

Spyhole



BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN

University of Applied Sciences

Abschlussbericht für die Lehrveranstaltungen

Entwurf Eingebetteter Systeme bei Prof. Dr.-Ing. Alfred Rozek
Netzwerk-Programmierung bei Prof. Dr.-Ing. René Görlich

Eingereicht von:

Matthias Hansert s791744

Marcus Perkowski s798936

Eren Büyükkirali s805833

Inhaltsverzeichnis

1	Einleitung	2
2	Raspberry Pi	3
2.1	Hardware	4
2.1.1	Technische Spezifikationen	4
2.1.2	Kamera	5
2.2	Software und Einstellungen	6
2.2.1	Betriebssystem	6
2.2.2	LAMP Webserver	6
2.2.3	SSH Zugriff	6
2.2.4	DNS	7
2.2.5	Datenbanken	8
2.3	OpenCV und Gesichtserkennung	10
2.3.1	Logarithmus trainieren	10
2.3.2	Kamera initialisieren	10
2.3.3	Bild auswerten	11
2.4	MJPEG Streamer	11
2.5	Socket	12
3	Clients	13
3.1	Konzept	14
3.1.1	Struktur und Aufbau der Applikationen	14
3.1.2	Anmelden des Users	14
3.1.3	Registrierung	14
3.1.4	Stream starten und Tür öffnen	15
3.1.5	Datenbankverbindung	15
4	Technische Umsetzung	16
4.1	Struktur und Aufbau der App	16

<i>INHALTSVERZEICHNIS</i>	1
4.1.1 Login	16
4.1.2 Registrierung	16
4.1.3 Datenbankverbindung	17
4.1.4 Control	18
4.1.5 Bilder Log	19
5 Aktueller Projektzustand	20
A	23
A.1 Projektüberblick	23
A.2 Desktopanwendung	24
A.3 Android app	25
Literatur- und Quellenverzeichnis	27

Kapitel 1

Einleitung

Der stetige technologische Fortschritt sorgt in allen Lebensbereichen für immer neue Erfindungen und Ideen. Eine der größten Entwicklungsbereiche für Privatanwender liegt im intelligenten Wohnen. Hierbei handelt es sich im Allgemeinen um technische Hilfsmittel, die einer Person oder einer Personengruppe das Leben erleichtert. Die Entwicklung dieser Geräte wird durch leistungsfähige Hardware sowie die mächtigen Entwicklungswerkzeuge ermöglicht. Die geringen Kosten und die umfangreiche Dokumentation im Internet machen es auch Privatanwendern möglich ihre eigenen Ideen zu verwirklichen.

Ein Teilbereich des intelligenten Wohnens ist die Überwachung, in den sich auch dieses Projekt einordnen lässt. Mit dem *Spyhole* soll es möglich werden, Kontrolle und Sicherheit über die Eingangstür zu bekommen. Die Idee ist, von überall und jederzeit durch den Türspion seiner Wohnung schauen zu können. Weiterhin ist das ferngesteuerte Öffnen sowie das Abfragen der letzten Besucher eine erstrebenswerte Funktionalität. In Zeiten dauerhafter Vernetzung und der Smartphones steht es auch außer Frage, dass eine entsprechende Applikation für diese Systeme bereitgestellt werden muss. Es ist jedoch ebenso an Alternativsysteme zu denken, da es zur Projektvision gehört den Zugriff von überall zu ermöglichen.

Als Grundlage für dieses Projekt soll dabei hardwareseitig das Raspberry Pi verwendet werden, worauf in Kapitel zwei eingegangen wird.

Um ein besseren Überblick des Projektes zu haben befindet sich im Anhang A.1 eine Übersicht der Komponenten und Struktur des Projektes.

Kapitel 2

Raspberry Pi

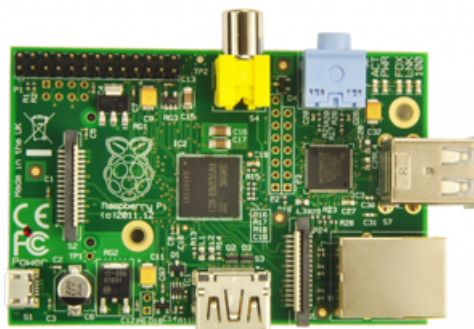


Abbildung 2.1: Raspberry Pi Model B

Das Raspberry Pi ist ein kreditkartengroßer Einplatinencomputer von der *Raspberry Pi Foundation*¹ entwickelt wurde. Mithilfe seiner 700 MHz starken ARM-CPU kann er nahezu alles, was auch normale Desktoprechner können.

Ziel der Entwicklung des Raspberry Pi ist das Interesse an dem Studium der Informatik und ähnliche Fachgebiete zu fördern, bzw. einen günstigen Computer zum Programmieren und Experimentieren anbieten zu können.

Durch die geringe Leistungsaufnahme, günstigen Preis und viele verschiedene Ausbau- und Personalisierungsmöglichkeiten können Raspberry Pi's für verschiedene Zwecke eingesetzt werden. Unter anderem sind beliebte Projekte für den Haushalt, Wetterstationen, Radiosender, Media Center, NAS, etc.

¹Stiftung und Wohltätigkeitsorganisation in Großbritannien

2.1 Hardware

2.1.1 Technische Spezifikationen

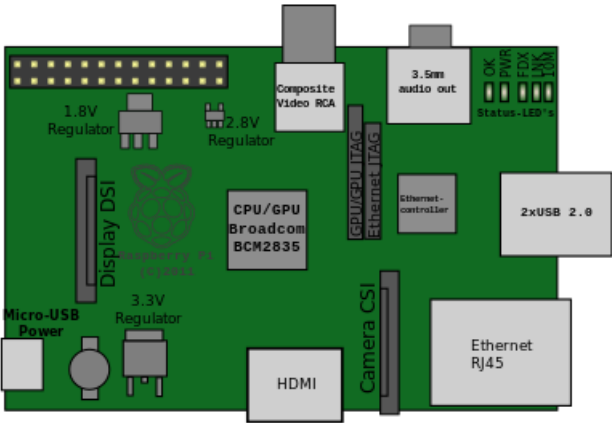


Abbildung 2.2: Raspberry Pi Model B Komponenten

Komponente	Kenndaten
Größe	85,60 x 53,98 x 17 mm
Soc	Broadcom BCM2835
CPU	ARM1176JZF-S(700MHz)
GPU	Broadcom VideoCore IV
SDRAM	512MB
USB 2.0	2
Videoausgabe	HDMI & S-Video
Tonausgabe	3,5mm Klinkenstecker & HDMI
Speicher	SD(SDHC/SDXC)/MMC/SDIO Kartenleser bis 128GB
Netzwerk	10/100 MBit Ethernet
Schnittstellen	17 GPIO-Pins, SPI, 21C, UART
Stromversorgung	5V Micro-USB Anschluss

[1][2]

Während der Entwicklung dieses Projektes haben wir bemerkt, dass der Prozessor sehr belastet wird (besonders wenn die Gesichtserkennung aktiviert wird). Darum haben wir uns als Team entschieden, den Raspberry Pi zu übertakten. Wir sind ein Kompromiss eingegangen, das Raspberry Pi auf eine mittlere Übertaktung einzustellen.

2.1.2 Kamera



2.2 Software und Einstellungen

2.2.1 Betriebssystem

Für das Raspberry Pi wurden mehrere Open-Source Betriebssysteme programmiert. Alle basieren auf verschiedene Linux Distributionen (u.a. OpenSUSE, Fedora, FreeBSD, Debian). Je nach Bedarf kann sich jeder Nutzer für eine Distribution entscheiden. Eine der beliebtesten Distributionen ist Raspbian, basierend auf Debian, was auch in diesem Projekt benutzt wird. Raspbian beinhaltet alle Grundprogramme und -Utilities, die mit vielen verschiedenen Packages kompatibel sind. Dieses Betriebssystem hat sich im Laufe der Zeit als sehr stabil bewiesen.

Es gibt auch die Möglichkeit, Windows auf das Raspberry Pi zu installieren, obwohl Nutzer davon abgeraten werden. Das Windows Betriebssystem benötigt zu viele Ressourcen und läuft sehr langsam und instabil auf das Raspberry Pi.

Andere Distributionen sind gebrauchorientiert aufgestellt, wie zum Beispiel für Media Center das beliebte XBMC (OpenELEC, Raspbmc, XBian). Weiterhin gibt es auch Android-Systeme, die auf das Raspberry Pi portiert worden sind. [3]

2.2.2 LAMP Webserver

LAMP steht für **L**inux **A**pache **M**ySQL **P**HP (sowie Perl und Python) Webserver. Auf das Raspberry Pi wurde ein LAMP Server installiert, um die Desktop/Mobil Applikation mit dem Pi zu verbinden und Informationen auszutauschen. Durch die Einstellung eines DNS kann auf das Video-Stream sowie auf die MySQL Datenbank zugegriffen werden. Damit *http* oder *ssh* Anfragen zum Server im Raspberry Pi ankommen, müssen Ports für die Verbindung zur Verfügung gestellt werden. Im diesen Fall muss jeweils ein Port für die SSH-Verbindung, den Videostream und die Datenbankzugriffe freigegeben werden.

Um den LAMP Webserver zu installieren, werden folgende Kommandos mit Administratorrechte ausgeführt:

```
apt-get install apache2
apt-get install mysql-server
sudo apt-get install php5
sudo apt-get install php5-mysql
```

2.2.3 SSH Zugriff

Damit das Team gleichzeitig auf dem Raspberry Pi arbeiten können, wurde auf das System eine SSH-Verbindung eingestellt. So wurden auch Entwicklungskosten gespart, weil nicht jedes Teammitglied ein System benötigte.

Um das Raspberry Pi per Fernzugriff zu betreiben, ist es nötig, eine *Secure Shell* Verbindung aufzubauen. Durch die Verfügung einer SSH-Verbindung können Updates, Support und Wartungsarbeiten per Fernzugriff auf das System ausgeübt werden, dieses spart Kosten für den Kunden sowie für den Support.

Als Sicherheitsmaßnahmen wurde der Standardport für SSH-Verbindungen (Port 22) auf anderen Port geändert und eine Public-Key Authentifizierung implementiert. Um den Port für die SSH-Verbindung zu ändern, muss in der Datei `/etc/ssh/sshd_config` den Wert `#Port` auf den gewünschten Wert gesetzt werden. Danach muss der Dienst neu gestartet werden und somit ist der neue Port eingestellt. Wichtig ist, dass der neu eingestellte Port im Router freigeschaltet wird, oder gezielt zur IP-Adresse des Systems weitergeleitet wird.

Die Public-Key Authentifizierung erfolgt, indem der Benutzer einen privaten und einen öffentlichen Schlüssel erstellt. Der private Schlüssel wird am Client gespeichert und der öffentliche Schlüssel am Server. Mit dem Befehl

```
ssh-keygen -t dsa
```

werden die Schlüssel generiert. Hier muss der Benutzer ein Passwort eingeben, mit dem sich er am Server anmelden wird. In der Regel heißen beiden Schlüssel `id_dsa` und `id_dsa.pub`. Der private Schlüssel muss am Client bekannt gemacht werden, indem man in der `/ssh2/identification` Datei die Zeile

```
idkey id_dsa
```

bearbeitet oder einbaut. Der Inhalt des öffentlichen Schlüssels muss in der Datei `/ssh/authorized_keys` hinzugefügt werden.

```
cat new_key.pub >> .ssh/authorized_keys
```

Jetzt ist der Client und der Server so konfiguriert, dass nur Rechner mit dem privaten Schlüssel Zugriff über den konfigurierten Port auf dem Server haben. Eine SSH-Verbindung wird mit dem Befehl folgendermaßen aufgebaut:

```
ssh -p XXXX benutzer@serverAdresse
```

2.2.4 DNS

Um das Raspberry Pi über das lokale Netzwerk erreichbar zu sein, wurde erstmal ein Webserver aufgebaut. Um jetzt das System überall über das Internet erreichbar zu sein, muss eine Domain reserviert werden. Somit kann ein Benutzer Zugriffe auf die Datenbank, sowie auf das Videostream aus jedem Rechner oder Android Mobilgerät weltweit ausüben. Der Dienst “no ip” bietet solche Dienstleistung kostenlos an. Für dieses Projekt wurde die Adresse `spyhole.no-ip.biz` eingestellt und durch die Freigabe und Weiterleitung von Ports können die programmierten Applikationen auf das Raspberry Pi zugreifen.

Da dieses Projekt in eine privaten Haushalt verwendet wird, besitzt in der Regel der Benutzer keine feste IP Adresse. Deswegen muss die IP Adresse hinter des DNS zyklisch geprüft werden. Der Dienst “no ip” stellt ein Programm zur Verfügung, das sich um die Überprüfung der Gültigkeit der IP Adresse kümmert. Das Programm muss dann mit den Kommandos

```
make  
make install
```

installiert werden. Während der Installation wird von den Benutzer das Login, Passwort und die Aktualisierungszeit verlangt. Danach wird der Dienst mit

```
/usr/local/bin/noip2
```

gestartet. Der Dienst läuft bis das System runtergefahren wird. Daher ist es ratsam, den Dienst im `/etc/init.d` einzutragen.

2.2.5 Datenbanken

Hier wird über die Datenbank und der Aufbau der Tabellen beschrieben. Als Datenbank benutzen wir MySQL-Server zur Speicherung der Daten. Zur Verwaltung der Daten in der Datenbank verwenden wir das Tool `phpMyAdmin`.

Was ist SQL ?

SQL ist einer der verbreitetsten Standards, um eine Kommunikation zwischen Server und Client mit Datenbanken zu ermöglichen. Mit SQL ist es möglich, Daten aus einer Datenbank zu selektieren, einzutragen, zu aktualisieren und zu löschen. Auch besteht die Möglichkeit mit Hilfe der SQL Tabellen in einer Datenbank zu erstellen, zu modifizieren oder zu löschen. Integriert ist ebenfalls eine Zugriffsverwaltung auf die Tabellen.

Das Schema der Datenbank legt fest, welche Daten in einer Datenbank in welcher Form gespeichert werden können und welche Beziehungen zwischen den Daten bestehen. Insbesondere bei relationalen Datenbanken legt das Schema die Tabellen und deren Attribute sowie zur Sicherstellung der Konsistenz die Integritätsbedingungen fest. Dabei wird auch speziell der Primärschlüssel bzw. die Eindeutigkeitsbedingungen festgelegt.

Was ist phpMyAdmin ?

`phpMyAdmin` ist ein freies praktisches PHP-Tool zur Verwaltung von MySQL-Datenbanken. Das Tool bietet die Möglichkeit über HTTP mit einem Browser in MySQL neue Datenbanken zu erstellen bzw. zu löschen. Des Weiteren erlaubt das Tool die Erstellung, Löschung und Veränderung von Tabellen und Datensätzen. Auch die Verwaltung von Schlüssel-Attributen ist implementiert. Eine Anwendungsmöglichkeit des Tools `phpMyAdmin` ist die SQL-Statements direkt auszuführen und somit eine ausführliche grafische Abfrage der vorhandenen Daten in Form von Tabellen darzustellen. Für die Bedienung des Tools sind kaum Kenntnisse in SQL erforderlich, da das Tool nach dem WYSIWYG² - Prinzip arbeitet.

Aufbau

Die Datenbank `db_spyhole` besteht aus drei Tabellen (`tb_doorloggers`, `tb_users` und `tb_images`). In der Tabelle `tb_doorloggers` werden die Tür-Aktivitäten (Zeitpunkt) gespeichert, in der Tabelle `tb_users` sind die registrierte Users gespeichert, und die Tabelle `tb_images` dient zur Speicherung der Bilder, die mithilfe der Kamera vom Raspberry Pi abgelichtet wurden. Die Bilder in der Tabelle `tb_images` sind binär abgelegt.

Die genauen Details der drei Tabellen:

```
--  
-- Tabellenstruktur für Tabelle 'tb_doorlogger'  
--  
  
CREATE TABLE IF NOT EXISTS `tb_doorlogger` (  
  `ID` int(8) NOT NULL AUTO_INCREMENT,  
  `U_ID` int(8) NOT NULL,  
  `date` datetime NOT NULL,
```

²ist das Akronym für das Prinzip „What You See Is What You Get“ (deutsch „Was du siehst, ist was du bekommst.“).

```

PRIMARY KEY (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
  AUTO_INCREMENT=1 ;

-----

--
-- Tabellenstruktur für Tabelle `tb_images`
--

CREATE TABLE IF NOT EXISTS `tb_images` (
  `ID` int(8) NOT NULL AUTO_INCREMENT,
  `dl_ID` int(8) NOT NULL,
  `U_ID` int(8) NOT NULL,
  `imgpath` varchar(255),
  `imgtype` varchar(32) COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
  AUTO_INCREMENT=1 ;

-----

--
-- Tabellenstruktur für Tabelle `tb_user`
--

CREATE TABLE IF NOT EXISTS `tb_user` (
  `ID` int(8) NOT NULL AUTO_INCREMENT,
  `name` varchar(64) COLLATE utf8_unicode_ci NOT NULL,
  `firstname` varchar(64) COLLATE utf8_unicode_ci NOT NULL,
  `email` varchar(128) COLLATE utf8_unicode_ci NOT NULL,
  `user` varchar(16) COLLATE utf8_unicode_ci NOT NULL,
  `password` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  `u_timestamp` timestamp NOT NULL DEFAULT '0000-00-00_00:00:00' ON
    UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
  AUTO_INCREMENT=1 ;

```

Listing 2.1: Aufbau der Datenbanktabellen

Das Attribut „ID“ vom Typ „INT“ kann Werte von -2147483648 bis 2147483647 erfassen. Die Attribute vom Typ „VARCHAR()“ können beliebige Zeichenkombinationen speichern. Die maximale Zeichenlänge ist dabei von der in der Klammer befindlichen Zahl abhängig. Der Befehl „NOT NULL“ in der Definition der Attribute bewirkt, dass die entsprechende Spalte mit Daten vorhanden sein muss, sobald ein neuer Datensatz angelegt wird. Anders sieht es aber beim Befehl „DEFAULT NULL“ aus, dass die entsprechende Spalte nicht zwingend mit Daten gefüllt werden muss und standardmäßig auf NULL gesetzt wird, wenn ein neuer Datensatz keine Daten enthält. Das Attribut „ID“ wird als Primärschlüssel (Befehl „PRIMARY KEY (‘ID‘)“ definiert und mit AUTO_INCREMENT zusätzlich ausgestattet. Das hat die folgende Bedeutung, dass das Attribut „ID“ fortlaufend um eins erhöht wird, wenn ein neuer Datensatz hinzugefügt wird. Ein Primärschlüssel dient der eindeutigen Identifizierung der einzelnen Zeilen in einer Tabelle.

2.3 OpenCV und Gesichtserkennung

OpenCV ist eine freie Programmbibliothek (unter BSD-Lizenz) mit Algorithmen für Bildverarbeitung und maschinelles Sehen. Es beinhaltet C/C++, Python und Java Interfaces und unterstützt Windows, Linux, Mac OS, iOS und Android. OpenCV wird sowohl im kommerziellen wie im privaten Bereich stark benutzt.[4]

Unter die vielen Möglichkeiten das OpenCV anbietet, ist für dieses Projekt die Funktionen der Gesichtserkennung relevant. Um Bewegungen und Gesichter zu erkennen, muss die Aufnahme der Kamera in drei Schritte ausgewertet werden.

2.3.1 Logarithmus trainieren

Unter Logarithmus trainieren wird gemeint, der Gesichtserkennungsalgorithmus wird bekannt gemacht anhand vorhandenen Bilder, welche Personen identifiziert werden. Dafür müssen Bilder von den jeweiligen Personen im System gespeichert sein. Um eine höhere Übertragungsrate (nach Erfahrungen 13-17 FPS) zu erreichen, werden die Bilder der Kamera nur in Grautöne aufgenommen. Deswegen müssen die Trainingsbilder im gleichen Format vorliegen. Mittels eines C-Programmes werden die Gesichter geschnitten und in ein neues Bild in Graustufen gespeichert.

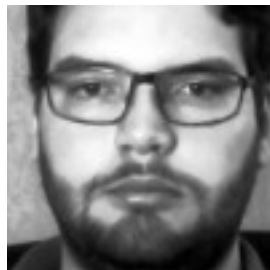


Abbildung 2.4: Trainingsbild

Mittels einer csv-Datei wird zu jeder Person eine Identifikationsnummer zugewiesen und dem Hauptprogramm bekannt gegeben, wo diese Bilder gespeichert sind. Zum Beispiel:

```
/home/pi/pictures/Person1.jpg;1  
/home/pi/pictures/Person2.jpg;2  
/home/pi/pictures/Person3.jpg;3
```

Leider ist uns nicht gelungen, ein stabiles System zu bauen, indem die Personenerkennung als sicheres Identifikationsmerkmal betrachtet werden kann. Es ist nötig, mehrere Merkmale auszuwerten und dafür stoßen wir an den Grenzen unseres Prozessors, obwohl dieser für die Versuche übertaktet wurde.

2.3.2 Kamera initialisieren

Videos sind die Wiedergabe von einer Bildersequenz, also werden einzelne Bilder bearbeitet und ausgewertet. Zuerst wird die angeschlossene Kamera angesprochen und die Videoaufnahme gestartet. Dies folgt dem gleichen Prinzip wie einer Datei öffnen und aus dieser Datei zeilenweise auslesen. Jede gelesene Zeile wird gespeichert und von einem andere Algorithmus bearbeitet. Bei

der Kamera werden einzelnen Bilder gespeichert und danach bearbeitet.

2.3.3 Bild auswerten

Wenn ein Bild gespeichert worden ist, kann dieser ausgewertet werden. Als Erstes wird versucht, mittels OpenCV, ein Gesicht in das jeweilige Bild zu erkennen. Ist dieser Schritt erfolgreich, wird um das erkannte Gesicht ein Viereck gezeichnet. Danach versucht das Algorithmus anhand der mit der csv-Datei ausgewertete Bilder festzustellen, ob diese Person bekannt ist. Falls ja, wird anhand der Identifikationsnummer ein Name (im Programmcode fest kodiert) hinzugefügt (Siehe Abbildungen 2.5 und 2.6).



Abbildung 2.5: Nicht erkannte Person



Abbildung 2.6: Erkannte Person mit festkodierten Nametag

Dieses bearbeitete Bild wird für die Ausgabe über den Streamer gespeichert. Der Streamer liest das bereits ausgewertete Bild nach der Gesichtserkennung und stellt es als Stream zur Verfügung.

2.4 MJPG Streamer

Der MJPG-Streamer bietet die Möglichkeit, Videodaten von einer Videoquelle als Motion-JPEG streamen zu lassen. Modernere Netzwerkkameras erzeugen Video-Streams automatisch, während mit Hilfe von diesem Programm, auch einfache Webcams an einem Rechner mit Internet-Zugang Streams erzeugen können. Der MJPG Streamer gehört nicht zu den offiziellen Paketen von Linux und muss manuell kompiliert werden. Dafür sind folgende Pakete nötig:

- build-essential
- libjpeg-dev
- imagemagick
- subversion

Der aktuelle Quellcode muss von der Webseite <http://sourceforge.net/projects/mjpg-streamer/> heruntergeladen werden und dementsprechend kompiliert werden.[5] Über einem Webbrowser hat man Zugriff auf dem Stream. Der Stream basiert auf HTTP und TCP. In diesem Projekt lautet

die Adresse für den Stream *spyhole.no-ip:1900* (der Default Port 8080 wurde auf 1900 geändert). Hier wird eine Oberfläche mit verschiedenen Stream-Optionen dargestellt. Unter anderem Java und Javascript, sowie die Möglichkeit das Stream mittels das Programm VLC zu sehen. Durch Personalisieren dieser Webseite haben unsere Desktop und Android App Zugriff auf dem Stream.

2.5 Socket

Damit die Clients (Android App und Desktop App) mit dem Server (Raspberry Pi) kommunizieren können, haben wir ein Socket ³ programmiert. Wir haben uns für ein TCP Socket entschieden, da wir die Verbindungsorientierung und Ausfallsicherheit von TCP benötigen, um eine gewisse Sicherheit des System gewährleisten zu können. Ein weiterer Vorteil ist, dass Kommandos als lokaler Benutzer ausgeführt werden können. Das heißt, man vermeidet viele Probleme mit der Rechtevergabe von ausführbaren Dateien und Programme.

Auf der Seite des Servers werden zwei Kommandos erwartet. Diese lauten “Tür öffnen” oder “Stream starten”. In dem Fall, dass der Stream gestartet werden soll, wird hier ein Shell-Script aufgerufen, das die Kamera und die Gesichtserkennung aktiviert. Bei der Kamera wurde ein Timeout von einer Minute eingestellt, für den Fall, dass der Benutzer sich nicht abmeldet. Somit wird vermieden, dass die Kamera unnötigerweise aktiv ist.

Will der Benutzer die Tür öffnen, werden zwei weitere Informationen benötigt. Der Client schickt dem Server folgenden Befehl: “2xxxxyyy”; wo “xxxx” für die ID des Eintrages in der Tabelle *tb_doorlogger* steht und “yyy” für die User ID, die die Tür öffnen möchte. Beide Werte werden in der Tabelle *tb_images* eingetragen, sowie der Dateiname des aufgenommenes Bildes (bestehend aus Datum und Uhrzeit).

Wenn die Nachrichten abgearbeitet worden sind, bekommt der Client ein “ACK” zurück und die Verbindung wird geschlossen. Sollten verfälschten Nachrichten ankommen, werden diese abgefangen und der Client bekommt als Antwort ein Fehlercode (durch die Benutzung von TCP erwarten wir selten solche Fälle).

³Software-Modul mit dessen sich ein Computerprogramm in einem Rechnernetz verbindet und mit anderen Computern Daten austauscht

Kapitel 3

Clients

Im Rahmen des Projektes haben wir uns für eine Desktopanwendung und eine Mobile Applikation für Smartphones entschieden. Die Desktopanwendung wurde mit *JavaFX* realisiert um die Anwendung auf möglichst viele Systeme zum Laufen zu bringen. Um den Programmieraufwand für die Entwicklung der Smartphone-Applikation gering zu halten, haben wir uns für Android entschieden. Da *JavaFX* und Android Java basierend sind, können viele Klassen auf beiden Systemen verwendet werden, dies hat die Entwicklungszeit deutlich verringert.

JavaFX

JavaFX ist eine Java-Spezifikation, die als Hauptkonkurrenten Adobe Flash und Microsoft Silverlight hat. Ein positiver Punkt ist die Lauffähigkeit auf diversen Geräten wie z.B. Mobilfunk, Desktop-Computern, Embedded Geräten und Blu-ray Geräten. Die Programmierung wird normal in Java programmiert. Die dazugehörigen Bibliotheken werden seit der Java SE Runtime 7 Update 6 automatisch mit installiert. Es ist unter anderem auf die Grafikprogrammierung ausgelegt. Dadurch lassen sich grafische Elemente schnell programmieren und mit CSS gestalten[6].

Android

Das Betriebssystem ist komplett in der Sprache Java programmiert worden. Dabei handelt es sich um ein Open-Source Betriebssystem, welches von der Firma Google entwickelt worden ist. Kern des Betriebssystems ist ein angepasster Linux-Kernel 2.6. Das Framework Android-SDK bietet es dem Nutzer an die grafische Oberfläche von der eigentlich Logik zu trennen. Die grafischen Oberflächen werden als Views betitelt. Die Views werden mithilfe von XML gestaltet.

3.1 Konzept

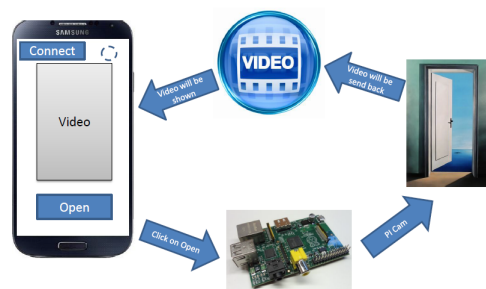


Abbildung 3.1: Prozess der App

3.1.1 Struktur und Aufbau der Applikationen

Sobald die Control-View geladen ist verbindet sich das Client mit dem Raspberry Pi. Nachdem die Verbindung aufgebaut wurde, kann der Benutzer die Schaltflächen `Stream starten`, `Tür öffnen`, `Bilder-Log betätigen`. Nach der erfolgreichen Verbindung zum Server sendet dieser einen kontinuierlichen Live-Stream an das Endgerät. Wenn der Benutzer nun die Schaltfläche `Tür öffnen` betätigt, wird ein Befehl über die Socket-Schnittstelle zum Öffnen der Tür gesendet und ein GPIO angesteuert, der den Türöffner betätigt, um die Tür zu öffnen. Auf das gleiche Prinzip baut auch `Stream starten` auf. `Bilder-Log` schick eine Anfrage an die Datenbank und erstellt eine Liste der Personen die die Tür geöffnet haben.

3.1.2 Anmelden des Users

Um zu verhindern, dass nicht jede Person auf dem Stream zugreifen kann, sollte aus Sicherheitsaspekten ein Login implementiert werden. Dieser besteht aus zwei Felder, ein Textfeld, und ein maskiertes Passwortfeld sowie ein Login-Button.

3.1.3 Registrierung

Abbildung 3.2: Konzept der Registrierung

Um sich einloggen zu können, muss sich der Benutzer beim allerersten Mal mit seinen Kontaktdaten registrieren. Dabei muss der Nutzer folgende Daten eingeben:

- Vorname
- Nachname
- E-Mail
- Benutzername
- Passwort mit Überprüfung

3.1.4 Stream starten und Tür öffnen

Es wird über ein Socket eine Verbindung aufgebaut und Daten an dem Server (Raspberry Pi) gesendet, wie in Kapitel 2.5 schon beschrieben wurde. Der Client erstellt dabei ein Socket-Objekt, mit die vereinbarte IP-Adresse und Port, und schickt die gewünschte Befehle.

3.1.5 Datenbankverbindung

Um Einträge der Datenbank darstellen, bearbeiten und erzeugen zu können, muss sich der Client mit der Datenbank des Servers verbinden. Nach der Voruntersuchung haben wir festgestellt, dass die Clients unterschiedlich vorgehen müssen. Anders als für die Desktopanwendung, gibt es für die Androidentwicklung keine Bibliothek die die direkte Verbindung zu einer MySQL Datenbank ermöglicht.

Kapitel 4

Technische Umsetzung

Im Folgenden wird auf die Vorgehensweise eingegangen, welche Technologien verwendet worden sind und wie der Login- und Registrierungsprozess ablaufen soll, sowie die Unterschiede zwischen der Desktopanwendung und der Android-Applikation.

4.1 Struktur und Aufbau der App

Es gibt insgesamt vier verschiedene View's in der App.

- Login
- Registrierung
- Control
- Bilderlog

Im Anhang A.3 und A.2 befinden sich Screenshots der Applikationen.

4.1.1 Login

Bei der Login-View kann sich der Nutzer mit seinem Benutzernamen und Passwort, was in der Datenbank hinterlegt ist, anmelden und somit zur Control-View gelangen. Ist einer der Felder nicht ausgefüllt oder das Passwort bzw. der User nicht korrekt, wird das mit einer Nachricht dargestellt und der Nutzer kann es erneut versuchen. Das Passwort-Feld ist immer maskiert. Von der Login-View ist es auch möglich, zur Registrierung-View zu wechseln.

4.1.2 Registrierung

Falls der Benutzer noch kein Account in der Datenbanktabelle `tb_users` hat, hat er die Möglichkeit, sich über die Registrierung-View zu registrieren. Software-seitig wurde eine Überprüfung eingebaut, dass jedes Textfeld etwas beinhalten muss (Pflichtfelder). Die Passwörter werden auf Konsistenz überprüft. Wenn alle Überprüfungen korrekt sind, wird aus den Eingaben ein String gebaut und an die Datenbank geschickt. Falls die Überprüfung fehlgeschlagen ist, wird wie bei der

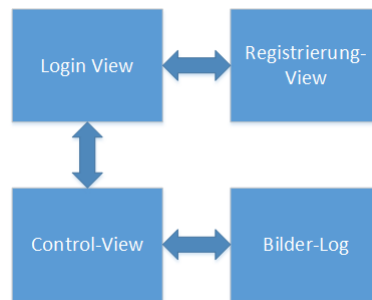


Abbildung 4.1: Wechseln zwischen Views

Login-View eine Nachricht ausgegeben. An dieser Stelle wird auch überprüft, ob der Username schon in der Datenbanktabelle `tb_users` existiert. Somit wird Doppeleinträge in der Datenbanktabelle verhindert.

4.1.3 Datenbankverbindung

Bei der Datenbankverbindung ist zu beachten, dass die Desktopanwendung sich direkt mit dem MySQL Server verbinden kann, die Android Applikation allerdings nicht, weil die entsprechenden MySQL Bibliotheken für Android zu diesem Zeitpunkt nicht existieren. Die Verbindung findet über ein JSON Objekt statt. Server-seitig existiert ein PHP-Skript welches das Objekt abfängt und die Verbindung zur Datenbank herstellt.

JSON

Es wird eine Liste `params` mit dem Typ `NameValuePair` erzeugt. Anschließend werden die Übergabeparameter in die Liste eingefügt mit der Methode `params.add()`, zusätzlich mit einem Tag `register_tag`. Im Nachhinein wird das JSON-Objekt zusammen gefügt und am Ende der Funktion das fertig erzeugte JSON-Objekt mit allen Inhalten zurück gegeben. Das erzeugte JSON-Objekt wird per POST-Methode an den Server gesendet. Auf dem Server wird in der `insert.php` anhand des Tags erkannt, dass sich ein neuer Nutzer registrieren möchte. Das PHP-Skript baut eine Verbindung zur MySQL-Datenbank auf und sendet die Daten als SQL-Syntax dorthin und trägt diese in die jeweiligen Spalten ein. Im Login-View ist der Vorgang gleich, nur wird auf dem Server diesmal die `login.php` angesprochen.

```

public JSONObject loginUser(String name, String password);
public JSONObject registerUser(String name, String password);
public boolean isUserLoggedIn(Context context);
public boolean logoutUser(Context context);

// Building Parameters

List<NameValuePair> params = new ArrayList<NameValuePair>();
params.add(new BasicNameValuePair("tag", register_tag));
params.add(new BasicNameValuePair("name", name));
params.add(new BasicNameValuePair("password", password));

```

```
// getting JSON Object
JSONParser jsonParser = new JSONParser(registerURL, params,mContext)
;
try {
    Thread.sleep(2000);
} catch (InterruptedException e) {
    e.printStackTrace();
}
JSONObject json = jsonParser.getJSONFromUrl();
Log.i("JSON4", json.toString());
// return json
return json;
```

Listing 4.1: User Functions

MySQL

Bei der Desktopanwendung existiert eine Klasse namens `DBConnector.java` die für die Verbindung zur Datenbank zuständig ist. Dank dieser Klasse und der direkten Verbindung kann die Anwendung SQL Befehle direkt an den Server schicken. Um ein Eintrag zu erstellen, bereitet die Applikation ein String vor, mit der SQL-Syntax und die Parameter die eingetragen werden sollen. Der folgende Code zeigt die Darstellung, wie ein String mit Nutzerdaten aussieht und an die Datenbank gesendet wird.

```
String SQL = "INSERT INTO tb_user VALUES (null, ' "
            + txtVorname.getText() + ", ' "
            + txtNachname.getText() + ", ' "
            + txtEmail.getText() + ", ' "
            + txtUserName.getText() + ", ' "
            + txtPw.getText() + ", NOW())";
```

Listing 4.2: String neuer Benutzer

```
INSERT INTO tb_user VALUES (null, 'Heinz', 'Müller', 'heinz.
mueller@gmail.com',
'H.Mueller', '007', NOW())
```

Listing 4.3: SQL Benutzer eintragen

Der Befehl `NOW()` wird in der Datenbank mit dem aktuellen Datum und Uhrzeit gesetzt.

4.1.4 Control

In der View `Control` ist es möglich, das Live-Video von der Kamera zu betrachten. Es wird mit Hilfe der Klasse `WebEngine` und `WebView` realisiert. D.h. es wird durch `WebEngine` die Webseite geladen und durch `WebView` wird die geladene Webseite in der View angezeigt. Diese Webseite wird vom *MJPEG Streamer* (siehe Kapitel 2.4) zur Verfügung gestellt. Der Stream wird erst gestartet, wenn der Button `Stream starten` getätigt ist.

```
WebView webview = new WebView();
webview.setVisible(true);
WebEngine webengine = webview.getEngine();
webengine.setJavaScriptEnabled(true);
File file = new File("http://<IP-Adresse_vom_PI>/javascript\_simple.
html");
```

```
webengine.load(file.toString());
```

Listing 4.4: Stream Einbindung

Beim Betätigen des Buttons *Benutzertabelle* wird eine neue View geöffnet (siehe Kapitel 4.1.5) und die View `Control` wird geschlossen. Nach dem Betätigen des Buttons *Tür öffnen* werden zwei Funktionen aufgerufen. Bei der ersten Funktion wird über die Socket-Schnittstelle ein Befehl an das Pi gesendet, das dieses die Tür öffnen soll. Bei der zweiten Funktion wird ein neuer Eintrag in die Tabelle `tb_doorloogers` in Datenbank eingetragen. Dieser Eintrag beinhaltet den Zeitpunkt des Öffnens der Tür, sowie der Benutzer. Der Server erstellt ein zweiten Eintrag, das bereits in Kapitel 2.5 erläutert wurde.

4.1.5 Bilder Log

Diese View hat als Hauptobjekt eine Tabelle (TableView). Der Tabelleninhalt wird dynamisch erstellt. In einer zusätzliche Klasse wird überprüft, wie viele Spalten die Datenbanktabelle hat und fügt diese dann dem Tabellen-Objekt hinzu. Nach dem Hinzufügen der Spalten wird Zeile für Zeile aus der Datenbank gelesen und in die Tabelle geladen. Zu jedem Eintrag in die Datenbanktabelle `tb_doorlogger` gehört ein Bild. Um sich zu einen entsprechenden Tabelleneintrag das Bild anzusehen, kann der Benutzer ein Bild aus der dargestellten Tabelle auswählen.

Kapitel 5

Aktueller Projektzustand

Im Rahmen dieses Projekts waren vielfältige Software- sowie Hardwareentwicklungen umzusetzen. Die drei große Teile des Projektes funktionieren einzeln stabil und gut. Zur Zeit arbeiten wir als Team daran, die Kompatibilität und Funktionalität von Clients und Server zu garantieren. Wir entwickeln gerade die ausführliche Kommunikation und am Datenaustausch zwischen die Schnittstellen und an der Sicherheit dessen. Um die gestellte Anforderung zu erfüllen müssen noch Kleinlichkeit überarbeitet werden und dem entsprechend die Funktionalität der Module ausführlich getestet werden.

Abbildungsverzeichnis

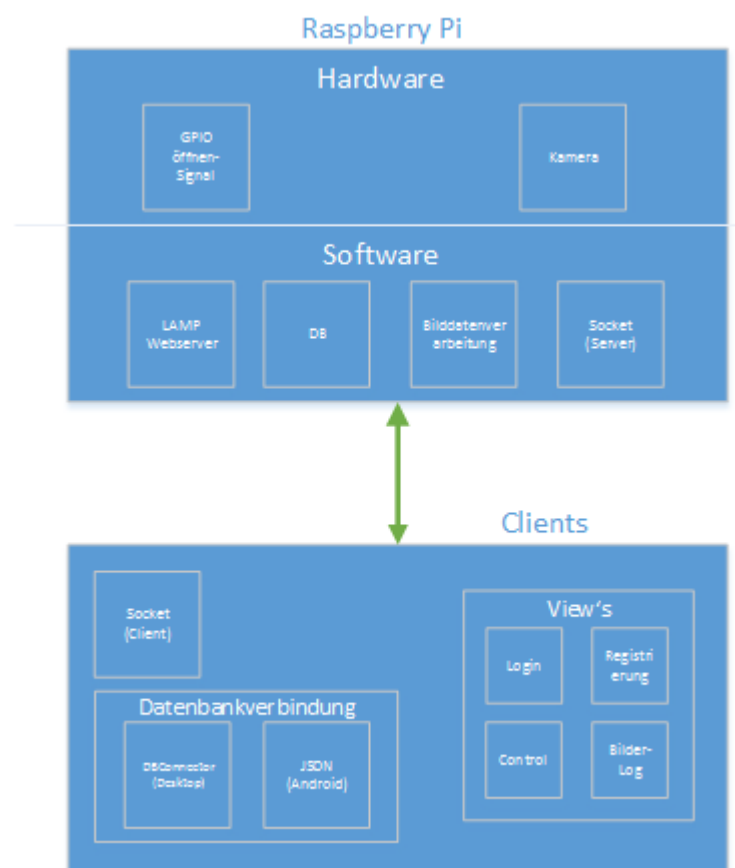
2.1	Raspberry Pi Model B	3
2.2	Raspberry Pi Model B Komponenten	4
2.3	Raspberry Kamera	5
2.4	Trainingsbild	10
2.5	Nicht erkannte Person	11
2.6	Erkannte Person mit festkodierten Nametag	11
3.1	Prozess der App	14
3.2	Konzept der Registrierung	14
4.1	Wechseln zwischen Views	17
A.1	Login View	24
A.2	Datenbank View	24
A.3	Control View ohne Stream	24
A.4	Control View	24
A.5	Table View	25
A.6	Login View	25
A.7	Registrierung View	25
A.8	Login View	26
A.9	Control View ohne Stream	26
A.10	Table View	26

Listings

2.1	Aufbau der Datenbanktabellen	8
4.1	User Functions	17
4.2	String neuer Benutzer	18
4.3	SQL Benutzer eintragen	18
4.4	Stream Einbindung	18

Anhang A

A.1 Projektüberblick

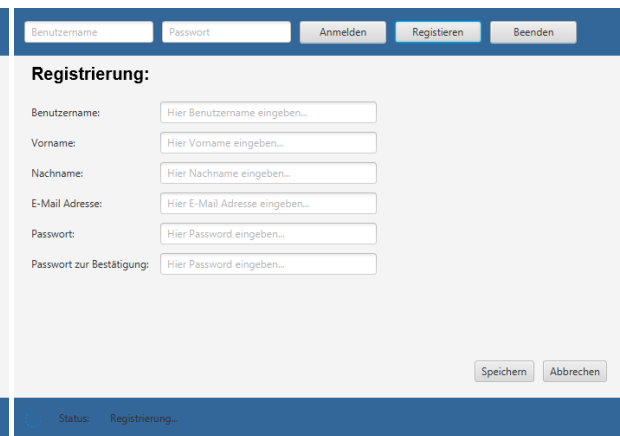


A.2 Desktopanwendung



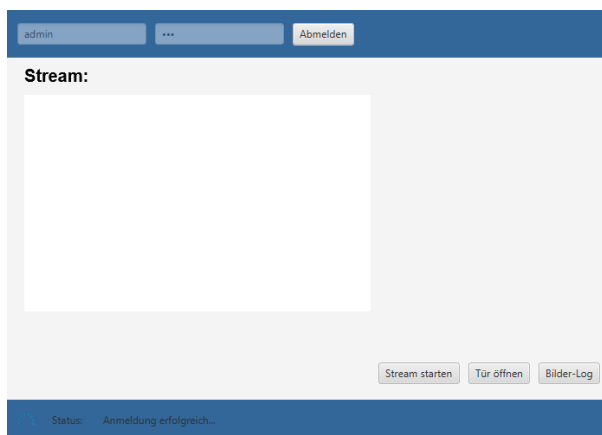
The Login View features a top navigation bar with a search input, a password input, and buttons for 'Anmelden', 'Registrieren', and 'Beenden'. The main content area is titled 'Bitte melden Sie sich an...' and is currently empty. The status bar at the bottom shows 'Status: Bereit...'.

Abbildung A.1: Login View



The Registration View has a top navigation bar with buttons for 'Anmelden', 'Registrieren' (highlighted), and 'Beenden'. The main content area is titled 'Registrierung:' and contains form fields for 'Benutzername', 'Vorname', 'Nachname', 'E-Mail Adresse', 'Passwort', and 'Passwort zur Bestätigung', each with a placeholder text. At the bottom right are 'Speichern' and 'Abbrechen' buttons. The status bar at the bottom shows 'Status: Registrierung...'.

Abbildung A.2: Datenbank View



The Control View without stream shows a top navigation bar with 'admin', a password input, and an 'Abmelden' button. The main content area is titled 'Stream:' and contains a large empty white box. At the bottom are buttons for 'Stream starten', 'Tür öffnen', and 'Bilder-Log'. The status bar at the bottom shows 'Status: Anmeldung erfolgreich...'.

Abbildung A.3: Control View ohne Stream



The Control View with stream shows a top navigation bar with 'admin', a password input, and an 'Abmelden' button. The main content area is titled 'Stream:' and contains a video stream of two people. A bounding box is drawn around one person, with the text 'id=marcus' displayed above it. At the bottom are buttons for 'Stream starten', 'Tür öffnen', and 'Bilder-Log'. The status bar at the bottom shows 'Status: Anmeldung erfolgreich...'.

Abbildung A.4: Control View

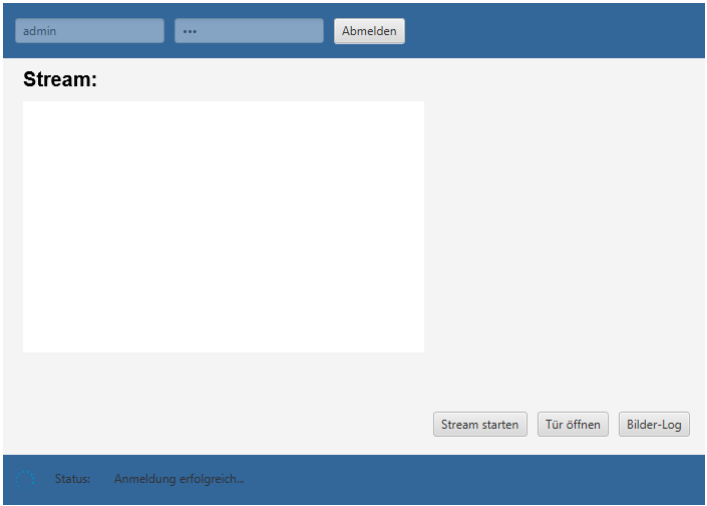


Abbildung A.5: Table View

A.3 Android app

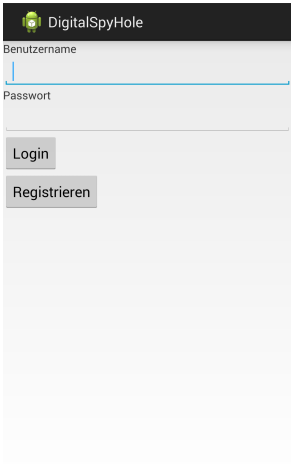


Abbildung A.6: Login View

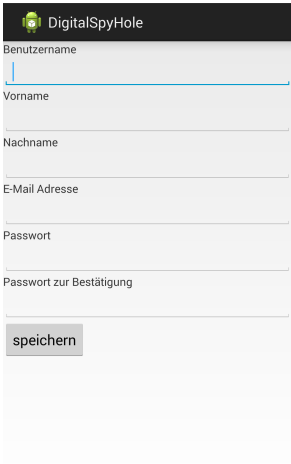


Abbildung A.7: Registrierung View

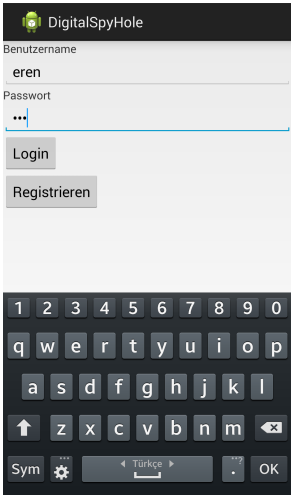


Abbildung A.8: Login View



Abbildung A.9: Control View ohne Stream

DigitalSpyHole		
ID	User	Date
1	niki	2014-02-07 15:38:45
2	niki	2014-02-07 15:40:39
3	niki	2014-02-07 15:41:10
4	niki	2014-02-07 15:41:16
5	niki	2014-02-07 15:41:47
6	niki	2014-02-07 15:42:10
7	niki	2014-02-07 15:42:20
8	niki	2014-02-07 15:42:33
9	niki	2014-02-07 15:42:37
10	niki	2014-02-07 15:42:41
11	niki	2014-02-07 15:42:44
12	niki	2014-02-07 15:42:48
13	niki	2014-02-07 15:42:51
14	niki	2014-02-07 15:42:54
15	niki	2014-02-07 15:42:58
16	niki	2014-02-07 15:43:02
17	niki	2014-02-07 15:43:05
18	niki	2014-02-07 15:44:07
19	niki	2014-02-07 15:44:16
20	niki	2014-02-07 16:04:59
21	niki	2014-02-07 16:05:06
22	niki	2014-02-07 16:05:25
23	niki	2014-02-07 16:07:43
24	niki	2014-02-07 16:07:56
25	niki	2014-02-07 16:30:45
26	niki	2014-02-07 16:30:51
27	niki	2014-02-07 16:47:14
28	niki	2014-02-07 16:47:19
29	niki	2014-02-07 17:32:15

Abbildung A.10: Table View

Literaturverzeichnis

- [1] *Raspberry Pi*, Raspberry Pi Foundation, <http://www.raspberrypi.org>, 15.01.2014.
 - [2] *Raspberry Pi Guide*, Raspberry Pi Foundation, <http://www.raspberrypiguide.de>, 15.01.2014.
 - [3] *Raspbian*, <http://www.raspbian.org>, 15.01.2014.
 - [4] *OpenCV*, <http://opencv.org>, 02.02.2014.
 - [5] *mjpg-streamer*, http://sourceforge.net/apps/mediawiki/mjpg-streamer/index.php?title=Main_Page, 02.02.2014.
 - [6] *JavaFX für das Raspberry Pi*, Oracle, https://blogs.oracle.com/java/entry/developer_preview_of_java_se, 2012.
-