Tally
digitale Strichliste am Raspberry Pi

Nicolai Tegtmeier
Dominik Scheffler
Philipp Frieling
Sebastian Reinke

23.06.2015

Abbildung 1: Tally Logo

Inhaltsverzeichnis

1 Projektvorfeld

1.1 Kaffeestrichliste bisher

In kleineren Unternehmen und Arbeitsgruppen gibt es oft einen zentralen Kaffee/ Getränke -automaten und Snacks, an denen sich jeder Mitarbeiter bedienen kann. Um die Getränke und Snacks kaufen zu können wird meistens eine Strichliste auf Papier, meist in der Nähe des Automaten oder der Snacks, angebracht damit sich jeder Mitarbeiter eintragen kann, was er gekauft hat. Diese Liste wird oft zur besseren Verwaltung in eine Datenbank oder Tabelle eingepflegt, damit der Administrator einsehen kann wer wie viel gekauft hat. Hier passiert es jedoch h'aufig das die Strichliste sehr unleserlich wird und es bei der Übertragung in die Datenbank auf den PC zu Fehlern kommt. Au/sserdem ist diese Methode für den Verantwortlichen sehr zeitaufwendig da alles per Hand 'ubertragen werden muss.

Um diesem Problem entgegen zu wirken erarbeiten wir eine neue Methode um diese Daten 'digital' und einfacher speichern zu können. [AHU61]

2 Rahmenbedingungen

2.1 Aufgabe - die neue Kaffeestrichliste

Um eine möglichst energiesparende und handliche Lösung zu finden, sollte die neue Strichliste auf einem Raspberry Pi 2 realisiert werden. Zunächst soll eine passende Benutzeroberfläche für den Raspberry entwickelt werden, damit der Benutzer am Raspberry selber seine Einkäufe verbuchen kann. Dies soll mithilfe der integrierten Entwicklungsumgebung 'Qt Creator', die besonders zur Entwicklung von plattformunabhängigen C++ Programmen gedacht ist, entwickelt werden.

Mithilfe eines Webservers, der ebenfalls auf dem Raspberry läuft, soll die Administration von jedem Computer im selben Netzwerk über eine Weboberfläche möglich sein. Hier sollen auch die Benutzer ihre Daten einsehen und ändern können. Dazu wird eine MySQL/ SQLite Datenbank, sowie PHP Unterstützung benötigt. Mittels eines Barcodescanners soll es möglich sein am Raspberry Produkte ein zu scannen.

3 Ziele des Projekts

Die neue 'digitale Kaffeestrichliste' wird auf Basis eines Raspberry Pi 2 entwickelt, um die alte Strichliste abzulösen. Dazu wurden folgende zu erreichende Ziele festgelegt:

3.1 Zielbestimmung

3.1.1 Der Benutzer Account

Über den Benutzer Account soll der Benutzer sich am Raspberry selbst und an der Weboberfläche anmelden k'onnen. Am Raspberry selbst können Getränke und Snacks gekauft werden. Mithilfe der Weboberfläche kann der Benutzer Statistiken einsehen und sein Passwort ändern.

3.1.2 Das Programm

Das Programm das auf dem Raspberry läuft aktualisiert die Anzahl der Produkte im Lagerbestand automatisch, gibt Meldungen bei zu geringem Warebestand aus und gibt generelle Fehlermeldungen aus.

3.1.3 Der Administrator Account

Die Administration mithilfe des Administrator Accounts findet ausschliesslich über die Weboberfläche statt. Hier kann der Administrator Accounts hinzufügen und verwalten, Waren hinzufügen und verwalten und den Lagerbestand verwalten.

3.2 Produkteinsatz

3.2.1 Anwendungsbereiche

Mitarbeiter, beziehungsweise die Administratoren, können eine Strichliste 'digital' anlegen. Diese fasst den zu bezahlenden Betrag für die Mitarbeiter zusammen.

3.2.2 Zielgruppen

Personengruppen und Unternehmen bei denen Getränke und Snacks privat für alle angeboten und privat bezahlt werden, jedoch Schwierigkeiten mit der Handhabung einer herkömmlichen Strichliste haben und den Bestand an Getränken und Snacks erfassen wollen.

3.2.3 Betriebsbedingungen

Die Strichliste soll mö glichst Wartungsarm und einen niedrigen Stromverbrauch haben. Desweiteren soll sie täglich vierundzwanzig Stunden laufen.

3.3 Produktumgebung

Das Produkt kann unabhängig an jedem Ort betrieben werden, solange eine Stromquelle in der nähe ist.

3.3.1 Software

Die einzige Software die vom Benutzer benötigt wird ist ein aktueller Webbrowser.

3.3.2 Hardware

Der Benutzer kann mit jedem Internetfähigen Gerät die Weboberfläche erreichen.

3.3.3 Orgware

Der Administrator kann die Betriebsparameter konfigurieren.

3.4 Produktfunktion

3.4.1 Benutzerfunktion

Ein im System registrierter Benutzer kann das System erst nutzen, wenn er angemeldet ist. Nur ein Administrator kann einen Benutzer anlegen und ihm einen Benutzernamen sowie Passwort zuwesien. Sobald ein Benutzer registriert ist kann er sich sowohl am Raspberry als auch an der Weboberfläche anmelden. Dafür benötigt er seinen Benutzernamen und sein Passwort. Abmelden ist bei beiden Oberflächen jederzeit möglich.

Der Administrator kann über die Weboberfläche Produkte hinzufügen,entfernen und deren Details ändern. Benutzer können über die Weboberfläche Favoriten einrichten und Statistiken einsehen.

Am Raspberry kann der registrierte und angemeldete Benutzer ein Produkt aus einer Liste wählen oder mithilfe eines Barcodescanners ein Prdoukt einscannen, welches im Warenkorb erscheint. Desweiteren kann er die Anzahl der Produkte erhöhen und seinen gesamten Warenkorb einsehen. Wenn er alle Waren im Warenkorb hat, kann er zur Kasse und die Waren durch einen Klick bezahlen, wodurch sein Konto belastet wird. Ein Abbruch oder das erreichen des Hauptmenüs ist jederzeit möglich.

3.5 Programmfunktion

Beim Kauf eines Produktes aktualisiert das Programm automatisch den Warenbestand. Bei geringem Warenbestand wird eine Warnmeldung ausgegeben und bei fehlen des Artikels wird dieser entfernt.

3.6 Serverfunktion

Auf dem Raspberry soll ein Apache 2 Server mit PHP Unterstützung laufen.

3.6.1 Datenbank

Als Datenbank soll die resourcenschonendere SQLite Datenbank verwendet werden

3.7 Benutzerschnittstelle

Die Bedienung des Raspberry erfolgt über einen eingebauten Touchscreen am Raspberry. Auß()erdem wird ein Barcodescanner installiert um Produkte einscannen zu können.

Zentrierter Text hingegen ist schon eher nützlich.

4 Meilensteine des Projektes

4.1 Raspberry Pi 2 Model B - Die Hardware mit der passenden Software

Mit dem Raspberry Pi 2 war schon im Projektvorfeld eine passende Basis für das Projekt gegeben. Der Raspberry Pi 2 besitzt einen Vierkern ARM Cortex-A7 Prozessor mit jeweils 900MHz und einen Gigabyte ARbeitsspeicher. Desweiteren befinden sich vier USB 2.0 Ports, eine 40 GPIO pin Steckerleiste, ein HDMI Anschluss, ein Ethernet Port und ein Micro SD Kartenslot am Raspberry.

Ausser dem Raspberry Pi 2 war zunächst ein 3,5 Zoll Touchscreen der Firma admatec [1] mit dem Namen C-Berry Touch vorhanden, das mittels eines Adapterboards an der GPIO Stiftleiste des Raspberry angesteckt wird. Das Display besitzt eine Auflösung von 320x240 Pixeln, eine LED- Hintergrundbeleuchtung und wird komplett vom Raspberry mit Strom versorgt.

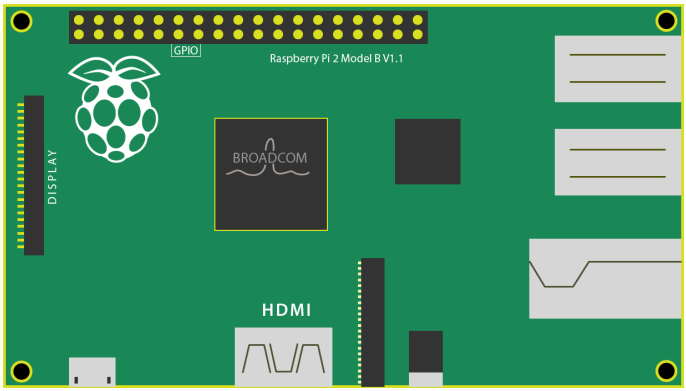

Abbildung 2: Der Raspberry Pi 2

4.1.1 Betriebssystem und Verbindung zu anderen Rechnern

Als erster Schritt wurde nach einem passenden Betriebssystem gesucht. Von Anfang an stand fest das ein Linuxderivat auf dem Raspberry laufen sollte. Hierbei fiel die Wahl auf das speziell f'ur den Raspberry optimierte Debianderivat 'Raspbian'.

'Raspbian' wurde so optimiert das es mit der geringen Hardware zurecht kommt und den ARM-Prozessor des Raspberrys unterst'utzt. Als Alternative stand eine Ubuntu Version ‚speziell für ARM-Prozessoren entwickelt, zu Verfügung die jedoch noch nicht voll ausgereift und funktionst'uchtig ist.

Um die weitere Einrichtung des Raspberrys zu vereinfachen wurde, neben der vorkonfigurierten und installierten SSH Verbindung zur Verwaltung über ein Konsolenprogramm (Windows: Putty), eine freie Implementierung des Remote Desktop Protocols für Linux Names 'xrdp' installiert. .

```
1 sudo apt-get install xrdp
```

Damit ist eine einfache Remote Desktopverbindung zum Raspberry möglich und die komplette Verwaltung kann 'uber einen Windows, Mac oder Linux Computer erfolgen.

4.1.2 Der Webserver

Als nächster Schritt wurde ein vollständiger Webserver auf dem Raspberry eingerichtet. Hierzu wurde zunächst ein Apache Server der Version 2 installiert .

```
1 sudo apt-get install apache2
```

der über eine hohe Stabilität und Geschwindigkeit verfügt und serverseitig die Skriptsprache PHP unterstützt. Im Anschluss wurde das PHP5 Paket installiert .

```
sudo apt-get install php5
```

um eine volle Unterstützung für die kommende Websites zu gewährleisten.

PHP ist eine Open Source-Skriptsprache die speziell f'ur die Webprogrammierung geeignet ist. Wenn eine Anfrage an die Website gestellt wird, wird der PHP-Code auf dem Server ausgef'uhrt und erzeugt eine HTML-Ausgabe die dann letztendlich an den Client gesendet wird.

4.1.3 Die passende Datenbank

Nun stand die Auswahl des passenden Datenbankverwaltungssystems an. Zur Auswahl standen die Systeme MySQL und SQLite. Bei Recherchen zu den beiden Datenbanken stellte sich schnell heraus das die MySQL Datenbank zwar auf dem Raspberry lauffähig ist aber keine hohe Stabilität besitzt und sehr resourcenlastig auf dem Raspberry läuft. Daher wurde das SQLite Datenbanksystem ausgewählt. .

```
apt-get install sqlite3
apt-get install php5-sqlite
```

SQLite ist eine relationale Datenbank und benötigt im Gegensatz zu MySQL keine ständig laufende Software da die Datenbank aus einer einzigen, wenigen Kilobyte grossen, Datei besteht. SQLite legt Wert auf Geschwindigkeit und Einfachheit.

4.1.4 Der Touchscreen

Als erstes wurde der Treiber für den bereitgestellte C-Berry Touchscreen installiert .

```
wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.36.tar.gz
tar zxvf bcm2835-1.36.tar.gz
cd bcm2835-1.36
./configure
make
sudo make check
sudo make install
```

und im Anschluss die Beispielsoftware zum anzeigen von Bildern installiert .

```
wget http://admatec.de/sites/default/files/downloads/C-Berry.tar
    .gz
tar zxvf C-Berry.tar.gz
cd C-Berry/SW/tft_test
make
sudo ./tft_test
```

Nach ersten Tests mit dem C-Berry Touchscreen fiel auf das es nicht als Primäres Display geeignet ist, da vom Desktop ein Screenshot erstellt wird und dieser auf dem Display ausgegeben wird. Starke Verzögerungen im Bildaufbau und eine schlechte Bedienung mithilfe des Touchscreens waren die Folge. Das Display ist mehr daf'uhr gedacht andere Inhalte als den Desktop wiederzugeben, die dann per Touchscreen bedient werden wie zum Beispiel einzelne Buttons.

Damit war der Touchscreen ungeeignet f'ur das Projekt und ein neuer Touchscreen mit besserer Anbindung an den Rapsberry und einer höheren Bildwiederholungsrate musste gesucht werden. Dabei kam die Idee auf, ein 7 Zoll resistives Touchscreen von Pollin zu verwenden das durch seine Grösse viel Platz für das Tally Programm bietet und eine sehr gute Touch Bedienung verspricht. Von Vorteil war auch die eigene externe Grafikeinheit die eine hohe Bildwiederholungsrateermöglicht und ein externer USB- Touchcontroller der die Bedienung des Touchscreens besser ausf'uhren soll.

Jedoch stellte die externe Stromversorgung ein Problem dar, da man jeweils ein Stromkabel für den Raspberry und eins für das Display benötigt, damit wäre eine Energiesparende Lösung ebenfalls nicht möglich.

Deshalb fiel die Entscheidung zugunsten des 3,5 Zoll Touchscreen 4DPi-35 von 4D System aus. Der Touchscreen besitzt eine Auflösung von 480x320 Pixeln und durch die High Speed 48Mhz SPI Verbindung werden hohe und konstante Bildwiederholungsraten ermöglicht. Die Vorteile bei diesem Display waren die kleine Bauform und keine zusätzlich benötigte Stromquelle. Ausserdem wird vom Hersteller direkt ein passender Kernel für Raspbian zur Verfügung gestellt. .

```
1 wget http://www.4dsystems.com.au /downloads/4DPi/kernel4dpi_1
      .3-3_pi2.deb
2
3 sudo dpkg -i kernel4dpi_1.3-3_pi2.deb
```

Nachdem das erforderliche Paket installiert wurde, musste der Raspberry herunter gefahren werden, das Display an den GPIO-Ports angebracht werden und der Raspberry wieder hoch gefahren werden. Danach war ein sofortiger Betrieb möglich da der Raspberry das Display als prim'ares Display erkennt und direkt auf den Desktop bootet. Dadurch ist kein HDMI Bildschirm mehr notwendig und die Bedienung kann ausschliesslich über den Touchscreen erfolgen.

4.1.5 Produkte scannen

Zur vereinfachten Eingabe von Produkten soll ausserdem ein Barcodescanner angebracht werden. Hier sollte entweder ein handeslüblicher Barcodescanner per USB an den Raspberry angeschlossen werden, der Barcode mittels USB

Webcam und passender Software oder aber ein Barcodescannermodul an den GPIO's des Raspberry als Lösung dienen.

Da jedoch das ein Barcodescannermodul nur mit ca. drei Volt betrieben werden kann, der Raspberry aber ca. fünf Volt an den GPIO's zur Verfügung stellt, wurde überlegt eine Webcam an den Raspberry anzuschliessen und mittels einer Software das Bild auf Barcodes zu untersuchen. Die Wahl fiel allerdings auf das Scannen mittels USB Barcodescanner, da dies eine einfacherere Handhabung mit verspricht und der Barcodescanner direkt als USB Tastatur erkannt wird.

4.1.6 Wlan für den Raspberry

Damit am Raspberry nicht immer ein Patchkabel angeschlossen werden muss, wurde der Edimax WLAN USB Stick am Raspberry angeschlossen. Der Edimax WLAN Stick wird automatisch von Raspbian erkannt da schon alle erforderlichen Pakete und Treiber installiert sind, weshalb dieser WLAN Stick empfohlen wird falls man WLAN am Raspberry benutzen möchte. Zunächst wurde in der automatische 'Power Saving' Modus abgeschaltet. Hierzu wurde eine Konfigurationsdatei für den Treiber des WLAN Sticks erstellt. .

```
sudo nano /etc/modprobe.d/8192cu.conf
```

mit folgendem Inhalt .

```
options 8192cu rtw_power_mgnt=0 rtw_enusbss=0
```

Um nun eine Verbindung mit dem Wlan herzustellen wurde die folgende Datei .

```
sudo nano /etc/network/interfaces
```

um folgenden Inhalt ergänzt .

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp

auto wlan0
allow-hotplug wlan0
iface wlan0 inet dhcp
wpa-ap-scan 1
wpa-scan-ssid 1
wpa-ssid "WLAN-NAME"
wpa-psk "WLAN-PASSWORT"
```

damit eine Verbindung mit dem WLAN Netz hergestellt werden kann. Anschliessend muss nur noch per .

```
sudo service networking restart
```

der Netzwerkdienst neu gestartet werden und eine Verbindung mit dem angegebenen WLAN Netz wird hergestellt.

Da diese M´oglichkeit jedoch relativ umst´andlich ist und Linux Kenntnisse erfordert, wird in das Tally Programm eine einfache WLAN konfiguration eingebaut in der man den WLAN-Namen un das WLAN-Passwort eingeben kann um sich mit dem WLAN-Netz zu verbinden.

4.1.7 angepasster Bootscreen

Während eines Meetings kam die Idee auf den Bootscreen zu verändern und ein eigenes Bild w´ahrend des hochfahrens anzuzeigen. Dazu wird eine weitere Software benötigt, der 'Frame Buffer Imageviewer', der ein Bild in den Speicherbuffer l´adt das w´ahrend des hochfahrens angezeigt werden kann. .

```
sudo apt-get install fbi
```

Danach muss ein passendes Skript geschrieben werden welches das Bild beim booten anzeigt. .

```
sudo nano asplashscreen
```

.

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          asplashscreen
# Required-Start:
# Required-Stop:
# Should-Start:
# Default-Start:     S
# Default-Stop:
# Short-Description: Show custom splashscreen
# Description:       Show custom splashscreen
### END INIT INFO

do_start () {

    /usr/bin/fbi -T 1 -noverbose -a /etc/splash.png
    exit 0
}

case "$1" in
  start|"")
    do_start
    ;;
  restart|reload|force-reload)
    echo "Error: argument '$1' not supported" >&2
    exit 3
    ;;
  stop)
    # No-op
    ;;
  status)
    exit 0
    ;;
```

```
33    *)
34      echo "Usage: asplashscreen [start|stop]" >&2
35      exit 3
36      ;;
37  esac
38
39  :
```

Dieses Skript wird nun in den passenden Ordner verschoben damit es beim Start ausgeführt werden kann .

```
1 sudo mv asplashscreen /etc/init.d/asplashscreen
```

Als letztes wird das Skript für den Raspberry ausführbar gemacht und als Bootscript registriert .

```
1 sudo chmod a+x /etc/init.d/asplashscreen
2 sudo insserv /etc/init.d/asplashscreen
```

Nun wird w'ahrend des Bootvorgangs das Tally Logo angezeigt.

4.1.8 Backups einrichten und versenden per Mail

Als mögliche Backupvarianten standen ein komplettes Backup der Speicherkarte des Raspberrys oder ein Backup der Datenbank zur Wahl.

Der Entschluss fiel auf ein Backup der Datenbank, da diese auch per Mail an den jeweiligen Administrator geschickt werden soll. Dazu wird Rsync verwendet, das mit folgendem Befehl installiert wird: .

```
1 sudo apt-get install rsync
```

Rsync ist ein Programm das Dateien oder Ordner in verschiedenen Pfaden speichern kann. Dazu überprüft es zunächst ob die zu kopierende Datei geänder wurde oder nicht, wodurch ein unnützes kopieren von Daten unterbunden wird.

Damit die backups autmatisch und täglich erstellt werden, wird ein cronjob hinzugefügt. Cronjobs, oder auch Cron-Daemon, ist ein Dienst um Skripte oder Programme zu einer bestimmten Zeit starten zu k'onnen. .

```
1 sudo crontab -e
```

Hier wird folgende Zeile eingefügt .

```
1 rsync -a QUELLE ZIEL
```

Dadurch wird das Programm jeden Tag um .... aufgerufen und die Datenabank von ... nach ... kopiert.

Diese Datei wird nun auch täglich, nach dem erstellen des Backups, an den Admin per mail gesendet. Dafür wird das Programm sendEmail und zwei

Perl- Libraries installiert. SendEmail hat den Vorteil das kein Mailserver auf dem Raspberry installiert werden muss und der Raspberry als ein eigener 'sende Client' erkannt wird. Die beiden Perl- Libraries dienen dazu damit eine verschlüsselte Verbindung per TLS zum Mailserver aufgebaut werden kann.

Zum autmatischen versenden der Mails wird folgendes Skript erstellt .

```
sudo nano /usr/local/bin/mailnotify.sh
```

Das Skript sieht wie folgt aus .

```
################################################################################

               Sending E-Mail notification with sendEmail

 Creation:    15.08.2013
 Last Update: 20.10.2013

 Copyright (c) 2013 by Georg Kainzbauer <http://www.gtkdb.de>

 This program is free software; you can redistribute it and/or
     modify
 it under the terms of the GNU General Public License as
     published by
 the Free Software Foundation; either version 2 of the License,
     or
 (at your option) any later version.

################################################################################
#!/bin/bash

# Sender of the mail
SENDER="absender@domain.de"

# Recipient of the mail
RECIPIENT="empfaenger@domain.de"

# SMTP server
SMTPSERVER="smtp.domain.de"

# User name on the SMTP server
SMTPUSERNAME="MeinMailAccount"

# Password on the SMTP server
SMTPPASSWORD="MeinMailPasswort"

# Enable TLS for the SMTP connection
USETLS=1

#########################################################################
# NORMALLY THERE IS NO NEED TO CHANGE ANYTHING BELOW THIS
    COMMENT #
#########################################################################


# Use first argument as mail subject
if [ -n "$1" ]; then
```

```
42   SUBJECT="$1"
43 else
44   # No subject specified
45   SUBJECT=""
46 fi
47
48 # Use second argument as mail body
49 if [ -n "$2" ]; then
50   BODY="$2"
51 else
52   # No mail body specified
53   BODY=""
54 fi
55
56 # Generate the options list for sendEmail
57 OPTIONS=""
58
59 if [ -n "${SMTPSERVER}" ]; then
60   OPTIONS="${OPTIONS} -s ${SMTPSERVER}"
61 fi
62
63 if [ -n "${SMTPUSERNAME}" ]; then
64   OPTIONS="${OPTIONS} -xu ${SMTPUSERNAME}"
65 fi
66
67 if [ -n "${SMTPPASSWORD}" ]; then
68   OPTIONS="${OPTIONS} -xp ${SMTPPASSWORD}"
69 fi
70
71 if [ -n "${USETLS}" ]; then
72   if [ ${USETLS} == 1 ]; then
73     OPTIONS="${OPTIONS} -o tls=yes"
74   else
75     OPTIONS="${OPTIONS} -o tls=no"
76   fi
77 fi
78
79 # Send the mail with sendEmail
80 sendEmail -f ${SENDER} -t ${RECIPIENT} -u "${SUBJECT}" -m "${
       BODY}" ${OPTIONS}
81
82 exit 0
```

Zu editieren sind folgende Punkte: .

```
1 # Sender of the mail
2 SENDER="absender@domain.de"
3
4 # Recipient of the mail
5 RECIPIENT="empfaenger@domain.de"
6
7 # SMTP server
8 SMTPSERVER="smtp.domain.de"
9
10 # User name on the SMTP server
11 SMTPUSERNAME="MeinMailAccount"
12
13 # Password on the SMTP server
14 SMTPPASSWORD="MeinMailPasswort"
```

Hier müssen nur noch die Empfänger-Emai-Adresse und Absende-Email-

Adresse mit dazugehörigem SMTP- Server, Benutzername und Passwort angegeben werden. Danach muss das Skript ausführbar gemacht werden .

```
sudo chmod +x /usr/local/bin/mailnotify.sh
```

Im Anschluss kann die erste Email versendet werden .

```
mailnotify.sh "Test" "Dies ist eine Testnachricht."
```

Damit wird das Skript aufgerufen, wobei der text in den ersten Anführungszeichen den Betreff, un der zweite den Inhalt der Email. Hierbei kam es jedoch zu folgendem Fehler .

```
invalid SSL_version specified at /usr/share/perl5/IO/Socket/SSL.
    pm line 332
```

Dieser Fehler entsteht da die dazugehörige SSL konfiguration einen kleinen Fehler enthählt. Folgende Datei muss aufgerufen werden .

```
sudo nano /usr/share/perl5/IO/Socket/SSL.pm
```

und in Zeile 1490 muss folgender Inhalt .

```
m{^(!?)(?:(SSL(?:v2|v3|v23|v2/3))|(TLSv1[12]?))$}i
```

durch .

```
m{^(!?)(?:(SSL(?:v2|v3|v23|v2/3))|(TLSv1[12]?))}i
```

ersetzt werden.Nur das Dollar Zeichen am Schluss muss entfernt werden. Anschliessend können Emails versendet werden .

```
mailnotify.sh "Test" "Dies ist eine Testnachricht."
Jul 15 16:32:30 raspberrypi sendEmail[4163]: Email was sent
    successfully!
```

Damit auch Anhänge mit versandt werden können muss lediglich das Skript etwas verändert werden Hinzu kommt folgender Ausdruck. .

```
# Use third argument as attachment
if [ -n "$3" ]; then
  ATTACHMENT="$3"
else
  # No attachment specified
  ATTACHMENT=""
fi
```

Somit sieht das Skript wie folgt aus .

```
#!/bin/bash

# Sender of the mail
SENDER="absender@domain.de"

# Recipient of the mail
RECIPIENT="empfaenger@domain.de"

```

```
# SMTP server
SMTPSERVER="smtp.domain.de"

# User name on the SMTP server
SMTPUSERNAME="MeinMailAccount"

# Password on the SMTP server
SMTPPASSWORD="MeinMailPasswort"

# Enable TLS for the SMTP connection
USETLS=1

######################################################################

# NORMALLY THERE IS NO NEED TO CHANGE ANYTHING BELOW THIS COMMENT #
######################################################################


# Use first argument as mail subject
if [ -n "$1" ]; then
  SUBJECT="$1"
else
  # No subject specified
  SUBJECT=""
fi

# Use second argument as mail body
if [ -n "$2" ]; then
  BODY="$2"
else
  # No mail body specified
  BODY=""
fi

# Use third argument as attachment
if [ -n "$3" ]; then
  ATTACHMENT="$3"
else
  # No attachment specified
  ATTACHMENT=""
fi

# Generate the options list for sendEmail
OPTIONS=""

if [ -n "${SMTPSERVER}" ]; then
  OPTIONS="${OPTIONS} -s ${SMTPSERVER}"
fi

if [ -n "${SMTPUSERNAME}" ]; then
  OPTIONS="${OPTIONS} -xu ${SMTPUSERNAME}"
fi

if [ -n "${SMTPPASSWORD}" ]; then
  OPTIONS="${OPTIONS} -xp ${SMTPPASSWORD}"
fi

if [ -n "${USETLS}" ]; then
  if [ ${USETLS} == 1 ]; then
    OPTIONS="${OPTIONS} -o tls=yes"
  else
```

```
68      OPTIONS="${OPTIONS} -o tls=no"
69    fi
70  fi
71
72  if [ -n "${ATTACHMENT}" ]; then
73    OPTIONS="${OPTIONS} -a ${ATTACHMENT}"
74  fi
75
76  # Send the mail with sendEmail
77  sendEmail -f ${SENDER} -t ${RECIPIENT} -u "${SUBJECT}" -m "${
        BODY}" ${OPTIONS}
78
79  exit 0
```

Im Befehl zum senden wird nun ein drittes Argument hinzugefügt das den Pfad mit der Datei die zu versenden ist enthält. .

```
1  mailnotify.sh "Test" "Dies ist eine Testnachricht." anhang.tar.
       gz
```

4.1.9 Tally autostarten

Damit nach dem starten des Raspberry Pi direkt das Tally Programm geladen wird musste der Autostart für das Programm konfiguriert werden.

Die erste Idee war das Programm per cronjob nach jedem neuen hochfahren zu starten. Dafür musste zunächst ein kleines Skript geschrieben werden damit das Programm autostarten kann, welches dann in einem cronjob nach jedem hochfahren ausgeführt wird. .

```
1  sudo nano /etc/init.d/NameDesSkripts
```

.

```
1  #! /bin/sh
2  ### BEGIN INIT INFO
3  # Provides: NAME DES PROGRAMMES
4  # Required-Start: $syslog
5  # Required-Stop: $syslog
6  # Default-Start: 2 3 4 5
7  # Default-Stop: 0 1 6
8  # Short-Description:
9  # Description:
10 ### END INIT INFO
11
12 case "$1" in
13   start)
14     echo "NAME DES PROGRAMMES wird gestartet"
15     # Starte Programm
16     /pfad/des/programmes
17     ;;
18   stop)
19     echo "NAME DES PROGRAMMES wird beendet"
20     # Beende Programm
21     killall noip2
22     ;;
```

```
23    *)
24      echo "Benutzt: /pfad/des/programmes {start|stop}"
25      exit 1
26      ;;
27  esac
28
29  exit 0
```

Danach wird das Skript ausführbar gemacht .

```
1  sudo chmod 755 /etc/init.d/NameDesSkripts
```

Um zu testen ob das Skript auch richtig funktioniert wird es mit folgendem Befehl aufgerufen .

```
1  sudo /etc/init.d/NameDesSkripts start
```

und wieder gestoppt .

```
1  sudo /etc/init.d/NameDesSkripts stop
```

Damit der Raspberry das Skript beim booten lädt wird noch folgender Befehl ausgeführt der das Programm in den passenden Runlevel bringt .

```
1  sudo update-rc.d NameDesSkripts defaults
```

Als nächstes wird das Skript durch einen cronjob jedes mal beim starten des Raspberrys ausgeführt

.

```
1  sudo crontab -e
```

Dazu wurde folgende zeile eingefügt .

```
1  @reboot \bin\bash .....
```

Das '@reboot' steht für das ausführen der Datei nach jedem neuen hochfahren des Raspberry. Jedoch startete das Programm nach dem hoch fahren des Raspberrys nicht. Erst als die Konsole geöffnet wurde startete der cronjob das Skript und somit das Programm.

Die beschriebene Methode war lediglich dafür gedacht Programme auszuführen die keine GUI benötigen. Alle Programme die autostarten sollen und eine GUI benötigen müssen über die globale LXDE (Lightweight X11 Desktop Environment) Autostart funktion gestartet werden. Dazu erstellt man in folgendem Pfad .

```
1  /etc/xdg/autostart/
```

eine Datei die wie folgt lauten muss 'NamedesProgramm.desktop'. Wichtig ist das .desktop da sonst das Programm nicht nach dem hochfahren gestartet wird. .

```
[Desktop Entry]
Name=NamedesProgramm
Comment=
Exec=NamedesProgramm -forever -rfbport 5900 -rfbauth ~/.vnc/
    x11vnc.pass -o ~/.vnc/x11vnc.log -display :0
Terminal=false
Type=Application
```

Diese Datei wieder ausführbar machen .

```
chmod +x x11vnc.desktop
```

Und das gewünschte Programm startet nachdem die GUI geladen ist.

4.2 QT Creator - Das Tally programm

Als Programmierumgebung, für den Raspberry Pi 2, wurde Qt gewählt. Qt ist eine Klassenbibliothek von C++ und bietet die Programmierung für graphische Benutzeroberfläche und Datenbankfunktionen. Wir arbeiten mit der Version 4.8.1 von Qt, da neuere Versionen noch nicht mit dem Raspberry Pi 2 kompatibel sind. Als Debugger haben wir GNU gdb 7.8 für MinGW 4.9.1 gewählt.

Programmiert wird hauptsächlich auf einem PC, da dieser schneller debuggen kann. Darüber hinaus musste der Raspberry PI 2 noch konfiguriert werden, wie in 4.1 beschrieben, wodurch ein zeitgleiches arbeiten erschwert wäre. Es wurde sich darauf geeinigt das Programm zu Demonstrationszwecke, während den Teammeetings, auf dem Raspberry PI 2 laufen zu lassen. Zum Ende des Projekts, sobald die Konfiguration beendet ist, sollten wir denn Raspberry PI 2 bekommen um die Benutzeroberfläche letztendlich anpassen zu können.

4.2.1 Die Benutzeroberfläche

Als erstes haben wir erste Grundzüge der Benutzeroberfläche thematisiert. Letztendlich haben wir eine herausgearbeitet, welche in dem Plichtenheft zu sehen ist. Außerdem wurden mehrere Strukturen für das Hauptprogramm besprochen und haben uns auf einen Automaten geeinigt. Näheres dazu findet sich in dem Unterpunkt Aufbau des Programmes. Daraufhin wurde sich in Qt eingelesen und die ersten Funktionen für den Login-Screen programmiert. Wie bereits in Abschnitt 4.1 erklärt, wurde entschieden das Display zu wechseln. Außerdem ist uns Aufgefallen das sehr viele Klicks gebraucht werden um eine Kaufvorgang abzuschließen, was Benutzerunfreundlich ist. Aus diesen beiden Gründen wurde die komplette Benutzeroberfläche überarbeitet. Die Knopfgröße und die Schriftgröße mussten vergrößert werden als auch das Knöpfe, aufgrund Platzmangels, weggelassen werden mussten. Die Screens

sollten aber erst beim Programmieren festgelegt werden um evtl. Hürden zu meistern. Außerdem wussten wir zu diesem Zeitpunkt noch nicht, welches Display für den Raspberry Pi 2 gewählt wird. Die endgültige Benutzeroberfläche befindet sich unter dem Unterpunkt Funktionen des Programmes.

Nach dem die ersten Funktionen des Login-Screens und des Passwort-Screen implementiert wurden, wurde begonnen die Datenbank mit einzuarbeiten. Aus Abschnitt 4.1 geht heraus, dass wir uns für SQLite entschieden haben. Um auf diese zuzugreifen, mussten wir uns zunächst darauf einigen welche Tabellen angelegt werden und womit diese befüllt werden. Eine genaue Beschreibung der Datenbank befindet sich unter Abschnitt 5. Nachdem wir uns auf die Datenbank geeinigt haben, wurde diese in das Programm implementiert. Zu diesem Zeitpunkt wurde noch nicht mit einer eindeutigen ID jedes Benutzer gearbeitet sondern mit seinem Namen, was später zu Komplikationen führen würde. Zeitgleich zum Login-Screen und dem Passwort-Screen wurden alle, die bis zu diesem Zeitpunkt erarbeitete Screen oberflächlich implementiert. So konnten man zu diesem Zeitpunkt sich anmelden, (die Daten dafür wurden aus der Datenbank genommen) und konnte sich über alle Screens klicken, auch wenn die meisten keine Funktionen hatten.

Nachdem das Grundgerüst des Programmes stand, wurde um die Kauffunktion implementieren zu können, die ID die der eingeloggten Person global im Programm bekannt gemacht. Dies haben wird entweder über get-Funktionen realisiert oder beim Erzeugen eines neuen Screens mitgegeben. Daraufhin konnten die restlichen Funktionen implementieren werden. Als erstes wurden die Tabellen implementiert in welche alle kaufbaren Artikel aufgelistet werden. Die Information über die Artikel konnten über die Datenbank in Erfahrung gebracht werden. Nachdem die Tabellen mit Artikeln vorhanden waren, wurde mit dem Hauptaspekt des Programmes begonnen, dem Kaufvorgang.

Für den Kaufvorgang ist hauptsächlich das Shoppingcart verantwortlich. Sobald ein Artikel dort liegt, wird es immer auf dem rechten Teil des Bildschirmes angezeigt. Außerdem ist es dafür verantwortlich, dass wenn Artikel gekauft werden, den Kaufvorgang abzuschließen. Dies bedeutet vor allem, dass die Datenbank mit neuen Informationen gefüllt wird. Einerseits wird die Anzahl der entsprechenden Artikel geändert und das Konto des Käufers belastet. Dies bürgte eine große Hürde. Das Problem mit der Synchronität des Shoppingcart über den gesamten Einkauf. Jedes Mal wenn der Zustannd gewechselt wurde(siehe Aufbau des Programmes), wurde das Shoppingcart gelöscht und damit auch der Inhalt. Das Shoppingcart ist ein eigenständiges Teil Programmes und somit sollte es unabhängig sein, wenn sich der Zustand des Programmes ändert. Dies tut es allerdings nicht und wird jedes Mal gelöscht, wenn sich der Zustand ändert. Da wir das Problem nicht finden konnten, warum es sich jedes Mal mit löscht, wird nun beim Verlassen des

Zustandes eine Liste mitgegeben in der alle Artikel gespeichert sind die im Shoppingcart lagen. Mit dieser Liste wird ein neues Shoppingcart erzeugt in der diese Artikel liegen.

Nachdem diese Hürde genommen wurde, gab es nur noch wenige Funktionen die noch implementiert werden mussten. Bis zu diesem Zeitpunkt wurde das Programm fast ausschließlich auf einem PC programmiert, debugt und auch die Größen wurden auf diesen angepasst, wie bereits anfangs erwähnt. Deswegen war der nächste Schritt die Benutz Oberfläche, mit dem Raspberry PI 2 und dem neuen Display, anzupassen. Zusätzlich wurde auch der Barcode Scanner implementiert, was kein Problem dargestellt hat, da sich das Signal des Barcode Scanners verhält wie ein Tastendruck mit der Tastatur. Qt stellt für solche Signale bereits Funktionen. Somit war das Programm so gut wie fertig. Es fehlten nur noch ein paar kleine Funktionen und vor allem musste noch eventuelle Bugs entfernt werden. Alle Funktionen des Programmes wurden nicht während dieses Abschnittes erfasst, da dies den Umfang dieser Dokumentation sprengen würde. Alle wichtigen Funktionen wurde erwähnt und gegebenenfalls erläutert, vor allem in dem Abschnitt Funktionen des Programmes werden alle wichtigen erwähnt und was das Programm leistet.

4.2.2 Aufbau des Programmes

Das Programm wurde in 12 Klassen unterteilt.Im folgendem werden die einzelnen Klassen erläutert. Allgemein lässt läuft das Programm nach einem Automaten ab. Bildschirm wurde in Teilbereiche unterteilt, wodurch nicht bei jedem Zustandswechsel der komplette Bildschirm neu geladen werden muss, sondern nur die betreffenden Teile. Dadurch Teilen wir den Bildschirm in drei Teile. Die Kopfteil,linker Teil und rechter Teil. Näheres dazu wird in den dementsprechenden Klassen erläutert. main.cpp : Die main.cpp ist verantwortlich für den Ablauf des Programmes. Die Struktur dafür ist aufgebaut wie ein Mealy-Automat.In den Zustanänden sind die jeweiligen möglichenkeien aus diesem Zustand herauszukommen. Dies wird realisiert durch den exitcode des aktuellen Screen, welche zurückgegeben wird, wenn das passende Event dazu geschehen ist(Bsp. Knopf gedrückt). Zustände:

```
1                0 : Es wird der Login-Screen auf dem Bildschirm
                     gezeigt.
2                         Exitcode:
3                                  10 : Es wurde der Benutzer gew\"
                                        ahlt und m\"ochte sich nun
                                        mit dem Account anmelden.
4                                  !10 & !100 : Das Programm
                                        beendet sich.
5
6                1 : Es wird der Passwort-Screen auf dem
                     Bildschirm angezeigt.
7                         Exitcode:
```

```
8                                   20 : Es wurde auf den 'Back'
                                         Knopf gedr\"uckt gelangt in
                                         Zustand 0.
9                                   21 : Das Passwort wurde korrekt
                                         eingegeben und man gelangt
                                         zum Coffee/Sweet/Scan Men\"u
                                         .
10                                  !20 & !21 & !100 : Das Programm
                                         beendet sich.
11
12                  2 : Es gibt die Auswahlpunkte zwischen Coffee/
                        Sweets/Scan als auch die Favoriten oder das
                        Shoppingcart.
13                          Exitcode:
14                                  31 : Es wurde auf 'Coffee'
                                         geklickt und man gelangt zur
                                          Getr\"ankeliste mit dem
                                         Shoppingcart.
15                                  32 : Es wurde auf 'Sweets'
                                         geklickt und man gelangt zur
                                          S\"u \{\ss} uigkeitenliste
                                         mit dem Shoppingcart.
16 ---->            33 : Es wurde auf 'Scan' geklickt und man wird
    ins Scan-Menue geleitet mit dem Shoppingcart.
17                                  34 : Es wurde ein Artikel
                                         gescannt und wird in das
                                         Scan_Menue geleitet mit dem
                                         Shoppingcart.
18                                  99 : Es wurde auf 'Buy' geklickt
                                         . Der Kaufvorgang wird
                                         beendet. Die Datenbank wird
                                         aktgualisiert und man wird
                                         in den afterbuyscreen
                                         geschickt.
19                                  !31 & !32 & !33 ! & !34 & !99 &
                                         !100 & !98 : Das Programm
                                         beendet sich.
20
21                  3 : Es wird die Getr\"ankeliste mit dem
                        Shoppingcart gezeigt.
22                          Exitcode:
23                                  51 : Es wurde auf 'Back'
                                         geklickt und man wird in
                                         Zustand 2 gesetzt.
24                                  99 : Es wurde auf 'Buy' geklickt
                                         . Der Kaufvorgang wird
                                         beendet. Die Datenbank wird
                                         aktgualisiert und man wird
                                         in den afterbuyscreen
                                         geschickt.
25                                  98 : Ein Artikel wurde aus dem
                                         Warenkorb entdfernt.
26
27                  4 : Es wird die S\"u \{\ss} uigkeitenliste mit
                        dem Shoppingcart gezeigt.
28                          Exitcode:
29                                  51 : Es wurde auf 'Back geklickt
                                          und man wird in Zustand 2
                                         gesetzt.
30                                  99 : Es wurde auf 'Buy' geklickt
                                         . Der Kaufvorgang wird
                                         beendet. Die Datenbank wird
```

```
                                    aktualisiert und man wird in
                                     den afterbuyscreen
                                    geschickt.
31                                98 : Ein Artikel wurde aus dem
                                    Warenkorb entdfernt.
32
33                  5 : Es wird das Scan-Menue angezeigt.
34                                51 : Es wurde auf 'Back geklickt
                                     und man wird in Zustand 2
                                    gesetzt.
35                                99 : Es wurde auf 'Buy' geklickt
                                    . Der Kaufvorgang wird
                                    beendet. Die Datenbank wird
                                    aktgualisiert und man wird
                                    in den afterbuyscreen
                                    geschickt.
```

Dafür haben wir eine QListWidget ,da diese mit QListWidgetItem's gefüllt werden kann welche alle Informationen über ein Artikel speichern kann. Ein QListWidgetItem kann man sich vorstellen wie ein Array an dessen Position spezielle Werte gespeichert werden. In unserem Fall stehen FOlgende Werte in dem QListWidgetItem:

- Der Name der Artikels wird der Name des QListWidgetItem's (item->setText(name);) - Das Bild des Artikels hat einen extra Platz im QListWidgetItem (item->setIcon(picture);) - An der vierten Stelle wird die Artikel ID gespeichert (item->setData(4,itmeID);) - An der fünften Stelle wird der Preis des Artikels gespeichert (item->setData(5,price);) - An der sechsten Stelle wird die Menge gespeichert die noch vorrätig ist (item->setData(5,amount);)

An den ersten drei Stellen sind wir uns nicht sicher was dort gespeichert wird. Wir können nichts darauf expliziet speichern und wenn wir ausgeben lassen was dort gespeichert ist, bekommen wir einen leeren String zurück.

4.2.3 Funktionen des Programmes

Wie wir im vorangegangen Text gesehen haben, sind die Zeilenumbche automatisch ganz gut. Neue Abschnitte werden durch eine Leerzeile erzeugt. Wenn man trotzdem eine Zeile in einem Absatz abbrechen will, so geht das natrlich auch.
Aber eigentlich sehen diese Umbche seltsam aus und man sollte sie vermeiden. Da ist es doch in der Regel besser, gleich einen neuen Absatz zu beginnen. In Abschnitt 4.3 sehen wir, wie eine gute Absatzstruktur aussieht. brigens knnen wir auch einen neuen Seitenbeginn erzwingen.

Aber auch das sollte man in der Regel vermeiden. LaTeX bricht Seiten in der Regel selbst vernnftig um, und wenn man spter noch Text einfgt, stehen manuelle Seitenumbrche fast immer an der falschen Stelle.[1]

4.3 Sonderzeichen

Viele Sonderzeichen haben in LaTeX eine spezielle Bedeutung und drfen nicht einfach so verwendet werden, sondern werden durch spezielle Befehle erzeugt. Das ist ganz hnlich wie HTML. Zum Beispiel ist & ein Trennzeichen in Tabellen, $ signalisiert den Beginn und das Ende von mathematischem Text, % macht eine Zeile zu einem Kommentar, der von LaTeX natrlich ignoriert wird, { und } sind fr die Parametebergabe bei Befehlen reserviert, _ und ^ haben ihre Bedeutung bei dem Setzen von mathematischen Formeln und # ist erforderlich bei selbst definierten Befehlen.

5 Auflistungen und Aufzhlungen

Wir fassen zusammen, was wir bisher knnen:

- Die Schrift anpassen
- Den Text ausrichten. Da gab es folgende Mglichkeiten:
  - Blocksatz
  - linksndig
  - rechtsbndig
  - zentriert. Zentrierte Text ist vor allem gut um
    - Text hervorzuheben
    - Objekte, wie Tabellen und Grafiken zu zentrieren
- Umbrche erzwingen
- Sonderzeichen setzen

Bisher knnen wir noch nicht:

1. Tabellen erzeugen

[1] Manuelle Seitenumbrche sollten deswegen hchstens dann eingefgt werden, wenn das Dokument vollstndig fertig ist und die Seitenumbrche von LaTeX nicht zufriedenstellend waren.

2. Grafiken einbinden
3. mathematischen Text setzen. Dazu gehrt
   (a) weitere mathematische Sonderzeichen, wie z.B.
      i. griechische Buchstaben, etwa α, ζ, usw.
      ii. spezielle Akzente, wie $\vec{a}$ oder $\ddot{x}$
      iii. echte Sonderzeichen, wie $\oplus$ oder $\perp$
      iv. alle Mglichen Klammern, wie $\lfloor x \rfloor$
      v. und noch vieles mehr...
   (b) spezielle mathematische Objekte, wie Matrizen
   (c) automatische Nummerierung von Formeln
4. Referenzen und Bibliographie erzeugen
5. Prentationsfolien erstellen

Offensichtlich kann man hier ziemlich tief schachteln. Ob das allerdings immer sinnvoll ist?

6 Tabellen, Grafiken und Gleitobjekte

6.1 Grafiken

Hier hatten wir Gtig kommt?

6.2 Tabellen

Name	gemessen von Dr. Tex		
	Alter (Jahre)	Gre (in cm)	Gewicht (in kg)
Andreas	7	120	25
	10	141	34
	14	163	50
Beate	6	110	22
	9	138	32
	13	156	46
Tina	8	132	30
	11	151	43
	15	174	51

6.3 Bewegliche Objekte

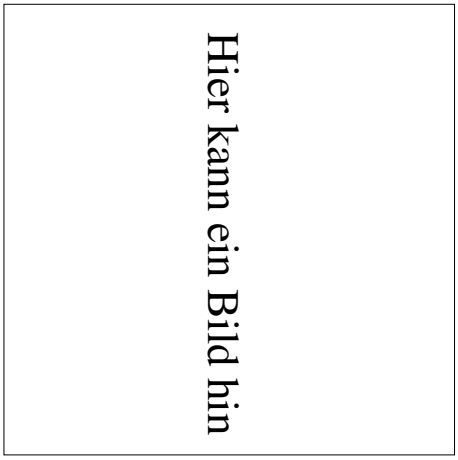

Abbildung 3: Dasselbe nochmal als Gleitobjekt.

Literatur

[AHU61] Arrow, Kenneth J. ; Hurwicz, Leonid ; Uzawa, Hirofumi: Constraint qualifications in maximization problems. In: *Naval Research Logistics Quarterly* 8 (1961), S. 175–191

Literatur

[1]

[2]