**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)

# Sistema per il docking automatico di Tobor

**Elaborato di esame di:**

**Bianchetti Paolo, Fenaroli Fabio,**
**Cauzzi Luciano, Piligriu Saimir.**

Consegnato il: **10 Dicembre 2011**

# Sommario

*Il lavoro svolto consiste nella configurazione del calcolatore presente su Tobor, nell'installazione di due fotocamere sul robot e nella realizzazione di due programmi. Nello specifico, il primo programma acquisisce una serie di immagini dalla webcam per determinare la posizione della docking station mentre il secondo controlla il movimento del robot, sulla base delle informazioni ottenute dalle osservazioni delle due fotocamere, verso la stazione di ricarica.*

# 1. Introduzione

La problematica del riposizionamento del robot Tobor all'interno della sua stazione di ricarica può essere suddivisa nelle seguenti attività:

- Rilevare e stimare la distanza della stazione di ricarica rispetto alla posizione del robot;
- Avvicinarsi in prossimità della stazione di ricarica secondo un percorso opportuno delineato dalle informazioni acquisite dalla webcam frontale;
- Spegnere la webcam frontale ed attivare quella posteriore una volta giunto in prossimità della stazione di ricarica;
- Compiere una rotazione per allinearsi alla stazione di ricarica per consentire il parcheggio in retromarcia;
- Sfruttare la webcam posteriore per allinearsi alla stazione ed eseguire la parte finale di rientro.

L'obiettivo del progetto è la realizzazione di un sistema per il rientro automatico del robot Tobor tramite l'utilizzo di marker attivi rilevati attraverso le webcam.

# 2. Il robot Tobor

Il robot "Tobor" appartiene alla famiglia dei Pioneer 1m, commercializzata da *ActivMedia Robotics*. Questo robot utilizza un sistema di movimento di tipo *differential drive*, con due ruote motrici fisse ed indipendenti ed una ruota folle e pivottante.

Dispone di sette sonar, posizionati sulla parte anteriore, che consentono di rilevare la presenza di ostacoli fino ad una distanza massima di circa 4 m, spaziando su un angolo di poco più di 180°.

Nella parte frontale è presente un pinza che consente al robot di afferrare oggetti. Tale dispositivo per la sua conformazione costringe il robot ad entrare nella stazione di ricarica in retromarcia.

Infine, sono state installate sulla macchina due webcam, rivolte rispettivamente verso la parte anteriore e posteriore del robot, che consentono di acquisire dati visuali relativi all'ambiente.

La comunicazione con il robot avviene attraverso un calcolatore, collocato al di sopra dello stesso, e collegato tramite cavo seriale.

# 3. Marker attivo e autolocalizzazione

Nel contesto della robotica mobile, l'espressione *auto-localizzazione* indica il processo attraverso il quale un robot stabilisce la propria posizione rispetto all'ambiente in cui è stato posizionato.

Per raggiungere tale scopo, è possibile adottare svariate strategie e la nostra scelta è ricaduta sull'utilizzo di *punti cospicui* o *landmark*, ovvero punti fissi rispetto al sistema di riferimento usato.

Le principali famiglie di marker sono:

- Beacon attivi: sono apparati che emettono energia e permettono al robot di individuare la direzione la fonte di tale energia;
- Beacon passivi: dispositivi che riflettono l'energia emessa dal robot;
- Trasponder: apparecchiature che, eccitate da una forma d'energia, rispondono con un'altra forma di energia.

Nel nostro caso, si è ricorso all'utilizzo di tre beacon attivi, che diffondono energia sotto forma di luce prodotta da LED. Le fotocamere installate sul robot sono state ottimizzate per riconoscere tale tipo di luce.

# 4. Problematiche affrontate

Il problema del docking di Tobor si è reso possibile dopo aver risolto i seguenti sotto-problemi:

- Miglioramento della geometria della stazione di ricarica per facilitare il parcheggio del robot;
- Scelta e disposizione dei marker;
- Acquisizione delle immagini dalle fotocamere e individuazione dei marker attraverso analisi di connettività;
- Commutazione delle fotocamere;
- Avvicinamento del robot alla stazione di ricarica;
- Allineamento rispetto alla stazione di ricarica mediante una rotazione.

## 4.1. Miglioramento della base

Le prove sperimentali hanno evidenziato che la topologia della base di ricarica originaria del robot non favoriva le dinamiche di parcheggio nel caso in cui l'automa non fosse orientato correttamente.

Si è scelto di inserire due guide metalliche per correggere meccanicamente la rotta del robot e favorire così il rientro di Tobor nella stazione di ricarica.

Le guide realizzate sono mostrate nella figura seguente.

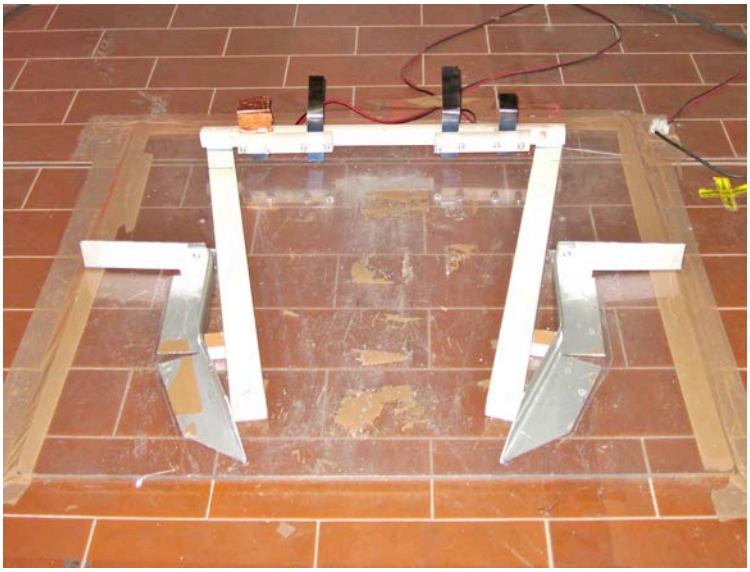


**Fig. 1 - La base dopo l'aggiunta delle guide metalliche**

## 4.2. Costruzione dei marker

In questa fase del progetto sono stati scelti e posizionati i landmark atti ad identificare la base di ricarica di Tobor.

Nello specifico, la scelta è caduta su tre LED a distanza fissa posti su un'asta rigida disposta orizzontalmente rispetto al pavimento. La scelta è stata dettata, innanzitutto, dalla facile reperibilità del materiale e, in secondo luogo, perché in teoria permetteva la costruzione dell'arco capace.

All'atto pratico, l'eccessiva vicinanza dei LED ha portato ad errori grossolani che hanno impedito di far ricorso all'arco capace. Di conseguenza, i marker sono stati impiegati esclusivamente per stimare la distanza e per il calcolo dell'angolo apparente formato dall'asse della fotocamera e dalla stazione di ricarica.

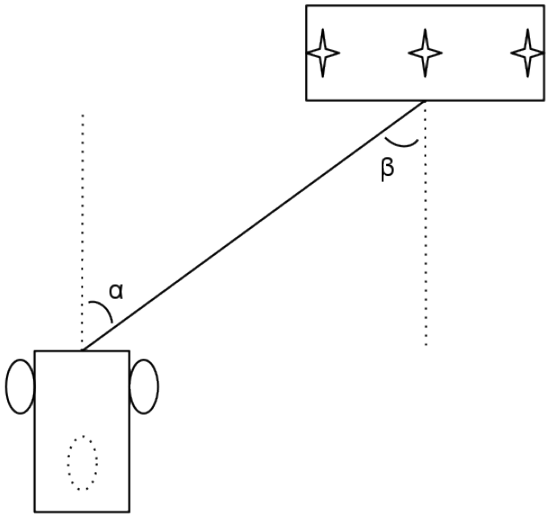



**Fig. 2 - Angolo di inclinazione apparente (α) e angolo reale (β)**

Infine, bisogna evidenziare che la luce emessa dai LED è direzionale. Questa peculiarità comporta che, se Tobor è distante dalla base o rivolto verso la stazione di ricarica con particolari angolazioni, la super-

ficie dei blob individuata dalle telecamere risulta molto ridotta. In questo contesto risulta quindi difficile discriminare i blob dal rumore ambientale.

Di conseguenza, per facilitare il riconoscimento, si è proceduto alla costruzione di piccole lanterne che aumentano la superficie del marker.

# 5. La soluzione adottata

In questo capitolo vengono presentate le soluzioni utilizzate per risolvere i problemi riscontrati durante la realizzazione del progetto.

## 5.1. Ottenimento delle figure da webcam e analisi di connettività

La prima parte del lavoro è stata incentrata sull'individuazione dei parametri, quali guadagno e tempo di integrazione, che consentissero le condizioni ottimali per individuazione dei marker.

Si noti che tali impostazioni non hanno una valenza universale ma dipendono fortemente dall'ambiente di lavoro in cui si trova il robot. Per questi motivi sono state condotte una serie di misure sperimentali per tarare le webcam.

I valori finali sono stati ottenuti dopo una serie di prove sperimentali, portate a termine in differenti condizioni di luminosità e ponendo il robot in diversi punti di rilevamento per ottenere una panoramica più ampia possibile, onde evitare di impostare valori ottimali solo per determinate configurazioni.

Le costanti impostate sono:

- guadagno: 30000
- tempo di integrazione: 38000

Una volta acquisita l'immagine si è proceduto ad elaborare l'immagine. Il processo consiste in due attività. Il primo passo consiste nella binarizzazione dell'immagine tramite un valore di *threshold* (valore soglia).

Il risultato di questa fase è un'immagine composta auspicabilmente da pixel bianchi in corrispondenza delle luci e neri nelle porzioni restanti dell'immagine.

La corretta individuazione delle luci del marker dipende inoltre dal settaggio della soglia di binarizzazione, utilizzata per separare i blob generati dalle sorgenti luminose del landmark dallo sfondo dell'immagine. Il valore impostato per questo parametro è stato 235.

La bontà di questa scelta è stata confermata dall'esito del processo di binarizzazione: i blob individuati corrispondono alle luci del marker e sono chiaramente distinguibili dal resto dell'immagine.

A questo punto si procede con la fase successiva che consiste nell'analisi di connettività che comprende l'individuazione dei blob, ovvero area e baricentro, presenti nell'immagine. Tali informazioni vengono memorizzate in apposite strutture dati.

Durante questa operazione vengono scartati tutti quei blob che sono legati al rumore presente nell'immagine acquisita.

Infine, i dati acquisiti vengono utilizzati dal programma di controllo dei movimenti del robot per modulare i movimenti in termini di velocità e sterzata.

L'analisi di connettività, successiva al processo di binarizzazione, è servita ad estrarre per ciascun blob i dati relativi a:

- coordinata x del baricentro;
- coordinata y del baricentro;
- area.

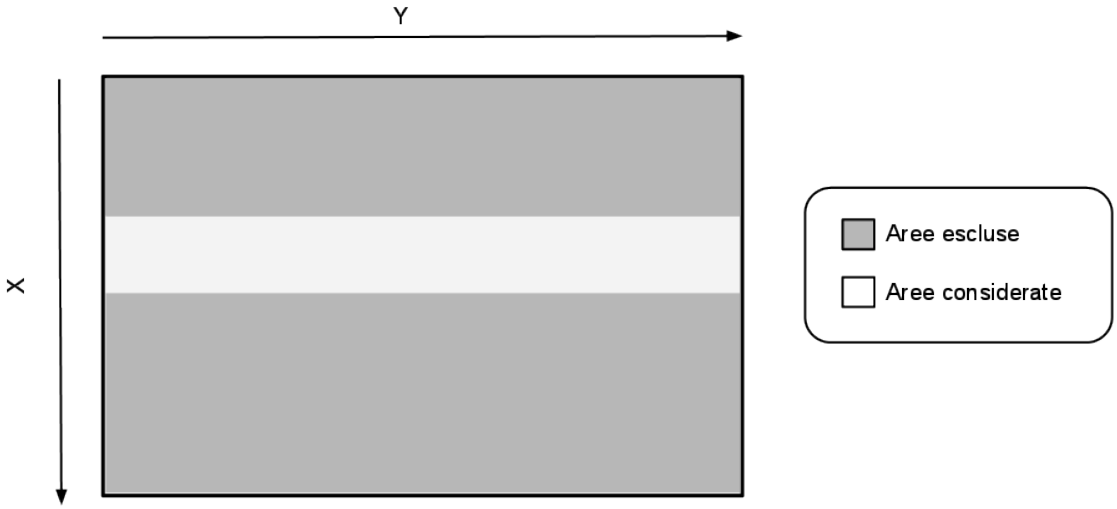
Queste informazioni vengono memorizzate in una struttura dati condivisa con il programma che porta il robot nella stazione di ricarica.

Nel caso venga rilevato più di un blob fra quelli appartenenti alla fascia interna del marker, si procede alla loro sostituzione con un unico blob, avente coordinate del baricentro pari alla media delle coordinate dei blob iniziali.

Al fine di eliminare eventuali blob derivanti da fonti luminose non appartenenti al landmark che possono disorientare il sistema di rientro del robot, vengono effettuati i seguenti controlli:

- I blob rilevati devono avere un'area maggiore di un valore minimo;
- Le zone dell'immagine aventi coordinata X minore di quella del blob più alto del marker e le zone aventi coordinata X maggiore di quella del blob più basso, vengono eliminate.

La figura seguente mostra i tagli effettuati sull'immagine.




Si noti che si è scelto, per convenzione, di riferirsi alle ordinate con l'asse X.

## 5.2. Modalità comportamento Tobor

L'applicazione che effettua il posizionamento nella stazione di ricarica del robot prevede due modalità di comportamento differente in funzione dei dati ricevuti dalle webcam:

- **Wander mode:** è la procedura che viene messa in azione all'avvio dell'applicazione che effettua il docking dell'automa. Il robot si muove in modo casuale fino a quando riesce ad individuare i marker.
- **Docking mode:** è la modalità che consente al robot di avvicinarsi, ruotarsi ed infine entrare nella stazione di ricarica, stimando la propria posizione in base alle coordinate dei blob individuati.

Dopo essere entrato in "docking mode", l'automa stima la propria distanza dalla base di ricarica misurando la differenza fra la posizione dei marker estremi: lo scarto sarà più piccolo in proporzione alla distanza dalla stazione.

Sempre in funzione di questa distanza fra i blob estremi, la macchina stima la propria distanza e la propria posizione.

Tale stima è viene effettuata mediante una proporzione così definita:

dist attuale (cm) : dist punto riferimento (cm) = dist blob punto riferimento (px) : dist blob attuale (px)

dove:

- *distanza attuale* è la distanza del robot dalla base;

- *distanza punto riferimento* è la misura nel punto di riferimento usato per tarare il robot;
- *distanza blob punto riferimento* è la misura espressa in pixel della distanza fra i blob misurati nel punto di riferimento;
- *distanza blob attuale* è il numero di pixel di differenza fra i blob.

È stato adottato un modello che prevede quattro zone in cui velocità e rotazione del robot vengono moderati in modo differente. Nello specifico si prevede una progressiva diminuzione rapportata alla vicinanza alla stazione di ricarica.

In base al valore calcolato, il robot sa di trovarsi in una delle quattro zone in cui è stato suddiviso lo spazio limitrofo alla base. La figura mostra la suddivisione effettuata.

## 5.3. Avvicinamento alla stazione di ricarica

Il processo di acquisizione ed elaborazione delle immagini mette a disposizione del programma di docking i dati necessari per capire come dirigersi verso la stazione e, in base alla posizione rilevata, decidere quali movimenti effettuare:

- Si accorge di essere troppo lontano dalla stazione di ricarica e si approssima alla stazione per procurarsi informazioni più dettagliate;
- Determina che l'angolo fra la sua telecamera frontale e i marker non consente il corretto posizionamento nella stazione di ricarica e provvede ad effettuare le opportune correzioni;
- Non riesce ad individuare i landmark e si muove in modo casuale per portarli nel proprio campo visivo;
- Si ferma quando sufficientemente vicino alla stazione di ricarica.

Il sistema di marker scelti non consente un'autolocalizzazione deterministica e quindi il comportamento della macchina dipende dall'angolo apparente misurato con le webcam.

Il problema in cui si ricade è noto in letteratura come il problema del "Inseguimento cane-lepre" che sfrutta il metodo delle differenze finite.

Il metodo delle differenze finite consiste nell'approssimare il valore della derivata di una funzione $u$ in un punto con un'espressione che ne tenga in conto solo un numero finito di valori della funzione.

## 5.4. Fase finale di parcheggio ed eliminazione sviamenti

La fase finale del parcheggio inizia quanto Tobor è giunto alla distanza stimata di 120 cm. In questa fase il robot viene innanzitutto arrestato ogni movimento, quindi viene spenta la telecamera frontale e attivata quella posteriore.

Successivamente, grazie all'ausilio della webcam posteriore, il robot compie una rotazione che lo porterà ad essere allineato al landmark centrale della stazione di ricarica.

Infine, l'automa tenta il rientro nella stazione di ricarica e ripete le seguenti azioni sino a che non raggiunge il corretto posizionamento nella stazione di ricarica:

- Procede in retromarcia verso la stazione di ricarica, correggendo gli eventuali sviamenti sulla base delle informazioni fornite dalla telecamera.
- Attua un breve accelerazione in prossimità della stazione per superare le difficoltà meccaniche;
- Se il robot rileva di essere in stallo, cioè dentro la base ma non in ricarica, ritenta il posizionamento allontanandosi linearmente dalla base di un metro.

L'evidenza sperimentale ha mostrato che l'angolo stimato nelle zone attigue alla base è molto impreciso e che le correzioni di traiettoria effettuate sulla base di queste informazioni erano controproducenti.

Alla luce di tali osservazioni si è scelto di far allontanare il robot in modo lineare dalla base nel caso in cui non sia riuscito ad entrarvi correttamente.

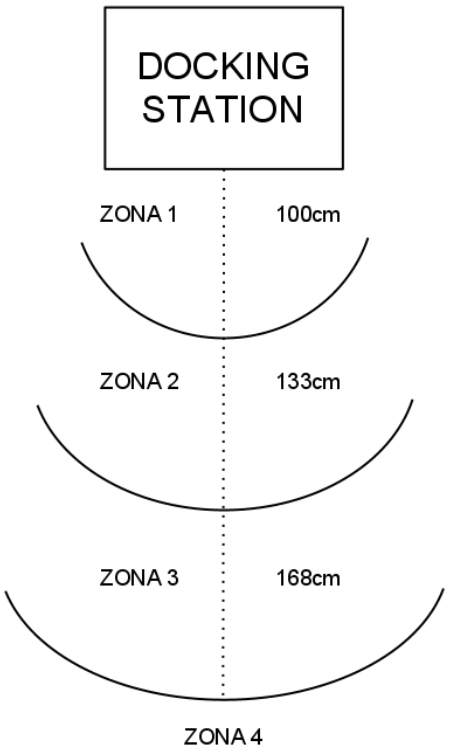



**Fig. 3 - Le zone usate per il parcheggio del robot**

### 5.5. Comunicazione inter-processo

Le applicazioni che controllano la visione e il movimento del robot comunicano fra di loro attraverso un meccanismo a "shared memory" (memoria condivisa) basato sui semafori.

## 6. Modalità operative

### 6.1. Software necessario

Il software prodotto in questo progetto necessita delle librerie Vislib per quanto riguarda la visione e delle librerie Aria per ciò che concerne gli spostamenti del robot.

Nello specifico è necessario compilare ed installare i seguenti pacchetti:

- Vislib 1.9.6;
- Aria 2.7.2 o superiori;
- Gcc.

Infine, per quanto riguarda le webcam si richiede che supportino V4L (Video for Linux v.1) o V4L2 (Video for Linux v.2).

# 7. Taratura dell'automa

## 7.1. Taratura del software tobordock

Prima di avviare i programmi che regolano il docking di Tobor è necessario impostare alcuni parametri relativi alle modalità di acquisizione dell'immagine:

- **X_MIN**: valore minimo dell'ordinata che il baricentro di un blob può assumere. Il valore è stato stimato catturando un'immagine dalla docking station;
- **X_MAX:** valore massimo dell'ordinata che il baricentro di un blob può assumere. Il valore è stato stimato catturando un'immagine dalla docking station;
- **AREA_MIN:** minima area tale per cui un blob viene preso in considerazione durante il riconoscimento del marker;
- **BIN_TH:** soglia di binarizzazione; indica il valore di discriminazione tra blob e sfondo;
- **SET_SHUTTER:** valore di velocità dell'otturatore. Usato per variare il tempo di integrazione della webcam;
- **SET_GAIN:** guadagno della webcam.

## 7.2. Posizionamento dei marker

Il marker deve essere posizionato alla stessa altezza delle fotocamere affinché possa essere identificato correttamente. Questo è requisito fondamentale per il corretto funzionamento del programma affinché i marker non si spostino in verticale all'interno dell'immagine.

# 8. Avvio del programma

Per coordinare correttamente la gestione dei semafori presenti all'interno delle applicazioni è necessario avvalersi del seguente comando:

- ./tobordock & ./docking –rp /dev/ttyUSB0

# 9. Problemi noti ed avvertenze

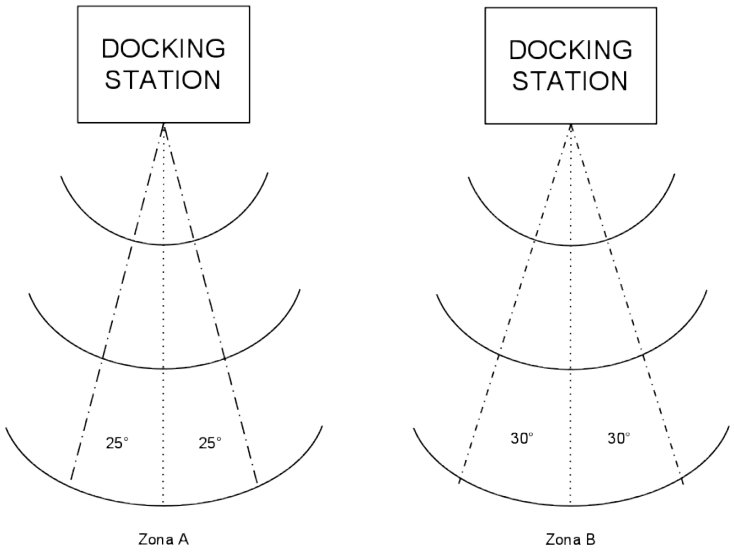
Le applicazioni sono state ottimizzate per l'utilizzo all'interno del laboratorio di Robotica Avanzata dell'Università degli Studi di Brescia ed è concepito per essere utilizzato in ambiente con luminosità moderata. Si tenga presente quindi che i parametri di binarizzazione impiegati sono molto sensibili alle variazioni di luminosità che potrebbero portare ad un mancato riconoscimento del landmark.

Pertanto si consiglia di effettuare nuovamente la taratura nel caso in cui si intenda usare l'automa in un ambiente diverso.

# 10. Conclusioni e sviluppi futuri

Il risultato di questo progetto è stata la realizzazione di un sistema che consente al robot di posizionarsi correttamente nella stazione di ricarica con buone probabilità di successo.

Sperimentalmente sono state individuate due aree che formano rispettivamente un angolo di 25 e 30 gradi rispetto alla normale alla stazione di ricarica. Questi spazi sono stati denominati Zona A e Zona B e sono mostrati in figura.




Nella zona A l'automa si parcheggia con probabilità di successo dell'85% mentre nella zona B le probabilità scendono al 50%.

La bontà delle soluzioni impiegate nel sistema è avvalorata dal fatto che sono stati impiegati strumenti molto semplici e dispositivi di acquisizione video non di alta precisione.

Un sviluppo futuro del progetto potrebbe prevedere l'impiego di landmark e sistemi di acquisizione più precisi.

# 11. Appendice

L'installazione delle librerie per la visione *Vislib 1.9.6* è risultata impossibile poiché il calcolatore era dotato di un sistema operativo troppo obsoleto (*Limpus Linux*) che, pur riuscendo a completare la compilazione del pacchetto, non era in grado di eseguire le librerie.

Per questo motivo si è scelto di sostituire il calcolatore con uno identico dotata di sistema operativo *Debian* "*Squeeze",* versione 6.0.1.

## 11.1. Installazione librerie Aria e Vislib

In prima battuta è stato necessario installare il software sul calcolatore installato sul robot.

Il PC in questione è il modello *Acer Aspire One* su cui risultava installata la distribuzione Linux Linpus2, ovvero una distribuzione derivata da Fedora.

L'obsolescenza di tale distribuzione ha reso impossibile l'installazione delle librerie *Vislib* e, in particolare, della libreria *V4L2 (Video for Linux v.2)*.

Di conseguenza si è reso necessaria la sostituzione del pc in questione con un pc dello stesso modello ma dotato di una distribuzione più moderna e nello specifico di Debian v.6.

Per questione di praticità, tutti i pacchetti installati non presenti nei repository sono stati salvati nella cartella **/home/users/installazione** al fine di renderli facilmente individuabili.

## 11.2. Installazione dei software Aria e MobileSim

I primi due software installati sul calcolatore sono stati Aria e Mobile. Entrambi questi software sono stati reperiti dalla pagina del corso di Robotica Mobile[1].
In aggiunta è stato installato il pacchetto *libstd++5* che, invece, risulta disponibile dal repository di Debian.
Segue il listato dei comandi dell'installazione:

```
apt-get install listdc++5
wget "http://riffraff.ing.unibs.it/~cassinis/ARIA e annessi/Versioni
    correnti/libaria_2.7.2_i386.deb"
dpkg -i libaria_2.7.2_i386.deb
wget "http://riffraff.ing.unibs.it/~cassinis/ARIA e annessi/Versioni
    correnti/mobilesim_0.5.0_i386.deb"
dpkg -i mobilesim_0.5.0_i386.deb
```

Inoltre, sono stati installati i seguenti pacchetti:

- Arln base 1.7.2
- Libarln 1.7.0

```
wget "http://riffraff.ing.unibs.it/~cassinis/ARIA e annessi/Versioni
    correnti/arnl-base_1.7.2+etch+gcc41_i386.deb"
dpkg -i arnl-base_1.7.2+etch+gcc41_i386.deb
wget "http://riffraff.ing.unibs.it/~cassinis/ARIA e annessi/Versioni
    correnti/Pacchetto ARNL.zip"
unzip "Pacchetto ARNL.zip"
cd "Pacchetto ARNL/ARNL-SONARNL"
dpkg -i libarnl_1.7.0_i386.deb
dpkg -i libarnl_1.7.0+etch+gcc41_i386.deb
```

Si è infine proceduto al test dei pacchetti per verificarne il corretto funzionamento.

## 11.3. Installazione librerie Vislib

La versione della libreria è Vislib 1.9 reperita presso il sito del corso[2].

Prima di procedere all'installazione vera e propria è stato necessario installare le seguenti dipendenze:

- libxt-dev
- libv4l-dev
- x11proto-xext-dev
- libxext-dev
- linux-source

[1] http://riffraff.ing.unibs.it/~cassinis/ARIA%20e%20annessi/Versioni%20correnti/

[2] http://riffraff.ing.unibs.it/~cassinis/ARIA e annessi/Versioni correnti/VISLIB/vislib-V1.9.6.tgz

```
apt-get install libxt-dev
apt-get install libv4l-dev
apt-get install x11proto-xext-dev
apt-get install libxext-dev
apt-get install linux-source
cd /usr/src
bunzip2 linux-source-2.6.32.tar.bz2
tar xvf linux-source-2.6.32.tar
ln -s linux-source-2.6.32 linux
```

Si noti la creazione del link simbolico che permette alla libreria di individuare i sorgenti del kernel.

Soddisfatti questi passi preliminari è possibile procedere alla compilazione della libreria:

```
su
cd /usr/local
wget http://riffraff.ing.unibs.it/~cassinis/ARIA e annessi/Versioni
   correnti/VISLIB/vislib-V1.9.6.tgz
tar -zxvf vislib-V1.9.6.tgz
cd /usr/local/vislib
make
```

A questo punto la libreria vislib _e con_gurata e pronta per il funzionamento.

## 11.4. Installazione del laser

Il laser installato su Tobor è il modello *URG-04LX4* prodotto da *Hokuyo*.

Prima di procedere alla compilazione si è reso necessario soddisfare le seguente dipendenze fra pacchetti:

- libsdl1.2-dev
- libboost-dev
- libboost-regex-dev

```
apt-get install libsdl1.2-dev
apt-get install libboost-dev
apt-get install libboost-regex-dev
```

Si è poi proceduto alla compilazione vera e propria dei driver del laser:

```
wget http://www.hokuyo-aut.jp/02sensor/07scanner/download/
   urg_programs_en/urg-0.8.12.zip
unzip urg-0.8.12.zip
cd urg-0.8.12
./configure
make
su
make-install
```

Il corretto funzionamento del laser □e stato testato utilizzando il software *MobileEyes* sul calcolatore *dambot*.

## 11.5. Installazione Mapper3 Basic

L'ultimo software installato è stato Mapper3 Basic ed è stato reperito dal sito del corso di Robotica Mobile[3].

```
wget http://riffraff.ing.unibs.it/~cassinis/ARIA e annessi/Versioni
   correnti/mapper3-basic_2.2.5-1_i386.deb
dpkg -i mapper3-basic_2.2.5-1_i386.deb
```

## 11.6. Altri pacchetti

Si è quindi proceduto all'installazione dei pacchetti necessari al funzionamento delle webcam.

```
apt-get install luvcview
apt-get install uvccapture
```

## 11.7. Impostazione permessi sui device

Infine è stato necessario impostare correttamente i permessi delle periferiche laser e webcam in mode che fosse possibile accedervi dall'utente *user*.

Per raggiungere l'obiettivo, l'utente *user* è stato reso membro:

- dei gruppi *dialout* e *uucp* per il corretto funzionamento del laser;
- del gruppo *video* per accedere alla webcam.

La porzione del codice seguente mostra la modifica perpetrata ed il contenuto del file */etc/group*.

```
uucp:x:10:user,cassinis
dialout:x:20:user,cassinis
video:x:44:cassinis,user
```

# Bibliografia

[1] Barbariga et al.: "Sistema per il docking automatico di Speedy", 2010.

[2] Chiodi et al.: "Adattamento di Vislib a V4L2", 2009.

[3] Johann Borenstein et al.: "Where am I?", University of Michigan

[4] http://www.mobilerobots.com

[5] http://www.hokuyo-aut.jp/

[3] http://riffraff.ing.unibs.it/~cassinis/ARIA e annessi/Versioni correnti/mapper3-basic_2.2.5-1_i386.deb

# Indice