UNIVERSITÀ DI BRESCIA
FACOLTÀ DI INGEGNERIA
Dipartimento di Elettronica per l'Automazione

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica Mobile
(Prof. Riccardo Cassinis)

Docking di Speedy con marker attivo

Elaborato di esame di:

Luca Milani, Matteo Berta

Consegnato il: **26 Giugno 2009**

Sommario

Il lavoro svolto consiste nella scrittura di un programma che controlla e guida il robot speedy verso la sua stazione di ricarica. Per l'individuazione della stazione è stato usato un marker attivo, composto da sei led rossi, montato sulla stazione di docking. Il programma scritto comunica con un altro software, sviluppato da un secondo gruppo di lavoro[1], il quale interagisce con la webcam montata sul robot

1. Introduzione

Inizialmente viene presentato il robot Speedy attraverso una breve descrizione delle sue caratteristiche peculiari, e delle modifiche apportate per poter essere inserito nel contesto presentato. Segue una breve spiegazione del docking di un robot mobile.

1.1. Il robot Speedy

Il robot denominato "Speedy" è il modello Pioneer 1 commercializzato da ActivMedia Robotics. Si tratta di un robot con architettura differential drive, con due ruote motrici fisse ed indipendenti ed una ruota folle e pivottante. Sulla parte superiore del robot è appoggiato un calcolatore portatile sul quale è installato il pacchetto dei software utilizzati per programmare la movimentazione del robot. Calcolatore e robot comunicano attraverso un cavo seriale con protocollo RS232. Questo modello dispone di sette sonar disposti sulla parte anteriore che permettono di rilevare la presenza di ostacoli, fino ad una distanza massima di circa 4 m, su un intervallo poco più ampio da -90° a 90°. Sulla parte frontale del robot è montata una webcam utilizzata per la discriminazione dei marker attivi.

Figura 1. (a) Il robot Speedy. (b) Webcam e sonar

1.2. Il docking

Il docking nel campo della robotica mobile è inteso come la capacità di un robot di individuare e raggiungere la propria stazione di docking. Cioè la postazione adibita per la lunga sosta del robot e per l'indispensabile ricarica delle batterie. Ci sono innumerevoli tecniche per implementare questa abilità, innanzitutto il robot deve essere in grado di conoscere la posizione della stazione da raggiungere. Successivamente deve potere raggiungere la stazione di ricarica entrandoci nel modo appropriato, in funzione di come è stata costruita. Una volta raggiunta la stazione il robot deve riconoscere di essere arrivato e spegnersi se necessario.

2. Il problema affrontato

Come descritto nel paragrafo precedente per consentire a un robot mobile di potersi ricaricare autonomamente all'interno della propria stazione di ricarica è necessario che esso sia in grado di raggiungerla. Per far ciò è possibile sfruttare diversi tipi di conoscenza, nel nostro caso si è deciso di utilizzare le informazioni fornite dalla webcam posta nella zona frontale del robot. Subito alle spalle della stazione di ricarica, infatti, è posizionato un Landmark attivo facilmente individuabile e distinguibile, composto da sei LED rossi ad alta luminosità disposti in modo da comporre un pattern regolare sulla superficie di un box di materiale plastico. Grazie al marker appena descritto il robot è quindi in grado di identificare la propria stazione di ricarica. Segue ora una breve descrizione della stazione di ricarica.

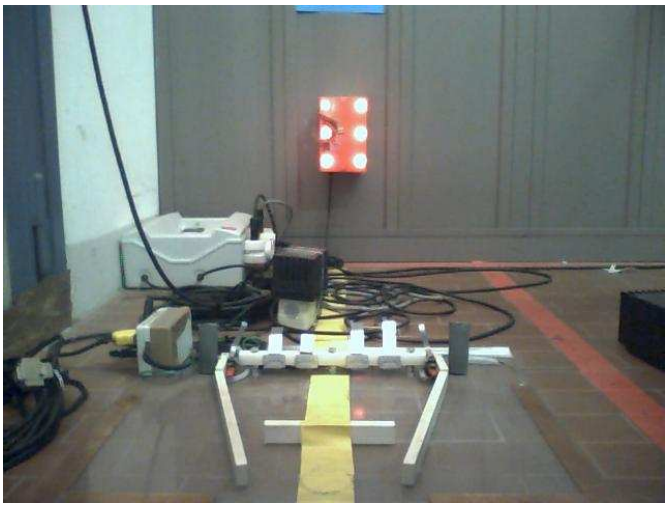

Fig.2 stazione di ricarica con il Landmark.

La disposizione delle guide fissate sulla base permette al robot di arrivare alla stazione con un sufficiente margine di errore, conducendolo fino al corretto posizionamento finale. La stazione di ricarica è completata dalla struttura dei contatti per la ricarica delle batterie e da due squadrette di fine corsa. Tali squadrette, utilizzate per fermare il robot nella posizione desiderata, bloccano le ruote mettendo in stallo i motori. Inoltre una volta che il robot raggiunge entrambi i fine corsa viene spenta l'alimentazione del marker attivo. Grazie a queste due condizioni è possibile capire se la stazione di ricarica è stata raggiunta.

2.1. Il software di visione e "shared memory"

Dovendo elaborare i dati forniti dalla webcam è necessaria la presenza di un software per il riconoscimento del Landmark. Tale software, sviluppato da un secondo gruppo di lavoro[1], è in grado di rilevare la presenza del Landmark una volta che esso è entrato nel campo visivo della webcam. Una volta riconosciuto il marker è stato necessario mettere in comunicazione i due processi, in modo che i dati reperiti dalla webcam potessero essere utilizzati dal software atto alla movimentazione del robot. A tal fine è stato deciso di utilizzare il meccanismo di "shared memory" protetto mediante un semaforo: tale soluzione permette, infatti, sia di sovrascrivere comodamente il contenuto della memoria condivisa sia di ottimizzare l'occupazione delle risorse. Per l'intera comunicazione, infatti, il software di visione non fa nient'altro che riempire un vettore di 14 interi: le prime 12 posizioni contengono le coordinate presunte del marker, la posizione 13 il valore booleano di freschezza dei dati e la posizione 14 l'indicatore, sempre booleano, di terminazione dell'algoritmo. Entrambi i processi hanno accesso alla memoria condivisa sia in lettura sia in scrittura: così facendo il programma lettore, una volta ottenuto il "lock" sul semaforo, è in grado di segnalare l'avvenuto consumo dei dati ponendo a false la posizione 13 dell'array. Con un accordo di entrambi i gruppi di lavoro si è deciso che se il programma di visione non individua il Landmark i dati contenuti nella memoria condivisa vengono forzati a zero. In questo modo il processo di

docking è in grado di sapere se il Landmark è all'interno del campo visivo della webcam. La memoria condivisa è un array composto da 14 interi, strutturata nel seguente modo:

ARRAY[0], ARRAY[1] : coordinate X e Y del LED 1.

ARRAY[2], ARRAY[3] : coordinate X e Y del LED 2.

ARRAY[4], ARRAY[5] : coordinate X e Y del LED 3.

ARRAY[6], ARRAY[7] : coordinate X e Y del LED 4.

ARRAY[8], ARRAY[9] : coordinate X e Y del LED 5.

ARRAY[10], ARRAY[11]: coordinate X e Y del LED 6

ARRAY[12] :flag di freschezza dei dati.

ARRAY[13] : flag di terminazione del programma

3. La soluzione adottata

3.1. Scelta della soluzione e modifiche del Landmark

Una volta sistemata la problematica della ricezione dei dati relativi l'individuazione del Landmark, il software deve sfruttare tali informazioni per il raggiungimento della stazione di ricarica. Di primo acchito si pensava che, una volta che il Landmark fosse entrato nel campo visivo della webcam, sarebbe bastato muovere il robot in modo da portare i sei led al centro dell'immagine, e infine, seguire tale traiettoria fino all'arrivo nella stazione di ricarica. Ci si è però ben presto accorti che tale soluzione non era accettabile. Infatti, data la morfologia dell'ambiente circostante e l'architettura della stazione di ricarica, il robot riusciva nel suo intento solamente quando si trovava in posizione frontale rispetto alla stazione. Da qualsiasi altra posizione, nonostante il Landmark fosse visibile, il robot non era in grado di entrare nella stazione di ricarica, rischiando spesso di andare a sbattere contro la stazione di ricarica vicina.

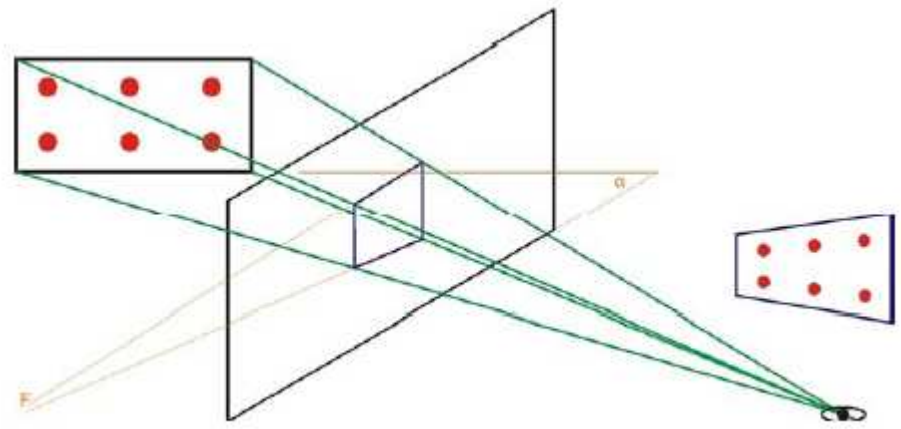

Fig.3 distorsione trapezioidale

Successivamente si è pensato sfruttare la distorsione trapezoidale dell'immagine del marker di forma nota per risalire alla posizione da cui è stato registrato lo scatto. In questo modo si sarebbe potuto determinare una traiettoria possibile per il raggiungimento della stazione. Tuttavia, a causa della bassa risoluzione della webcam, la deformazione prospettica del Landmark è inutilizzabile e non permette di risalire alla vera posizione del robot.

È quindi stato necessario ideare un'alternativa che permettesse di stabilire la posizione del robot. In primo luogo una possibile soluzione era stata identificata nel rendere i due led centrali del marker fonti luminose direzionali, incassando i due led nel box del marker e sovrapponendoci due tubicini. In questo modo le due luci centrali sarebbero state visibili alla videocamera solo se il robot si fosse trovato in un corridoio frontale al marker, condizione sufficiente per permettere l'arrivo nella stazione di ricarica. Il problema fondamentale di questa soluzione deriva dall'impossibilità di restringere l'ampiezza del corridoio nel quale i led centrali vengono visti e, quindi, avere un allineamento tra robot e marker sufficiente a guidarlo in modo appropriato verso la destinazione. Inoltre, osservando solo i quattro led angolari, è impossibile stabilire se il robot si trovi alla destra o alla sinistra del landmark e, di conseguenza, non è possibile decidere il movimento da compiere per portarlo all'interno del corridoio sopra descritto.

La necessità di discriminare il semipiano in cui si trova il robot rispetto al marker ha portato all'idea di non incassare i led centrali, ma lasciarli affiorare come gli altri dalla superficie. È così nata l'idea di applicare due barriere fisiche sporgenti per rendere i due led centrali alternativamente visibili da semipiani diversi, lasciandoli entrambi visibili solamente quando il robot fosse stato nella zona frontale alla docking station.

Fig.4 barriere opache

Come si vede in figura 4 la mezza luna di cartone oscura il led centrale sinistro se visto da una posizione laterale, mentre in figura 2 si può notare come la mezza luna di cartone non influenzi la visione del led se il robot si trova in posizione frontale. Con questa configurazione è possibile capire se il robot si trova a destra, a sinistra oppure in zona frontale rispetto al Landmark; inoltre, aumentando la lunghezza della mezza luna di cartone è stato possibile restringere l'ampiezza del corridoio sopra citato, permettendo quindi la visione del led centrale solamente quando il robot è correttamente allineato alla stazione di ricarica.

3.2. Soluzione implementata

Il nucleo della soluzione adottata consiste nel raggiungimento della stazione da parte del robot, nel caso in cui il robot riesca a vederla; ovvero che il marker attivo sia all'interno del campo visivo della webcam. Con questa premessa iniziale, il robot è in grado di valutare la distanza della stazione e anche la direzione in cui essa è posta rispetto al proprio asse, vedi figura seguente. Per il calcolo della distanza del robot dal marker è stata utilizzata la legge di proporzionalità definita nella relazione [1]. Tale legge è una relazione di proporzionalità inversa per la quale la distanza *d*, in metri, del robot dal landmark si ottiene moltiplicando l'altezza del marker *h* misurata in pixel per un coefficiente *c* stimato attraverso una serie di misure sperimentali.

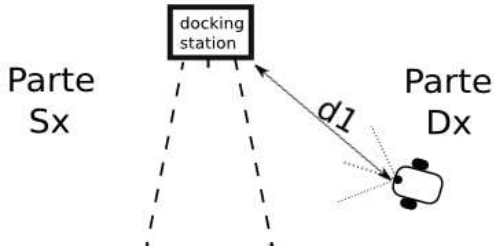

Fig.5 Docking station vista dall'alto

Inoltre con la presenza o l'assenza dei led centrali è possibile determinare se il robot si trovi a destra oppure a sinistra della stazione. Con queste informazioni è possibile creare un comportamento del robot che lo porti in posizione frontale al marker e da qui raggiunga la stazione di ricarica. Bisogna considerare che il robot non può perdere mai di vista il marker attivo e, di conseguenza, durante la manovra di posizionamento frontale l'angolo di sterzo permesso è limitato. Tale restrizione inoltre è in funzione della distanza tra robot e stazione: infatti, se il robot è lontano, vede il marker attivo piccolo rispetto a tutta l'immagine prodotta dalla webcam, vedi figura 6, e quindi l'angolo di sterzo risulta essere meno vincolato. Viceversa, quando il robot è vicino il marker diventa più grande, concedendo meno libertà alla manovra.

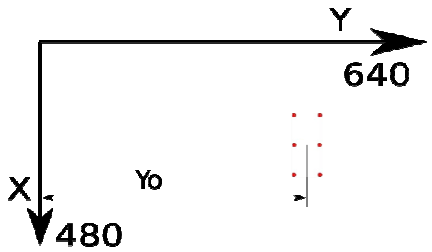

Fig.6 immagine webcam

Con queste considerazioni si è scelto di guidare il robot impostando una velocità costante relativamente bassa e un angolo di sterzo variabile in funzione delle informazioni ricevute dalla webcam. Si è perciò divisa la zona limitrofa alla stazione in diverse aree, come mostra la figura 7, e si è associata ad ognuna di esse la direzione che il robot deve seguire.

Nella zona blu chiaro e scuro la webcam è in grado di vedere tutti e sei i led, perciò, essendo di fronte alla stazione di docking, il robot non deve fare altro che puntare diritto verso l'obbiettivo. Viceversa nelle due zone verdi la webcam non riesce a vedere il sesto led centrale e, quindi, il robot deve sterzare in modo da raggiungere la zona blu senza però perdere di vista il marker. La distinzione tra area verde scuro e verde chiaro è dovuta alla problematica di sterzo descritta precedentemente. Se il robot è lontano (area verde scuro) ha la possibilità di compiere una manovra di allineamento piuttosto ampia e, quindi, non necessità di un grande angolo di sterzo, rendendo quasi nulla la probabilità di perdere di vista il marker. Viceversa, se il robot si trova nei pressi della stazione è necessario che l'angolo di sterzo sia il più grande possibile, al fine di garantire l'entrata nella stazione di ricarica con la giusta angolazione.

Per far seguire al robot la direzione stabilita si è scelto di utilizzare la posizione del marker nell'immagine prodotta della webcam, come mostra la figura 6. Impostando una coordinata Y desiderata, a seconda dell'area corrente, e calcolando la coordinata Yo è possibile quantificare l'errore tra le due, cioè la differenza. In funzione di questo errore è possibile guidare il robot in modo che porti l'errore a zero invocando la funzione setDeltaHeading(**α**), la quale sterza il robot dell'angolo **α** rispetto alla direzione corrente. Infatti se la coordinata Yo coincide con quella desiderata il robot non deve sterzare ponendo **α=0.**

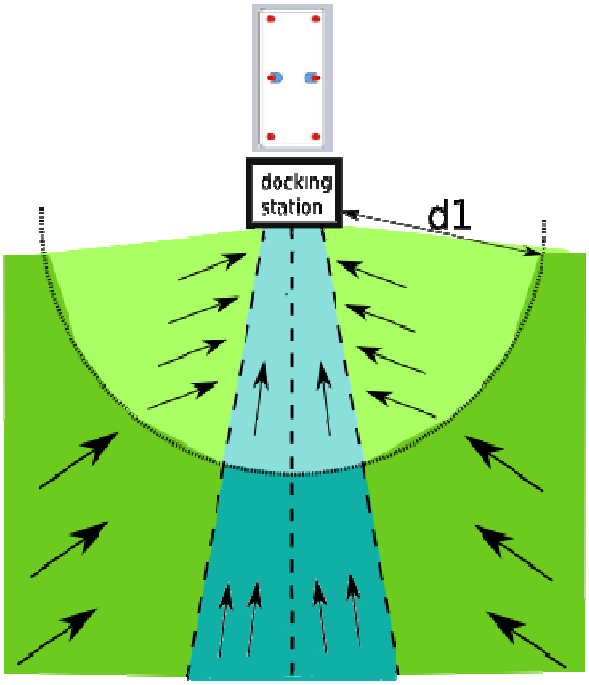

Fig.7 mappa delle direzioni

Facendo varie prove partendo da una posizione iniziale all'interno della zona verde chiaro si è notato che è quasi sicuro che la webcam perda di vista il marker almeno una volta. Si è cercato di diminuire l'angolo di sterzo, ma non è stato possibile trovare un giusto compromesso. Perciò si è corretta l'azione descritta precedentemente, con una piccola modifica. Una volta che il robot ha individuato la stazione e ha iniziato a raggiungerla è possibile tollerare che esso perda di vista i led a patto che esso faccia due cose:

- si fermi, impostando la velocità costante a zero
- continui a sterzare nella direzione in cui è stato visto per l'ultima volta il marker

In questo modo il robot è in grado di riportare velocemente il marker attivo all'interno del campo visivo della webcam. Questa modifica si comporta bene anche se il marker viene oscurato da un ostacolo esterno temporaneo: in questo caso il robot si fermerebbe e continuerebbe a girare su se stesso fino a che l'ostacolo non oscura più il marker. Il risultato di questa modifica permette di far sterzare il robot con il minimo raggio possibile a discapito di un’andatura meno fluida.

Una volta che il robot ha raggiunto la stazione di ricarica deve capire di essere arrivato. Le condizioni che devono essere soffisfatte nello stesso momento sono:

- avere entrambi i motori in stallo, procurato con lo scontro della stazione
- non vedere più i sei led, dato che sono stati spenti con l'arrivo del robot in stazione
- avere l'ultima distanza calcolata inferiore a un soglia prestabilita

A questo punto il robot si ferma e si disconnette dal proprio calcolatore; poi il processo, prima di terminarsi, alza il flag di terminazione del processo di visione all'interno della memoria condivisa.

Il risultato ottenuto da tutto ciò che è stato descritto è una singola azione, chiamata “ArCentro”, la quale permette di portare il robot da una posizione iniziale qualsiasi, purché sia visibile il marker attivo, fino alla stazione di docking. Perciò per completare il lavoro si è deciso di aggiungere il gruppo di azioni associate al programma Wander contenuto negli esempi del pacchetto Aria. Ciò significa che all'avvio del docking se il robot non vede il marker attivo esso comincia ad esplorare lo spazio circondante in modalità wander fino a che non lo vede. Una volta individuata la stazione viene disabilitato il gruppo di azioni contenenti le azioni di wander, e viene attivato in modo esclusivo il gruppo di azioni contenente l'unica azione descritta precedentemente.

Inoltre è stato limitato il tempo per cui il robot può rimanere in modalità docking, usando il contatore decrescente timeout, il quale partendo da un valore pari a 12000 decresce di venti unità al secondo, pari a dieci minuti. In questo modo se passa troppo tempo dal primo avvistamento del marker attivo e se il robot non ha ancora raggiunto la stazione si attiva il comportamento wander, liberando il robot da eventuali situazioni critiche.

Fino ad adesso è stata descritta l'idea con cui è stato implementato il docking di speedy, tuttavia il programma realmente implementato è differente seppur in maniera lieve. La posizione attuale della stazione di docking all'interno del laboratorio ARL è molto vicina ad un muro, perciò non è possibile che il robot si trovi nella parte sinistra della stazione di ricarica, vedi figura 4.

Infatti il robot può trovarsi solo nella zona blu centrale oppure nella zona verde a destra della stazione. Perciò nella scrittura del codice è stato inserito solo il controllo della presenza del sesto led di sinistra della riga centrale. Di conseguenza lo spazio limitrofo alla stazione di ricarica è stato suddiviso in sole quattro zone, e non sei come descritto precedentemente. Di seguito è presente la tabella che illustra le coordinate Y desiderate in funzione della zona:

Zona	Coordinata obiettivo Y
Blu chiaro	300
Blu scuro	350
Verde chiaro	540
Verde scuro	575

La distanza che divide le zone scure da quelle chiare si impostata ad un valore pari a 2.5 metri. In caso di necessità è possibile aggiungere le due zone verdi mancanti con un piccolo inserimento nel codice, semplicemente con un controllo sul led centrale di destra e con due valori per la tabella vicini allo zero invece che al valore 640. Dopo aver fatto questo è anche consigliabile diminuire anche il valore relativo alla zona di blu scuro, per ottenere una simmetria più elegante.

3.3. Modifiche del software di visione

Durante l'implementazione della soluzione adottata ci si è resi conto che in alcune situazioni il programma di visione ritorna dei dati diversi da zero pur non vedendo il marker attivo. Questo tipo di errore è causato dalla presenza di altre sorgenti luminose, ad esempio i riflessi metallici della luce esterna. Perciò se il robot si trova lontano dalla stazione e se accade questo tipo di errore, il robot entra in modalità docking puntando il falso marker. Inoltre ci si è resi conto che anche nelle vicinanze della stazione il programma di visione cadeva in errore minacciando la riuscita del docking. Il pericolo consiste nel fatto che se durante il raggiungimento della stazione si presenta un ostacolo, il programma di visione potrebbe dirottare il robot portandolo verso il falso marker. Perciò si è deciso di modificare il codice del programma di visione in modo da ridurre l'alta probabilità di incombere in errori. Si è prodotti il file speeydock.c modificando il codice preso dalla relazione [1]. Le modifiche effettuate consistono nella aggiunta di ulteriori controlli sull'elaborazione dell'immagine della webcam. Si è controllato che:

- il numero di sorgenti luminose sia compreso tra 4 e 10

  dato che se è visibile il marker lo sono anche almeno i quattro led più esterni, mentre se sono presenti troppe sorgenti è alta la probabilità di errore

- la coordinata X del baricentro dei quattro punti più esterni sia in un range limitato (200<x<280)

siccome l'altezza a cui è posta la webcam è pari all'altezza del centro del marker sicuramente i due led superiori saranno contenuti nella metà superiore dell'immagine, mentre nella metà inferiore saranno contenuti i due led inferiori.

Se i requisiti imposti da queste due condizioni non sono soddisfatti allora il programma di visione pone tutti i dati in memeria condivisa a zero, dicendo così al programma di docking che il marker non è stato individuato. Il risultato ottenuto consiste nel fatto che il programma di docking riceve dati più sicuri, essendo scartate tutte le situazioni critiche. Avendo prodotto un secondo programma, contenuto nel file speedydock.c, è stato necessario compilarlo utilizando le librerie vislib. Tutte le operazioni sono state eseguite seguendo le istruzioni descritte nella relazione [1].

4. Modalità operative

4.1. Componenti necessari

Questo progetto è consiste nella esecuzione di due programmi. I due file eseguibili devono essere contenuti nella stessa directory, per rispettare le regole di inizializzazione della memoria condivisa. Per quanto riguarda la compilazione del sorgente riguardante la parte di visualizzazione si rimanda tutto al riferimento[1]. Comunque si è già compilato il file speedydock.c ottenendo il file eseguibile chiamato speedydock, contenuto all'interno del nostro software.

4.2. Modalità di installazione

Per la compilazione del software sono necessari il pacchetto Aria e il compilatore gcc. Utilizzando il comando make nella cartella dove sono contenuti i sorgenti docking.cpp e docking.h si ottiene l'eseguibile docking.

4.3. Avvertenze

Entrambi gli eseguibili devono essere nella stessa directory e lanciati con i seguenti comandi:

- ./speedydock
- ./docking -rp /dev/ttyUSB0

E' possibile lanciare l'esecuzione con un unico comando:

- ./speedydock & ./docking -rp /dev/ttyUSB0

5. Conclusioni e sviluppi futuri

Questo progetto è riuscito ad implementare il docking del robot Speedy in modo efficace. Inizialmente si è trovato una soluzione più generale al problema, che è stata poi adattata al caso specifico della posizione della stazione all'interno del laboratorio. Perciò questo lavoro è facilmente adattabile ad altre situazioni. Un possibile sviluppo futuro potrebbe essere quello di mettere il marker attivo più vicino alla stazione di ricarica, facendo così è possibile ridurre la lunghezza del cartone che nasconde i led centrali. Si pensa che probabilmente l'unica modifica da fare al codice sia sui valori contenuti nella tabella del capitolo 3.

Bibliografia

[1] Gandelli C., Preosti G., Turelli G.: “Speedy Docking marker detection”

[2] Ricchetti A.: “Software per il parcheggio del robot Speedy”

Indice