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

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica
(Prof. Riccardo Cassinis)

M.A.R.M.O.T.Cam

Elaborato di esame di: **Stefano Caprini, Marcello Mancini, Michele Scaroni**

Consegnato il: **10 Luglio 2003**

Sommario

L'obbiettivo del presente elaborato è l'installazione sul robot MARMOT di una webcam per l'acquisizione di immagini e la trasmissione, tramite wavelan, ad un calcolatore remoto. Per il funzionamento della webcam, dotata di interfaccia USB, sono stati installati sul PC104 di MARMOT i driver necessari ed il software di acquisizione ed invio dei frame in formato JPEG. Sono stati inoltre configurati il server SSH per la gestione remota della telecamera e la connessione di rete ad un server FTP incaricato di immagazzinare le immagini e di renderle eventualmente disponibili per un successivo utilizzo.

1. Introduzione

Per svolgere il lavoro assegnatoci siamo partiti da un'attenta analisi della documentazione relativa ad un precedente elaborato riguardante l'installazione di una telecamera nel Laboratorio di Robotica Avanzata. La relazione esaminata, sebbene riguardasse l'installazione di una webcam con interfaccia parallela su un calcolatore fisso, ci ha fornito importanti informazioni circa i siti internet dove poter reperire i driver per Linux delle webcam più diverse, riguardo i diversi tipi di software disponibili per l'acquisizione di immagini e la trasmissione e per quanto concerne la configurazione del kernel di Linux per la gestione delle telecamere con interfaccia USB.

2. Il problema affrontato

L'installazione di una webcam su MARMOT richiede necessariamente che essa venga collegata al PC104, vale a dire al calcolatore sul quale risiede il software di gestione del robot. Tale unità è realizzata in modo da risultare estremamente compatta, così da poter essere agevolmente trasportata dal robot. Questo naturalmente implica che le sue risorse hardware e di calcolo siano abbastanza limitate: nello specifico il PC104 è dotato di soli 64 MB di memoria RAM e di 32 MB di spazio su disco rigido. Per via di tali limitazioni, su di esso è stato installata una mini-distribuzione di Linux. Questo ha creato non pochi problemi per quanto riguarda i driver delle telecamere.

2.1. Reperimento dei driver

Le webcam candidate ad essere installate sul PC104 erano due: la Logitech QuickCam Pro 4000 e la Logitech QuickCam Zoom. Sfortunatamente la casa produttrice di queste periferiche non ne contempla l'installazione su sistemi Linux,

pertanto non fornisce in alcun modo driver per questo sistema operativo. Di conseguenza abbiamo fatto ricorso ad Internet per cercare dei driver "non ufficiali".

Dopo una breve ricerca su www.google.it con la stringa "+webcam +linux +driver" abbiamo trovato il sito www.smcc.demon.nl/webcam/ (peraltro citato anche nella relazione dell'elaborato in nostro possesso), dedicato interamente all'installazione delle telecamere Philips su sistemi Linux. Qui veniva fornito un elenco con le webcam supportate dai driver resi disponibili e, con sommo sollievo, abbiamo appurato che i modelli a nostra disposizione comparivano nella lista.

Entrando quindi nella sezione "Download" siamo venuti a conoscenza del fatto che il nucleo del driver per webcam, denominato PWC, fa parte del kernel Linux sin dalla versione 2.4.5. Ovviamente questo modulo evolve con l'evolversi del kernel Linux oltre che con l'uscita sul mercato di nuovi modelli di webcam: pertanto le versioni più recenti del driver, che supportano le webcam più recenti, funzionano solo con le ultime versioni del kernel Linux.

Riportiamo di seguito la tabella che elenca i vari modelli di telecamera, la versione modulo PWC che la supporta e la versione del kernel in cui il modulo è incluso:

Camera	PWC version	2.4 kernel	2.5 kernel
PCA 645VC/646VC	8.0	2.4.6	-
PCV C675K/680K/690K	8.0	2.4.6	-
PCV C720K/40[2]	8.10.1	-	-
PCV C730K/740K	8.0	2.4.6	-
PCV C750K	8.1	2.4.9	-
Askey VC010	8.0	2.4.6	-
Creative Labs Webcam 5[2]	8.5	2.4.18	2.5.2
Creative Labs Webcam Pro Ex	8.10.1	-	-
Logitech QuickCam 3000 Pro	8.2	2.4.13	2.5.0
Logitech QuickCam 4000 Pro	**8.8**	**2.4.19**	**2.5.33**
Logitech QuickCam Notebook Pro	8.9	- (2.4.21)[1]	2.5.50
Logitech QuickCam Zoom	**8.9**	**- (2.4.21)[1]**	**2.5.50**
Samsung MPC-C10 Samsung MPC-C30	8.3	2.4.13	2.5.0
Sotec Afina Eye	8.5	2.4.18	2.5.2
Visionite VCS UM100 Visionite VCS UC300	8.10	- (2.4.21)[1]	2.5.50

1: Sono state inviate delle patch per l'inclusione nella versione 2.4.21, ma finché il kernel non viene rilasciato è possibile scaricare il codice per il PWC 8.10.

2: Queste telecamere sono disponibili in varie versioni, alcune sono supportate, altre no.

Successivamente venivano rese disponibili per lo scaricamento le diverse versioni del modulo PWC.

È stato inoltre necessario scaricare il modulo aggiuntivo denominato PWCX, che permette l'acquisizione di immagini ad una maggiore risoluzione, fino a 640x480 pixel: si tratta di un modulo non rilasciato sotto licenza GPL perché include tecnologie proprietarie, pertanto è disponibile solo in versione pre-compilata.

2.2. Scelta della webcam e della versione del kernel

Come già detto in precedenza, le webcam a nostra disposizione erano due: la Logitech QuickCam 4000 Pro e la QuickCam Zoom, del medesimo produttore.

Logitech QuickCam 4000 Pro

Logitech QuickCam Zoom

A causa dei diversi driver richiesti dalle due telecamere, è diventata di primaria importanza anche la versione del kernel installata sul PC104. Infatti qualora la distribuzione installata fosse stata sufficientemente aggiornata, non sarebbe stato necessario aggiornare il modulo PWC, ma sarebbe bastato compilare opportunamente il kernel per attivare il supporto alla webcam desiderata.

A questo punto abbiamo deciso che avremmo utilizzato la QuickCam 4000 Pro per due motivi:

- il primo è che è stata la prima webcam ad essere disponibile e quindi a permetterci di iniziare subito il lavoro.
- in secondo luogo la versione 2.4.21 del kernel non era ancora definita stabile, quindi poteva presentare qualche errore nel codice.

Infatti optando per la sopra citata telecamera potevamo utilizzare una versione stabile del kernel già largamente diffusa (la 2.4.19) ed attualmente installata sui calcolatori presenti nel Laboratorio di Robotica.

Non essendo però la scelta della versione del kernel di nostra competenza, abbiamo contattato il gruppo di lavoro incaricato di installare Linux sul PC104. Una volta descritta loro la situazione ed esposti i requisiti necessari al funzionamento della webcam, molto gentilmente ci hanno garantito la massima disponibilità e collaborazione adottando il kernel più aggiornato disponibile, il 2.4.20.

2.3. Reperimento del software di acquisizione delle immagini

Per quanto riguarda il software per l'acquisizione delle immagini, inizialmente si era pensato di utilizzare "cqcam", adottato per la webcam del Laboratorio di Robotica. Purtroppo, sempre sul sito www.smcc.demon.nl/webcam/, nella sezione "Working software" abbiamo appreso che tale programma, insieme con "gqcam" (versione di cqcam per Server X) ed altri software non sono più funzionanti con le versioni recenti dei driver. Fortunatamente viene però fornita una lista dei programmi compatibili con i driver, che riportiamo di seguito:

1. camE
   - un applicativo per acquisire immagini statiche ed in movimento (http://linuxbrit.co.uk/camE/)
2. Camserv
   - un semplice programma per fornire streaming video attraverso un browser internet (http://cserv.sourceforge.net)
3. CamStream
   - programma di acquisizione video in streaming con interfaccia grafica propria (http://www.smcc.demon.nl/camstream/)
4. FFMpeg
   - una soluzione completa gratuita per trasmettere audio e video su Internet (http://ffmpeg.sourceforge.net/)
5. GnomeMeeting
   - clone di Netmeeting per Gnome (http://www.gnomemeeting.org/)
6. Motion
   - un programma di sorveglianza che rileva il movimento ed attiva vari tipi di allarme (http://motion.sourceforge.net/)
7. Palantir
   - una soluzione interattiva per lo streaming multicanale ottimizzato per server di fascia bassa (http://www.fastpath.it/products/palantir/)
8. QastroCam
   - un semplice strumento per l'osservazione astronomica (http://3demi.net/astro/qastrocam/)
9. RealProducts
   - la nuova versione dei prodotti della Real Networks è finalmente compatibile con i driver PWC (http://www.realnetworks.com)
10. SDLcam
    - programma per la manipolazione delle immagini (http://sdlcam.raphnet.net)
11. VGrabbj
    - un programma per acquisire, analizzare ed esportare immagini nei formati jpeg, pnm o png (http://gecius.de/vgrabbj/)
12. VIC-UCL
    - non più in fase di sviluppo, programma per la videoconferenza (http://www-mice.cs.ucl.ac.uk/multimedia/)
13. Xawtv

- Il più popolare software di acquisizione sia per immagini statiche che per video in streaming (http://bytesex.org/xawtv/)

14 W3Cam

- Una collezione di strumenti per acquisire immagini da una varietà di sorgenti e visualizzarle su una pagina web (http://www.hdk-berlin.de/~rasca/w3cam/)

Abbiamo quindi analizzato tali programmi alla ricerca di quello che fornisse le funzionalità a noi necessarie. La rosa dei software disponibili si è subito ristretta quando abbiamo scartato quelli che si occupavano di fornire flussi video in streaming, filmati in formato compresso (AVI e MPEG) e servizi di videoconferenza.

Tra i restanti, i programmi che risultavano essere più adatti ai nostri scopi erano:

- "camE", che fornisce un gran numero di funzionalità quali l'acquisizione di immagini (ovviamente), le regolazioni di luminosità, contrasto, saturazione e l'invio tramite FTP;
- "VGrabbj" che offre numerose funzionalità non necessarie per il nostro progetto

Pertanto abbiamo optato per il primo, provvedendo a scaricare i sorgenti e le librerie necessarie. In seguito abbiamo compilato il programma per poterlo provare sui calcolatori del laboratorio, ma l'assenza di alcune librerie ci ha impedito di portare a termine il test. È a questo punto che ci siamo accorti, leggendo il codice sorgente del programma, che venivano incluse alcune librerie del server X. Questo rendeva camE inutilizzabile su MARMOT in quanto, dato lo spazio su disco limitato, sul PC104 non sarebbe stato installato X server.

Avevamo quindi bisogno di un altro programma che funzionasse da terminale: si è resa perciò necessaria un'ulteriore ricerca su Google.it con la stringa "+webcam +capture" e, dopo una lunga analisi dei risultati trovati, siamo giunti a quello che è risultato essere il programma ideale: "camsrv", reperibile sul sito http://crash.ihug.co.nz/~elysium/programming.html.

Questo software non solo funziona da terminale, ma non richiede nessuna libreria aggiuntiva per una corretta acquisizione: abbiamo quindi potuto testarlo subito. Abbiamo collegato una webcam presente in laboratorio (Philips Vesta ToUCam Pro, i cui driver erano già installati) ad uno dei calcolatori e, con uno script, abbiamo provato prima l'acquisizione di un'immagine ed il salvataggio sul calcolatore locale, successivamente abbiamo modificato lo script per effettuare anche l'invio tramite protocollo FTP ad un server di prova.

L'esito del test è stato positivo sotto tutti i punti di vista ed ha decretato il successo definitivo di camsrv su tutti gli altri programmi.

2.4. Installazione del driver

Dato che il nostro lavoro e quello del gruppo addetto al PC104 procedevano in parallelo, per collaudare la webcam che sarebbe poi stata installata su MARMOT abbiamo deciso di effettuare alcune prove su uno dei calcolatori del laboratorio.

Su tali computer è attualmente installata una distribuzione Linux Mandrake 9 e, data la versione 2.4.19 del kernel, ci aspettavamo che fosse già supportata la webcam Logitech QuickCam Pro 4000. Invece, dopo un rapido controllo, con il comando

*grep "PWC_VERSION" /usr/src/linux-2.4.19/drivers/usb/pwc**

abbiamo constatato che la versione del modulo PWC era la 8.6, non la 8.8 a noi necessaria.

Abbiamo perciò preso la decisione di aggiornare il driver. Per poter fare questo dovevamo però copiare dei file in una directory di sistema, cosa impossibile per un utente normale. Abbiamo quindi chiesto al prof. Cassinis se poteva istituire un account con maggiori diritti di lettura/scrittura e, gentilmente, ci sono stati forniti i diritti di "sudo".

A questo punto abbiamo potuto aggiornare i file necessari copiandoli nella cartella

/usr/scr/linux-2.4/drivers/usb/

e abbiamo proceduto alla ricompilazione del kernel per rendere attiva la nuova versione del modulo.

I comandi utilizzati sono stati:

sudo make menuconfig

che ha mostrato un menu di configurazione con il quale abbiamo attivato il supporto USB per le webcam Philips (il driver è scritto per le webcam Philips, ma è compatibile con un gran numero di altre telecamere, tra cui quelle Logitech).

Successivamente abbiamo aggiornato tutte le dipendenze dei vari moduli con il comando

sudo make dep

e creato l'immagine del nuovo kernel con

sudo make bzImage

Quindi abbiamo compilato i nuovi moduli con

sudo make modules

Sfortunatamente la compilazione non è andata a buon fine in quando ci è stato ripetutamente segnalato un errore dovuto al parser del compilatore. Nemmeno l'utilizzo di versioni più aggiornate del compilatore gcc (dalla versione 2.95 alla 2.96 ed infine alla 3.2) ha risolto il problema, pertanto abbiamo deciso di scaricare un nuovo kernel da compilare. Consigli da altri utenti Linux ci hanno spinti a scegliere un kernel VANILLA versione 2.4.20, data la sua stabilità. Abbiamo quindi proceduto a scaricare i sorgenti del nuovo kernel dal sito www.kernel.org e a ripetere da capo le procedure sopra illustrate per effettuarne la compilazione.

Al momento di installare i nuovi moduli con

sudo make modules_install

ci è stata però negata la possibilità di portare a termine l'operazione. Abbiamo quindi dovuto fare nuovamente ricorso al prof. Cassinis per avere la password di accesso al calcolatore come root. Siamo così riusciti ad installare i nuovi moduli, e successivamente il nuovo kernel nella posizione adeguata con il comando

make install

Abbiamo verificato che, per poter selezionare il nuovo kernel ad un successivo riavvio del sistema, il file */etc/lilo.conf* contenesse le impostazioni necessarie.

Al riavvio ci è stato segnalato un problema con la configurazione del Virtual File System, segno di un errore nella configurazione del kernel da noi effettuata. Il sistema riusciva comunque ad avviarsi e ci ha permesso di testare la QuickCam Pro 4000 con i relativi driver.

Anche con questa telecamera il programma "camsrv" ha acquisito in modo corretto l'immagine. A questo punto abbiamo realizzato uno script che automatizzasse il procedimento di acquisizione ed invio delle immagini. Per poter eseguire tale script anche su MARMOT abbiamo optato per il bash, componente standard di Linux.

2.5. Reperimento del software per l'invio delle immagini

A causa delle scelte effettuate si è reso necessario l'utilizzo di un programma per la trasmissione via FTP delle immagini. Inizialmente la scelta è caduta su "ftp" perché saremmo stati sicuri di trovarlo già installato su MARMOT, essendo parte della dotazione standard di Linux.

Il nostro obiettivo era realizzare uno script che si occupasse in modo autonomo di eseguire tutto il procedimento di acquisizione ed invio, ma "ftp" si dimostrava inadatto perché in caso di errore (per esempio: 'server irraggiungibile') passava in modalità interattiva, interrompendo il flusso dello script e presentando il prompt dei comandi.

Conseguentemente abbiamo cercato un software alternativo su Internet tramite http://freshmeat.net con la stringa di ricerca "ftp client" ed abbiamo trovato il pacchetto NcFTP che comprende "ncftpput", programma dedicato all'upload di un singolo file su FTP. Questo software corrispondeva perfettamente alle nostre esigenze perché consentiva di trattare all'interno dello script tutte le diverse condizioni di errore tramite la gestione dei "return code".

2.6. Esecuzione remota dello script di acquisizione ed invio

Obiettivo primario dell'elaborato è la possibilità di catturare immagini dalla webcam montata su MARMOT da un calcolatore remoto. Per questo motivo abbiamo pensato che la soluzione ideale fosse l'utilizzo di un server SSH sul PC104 in modo che, da un calcolatore remoto si potesse effettuare il login su tale macchina, per eseguire lo script da noi realizzato.

Abbiamo quindi effettuato una prima prova sui calcolatori del laboratorio strutturata in questo modo:

- abbiamo copiato "camsrv" su Golemnew nella home directory dell'account marmotcam
- abbiamo collegato la webcam Philips Vesta ToUCam Pro a Golemnew
- abbiamo effettuato il login su Nofivenew come eserc2
- su Nofivenew abbiamo eseguito il comando *ssh golemnew –l marmotcam*
- a questo punto ci è stato richiesto di introdurre la password di tale account
- inserita la password abbiamo avuto accesso a marmotcam dall'altro calcolatore (Nofivenew)
- successivamente abbiamo lanciato camsrv con gli opportuni parametri (–d 3 –r VGA –f click.jpg)

L'immagine è stata così correttamente acquisita e salvata nella home directory dell'account marmotcam.

La prova ha avuto esito positivo, ma nelle nostre intenzioni volevamo che il login tramite SSH fosse il più semplice possibile e che non fosse richiesto all'utente di inserire la password. Abbiamo quindi pensato che la soluzione ideale fosse quella di creare un utente privo di password sul calcolatore remoto.

3. La soluzione adottata

Vediamo ora nello specifico i programmi e le scelte fatte per assolvere il compito assegnatoci.

La webcam che abbiamo deciso di adottare è la Logitech QuickCam 4000 Pro (nella foto a lato) in quanto immediatamente disponibile (la Logitech QuickCam Zoom è stata consegnata solo due settimane più tardi).

Logitech QuickCam 4000 Pro

Inoltre le due telecamere si sono rivelate essere differenti solo nella dotazione software che permette per la prima di effettuare acquisizioni con una risoluzione di 1024x768 interpolata, per la seconda di realizzare uno zoom digitale sul soggetto inquadrato. Essendo questi software non disponibili per Linux, si è scoperto quindi che, per il compito da svolgere, le due telecamere avevano le medesime funzionalità hardware ed erano del tutto equivalenti.

La scelta del modello di telecamera ha naturalmente influenzato anche quella dei driver che è necessariamente ricaduta sul modulo PWC v. 8.8 (come dalla tabella riportata a pag. 2), già compreso nel kernel v. 2.4.20, e sul relativo modulo aggiuntivo PWCX per l'acquisizione in alta risoluzione.

Il software adottato comprende “camsrv” per l’acquisizione di immagini statiche e “ncftpput” del pacchetto NcFTP per l’invio di singoli file tramite protocollo FTP. Per le informazioni relative all’utilizzo del software rimandiamo il lettore al capitolo 4 dove vengono definite tutte le istruzioni e le opzioni necessarie.

Per la connessione FTP con frank.ing.unibs.it abbiamo usufruito di un account dedicato con login “cameras” e password “cameras” con diritto di scrittura nella cartella /Frank HD/ftp/pub/cameras, nella quale abbiamo quindi salvato le immagini.

Per poter effettuare l’acquisizione e l’invio delle immagini in modo automatico abbiamo realizzato il seguente script bash:

```
#----------------------------- SCRIPT CLICK -------------------------------
#! /bin/bash

# ~~ click ~~
versione="2.0 (01/07/2003)"
# by Caprini Mancini Scaroni
# Acquisisce un'immagine dalla webcam e la spedisce all'ftp specificato
# usando ncftpput (http://www.ncftp.com/)

# History delle versioni:
# 2.0 (01/07/2003):
#  - Modificati i parametri per la connessione ftp per funzionare su
#    MARMOT, senza DNS.
# 1.4 (21/06/2003):
#  - Aggiunta l'opzione per cancellare l'immagine prodotta da camsrv
#    dopo il suo trasferimento sull'ftp.
# 1.31 (21/06/2003):
#  - Ora l'output dei comandi camsrv e ncftpput è completamente eliminato
#    nell'uso normale, mentre viene visualizzato usando la
#    modalità  verbose.
# 1.3 (21/06/2003):
#  - Aggiunta la rilevazione di errori generati da camsrv e ncftpput;
#  - In caso di terminazione anomala il programma esce con un returncode
#    appropriato.
# 1.2 (20/06/2003):
#  - Lo script ora usa ncftpput invece di ftp.
# 1.1 (20/06/2003):
#  - Aggiunta la funzionalità  "verbose" per aumentare l'output a console.
# 1.01 (19/06/2003);
#  - Bug fixing.
# 1.0 (19/06/2003):
#  - Prima release.

# ATTENZIONE!
# Lo script va configurato cambiando le righe seguenti e SOLO queste!
# Path completo del comando camsrv
path_to_camsrv="/bin/camsrv"
# Device della webcam
video_device="/dev/video0"
# Path completo del comando ncftpput
path_to_ncftpput="/bin/ncftpput"
# Opzioni aggiuntive da passare al comando ncftpput
ncftpput_opzioni=-V
# Comando per cancellare un file
cancella="rm -f"
```

```
# ATTENZIONE!!! Non modificare da qui in poi!!!
# Definisce le variabili di default
host=192.167.20.94
dir="Frank HD/ftp/pub/cameras"
nomefile="click.jpg"
ritardo=3      # La webcam ha bisogno di alcuni istanti per funzionare
username=cameras
password=cameras
larghezza=640
altezza=480
verbose_output=0

CANCELLA_FILE=0

# Definizione degli exit-codes
# Uso errato degli argomenti del comando:
E_BADARGS=65
# Acquisizione non riuscita a causa di un problema con la webcam:
E_WEBCAMFAIL=66
# Trasferimento ftp fallito:
E_FTPFAIL=67

# Funzione per scrivere il messaggio di aiuto del comando:
scrivi_help()
{
   echo "click by Caprini Mancini Scaroni - versione $versione"
   echo "Cattura un'immagine dalla webcam e la invia ad un ftp."
   echo "Opzioni:"
   echo "-r RISOLUZIONE    L'immagine sarà  acquisita ad una delle
seguenti"
   echo "            risoluzioni:"
   echo "            HI  = 640x480 (default)"
   echo "            MED = 320x240"
   echo "            LO  = 160x120"
   echo "-d ritardo  Acquisisce l'immagine dopo 'ritardo' secondi"
   echo "            (default: $ritardo)"
   echo "-H ftp.host Spedisce l'immagine in ftp all'host indicato"
   echo "            (default: $host)"
   echo "-D path/to/dir    Copia il file sull'host nella directory "
   echo "            specificata (default: $dir)"
   echo "-f nomefile.jpg   Crea localmente un file col nome specificato"
   echo "            (default: $nomefile)"
   echo "-y          Cancella il file creato localmente dopo averlo"
   echo "            spedito con successo all'host"
   echo "-u username Effettua il login all'ftp usando 'username'"
   echo "            come username (default: $username)"
   echo "-p password Effettua il login all'ftp usando 'password'"
   echo "            come password (default: $password)"
   echo "-v          Modalità  prolissa (verbose)"
   echo "-h          Stampa questo messaggio"
}
# Parsing degli argomenti
while getopts ":r:d:H:D:f:yu:p:vhV:" opzione
do
   case $opzione in
   "r" ) case $OPTARG in
               "HI" ) larghezza=640 ; altezza=480;;
```

```
               "MED") larghezza=320 ; altezza=240;;
               "LO" ) larghezza=160 ; altezza=120;;
               *    ) echo "Risoluzione non prevista." ; scrivi_help ; exit
$E_BADARGS;;
         esac;;
   "d" ) ritardo=$OPTARG;;
   "H" ) host=$OPTARG;;
   "D" ) dir=$OPTARG;;
   "f" ) nomefile=$OPTARG;;
   "y" ) CANCELLA_FILE=1;;
   "u" ) username=$OPTARG;;
   "p" ) password=$OPTARG;;
   "v" ) verbose_output=1;;
   "h" ) scrivi_help ; exit;;
   "V" ) video_device=$OPTARG;; # Nota: opzione nascosta perchè presente
solo ai fini del debug
   *   ) echo "ERRORE: Parametro non riconosciuto." ; scrivi_help ; exit
$E_BADARGS;;
   esac
done
# Scrive a console i parametri generati
if [ $verbose_output -eq 1 ]
then
   echo "click (versione $versione) in funzione!!!"
   echo
   echo "Il programma opera usando i seguenti parametri:"
   echo "------------|---------------------------"
   echo "ritardo     = $ritardo"
   echo "file        = $nomefile"
   echo "risoluzione = $larghezza x $altezza"
   echo "host        = $host"
   echo "directory   = $dir"
   echo "username    = $username"
   echo "password    = $password"
fi

if [ $verbose_output -eq 1 ]
then
   echo
   echo Acquisizione immagine...
   echo
   echo Eseguo il comando:
   echo "$path_to_camsrv -d $ritardo -f $nomefile -W $larghezza -H
$altezza -D $video_device"
   $path_to_camsrv -d $ritardo -f $nomefile -W $larghezza -H $altezza -D
$video_device
else
   # Esegue il comando nascondendo l'output
   $path_to_camsrv -d $ritardo -f $nomefile -W $larghezza -H $altezza -D
$video_device >/dev/null
fi
# Controllo se l'esecuzione è andata a buon fine usando il return code del
# comando precedente ($?)
if ! [ $? -eq 0 ]    # Se il return code è diverso da zero...
then
   if [ $verbose_output -eq 1 ]
   then
         echo
```

```
         echo "Errore nell'acquisizione dell'immagine!"
         echo "Possibili cause:"
         echo "- Errore nella configurazione della webcam;"
         echo "- Il device $video_device non corrisponde alla webcam;"
         echo "- Webcam spenta o non funzionante."
         echo
         echo "Fine esecuzione."
   fi
   # Esco segnalando che è avvenuto un errore
   exit $E_WEBCAMFAIL
fi

if [ $verbose_output -eq 1 ]
then
   echo
   echo Trasferimento via ftp...
   echo
   echo Eseguo il comando:
   echo $path_to_ncftpput $ncftpput_opzioni -u $username -p $password
$host "$dir" "$nomefile"
   echo
   $path_to_ncftpput $ncftpput_opzioni -u $username -p $password $host
"$dir" "$nomefile"
else
   # Esegue il comando nascondendo l'output
   $path_to_ncftpput $ncftpput_opzioni -u $username -p $password $host
"$dir" "$nomefile" 2>/dev/null
fi

# Controllo se l'esecuzione è andata a buon fine usando il return code del
# comando precedente ($?)
if ! [ $? -eq 0 ]    # Se il return code è diverso da zero...
then
   if [ $verbose_output -eq 1 ]
   then
         echo
         echo "Errore nel trasferimento dell'immagine sull'ftp!"
         echo
         echo "Fine esecuzione."
   fi
   # Esco segnalando che è avvenuto un errore
   exit $E_FTPFAIL
fi
if [ $CANCELLA_FILE -eq 1 ]
then
   if [ $verbose_output -eq 1 ]
   then
         echo "Adesso cancello il file $nomefile..."
   fi
   # Cancella il file
   $cancella $nomefile
fi
if [ $verbose_output -eq 1 ]
then
   echo
   echo Esecuzione terminata!
   echo "Questo programma vi è stato offerto da:"
   echo "Caprini Mancini Scaroni - gruppo Marmotcam"
```

```
   echo
   echo "Grazie e arrivederci! :-)"
   echo
fi
#------------------------------ FINE SCRIPT -------------------------------
```

Questo script risiede sul calcolatore di MARMOT sul quale sono collocati anche i programmi di acquisizione ed invio delle immagini. Per poter quindi eseguire lo script l'utente deve accedere a MARMOT tramite SSH. A tale scopo abbiamo creato sul PC104 un account specifico con userid "webcam", ma senza password, con diritti di lettura su /dev/video0 (l'indirizzo della webcam). In questo modo per accedere a MARMOT tramite SSH basta introdurre il nome utente. L'accesso è così garantito per qualsiasi utente da qualsiasi calcolatore del laboratorio di robotica, eseguendo semplicemente il comando

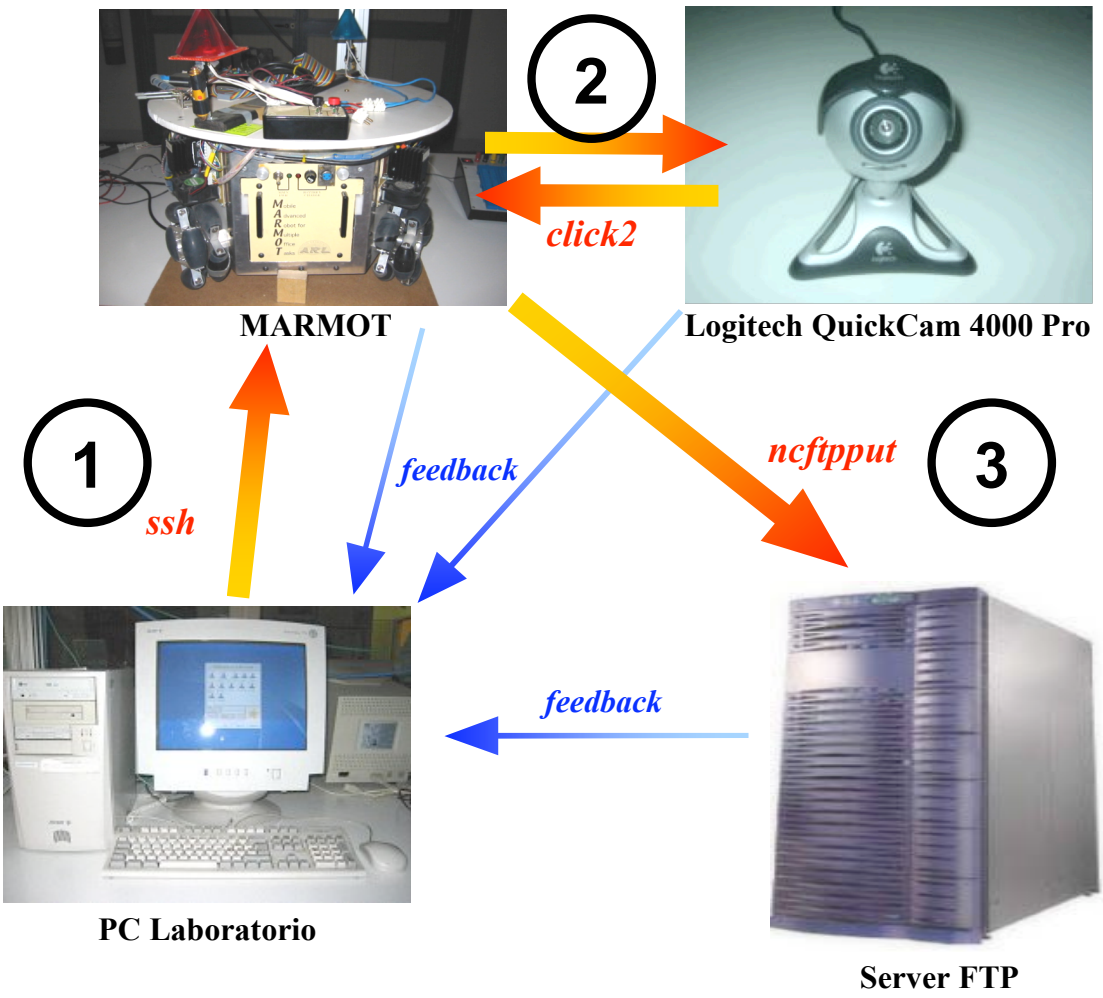
ssh 192.0.2.8 –l webcam

(il comando è "*ssh 192.0.2.7 –l webcam"* se si usa un cavo di rete anziché il collegamento wireless)

Abbiamo lavorato in tale direzione anche in quanto il robot ha un indirizzo IP statico che è visibile solo all'interno della rete del laboratorio, pertanto era inutile proteggere il sistema da accessi esterni non autorizzati.

3.1. Schema di funzionamento

Cerchiamo ora di rappresentare schematicamente come funziona concretamente l'intero sistema di acquisizione e trasmissione delle immagini.



*NOTA: Abbiamo indicato in **rosso** i vari comandi che mettono in comunicazione i diversi componenti, mentre abbiamo indicato in **blu** le retroazioni fornite dai messaggi che, comparendo a video, informano l'utente sull'esito delle varie fasi dell'esecuzione.*

1. Il PC del laboratorio accede a MARMOT tramite protocollo SSH. Eventuali errori nel collegamento remoto vengono segnalati a terminale (1° feedback).
2. MARMOT esegue lo script click2 che acquisisce l'immagine dalla webcam e la salva localmente sul PC104. Se si verificano dei problemi con la webcam (non presente o non correttamente configurata), vengono mostrati a video dei messaggi d'errore sulla terminazione anomala dell'esecuzione (2° feedback).
3. A questo punto l'esecuzione dello script lancia (sul PC104) il comando "ncftpput", inviando l'immagine salvata in precedenza ad un server FTP, nel nostro caso Frank. Anche in questo caso un eventuale errore viene segnalato all'utente (3° feedback).

3.2. Risultati prodotti

Riportiamo di seguito alcune foto di prova ottenute durante le varie fasi di test. Tutte le immagini sono state acquisite nel laboratorio di robotica con la Logitech QuickCam 4000 Pro.

Fig.1a: Corridoio antistante il laboratorio (risoluzione 640x480)

Fig.1b: Corridoio antistante il laboratorio (risoluzione 320x200)

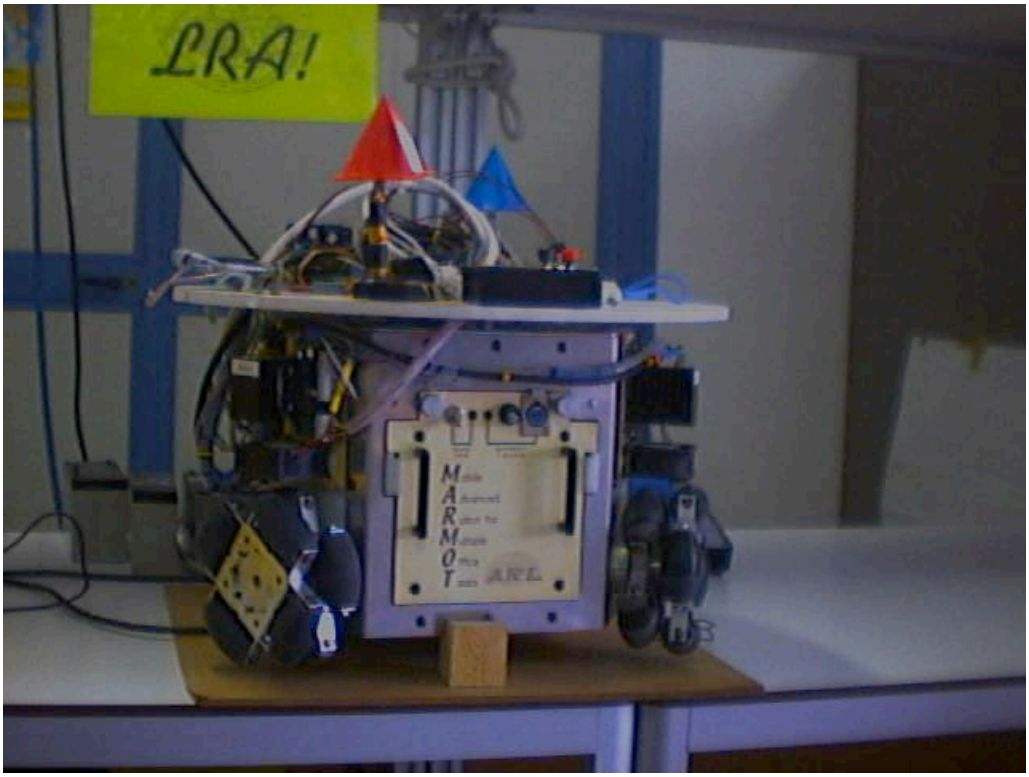


Fig.2: MARMOT, il robot che trasporterà il PC104 e la webcam (640x480)

Fig.3: Il team di lavoro all'opera durante la fase di test (640x480)

4. Modalità operative

Vediamo ora nel dettaglio quali sono le procedure seguite nello svolgere il lavoro.

4.1. Componenti necessari

4.1.1. Componenti hardware

Innanzitutto è necessario disporre di una telecamera compatibile con il modulo PWC 8.8, noi abbiamo adottato la Logitech QuickCam 4000 Pro.

In secondo luogo sono necessari un PC104 con porta USB, un server FTP ed un calcolatore remoto, tutti connessi in rete.

4.1.2. Componenti software

La dotazione software dei vari componenti deve essere la seguente:

- PC104
  - Linux kernel v.2.4.20
  - interprete dei comandi bash
  - modulo PWC v.8.8 per il riconoscimento della webcam
  - modulo PWCX v.8.2.2 per l'acquisizione in alta risoluzione
  - demone "sshd" per la gestione del collegamento remoto
  - "camsrv" per l'acquisizione dell'immagine
  - "ncftpput" per l'invio via FTP del file
  - "click2" per l'esecuzione automatica dei comandi necessari
- Server FTP
  - server ftp
  - un account con accesso in scrittura
- Calcolatore client
  - "ssh" per l'esecuzione remota dello script click2

4.2. Modalità di installazione

Collegare la webcam al PC104, copiare i moduli e disporre i programmi come segue:

- Copiare il file "pwcx-2.4.20.o" nella directory */lib/modules/drivers/usb/*
- Copiare "camsrv", "ncftpput" e "click2" in */bin/*
- Aggiungere al file "rc.modules" la riga:

  insmod –f /lib/modules/drivers/usb/pvcx-2.4.20.o

  per ottenere il caricamento del modulo PWCX all'avvio del calcolatore

- Aggiungere nel file di configurazione di sshd chiamato "/usr/local/etc/ssh/sshd_config" le righe: *PasswordAuthentication yes* e *PermitEmptyPasswords yes* che servono per consentire l'accesso al PC104 senza introdurre la password
- Creare un utente ("webcam" nel nostro caso) sul PC104 usando il comando *useradd <nome_utente>* cancellando successivamente la relativa password dal file "/etc/passwd" e creando una directory home in */home/<nome_utente>*

  In questo modo l'accesso a MARMOT per l'utilizzo della webcam avverrà tramite tale utente e senza l'introduzione della password.
- Per il primo accesso a MARMOT è necessario effettuare il login attraverso un utente dotato di password (ad esempio "root"), in questo modo il server ssh aggiorna i diritti di accesso per il calcolatore dal quale si intende inviare i comandi. Successivamente sarà possibile "loggarsi" su MARMOT da tale calcolatore senza fare più ricorso all'autenticazione.

4.3. Modalità di taratura

L'unica taratura che abbiamo effettuato è quella riguardante il programma "camsrv". Esso infatti permette di impostare un gran numero di opzioni per l'acquisizione dei fotogrammi.

Ad esempio:

-W --width NUM larghezza dell'immagine catturata

-H --height NUM altezza dell'immagine catturata

-r --resmode MODE tipo di risoluzione adottata (QSIF, QCIF, VGA etc.)

-q --jpegquality NUM qualità dell'immagine JPEG (tra 0 e 100, default 75)

-d --delay NUM-NUM | NUM ritardo, in secondi, tra un fotogramma e l'altro

Per quanto riguarda dimensioni e risoluzione dell'immagine, abbiamo già detto che, nelle prove da noi effettuate, le immagini hanno tutte dimensione 640x480 (risoluzione VGA). Va invece sottolineata l'importanza del ritardo nell'acquisizione: esso stabilisce dopo quanto tempo dall'invio del comando viene catturata l'immagine ed eventualmente dopo quanto tempo viene eseguito un nuovo "scatto". Nelle prime foto effettuate avevamo adottato un ritardo di 1 secondo, ma si evidenziavano dei disturbi che creavano strisce colorate in zone casuali dell'immagine.Successivamente abbiamo provato a portare il "delay" a 3 secondi, ottenendo risultati decisamente migliori. Crediamo che questo fenomeno sia dovuto principalmente al tempo di stabilizzazione dell'immagine all'accensione della webcam. Quando infatti viene predisposto il sistema per l'acquisizione, la webcam è inizialmente spenta; all'esecuzione dello script essa si accende, lascia trascorrere il ritardo stabilito e quindi cattura il fotogramma. Se il tempo di attesa è troppo ridotto, l'immagine viene acquisita non appena la webcam si accende e questo può far sì che l'immagine non sia perfettamente visualizzata e, quindi, riprodotta.

4.4. Modalità di utilizzo

Riportiamo di seguito le informazioni necessarie all'utilizzo del software da noi impiegato.

4.4.1. click2

Lo script di acquisizione delle immagini può ricevere in ingresso alcuni parametri opzionali che modificano il comportamento di default.

La sintassi per l'esecuzione è semplicemente

click2 *[opzioni]*

Per avere una descrizione sintetica dei parametri disponibili è possibile digitare dal prompt:

click2 -h

Riportiamo ora una descrizione più dettagliata delle varie funzionalità:

-r: RISOLUZIONE	L'immagine può essere acquisita con le seguenti risoluzioni. HI = 640x480 (default) MED = 320x240 LO = 160x120
-d: RITARDO	Acquisisce l'immagine dopo un tempo di 'RITARDO' secondi. Il ritardo di default è 3 secondi.
-H: SERVER FTP	Spedisce l'immagine in ftp all'host indicato. L'host di default è 192.167.20.94, che corrisponde a frank.ing.unibs.it
-D: PERCORSO	Copia il file sul server FTP nel percorso indicato. La directory di default è /Frank HD/ftp/pub/cameras/
-f: NOME FILE	Acquisisce l'immagine e la salva sul calcolatore con il nome specificato. Il nome di default è "click.jpg" E' però possibile salvare l'immagine anche in formato PPM.
-y: CANCELLA	Cancella il file creato localmente dopo averlo spedito con successo all'host
-u: NOME UTENTE	Effettua il login all'ftp usando il nome utente specificato. Il valore di default è "cameras"
-p: PASSWORD	Effettua il login all'ftp usando la password indicata, in caso non venga specificata viene usato il valore "cameras"
-v: FEEDBACK	Attiva la modalità prolissa nella quale vengono forniti a video i messaggi relativi all'esecuzione e ad eventuali situazioni di errore
-h: HELP	Mostra a video un breve aiuto su questi parametri

Nota: lo script si chiama "click2" in quanto costituisce la seconda versione prodotta. La prima versione utilizzava il client ftp per l'invio delle immagini, quest'ultima fa invece ricorso a ncftpput.

4.4.2. camsrv

Questo è il programma che acquisisce effettivamente le immagini dalla webcam. Tale software è costituito da due parti distinte:

- il server che può funzionare in modalità stand-alone (con output nei formati JPG e PPM) oppure come sorgente di un flusso continuo di immagini
- il client, basato sul linguaggio Java, che si occupa di ricevere e mostrare ciò che viene ripreso e che può essere posto in una pagina web

Per i nostri obbiettivi l'unico componente di interesse è stato il server camsrv in quanto dovevamo acquisire immagini fisse. Il client è invece totalmente dedicato ai filmati in streaming e non si è dimostrato di alcuna utilità.
La sintassi del comando è:

camsrv *[opzioni]*

Riportiamo di seguito la lista completa delle opzioni disponibili:

-v --verbose	mostra a video, durante l'esecuzione, molte informazioni
-W --width NUM	larghezza dell'immagine catturata
-H --height NUM	altezza dell'immagine catturata
-r --resmode MODE	tipo di risoluzione adottata (QSIF, QCIF, VGA etc.)
-f --file NOMEFILE	save PPM/JPG image of first frame captured and exit
-q --jpegquality NUM	qualità dell'immagine JPEG (tra 0 e 100, default 75)
-p --script NOMEFILE	esegue NOMEFILE dopo aver catturato ogni fotogramma
-P --port PORT	porta sulla quale eseguire camsrv server
-d --delay NUM	ritardo, in secondi, tra un fotogramma e l'altro
-l --loop	acquisizione continua di immagini statiche. Disponibile se anche lo streaming è abilitato
-L --log NOMEFILE	salva i messaggi di esecuzione in NOMEFILE
-D --device PATH	percorso del dispositivo video da utilizzare (quello di default è /dev/video0)
-S --stream	invia in streaming al client java. Questo è abilitato di default se –file non è utilizzato
-h --help	visualizza help

Opzioni specifiche per webcam Philips:

-F --framerate NUM framerate della webcam (intero tra 4 e 30)

-c --compression NUM compressione della telecamera (0..3)

-s --shutter NUM velocità di scatto (0..65535) default: automatica

-g --agc NUM override del controllo automatico del guadagno (0..65535)

Modalità di risoluzione standard:

sQCIF	128x96
QSIF	160x120
QCIF	176x144
SIF	320x240
CIF	352x288
VGA	640x480

4.4.3. ncftpput

Lo scopo di "ncftpput" è quello di trasferire file tramite linea di comando senza fare ricorso alla modalità interattiva. Ciò consente di utilizzare degli script o altri processi per effettuare la connessione senza passare attraverso programmi FTP interattivi.

Vediamo ora qual è la sintassi da usare:

ncftpput *[opzioni] host_remoto "directory-remota" "file_locale"*

ncftpput *-f login.cfg [opzioni] directory_remota "file-locale"*

ncftpput *-c host_remoto "percorso_remoto" < stdin*

Come appare evidente dalla sintassi dei comandi, il programma è in grado di accettare dei parametri d'ingresso per impostare un gran numero di opzioni. Riportiamo di seguito le opportune "flag":

-u *XX*	utilizza il nome utente *XX* invece che anonimo
-p *XX*	usa la password *XX* con il nome utente indicato
-P *XX*	usa la porta numero *XX* anziché la porta di default (21).
-j *XX*	usa l'account *XX* in aggiunta a login e password
-d *XX*	usa il file *XX* per il debug del login
-a	usa il trasferimento ASCII anziché binario
-m	prova a creare la directory di destinazione prima di trasferire il file

-t *XX*	timeout dopo *XX* secondi
-f *XX*	legge il file *XX* per l'host, l'utente e la password
-A	aggiorna il file remoto anziché sovrascriverlo
-T *XX*	fa l'upload tramite file temporanei con prefisso *XX*
-S *XX*	fa l'upload tramite file temporanei con suffisso *XX*
-R	modalità ricorsiva: copia interi alberi di directory
-r *XX*	ritenta la connessione *XX* volte finché non si connette all'host
-z/**-Z**	effettua il "resume" dei file. Di default non prova e fare il "resume"
-E	usa i dati per una connessione regolare (PORT)
-F	usa i dati per una connessione passiva (PASV). Di default prova a stabilire una connessione passiva, se questa fallisce (o va in timeout), ritenta con una connessione regolare
-DD	cancella il file locale dopo un upload avvenuto con successo
-y	prova con "SITE UTIME" per preservare i timestamp sull'host remoto. Non tutti I server FTP supportano questa funzionalità
-b	esecuzione in background (sottoponendo un "job" a *ncftpbatch*)
-B *XX*	prova ad impostare il buffer del socket TCP/IP a *XX* bytes

5. Conclusioni e sviluppi futuri

Il lavoro svolto consente di acquisire immagini statiche dal PC104 di MARMOT tramite una webcam con interfaccia USB, operando da un qualsiasi calcolatore collegato alla rete del laboratorio di robotica e di inviarle ad un qualsiasi server FTP.

L'acquisizione delle immagini viene attualmente effettuata con un comando lanciato dall'utente. Il nostro lavoro può essere la base di partenza per l'acquisizione continua delle immagini ed il loro utilizzo per la navigazione del robot. Il programma di acquisizione "camsrv" dispone infatti di un client dedicato alla gestione di flussi video continui. Esso è inoltre basato sul Java e può essere facilmente integrato in una pagina web, rendendo così possibile la trasmissione via Internet delle immagini.

6. Ringraziamenti

Si ringraziano il prof. Cassinis per la disponibilità e la celerità nel fornirci materiale ed assistenza, gli studenti Baranzelli e Brignani per l'averci messo a disposizione il PC104 ed aver soddisfatto tutte le nostre richieste.

Appendice A

Acquisizione continua delle immagini

In questa appendice ci occuperemo di testare il sistema adottato per l'acquisizione a ciclo continuo delle immagini.

Per poter fare ciò il sistema di acquisizione ed invio deve essere diviso in almeno tre fasi:

1. accensione della webcam e acquisizione continua (in background) delle immagini
2. invio delle immagini durante l'acquisizione
3. spegnimento della webcam

Per questo motivo il processo di acquisizione deve continuamente fornire immagini al processo di trasmissione perché possa trasferirle ad un server remoto. È ovvio, a questo punto, che questi processi debbano essere del tutto indipendenti per poter essere eseguiti in parallelo. Ciò non avveniva nell'acquisizione di immagini singole, dove tutte le attività erano strettamente sequenziali.

Data la nuova struttura del sistema, emergono alcuni problemi:

- in primo luogo la qualità dell'immagine: nell'acquisizione di foto statiche, infatti, avevamo dovuto impostare un ritardo di 3 secondi per evitare distorsioni e disturbi
- inoltre, per l'acquisizione di fotogrammi in rapida successione, il trasferimento via FTP risulta essere troppo lento per poter inviare le immagini in tempo utile.

Per quanto riguarda l'acquisizione ripetuta di immagini, abbiamo deciso di sfruttare la funzionalità apposita del programma "camsrv" precedentemente adottato, utilizzando l'opzione **–l**. Questa consente infatti l'esecuzione in loop del programma che, in tal modo, continua a catturare i fotogrammi salvandoli su disco con il nome specificato. Va sottolineato che, fornendo nella linea di comando il nome del file di destinazione con il parametro **–f**, l'acquisizione continua ad aggiornare lo stesso file, mantenendo così limitato lo spazio occupato su disco.

Abbiamo a questo punto effettuato le prime prove sui calcolatori del laboratorio utilizzando inizialmente un ritardo di 1 secondo tra un frame e l'altro, verificando che l'immagine non risultasse disturbata. Il comando utilizzato è:

./camsrv –D /dev/video0 –r VGA –f foto.jpg -d 1 –l

A seguito dell'esecuzione abbiamo potuto constatare che i fotogrammi non erano soggetti a distorsione come le immagini statiche e che, a patto di non muovere la webcam troppo velocemente, la qualità e la nitidezza delle immagini era buona.

Per poter però eseguire contemporaneamente anche le operazioni di invio dei file, è necessario che "camsrv" sia eseguito in background. Pertanto abbiamo ripetuto la prova aggiungendo il carattere "&" al termine del comando:

./camsrv –D /dev/video0 –r VGA –f foto.jpg -d 1 –l &

Le prove così effettuate hanno avuto esito positivo sotto tutti i punti di vista.

Successivamente abbiamo provato ad incrementare il frame-rate eliminando il ritardo tra fotogrammi successivi. In questo modo la webcam acquisiva le immagini al ritmo di 15 fotogrammi al secondo. Anche in questo caso le prove hanno dimostrato l'efficacia del sistema.

Per poter terminare l'esecuzione del programma "camsrv" è però necessario fare ricorso al comando "kill", cosa che richiede di conoscere il suo PID (numero identificativo del processo in esecuzione). Linux dispone del comando "ps" che elenca i programmi in esecuzione con i relativi PID: il problema è selezionare solo il Process ID per inserirlo nel comando "kill".

Una prima soluzione, su consiglio dei colleghi Baranzelli e Brignani, è stata quella di realizzare uno script come segue:

```
#! /bin/sh
id=$(ps | grep camsrv)
indice=$[0]
for j in $id
do
      if [$indice=0]
            then
            posizione=$[$j]
      fi
indice=$[1]
done
kill $posizione
```

Questa soluzione non ci lasciava però soddisfatti, pertanto abbiamo cercato un sistema più elegante per ottenere lo stesso risultato.

Abbiamo quindi provato con il comando "awk" che è un programma di scansione ed analisi dei pattern:

kill`ps | awk '$4=="camsrv" {print $1}'`

In questo modo con un solo comando riusciamo a terminare il loop di acquisizione.

Abbiamo quindi realizzato due script, uno per l'avvio del sistema ed uno per terminare l'esecuzione:

```
----------------------- script start_webcam -------------------------
-

#! /bin/sh
camsrv –D /dev/video0 –r VGA –f foto.jpg –l &
```

```
--------------------------- fine script ------------------------------
-

----------------------- script stop_webcam ---------------------------
-

#! /bin/sh
kill `ps | awk '$4=="camsrv" {print $1}'`

--------------------------- fine script ------------------------------
-
```

Abbiamo quindi testato questi script sui calcolatori del laboratorio sia in locale che in remoto, tramite ssh.

L'esecuzione su computer locale si è svolta alla perfezione consentendo l'avvio e la terminazione del sistema ed aggiornando l'immagine con successo. Eseguendo invece lo script di avvio in modalità remota, l'acquisizione delle immagini avveniva correttamente, ma sul calcolatore dal quale era stato lanciato il comando non veniva restituito il prompt. Pertanto era necessario interrompere ssh (con CTRL+C), in questo modo veniva terminata solamente l'esecuzione del client ssh, ma non quella dello script di acquisizione.

Il sistema, sebbene poco elegante, si è dimostrato perfettamente funzionante.

A questo punto abbiamo effettuato le prove su MARMOT copiando i nuovi script sul PC104. Il primo problema che si è presentato in questa fase è stata l'assenza del programma "awk" utilizzato nello script di stop. Abbiamo pertanto dovuto sostituire la riga presente nello script stop_webcam con:

kill `ps | grep camsrv `

Effettuando le prove di acquisizione ci siamo però accorti che l'esecuzione ciclica di "camsrv" veniva terminata dopo breve tempo a causa dell'insufficiente disponibilità di memoria fisica del PC104. Dopo un breve controllo sui calcolatori del laboratorio, eseguendo il comando *top*, abbiamo constatato che il programma necessita di circa 18 MB di memoria fisica (RAM) disponibile. A nulla sono valsi i tentativi di ridurre frame-rate e risoluzione per contenere lo spazio occupato.

Il sistema sarebbe quindi perfettamente funzionante e fornirebbe ottime prestazioni in termini di velocità di acquisizione se fosse possibile ampliare le risorse del PC104.

In alternativa abbiamo studiato una nuova soluzione meno efficace per acquisire un flusso continuo di immagini. Anziché utilizzare lo streaming video come fatto in precedenza, abbiamo fatto ricorso all'acquisizione di singole immagini.

Per ridurre i tempi di acquisizione siamo riusciti ad azzerare il ritardo riducendo la risoluzione dei fotogrammi ed evitando quindi i disturbi che si erano verificati in precedenza su immagini più grandi. In questo modo siamo riusciti a garantire un tempo di acquisizione delle immagini di poco inferiore ad 1 secondo.

Il comando utilizzato è:

camsrv –r QSIF –D /dev/video0 –f foto.jpg –d 0

Per rendere continua l'esecuzione abbiamo realizzato uno script che include questo comando in un opportuno ciclo.

```
#! /bin/bash
while [ 1 -eq 1 ]
do
camsrv -r QSIF -D /dev/video0 -f foto.jpg -d 0
echo click
done
```

Con questo sistema abbiamo potuto realizzare un ciclo infinito di acquisizione di immagini singole con frame-rate di circa 1 fotogramma/s.

Appendice B

Logitech QuickCam Zoom

Questa sezione è dedicata alle prove effettuate con la webcam Logitech QuickCam Zoom. Abbiamo infatti voluto verificare se questa telecamera era compatibile con il sistema da noi realizzato, in particolare se i driver v.8.8 installati erano in grado di supportarla.

Per effettuare le prove abbiamo collegato la suddetta webcam al PC104 ed abbiamo provato ad avviare il programma di acquisizione. Sfortunatamente, sebbene le due telecamere siano pressoché uguali dal punto di vista delle caratteristiche tecniche e il sistema operativo ne rilevi la presenza, il flusso video prodotto dalla QuickCam Zoom non viene riconosciuto come valido.

Alla luce di questi fatti possiamo affermare che per utilizzare questa telecamera è necessario aggiornare i driver alla versione 8.10, come già indicato nella documentazione del modulo PWC.

Bibliografia

Il materiale relativo ai programmi è stato prelevato da Internet. La documentazione relativa a Linux è stata ricavata dai “man files”.

[1] Saetti A., Spinoni S.: “Web-camera per LDRA”, *Relazione relativa ad elaborato per il corso di Robotica*, 2001.

[2] Panteghini M., Cagno M.: “Un sistema distribuito per la localizzazione di robot mobili dotati di marker attivi tramite webcam”, *Tesi di Laurea in Ingegneria Elettronica*, 2002.

Indice