INSTITUT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

SEP

Fernsteuerung einer Videokamera mit Dreh-Schwenkkopf über eine WEB-Schnittstelle

Bearbeiter:	Simon Bichler
Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Martin Garschhammer Helmut Reiser

INSTITUT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

SEP

Fernsteuerung einer Videokamera mit Dreh-Schwenkkopf über eine WEB-Schnittstelle

Bearbeiter:	Simon Bichler
Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Martin Garschhammer Helmut Reiser

Abgabetermin:	15. Juni 2004

Hiermit versichere ich, dass ich das vorliegende Systementwicklungsprojekt selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 15. Juni 2004

. . . .
(Unterschrift des Kandidaten)

Zusammenfassung

Am Lehrstuhl befindet sich eine Videokonferenz-Kamera vom Typ Sony EVI-D31, die mit einem Dreh-Schwenk-Kopf ausgestattet ist. Die Steuerung der Dreh- und Schwenkbewegung sowie der anderen Funktionen der Kamera erfolgt bislang mit der mitgelieferten Infrarot-Fernbedienung.

Da es bei Videokonferenzen wünschenswert ist, daß die entfernte Seite den Bildausschnitt bestimmen kann, wurde im Rahmen dieses Systementwicklungsprojektes die Funktionalität der Fernbedienung in einem Web-Interface umgesetzt.

Die Ansteuerung erfolgt dabei über einen Linux-PC, der das eingebaute VISCA-Interface der Kamera über die serielle Schnittstelle anspricht. Auf diesem PC läuft auch ein Apache-Webserver, der mit Hilfe von PHP das Frontend zur Verfügung stellt.

Die Verteilung des Video-Bildes erfolgt durch einen Open H323 Server. Dieser wird allerdings nicht Gegenstand dieser Ausarbeitung sein.

Inhaltsverzeichnis

Kapitel 1

Einleitung

In dieser Ausarbeitung soll der Entwurf und die Umsetzung eines Systems beschrieben werden, das die Fernsteuerung der am Lehrstuhl befindlichen Videokonferenz-Kamera vom Typ Sony EVI-D31 über ein Webinterface erlaubt. Die steuerbaren Funktionen umfassen den Dreh-/Schwenkkopf, den Zoom, die Belichtungseinstellungen, die automatische Zielverfolgung sowie einige weitere Features, die im Anhang A aufgelistet sind.

Allgemeine Informationen und Links zu Websites über die Sony D30/31 Kameras finden sich unter [1]. Das offizielle Benutzerhandbuch findet sich unter [2].

1.1 Motivation

Die Steuerung der oben genannten Kamera erfolgte bislang durch die mitgelieferte Infrarot-Fernbedienung. Bei Videokonferenzen ist es aber unter anderem wünschenswert, daß die entfernte Seite den Bildausschnitt bestimmen kann. Das ist mit der bestehenden Lösung aber nicht möglich.

Um die Steuerung auch von der entfernten Seite aus zu ermöglichen bietet es sich an, die Funktionen der Fernbedienung in einem Webinterface abzubilden. Der Zugriff auf dieses Interface ermöglicht der entfernten Seite die Steuerung des Bildausschnittes und der anderen Funktionen der Kamera.

1.2 Design & Konzept

Das in dieser Ausarbeitung vorgestellte System besteht aus mehreren Schichten, die aufeinander aufbauen. Die folgenden Schichten können dabei unterschieden werden:

- Serielle Schnittstelle RS232
- libVISCA
- libVISCA-Highlevel-Interface
- VISCA-daemon
- phpVISCA

- php-Frontends

Die Schichten sind jeweils in sich abgeschlossen und werden nur von der nächsthöheren Schicht benutzt, bzw. nutzen selbst jeweils nur die nächstniedrigere Schicht.

Besonderes Augenmerk wurde bei der Entwicklung der Protokolle, mit denen die Schichten untereinander kommunizieren, darauf gelegt, daß menschenlesbare Ascii-Kommandos verwendet wurden. Dies erleichtert einerseits das Debugging, und unterstützt andererseits die Entwicklung neuer Frontends.

Die erste Schicht ist die serielle Schnittstelle, an die die Sony EVI-D31 Kamera mit einem proprietären Kabel angeschlossen wird. Die Pinbelegung des Kabels findet sich auf Seite 17 von [3]. Die Behandlung dieser Schicht erfolgt durch den Linux-Kernel. Im Rahmen des SEPs wurde in dieser Schicht des Systems keine Eigenentwicklung vorgenommen.

libVISCA, die nächsthöhere Schicht, ist eine C-Bibliothek, die das von der Kamera verwendete VISCA-Protokoll umsetzt. Diese Bibliothek wurde im Rahmen des SEP um die Befehle für die EVI-D31 erweitert. Alle höheren Schichten wurden für das SEP von Grund auf neu entwickelt.

Die dritte Schicht, das libVISCA-Highlevel-Interface, abstrahiert die libVISCA und fasst die Einzelfunktionen zusammen.

Der VISCA-daemon schließlich stellt den Zugriff auf das Highlevel-Interface über ein Telnet-artiges Protokoll auf einem Netzwerkport zur Verfügung.

Die php-VISCA Schicht stellt eine API zur Verfügung um mit PHP auf den VISCA-daemon zuzugreifen.

Als oberste Schicht wurden im Rahmen dieses SEP drei verschiedene PHP-Frontends entwickelt: Ein rein informatives Interface, über das keine Änderungen an den Kameraeinstellungen vorgenommen werden können, ein Interface, das die Infrarot-Fernbedienung emuliert und ein Listen-Interface, in dem alle Befehle, die die Kamera unterstützt, aufgerufen werden können.

Im weiteren Verlauf der Ausarbeitung sollen diese Schichten detailiert erläutert werden.

Kapitel 2

libVISCA

Das VISCA-Protokoll ist ein von Sony entwickeltes Protokoll um Kameras und Videorecorder über eine serielle Schnittstelle fernzusteuern.

Um dieses Protokoll von einem Linux-PC aus nutzen zu können entwickelte Damien Douxchomps eine C-Bibliothek, die libVISCA [4]. Sie ermöglicht es Befehle an Kameras zu verschicken, die an eine serielle Schnittstelle des PCs angeschlossen sind und das VISCA-Protokoll beherrschen.

Das VISCA-Protokoll wird von verschieden Kameras unterstützt, die jeweils einen unterschiedlichen Funktionsumfang besitzen. In der libVISCA in der Version 0.06, wie sie zu Beginn des SEPs zur Verfügung stand, waren 139 Befehle implementiert von denen etwa 2/3 von der D31 verstanden wurden. 45 Befehle, die die D31 versteht, waren dagegen in der libVISCA nicht enthalten.

Da die libVISCA von Damien Douxchomps unter die LGPL gestellt wurde war es möglich, die fehlenden Opcodes zu implementieren. Die Opcodes sind in [3] zu finden. Die Erweiterung wurde in die Version 0.07 der libVISCA mit aufgenommen.

Im Anhang findet sich eine Liste der Befehle und ihrer Bedeutung. Dort ist auch ersichtlich, welche Befehle vorhanden waren und welche im Rahmen des SEPs implementiert wurden.

Die interne Funktionsweise der libVISCA soll nicht Gegenstand dieser Ausarbeitung sein. Die Benutzung und die Erweiterung der libVISCA wird jedoch im Folgenden erläutert.

2.1 Verwendung der libVISCA

Da die libVISCA ausser durch Kommentare im Quellcode kaum dokumentiert ist, hier ein paar Worte zur Verwendung der Bibliothek:

Wenn die libVISCA richtig installiert wurde, kann man sie mit folgendem include-Befehl verwenden:

```
#include <libvisca/libvisca.h>
```

Die folgenden Variablen sind zur Verwendung der libVISCA vorzusehen. `ttydev` gibt die serielle Schnittstelle an, an der die Kamera angeschlossen ist. `interface` und `camera` sind libVISCA-interne Variablen, die bei allen Aufrufen der libVISCA wieder benötigt werden und `camera_num` nimmt die Anzahl der angeschlossenen Kamera auf.

```
char *ttydev = "/dev/ttyS0";
VISCAInterface_t interface;
VISCACamera_t camera;
int camera_num;
```

Folgender Code-Abschnitt initialisiert die serielle Verbindung zur Kamera. Dieser Code-Abschnitt ist aus dem `testvisca`-Programm der ursprünglichen libVISCA entnommen.

```
if (VISCA_open_serial(interface, ttydev)!=VISCA_SUCCESS) {
  fprintf(stderr,"unable to open serial device %s\n",ttydev);
  exit(1);
}

(*interface).broadcast=0;
VISCA_set_address(interface, &camera_num);
(*camera).address=1;
VISCA_clear(interface, camera);
VISCA_get_camera_info(interface, camera);
```

Damit ist die libVISCA vorbereitet, um Befehle an die Kamera zu übertragen.

Befehle weisen als Rückgabewert immer entweder `VISCA_SUCCESS` oder `VISCA_FAILURE` auf. Als Parameter müssen immer die oben erwähnten `interface` und `camera` übergeben werden. Je nach Art des Befehls können weitere Parameter dazukommen. Für set-Befehle etwa der Wert, auf den etwas gesetzt werden soll oder für get-Befehle Pointer auf die Variablen, in die die Rückgabewerte abgelegt werden sollen. Hier ein beispielhafter Befehlsaufruf der libVISCA, um die Kamera abzuschalten:

```
int power=0;
if (VISCA_set_power(interface, camera, power)!=VISCA_SUCCESS) {
  fprintf(stderr,"unable to set power to %i\n",power);
  exit(1);
}
```

Hier ein Beispiel für einen Befehl mit Rückgabewert. Die Variable, in der der Wert gespeichert wird, muß je nach Befehl vom Typ `UInt8_t` oder `UInt16_t` sein. Für Boolsche Werte wird entweder `VISCA_ON` oder `VISCA_OFF` zurückgegeben, für numerische Rückgabewerte wird der numerische Wert zurückgegeben.:

```
UInt8_t power;
if (VISCA_get_power(interface, camera, &power)!=VISCA_SUCCESS) {
  fprintf(stderr,"unable to get power");
  exit(1);
}
```

Nach der Benutzung muß die Verbindung zur Kamera wieder abgebaut werden. Der folgende Code ist wiederum aus der `testvisca`-Anwendung der libVISCA übernommen:

```
unsigned char packet[3000];
int i, bytes;

// read the rest of the data: (should be empty)
ioctl((*interface).port_fd, FIONREAD, &bytes);
if (bytes>0) {
  fprintf(stderr, "ERROR:␣%d␣bytes␣not␣processed:␣", bytes);
  read((*interface).port_fd, &packet, bytes);
  for (i=0;i<bytes;i++) {
    fprintf(stderr,"%2x␣",packet[i]);
  }
  fprintf(stderr,"\n");
}
VISCA_close_serial(interface);
```

2.2 Erweiterung der libVISCA

Um die libVISCA um zusätzliche Befehle zu erweitern ist es lediglich notwendig, die entsprechenden Opcodes in einzelne Funktionen zu verpacken.

Hier ein Beispiel, an dem man die wesentlichen Bestandteile einer libVISCA-Funktion gut erkennen kann.

```
unsigned int
VISCA_set_wide_con_lens(VISCAInterface_t *interface, VISCACamera_t *camera,
UInt8_t power)
{
  VISCAPacket_t packet;

  _VISCA_init_packet(&packet);
  _VISCA_append_byte(&packet, VISCA_COMMAND);
  _VISCA_append_byte(&packet, VISCA_CATEGORY_CAMERA2);
  _VISCA_append_byte(&packet, VISCA_WIDE_CON_LENS);
  _VISCA_append_byte(&packet, VISCA_WIDE_CON_LENS_SET);
  _VISCA_append_byte(&packet, power);

  return _VISCA_send_packet_with_reply(interface, camera, &packet);
}
```

Zunächst wird eine Variable vom Typ `VISCAPacket_t` benötigt. Mit `_VISCA_init_packet` wird das Paket initialisiert. Dann werden Byte für Byte die Bestandteile des Opcodes dem Paket hinzugefügt. Der Lesbarkeit halber werden hier keine numerischen Werte verwendet, sondern Makros, die in der Header-Datei der libVISCA definiert wurden. Schließlich wird das fertige Paket mit `_VISCA_send_packet_with_reply` an die Kamera geschickt und die Antwort direkt zurückgegeben.

Hier noch ein Beispiel für die Implementierung eines lesenden Befehls:

```
unsigned int
VISCA_get_wide_con_lens(VISCAInterface_t *interface, VISCACamera_t *camera,
```

```
UInt8_t *power)
{
  VISCAPacket_t packet;
  unsigned int err;

  _VISCA_init_packet(&packet);
  _VISCA_append_byte(&packet, VISCA_INQUIRY);
  _VISCA_append_byte(&packet, VISCA_CATEGORY_CAMERA1);
  _VISCA_append_byte(&packet, VISCA_WIDE_CON_LENS);
  err=_VISCA_send_packet_with_reply(interface, camera, &packet);
  if (err!=VISCA_SUCCESS)
    return err;
  else
    {
      *power=interface->ibuf[2];
      return VISCA_SUCCESS;
    }
}
```

Wieder wird eine Variable vom Typ `VISCAPacket_t` benötigt. Wie bei einem schreibendem Befehl wird das Paket initialisiert und mit den Opcodes bestückt. Nach dem Verschicken des Paketes kann der Rückgabewert der Kamera im `ibuf`-Feld des Interfaces abgerufen werden.

Kapitel 3

libVISCA Highlevel Interface

Die libVISCA stellt die Funktionalität der Kamera auf einer niederen Ebene bereit. Um in höheren Schichten nicht immer die über hundert Funktionen der libVISCA mitschleifen zu müssen ist es sinnvoll, eine Zwischenschicht einzuführen, die die libVISCA abstrahiert.

Das libVISCA Highlevel Interface stellt folgende drei Funktionen bereit

- `VISCA_openInterface`: Öffnet die Verbindung zur Kamera.
- `VISCA_doCommand`: Führt das gewünschte Kommando aus und liefert evtl. Rückgabewerte
- `VISCA_closeInterface`: Schließt das Interface zur Kamera

Die Befehle `open_interface` und `close_interface` umfassen lediglich die im letzten Kapitel vorgestellten Standard-Vorgehensweisen, wie sie in Beispielprogrammen der libVISCA angegeben sind. Der Befehl `doCommand` hingegen umfasst einiges mehr an Funktionalität. Er parst den übergebenen String nach Befehl und Parametern, überprüft den Wertebereich der Parameter passend zum jeweiligen Befehl, führt den Befehl aus und gibt differenzierte Fehlercodes zurück.

Folgende Fehlercodes werden dabei unterschieden:

- 10: Befehl erfolgreich ausgeführt
- 11: Befehl erfolgreich ausgeführt, ein Rückgabeparameter
- 12: Befehl erfolgreich ausgeführt, zwei Rückgabeparameter
- 13: Befehl erfolgreich ausgeführt, drei Rückgabeparameter
- 40: Unbekannter Befehl
- 41: Fehlendes oder nicht im Wertebereich liegender Parameter 1
- 42: Fehlendes oder nicht im Wertebereich liegender Parameter 2
- 43: Fehlendes oder nicht im Wertebereich liegender Parameter 3
- 44: Fehlendes oder nicht im Wertebereich liegender Parameter 4
- 45: Fehlendes oder nicht im Wertebereich liegender Parameter 5

- 46: Fehler beim Senden an die Kamera

Die genaue Signatur der doCommand-Funktion sieht folgendermaßen aus:

```
int VISCA_doCommand(char *commandline, int *ret1, int *ret2, int *ret3,
    VISCAInterface_t *interface, VISCACamera_t *camera) {
```

In `commandline` wird die zu parsende Befehlszeile übergeben, in `ret1` bis `ret3` werden ggf. die Rückgabeparameter gespeichert. `interface` und `camera` sind libvisca-interne Variablen, die den Kontext der seriellen Schnittstelle und der angeschlossenen Kamera enthalten. Sie werden beim Aufruf von `VISCA_openInterface` initialisiert.

Die Implementierung teilt den Befehlsstring zuerst mit Hilfe von `strtok` in Befehl und Parameter auf. Als Trennzeichen wird hier ein blank verwendet. Als nächstes werden die Parameter mit Hilfe der Funktoin `atoi` in Integer-Werte umgewandelt.

Da solche Befehle der libVISCA, die boolsche Eingabeparameter erwarten, den Wert 3 als false und den Wert 2 als true interpretieren, werden in einem nächsten Schritt die Parameterwerte 0 bzw "false" als Wert 3 bzw. die Werte 1 oder "true" als Wert 2 in die Variable `boolarg` abgespeichert. Diese Variable wird später Befehlen übergeben, die einen Boolschen Eingabewert erwarten.

Der Rest von `VISCA_doCommand` besteht aus einer if-Abfrage pro Befehl. Im folgenden ein Beispiel dafür:

```
if (strcmp(command, "set_power") == 0) {
  if ((arg1 == NULL) || (boolarg == -1)) {
    return 41;
  }
  if (VISCA_set_power(interface, camera, boolarg)!=VISCA_SUCCESS) {
    return 46;
  }
  return 10;
}
```

Zuerst wird das Argument auf sein Vorhandensein und seinen Boolschen Wert überprüft. Ist es nicht vorhanden, so wird als Fehlerwert 41 zurückgegeben. Dann wird versucht, den Befehl `VISCA_set_power` auszuführen. Schlägt er fehl, wird 46 als Fehlerwert zurückgegeben. Ist dagegen alles glatt gegangen, so wird 10 als Erfolgswert zurückgegeben.

Manche der if-Abfragen sind von einer Abfrage umgeben:

```
#if !D30ONLY
...
#endif
```

Damit kann man zur Compilezeit die Unterstützung für andere Kameras als die D30/31 abschalten.

Kapitel 4

VISCAdaemon

Um den Zugriff auf die Funktionalität des VISCA-Interfaces netzwerktransparent zu machen wurde ein Daemon entwickelt, der auf einem Netzwerkport auf ankommende Verbindungen wartet. Dabei wurde in großem Unfang auf Beispielcode aus der ”Unix-Socket-FAQ [5]” zurückgegriffen.

Der Daemon kann im Prinzip als beliebiger Benutzer gestartet werden. Wichtig ist lediglich, daß der Benutzer Zugriffsrechte für die serielle Schnittstelle besitzt.

Sicherheitüberlegungen sind in die Entwicklung des Daemons nicht eingeflossen. Es ist dafür zu sorgen, daß der Zugriff auf den Netzwerkport, auf den der Daemon lauscht, nur den Webserver mit dem PHP-Frontend gestattet ist. Für solche Aufgaben existieren Standardwerkzeuge wie etwa `iptables`. Ein selbstgestricktes System könnte sich mit diesen erprobten Tools wohl kaum messen.

4.1 Kommandozeilenoptionen des VISCAdaemon

Der VISCAdaemon besitzt folgende Kommandozeilenoptionen:

```
Usage: visca-daemon [-s <serial port device>] [-p <tcp/ip port>] [-d] [-v]
  default serial port device: /dev/ttyS0
  default tcp/ip port: 3376 (0xD30)
  -d sends the visca-daemon to daemon mode
  -v makes the visca-daemon tell about every command it executes
```

Es ist möglich mit -s die serielle Schnittstelle anzugeben, an der die Kamera angeschlossen ist, sowie mit -p den TCP/IP port, an dem der Daemon auf Anfragen warten soll. Der Standard TCP/IP port ist 3376, das entspricht 0xD30 im hexadezimalen und gibt somit im Namen schon Aufschluss darüber, wofür der Port genutzt wird.

Im Normalfall bleibt der VISCA-daemon im Vordergrund, mit der Option -d kann er aber auch in den Hintergrund verschoben werden. Fehlermeldungen und andere Ausgaben erfolgen dann in den Syslog anstatt auf stderr.

Mit der Angabe von -v wird der Daemon dazu gebracht, über jede einzelne Verbindung Aufschluss zu geben, was für Debugging-Zwecke nützlich sein kann.

4.2 Protokoll des VISCAdaemon

Wenn man - etwa mit Telnet - eine Verbindung zum Port des Daemons aufmacht, wird man als erstes mit "`Welcome to the VISCA camera server`" begrüßt. Man kann dann einen Befehl absetzen, der mit dem Ende der Zeile abgeschlossen wird. Als Antwort erhält man einen der folgende Fehlercodes:

- 10 OK - no return value
- 11 OK - one return value
- 12 OK - two return values
- 13 OK - three return values
- 40 ERROR - command not recognized
- 41 ERROR - argument 1 not recognized
- 42 ERROR - argument 2 not recognized
- 43 ERROR - argument 3 not recognized
- 44 ERROR - argument 4 not recognized
- 45 ERROR - argument 5 not recognized
- 46 ERROR - camera replied with an error
- 47 ERROR - camera replied with an unknown return value

Im Falle einer oder mehrerer Rückgabewerte werden diese in den auf den Fehlercode folgenden Zeilen übertragen.

Nun kann der nächste Befehl übertragen werden. Abgeschlossen wird die Verbindung, indem man eine leere Zeile übermittelt. Eine Übersicht über die verfügbaren Befehle findet sich im Anhang.

4.3 Anmerkungen zur Implementierung des VISCAdaemon

Bei der Implementierung des Daemons wurde die Unix-Socket-FAQ [5] und vor allem deren Beispielimplementierungen [6] intensiv verwendet.

Der Daemon analysiert zuerst die Kommandozeilenoptionen. Dann prüft er mit Hilfe eines Lock-Files nach, ob schon eine Instanz des Daemons läuft. Falls gewünscht begibt er sich dann in den Hintergrund und schreibt seine Ausgaben in den Syslog anstatt auf stderr. Schließlich öffnet er die Verbindung zur Kamera und wartet auf Anfragen auf dem genannten Netzwerkport. Die Anfragen führt er aus und generiert eine Antwort nach dem oben beschriebenen Protokoll.

Falls der Daemon mit einem Signal zum Beenden gezwungen wird, werden mit Hilfe eines signalhandlers zuerst noch die geöffneten sockets und Dateien geschlossen sowie das Lock-File gelöscht.

Kapitel 5

PHP-Interface

Die Aufgabenstellung lautete, die Benutzerschnittstelle auf ein Webinterface umzusetzen. Als Programmiersprache für dieses Frontend wurde PHP gewählt. Besonders hilfreich bei der Entwicklung der Webschnittstelle waren die Websites SelfHTML [7] und SelfPHP [8].

Es wurde zum einen eine kleine PHP-Bibliothek entwickelt, die den Zugriff auf den VISCAdaemon von PHP aus ermöglicht, zum andern wurden mehrere Frontends implementiert, um verschiedenen Benutzerklassen gerecht zu werden:

- Informations-Interface
- Fernbedienungs-Interface
- Listen-Interface

Die PHP-Bibliothek stellt - genauso wie die libVISCA-Highlevel-Schicht drei Funktionen zur Verfügung:

- `function openSocket(&$fp)` - Öffnet die Verbindung zum VISCAdaemon
- `sendCommand($fp, $command, &$answer, &$ret1, &$ret2, &$ret3)` - Schickt einen Befehl an den VISCAdaemon und liefert die Rückgabewerte
- `closeSocket($fp)` - Schließt die Verbindung zum VISCAdaemon wieder

5.1 Info-Interface

Das erste Interface liest einfach alle verfügbaren Informationen der Kamera aus und gibt sie in einer Liste wieder. Dabei werden lediglich die momentan relevanten Daten auch angezeigt, wenn etwa das Autotracking abgeschaltet ist, so werden auch keine Informationen über die Details des Autotracking angezeigt.

5.2 Fernbedienungs Interface

Im Fernbedienungs-Interface wird die Fernbedienung nachgebildet. Mittels einer Image-Map kann man die einzelnen Knöpfe der Fernbedienung anklicken. Die Unterschiede zur echten Fernbedienung werden unten näher beschrieben.

5.2.1 Unterschiede zur echten Fernbedienung

Folgende Unterschiede gibt es zur Handhabung der echten Fernbedienung:

- Die Auswahl der Kamera 1, 2 oder 3 funktioniert nicht. Bislang unterstützt die zugrundeliegende libVISCA nur eine einzelne Kamera.
- Die Focus-, Dreh-, Schwenk- und Zoom-Knöpfe bleiben nicht wie bei der echten Fernbedienung in Aktion, solange man sie gedrückt hält, sondern sie werden bei einmaligem Drücken aktiviert und beim zweiten Drücken wieder deaktiviert.
- Die Dreh- und Schwenk-Knöpfe beschleunigen nicht mit längerer Betätigungsdauer. Dafür gibt es eine Gruppe von radiobuttons mit der man die Dreh- und Schwenk-Geschwindigkeit einstellen kann.
- Der Menu-Knopf und der Cursor-Knopf funktionieren nicht, weil diese Funktionen über das VISCA-Interface nicht zu erreichen sind.

5.2.2 Implementierung des Fernbedienungs-Interfaces

Das Fernbedienungs-Interface besteht aus zwei Frames. Im unteren wird die Image-Map mit der Fernbedienung und der Geschwindigkeitsregelung angezeigt, im oberen Frame wird jeweils das Ergebnis des PHP-Skriptes angezeigt, das die Befehle ausführt. Diese Aufteilung hat den Vorteil, daß das Bild nur einmal geladen werden muß, was die subjektiv empfundenen Geschwindigkeit deutlich erhöht.

Jeder Knopf der Fernbedienung ruft also das PHP-Skript im oberen Frame auf. Dabei wird der Parameter ”`button`” mit dem Namen des jeweiligen Knopfes übergeben. Falls die Geschwindigkeitsregelung angeklickt wird, wird als Parameter ”`speed`” die gewünschte Geschwindigkeit übergeben.

Die aktuell eingestellte Geschwindigkeit wird auf dem Server in einer speed.inc Datei abgelegt, die nur eine Zeile enthält, etwa:

```
"<?php␣$speed=10;␣?>
```

Diese speed.inc Datei wird bei jedem Aufruf des PHP-Skriptes eingelesen, so daß gewährleistet ist, daß immer die aktuell eingestellte Geschwindigkeit verwendet wird.

Wenn einer der Focus-, Dreh-, Schwenk- und Zoom-Knöpfe gedrückt wurde, so muß diese Tatsache irgendwo festgehalten werden, damit bei erneuter Betätigung die Bewegung gestoppt werden kann. Zu diesem Zweck wird bei erstmaligem Betätigen eines solchen Knopfes eine leere Datei auf dem Server angelegt, etwa ”movingleft”. Beim nächsten Betätigen kann dann festgestellt werden, daß sich die Kamera bereits nach links bewegt, und daß diese Bewegung gestoppt werden soll. Nachdem die Bewegung gestoppt wurde, wird die Datei gelöscht und das Spiel kann von vorne losgehen.

Auch die Behandlung der preset- bzw. reset-Knöpfe für vorbelegte Positionen erfolgt nach diesem Schema.

Bei manchen Befehlen ist es auch notwendig zunächst eine Information von der Kamera auszulesen. Wenn etwa der Power-Knopf gedrückt wird, so wird zuerst der Zustand der Kamera ermittelt. Wenn sie ausgeschaltet war, wird der Befehl zum einschalten gesendet. Wenn sie eingeschaltet war, erfolgt der Befehl zum ausschalten.

Ähnlich verhält es sich auch mit der Dreh- und Schwenk-Geschwindigkeit. Abhängig vom Zoomfaktor muß diese nämlich höher oder niedriger ausfallen. Der Zoomfaktor wird deshalb ausgelesen und nach folgender Formel mit dem voreingestellten speed-Wert in eine Geschwindigkeit umgewandelt:

```
$speedcalc = $speed - floor($zoom / (1024/$speed))
```

Bei manchen Browsern (insbesondere beim Konqueror) gab es Probleme, wenn man zweimal hintereinander den gleichen Befehl ausführen wollte, etwa um eine begonnene Drehung wieder zu stoppen. Die Seite wurde aus dem Cache genommen, anstatt eine Anfrage an den Server zu schicken. Dieses Problem wurde gelöst, indem der folgende Befehl in den Header der HTML-Ausgabe des PHP-Skriptes eingebaut wurde:

```
<meta http-equiv="cache-control" content="no-cache">
<meta http-equiv="expires" content="0">
```

5.3 Listen-Interface

Das Befehlslisten-Interface ist genau wie das Fernbedienungs-Interface mit zwei Frames aufgebaut. Dadurch bleibt die Position der Liste, die im unteren Frame angezeigt wird, erhalten, während im oberen Frame die Ergebnisse der Befehlsausführung angezeigt werden.

Einige Befehle, etwa die Schwenkreichweitenbegrenzung, sind nur über dieses Interface verfügbar, da sie über die Fernbedienung nicht zu erreichen sind.

Die Befehle sind in folgende Gruppen unterteilt:

- General commands
- Zooming commands
- Focusing commands
- Exposure commands
- Pan/Tilt commands
- Auto tracking and motion detection commands

Kapitel 6

Sicherheitaspekte

Wenn die Steuerung der Kamera über das Internet erfolgen kann, so muß auf eine ausreichende Absicherung gegen Unbefugte geachtet werden.

Die Eigenentwicklung sicherheitsrelevanter Software ist mit großen Risiken verbunden, da man immer Gefahr läuft, nicht alle möglichen Angriffspunkte zu bedenken. Die Absicherung des Systems basiert daher auf Standardkomponenten, die sich seit langem bewährt haben.

Der Zugriff auf den Port des visca-daemon sollte nur vom Webserver aus zugelassen werden. Dazu kann das im Linux-Kernel integrierte iptables verwendet werden. Wenn der visca-daemon und der Webserver auf dem gleichen Rechner laufen, wie im SEP implementiert, genügt es, den Port des visca-daemons für alle externen Rechner zu sperren.

Der Zugriff auf die verschiedenen Web-Interfaces wird über den .htaccess Mechanismus des Apache-Webservers passwortgeschützt. Dadurch ist eine Abstufung der Benutzerrechte möglich, indem etwa der Zugriff auf die ausführlichen Befehlslisten einem kleineren Kreis vorbehalten wird. Wiederum hängt die Sicherheit der Zugriffskontrolle nicht von selbst entwickelter und potentiell unsicherer Software ab, sondern allein von der Sicherheit des Apache-Webservers, der bei Bekanntwerden von Sicherheitslücken gepatcht werden sollte.

Kapitel 7

Zusammenfassung und Ausblick

7.1 Zusammenfassung

Die gestellte Aufgabe, die Fernsteuerung der Videokonferenz-Kamera über ein Webinterface zu ermöglichen, wurde erfüllt. Im Rahmen der Abschlußpräsentation konnte die Funktionsfähigkeit des Systems demonstriert werden.

Im Laufe der Bearbeitung des Projekts entstand eine wesentliche Erweiterung der libVISCA, um auf die Funktionen der D30/31 Kamera zugreifen zu können.

Außerdem entstand eine Bibliothek, um die Funktionqalität der libVISCA zu abstrahieren, ein Server, der diese Funktionalität netzwerktransparent zur Verfügung stellt, eine PHP-Bibliothek, um auf diesen Server zugreifen zu können, sowie verschiedene Frontends, die wiederum auf die PHP-Bibliothek zugreifen.

7.2 Ausblick

Im Rahmen des SEP wurde nur ein Basissystem implementiert. Es wären verschieden Erweiterungen des Systems denkbar.

Man könnte etwa versuchen, ein benutzerfreundlicheres Interfaces als die Infrarot-Fernbedienung, zu entwickeln. Auf der Fernbedienung sind z.B. aus Platzmangel einige Knöpfe doppelt belegt. In einem Webinterface sind solche Beschränkungen natürlich irrelevant und hinderlich.

Die Einbindung des Live-Bildes der Kamera in das Webinterface würde es erlauben, z.b. den Zoom-Ausschnitt durch das Aufziehen eines Rahmens zu wählen. Dadurch wären sehr intuitiv zu bedienende Benutzerschnittstellen realisierbar.

Die bisher implementierten Zugangskontrollen sind relativ grob. Mit feiner abgestuften Zugriffsrechten auf die Kamerafunktionen, könnte man z.B. der entfernten Stelle die Auswahl des Bildausschnittes erlauben, aber verhindern, daß die Belichtungseinstellungen verändert werden.

Das VISCA-Protokoll unterstützt mehrerer Kameras an einer seriellen Schnittstelle. Man könnte das System so erweitern, daß mehr als eine Kamera angesteuert werden kann. Da am Lehrstuhl jedoch sowieso nur eine Kamera vorhanden ist, ist diese Funktion jedoch wohl vorerst nicht sehr relevant.

Anhang A

Befehle der D30/31

Folgende Befehle versteht die D30/31 Kamera:

```
General commands
================
set_power boolean
  Schaltet die Kamera an oder aus

set_irreceive_on
  Schaltet den Infrarot-Empfang an

set_irreceive_off
  Schaltet den Infrarot-Empfang aus

set_irreceive_onoff
  Schaltet den Infrarot-Empfang um

set_datascreen_on
  Zeige Datenbildschirm an

set_datascreen_off
  Zeige Datenbildschirm nicht an

set_datascreen_onoff
  Schalte datenbildschirmanzeige um

set_keylock boolean
  Schaltet die Tastensperre an oder aus
  => Es kann kein anderer Befehl gesendet werden
  Neu in libVISCA 0.07

memory_set channel(0..5)
  Speichere die momentane Position

memory_recall channel(0..5)
  Rufe die gespeicherte Position ab

memory_reset channel(0..5)
```

```
  Setze einen Positionsspeicher zurück

set_wide_con_lens conversion(0..7)
  Setze die Kompensationsstufe falls
  eine Weitwinkel-Linse installiert wurde.
  0: Keine Kompensation 7: X0.6 Kompensation
  Neu in libVISCA 0.07

Zooming commands
================
set_zoom_value value(0..1023)
  Setze den Zoom

set_zoom_tele
  Setze den Zoom auf Maximum

set_zoom_tele_speed speed(2..7)
  Zoome hinein mit gegebener Geschwindigkeit

set_zoom_wide
  Setze den Zoom auf Minimum

set_zoom_wide_speed speed(2..7)
  Zoome heraus mit gegebener Geschwindigkeit

set_zoom_stop
  Beende den Zoom-Vorgang

Focusing commands
=================
set_focus_auto boolean
  Schalten den Autofocus an oder ab

set_focus_value value(1000..40959)
  Setze den Focus auf einen bestimmten Wert

set_focus_far
  Setze den Focus auf Fern

set_focus_near
  Setze den Focus auf Nah

set_focus_stop
  Beende edn Fokussiervorgang

Exposure commands
=================
set_backlight_comp boolean
  Schalte die Backlight-Compensation an oder ab

set_whitebal_mode mode (0..3)
  Setze den Whitebalance-Modus auf einen bestimmten Wert
  0: Auto, 1: Indoor, 2: Outdoor, 3: OnePush
```

```
set_whitebal_one_push
  Auslöser für den Whitebalance-OnePush-Modus

set_auto_exp_mode mode(0..13)
  Setze den Auto-Exposure Modus auf einen bestimmten Wert
  0: Auto, 3: Manual, 10: Shutter priority,
  11: Iris priority, 13: Bright Mode

set_shutter_value value(0..27)
  Setze die Belichtungszeit auf einen bestimmten Wert
  0: 1/60 bis 27: 1/10000

set_shutter_up
  Erhöhe die Belichtungszeit
  Nur mit Shutter Priority oder Manuellem Exposure-Modus

set_shutter_down
  Verringere die Belichtungszeit
  Nur mit Shutter Priority oder Manuellem Exposure-Modus

set_shutter_reset
  Setze die Belichtungszeit zurück
  Nur mit Shutter Priority oder Manuellem Exposure-Modus

set_iris_value value(0..17)
  Setze  die Blendenöffnung auf einen bestimmten Wert
  0: geschlossen bis 17: F1.8

set_iris_up
  Öffne die Blende weiter
  Nur mit Iris Priority oder Manuellem Exposure-Modus

set_iris_down
  Schließe die Blende weiter
  Nur mit Iris Priority oder Manuellem Exposure-Modus

set_iris_reset
  Setze die Blendenöffnung zurück
  Nur mit Iris Priority oder Manuellem Exposure-Modus

set_gain_value value(1..7)
  Setze die Aufhellfunktion auf einen bestimmten Wert
  1: 0dB bis 7: +18dB

set_gain_up
  Setze die Aufhellfunktion auf einen höhreren Wert
  Nur mit Manuellem Exposure-Modus

set_gain_down
  Setze die Aufhellfunktion auf einen niedrigeren Wert
  Nur mit Manuellem Exposure-Modus
```

```
set_bright_up
  Helle das Bild auf
  Nur mit Bright-Modus

set_bright_down
  Dunkle das Bild ab
  Nur mit Bright-Modus

set_bright_reset
  Setze die Helligkeit zurück
  Nur mit Bright-Modus

Pan/Tilt commands
=================
set_pantilt_reset
  Initialisiere die Dreh- und Schwenk Motoren

set_pantilt_home
  Setze den Dreh/Schwenk-Kopf auf Nullstellung

set_pantilt_limit_upright pan_limit(-879..880) tilt_limit(-299..300)
  Setze die obere rechte Ecke der Dreh/Schwenk-Begrenzung

set_pantilt_limit_upright_clear
  Setze die obere rechte Ecke der Dreh/Schwenk-Begrenzung zurück

set_pantilt_limit_downleft pan_limit(-879..880) tilt_limit(-299..300)
  Setze die untere linke Ecke der Dreh/Schwenk-Begrenzung

set_pantilt_limit_downleft_clear
  Setze die untere linke Ecke der Dreh/Schwenk-Begrenzung zurück


set_pantilt_absolute_position pan_speed(1..24) tilt_speed(1..20)
                pan_position(-879..880) tilt_position(-299..300)
  Steuere eine bestimmte Position mit
  einer bestimmten Geschwindigkeit an

set_pantilt_relative_position pan_speed(1..24) tilt_speed(1..20)
                pan_position(-879..880) tilt_position(-299..300)
  Steuere eine bestimmte Relativ-Position mit
  einer bestimmten Geschwindigkeit an

set_pantilt_up pan_speed(1..24) tilt_speed(1..20)
  Drehe den Kopf mit einer bestimmten Geschwindigkeit nach oben

set_pantilt_down pan_speed(1..24) tilt_speed(1..20)
  Drehe den Kopf mit einer bestimmten Geschwindigkeit nach unten

set_pantilt_left pan_speed(1..24) tilt_speed(1..20)
  Drehe den Kopf mit einer bestimmten Geschwindigkeit nach links

set_pantilt_right pan_speed(1..24) tilt_speed(1..20)
```

```
  Drehe den Kopf mit einer bestimmten Geschwindigkeit nach rechts

set_pantilt_upleft pan_speed(1..24) tilt_speed(1..20)
  Drehe den Kopf mit einer bestimmten Geschwindigkeit nach oben links

set_pantilt_upright pan_speed(1..24) tilt_speed(1..20)
  Drehe den Kopf mit einer bestimmten Geschwindigkeit nach oben rechts

set_pantilt_downleft pan_speed(1..24) tilt_speed(1..20)
  Drehe den Kopf mit einer bestimmten Geschwindigkeit nach unten links

set_pantilt_downright pan_speed(1..24) tilt_speed(1..20)
  Drehe den Kopf mit einer bestimmten Geschwindigkeit nach unten rechts

set_pantilt_stop pan_speed(1..24) tilt_speed(1..20)
  Beende die Dreh/Schwenk Bewegung

Auto tracking and motion detection commands
===========================================
set_at_mode boolean
  Schalte den Target Tracking Modus an oder ab
  Neu in libVISCA 0.07

set_at_mode_onoff
  Schalte den Target Tracking Modus an oder ab
  Neu in libVISCA 0.07

set_md_mode boolean
  Schalte den Motion Detection Modus an oder ab
  Neu in libVISCA 0.07

set_md_mode_onoff
  Schalte den Motion Detection Modus an oder ab
  Neu in libVISCA 0.07

set_atmd_startstop
  Fange an mit oder beende das
  Autotracking oder Motion detection
  Neu in libVISCA 0.07

set_at_ae boolean
  Schalte Automatische Beleuchtung
  für Automatisches Tracking an oder ab
  Neu in libVISCA 0.07

set_at_ae_onoff
  Schalte Automatische Beleuchtung
  für Automatisches Tracking an oder ab
  Neu in libVISCA 0.07

set_at_autozoom boolean
  Schalte Automatisches Zoomen
  für Automatisches Tracking an oder ab
```

```
  Neu in libVISCA 0.07

set_at_autozoom_onoff
  Schalte Automatisches Zoomen
  für Automatisches Tracking an oder ab
  Neu in libVISCA 0.07

set_atmd_framedisplay boolean
  Schalte Rahmenanzeige für Automatisches Tracking
  und Motion Detection an oder ab
  Neu in libVISCA 0.07

set_atmd_framedisplay_onoff
  Schalte Rahmenanzeige für Automatisches Tracking
  und Motion Detection an oder ab
  Neu in libVISCA 0.07

set_at_frameoffset_onoff
  Schalte zur Rahmenpositionskontrolle
  f. Auto. Tracking und Motion Detection um
  Neu in libVISCA 0.07

set_md_frame
  Setze die Position oder die Größe
  für die Motion Detection
  Neu in libVISCA 0.07

set_md_detect
  Wechsle zwischen den beiden Rahmen
  für Motion Detection: 1, 2 oder 1&2
  Neu in libVISCA 0.07

set_md_measure_mode1 boolean
  Schalte Motion Detection
  Messverfahren 1 an oder ab
  Neu in libVISCA 0.07

set_md_measure_mode1_onoff
  Schalte Motion Detection
  Messverfahren 1 an oder ab
  Neu in libVISCA 0.07

set_md_measure_mode2 boolean
  Schalte Motion Detection
  Messverfahren 2 an oder ab
  Neu in libVISCA 0.07

set_md_measure_mode2_onoff
  Schalte Motion Detection
  Messverfahren 2 an oder ab
  Neu in libVISCA 0.07

set_at_chase_next
```

```
  Wechsle die Verfolgunsmethode für AutoTracking
  Neu in libVISCA 0.07

set_at_chase (mode 0..2)
Wähle die Verfolgunsmethode für AutoTracking
  Neu in libVISCA 0.07

set_at_entry entry (0..3)
Wähle die Zielmethode für AutoTracking
  Neu in libVISCA 0.07

set_md_adjust_ylevel (level 0..15)
Stelle Y-Level für Motion Detection ein
  Neu in libVISCA 0.07

set_md_adjust_huelevel (level 0..15)
Stelle Hue-Level für Motion Detection ein
  Neu in libVISCA 0.07

set_md_adjust_size (level 0..15)
Stelle Größen-Level für Motion Detection ein
  Neu in libVISCA 0.07

set_md_adjust_disptime (level 0..15)
Stelle Bewegungs-Level für Motion Detection ein
  Neu in libVISCA 0.07

set_md_adjust_refmode (mode 0..2)
Stelle Auffrisch-Rate für Motion Detection ein
  Neu in libVISCA 0.07

set_md_adjust_reftime (time 0..15)
Stelle Referenz-Zeit-Level für Motion Detection ein
  Neu in libVISCA 0.07

Befehle, die einen oder mehrere Werte zurückliefern
===================================================
get_power
  (boolean)

get_focus_auto
  (boolean)

get_backlight_comp
  (boolean)

get_datascreen
  (boolean)

get_keylock
  (boolean)
  Neu in libVISCA 0.07
```

```
get_zoom_value
  (0..1023)

get_focus_value
  (1000..40959)

get_whitebal_mode
  (0: Auto, 1: Indoor, 2: Outdoor, 3: OnePush)

get_auto_exp_mode
  (0: Auto, 3: Manual, 10: Shutter priority,
  11: Iris priority, 13: Bright Mode)

get_shutter_value
  (0: 1/60 .. 27: 1/10000)

get_iris_value
  (0: closed .. 17: F1.8

get_gain_value
  (1: 0dB .. 7: +18dB)

get_memory
  (Momentane Speicherposition 0 .. 5)

get_id
  (camera id)

get_videosystem
  (0 for NTSC, 1 for PAL)

get_pantilt_mode
  (returns the pantilt status, what is this?)

get_pantilt_maxspeed
  (max_pan_speed) (max_tilt_speed)

get_pantilt_position
  (pan_position: -860..862)
  (tilt_position: -281..283)

get_wide_con_lens
  (Weitwinkel-Linsenkompensation:
   0: Keine ..7: X0.6 conversion)
   Neu in libVISCA 0.07

get_atmd_mode
  (0: Normal mode, 1: AT mode, 2: MD mode)
   Neu in libVISCA 0.07

get_at_mode
  (AT status, see D30/31 Command List for details)
  Neu in libVISCA 0.07
```

```
get_at_entry
  (AT Wintrag: 0 .. 3)
  Neu in libVISCA 0.07

get_md_mode
  (MD status, see D30/31 Command List for details)
  Neu in libVISCA 0.07

get_md_ylevel
  (level of detection from 0 to 15)
  Neu in libVISCA 0.07

get_md_huelevel
  (level of detection from 0 to 15)
  Neu in libVISCA 0.07

get_md_size
  (level of detection from 0 to 15)
  Neu in libVISCA 0.07

get_md_disptime
  (level of detection from 0 to 15)
  Neu in libVISCA 0.07

get_md_refmode
  (refreshmode from 0 to 2)
  Neu in libVISCA 0.07

get_md_reftime
  (refreshtime from 0 to 15)
  Neu in libVISCA 0.07

get_at_obj_pos
  (x-Position des detection frame geteilt durch 48)
  (y-Position des detection frame geteilt durch 30)
  (status: 0=Setting, 1=Tracking, 2=Lost)
  Neu in libVISCA 0.07

get_md_obj_pos
  (x-Position des detection frame geteilt durch 48)
  (y-Position des detection frame geteilt durch 30)
  (status: 1=UnDetect, 2=Detected)
  Neu in libVISCA 0.07

Befehle die in der libVISCA implementiert sind, aber nicht in der D30/31
=========================================================================
set_focus_one_push
set_focus_infinity
set_focus_autosense_high
set_focus_autosense_low
set_rgain_up
set_rgain_down
```

```
set_rgain_reset
set_bgain_up
set_bgain_down
set_bgain_reset
set_aperture_up
set_aperture_down
set_aperture_reset
set_exp_comp_up
set_exp_comp_down
set_exp_comp_reset
set_title_clear
set_dzoom boolean
set_exp_comp_power boolean
set_slow_shutter_auto boolean
set_zero_lux_shot boolean
set_ir_led boolean
set_mirror boolean
set_freeze boolean
set_display boolean
set_date_display boolean
set_time_display boolean
set_title_display boolean
set_focus_far_speed speed
set_focus_near_speed speed
set_focus_near_limit limit
set_rgain_value value
set_bgain_value value
set_bright_value value
set_aperture_value value
set_exp_comp_value value
set_wide_mode mode
set_picture_effect mode
set_digital_effect mode
set_digital_effect_level level
set_zoom_and_focus_value zoom(0..1023) focus(1000..40959)
set_title_params vposition hposition color blink
set_date_time year month day hour minute
set_title vposition hposition color blink text
get_dzoom
get_exp_comp_power
get_zero_lux_shot
get_ir_led
get_mirror
get_freeze
get_display
get_focus_auto_sense
get_focus_near_limit
get_rgain_value
get_bgain_value
get_slow_shutter_auto
get_bright_value
get_exp_comp_value
get_aperture_value
```

```
get_wide_mode
get_picture_effect
get_digital_effect
get_digital_effect_level
```

Anhang B

Installation des Systems

Um das vorgestellte System auf einem Linux-PC zum laufen zu bringen sind folgende Schritte notwendig:

- libVISCA installieren
- Apache Webserver mit PHP-Unterstützung installieren
- visca-daemon installieren
- PHP-Frontends installieren

B.1 libVISCA installieren

Die aktuelle Version der libVISCA kann von sourceforge.net heruntergeladen werden. Nach dem Entpacken des tar-Archives kann mittels `./configure && make && make install` die libvisca installiert werden.

B.2 Apache Webserver mit PHP-Unterstützung installieren

Der Apache-Webserver wird am zweckmäßigsten über das jeweilige Paket-Management der verwendeten Distribution installiert. Dabei ist darauf zu achten, daß die entsprechenden Pakete für die PHP-Unterstützung mit installiert werden. Informationen über diesen Vorgang findet man im Handbuch oder auf der Website der jeweiligen Linux-Distribution.

B.3 visca-daemon installieren

Im Verzeichnis Sourcen/src des SEP-Paketes befindet sich der Sourcecode des visca-daemons. Wenn die libvisca richtig installiert wurde genügt der Befehl `make` um die ausführbare Datei visca-daemon zu erzeugen.

Der visca-daemon kann jetzt mit `./visca-daemon` von Hand gestartet werden. Falls der visca-daemon bei jedem Starten des Systems automatisch geladen werden soll, sind entsprechende Eintragungen im Verzeichnis `/etc/init.d` vorzunehmen. Auskunft über den Bootmechanismus der verwendeten Distribution gibt wiederum das Handbuch oder die Website des Herstellers.

Kommandozeilenparameter für den visca-daemon sind nur notwendig, wenn von den Standardwerten abgewichen werden soll.

B.4 PHP-Frontends installieren

Um die PHP-Frontends zu installieren ist der Inhalt des Verzeichnises Sourcen/web des SEP-Paketes in den Pfad des Webservers kopieren. Aufschluß darüber, wo der Webserver die auszuliefernden Dateien ablegt, gibt der Eintrag `DocumentRoot` in der Konfigurationsdatei des Webservers (etwa `/etc/apache/httpd.conf`). Je nach verwendeter Distribution kommen verschiedene Pfade in Frage, so daß eine absolut gültige Angabe an dieser Stelle nicht gemacht werden kann.

Literaturverzeichnis

[1] Joseph J. Janus: *Sony EVI-D30/D31 Information Website*, http://www.j3soft.com/webcam/evi-d31.htm .

[2] Sony-Corporation: *EVI-D30/D31 Operating Instructions*, http://www.j3soft.com/webcam/evi-d30_operations.zip .

[3] Sony-Corporation: *EVI-D30/D31 VISCA Command List*, http://www.j3soft.com/webcam/evi-d30_commands.zip .

[4] Damien Douxchamps: *libVISCA Home Page*, http://www.tele.ucl.ac.be/PEOPLE/DOUXCHAMPS/libvisca/ .

[5] Vic Metcalfe: *UNIX Socket FAQ*, http://www.developerweb.net/sock-faq/flatfaq.php .

[6] Vic Metcalfe: *UNIX Socket FAQ - Examples*, http://www.developerweb.net/sock-faq/examples.tar.gz .

[7] Stefan Münz: *SELFHTML Homepage*, http://selfhtml.teamone.de/ .

[8] Damir Enseleit: *SELFPHP Homepage*, http://www.selfphp.info/ .