



Fachhochschule Stralsund
Fachbereich ETI
Zur Schwedenschanze 15
18435 Stralsund

Sommersemester 2009

Dozentin: Prof. Dr. rer. nat. Gudrun B. Falkner

Projektarbeit

Erstellung einer Webseite zur Kameraüberwachung eines Kleintierkäfigs

Lars Formella
Matrikelnummer: xxxx
Bachelor of Science (Informatik)
6. Semester

Inhaltsverzeichnis

1 Einleitung	4
1.1 Motivation	4
1.2 Der Kleintierkäfig	4
2 Hardware Komponenten	5
2.1 Bestehendes System	5
2.1.1 Linux-Router	5
2.1.2 Kleiner 7 Zoll Touchscreen	6
2.2 Auswahl der Lichtsteuerungsmethode	7
2.2.1 LED-Spots als Beleuchtung	7
2.2.2 Eine Steckdosenleiste zur Kontrolle	8
2.3 Auswahl der Kamera	8
2.3.1 Auswahl der Logitech „Sphere AF“	8
2.3.2 Erweiterung der USB-Kabellänge	9
3 Software Komponenten	10
3.1 Kommunikationsweg	10
3.1.1 Zentralisierte Variante	10
3.1.2 Verteilte Variante	11
3.2 Bestehendes System	12
3.2.1 Lokaler Router	12
3.2.2 Software zur Kamerasteuerung	12
3.2.3 Webserver im Internet	12
3.3 Auswahl der Programmiersprache	13
3.3.1 Native Lösung mit C / C++	13
3.3.2 Skript Lösung mit PHP	13
4 Erstellung der Software	14
4.1 Kommunikationsabfolge	14
4.2 Lokale Skripten auf dem Router	16
4.2.1 webcam_config.php	16
4.2.2 loader.php	16

4.2.3 webcam_daemon.php	17
4.2.4 webcam_ptz.php	17
4.3 Webseite auf dem Webserver	18
4.3.1 Alles funktioniert mit Ajax	18
4.3.2 Die Ajax-Bibliothek MooTools	19
4.3.3 Administrative Funktionen	19
4.3.4 Bilderarchiv Funktion	20
5 Test	21
5.1 Auftretende Probleme	21
5.1.1 Ajax / Javascript blockiert manchmal	21
5.1.2 Das Ladeskript beendet sich sofort	21
5.1.3 Webcam bewegt sich fehlerhaft	22
5.1.3.1 Neustart nach einer Kernel Panic	22
5.1.3.2 Die Webcam dreht durch	22
5.1.3.3 Ein neuer Kernel als Verursacher	22
5.1.3.4 Wie lange kann diese Lösung halten	23
5.1.4 Probleme bei USB Ports	23
5.1.5 Der Rest funktioniert ohne Probleme	24
6 Zusammenfassung	25
6.1 Fazit	25
6.2 Aussicht	25
7 Softwarelistings	26
7.1 webcam_config.php	26
7.2 loader.php	26
7.3 webcam_daemon.php	27
7.4 webcam_ptz.php	29
8 Quellen	31
9 Abbildungsverzeichnis	32

1 Einleitung

1.1 Motivation

Meine Haustiere sind zwei Meerschweinchen und zwei Hasen. Da einer der beiden Hasen in der Vergangenheit einmal aus seinem Käfig ausgebrochen ist, habe ich vor etwa zwei Jahren eine Netzwerkkamera installiert. Diese war über einen eigenen Port aus dem Internet erreichbar und nicht für die Öffentlichkeit bestimmt. Aufgrund eines Defektes ist Sie jedoch vor einem Jahr ausgefallen und seitdem sind die Haustiere unbeobachtet.

Aus persönlichem Interesse und weil ich bereits verschiedene Webseiten mit Haustierkameras gesehen habe, möchte ich nun selbst eine erstellen. Dabei sollen möglichst schon vorhandene Komponenten sinnvoll genutzt und benötigte Komponenten günstig erworben werden. Außerdem soll das Ganze automatisiert und ohne manuelles Eingreifen auf Dauer funktionieren.

1.2 Der Kleintierkäfig

Die Behausung für meine vier Haustiere ist von mir selbst aus Holz gebaut und bereits beim Bauen so gestaltet worden, dass die Kameras ohne weiteres angebracht werden können. Darüber hinaus sind Einsparungen für kleine Lampen vorhanden um eventuell auch bei Dunkelheit etwas sehen zu können.

Der Käfig selbst steht im Wohnzimmer und besteht aus drei gleich großen Holzquadern, die jeweils 60 cm tief, 100 cm lang und 60 cm hoch sind. Sie haben Plexiglasscheiben, sind untereinander verbunden und haben selbstverständlich auch einen Ausgang, durch den ein Auslauf möglich ist.

Ungefähr fünf Meter vom Käfig entfernt steht ein ständig laufender Linux-Router, der den Zugriff auf das Internet verwaltet. Dieser ist an ein Kabelmodem angeschlossen.



Abbildung 1: Holzkäfig mit den beiden Kameras und vier Bewohnern

2 Hardware Komponenten

In diesem Punkt geht es um die Wahl der entsprechenden Hardware. Dazu gehört zum einen die entsprechende Kamera, die die Bilder bereit stellen soll und eine oder mehrere Lampen, um den Käfig auch bei Dunkelheit zu beobachten.

2.1 Bestehendes System

2.1.1 Linux-Router

Da wie schon angemerkt ein Linux-Router vorhanden ist, der 24 Stunden am Tag läuft, wird dieser als Anschlussstelle für die Kameras gewählt.

Der Linux-Router besteht aus einem „Jetway J7F4K1GE“ Mini-ITX Mainboard in einem Aluminiumgehäuse. Der Mini-ITX Formfaktor gibt das Größenformat 17 cm x 17 cm vor, so dass sich damit ziemlich kleine Computer bauen lassen.

Das Mainboard selbst besitzt einen „VIA Eden 1Ghz Prozessor“, der über einen passiven Kühlkörper verfügt und maximal sechs Watt verbraucht. Er ist vollkommen ausreichend, da ein ressourcenschonendes „Ubuntu¹ Server 8.04“ Linux als Betriebssystem verwendet wird.

Es ist allerdings ein extra 12 cm Lüfter angeschlossen, da besonders im Sommer und bei hoher Systemlast die Temperatur der CPU stark ansteigen kann. Somit lässt sich einfach und ohne großen Aufwand das gesamte System auf einer niedrigen Temperatur halten.

Am Mainboard selbst sind direkt vier externe und zwei interne USB 2.0 Anschlüsse vorhanden, an die die Kameras angeschlossen werden können.



Abbildung 2: Mini ITX Gehäuse mit 12cm Lüfter

1 Ubuntu ist eine auf Debian basierende Linux-Distribution und gehört zu den meist-benutzten Linux-Distributionen.

2.1.2 Kleiner 7 Zoll Touchscreen

Direkt über dem Linux-Router befindet sich ein sieben Zoll Touchscreen, der an den Rechner angeschlossen ist. Auf dem Linux-Router wurde ein einfacher Fenstermanager und ein simpler Browser installiert und somit die Grundlage für eine Webanwendung geschaffen.

Der 1GHz Prozessor schafft selbst diese zusätzlichen Prozesse und den Speicherbedarf mühelos, so dass es zu keiner Beeinträchtigung der eigentlichen Funktionalität des Rechners kommt.

In dem Browser läuft eine lokale Webseite im Vollbildmodus und bietet unter anderem Zugriff auf verschiedene Einstellungen des Linux-Systems. Diese Webseite liegt auf dem Server selbst und ist nicht über das Internet erreichbar.

Es lassen sich graphische Statistiken über die Festplattenbelegung ebenso anzeigen wie z.B. der aktuelle Speiseplan der Mensa in Stralsund. Somit kann man ganz einfach Informationen abfragen, ohne seine Laptop anmachen zu müssen.

Außerdem bietet sich somit die Möglichkeit, damit auch die beiden Kameras zu überwachen und zu steuern. Man kann somit bequem das Licht kontrollieren oder auch die Kameras zu verschiedenen Positionen bewegen. Das Ganze ist intuitiv mit einem Finger bedienbar und ohne eine extra Tastatur und Maus steuerbar.



Abbildung 3: Sieben Zoll Touchscreen mit Bildern der beiden Kameras

2.2 Auswahl der Lichtsteuerungsmethode

2.2.1 LED-Spots als Beleuchtung

Die entsprechende Lichtquelle soll direkt in den Käfig integriert werden, aber für die Bewohner natürlich weder durch Wärmeentwicklung, noch frei liegenden Kabeln / elektrischen Strom gefährlich werden können.

Dabei gibt es verschiedene Möglichkeiten. Zum einen gibt es Leuchtfolie, die man einfach hinter eine Plexiglasscheibe kleben kann. Die Leuchtstärke, dieser eigentlich für Dekorationen gedachten Folien, ist zwar ausreichend, allerdings liegen die Preise in einem unerschwinglichen Bereich. Alleine eine A4 Folie kostet schon um die 80 Euro - für eine ausreichende Beleuchtung eines so großen Käfigs benötigt man aber mindestens drei Stück. Dieser Preis würde deutlich den gesetzten Rahmen sprengen.

Die einzige günstige Möglichkeit besteht daher in der Verwendung von LED-Spots. Diese weisen nur eine geringe Wärmeentwicklung auf und können über eine äußere Kabelführung auch nicht zu Gefahrenquellen für die Nagetiere werden.

Sie sind leider durch ihre Bauweise bedingt fünf bis sechs Zentimeter tief, allerdings ließ sich das Problem durch einen vorgesetzten „Holzkreis“ elegant kaschieren. Dadurch wirkt das Ganze wie aus einem Guss und durch die zusätzlich gewonnene Tiefe verschwindet der Spot komplett im Holz.

Die LED-Spots gibt es nur im 5er-Pack mit einem Netzteil, welches alle fünf mit Strom versorgt. Sollte man jedoch nur vier oder weniger brauchen, ist es anzuraten die anderen dennoch angeschlossen zu lassen, da sonst das Netzteil zu viel Strom an die übrigen verteilt und sie somit irreparabel beschädigen kann.

Da für den Käfig nur neun gebraucht werden, liegt ein LED-Spot neben dem Käfig. Auch er ist angeschlossen um die anderen vier an dem Netzteil nicht zu belasten.



Abbildung 4: Einer von neun LED-Spots in einem Holzkreis

2.2.2 Eine Steckdosenleiste zur Kontrolle

Es gibt nicht viele Möglichkeiten um einen normalen 230V Stecker mit einem PC zu kontrollieren. Insbesondere die Suche nach eine preiswerten Lösung hat etwas Zeit in Anspruch genommen.

Nach einiger Suche hat sich für die Kontrolle der LED-Spots eine steuerbare Steckdosenleiste angeboten. Diese gibt es in preiswerten Varianten, die sich mit einem Kommandozeilen Programm auch unter Linux ansteuern lassen. Sie wird ganz einfach per USB-Kabel mit dem Rechner verbunden und bietet die Möglichkeit vier von sechs Anschlüssen separat zu Beschalten.



Abbildung 5: USB-Steckdosenleiste - der braune Stecker ist für die LEDs

Da für die Lampen nur ein Steckplatz gebraucht wird, bietet sich so die Möglichkeit, dass später andere elektrische Geräte hinzugefügt werden können

2.3 Auswahl der Kamera

Vor der Auswahl wurden mehrere Kameras in unterschiedlichen Preissegmenten von verschiedenen Herstellern getestet. Sie unterschieden sich teilweise sehr stark in der Bildqualität, die mit dem entsprechenden Linux-Treiber dafür erreicht wurde. Leider ist es nach wie vor so, dass auf die Entwicklung der Windows-Treiber mehr Wert gelegt wird und die Linux-Versionen hinter her hinken, oder sogar überhaupt nicht vorhanden sind.

2.3.1 Auswahl der Logitech „Sphere AF“

Außerdem ist die Auswahl an beweglichen USB-Kameras auf dem Markt recht begrenzt, so dass die Wahl schnell auf die Kamera „Sphere AF“ von Logitech fiel. Diese Kamera besitzt einen zwei Megapixel Sensor, sowie verschiedene interne Technologien, um auch bei widrigen Aufnahmeverhältnissen dennoch gute Bilder zu machen. Sie bietet dazu die Möglichkeit mit zwei kleinen eingebauten Servomotoren den Kamerakopf um bis zu 180° horizontal zu schwenken und 60° vertikal zu neigen.

Der Hersteller Logitech zeichnet sich zudem durch eine gute Linux-Unterstützung seiner Produkte aus. Somit ist auch eine problemlose Ansteuerung der SNK-Fähigkeit² der Kamera unter Linux möglich. Die Bildqualität, die mit der „Sphere AF“ und dem Linux-Treiber dafür erreicht wurde, hat alle anderen Kameras überzeugend übertroffen.

² SNK bedeutet Schwenk-Neige-Kopf und bezeichnet die Funktion den Kamerakopf waagrecht zu schwenken und senkrecht zu drehen. Ins englische wird SNK als PTZ für „Pan Tilt Zoom“ übersetzt.



Abbildung 6: Eine von zwei Logitech „Sphere AF“ Kameras überkopf hängend

Die Kameras sitzen auf einem Plastiksockel in einer Vertiefung, um sich drehen zu können. Damit Sie nicht beschädigt oder verschmutzt werden, wurden die Kameras in 60 cm Höhe am oberen Ende des Käfigs befestigt.

Dadurch werden die Bilder zwar verkehrt herum aufgenommen, allerdings lässt sich dieses Manko durch die verwendete Software wieder rückgängig machen. Mit dieser lassen sich ganz einfach die Bilder drehen oder auch spiegeln.

2.3.2 Erweiterung der USB-Kabellänge

Die maximale Kabellänge bei USB beträgt fünf Meter, kann jedoch durch Repeater³ auf bis zu 25 Meter ausgedehnt werden. Da der Käfig mehr als fünf Meter weit vom Rechner entfernt steht, werden die beiden Kameras jeweils mit einer aktiven USB-Verlängerung an einen freien USB-Port angeschlossen.

Diese ist für ungefähr zehn Euro zu bekommen und hat eine Länge von fünf Metern. Die Kamera selbst hat eine Kabellänge von zwei Metern, so dass insgesamt sieben Meter zu Verfügung stehen. Damit bleibt auch in der Zukunft genügend Spielraum für verschiedene Kamerapositionen.

Außerdem ist so ein zusätzlicher Puffer für eine Umstellung des Käfigs vorhanden. Dadurch kann beispielsweise die ganze Konstruktion auch nach einem Umzug ohne große Änderungen an der Verkabelung aufgebaut werden.

³ Repeater wiederholen Signale die über Leitungen gesendet werden und gleichen Leitungsverluste aus.

3 Software Komponenten

Da die Auswahl der Hardware nun komplett ist, erfolgt jetzt die Auswahl der Software.

3.1 Kommunikationsweg

Die meisten der erhältlichen Netzwerkkameras können direkt mit dem Webbrowser über einen eigenen Port angesprochen werden. Somit muss man einfach nur einen Port in seiner Firewall, bzw. auf dem Router freigeben und hat eine funktionierende, über das Internet erreichbare Kamera.

3.1.1 Zentralisierte Variante

Die Linien auf dem Bild geben die notwendige Bandbreite an. Sie erhöht sich jeweils um eins, wenn ein neuer Client dazu kommt. Wie zu sehen ist, muss der Router alle Clients mit den Daten, bzw. dem Videostream versorgen.

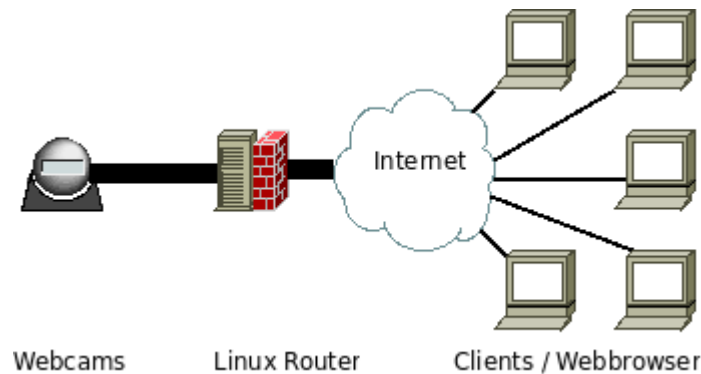


Abbildung 7: Kommunikationsweg 1

Die Probleme dabei sind folgende:

1. Die Clients verbinden sich direkt auf die Kamera und müssen somit die IP-Adresse des Linux-Routers kennen. Dadurch entstehen eventuell Sicherheitsrisiken, selbst wenn der Router über eine Firewall verfügt.
2. Da sich jeder Client direkt auf die Kamera verbindet, erhöht sich damit auch die benötigte Bandbreite für die Übertragung. Die Kamera baut für jeden Client eine eigene Verbindung auf, die nicht von anderen benutzt werden kann.
Bei mehreren Clients kann somit die Verbindung zusammenbrechen, oder auch den normalen Netzwerkbetrieb des Routers stören.
3. Die Manipulation der Kamera Bilder ist nur schwer möglich. Somit kann man zum Beispiel keine Bilder in Galerien speichern, oder Bildbereiche ausblenden, die nicht für die Öffentlichkeit bestimmt sind.
4. Die Kameras können nicht ohne weiteres ausgeschaltet werden, da meist keine Informationen darüber vorliegen, ob gerade jemand verbunden ist.

Aufgrund dieser Probleme wird eine anderes Szenario gewählt.

3.1.2 Verteilte Variante

Hierbei ist ein zusätzlicher Webserver im Internet vorhanden, über den die gesamte Kommunikation abgewickelt wird. Auch hier gibt die Dicke der Linien auf dem Bild die notwendige Bandbreite an. Sie erhöht sich beim Webserver auf der rechten Seite jeweils um eins wenn ein neuer Client dazu kommt. Allerdings bleibt die Bandbreite auf der linken Seite konstant bei zwei - egal wie viele Clients dazu kommen.

Somit können auch mehrere Clients die Bandbreite des Linux-Routers nicht negativ beeinträchtigen.

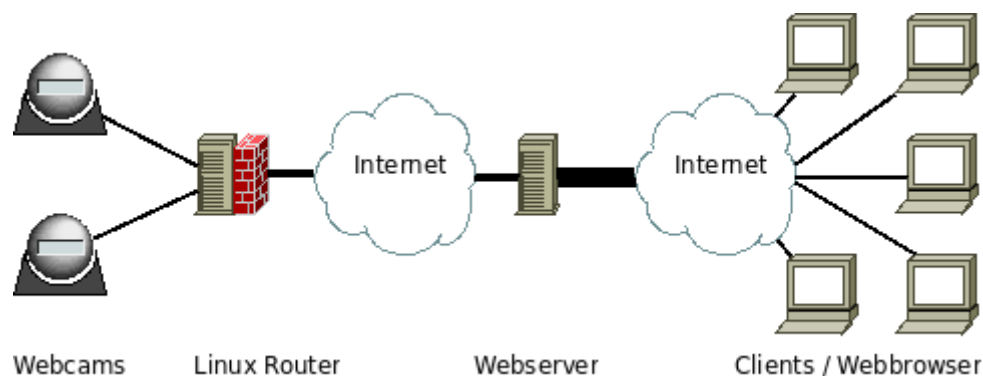


Abbildung 8: Kommunikationsweg 2

Das hat zwar den Nachteil, dass nur einzelne Standbilder und kein kompletter Videostream übertragen werden kann, allerdings entstehen dadurch die folgenden überwiegenden Vorteile:

1. Die Bilder müssen nur einmal ins Internet, bzw. auf den Webserver übertragen werden und können dann an eine beliebige Anzahl Clients weiter verteilt werden. Dadurch wird die Netzwerklast auf den besser ans Internet angebundenen Webserver verschoben.
2. Die IP und das Kommunikationsprotokoll des Routers muss nur dem Webserver bekannt sein, der gegen Angriffe besser gesichert ist. Dass es einen Router gibt an dem die Kameras angeschlossen sind, wissen die Clients gar nicht.
3. Auf dem Webserver selbst können die Bilder manipuliert werden, oder auch in verschiedenen Formen abgespeichert werden. Eine Galerie lässt sich ebenso einfach erstellen, wie bestimmte Bildbereiche auszublenden.
4. Wenn zeitweise keine Clients zum Webserver verbunden sind, kann dieser den Router, bzw. die Kameras in eine Art Ruhezustand versetzen und erst bei einem erneuten Besuch durch einen Client wieder aktivieren.

3.2 Bestehendes System

3.2.1 Lokaler Router

Auf dem Linux-Router ist „Ubuntu Server 8.04“ installiert, welches eine gute Grundlage für die folgenden Software bietet. Ubuntu ist eine auf Debian basierende Linux-Distribution, die es seit 2004 gibt. Es gibt verschiedene Versionen, wobei Ubuntu Server ohne eine graphische Benutzeroberfläche installiert wird. Dadurch verbraucht er weniger Ressourcen und ist somit auch auf weniger performanten Systemen einsetzbar.

Auf dem Router ist neben einem Apache-Webserver⁴ mit den notwendigen PHP-Erweiterungen auch eine dynamischer DNS-Client installiert, so dass der Rechner problemlos im Internet erreichbar ist.

Da vom Internetzugangs Anbieter eine Zwangstrennung der Verbindung erfolgt, muss die ständig wechselnde IP dem Webserver bekannt sein - andernfalls könnte er sich nicht zum Linux-Router verbinden und die Kamerabefehle absetzen.

Mit Hilfe einer DynDNS-Software lassen sich solche dynamischen IP-Adressen eines Rechners einem immer gültigem Hostnamen wie zum Beispiel „webcam.dyndns.org“ zuweisen. Dabei wird einfach bei jeder neuen zugewiesenen IP ein Programm aufgerufen, welches die IP dem DynDNS-Server bekannt macht.

Der Webserver verbindet sich damit immer nur zu der Adresse „webcam.dyndns.org“, welche auf die jeweilige aktuelle IP des Linux-Routers aufgelöst wird.

3.2.2 Software zur Kamerasteuerung

Für die Ansteuerung der Kameras wird „motion“ verwendet. Motion ist eine Software, die im Hintergrund läuft und die Bilder der angeschlossenen Kameras zur Verfügung stellt. Sie kann außerdem Bewegungen erkennen und diese auch farbig auf den Bildern hervor heben.

Leider funktioniert bei der Version aus den Ubuntu Softwarequellen die Steuerung der SNK-Fähigkeit der Kamera nicht richtig, so dass ein zusätzliches Programm namens „uvcdynctrl“ zum Einsatz kommt.

Dieses Kommandozeilen Programm dient lediglich dazu verschiedene Optionen, die die Kamera bietet, zu steuern. Dazu gehört neben der SNK-Fähigkeit beispielsweise auch der Weißabgleich oder Autofokus.

3.2.3 Webserver im Internet

Der Webserver im Internet ist ein angemietetes Paket „Virtual Server Standard“ der Firma Domainfactory und über die Adresse <http://www.bimmel-bommel.de> erreichbar.

Er bietet zwei Gigabyte Speicherplatz und neben mehreren MySQL-Datenbanken⁵ zum Beispiel auch PHP-Unterstützung.

Das Webspace Paket beherbergt dazu noch mehr als 10 andere private Seiten, so dass sich die Performance des Webserver schon mehrfach positiv bestätigt hat.

⁴ Apache gehört zu den bekanntesten Webservern im Linux-Bereich

⁵ MySQL ist ein relationales Datenbankverwaltungssystem

3.3 Auswahl der Programmiersprache

Die Software kann auf dem Linux-Router mit zwei verschiedenen Technologien realisiert werden. Exemplarisch ist für die erste die Programmiersprache C / C++ gewählt worden.

Auf dem Webserver muss die zweite Technologie genommen werden, da es dort keine Möglichkeit gibt ein natives Programm auszuführen.

3.3.1 Native Lösung mit C / C++

Eine native Lösung bietet sich unter anderem immer dann an, wenn es sich um zeitkritische Programme handelt. Muss etwas sehr schnell gehen, oder werden ressourcenschonende Algorithmen benötigt, dann ist C++ mit eine der geeignetsten Sprachen.

Bei einer nativen Software, wie beispielsweise mit C++, wird der Quellcode in ein effizientes, maschinennahes Programm kompiliert. Der Vorteil liegt dabei in der Geschwindigkeit der Software.

Allerdings gestaltet sich die Entwicklung in diesem Fall als problematisch, da die Software auf dem Linux-Router laufen muss, aber nicht auf ihm entwickelt werden kann. Das macht das Beseitigen der Fehler schwieriger.

Da es bei der Kameralösung nicht auf Geschwindigkeit ankommt, wird die Entwicklung in einer Skriptsprache vorgezogen.

3.3.2 Skript Lösung mit PHP

Als Programmiersprache wird daher PHP gewählt, da diese sowohl auf dem Router als auch auf dem Webserver vorhanden ist. Somit können die Programme schneller und einfacher geschrieben werden, da Code-Fragmente auf beiden System genutzt werden können.

PHP ist eine Skript-Sprache mit einer an C angelehnten Syntax. Sie wird hauptsächlich zur Erstellung von dynamischen Webseiten oder Webanwendungen verwendet. Neben der Integration in den Apache-Webserver, besteht auch die Möglichkeit PHP-Skripten direkt auf der Kommandozeile zu starten.

Wie bei Skript-Sprachen üblich, muss das Programm nicht kompiliert werden, sondern kann direkt vom Interpreter ausgeführt werden. Das vereinfacht die Fehlerbehebung und Veränderungen an der Software um ein Vielfaches, da Änderungen sofort ausgeführt und getestet werden können.

In den Ubuntu-Quellen ist PHP in der Version 5 vorhanden. Ebenso sind alle eventuell auch später notwendigen Erweiterungen, wie zum Beispiel die Grafikbibliothek GD⁶ problemlos über die Ubuntu-Quellen verfügbar.

6 GD ist eine Opensource Bibliothek um dynamische Bilder zu erzeugen

4 Erstellung der Software

Da nun die Auswahl der Hard- und Software abgeschlossen ist, kann jetzt die Umsetzung in Software erfolgen.

4.1 Kommunikationsabfolge

Die Auswahl der Kommunikation wurde schon im vorherigen Kapitel erörtert, so dass nun die Abfolge der Kommunikation beschrieben wird.

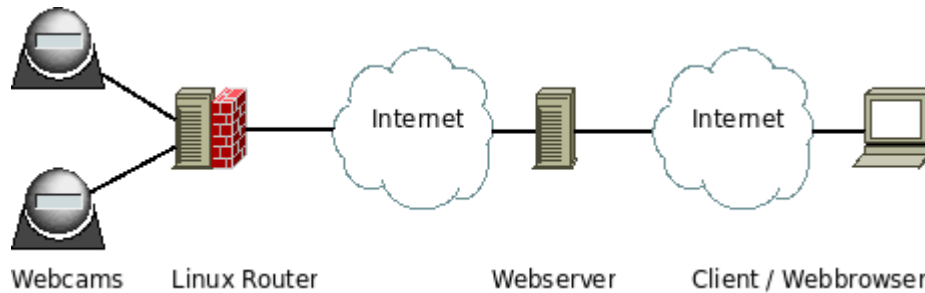


Abbildung 9: Überblick über Kommunikation

Zu aller erst ruft der Client die Webseite auf dem Webserver auf.

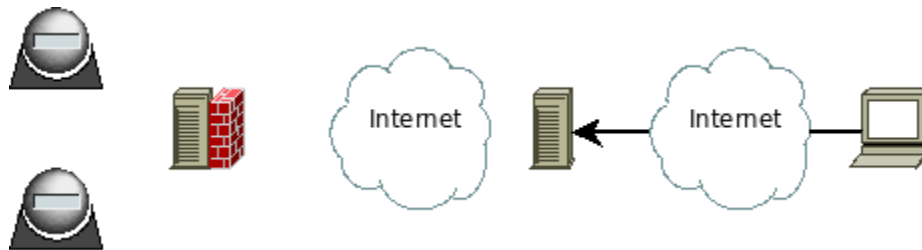


Abbildung 10: Schritt 1

Dieser prüft nun ob bereits aktuelle Bilder der Kameras auf dem Webserver vorhanden sind. In diesem Fall schickt er Sie sofort an den Client (siehe Schritt 6). Sollten jedoch keine vorhanden sein, ruft der Server eine bestimmte Adresse auf dem Router auf.



Abbildung 11: Schritt 2

Kapitel 4 - Erstellung der Software

Dadurch startet der Router nun entweder das Programm um die Bilder auf den Webserver zu laden oder er führt den entsprechenden Befehl aus. Sollte der Webserver einen SNK-Befehl gesendet haben, wird die Kamera bewegt, oder zum Beispiel durch einen anderen Befehl das Licht angeschaltet.



Abbildung 12: Schritt 4

Das gestartete Programm lädt nun jede Sekunde die Bilder der beiden Kameras auf den Webserver. Dabei prüft es natürlich in einem bestimmten Intervall ob überhaupt noch Benutzer die Internetseite geöffnet haben und Bilder anschauen. Sollte kein Benutzer mehr online sein, beendet sich das Programm von selbst.

Für den Transfer wird der FTP-Dienst⁷ verwendet.

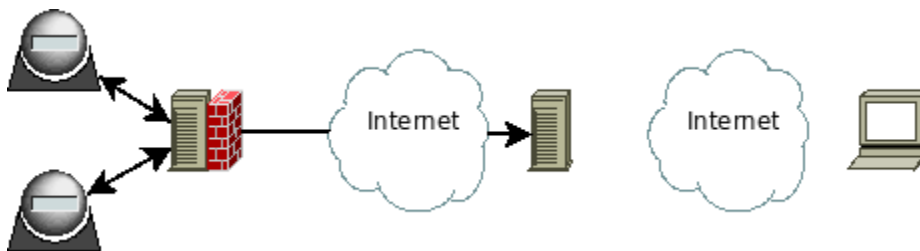


Abbildung 13: Schritt 5

Der Webserver prüft dabei jede Sekunde, ob ein neues Bild vorhanden ist und schickt es an den Client.

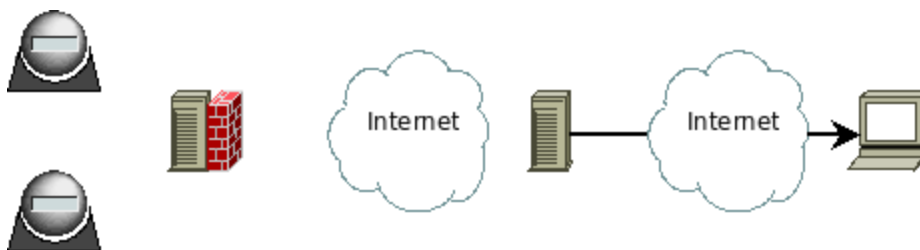


Abbildung 14: Schritt 6

⁷ FTP ist ein spezifiziertes Netzwerkprotokoll zur Übertragung von Dateien über TCP/IP-Netzwerken.

4.2 Lokale Skripten auf dem Router

Auf dem Router werden verschiedene PHP-Skripten benötigt, deren wichtigste Stellen nun kurz erläutert werden sollen. Die entsprechende Blöcke sind in den Softwarelistings farblich hervor gehoben.

Grundsätzlich werden die Ergebnisse der Skripten an den Webserver zurück gegeben. Dabei steht „0“ für einen Fehler, oder den „Auszustand“ und 1 für einen Erfolg, oder den „Anzustand“. Die beiden Zustände werden unter anderem bei der Kontrolle der Lichtsteuerung gebraucht.

4.2.1 webcam_config.php

Zunächst wird eine Konfigurationsdatei benötigt, in der verschiedene Parameter global definiert werden können. Diese Datei wird in jedes Skript eingebunden und somit müssen Änderungen nur an einer Stelle gemacht werden, die sich dann auf alle anderen Skripten auswirken.

Zeile 7	Hier wird die Sperrdatei definiert, die benötigt wird, damit das Skript, welches die Kamerabilder transferiert, nicht zweimal gestartet werden kann.
Zeile 8	Der Pfad um verschiedene temporäre Dateien zu speichern.
Zeile 10 - 12	Hier werden die Pfade zu den verschiedenen Skripten beschrieben.
Zeile 14 - 19	Das sind die Zugangsdaten für den Webserver.
Zeile 20 - 36	Die drei Funktionen um das Licht zu steuern werden ebenfalls in der Konfigurationsdatei definiert, damit Sie in anderen Skripten zur Verfügung stehen.

4.2.2 loader.php

Dieses Skript wird als einziges direkt vom Webserver aufgerufen und empfängt somit sämtliche Befehle, die vom Webserver an den Linux-Router übertragen werden. Es startet gegebenenfalls die benötigten Dienste und gibt das Ergebnis zurück.

Zeile 4 - 14	An dieser Stelle wird, sofern die Sperrdatei nicht vorhanden ist, das Skript gestartet, welches die Kamerabilder transferiert.
Zeile 16 - 35	In diesem Codeblock wird das Licht an und ausgeschaltet.
Zeile 17 - 53	Im letzten Abschnitt wird schließlich die Bewegung der Kameras kontrolliert.

4.2.3 webcam_daemon.php

Darüber hinaus wird das Programm benötigt, welches die Bilder auf den Webserver überträgt. Dieses Skript wird vom Skript „loader.php“ gestartet und beendet sich bei Bedarf von alleine.

Zeile 10 - 14	An dieser Stelle werden die Curl ⁸ Parameter festgelegt.
Zeile 21 - 27	Sollte kein sich kein Benutzer mehr auf der Webseite befinden, wird der letzte Code-Block ausgeführt.
Zeile 31 - 39	Hier wird nun das aktuelle Kamerabild von der Software motion abgegriffen und in dem angegebenen Verzeichnis abgelegt.
Zeile 41 - 55	In diesem Abschnitt werden die beiden Bilder der Kameras auf den FTP-Server des Webserver geladen.
Zeile 61 - 64	Kurz bevor das Skript beendet wird, schaltet es alle aktiven Lichtquellen aus und entfernt die Sperrdatei, damit es bei Bedarf erneut gestartet werden kann.

4.2.4 webcam_ptz.php

Mit diesem Skript wird die SNK-Fähigkeit der Kameras kontrolliert. Leider ist das Kommandozeilen Programm, welches die Kameras bewegt, nicht in der Lage feste Positionswerte entgegen zu nehmen. Es lässt sich nur um einen variablen Winkel, nicht zu einem festen Winkel ausgehend von der Nullposition, bewegen. Außerdem ist es nicht möglich die aktuelle Position der Kameras abzufragen.

Daher muss die aktuelle Position ausgehend vom Nullpunkt separat mit geschrieben werden. Somit können trotzdem feste Positionen angefahren werden, in dem einfach die Differenz zwischen der gespeicherten aktuellen Position und der neuen Position berechnet wird. Um diese Differenz wird die Kamera dann bewegt und kann somit die neue feste Position anfahren.

Zeile 9 - 20	Mit diesem Bereich kann ein Reset der Kameraposition erreicht werden. Dabei wird die Kamera in die Nullposition zurück bewegt.
	Die nächsten vier Böcke beschreiben die jeweiligen Kamerapositionen der beiden Kameras jeweils als pan und tilt Werten ausgehen von Nullpunkt.
Zeile 32 - 44	Position 1
Zeile 45 - 49	Position 2
Zeile 51 - 55	Position 3
Zeile 57 - 69	Position 4
Zeile 77 - 88	Anschließend wird der Befehl zum Schwenken und Neigen der Kameras abgesetzt. Dabei wird jedoch vorher überprüft ob die Kamera überhaupt geschwenkt oder geneigt werden muss.

⁸ Curl ist ein Programm, um einzelne Dateien aus oder zum Internet ohne Browser zu transferieren. Es arbeitet meist schneller als die eingebauten Mechanismen. Beispielsweise ist es in einem eigenen Test 2 bis 3 mal so schnell wie die PHP interne FTP Funktion gewesen.

4.3 Webseite auf dem Webserver

Die Webseite auf dem Webserver ist bewusst simpel gehalten und bietet verschiedene Unterseiten für die unterschiedlichen Funktionen. Die Größe ist dabei auf eine Breite für die Auflösung 800x600 optimiert.

Somit sieht die Webseite selbst auf mobilen Geräten akzeptabel aus - bietet aber dennoch bei größeren Bildschirmauflösungen eine ansprechende Optik.

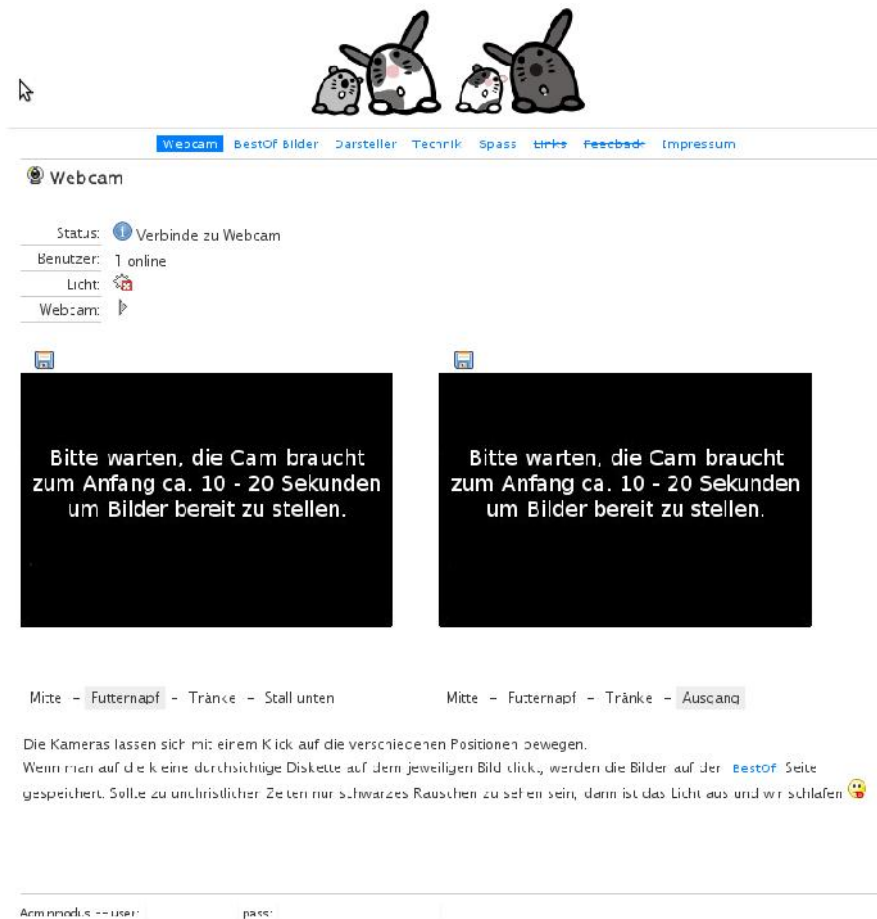


Abbildung 15: 1. Aufruf der Seite mit Ladebildern

4.3.1 Alles funktioniert mit Ajax

Ajax ist ein Acronym für die Wortfolge „Asynchronous JavaScript and XML“. Es bezeichnet ein Konzept der asynchronen Datenübertragung zwischen einem Server und dem Browser. Dieses ermöglicht es innerhalb einer HTML-Seite eine HTTP-Anfrage durchzuführen, ohne die Seite komplett neu laden zu müssen.

Die Besonderheit besteht in der Tatsache, dass nur gewisse Teile einer HTML-Seite oder auch reine Nutzdaten sukzessiv bei Bedarf nachgeladen werden, womit Ajax eine Schlüsseltechnik zur Realisierung des Web 2.0 darstellt.

Somit lässt sich beispielsweise das Kommando zum Drehen der Kamera mit einem Klick auf ein Bild absetzen. Es wird im Hintergrund übertragen und das Ergebnis als Text oder auch als Bild dargestellt. Bei der gesamten Browsersession muss die Seite nicht ein einziges Mal neu geladen werden.

Es werden ganz einfach bestehende HTML-Elemente ergänzt, ersetzt oder auch neue hinzugefügt oder alte entfernt.

4.3.2 Die Ajax-Bibliothek MooTools

Um die Ajax-Funktionalität möglichst einfach zu gewährleisten, wird eine Bibliothek benötigt. Somit muss nichts mehr selbst geschrieben werden und es können einfach vorgefertigte Elemente verwendet werden.

Es sind verschieden fertige Bibliotheken im Internet frei verfügbar. Sie bieten meist alle den gleichen Umfang und unterscheiden sich nicht großartig von einander. Die Auswahl erfolgte daher aufgrund persönlicher Erfahrungen, da schon früher bereits mit MooTools gearbeitet wurde.

MooTools selbst ist eine freie Javascript-Bibliothek zur einfachen Entwicklung von Ajax-Webseiten. Diese Seiten sind in allen modernen Browsern lauffähig. Dazu gehören Internet Explorer ab Version 6, Firefox ab Version 2 und Safari ab Version 2.

MooTools bietet neben der Möglichkeit von grafischen Effekten aber auch verschiedene andere hilfreiche Unterstützungen.

4.3.3 Administrative Funktionen

Auch die Administration der Seite lässt sich bequem mit Ajax realisieren. Im Fuß der Seite befinden sich zwei Eingabefelder für einen Benutzernamen und ein Passwort, die verschiedene Funktionen freischalten.

Als Admin ist es möglich ein Bild aus dem Archiv zu löschen, oder auch das Licht zu kontrollieren. Natürlich gibt es auch abgestufte Rechte, damit zum Beispiel einzelne Nutzer nur das Licht bedienen, nicht aber Bilder löschen können.

Die Rechte werden in einer PHP-Datei abgelegt und können je nach Bedarf natürlich auch geändert werden.

4.3.4 Bilderarchiv Funktion

Natürlich wäre die Seite nur halb so interessant, wenn man lustige Bilder nicht in einem Archiv ablegen könnte.

Daher gibt es über jeden Kamerabild ein kleines Icon in Form einer Diskette. Die Funktion dahinter nimmt das aktuell gezeigte Bild und legt es in einem speziellen Ordner ab. Dieser wird dann beim Aufruf des Archivs ausgelesen und dargestellt.

Natürlich funktioniert auch das alles mit Ajax, so dass der Benutzer weder eine neue Seite aufrufen, noch die alte verlassen muss. Alle Funktionen werden transparent im Hintergrund abgewickelt und stören somit nicht die Browser-Sitzung und das Betrachten der aktuellen Kamerabilder.

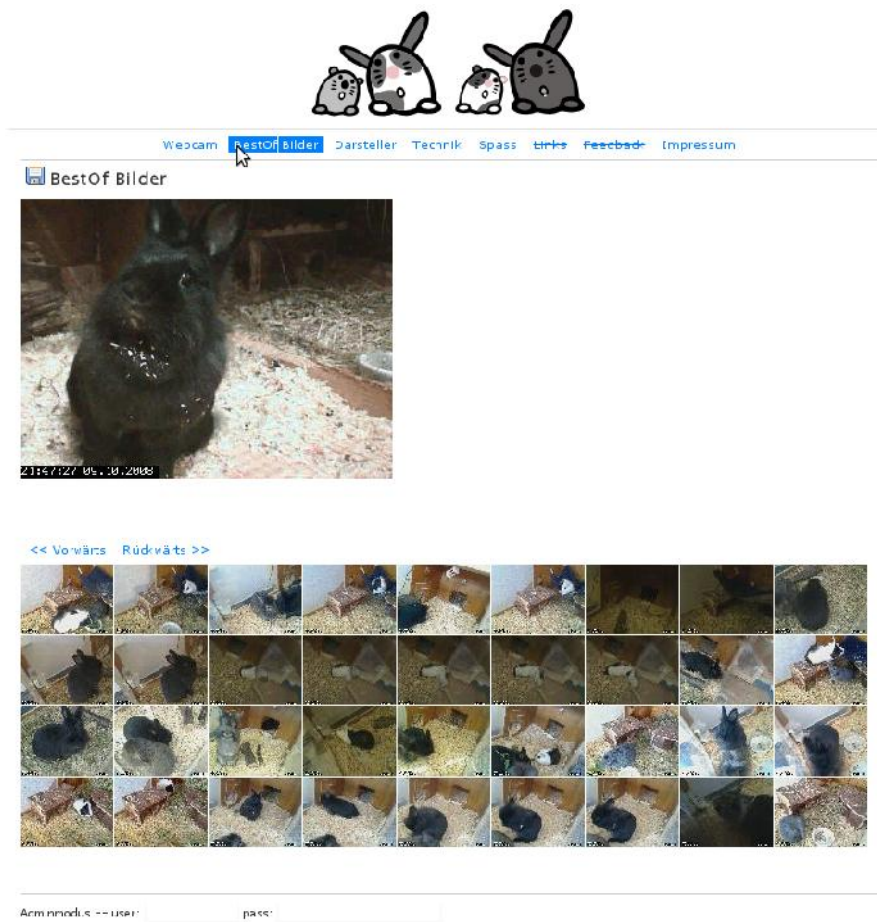


Abbildung 16: Bildergalerie mit gespeicherten Bildern

5 Test

Die Softwareentwicklung ist nun abgeschlossen und die Funktionstüchtigkeit kann jetzt durch verschiedene Tests geprüft werden.

Zum einen wird in verschiedenen Abständen die Webseite des Webservers aufgerufen und überprüft, ob alle Skripten korrekt arbeiten. Zum anderen werden die Prozesse auf dem Linux-Router beobachtet - in wie fern dort eine Anomalie auftritt oder beispielsweise eine hohe Systemlast erzeugt wird.

5.1 Auftretende Probleme

5.1.1 Ajax / Javascript blockiert manchmal

Im Dauerbetrieb der Ajax-Skripten auf dem Webserver ist ein Problem aufgetreten, welches die ganze Webseite zum Stillstand bringen kann.

Per Ajax werden im Hintergrund verschiedene PHP-Skripten per HTTP aufgerufen, und das Ergebnis ausgewertet. Dabei kann es vorkommen, das eine Funktion in der Ajax-Bibliothek nichts zurück gibt und einfach „hängen“ bleibt.

Die aufrufende Javascript-Funktion wartet dann ewig auf die Rückgabe der Werte und blockiert den Programmablauf.

Als Lösung wurde ein Timer in die Ajax-Abfragen integriert. Dadurch kann festgelegt werden, das eine Funktion nur eine bestimmte Zeitspanne zur Verfügung hat. Danach wird das Ergebnis einfach verworfen und die Funktion erneut ausgeführt.

Mit dieser Möglichkeit läuft nun auch in Dauertests die Webseite ohne nennenswerte Probleme oder Programmhänger durch.

5.1.2 Das Ladeskript beendet sich sofort

Zwischendurch kam es einmal zu einem Problem, bei dem sich das Skript zum Laden der Bilder gleich nach dem Aufruf sofort beendet hat. Allerdings ist dies ohne eine Änderung am Quellcode und an den restlichen relevanten Parametern passiert.

Nach einigem hin und her wurde schließlich die Uhrzeit als Problem erkannt. Scheinbar geht die Uhr des Linux-Routers etwas vor und ohne eine Korrektur haben sich nach 2 Monaten mehr als 2 Minuten summiert.

Da die Uhr von Webserver diese Differenz nicht aufwies, führte das dazu, das sich die Zeitstempel von einer wichtigen Datei unterschieden. Diese Datei gibt an, ab wann der letzte Benutzer den Webserver verlassen hat. Das Skript zum Laden der Bilder nahm nun also an, das der letzte Besucher vor 2 Minuten auf dem Webserver war. Da dies als Deadline in der Konfiguration eingetragen ist, hat es sich einfach sofort beendet.

Als Lösung wurde einfach der NTP-Dienst⁹ des Linux-Routers so eingestellt, das er täglich die Uhrzeit mit einem Zeitserver abgleicht. Somit gehören Zeitdifferenzen der Vergangenheit an.

⁹ Das Network Time Protocol ist ein Standard zur Synchronisierung von Uhren in Computersystemen über TCP/IP-Netzwerke.

5.1.3 Webcam bewegt sich fehlerhaft

5.1.3.1 Neustart nach einer Kernel Panic

Auch Linux ist nicht vor Bluescreens¹⁰ gefeilt, nur heißt es hier „Kernel-Panic“. Die Folgen sind jedoch die gleichen - das System steht und muss von Hand neu gestartet werden.

Ungefähr zwei bis drei mal im Jahr tritt dieses Phänomen auch beim Linux-Router auf. Er wird dann einfach neu gestartet und alle nötigen Dienste stehen nach ca. 2 Minuten Startdauer wieder automatisch zur Verfügung.

5.1.3.2 Die Webcam dreht durch

Nach einem dieser Abstürze, wurde der Rechner neu hochgefahren und ab dem Zeitpunkt kam es zur einer Anomalie bei der Kontrolle der Kamerabewegung. Beim horizontalen Schwenken hat sich die Kamera immer auch ein Stück vertikal geneigt. Beim Neigen hat sie dann ein Stück geschwenkt.

Da bei der Kamera Bewegung feste Positionen verwendet werden ist das natürlich ein großes Problem. Die Kameras sind auf Grund des 180° Winkel in der Lage, aus dem Käfig heraus in das Wohnzimmer zu sehen.

Das Problem war auch nach einem erneuten Neustart reproduzierbar und eine Lösung auch durch die Aneinanderreihung verschiedener Kommandos nicht möglich. Auch im Internet war keine brauchbare Lösung zu finden.

5.1.3.3 Ein neuer Kernel als Verursacher

Als nächstes wurden die Dinge, die sich seit dem letzten Start verändert haben, genauer angeschaut und dabei eine Veränderung zum letzten Start festgestellt.

Da Ubuntu in unregelmäßigen Abständen Programm-Updates bereit stellt, werden diese auch ein Mal in der Woche manuell eingespielt. In einem dieser Updates, die zwischen dem letzten funktionierenden Start und dem jetzigen lagen, wurde eine neue Kernel-Version¹¹ installiert. Diese wird standardmäßig so eingerichtet, das bei einem Neustart immer der neueste Kernel genommen wird.

Scheinbar ist in diesem Kernel irgendetwas durcheinander geraten und beeinträchtigt die Kamerafunktionen. Da keine neue Funktionalität hinzugekommen ist, die die Benutzung des neuen Kernels notwendig macht, wurde einfach wieder der alte verwendet.

Dieser startet nun nach einer Änderung der Konfiguration bei einem Neustart automatisch und bietet wieder den fehlerlosen Kamera Zugang.

10 Ein Bluescreen ist eine Beschreibung einer bestimmten Kategorie von Fehlermeldungen, häufig zu sehen auf einem Windows-PC. Um Schäden an Betriebssystem und Hardware zu verhindern, wird nach einem kritischen Systemfehler das System gestoppt und die Bedienoberfläche des Betriebssystems vollständig durch einen blauen Bildschirm ersetzt, auf dem in weißer Schrift die Fehlerinformationen erscheinen.

11 Der Kernel ist der zentrale Bestandteil eines Betriebssystems. In ihm ist die Prozess- und Datenorganisation festgelegt, auf der alle weiteren Softwarebestandteile des Betriebssystems aufbauen. Er bildet die unterste Softwareschicht des Systems und hat direkten Zugriff auf die Hardware.

5.1.3.4 Wie lange kann diese Lösung halten

Es bleibt abzuwarten, wie lange die Benutzung eines alten Kernels möglich ist. Die Ubuntu-Version auf dem Linux-Router ist eine LTS-Version. LTS heißt „Long Term Support“ und bedeutet, dass die Server-Version bis April 2013 offiziell unterstützt wird. Bis zu diesem Datum wird es Updates und Patches geben - danach ist man bei auftretenden Bugs oder Sicherheitslücken auf sich allein gestellt.

Der Kernel und die Einstellungen einer älteren Version sind stellenweise nicht mit denen einer neueren kompatibel. Daher muss man bei einem Versionswechsel auch einen neuen Kernel akzeptieren.

Es sind zwar noch knapp 4 Jahre Zeit - sollte aber bis zum April 2013 der „SNK-Fehler“ im Kernel nicht verschwinden, muss zu gegebener Zeit eine andere Möglichkeit in Betracht gezogen werden. Entweder es findet sich eine Umgehung des Problems oder aber es wird die alte Version beibehalten. Dabei entsteht allerdings ein Sicherheitsrisiko aufgrund der dann fehlenden Fehlerkorrekturen.

5.1.4 Probleme bei USB Ports

Beim Test sind wiederholt kleinere Probleme bei der simultanen Benutzung der zwei USB-Kameras aufgetreten. Wenn zeitgleich die Bilder beider Kameras abgefragt werden, kommt es auf dem Bild der zweiten Kamera gelegentlich zu Artefakten und fehlerhaften Bildern.



Abbildung 17: Kamerabild mit Artefakten

Der USB 2.0 Standard erlaubt eine maximale Bandbreite von 480 Mbit/s - das sind 60 MB pro Sekunde. Da zum Testen kein weiteres USB-Gerät an den Rechner angeschlossen ist, sind somit maximal 30 MB/s pro Kamera möglich.

Die Kamera besitzt einen zwei Megapixel Sensor, der Bilder mit der Größe von 1600 x 1200 Pixeln erlaubt. Leider ließ sich nicht herausfinden, ob die Bilder von der Kameraelektronik tatsächlich in dieser Größe und in welchem Format über das USB Kabel transportiert werden. Zum Beispiel sind JPEG¹² komprimierte Bilder selbst in der besten Qualitätsstufe in dieser Auflösung maximal ein MB

¹² JPEG ist eine Norm, die verschiedene Methoden der Bildkompression beschreibt. Damit lässt sich die Dateigröße vieler Fotos ohne große Einbußen in der Bildqualität verringern.

groß. Es sollte somit rein rechnerisch problemlos möglich sein diese Bilder ohne Bildfehler zu übertragen.

Da dies jedoch auch unter verschiedenen Konfigurationen nicht möglich war, wurde eine separate USB PCI-Karte¹³ gekauft, da der Router noch einen freien PCI-Slot besaß. Damit ist nun problemlos die gleichzeitige, fehlerlose Aufnahme von zwei Kamerabildern möglich.

5.1.5 Der Rest funktioniert ohne Probleme

Glücklicherweise sind die vorherigen Punkte die einzigen Probleme. Alles andere funktioniert problemlos. Es ließen sich in einem Dauertest beide Kameras immer wieder hin und her drehen, Bilder im Archiv ablegen und wieder entfernen und auch die Lichtkontrolle funktionierte auf Dauer tadellos.

Somit stand einer Live-Schaltung nichts mehr im Wege.

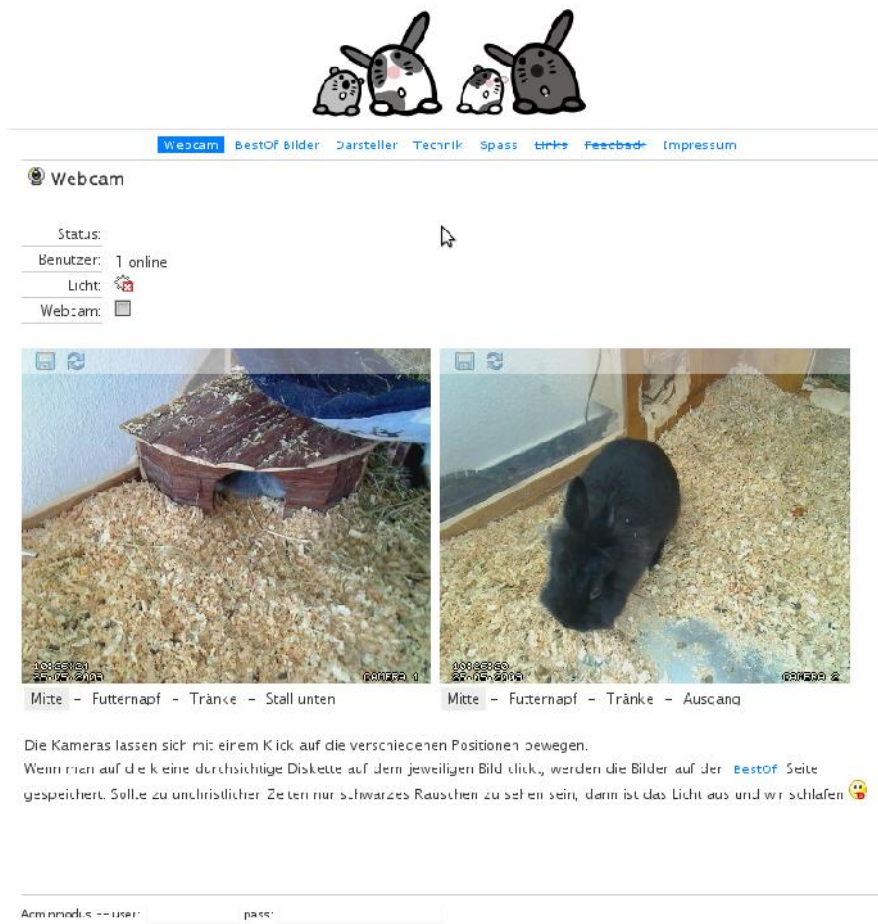


Abbildung 18: Live-Kamerabilder mit Aktualisierungsfunktion

¹³ PCI ist ein Bus-Standard um verschiedene Peripheriegeräte mit dem Chipsatz eines Prozessors zu verbinden.

6 Zusammenfassung

6.1 Fazit

Die Entwicklung der Kamera-Webseite und der dazugehörigen Skripten ist trotz der kleinen Rückschläge ein voller Erfolg gewesen.

Dabei konnten bereits vorhandene Komponenten, wie zum Beispiel der Linux-Rechner, sinnvoll zum Einsatz kommen. Darüber hinaus benötigte Hardware konnte kostengünstig erworben werden um die bestehende zu verstärken.

Die beiden Kameras laufen seit einem halben Jahr ununterbrochen durch und abgesehen von den vorher angemerkten Problemen, musste währenddessen keine Korrektur der Parameter oder der Software erfolgen. Es wurden die Zielsetzungen des Projektes somit durchaus erfüllt.

6.2 Aussicht

Für die Zukunft ist auf jeden Fall der Ausbau der Webseite geplant. Es soll beispielsweise möglich sein, dass man Archivbilder kommentieren kann. Des Weiteren ist ein Gästebuch und eine Anleitung geplant, wie andere Leute die Kamera Anbindung für ihre eigenen Haustiere nachbauen können.

Außerdem sind noch drei Steckplätze der Steckdosenleiste frei, so dass sich noch andere interessante Möglichkeiten bieten.

Natürlich muss die Webseite und die Skripten immer wieder auf Fehler überprüft werden und verschiedene Tests gemacht werden. Nur so ist sichergestellt, dass alles automatisiert und ohne manuelles Eingreifen auf Dauer funktionieren kann.

7 Softwarelistings

7.1 webcam_config.php

```
1 <?
2 $debug          = false;
3 $time_sleep    = 0;
4 $file_ptz_pos  = "/tmp/webcam_ptz";
5 $min_ptz_time  = 5;
6
7 $file_lock     = "/tmp/webcam_daemon.lock";
8 $file_path     = "/tmp/motion/";
9
10 $file_run      = "/home/lars/daemon/webcam_daemon.php";
11 $file_jabber   = "/home/lars/daemon/webcam_jabber.php";
12 $file_ptz      = "/home/lars/daemon/webcam_ptz.php";
13
14 $remote_host   = "www.bimmel-bormel.de";
15 $ftp_user      = "29452-bimmel-bormel";
16 $ftp_pass      = "*****";
17 $ftp_port      = 21;
18 $update_url    = "http://".$remote_host."/command.php?action=update";
19 $update_ptz_url = "http://".$remote_host."/command.php?action=update_ptz";
20
21 function light_on($id)
22 {
23     exec("sudo sispmtl -o $id > /dev/null &");
24     return 1;
25 }
26 function light_off($id)
27 {
28     exec("sudo sispmtl -f $id > /dev/null &");
29     return 1;
30 }
31 function light_state($id)
32 {
33     $str = exec("sudo sispmtl -g $id");
34     if(strpos($str, "on") > 0) return 1;
35     else return 0;
36 }
37 ?>
```

Tabelle 1: webcam_config.php

7.2 loader.php

```
1 <?
2 require_once( "/home/lars/daemon/webcam_config.php");
3
4 if($_GET['type'] == "webcam")
5 {
6     if(!file_exists($file_lock))
7     {
8         echo "1";
```

```
9     fclose(fopen($file_lock, 'w'));
10     exec("php $file_run > /dev/null &");
11 }
12 else
13     echo "0";
14 }
15
16 elseif($_GET['type'] == "light")
17 {
18     if(!$_GET['id'] || $_GET['id'] == "1")
19         $id = 4;
20
21     if($_GET['action'] == "on")
22         echo light_on($id);
23     elseif($_GET['action'] == "off")
24         echo light_off($id);
25
26     elseif($_GET['action'] == "flip")
27     {
28         if(light_state($id))
29             echo light_off($id);
30         else
31             echo light_on($id);
32     }
33     elseif($_GET['action'] == "state")
34         echo light_state($id);
35 }
36
37 elseif($_GET['type'] == "ptz")
38 {
39     $id = $_GET['id'] ? $_GET['id'] : 1;
40     $pos = $_GET['pos'] ? $_GET['pos'] : 0;
41
42     clearstatcache();
43     if(efilemtime($file_ptz_pos.$id) + $min_ptz_time < time())
44     {
45         if($_GET['action'] == "move")
46             exec("php $file_ptz $id $pos > /dev/null &");
47         elseif($_GET['action'] == "reset")
48             exec("php $file_ptz $id > /dev/null &");
49         echo "1";
50     }
51     else
52         echo 0;
53 }
54 ?>
```

Tabelle 2: loader.php

7.3 webcam_daemon.php

```
1 <?
2 require_once("/home/lars/daemon/webcam_config.php");
3 set_time_limit(0);
4 ignore_user_abort(true);
```

Kapitel 7 - Softwarelistings

```
5  fclose(fopen($file_lock, 'w'));
6
7  $time_update_session= 0;
8  $ftp_id = 0;
9
10 $curl = curl_init();
11 curl_setopt($curl, CURLOPT_PORT, $ftp_port);
12 curl_setopt($curl, CURLOPT_USERPWD, "$ftp_user:$ftp_pass");
13 curl_setopt($curl, CURLOPT_UPLOAD, true);
14 curl_setopt($curl, CURLOPT_TIMEOUT, 3);
15
16 $time_snap = array(0, 0, 0);
17 $time_load = array(0, 0, 0);
18
19 while (true)
20 {
21     if($time_update_session < time())
22     {
23         $time_update_session = file($update_url);
24         $time_update_session = $time_update_session[0];
25         if($time_update_session < time())
26             break;
27     }
28
29     for($a = 1; $a <= 2; $a++)
30     {
31         $file = $file_path.$a."_".date("Y-m-d-H-i-s", time())."-snap.jpg";
32         if($time_snap[$a] < time() && !is_file($file))
33         {
34             echo "snaping $file\n";
35             $fp = fopen("http://127.0.0.1:8080/$a/action/snapshot", "r");
36             fclose($fp);
37             clearstatcache();
38             $time_snap[$a] = time();
39         }
40
41         if($time_load[$a] < time())
42         {
43             $file = $file_path.$a."_".date("Y-m-d-H-i-s", time() - 1)."-snap.jpg";
44             if(is_file($file))
45             {
46                 echo "uploading $file\n";
47                 $fp = fopen($file, "r");
48                 curl_setopt($curl, CURLOPT_URL, "ftp://$remote_host/images/cam/$a/".(time() - 1)."_$a.jpg");
49                 curl_setopt($curl, CURLOPT_INFILE, $fp);
50                 curl_setopt($curl, CURLOPT_INFILESIZE, filesize($file));
51                 curl_exec($curl);
52                 unlink($file);
53                 $time_load[$a] = time();
54             }
55         }
56     }
57     if($time_sleep)
58         sleep($time_sleep);
59 }
```

```
60 curl_close($curl);
61 light_off("all");
62
63 if(file_exists($file_lock))
64     unlink($file_lock);
65 ?>
```

Tabelle 3: *webcam_daemon.php*

7.4 webcam_ptz.php

```
1 <?
2 require_once("/home/lars/daemon/webcam_config.php");
3 set_time_limit(0);
4 ignore_user_abort(true);
5
6 $id = (int)$argv[1];
7 $pos = (int)$argv[2];
8
9 if($pos == "" || $pos == 0 || !$pos)
10 {
11     $pos = 1;
12     $file = fopen($file_ptz_pos.$id, 'w');
13     fwrite($file, "0\n0\n");
14     fclose($file);
15
16     exec("sudo /usr/local/bin/uvcdynctrl -d video".($id - 1). " -s \"Pan Reset \" 1");
17     sleep(2);
18     exec("sudo /usr/local/bin/uvcdynctrl -d video".($id - 1). " -s \"Tilt Reset \" 1");
19     sleep(2);
20 }
21
22 $span = 0;
23 $tilt = 0;
24 $file = fopen($file_ptz_pos.$id, 'r+');
25
26 $span_cur = (int)trim(fgets($file));
27 $tilt_cur = (int)trim(fgets($file));
28 fseek($file, 0);
29
30 switch($pos)
31 {
32     // mitte
33     case 1:
34         if($id == 1)
35         {
36             $span = 500;
37             $tilt = 0;
38         }
39         else if($id == 2)
40         {
41             $span = 1000;
42             $tilt = 500;
43         }
44     break;
```

Kapitel 7 - Softwarelistings

```
45 // napf
46 case 2:
47     $span = 0;
48     $tilt = -1800;
49     break;
50
51 // tränke
52 case 3:
53     $span = -1500;
54     $tilt = -600;
55     break;
56
57 // unten / ausgang
58 case 4:
59     if($id == 1)
60     {
61         $span = 2250;
62         $tilt = -1800;
63     }
64     else if($id == 2)
65     {
66         $span = 2500;
67         $tilt = -500;
68     }
69     break;
70 }
71 $span_cur = $span - $span_cur;
72 $tilt_cur = $tilt - $tilt_cur;
73
74 fwrite($file, "$span\n$tilt\n");
75 fclose($file);
76
77 if($span_cur != 0)
78 {
79     $vid = $id - 1;
80     exec("sudo /usr/local/bin/uvcctrl -d video$vid -s \"Pan (relative)\" -- $span_cur");
81     sleep(2);
82 }
83 if($tilt_cur != 0)
84 {
85     $vid = $id - 1;
86     exec("sudo /usr/local/bin/uvcctrl -d video$vid -s \"Tilt (relative)\" -- $tilt_cur");
87     sleep(2);
88 }
89
90 $url = fopen($update_ptz_url."&id=$id&pos=$pos", "r");
91 fclose($url);
92 ?>
```

Tabelle 4: webcam_ptz.php

8 Literaturverzeichnis

<http://www.jetway.com.tw/jetway/system/products/show2.asp?id=391&prname=J7F4K1GE>

(17. Mai 2009)

http://www.logitech.com/index.cfm/webcam_communications/webcams/devices/3480&cl=de,de

(17. Mai 2009)

<http://de.wikipedia.org/wiki/LED>

(17. Mai 2009)

http://www.via.com.tw/en/downloads/whitepapers/initiatives/spearhead/ini_mini-itx.pdf

(21. Mai 2009)

<http://www.usb-infos.de/>

(21. Mai 2009)

http://wiki.ubuntuusers.de/Was_ist_Ubuntu

(21. Mai 2009)

<http://www.lavrsen.dk/twiki/bin/view/Motion/WebHome>

(21. Mai 2009)

<http://www.jpeg.org/>

(24. Mai 2009)

<http://www.elektronik-kompodium.de/sites/com/0310091.htm>

(24. Mai 2009)

<http://de.php.net/>

(24. Mai 2009)

<http://www.boutell.com/gd/>

(25. Mai 2009)

<http://curl.haxx.se/>

(10. Juni 2009)

http://ajaxpatterns.org/On-Demand_Javascript

(10. Juni 2009)

<http://www.adaptivepath.com/ideas/essays/archives/000385.php>

(10. Juni 2009)

http://www.jasik.de/shutdown/stop_fehler.htm

(10. Juni 2009)

<http://de.wikipedia.org/wiki/Betriebssystemkern>

(10. Juni 2009)

9 Abbildungsverzeichnis

Alle Bilder, Grafiken, Screenshots und ähnliches wurden von mir selbst erstellt.

Abbildung 1: Holzkäfig mit den beiden Kameras und vier Bewohnern	4
Abbildung 2: Mini ITX Gehäuse mit 12cm Lüfter	5
Abbildung 3: Sieben Zoll Touchscreen mit Bildern der beiden Kameras	6
Abbildung 4: Einer von neun LED-Spots in einem Holzkreis	7
Abbildung 5: USB-Steckdosenleiste - der braune Stecker ist für die LEDs	8
Abbildung 6: Eine von zwei Logitech „Sphere AF“ Kameras überkopf hängend	9
Abbildung 7: Kommunikationsweg 1	10
Abbildung 8: Kommunikationsweg 2	11
Abbildung 9: Überblick über Kommunikation	14
Abbildung 10: Schritt 1	14
Abbildung 11: Schritt 2	14
Abbildung 12: Schritt 4	15
Abbildung 13: Schritt 5	15
Abbildung 14: Schritt 6	15
Abbildung 15: 1. Aufruf der Seite mit Ladebildern	18
Abbildung 16: Bildergalerie mit gespeicherten Bildern	20
Abbildung 17: Kamerabild mit Artefakten	23
Abbildung 18: Live-Kamerabilder mit Aktualisierungsfunktion	24

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel sind angegeben. Die Arbeit hat mit gleichem bzw. in wesentlichen Teilen gleichem Inhalt noch keiner Prüfungsbehörde vorgelegen.

Unterschrift

Ort, Datum