UNIVERSITÀ DI BRESCIA
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robot industriali e di servizio
(Prof. Riccardo Cassinis)

Trasferimento SW Morgul su calcolatore Aspire One

Elaborato d'esame di:

Mattia Asperti, Mirko Lavarini, Matteo Venturelli

Consegnato in data: **18 agosto 2012**

Sommario

L'obiettivo di questo progetto è la sostituzione del vecchio calcolatore presente sul robot Morgul, un Acer TravelMate 630, con un pc di nuova generazione, un Acer Aspire One 110. Il lavoro svolto dal gruppo è consistito in un trasferimento e aggiornamento del software di Morgul e di tutti i file associati al suo funzionamento. Inoltre, il pc è stato configurato per poter dialogare con i server del laboratorio che contengono risorse dati e sistemi di controllo.

1. Introduzione

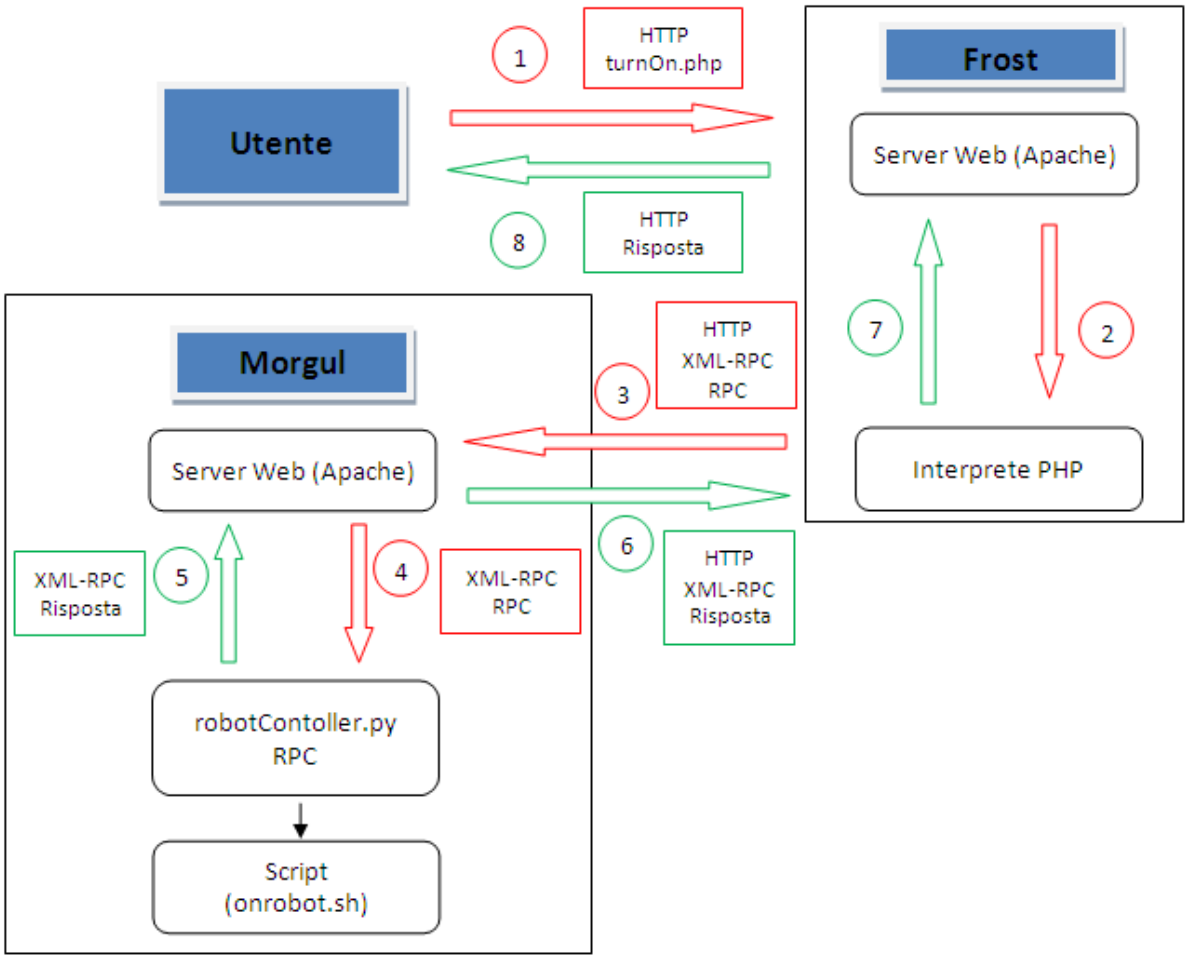
Il progetto del corso di robotica è stato sviluppato attorno al robot Morgul, acronimo di "MObile Robot for Guarding University Laboratory". Si tratta di un robot mobile basato sul modello PIONEER 3-AT prodotto dalla ActivMedia Robotics.
Il sistema di controllo, ovvero la gestione di tutte le operazioni che il robot è in grado di eseguire, è affidato ad un calcolatore esterno collocato sopra il deck del robot.
L'obiettivo dell'elaborato è conoscere in modo approfondito il sistema di controllo in modo da poterlo rimpiazzare con uno più moderno: abbiamo sostituito il calcolatore presente, un Acer TravelMate 630, con un pc di nuova generazione, un Acer Aspire One 110.
Questa esigenza è nata sia per avere maggiori prestazioni (più memoria, capacità di calcolo superiori,ecc..) sia per risolvere problemi legati alla connessione e al surriscaldamento del calcolatore precedente.

1.2. Architettura di Morgul

1.1.1. Controllo web [1]

Il robot Morgul, in alcune delle sue funzioni, è controllabile dal web. Per fornire una visione generale dell'architettura del sistema è bene analizzare un esempio che permetta di descriverne concretamente i componenti.
Supponiamo che un utente remoto e autorizzato decida di impartire un comando a Morgul: si desidera trasmettere il comando di accensione al microcontrollore del robot.
L'utente quindi utilizzerà il browser web di un qualsiasi calcolatore collegato ad internet per richiedere la pagina *http://frost.ing.unibs.it/protected/fdr/turnOn.php* (tipicamente arriverà a richiedere la pagina attraverso un collegamento presente nella pagina principale del sistema di controllo, ma per semplicità si supponga ora che la richiesta sia diretta).
Tale richiesta è servita da Frost, calcolatore che assolve, tra l'altro, alla funzione di server web per il laboratorio. La pagina in questione è scritta utilizzando il linguaggio PHP: il server web di Frost pertanto invocherà, al momento di servire la richiesta, l'interprete PHP affinché esegua il codice contenuto nella pagina.
Il codice PHP che compone la pagina contiene al suo interno una chiamata a procedura remota (RPC – *Remote Procedure Call*). Per la precisione, si tratta di una chiamata a procedura remota che sfrutta il protocollo XML-RPC e che invoca su Morgul la procedura di nome `turnOnRobot`.
Poiché lo standard XML-RPC prevede che il trasporto sia effettuato su protocollo HTTP, su Morgul è stato installato il server web Apache. Esso riceve la richiesta HTTP, ne estrae il contenuto XML-RPC (che non saprebbe altrimenti interpretare) e lo inoltra, attraverso l'interfaccia CGI, all'applicazione `robotController`.
L'applicazione robotController esamina il contenuto della richiesta XML-RPC, estrae il nome della procedura (in questo caso `turnOn`) e la esegue. La procedura `turnOnRobot` invoca in modo non bloccante lo script di shell `onrobot.sh`. Inoltre, restituisce un valore (in questo esempio, una stringa che contiene il messaggio per indicare che il robot è in fase di accensione) il quale viene impacchettato in una risposta XML-RPC e restituito al server web di Morgul. Il server web di Morgul può a questo punto rispondere alla richiesta XML-RPC che gli era stata inviata dall'interprete PHP in esecuzione su Frost. In seguito, l'interprete PHP può usare la risposta ottenuta (in questo caso una stringa di testo)

per confezionare una pagina HTML che restituisce al server web in esecuzione su Frost, il quale, a sua volta, la inoltra al calcolatore dell'utente.
Nel frattempo su Morgul è stato eseguito lo script `onrobot.sh`, che esegue le operazioni di accensione del robot.



1.1.2. Database centralizzato [2]

Per il normale funzionamento di Morgul e del servizio web Sauron è necessario disporre di numerosi dati, generati sia dal robot stesso che da alcune applicazioni in esecuzione su di esso. Il sistema implementato per la lettura e scrittura di tali dati opera su un database centralizzato remoto.
Per rendere possibile la connessione al database, in modo trasparente rispetto alla rete a cui il robot è collegato, viene utilizzato un port forwarding tramite il servizio criptato SSH.
La porta locale sulla quale il database ascolta, viene reindirizzata alla porta sul server remoto, in questo modo il robot è in grado di fare richieste al database remoto esattamente come se fosse in locale.

2. Il problema affrontato

Il trasferimento di software da un calcolatore vecchio ad uno più moderno può incontrare problemi di compatibilità dovuti alle differenze tra i due pc.
Sul calcolatore nuovo è installato un sistema operativo Linux, ma con una versione più recente rispetto a quella presente sul vecchio.
La maggior parte del lavoro ha riguardato il trasferimento e, in parte, la compilazione di tutti i software relativi al funzionamento del robot.
Le difficoltà principali sono dovute a diversi problemi:

- assenza della porta parallela sul nuovo calcolatore;
- compatibilità tra le due versioni di Debian;
- diversa gestione dei file della batteria;
- correzione librerie per l'utilizzo della webcam Philips;
- connessione con il database;
- nuova versione del web server Apache2.

3. La soluzione adottata

3.1. Assenza della porta parallela sul nuovo calcolatore [3]

La porta parallela viene utilizzata dal vecchio calcolatore per l'accensione e lo spegnimento dei motori, delle luci e del lampeggiante presenti sul robot.
Sul nuovo calcolatore questa porta non è presente, quindi abbiamo utilizzato un modulo hardware, già realizzato, il cui funzionamento è descritto all'interno del Rapporto Tecnico "Emulazione e gestione di una porta parallela di PC V4.0".
Questo dispositivo adatta la porta parallela alla porta USB, ma con qualche difetto di comunicazione: ogni volta bisogna riavviare i driver della porta USB perché, durante la fase di accensione, il calcolatore li resetta troppo presto e non comunica più con il dispositivo collegato. Di fronte a questo problema si delineano due possibili soluzioni: hardware o software.
L'aggiunta di una modifica al software si è rivelata il rimedio più semplice e immediato.
Per fare un reset manuale dei driver della porta USB sono stati inseriti all'interno del file `/etc/rc.local` i seguenti comandi:

```
#sleep 30;
#rmmod ehci_hcd;
#modprobe ehci_hcd ;
```

Il primo dei tre comandi serve a ritardare di trenta secondi il reset dei driver durante l'accensione del pc, mentre gli altri due eseguono il vero e proprio reset.
Dopo aver inserito questi comandi abbiamo verificato il corretto funzionamento dell'adattatore.

3.2. Installazione software e librerie

I due calcolatori montano differenti sistemi operativi: su quello vecchio, Acer TravelMate 630, è installato un sistema operativo Debian 2.6.18-6-686, mentre il nuovo calcolatore monta un sistema operativo Debian Squeeze 2.6.32-5-686. Questa differenza ha causato problemi di compatibilità su alcuni pacchetti installati.

3.2.1. Software Aria e MobileSim

Aria contiene tutte le librerie necessarie ai programmi che vogliono interfacciarsi con il robot. MobileSim permette di simulare gli applicativi al calcolatore per verificarne il corretto funzionamento prima di caricarli sul robot.

I software, scaricati all'indirizzo *http://riffraff.ing.unibs.it/~cassinis/Aria e annessi/Versioni correnti*, sono stati installati mediante la procedura riportata sotto:

```
#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
```

Il makefile di Aria, contenuto in `/usr/local/Aria` è stato modificato con l'aggiunta del comando "`-lstdc++`" a questa riga del codice.

```
CXXLINK=-Llib  -L$(ARIA_LINK_PATH)  -lAria  -lpthread  -ldl  -lrt  -
lmysqlclient -lstdc++
```

A questo punto è stata ricompilata la libreria `libaria`:

```
#cd /usr/local/Aria
#make clean
#make
```

3.2.2. Librerie ARNL

Sono stati installati i pacchetti per la gestione dei sonar.

```
#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
```

3.2.3. Software Mapper3 Basic

È il programma con cui si possono creare le mappe degli ambienti in cui il robot deve muoversi.
La procedura di installazione è la seguente.

```
#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
```

All'interno della cartella `/home/morgulweb/maps` sono state inserite le mappe "`long_patrol.map`" e "`short_patrol.map`" già presenti nella stessa cartella sul vecchio calcolatore.

3.2.4. Librerie Vislib

Le librerie Vislib sono utilizzate per l'acquisizione e l'elaborazione delle immagini.
La procedura di installazione è analoga a quelle dei software precedenti, ma necessita di pacchetti aggiuntivi: `libxt-dev`, `libv4l-dev`, `x11proto-xext-dev`, `libxext-dev` e `linux-source`.
Il link simbolico serve per consentire alla libreria `Vislib` di trovare i sorgenti del kernel.

```
#apt -get install libxt-dev
#apt -get install libv4l-dev
#apt -get install x11protoc-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
```

Prima di installare la libreria è stato creato un nuovo link simbolico: il compilatore C in questa versione di Debian, situato nella cartella `/usr/bin`, è "`gcc-4.4`" mentre nella versione più vecchia era semplicemente "`gcc`". Per evitare di modificare il makefile della libreria (e in futuro di altri programmi) il comando "`gcc`" è stato collegato direttamente al comando "`gcc-4.4`" in modo che quando un qualsiasi programma chiama "`gcc`" in realtà fa riferimento a "`gcc-4.4`".

```
#ln -s /usr/bin/gcc-4.4 gcc
```

Di seguito vengono compilate le librerie.

```
#su
#cd /usr/local
#wget http://riffraff.ing.unibs.it/~cassinis/Aria e annessi/Versioni
correnti/VISLIB/vislib-V1.9.4.tgz
#tar -zxvf vislib-V1.9.4.tgz
#cd usr/local/vislib
#make
```

3.3. Trasferimento file per il controllo di Morgul da remoto

Nella cartella `/home/morgulweb/public_html` sono stati copiati i file necessari per controllare il robot da remoto:

- "`log.php`" v.4.1;
- "`RobotControllerApplet.class`" in `/home/morgulweb/public_html/applets/robotcontroller;`
- Nella cartella `home/morgulweb/public_html/cgi-bin/:`
  - `rc.cgi;`
  - `robot Controller.py`

- robotController (6_21_11 1_34 PM).py;
- rs_dbinterface.pyc;
- rs_dbinterface_test.py
- MAKEDIST;
- INSTALL.

Tutti i file nella cartella /home/morgulweb/public_html/cgi-bin/ corrispondono alla versione 4.1.
In realtà, "rs_dbinterface.pyc" e "rs_dbinterface_test.py" sono contenuti nella cartella /home/morgulweb/rs_lib/python/rs_dbinterface.py, mentre nella cartella /home/morgulweb/public_html/cgi-bin/ sono stati creati due collegamenti a quei file.
La creazione dei link ha lo scopo di riunire nella stessa cartella "robot Controller.py" e i due file "rs_dbinterface.pyc" e "rs_dbinterface_test.py", di cui "robot Controller.py" ha bisogno per funzionare. Avremmo potuto copiare i file originali nella cartella /home/morgulweb/public_html/cgi-bin/, ma in questo modo, una modifica ai file originali non si sarebbe estesa alle copie.
Il compito di "robot Controller.py" è gestire la comunicazione tra il calcolatore e un server remoto[1].
Ad esempio, quando si vuole controllare Morgul attraverso il server *frost.ing.unibs.it*, i comandi inviati dal server arrivano al robot passando attraverso "robot Controller.py".

3.3.1. Modifiche al programma "*robotController.py*"

Il software "robotController.py" contiene all'interno le chiamate ai vari script di Morgul, ad esempio può dire al robot di accendersi ("*turnOnRobot*"), di spegnersi ("turnOffRobot")facendo riferimento ai vari script ("onrobot.sh", "offrobot.sh").
Di seguito si riportano i riferimenti ai vari script.

```
    def turnOnRobot(self):
        return "Robot controller turned on <br> Status " + \
                str(os.system(SCRIPTDIR+'onrobot.sh 0.0.0.0')/256);

    def turnOffRobot(self):
        return "Robot controller turned off <br> Status " + \
                str(os.system(SCRIPTDIR+'offrobot.sh
0.0.0.0')/256);

    def emergencyStop(self):
        return "Emergency stop activated <br> Status " + \
                str(os.system(SCRIPTDIR+'emergencyStop.sh
0.0.0.0')/256);

    def flashHeadlights(self):
           os.system("morgulc -LlLl");
           return "Flashing..."

    def turnOnHeadlights(self):
           os.system("morgulc -L");
           return "Turning on headlights"

    def turnOffHeadlights(self):
           os.system("morgulc -l");
           return "Turning off headlights"
```

[1] Vedi capitolo 2.1. Controllo web

```
    def enablePhotoSnapper(self):
          os.system('photoreporter -r -t 1 > /dev/null');
          return "Enabling photo snapper..."

    def disablePhotoSnapper(self):
        os.system('photoreporter -k');
        return "Disabling photo snapper..."
```

Nascono problemi nel funzionamento di "`robotcontroller.py`" quando gli script che richiama generano un standard output in uscita.
Per evitare questo inconveniente ad ogni script invocato è stato aggiunto il comando "`> /dev/null`" in modo che lo standard output venga buttato via.
Per esempio, quando viene chiamato "`turnOnRobot`" e il robot è già acceso, lo script genera una stringa del tipo 'robot is already on'. Con la modifica effettuata la stringa viene gettata via e la chiamata a "`turnOnRobot`" non restituisce nulla.

```
def turnOnRobot(self):
          return "Robot controller turned on <br> Status "  + \
               str(os.system(SCRIPTDIR+'onrobot.sh 0.0.0.0 >
/dev/null')/256)

    def turnOffRobot(self):
        return "Robot controller turned off <br> Status " + \
               str(os.system(SCRIPTDIR+'offrobot.sh 0.0.0.0 >
/dev/null')/256);

    def emergencyStop(self):
        return "Emergency stop activated <br> Status " + \
               str(os.system(SCRIPTDIR+'emergencyStop.sh 0.0.0.0 >
/dev/null')/256);

    def flashHeadlights(self):
          os.system("morgulc -LlLl > /dev/null");
          return "Flashing..."

    def turnOnHeadlights(self):
          os.system("morgulc -L > /dev/null");
          return "Turning on headlights"

    def turnOffHeadlights(self):
          os.system("morgulc -l > /dev/null");
          return "Turning off headlights"

    def enablePhotoSnapper(self):
          os.system('photoreporter -r -t 1 > /dev/null');
          return "Enabling photo snapper..."

    def disablePhotoSnapper(self):
        os.system('photoreporter -k > /dev/null');
        return "Disabling photo snapper..."
```

3.4. Trasferimento script di Morgul

Gli script presenti sul vecchio calcolatore sono stati copiati, senza nessuna modifica, nel nuovo, dentro la cartella `/home/morgulweb/script`:

- `emergencyStop.sh;`
- `checksetlock.sh;`
- `MAKEDIST.sh;`
- `unlock.sh;`
- `onrobot.sh;`
- `offrobot.sh;`
- `exitDock.sh;`
- `SETUPDIRECTORIES.sh;`
- `goHome.sh;`
- `patrolmap.sh;`
- `autopatrolmap.sh;`
- `copiaFoto.sh;`
- `robotlogrotate.sh;`
- `robotparameters.sh;`

Tutti questi script sono aggiornati alla versione 3.1.

3.5. Trasferimento programmi di Morgul

3.5.1. Script in /usr/bin

La cartella `/usr/bin` del vecchio pc conteneva script e programmi. Gli script sono stati copiati e in parte modificati, mentre i programmi hanno dovuto essere ricompilati.
Di seguito si trova l'elenco degli script copiati, la vecchia e la nuova versione e se sono stati modificati. Tutti questi script si trovano nella cartella `/usr/bin`.

Script	*Versione precedente*	*Versione corrente*	*Modificato?*
Checkbatt	1.0	**1.1**	**Si**
Checkdock	2.1	**2.2**	**Si**
Handdrive	1.0	**1.0**	**No**
Photoreporter	2.1	**2.1**	**No**
photoreporter-panorama	2.0	**2.0**	**No**

3.5.2. Modifiche agli script in /usr/bin

Gli script modificati sono due: "`checkdock`" e "`checkbatt`".

checkdock. Il nuovo calcolatore gestisce in maniera diversa i file relativi alla batteria e all'alimentatore. Per gestire questa novità, nel codice dello script sono state fatte due sostituzioni alle righe 48 e 88. Il codice vecchio

```
#48     statefile="/proc/acpi/ac_adapter/AC/state"

#88     if cat $statefile | grep 'on-line' > /dev/null ;
```

è stato sostituito con

```
#48     statefile="/sys/class/power_supply/ACAD/online"

#88     if cat $statefile | grep '1' > /dev/null ;
```

checkbatt. Per lo stesso motivo in questo script sono state fatte due modifiche e un'aggiunta alle righe 35,54 e 55. Il codice

```
#35     battfile="/proc/acpi/battery/BAT0/state"

#54     variabile=$(cat $battfile | grep 'remaining capacity' | \
#55      sed -e 's/[^0-9]//g')
```

è stato modificato come segue.

```
#35     battfile="/sys/class/power_supply/BAT1/charge-now"

#54     charge $(cat $battfile)
#55     variabile=$(expr $charge | 1000)
```

3.5.3. Programmi in /usr/bin

Tutti i sorgenti dei programmi sono stati ricompilati sul nuovo calcolatore dopo aver aggiunto al "`makefile`" di ognuno il comando "`–lstdc++`".

```
CXXLINK=-Llib -L$(ARIA_LINK_PATH) -lAria -lpthread -ldl -lrt -
lmysqlclient -lstdc++
```

Durante la compilazione ci sono stati problemi dovuti alla libreria "`listdc++5`" installata insieme alle librerie di Aria.
È nato un conflitto tra questa libreria e "`listdc++6`" quando viene invocato all'interno del makefile di alcuni programmi, ad esempio `morguldock`, il compilatore "`g++`".
Per risolvere questo problema è stato creato un link al compilatore "`gcc-4.4`" sfruttando il sistema delle alternatives.

```
update-alternatives –install /usr/bin/g++ g++ /usr/bin/gcc-4.4 40
```

La compilazione è stata ottenuta spostandosi nella cartella del programma e utilizzando il comando make, eccetto per il programma "`beaconFinder`".

```
#cd percorso_cartella_programma
#make
```

Nel caso di "`beaconFinder`" la compilazione è avvenuta come segue.

```
#cd percorso_cartella_programma
#./configure
#make install
```

Di seguito si riporta la lista dei programmi inseriti, le relative versioni e se è stata effettuata una modifica.

Programmi	*Versione precedente*	*Versione corrente*	*Modificato?*
exitFromDock	2.1	**2.2**	**Si**
Patrol_map	3.0	**3.1**	**Si**
Recoverdock	2.1	**2.2**	**Si**
remoteControlServer	3.1	**3.2**	**Si**
Morguldock	2.1	**2.2**	**Si**
Beaconfinder			**No**

4. Utilizzo webcam Philips [4]

Per utilizzare alla massima risoluzione la webcam Philips installata sul robot, evitando il continuo intervento del controllo automatico del guadagno è stato necessario apportare delle modifiche.
Infatti il driver fornito con il kernel contiene l'abilitazione del controllo automatico di sensibilità della telecamera. Non è quindi possibile utilizzare i comandi per il settaggio manuale della sensibilità perché quando la telecamera entra in funzione i valori settati manualmente vengono sostituiti con quelli calcolati in automatico dalla telecamera.

4.1. Installazione software per l'utilizzo della webcam

Per poter utilizzare la webcam è stato necessario installare, tramite il gestore pacchetti Synaptic Package Manager, i seguenti pacchetti:

- **xawtv**: programma per utilizzare la webcam in modalità video;
- **vgrabbj** : programma per l'acquisizione delle immagini;
- **setpwc** : programma per il settaggio del guadagno, della velocità dell'otturatore e di altri parametri relativi alle webcams che montano driver Philips.

Le modifiche suggerite in seguito fanno riferimento al device `/dev/video1` associato alla webcam Philips esterna (il device `/dev/video0` è associato alla webcam interna integrata).

4.2. Verifica della necessità delle modifiche

Le modifiche suggerite sono necessarie se, dopo aver eseguito i seguenti comandi:

```

```

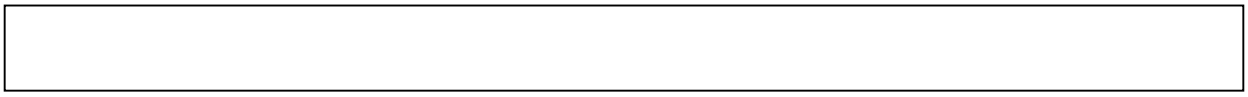
l'immagine che appare è inizialmente scura, ma poi si schiarisce con il passare del tempo.
Questo comportamento indica che è rientrata in funzione la regolazione automatica del guadagno, che invece con il comando `setpwc` era stata disabilitata.
Se invece l'immagine rimane molto scura, le modifiche qui descritte sono evidentemente già state applicate.

4.3. Modifiche adottate

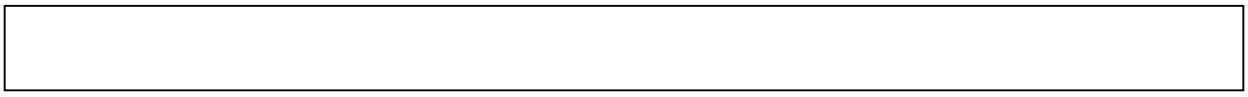
La soluzione consiste nel ricompilare il driver dopo aver eliminato le due istruzioni che causano il problema.

Procedura da adottare, dopo aver acquisito i privilegi di root:
Modificare a mano il file `/usr/src/linux/drivers/media/video/pwc/pwc-if.c` commentando le due righe:

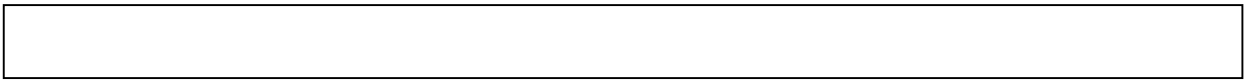
Inizio procedura compilazione driver:



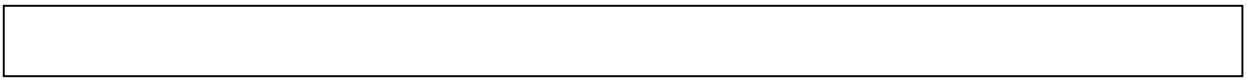
Recuperare l'ultima configurazione del kernel debian:



Compilare il modulo `pwc`:



Rimuovere il vecchio modulo `pwc`:



Copiare il nuovo modulo `pwc.ko`:

Caricare il nuovo modulo:

A questo punto viene segnalata un'incompatibilità del nuovo modulo (dovuta al fatto che il kernel è stato compilato con un file di configurazione diverso). È quindi sufficiente modificare il file `/etc/rc.local` inserendovi la seguente linea:

Quindi riavviare il pc e poi verificare che il modulo `pwc` sia stato caricato correttamente:

Abbiamo verificato l'avvenuto successo delle modifiche apportate eseguendo i comandi riportati nel paragrafo 4.2.

5. Configurazione connessione rete Wi-Fi

Per esigenze di configurazione della rete del laboratorio il calcolatore di Morgul non può ricevere l'indirizzo IP tramite DHCP ma deve essere assegnato manualmente. Va inoltre garantita la connessione in automatico alla rete Wi-Fi "Robotica", altrimenti il robot non può interagire con le strutture del laboratorio.
Il settaggio della configurazione di rete è stato effettuato tramite il tool Network Manager.

Impostazioni:
- indirizzo IP: 192.0.2.10
- subnet mask: 255.255.255.0
- default gateway: 192.0.2.1
- server DNS: 192.0.2.1

Inoltre, è stato necessario togliere la spunta all'opzione di connessione automatica per la rete wifi "Ingegneria", per evitare che all'avvio il pc si connetta automaticamente a una rete diversa da "Robotica".

6. Configurazione database Sauron [2]

Tutti i file di libreria necessari al funzionamento dei programmi sono contenuti nella cartella `rs_lib`.
È stato quindi necessario creare una cartella nella home dell'utente chiamata `rs_lib` tramite:

```
```

copiandovi i file contenuti nell'omonima cartella del vecchio calcolatore.
Per garantire il funzionamento dei programmi che utilizzano tali librerie sono stati installati, tramite il gestore pacchetti Synaptic Package Manager, i seguenti pacchetti:

- `libmysqlcppconn4;`
- `libmysqlclient15-dev;`
- `php5-mysql;`
- `php5-gd;`
- `mysql-client;`
- `phytonmysql-db.`

6.1. Impostazione certificato

Al fine di garantire l'autenticazione automatica della connessione con il server "Rosie" su cui risiede il database, sono state create le chiavi di accesso pubblica e privata del robot. Quella pubblica è stata copiata all'interno del file `~/.ssh/authorized_keys` sul server "Rosie" in modo che, al collegamento, non venga chiesta la password. Per far ciò, occorre eseguire i seguenti comandi:

```
```

6.2. Installazione tunnel SSH

Per l'impostazione automatica del port forwarding all'avvio del sistema operativo va eseguito il demone `rs_sauron-daemon.sh`. Lo script `install-daemon` si occupa delle operazioni di copia del demone nella cartella `init.d` e di avvio automatico del servizio con l'avvio del sistema operativo. A tal proposito è stato modificato lo script `install-daemon.sh` sostituendo la riga:

```
```

con:

```
sudo insserv rs sauron-daemon.sh defaults
sudo ~/rs_lib/sh/install-daemon
NameVirtualHost *:8081
Listen 8081
```

causa la segnalata obsolescenza del servizio `update-rc.d`.

A questo punto lo script è stato lanciato così:

7. Configurazione Server Apache2 [5]

Il server di Morgul risponde alle richieste di XML-RPC sulla porta 8081: nei file di configurazione del server Apache2 è stata aggiunta tale porta. In più, sono state apportate delle ulteriori modifiche a tali file affinché sul calcolatore di Morgul possano essere eseguiti programmi CGI contenuti in directory private degli utenti.
Per l'aggiunta della porta 8081 all'elenco delle porte su cui è in ascolto il server, si procede modificando il file `/etc/apache2/ports.conf` con l'aggiunta delle seguenti righe:

L'Abilitazione dell'esecuzione di programmi CGI contenuti in directory private degli utenti si ottiene aggiungendo all'inizio del file `/etc/apache2/sites-available/default` le seguenti righe:

```
<VirtualHost *:8081>
      ServerAdmin webmaster@localhost

      DocumentRoot /var/www
      <Directory />
            Options FollowSymLinks
            AllowOverride None
      </Directory>
      <Directory /var/www/>
            Options Indexes FollowSymLinks MultiViews
            AllowOverride None
            Order allow,deny
            allow from all
      </Directory>

      ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
      <Directory "/usr/lib/cgi-bin">
            AllowOverride None
            Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
            Order allow,deny
            Allow from all

     </Directory>

#Aggiunte da RC 01/13/2007

        UserDir public_html
```

```
dialout:x:20:cassinis,morgulweb,www-data
video:x:44:cassinis,morgulweb,www-data
```

```
        <Directory /home/*/public_html/cgi-bin>
                Options ExecCGI FollowSymLinks
                SetHandler cgi-script
        </Directory>
        <Directory /home/*/public_html>
#L'opzione Indexes aggiunta da RC il 18/9/2009
                Options Indexes FollowSymLinks
        </Directory>

     ErrorLog ${APACHE_LOG_DIR}/error.log

     # Possible values include: debug, info, notice, warn, error, crit,
     # alert, emerg.
     LogLevel warn

     CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>
```

Di seguito, l'aggiunta dei moduli `userdir.conf` e `userdir.load` (operazione eseguita da root):

```

```

Infine, il web server è stato riavviato:

```

```

7.1. Gestione periferiche da remoto

Il controllo web di Morgul prevede che gli script necessari vengano chiamati dal programma `robotcontroller.py`[2]. Quando tali script vengono invocati da remoto girano sotto l'utente "`www-data`". Questo utente deve essere aggiunto ai gruppi che utilizzano le periferiche richiamate in tali script. Per far ciò nel file `/etc/group` sono state sostituite le righe:

```

```

con

```

```

per poter utilizzare rispettivamente la porta seriale e la telecamera.

[2] Vedi cap 2.1 Controllo web

8. Descrizione dei programmi presenti su Morgul

Viene presentata una descrizione di tutti i programmi e gli script trasferiti sul nuovo calcolatore cercando di mettere in luce le operazioni che ogni software comanda al robot.

8.1. Script in */home/morgulweb/script*

- ***emergencyStop.sh***
  Mette in atto uno stop di emergenza e pronuncia il relativo messaggio.
  Per funzionare chiama il programma "`morgulc`" con l'attributo "`-lr`" in modo da spegnere il robot e sbloccarlo.

- ***checksetlock.sh***
  Il comando lock serve a fornire ad un unico utente il controllo esclusivo del robot, ovvero permette ad un unico indirizzo ip di operare sul robot bloccando l'accesso a tutti gli altri. Questo script verifica se il robot si trova in stato locked andando a controllare dentro il file "`LOCKFILENAME`" la variabile "`STATUSDIR`". Se il file non esiste, ne crea uno e affida il controllo esclusivo al calcolatore che ha invocato "`checksetlock.sh`". Se il file esiste, controlla il bit di stato: se è pari a 0 significa che l'utente ha già l'accesso esclusivo, mentre se vale 3 allora l'utente non è abilitato ad accedere al robot.

- ***MAKEDIST.sh***
  Permette di creare file Tarball e accetta come argomenti il nome e la versione del programma di cui si vuole creare l'archivio.

- ***unlock.sh***
  Viene invocato ogni volta in cui non è più necessario un controllo esclusivo del robot. Questo script controlla se è il robot è locked e in tal caso lo sblocca.

- ***onrobot.sh***
  Accende il robot dopo aver effettuato l'operazione di lock. Utilizza lo script "`checksetlock.sh`" e il programma "`morgulc`"con l'opzione "`-R`".

- ***offrobot.sh***
  Spegne il robot e pronuncia l'annuncio appropriato. Utilizza il programma "`morgulc`"con l'opzione "`-lr`".

- ***exitDock.sh***
  Fa uscire il robot dalla stazione di ricarica dopo aver effettuato il lock. Se il robot è già fuori dalla stazione di ricarica non fa nulla e ritorna uno stato d'errore. Invoca gli script "`checksetlock.sh`" e "`onrobot.sh`" per bloccare e accendere il robot. Successivamente chiama il programma "`exitFromDock`" per far muovere il robot e "morgulc" con l'opzione "`-L`", prima, e "`-l`" poi, per accendere le luci all'inizio della manovra e spegnerle a movimento eseguito.

- ***SETUPDIRECTORIES.sh***
  Imposta varie operazioni per il funzionamento di SAURON. Lanciando questo script viene creata la cartella "`status`" in `/home/morgulweb`. All'interno di questa cartella è stato copiato il file "`rs_sauron.log`", già presente sul vecchio calcolatore.

- ***goHome.sh***
  Permette il rientro automatico del robot nella stazione di ricarica. Se il robot si trova già nella stazione, lo script non fa nulla e restituisce lo stato 0. Si appoggia su "`checkdock`", "`checksetlock.sh`" e "`onrobot.sh`". Per rientrare utilizza i programmi

"beaconFinder" e "morguldock". Nel caso in cui il robot si incastri durante il rientro entra in funzione il programma "recoverdock" e a rientro eseguito viene invocato lo script "offrobot.sh".

- ***patrolmap.sh***
  Questo script comanda al robot il pattugliamento di una certa area e durante il suo percorso scatta fotografie. Accetta come argomenti una mappa in cui fare il pattugliamento e ogni quanti secondi scattare la foto.
  Inizialmente chiama alcuni script per uscire dalla stazione di ricarica ("checkdock", "checksetlock.sh" e "onrobot.sh"). Per compiere il percorso invoca il programma "patrol_map" e si appoggia al software "photoreporter".
  A pattugliamento terminato rientra nel dock sfruttando lo script "goHome.sh".

- ***autopatrolmap.sh***
  Script basato su "patrol_map". Esegue le stesse funzioni.

- ***copiaFoto.sh***
  Copia le foto scattate durante l'ultima pattuglia in una cartella denominata *"OLDPICTURESFOLDER"* e le nomina in base alla data e all'ora in cui sono state scattate.
  Invoca lo script "robotparameters.sh"che associa ad "OLDPICTURESFOLDER" la cartella /home/morgulweb/public_html/pictures/oldpatrols, copiata dal vecchio calcolatore.

- ***robotlogrotate.sh***
  Esegue una rotazione dei file di log del robot. È progettato per essere chiamato periodicamente.

- ***robotparameters.sh***
  Contiene una serie di parametri che utilizzano altri script o programmi. Tra i parametri che contiene, a titolo di esempio, si può citare "ROBOTNAME", che indica il nome del robot e "SCRIPTDIR", la cartella contenente gli script.

8.2. Script in */usr/bin*

- ***checkbatt***
  Controlla lo stato della batteria del computer. Quando è chiamato da un altro programma Restituisce 0 se la batteria è al di sopra della soglia, e 256 se si trova sotto, dove la soglia è pari a 1000 mAh.

- ***checkdock***
  Verifica se il robot si trova o meno nella stazione di ricarica controllando se è collegato all'alimentazione. Ritorna 1 quando il robot è nel dock e 0 se è fuori.

- ***handdrive***
  Esegue il programma Aria "demo" fornito con la suite di Aria. Questo script consente al robot di essere pilotato dalla tastiera del pc. Prima di lanciare il demo accende il robot e a movimenti conclusi, spegne il robot appoggiandosi al programma "morgulc".

- ***photoreporter***
  Scatta una serie di fotografie, una ogni t-secondi (dove t è un parametro selezionabile) e le memorizza in una determinata cartella. Dopo aver scattato un certo numero di foto (49 di default) comincia a sovrascrivere quelle esistenti. Le immagini sono prese dalla webcam posizionata sopra il robot e vengono memorizzate in formato jpeg(320x240).

Richiede il file `timestamp.php`, copiato nella cartella `/usr/bin` e necessita dei programmi vgrabbj e setpwc.
Infine, ha bisogno della cartella `/home/morgulweb/public_html/pictures` e di alcuni file contenuti in essa: "`blackimage_sif.jpeg`", "`index.html`" e "`webcam.jpeg`".

- ***photoreporter-panorama***
  Scatta una serie di fotografie , una ogni t-secondi (dove t è un parametro selezionabile) e le memorizza in una determinata cartella. Dopo aver scattato un certo numero di foto (49 di default) comincia a sovrascrivere quelle esistenti. Le immagini sono prese dalla webcam posizionata sopra il robot e vengono memorizzate in formato jpeg(320x240).
  Ha bisogno di una cartella (`/home/morgulweb/public_html/pictures-panorama`), che deve contenere inizialmente il file `black_image_qcif.jpg`, e di altri programmi quali `timestamp.php`, `uvccapture` and `uvcdynctrl`.

8.3. Programmi in */usr/bin*

- ***exitFromDock***
  Programma che fa uscire il robot dalla stazione di ricarica. Dopo un movimento rettilineo il robot si ferma e compie mezzo giro su se stesso. La nuova posizione che il robot ha assunto viene poi inviata al database.

- ***patrol_map***
  Guida il robot tramite un percorso predefinito aggiornando continuamente il database sulle nuove posizioni che assume nell'ambiente in cui si trova.

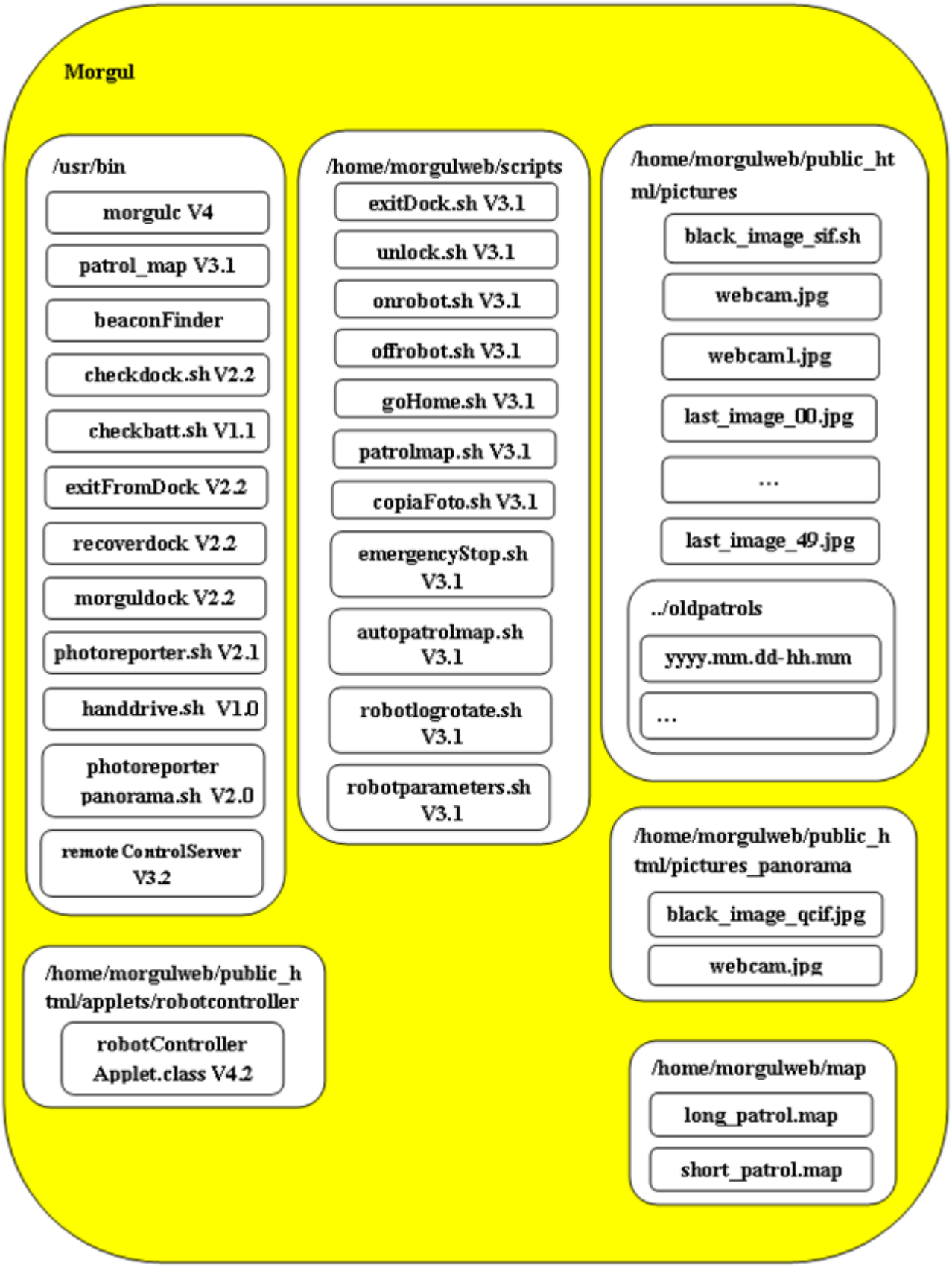
- ***recoverdock***
  Seleziona la strategia migliore per riportare il robot nella stazione di ricarica quando esso urta le pareti della stazione in fase di rientro. A seconda del bumper che ha urtato, un algoritmo sceglie la migliore opzione per portare il robot nel dock.

- ***remoteControlServer***
  Questo programma ascolta su una determinata porta per ricevere le richieste di un applicativo per la guida manuale presente sul server *frost.ing.unibs.it*. Questo applet permette di pilotare manualmente, da remoto, il robot. Durante il movimento, il robot aggiorna continuamente la sua posizione e la invia al database.

- ***morguldock***
  Software che fa rientrare il robot nella stazione di ricarica, aggiornando la sua posizione sul database.

- ***beaconfinder***
  Grazie a questo programma il robot può orientare la sua posizione in base all'allineamento delle luci poste sopra la stazione di ricarica. Quando le luci sono perfettamente allineate il robot capisce di essere nella direzione giusta e prosegue dritto verso la stazione altrimenti effettua alcune manovre per allinearsi al parcheggio.

Mappa dei programmi presenti su Morgul



9. Conclusioni e sviluppi futuri

Sostituire il vecchio calcolatore con uno più moderno ci ha permesso di conoscere ed indagare a fondo il reale funzionamento di un robot mobile. Non è stato semplicemente un trasferimento di file e programmi, ma soprattutto un'analisi dei software, di come inviano i comandi al robot, delle capacità di Morgul e in generale di tutto il sistema di controllo del robot, dal pc dell'utente fino al pc del robot.
Il nuovo calcolatore installato funziona correttamente ed esegue tutte le funzioni del suo predecessore, tuttavia ci sono alcuni programmi che necessitano di un miglioramento.
In particolare, non funziona ancora il lock del robot, ovvero non si riesce ad ottenere il controllo esclusivo del robot tramite il proprio indirizzo ip e l'applet per la guida manuale, utilizzabile dall'indirizzo *http://frost.ing.unibs.it* non viene eseguita sempre correttamente provocando problemi nella guida del robot. Quando sul pannello web l'applet è invocata la prima volta, viene eseguita correttamente, ma se viene invocata una seconda volta non riesce più a muovere il robot ed è necessario ricaricare la pagina web per poter riutilizzare l'applet. È un problema che esisteva già sul vecchio calcolatore e probabilmente non dipende dal pc di Morgul: eseguendo l'applicativo remoteControlServer direttamente da Morgul, senza richiamare l'applet dal portale di Sauron, parte il demo e tutto funziona correttamente.
Se invece si utilizza l'applet sul portale di Sauron a volte non si carica completamente l'interfaccia grafica e a quel punto non è possibile muovere il robot. L'unica soluzione è ricaricare la pagina.
La correzione di questi programmi potrebbe essere uno spunto per successivi progetti.

10. Bibliografia

[1] **"Sistemazione del controllo web di Sauron"** *di Emanuel Bonusi, Francesco De Rose, Nicola Ferrari*

[2] **"Riorganizzazione del database Sauron"** *di Oscar Venturini, Stefano Bennati, Giacomo Negretti, Simone Frassanito*

[3] **"Emulazione e gestione di una porta parallela di PC V4.0"** *di Riccardo Cassinis*

[4] **"ARL-TR-12-01"** *di Riccardo Cassinis*

[5] **"ARL-TR-07-02"** *di Riccardo Cassinis*

Indice