Beuth Hochschule für Technik Berlin

Spyhole

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN

University of Applied Sciences

Seminararbeit (Zwischenbericht) für die Lehrveranstaltung
ESD bei Herr Rozek

Eingereicht von der Gruppe 14:

Matthias Hansert	s791744
Marcus Perkowski	s798936
Eren Büyükkircali	s805833

Inhaltsverzeichnis

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 gucken 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 Rasberry Pi verwendet werden, worauf in Kapitel 2 eingegangen wird. In Kapitel 3 wird die Android App beschrieben und auf dessen Funktionalitäten eingegangen. Auf Deskop App wird in Kapitel 4 genauer eingegangen.

Kapitel 2

Raspberry Pi

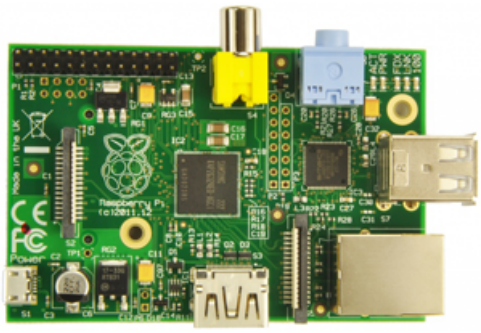

Abbildung 2.1: Raspberry Pi Model B

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.

[1] Stiftung und Wohltätigkeitsorganisation in Großbritannien

2.1 Hardware

2.1.1 Technische Spezifikationen

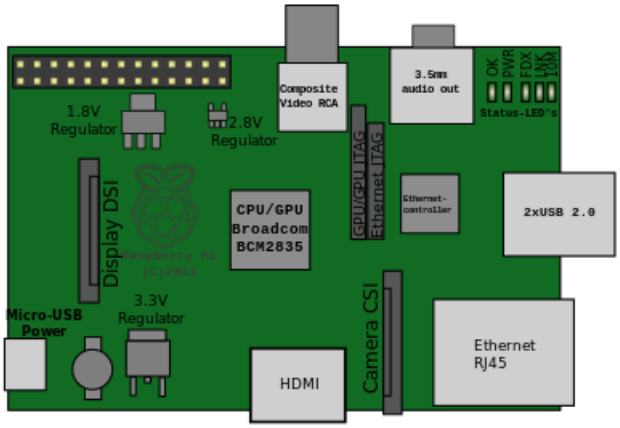

Abbildung 2.2: Raspberry Pi Model B Komponenten

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 und das Raspberry Pi auf eine mittlere Übertaktung einzustellen.

2.1.2 Kamera

Abbildung 2.3: Raspberry Kamera

Komponente	Kenndaten
Größe	20 x 24 mm
Sensor	OmniVision OV5647 (5MP)
Auflösung	2592 x 1944
Video	1080p 30fps
Anschluss	15-Pin Flachbandkabel

Die Kamera wurde speziell für den Raspberry Pi entwickelt, diese wird an die CSI-Schnittstelle (Camera Serial Interface) des Pi angeschlossen. Anschließend muss die Kamera-Unterstützung mittels des Konfigurationsprogramms `raspi-config` noch aktiviert werden.

2.2 Software und Einstellungen

2.2.1 Betriebssystem

Für das Raspberry Pi worden 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 entscheidend. Eins der beliebtestens Distrubutionen ist Raspbian, basierend auf Debian, was auch in diesem Projekt benutzt wird. Raspbian inkludiert alle Grundprogramme und -Utilities, dass mit vielen verschiedenen Packages kompatibel sind. Dieser 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 Androidsysteme, die auf das Raspberry Pi portiert worden sind. [3]

2.2.2 LAMP Webserver

LAMP steht für Linux Apache MySQL PHP 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 Videostreams sowie auf die MySQL Datenbank zugegriffen werden. Damit *http* oder *ssh* Anfragen an das 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 das 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 das Raspberry Pi arbeiten konnte wurde auf das System eine SSH-Verbindung eingestellt. So wurden auch Entwicklungskosten gespart, weil nicht jeder 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 ein 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* der Wert *#Port* auf den gewünschten Wert gesetzt werden. Danach muss das 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 ein privaten und einen öffentlichen Schlüssel erstellt. Der private Schlüssel wird am Client gespeichert und der öffentlicher Schlüssel am Server. Mit dem Befehl

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

werden die Schlüsseln generiert. Hier muss der Benutzer ein Passwort eingeben, mit dem sich er am Server anmelden wird. In der Regel heißen beider Schlüssel *id_das* und *id_das.pub*. Der private Schlüssel muss an das 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 auf den Server haben über den konfigurierten Port. Eine SSH-Verbindung wird mit folgendermaßen aufgebaut:

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

2.2.4 DNS

Um das Raspberry Pi über das lokale Netz erreichbar zu machen, wurde erstmal ein Webserver aufgebaut. Um jetzt das System überall über das Internet erreichbar ist, 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 im 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 runter gefahren wird. Daher ist es ratsam der 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 WYSIWYG[2] - 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 registierte Users gespeichert, und die Tabelle `tb_images` dient zur Speicherung der Bilder, die mithilfe der Kamera vom Rasberry 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,
```

[2]ist das Akronym für das Prinzip „**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et“ (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,
  `imgdata` longblob,
  `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 hinzufügt wird. Ein Primärschlüssel dient der eindeutigen Identifizierung der einzelnen Zeilen in einer Tabelle.

2.3 OpenCV und Gesichtserkennung

Ist eine freie Programmbibliothek(unter BSD-Lizens) mit Algorithmen für Bildverarbeitung und maschinelles sehen. Es beinhaltet C/C++, Python und Java Interfaces und unterstützt Windows, Linux, Mac OS, iOS and 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 auswertet werden.

2.3.1 Logarithmus traineren

Unter Logarithmus trainieren wird gemeint, den 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 zu erreichen werden die Bilder der Kamera nur in Schwarz-Weiß aufgenommen. Deswegen müssen die Trainingsbilder im gleichen Format vorliegen. Mittels eines C-Programmes werden die Gesichter geschnitten und in einen neuen Schwarz-Weiß Bild 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 Versuche übertaktet wurde.

2.3.2 Kamera Initializieren

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 eine Datei öffnen und aus dieser Datei Zeilenweise lesen. Jede gelesene Zeile wird gespeichert und von einem andere Algorithmus bearbeitet. Bei der

Kamera werden einzelne 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 im 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 bearbeitetes 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

MJPG-Streamer bittet die Möglichkeit Videodaten von einer Videoquelle als Motion-JPEG streamen zu lassen. Modernere Netzwerkkameras erzeugen Videostreams automatisch, während mit Hilfe von diesem Programm, auch einfache Webcams an einem Rechner mit Internet-Zugang Streams erzeugen können. 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

Das 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 das Stream. Im diesem Projekt lautet die Adresse für das Stream *spyhole.no-ip:1900* (der Default Port 8080 wurde auf 1900 geändert). Hier wird eine Oberfläche mit verschiedenen Streamoptionen 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 Mobile App zugriff auf das 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 Übertragungssicherheit von TCP benötigen, um eine gewisse Sicherheit des System gewährleisten zu können. Ein weitere 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 Serverseite werden zwei Kommandos erwartet. Diese lauten “Tür öffnen” oder “Stream starten”. Im Fall dass der Stream gestartet werden soll, wird hier ein Shell-Script aufgerufen, dass 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 folgendes Befehl: “2xxxxyyy”; wo “xxxx” für die ID des Eintrags 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 Name des aufgenommenes Bildes.

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

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

Kapitel 3

Android App

3.1 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.

3.1.1 Struktur und Aufbau der App

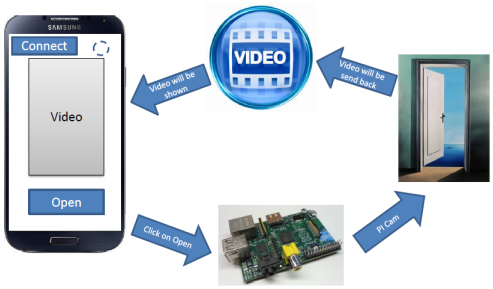

Abbildung 3.1: Prozess der App

Sobald die Control-View geladen ist verbindet sich das mobile Endgerät mit dem Raspberry Pi. Nachdem die Verbindung aufgebaut wurde, kann der Benutzer der App die Schaltfläche `Open` betätigen. Nach der erfolgreicher Verbindung zum Server sendet dieser einen kontinuierlichen Live-Stream an das Endgerät. Wenn der Benutzer nun die Schaltfläche `Open` 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.

3.1.2 Anmelden des Users

Um zu verhindern, dass nicht jede Person auf den Stream zugreifen kann, sollte aus Sicherheitsaspekten ein Login implementiert werden. Die Login funktioniert genau wie bei JavaFX App, es besteht aus zwei Felder, ein Textfeld, und ein maskierten Passwordfeld sowie ein Login Button. Jedoch wird im Folgenden auf die Vorgehensweise eingegangen, welche Technologien verwendet worden sind und wie der Login- und Registrierungsprozess ablaufen soll beschrieben.

Registrierung

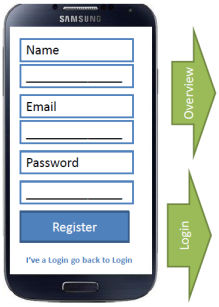

Abbildung 3.2: Registrierung

Um sich überhaupt einloggen zu können muss sich der Benutzer beim aller ersten Mal mit seinen Kontaktdaten registrieren. Dabei muss der Nutzer seine

- E-Mail
- Nutzername
- Passwort

eingeben. 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.

```
<Button
       android:id="@+id/btnLinkToLoginScreen"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:layout_marginTop="40dip"
       android:background="@null"
       android:text="Already␣registred.␣Log␣Me␣In!"
       android:textColor="#21dbd4"
       android:textStyle="bold" />
```

Listing 3.1: Android - Button erstellen

In dem obigen Codebeispiel wird ein Button in der View erzeugt. Dabei wird dieser mit einer sog. ID versehen, um den Button über diese ID aus dem Quellcode ansprechen zu können. Mithilfe von `android:layout_width` und `android:layout_height` wird die Höhe und Breite des Buttons beschrieben. Die Option `fill_parent` lässt den Button über die ganze Breite des Bildschirmes anzeigen, abhängig von der Auflösung des Endgerätes. Die Option `wrap_content` lässt den Button nur so groß werden, dass alle Inhalte gut zu erkennen sind.
Parallel zu der grafischen Gestaltung der Activity `Register.xml` wird die eigentliche Logik in Java-Klassen ausgelagert, um eine strikte Trennung von GUI und Fachlogik zu erreichen. Die Klasse `SignUp.java` ist diesem Fall die Klasse die sich auf das XML-File `Register.xml` referenziert. Um die einzelnen Objekte des Layouts ansprechen zu können, werden im ersten Schritt alle einzelnen Komponenten erzeugt und im Nachhinein mit der Funktion `Objekt.findViewByID.ObjektID` auf das jeweilige gerade erzeugte Objekt in der Java-Klasse referenziert.

```
Button btnRegister;
btnRegister = (Button) findViewById(R.id.btnRegister);
```

Listing 3.2: Objekt Erzeugung und Referenzierung

Um überhaupt eine funktionierende Activity zu programmieren braucht man die Funktion `onCreat()`. In dieser Funktion wird all das ausgeführt was beim Starten der Activity passieren soll. Zum einen das Referenzieren auf die erzeugten Objekte und weitere Funktionsaufrufe. Um den neuen Nutzer zu registrieren wurden einige Funktionen in die Klasse `UserFunctions.java` ausgegliedert. Diese Funktion beinhaltet verschiedene Funktionen, die anderen Klassen wieder verwendet werden um einen Nutzer zu registrieren, zu prüfen ob er schon eingeloggt ist oder um ihn auszuloggen.

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

Listing 3.3: User Functions

Die Funktion `registerUser(...)` bekommt zwei verschiedene String - Parameter um den neuen Nutzer in die Liste eintragen zu können. Zu einen den Name und zum anderen sein gewähltes Password.

```
// 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 3.4: registerUser(...)

Es wird eine Liste `params` mit dem Typ `NameValuePair` erzeugt. Anschließend werden die Übergabeparameter in die Liste eingefügt mit der Methode `params.add()`, plus 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 Index.php anhand des Tags erkannt, dass sich ein neuer Nutzer anmelden möchte dem entsprechend wird in einer Mehrfachauswahl (Switch-Case) ausgewertet und in einem weiteren PHP-Skript weiter bearbeitet. Schlussendlich werden die Daten in eine MySQL-Datenbank in die jeweiligen Spalten geschrieben. Das eingegebene Passwort wird beim Eintragen in die Datenbank verschlüsselt, dass mögliche Hacker die das Password nicht auslesen können.
Die weiteren Funktionen die in der Klasse enthalten sind, sprechen an für sich selbst und werden dem entsprechen nicht weiter erläutert.
In der `SignUp.java` Klasse wird auf die response des Servers gewartet und bei einem erfolgreichen Registrieren wird der Nutzer der App weitergeleitet auf die Activity `OverView`.

Login

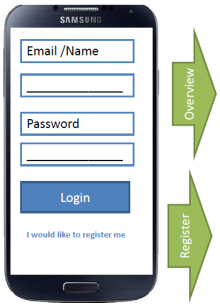

Abbildung 3.3: Login

Beim Login verläuft der Vorgang wie bei der eben beschriebenen Registrierung, bloß in die andere Richtung. Dabei wird ein JSON-Objekt vom Server an das mobile Endgerät geschickt und in der App aufgeschlüsselt und interpretiert. Danach wird verglichen ob sich der Nutzer, der sich gerade einloggen möchte schon eingetragen ist, wenn ja wird er auf die nächste Activity weitergeleitet, ansonsten wird er zur Registrierung geführt und gebeten sich anzumelden.

3.1.3 Control View

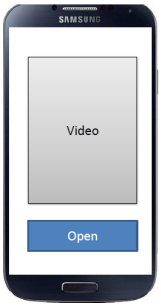

Abbildung 3.4: Control View

Die Activity `ControlView` ist die Activity in der der Nutzer den Stream und die Öffnung der Tür vornehmen kann. Die Activity beinhaltet zwei Komponenten, zum einen die WebView in der der Stream mit der Gesichtserkennung dargestellt wird und zum anderen der `Open Button`, mit dem das Signal über die Socket-Schnittstelle zum Türöffnen gesendet wird.

```
webView.getSettings().setJavaScriptEnabled(true);
//webView.setAlwaysDrawnWithCacheEnabled(true);
webView.setClickable(false);

// load and show the website
webView.loadUrl("http://spyhole.no-ip.biz/javascript_simple.html");
```

Listing 3.5: Aufbau der Verbindung und darstellen des Streams

Sobald der Nutzer sich auf der Activity Control View befindet und der Button Stream starten betätigt ist, beginnt die App sich mit dem Server zu verbinden und baut die Verbindung auf. Dabei war zu beachten das wir Javascript aktivieren, da wir den Stream mithilfe von Javascript auf der Webseite darstellen.

3.1.4 Datenbank

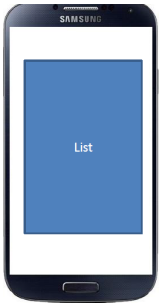

Abbildung 3.5: Datenbank

Diese Activity stellt alle Öffnungen der Tür da. Die dargestellten Daten sind die selben die auch in der JavaFX App benutzt werden, sie werden aus der MySQL Datenbank ausgelesen in der Tabelle dargestellt. Die Database Activity wird von `ListActivity` erweitert. `Database` beinhaltet zwei verschiedene Methoden zum einen die `getData()` und `fillList().getData()` stellt zuerst eine Verbindung zum MySQL- Server her und liest die Daten aus und speichert diese im Objekt `result` vom Typ ArrayList.
In der Methode `fillList()` wird dieses Objekt wieder verwendet und anhand der ausgelesenen Daten in die Liste in der App eingetragen.

Kapitel 4

Desktop App

Zu der Android App gibt es parallel eine App für den Desktop. Diese wurde mit JavaFX programmiert. In den folgenden Kapiteln wird die ungefähre Programmierung der einzelnen View's beschrieben.

4.1 Was ist 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. Ein sehr bekanntes Embedded Gerät, wofür es auch JavaFX gibt, ist das Raspberry Pi. [6]

4.2 Struktur und Aufbau der App

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

- Login (siehe 4.2.1)
- Registrierung (siehe 4.2.2)
- Control (siehe 4.2.3)
- Datenbank (siehe 4.2.4)
- Foto (siehe 4.2.5)

In JavaFX ist ein Fenster ein Stage-Objekt. In diesem Stage-Objekt sind fünf verschiedenen View's implementiert. Diesen View-Objekte können mehrere anderer Objekte hinzugefügt werden. Bei diesen anderen Objekten kann es sich um Buttons, eine Tabelle, ein Textfeld usw. handeln.

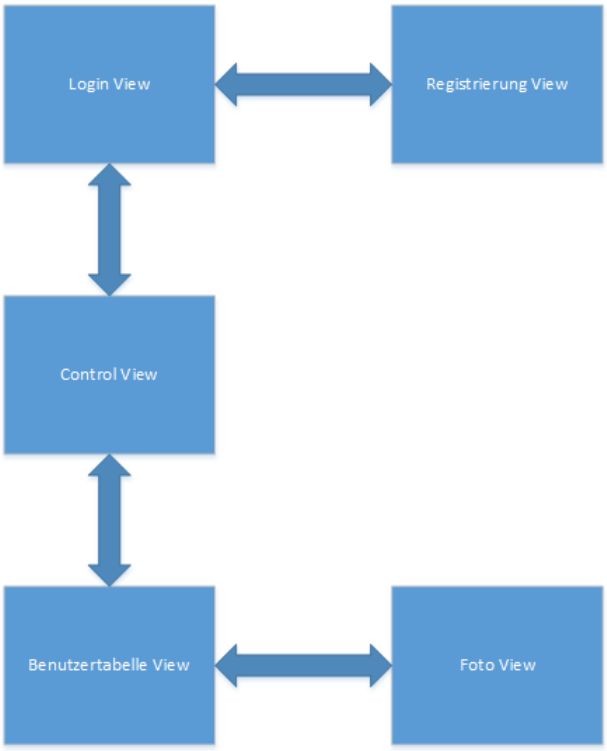

Abbildung 4.1: Aufbau der App

4.2.1 Login

Bei der Login View muss sich der Nutzer mit seinem Usernamen und Passwort, was in der Datenbank hinterlegt ist, anmelden. Ist einer der Felder nicht ausgefüllt oder das Passwort bzw. der User nicht korrekt, wird das mit einem `Label` als Message dargestellt.

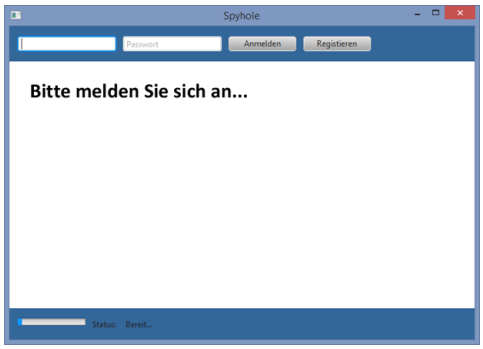

Abbildung 4.2: Login View

4.2.2 Registrierung

Falls der Benutzer noch kein Konto in der Datenbanktabelle `tb_users` hat, hat er die Möglichkeit, sich über die Registrierung View anzumelden. Softwareseitig wurde eine Überprüfung eingebaut, dass jedes Textfeld etwas beinhalten muss. Bei den Passwortfeldern wird überprüft, ob die beiden Passwörter, die eingegeben wurden, identisch sind. Wenn alle Überprüfungen korrekt

sind, wird aus den Eingaben und einem SQL Befehl `NOW()` ein SQL-String gebaut und an die Datenbank geschickt. Falls die Überprüfung fehlgeschlagen ist, wird wie bei der Login View (s. 4.2.1) eine `Label` Message ausgegeben.

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

Listing 4.1: Java-SQL neuer Benutzer

Der folgende String zeigt die Darstellung, wie der obige String mit Nutzerdaten aussieht und an die Datenbank gesendet wird.

```
INSERT INTO tb_user VALUES (null,'Gernot', 'Hassknecht', '
   heutShow@zdf.de',
     'Hassi', '007', NOW())
```

Listing 4.2: SQL Beispiel String

Der Befehl `NOW()` wird in der Datenbank mit dem aktuellen Datum und Uhrzeit ersetzt. Die Sichtbarkeit der Anmeldedaten ist nur direkt innerhalb der Applikation möglich. In diesem Fall mit einem System.out.println() in Eclipse. Was nicht mehr weiter implementiert wurde, ist eine zusätzliche Freischaltung des neuen Users von einem Admin. Nach direkter Registrierung kann sich der User sofort anmelden.

Abbildung 4.3: Registrierung View

4.2.3 Control

In der View `Control` ist es möglich, das Live-Video von der Cam 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 View angezeigt. Diese Webseite wird vom MJPEG Streamer 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 4.3: Stream Einbindung

Beim Betätigen des Buttons `Benutzertabelle` wird eine neue View `Datenbank` geöffnet (siehe Kapital 4.2.4) 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 kann. 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.

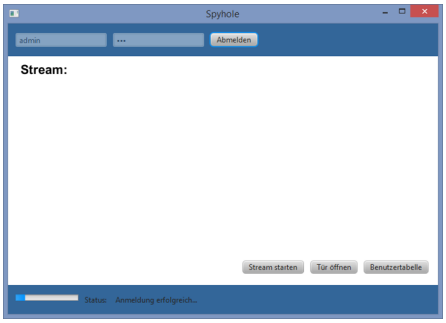

Abbildung 4.4: Control View

4.2.4 Datenbank

Diese View hat als Hauptobjekt eine Tabelle (TableView). Der Tabelleninhalt wird dynamisch erstellt. In einer zusätzliche Klasse wird überprüft, wie viele Spalten die Datenbank hat und fügt diese dann dem Tabellen-Objekt hinzu. Nach dem Hinzufügen der Spalten wird Zeile für Zeile aus der Datenbank geholt 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, muss man über eine Combobox die ID der Zeile auswählen und auf den Button `Bild laden` klicken. Im Moment ist die Implementierung der TableView noch nicht fertig, daher funktioniert die TableView mit Darstellung der Bilder nicht richtig.

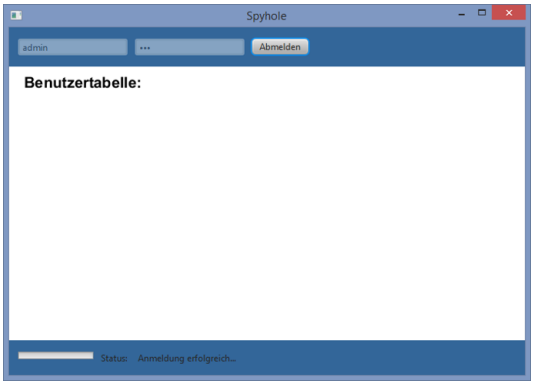

Abbildung 4.5: Datenbank View

4.2.5 Foto

Nachdem der `Bild laden` Button in der Datenbank Ansicht gedrückt wurde, wird mit Hilfe der angegebenen ID aus der Combobox der SQL-String gebaut.

```
String SQL = "SELECT * FROM tb_images WHERE ID = " + userID;
```

Listing 4.4: Java-SQL String Foto öffnen

Aus dem String ist relativ leicht zu erkennen, dass das Bild aus der Datenbanktabelle `tb_images` kommt. Das Bild ist aber in der Datenbank nur binär als BLOB Typ abgelegt. Dieser Typ existiert auch in Java. Nach dem Auslesen des Binärstreams wandeln wir das Gelesene in ein Byte-Array. Aus dem Byte-Array erzeugen wir dann ein Buffered-Image. In Swing könnten wir uns jetzt schon ein Bild anzeigen lassen. Aber die App wurde ja nicht mit Swing, sondern mit JavaFX geschrieben. Dank eines `.toFXImage(BufferedImage arg0, WritableImage arg1)` Befehles lässt sich unser Swing-Objekt einfach in ein JavaFX-Objekt umwandeln. Da im Moment die Benutzertabelle View noch nicht fertig implementiert ist, zeigen wir dieses Objekt vorrübergehend in einem zusätzlichen Stage Fenster an. In der Zukunft sollen die Bilder nicht in einem zusätzlichen Fenster dargestellt werden, sondern in einer zusätzlichen View-Objekt in Benutzertabelle View.

```
//Foto aus DB holen
byte[] imgData = imgShow.getImageDB(userID);
...
//Foto nach JavaFX Objekt wandeln
SwingFXUtils.toFXImage(bufImg, img2);

//Foto imageView hinzufügen
imageView.setImage(img2);
```

Listing 4.5: JavaFX Foto öffnen

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

Listings

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] *Raspian*, 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.