentre $\tilde{\theta}$ è l'angolo, calcolato a posteriori, della retta di regressione delle misure del tratto rettilineo percorso.

VIII. SIMULAZIONI DEL FILTRO DI KALMAN ESTESO

In questa sezione si analizzeranno i risultati ottenuti nella simulazione del filtro di Kalman applicato a traiettorie rettilinee le quali sono sembrate le più semplici e le più adatte per un sistema rumoroso quale il nostro. Come già detto in precedenza e come verrà discusso più avanti, la modellizzazione del rumore di misura con una gaussiana sembra abbastanza appropriata; nel filtro il rumore di misura verrà descritto da una normale di media zero e varianza 30cm. Tale scelta di varianza, come si può notare anche in figura 20 sembra essere ragionevole:

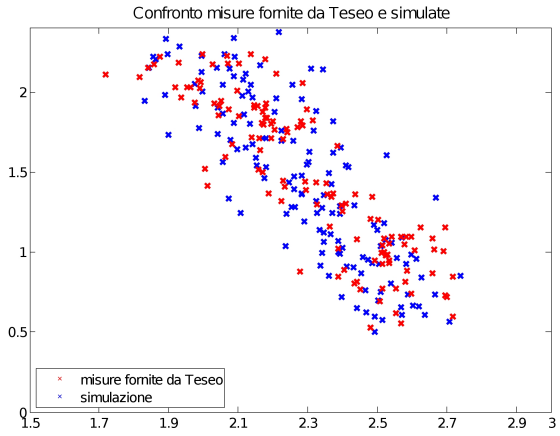

Fig. 20. *verifica varianza di rumore 30cm*

in un tratto rettilineo di circa due metri percorso dall'*e-puck* le misure ricevute dalla *WSN* (punti rossi in fig. 20) hanno la stessa dipersione di quelle ottenute simulando in Matlab (punti blu in fig. 20) un rumore di misura gaussiano di varianza 30cm.

Si è assunto come prima ipotesi la conoscenza esatta dell'orientazione dell'uniciclo. La suddetta ipotesi è resa accettabile se si posiziona l'*e-puck* sul piano in posizione casuale ma con orientazione nota 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.
Il periodo di acquisizione delle misure della *WSN* è stato fissato a 0.3 secondi, come è stato misurato in fase di test

sulla pedana in laboratorio.
Si procede ora all'analisi dei risultati ottenuti in simulazione grazie al software appena presentato; per tutte le simulazioni che seguiranno il tratto rosso costituisce la traiettoria realmente percorsa dell'*e-puck* mentre i triangoli sono le stime trovate applicando il filtro di Kalman esteso.

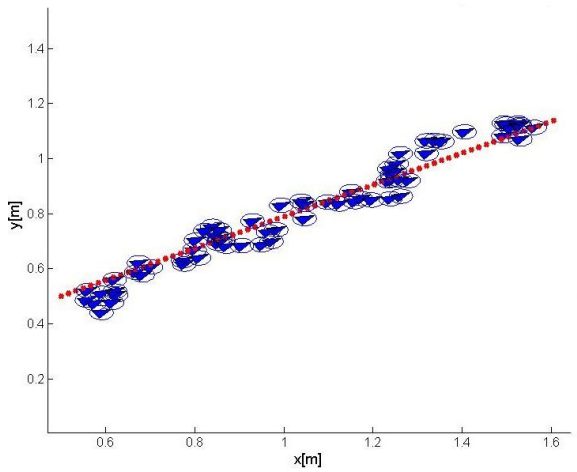

Fig. 21. *filtro di Kalman con conoscenza esatta di θ su un tratto rettilineo di un metro*

Come si nota dall'immagine 21, 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 $30cm$ a circa $6cm$ 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 $0.3sec$ ed una lunghezza del tracciato pari ad un metro.
Come già detto, la condizione ideale di conoscenza esatta dell'orientazione dell'uniciclo, è una condizione che non rispecchia la realtà che andremo ad affrontare in laboratorio e si rende necessario, per determinarla, l'uso della retta di regressione; come risulta chiaro dall'immagine 19, una volta tolta l'ipotesi di conoscenza di θ, 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 varianza di rumore di $30cm$ risulta troppo elevata per poter ricavare l'informazione dell'angolo con il solo filtro di Kalman.

La figura 22 mostra la stima dello stesso tratto rettilineo di figura 21 ottenuta sostituendo all'informazione esatta di θ quella di $\tilde{\theta}$ (angolo ottenuto grazie alla retta di regressione).
Confrontando le due varianze d'errore presentate in figura 23 risulta evidente che le prestazioni dei due filtri non differiscono più di tanto e quindi tale soluzione risulta soddisfacente.

A questo punto si richiede un precisazione importante sul

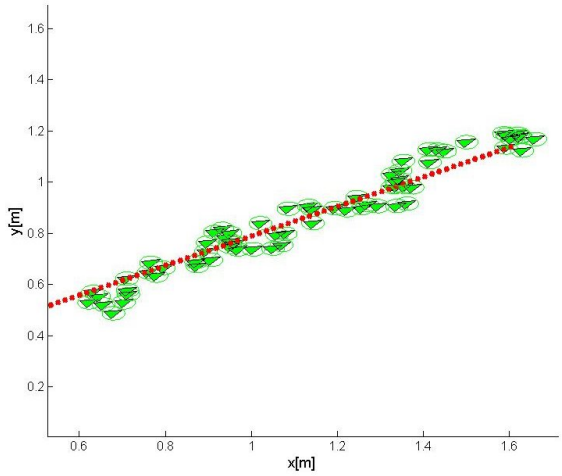

Fig. 22. *filtro di Kalman con $\tilde{\theta}$ stimato con la retta di regressione*

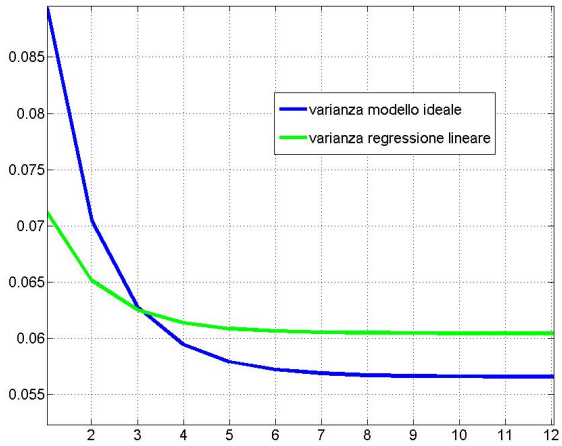

Fig. 23. *Confronto varianze*

filtro di Kalman esteso: il principio fondamentale su cui è basato l'*EKF* è la linearizzazione ad ogni istante di campionamento, delle equazioni dinamiche attorno alla miglior stima dello stato, disponibile a quel momento. Questo rende impossibile progettare il sistema di controllo separatamente dallo stimatore perchè, dopo ogni tratto rettilineo, quando si richiede di ricalcolare un nuovo ingresso di controllo, è necessario ricalcolare ogni volta la stima dello stato che però, per elaborare correttamente i dati di misura ed aver accesso all'informazione sull'angolazione $\tilde{\theta}$, deve necessariamente essere fatta a posteriori, dopo cioè che il tratto rettilineo precedente è stato percorso e quindi avendo a disposizione l'ingresso di controllo precedente per poter ricalcolare ogni volta le matrici dell'*EKF*. La matrice $P(t)$ non è quindi indipendente dall'ingresso di controllo $u(t)$ e quindi non vale il principio di separazione.

IX. Fase di inizializzazione

Questa fase risulta fondamentale per la determinazione iniziale dell'orientazione dell'uniciclo; come si può intuire un angolo stimato in modo soddisfacente fin dalla fase iniziale migliora anche le decisioni del controllo nelle fasi successive e di conseguenza le traiettorie saranno più performanti.
Come si può vedere in figura 24, la tecnica adottata è molto semplice: una volta che la rete di rilevazione acquisisce un numero sufficiente di misure (15-20 misure) della posizione iniziale, si fa muovere l'*e-puck* in linea retta per un tratto sufficientemente lungo (70 cm), successivamente si calcola la retta tra il punto medio della nuvola iniziale (punti blu in figura 24) ed il punto medio di quella finale (punti neri in figura 24); l'angolo di quest'ultima, fornisce l'orientazione dell'uniciclo.

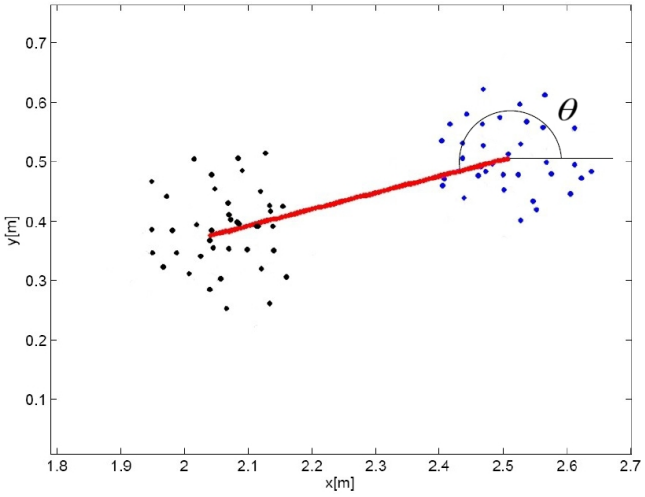

Fig. 24. *fase di inizializzazione*

Si noti come la retta percorsa per la determinazione dell'angolo iniziale può essere effettuata a velocità massima dato che le misure tra le due nuvole di punti non sono interessanti e possono essere ignorate.
Un'ulteriore step si è fatto simulando i sensori di prossimità dell'uniciclo *e-puck*, quest'ultimi, come si vede in figura 9 sono 8 e disposti uniformemente sulla superficie laterale dell'uniciclo. Quando uno di questi percepisce un ostacolo fa interrompere istantaneamente il moto dell'uniciclo a cui viene applicato una rotazione di novanta gradi; a questo punto una nuova fase di inizializzazione viene intrapresa: il punto d'arresto diviene il nuovo stato iniziale. Questo procedimento si reitera fintanto che l'*e-puck* non ha percorso i 70 centimetri prefissati; solo allora viene eseguito il calcolo dell'angolo.
La figura 25 illustra questo procedimento e, per completezza, si riporta in figura 26 un diagramma di flusso del principio di funzionamento dell'intera fase di inizializzazione.

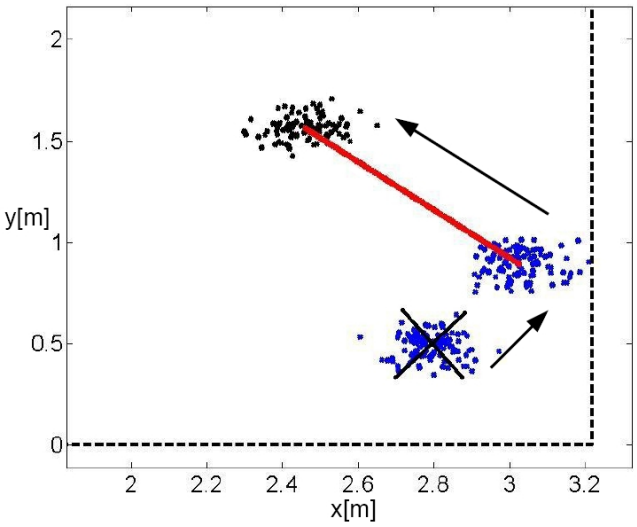

Fig. 25. *fase di inizializzazione con controllo di bordo*

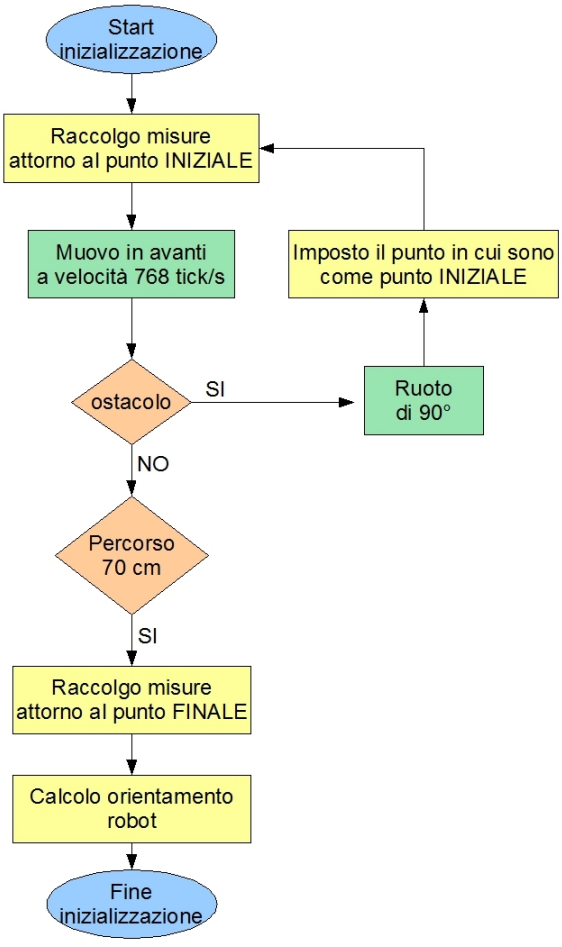

Fig. 26. *Diagramma di flusso dell'inizializzazione*

X. CONTROLLO

Terminata la fase di inizializzazione, si conosce l'orientazione del robot e si può quindi procedere a controllare lo stesso, in modo che raggiunga il target desiderato. La tecnica di controllo scelta è stata la più semplice, ovvero quella di comandare l'uniciclo in modo che si muova sempre per linee rette e a velocità costante; tale scelta si giustifica considerando l'elevata varianza delle misure fornite dalla rete di rilevazione che non permette di eseguire il tracking di traiettorie curvilinee a meno che non siano ad ampio raggio; inoltre la direzione dell'uniciclo è la variabile affetta dal maggior errore e quindi se ne può ridurre l'incidenza limitandone il più possibile le variazioni mantendo perciò una traiettoria ad angolazione costante. Inoltre, almeno in prima battuta, si è deciso di ignorare la possibile presenza di ostacoli sul tracciato. Il problema più difficile che si è affrontato è stato quello di determinare l'orientazione dell' *e-puck* poichè, come spiegato nelle sezioni precedenti, non è possibile utilizzare il filtro *EKF* con le sole misure di TESEO; ma è necessario costruire un modello con stato aumentato. É fondamentale, per avere risultati utilizzabili, che il vettore contenente le misure abbia anche l'informazione sull'angolo θ di orientazione del robot. In fase di simulazione, avendo a diposizione il modello matematico dell'uniciclo, si avrebbe facilmente accesso a questo dato ma questa situazione non è certo adattabile alla realtà del laboratorio.La soluzione trovata (che è la stessa poi utilizzata in laboratorio) è stata quella di lasciar percorrere al robot un tratto rettilineo, di lunghezza appropriata, in catena aperta e successivamente calcolare la retta di regressione sulle misure. A questo punto si aumenta la dimensione del vettore delle misure aggiungendo l'informazione sull'angolo di tale retta e quindi, a posteriori, si calcola la stima della posizione del robot ed è su questo dato che si effettua il controllo.Si noti come tale soluzione è anche quella a tempo minimo: avendo scelto un controllo a velocità costante, i tratti rettilinei, che esprimono la minor distanza tra due punti, sono quelli che minimizzano il tempo di percorrenza. L'algoritmo è il seguente:

1) A partire dalla stima dello stato e conoscendo la posizione di arrivo, si calcola l'angolo della traiettoria rettilinea che l'*e-puck* deve percorrere, tramite elementari formule di geometria.
2) Si fa ruotare il robot in modo che si orienti secondo la direzione appena calcolata.
3) Si comanda il robot in modo che percorra il tratto rettilineo della lunghezza desiderata. Questa deve essere scelta in maniera tale da avere un numero di misure sufficienti a determinare una stima corretta ma abbastanza corta da permettere un controllo frequente; infatti durante questo movimento si lavora in catena aperta. Con alcune prove si è stabilito che un tratto di 50 cm fosse un buon compromesso.
4) Dopo il tratto rettilineo si ferma il robot e si calcola l'angolo della retta di regressione delle misure.
5) Si aggiorna il vettore delle misure, aumentandone la dimensione con la misura dell'angolo così determinata.
6) Con il vettore delle misure così modificato si aggiorna la stima dello stato grazie alla funzione Matlab che implementa il filtro *EKF*.

A questo punto si cicla, ripetendo i 6 punti descritti, fino a che le coordinate dello stato stimato non si trovano all'interno di un'area di tolleranza (scelta come una circonfernza di diametro pari alla varianza dell'errore di misura) attorno al punto di arrivo.In fase simulativa si potrebbe in realtà diminuire il raggio di tale cironferenza vista la bassa varianza d'errore di stima; si è però scelto di eseguire le simulazioni mantenendo quella tolleranza perchè è la più adatta alla realtà del laboratorio in cui è preponderante l'incidenza dell'offset di misura non modelizzabile in simulazione in quanto è un fattore variabile nel tempo. Le figure 27 e 28 sono due esempi di simulazioni complete con due punti di arrivo e di partenza differenti; il *target* arancione è il centro della circonferenza di tolleranza di raggio 15cm (diametro pari alla varianza del rumore di misura), i segmenti rossi rappresentano la traiettoria realmente compiuta dall'*e-puck* mentre i triangoli blu sono le stime del filtro di Kalman.

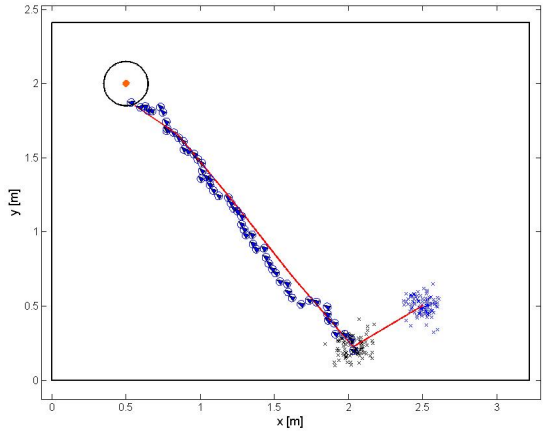

Fig. 27. *simulazione*

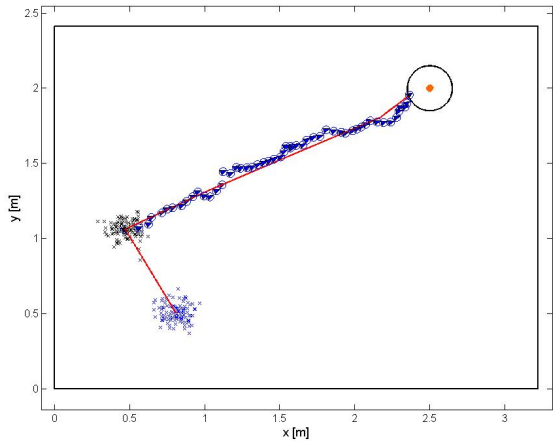

Fig. 28. *seconda simulazione*

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

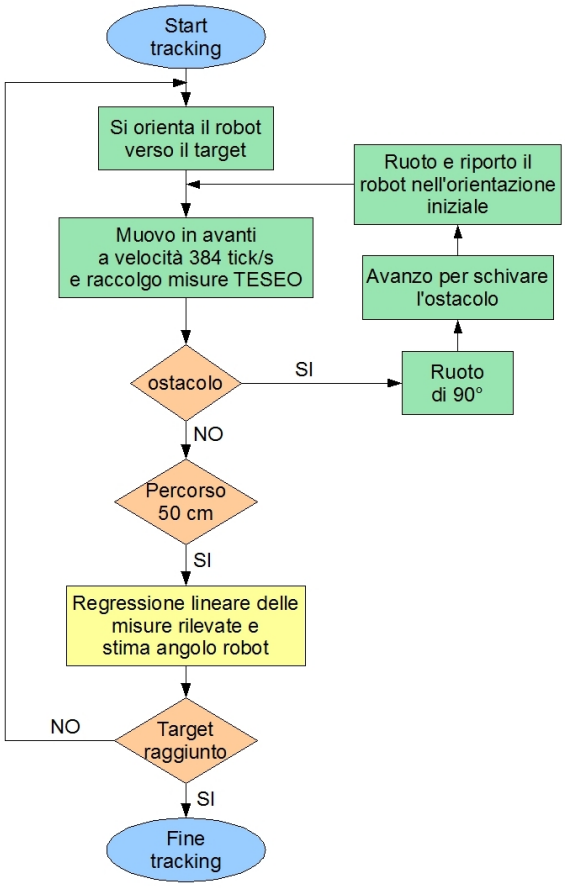

Fig. 29. *Flusso decisionale*

XI. Fase sperimentale in Navlab

Come già accennato, i primi sforzi in laboratorio sono stati fatti per interfacciare TESEO con Matlab pensando di poter facilitare poi la parte relativa al controllo sfruttando la nostra maggior dimestichezza con l'applicativo scientifico piuttosto che JAVA. Non disponendo di calcolatori di ultima generazione, la paura iniziale era di non riuscire a far funzionare NETBEANS, con il software di localizzazione, e Matlab, con la parte relativa al controllo, contemporaneamente sulla stessa macchina; con una serie di cicli temporizzati e minimizzando le risorse video utilizzate si è riusciti a bilanciare correttamente le due suit sorpassando il primo ostacolo. A questo punto le attenzioni sono andate alla localizzazione dell'*e-puck*: obiettivo principale era infatti posizionarlo in un punto qualsiasi della piastra e stimarne correttamente lo stato iniziale. Si dovrebbero avere ora a disposizione tutti i dati necessari al tracking verso il target prefissato. Come già spiegato, l'idea è di ripetere ciclicamente due eventi consecutivi: un primo movimento rettilineo di lunghezza variabile tra 30 e 50 centimetri in cui l'*e-puck* si muove in catena aperta e, a seguire, una nuova stima dell'angolo con conseguente nuova correzione della traiettoria. Questi due eventi si ripetono fintanto che il robot, terminato un tratto rettilineo, si trova entro la circonferenza di tolleranza prefissata centrata nel target o verifica di averlo appena attraversato fermandosi appena fuori.

In realtà, nel tratto in cui il robot si muove autonoma-

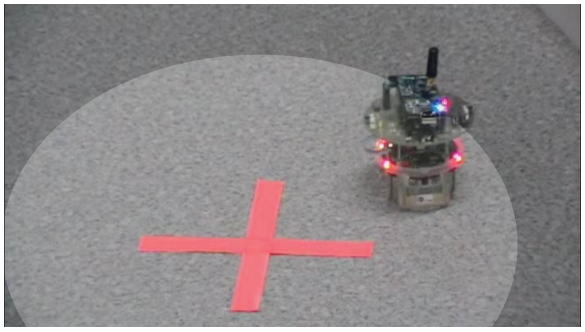

Fig. 30. *Arrivo dell'*e-puck *entro una certa tolleranza*

mente senza controllo, TESEO continua a stimare la posizione dell'uniciclo. Matlab calcola passo-passo il predittore di Kalman su quei dati così da avere sempre lo stato finale completo del robot. La velocità da imprimere alle ruote nel tratto rettilineo è stata scelta empiricamente con una serie di prove, come illustra 31, cercando quella più elevata che permettesse comunque di ottenere un numero di stime sufficienti per una discreta interpolazione, come spiegato al punto *3* del *controllo*.

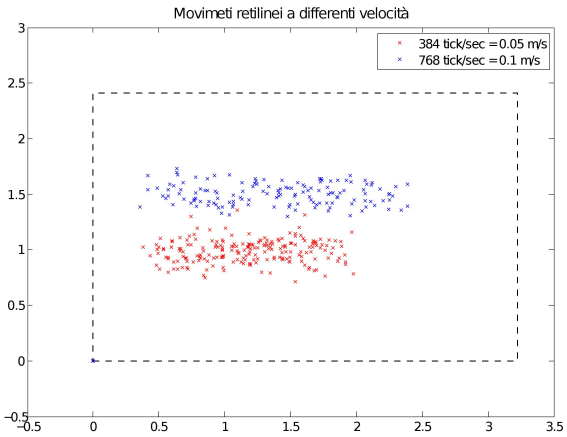

Fig. 31. *Prove rettilinee a diverse velocità*

Per muovere l'*e-puck* si sono utilizzate le librerie messe a disposizione dalla comunity all'indirizzo *http://www.e-puck.org*; due sono i comandi possibili per impostare la velocità delle ruote:

- `write(epuck1,'d,768,768')` permette di impostare con una stringa di tipo caratteri alcune proprietà del dispositivo; *d* indica le proprietà di velocità ed è seguito rispettivamente da quella relativa alla ruota destra e quella della ruota sinistra espresse in *tick al secondo*; con questa modalità però il robot si ferma all'arrivo dell'istruzione successiva, anche se è una semplice lettura degli encoder o dei sensori;
- `set(epuck1,'speed', [vl vl])` con il quale si imposta la proprietà *speed* dell'oggetto *epuck1*; affinché le impostazioni vengano passate all'*e-puck* è necessario l'uso del comando `update(epuck1)`; in questo caso la proprietà assegnata rimane tale fino alla prossima sua

modifica.

A. Gestione bordi ed ostacoli

In un secondo momento, solo dopo aver realizzato l'algoritmo di tracking punto-punto, si è passati allo studio della gestione dei bordi della piastra e di eventuali ostacoli lungo il tragitto verso il target prefissato. A tale scopo è stato necessario uilizzare i sensori di prossimità disposti tutto attorno all'*e-puck* e numerati da 0 a 7 e visibili nei quadrati rossi di figura 32.

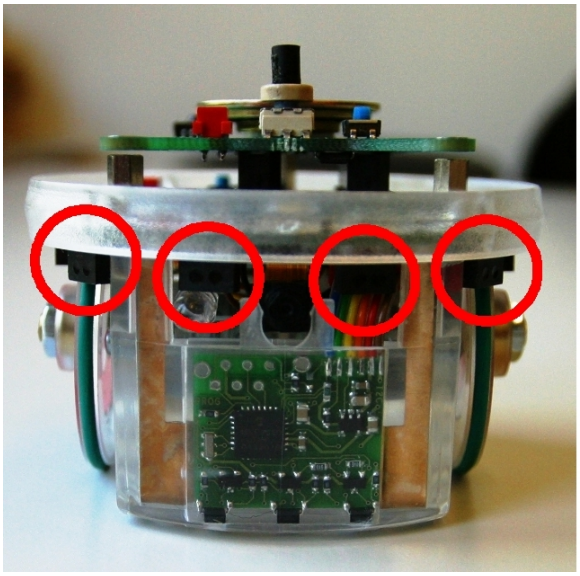

Fig. 32. *Sensori di prossimità*

Con il comando: `[val, up] = get(epuck1, 'proxi')` si va a leggere lo stato attuale delle varie proprietà; in questo caso *'proxi'* indica che siamo interessati ai sensori di prossimità. Il comando ritornerà nel vettore *val* lo stato degli otto rilevatori: avranno valore inversamente proporzionale alla distanza che intercorre tra loro e l'ostacolo; essendo anche questa lettura affetta da rumore con oscillazioni attorno al valore zero che corrisponde ad assenza di ostacoli, si è scelto come soglia il valore 70 che approssimativamente indica l'ostacolo a un centimetro e mezzo dal sensore. Si riesce a comandare in tempo l'arresto del *e-puck* e una modifica della traiettoria. Considerando che l'*e-puck* si muove solo in un verso, si sono effettuati i controlli sui sensori anteriori identificati da *0* e *7*. Con la lettura ciclica dei loro valori, non appena segnalano la presenza di ostacoli, si blocca il moto dell'uniciclo, lo si fa ruotare di novanta gradi e si effettua una localizzazione del punto in cui si è fermato; a questo punto si ricomincia il controllo come prima verso il *target* prefissato.

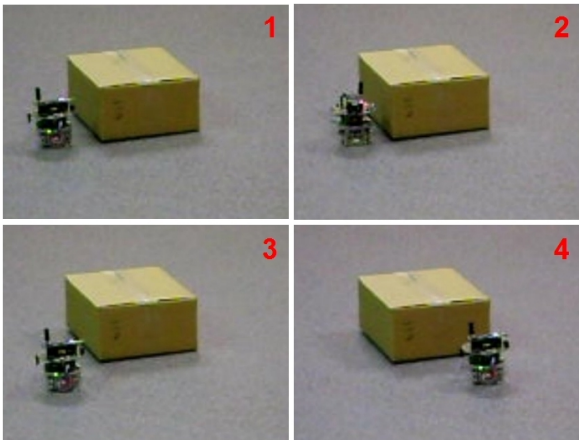

Fig. 33. *sequenza dimostrativa del superamento di un ostacolo*

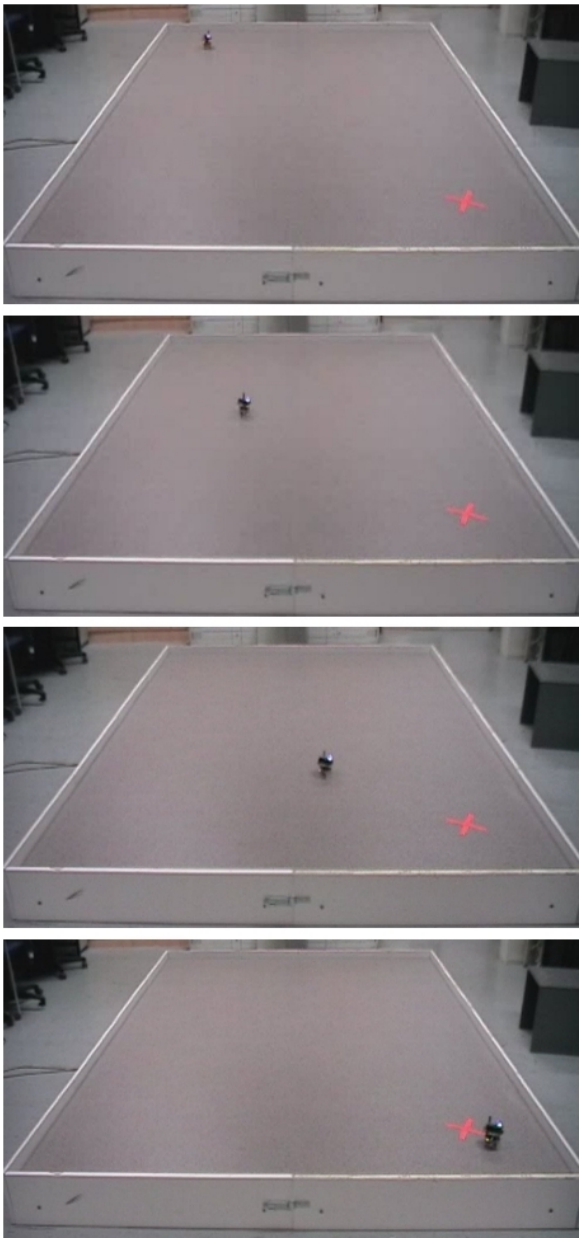

Fig. 34. *sequenza di una prova effettuata in laboratorio*

XII. CONCLUSIONI E SVILUPPI FUTURI

Al momento sono molte le applicazioni che possono richiedere un'automazione autonoma nel movimento; è passato alla ribalta in questi giorni il progetto *DustBot* [32] con la creazione di robot per la raccolta differenziata porta a porta con localizzazione basata su GPS e controllo degli ostacoli circostanti con sensori infrarossi; così come sono in aumento i progetti di *home automation* con particolare attenzione all'aiuto alle persone portatrici di handicap; altrettanto ambizioso è ridurre la presenza umana in ambienti industriali a rischio elevato sostituendo l'uomo con robot. Un campo sicuramente molto vasto che può fare i conti su di un gran numero di tecnologie. Nel corso del progetto si è fatto uso dello standard $IEEE802.15.4$ già molto decantato in letteratura; il punto dolente forse sono le complicazioni che affliggono le *WSN* in funzione della loro geografia e delle modifiche che il modello di campo subisce. Abbiamo accennato infatti alla necessità di uno studio più approfondito dei parametri che contraddistinguono il modello del canale della rete utilizzata. Questo porterebbe a delle diminuzioni degli errori e degli offset di misura migliorando il risultato finale: in simulazione è stata dimostrata la bontà del filtro di Kalman e la possibilità, in assenza di offset, di ridurre notevolmente l'area di tolleranza attorno all'obiettivo. Per gli obiettivi preposti non è sufficiente localizzare entro un'area di lavoro il robot, come poteva esserlo, invece, ai fini del monitoraggio di persone dentro edifici; è necessario invece ridurre al minimo ogni incertezza. Nonostante le numerose non idealità presenti in laboratorio, utilizzando metodi elementari, si è dimostrato possibile risalire alla posizione e orientazione del robot sulla piastra e raggiungere obiettivi prefissati anche con errori relativamente stretti. Si è tentata la gestione degli urti spostando l'attenzione alla sicurezza del movimento, aspetto importante se pensiamo il tutto applicato al servizio dell'uomo o di strumenti, magari costosi. Considerando la natura didattica dell'hardware utilizzato si possono ritenere soddisfacienti i risultati ottenuti sebbene i controlli appaiono macchinosi e, a volte, un po' lenti.

Si è descritta nell'articolo la semplificazione per cui si assume di far ruotare il robot su sé stesso mantenendo fermo il baricentro; rimane comunque da implementare la gestione delle rotazioni utilizzando un raggio di curvatura minimo. Si tratta comunque di alcune modifiche al codice che non modificano la natura dell'esperimento: si andrebbe infatti a lavorare su una diversa impostazione del moto dei motori passo passo che governano le ruote dell'uniciclo e non sulla struttura della localizzazione.

La possibilità di implementare la localizzazione e/o il controllo direttamente sul robot o sul *mote* in esso alloggiato è apparso un lavoro incompatibile con le tempistiche a nostra disposizione; potrà essere uno spunto per il futuro implementare una parte del controllo a più basso livello rispetto a come avviene ora o realizzare un software di localizzazione più snello del TESEO appositamente studiato per poter essere gestito dalla scarna architettura del *mote*.

Si era considerata la possibilità di un prefiltraggio delle misure fornite da TESEO ma dopo l'analisi effettuata sulle stesse si è deciso di procedere senza questo step; non era facile infatti trovare un metro con cui decidere se la misura al passo successivo era molto discosta dalla precedente a causa del rumore sovrapposto (trattandola quindi come *outlier*) o ritenerla attendibile come nuova posizione del robot, questo a causa della velocità del robot: non è infatti in grado di percorrere in pochi intervalli di tempo una distanza significativa rispetto alla dispersione dell'errore di misura. Un approcio al filtraggio si potrebbe pensare possibile se, anziché lavorare in un campo aperto, si pensasse di determinare delle zone percorribili (chiamiamole *strade*) entro le quali l'uniciclo può muoversi con la possibilità di considerare solo le misure che ricadono al loro interno.

APPENDIX A
DISCRETIZZAZIONE DEL MODELLO UNICICLO

La discretizzazione del modello è fatta con il metodo $Runge-Kutta$ del secondo ordine.
il modello a tempo continuo $\frac{dx_i}{dt} = f_i(t, x_1, \cdots, x_N)$ viene così discretizzato:

$$\mathbf{x}(k+1) = \mathbf{x}(k) + T_c \mathbf{f}\left[k + \frac{T_c}{2}, \mathbf{x}(k) + \frac{T_c}{2}\mathbf{f}(k, \mathbf{x}(k))\right]$$

Applicare la formula a questo punto risulta semplice; per chiarezza si illustrano i passaggi separatamente per ogni componente.
Per quanto riguarda x_1 si ha:

$$\begin{aligned} x_1(k+1) &= x_1(k) + T_c \cdot f_1(x_1 + \tfrac{T_c}{2} f_1(x_1, x_2, \theta), \\ &\quad , x_2 + \tfrac{T_c}{2} f_2(x_1, x_2, \theta), \theta + \tfrac{T_c}{2} f_3(x_1, x_2, \theta)) \\ &= x_1(k) + T_c \cdot u_1(k) \cos(\theta(k) + \tfrac{T_c \cdot u_2(k)}{2}) \end{aligned}$$

Analogamente per x_2 risulta:

$$\begin{aligned} x_2(k+1) &= x_2(k) + T_c \cdot f_2(x_1 + \tfrac{T_c}{2} f_1(x_1, x_2, \theta), \\ &\quad , x_2 + \tfrac{T_c}{2} f_2(x_1, x_2, \theta), \theta + \tfrac{T_c}{2} f_3(x_1, x_2, \theta)) \\ &= x_2(k) + T_c \cdot u_1(k) \sin(\theta(k) + \tfrac{T_c \cdot u_2(k)}{2}) \end{aligned}$$

Infine per θ:

$$\begin{aligned} \theta(k+1) &= \theta(k) + T_c \cdot f_3(x_1 + \tfrac{T_c}{2} f_1(x_1, x_2, \theta), \\ &\quad , x_2 + \tfrac{T_c}{2} f_2(x_1, x_2, \theta), \theta + \tfrac{T_c}{2} f_3(x_1, x_2, \theta)) \\ &= \theta(k) + T_c \cdot u_2(k) \end{aligned}$$

Passando alla più compatta notazione matriciale, si ottiene finalmente:

$$\begin{aligned} \mathbf{x}(k+1) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}(k) + \\ &+ \begin{bmatrix} T_c \cos(\theta(k) + \frac{u_2(k) T_c}{2}) & 0 \\ T_c \sin(\theta(k) + \frac{u_2(k) T_c}{2}) & 0 \\ 0 & T_c \end{bmatrix} \mathbf{u}(k) \end{aligned}$$

APPENDIX B
FILTRO DI KALMAN ESTESO

Il filtro di Kalman esteso è un algoritmo di impiego molto generale che risolve, seppur in modo approssimato, il problema di stima dello stato di un modello non lineare.
Come si evince da [14] la tecnica che si deve usare per derivare le equazioni dell'*EKF* è quella della linearizzazione. Questa tecnica è sicuramente la più ovvia e quella che più intuitivamente si presenta come soluzione per i problemi descritti da modelli non lineari. La chiave per spiegare il buon comportamento dell' EFK sta nella sua linearizzazione "intelligente" : il principio fondamentale su cui si basa è di linearizzare ad ogni istante di campionamento le equazioni dinamiche attorno alla miglior stima di $x(t)$ disponibile a quel momento.
La dinamica del modello a tempo discreto è la seguente:

$$\begin{aligned} x(k+1) &= f(k, x(k)) + v(k) \\ y(k) &= h(k, x(k)) + w(k) \end{aligned} \tag{2}$$

Ipotesi fondamentali, sotto le quali verranno poi ottenute le equazioni del filtro, sono:

- il rumore di modello $v(k)$ è un rumore bianco, di media nulla e varianza $Q = Q^T \geq 0$ (semidefinita positiva)
- il rumore di misura $w(k)$ è un rumore bianco, di varianza $R > 0$ (strettamente definita positiva)
- i rumori di modello e di misura sono scorrelati tra loro e dallo stato iniziale $x(0)$

APPENDIX C
ALGORITMO EKF

Per il modello non lineare descritto dalle equazioni dinamiche (2) gli stimatori $\hat{x}(k+1|k)$, $\hat{x}(k|k)$ e le rispettive varianze d'errore $P(k+1|k)$,$P(k|k)$ si possono calcolare ricorsivamente mediante il seguente algoritmo.

- Si risolve l'equazione differenziale a tempo continuo
$$\begin{aligned} \dot{x}(t) &= f(x(t)) \\ x(t_k) &= \hat{x}(k|k) \end{aligned} \tag{3}$$
nell'intervallo $[t_k, t_{k+1}]$ e si pone $\hat{x}(k+1|k) := x(t_{k+1})$
- Si definisce la matrice $\hat{\Phi}(k|k) = \frac{\partial f}{\partial x}|_{x=\hat{x}(k|k)}$
- Si calcola la varianza a priori
$P(k+1|k) = \hat{\Phi}(k|k) P(k|k) \hat{\Phi}(k|k)^T + Q$
- Si definisce la matrice $\hat{H}(k+1|k) = \frac{\partial h}{\partial x}|_{x=\hat{x}(k+1|k)}$
- Si definiscono le matrici
$$\begin{aligned} \Lambda(k+1) &= \hat{H}(k+1|k) P(k+1|k) \hat{H}(k+1|k)^T \\ &+ R \\ L(k+1) &= P(k+1|k) \hat{H}(k+1|k)^T \Lambda(k+1)^{-1} \end{aligned}$$
- Si calcola la stima a posteriori
$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + L(k+1)[y(k+1) - h(\hat{x}(k+1|k))]$
- Si calcola la varianza a posteriori
$$\begin{aligned} &P(k+1|k+1) = [I - L(k+1)\hat{H}(k+1|k)] \cdot \\ &\cdot P(k+1|k) \cdot [I - L(k+1)\hat{H}(k+1|k)] + \\ &+ L(k+1) R L(k+1)^T \end{aligned}$$
- Condizioni iniziali
$$\begin{aligned} \hat{x}(0|-1) &= E[x(0)] \\ P(0|-1) &= Var(x(0)) \end{aligned}$$

É importante notare come l'algoritmo appena descritto usi il modello linearizzato solo per il calcolo del guadagno $L(k)$ e delle matrici varianza di errore; mentre l'elaborazione dei dati di misura viene effettuata sulla base del modello non lineare originario. Gli stimatori $\hat{x}(k+1|k)$ e $\hat{x}(k|k)$ sono quindi essi stessi funzioni non lineari dei dati di misura. Per il calcolo del guadagno $L(k+1)$ infine, va sottolineato come si debbano usare le matrici $\hat{\Phi}(k|k)$ e $\hat{H}(k+1|k)$ che devono essere ricalcolate ad ogni passo di campionamento in corrispondenza alle stime correnti $\hat{x}(k|k)$ e $k+\hat{1}|k$: la successione dei guadagni $\{L(k)\}$ non puó quindi essere calcolata fuori linea.

APPENDIX D
TESEO

La tecnica che si effettua per localizzare in un punto della mappa la posizione del nodo mobile è denominata *triangolazione* (in questo caso particolare si chiama *trilaterazione* poiché 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 *ancora*, 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 nota 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, 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. Nel caso in cui 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 *ancora* 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 [14]):

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

. Questa tecnica è stata descritta poiché 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 poiché 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 [14] e sviluppato per questa applicazione in [8], la distanza stimata si calcola come:

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

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.:

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

in cui:

1) $\hat{d}_{i,j}^2$: 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 *ancora* 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.

REFERENCES

[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] Sartore F., Sassaro A., Vettori D., *Localizzazione e tracking a tempo minimo mediante rete di sensori wireless WSN*, 2008

[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] G. Cosi, *Controllo del Khepera con localizzazione tramite rete di sensori wireless Tesi di laurea triennale*, Università degli studi di Padova, Marzo 2007.

[10] Oriolo G., De Luca A., Venditelli M., *WMR Control Via Dynamic Feedback Linearization: Design, Implementation, abd Experimental Validation*, 2002 IEEE [1063-6536/02]

[11] Klancar G., Skrjanc I., *Tracking-error model-based predictive control for mobile robots in real time*, 2007

[12] Dinamica e controllo dei veicoli robotici, *Centro Interdipartimentale Enrico Piaggio", Università degli Studi di Pisa*

[13] Barbera G., Calore P., Cosi G. *Approccio comportamentale al controllo coordinato di WMR* 2008

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[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. Bertinato, G. Ortolan, F. Maran, R. Marcon, A. Marcassa, F. Zanella, P. Zambotto, L. Schenato, A. Cenedese, *RF Localization and tracking of mobile nodes in Wireless Sensors Networks: Architectures, Algorithms and Experiments*, DEI, Department of Information Engineering, University of Padua, Italy

[32] web: http://www.dustbot.org (sito del progetto Dustbot)