NAStrovje Pi

NAStrovje Pi – NAS-Service für KMU (und als Media Server für zuhause)

Projektdokumentation
im Rahmen des Moduls
BWI50205
Prof. Dr. rer. nat.Claus Brell

Erstellt von:
Dahmen, Jens, 839154
Frommen, Denis, 836724
Rezapour, Armin, 836531
Wiemer, Simon, 836519

Versionsstand: 3.0

Mönchengladbach, den 14.01.2014

Inhaltsverzeichnis

Anzahl Anschläge: 37728, Anzahl Worte: 4928

1 Management Summary

NAStrovje Pi ist der Name eines Network Attached Storages-Systems, welches mittels Raspberry Pi umgesetzt wurde und die Möglichkeit zur Überwachung via Android-App bietet.

Die Projektumsetzung erfolgt im Rahmen der Wahlpflichtveranstaltung „Web-Anwendungen“ im fünften Semester des Bachelor Studiengangs Wirtschaftsinformatik an der Hochschule Niederrhein.

Ziel des Projektes ist die Erschaffung eines kostengünstigen, lauffähigen NAS-Service samt Internetanbindung und einer Schnittstelle zu Android-Smartphones. Innerhalb des NAS sollen angeschlossene USB-Festplatten auf Änderungen hin überprüft und diese dokumentiert werden. Eine so entstandene Log-Datei wird via Webserver für den Außenzugriff via NAStrovje ApPi bereitgestellt.

Als Softwareentwicklungsprozess wird Scrum verwendet. Dabei übernimmt Denis Frommen eine übergeordnete Rolle als Projektleiter und Organisator ein. Die eigentliche Rollenverteilung aus Scrum wurde aufgrund der geringen Anzahl an Projektmitgliedern nicht übernommen. Lediglich die Art der Organisation und die Meetings wurden verwendet.

Dieses Projekt verwendet eine Client-Server-Architektur. Der Serverprozess wird mittels PHP-Skripte verwaltet, die Clientseite für die Schnittstelle zu Android-Endgeräten hingegen mittels Android SDK. Für die gesamte Architektur wurde ARIS als Modell gewählt.

Die ersten technischen Schritte bestanden aus der Konfiguration des Raspberry Pi, inklusive Installation des Samba-Servers für die Dateifreigabe und des Apache Webservers. Als nächstes wurden die PHP-Skripte für den Dateizähler, welcher die Log-Datei erstellt, programmiert. Danach musste eine Verbindung zum Webserver hergestellt und die Schnittstelle zu Android-Smartphones, ebenso wie die App als solche, entwickelt werden.

Ein Langzeittest wurde darüber hinaus durchgeführt, um ein stabiles, lauffähiges System garantieren zu können.

Die Ergebnisse der Planung und die Realisierung sind in dieser Dokumentation zu finden. Alle wichtigen Anforderungen konnten dabei erfüllt werden.

Der Name NAStrovje Pi leitet sich im Übrigen von der Abkürzung NAS (für Network Attached Storage) und dem Hauptakteur Raspberry Pi ab.

2 Projektbeschreibung

2.1 Ausgangslage

Für kleine bis mittelständische Unternehmen ist es nach aktueller Preislage häufig zu teuer ein fertiges NAS-System einzukaufen bzw. einrichten zu lassen.

Als günstige Alternative soll eine simple NAS-Lösung mittels Raspberry Pi, einem Mini-Computer, dienen, welche mit Gesamthardwarekosten von ungefähr 150 € belastet ist. Zudem bietet die Konfiguration des Raspberry Pi im Prinzip die Möglichkeit ebenso ein Firmen-Intranet aufzubauen bzw. diesen darin einzubinden sowie weitere Softwarekomponenten nachzurüsten.

Darüber hinaus kann eine solche Lösung auch für Privatanwender, insbesondere aufgrund des günstigen Einkaufspreises und der einfachen Handhabung, interessant sein.

2.2 Ziele des Projektes

Das Ziel dieses Projektes soll die Erschaffung eines kostengünstigen, lauffähigen NAS-Service mittels Raspberry Pi sein.

Zunächst muss der Raspberry Pi mit einer Standardinstallation (Installationspaket NOOBS, Raspian) versehen und für die weitere Anwendung vorkonfiguriert werden.

Später soll dieser eine angeschlossene USB-Festplatte scannen und die so erworbenen Informationen in einer Log-Datei initial als eine Art Inhaltsverzeichnis anlegen. Weiterhin soll der Raspberry Pi alle 10 Minuten die so angeschlossene Festplatte auf Änderungen überprüfen (d.h. belegter Speicherplatz, Anzahl der Ordner und Dateien) und diese aufzeichnen. Diese Überprüfung soll durch ein PHP-Skript realisiert werden.

Des Weiteren soll der Raspberry eine Anbindung zum Internet haben, welche ebenfalls zunächst einmal eingerichtet werden muss. Die Informationen über die externe Festplatte (ergo die erstellte Log-Datei) sollen anschließend übermittelt werden. Diese Verbindung und Datenübertragung soll ebenfalls mittels eines PHP-Skripts geschehen.

Darüber hinaus soll es dem Anwender möglich sein von einem beliebigen Client auf die so bereit gestellten Informationen zuzugreifen und die Daten auszulesen.

Als Erweiterung dieses Konstrukts soll zudem eine simple Android-App entwickelt werden, mit welcher die Log-Daten bequem von einem mobilen Endgerät außerhalb des Netzwerks abgerufen werden können. Die Aktualisierung der Daten in der Android-App soll mittels eines Refresh-Buttons auf den Stand des letzten Scans gebracht werden.

Der Aufbau der Hardwarekomponenten wird in im nachfolgenden Netzwerkdiagramm grafisch dargestellt:

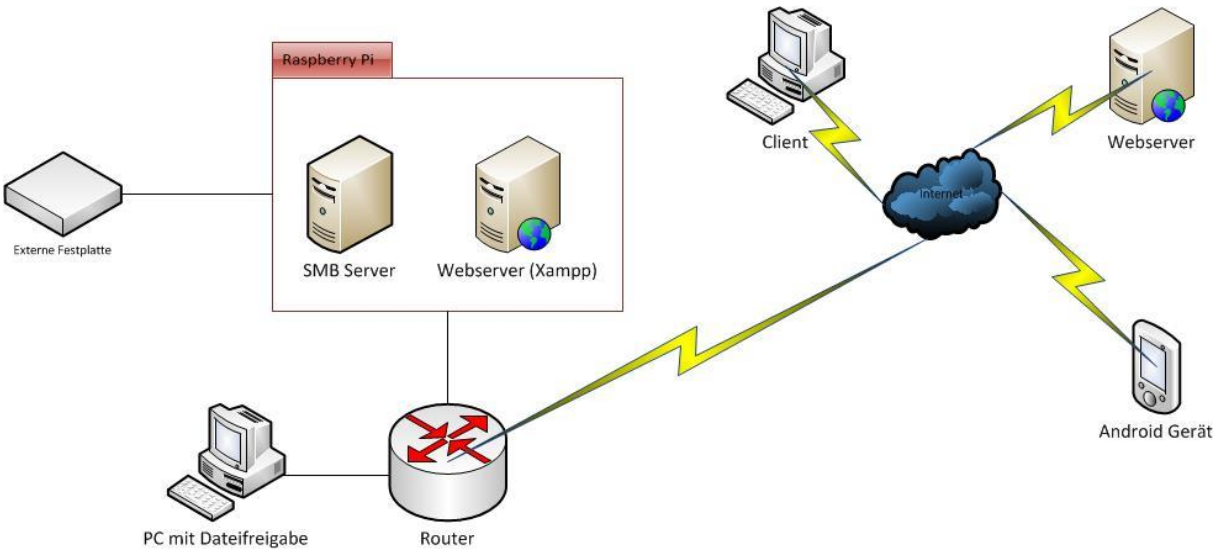


Abbildung 1: Netzwerkdiagramm

2.3 Ressourcen

2.3.1 Personen und Rollen

Name	**Rolle im Projekt**	**Spezialitäten**
Dahmen, Jens	Modellierer	Modellierung, Dokumentation, Projektplanung, Präsentation, Zeitplanung
Frommen, Denis	Projektleiter, Schriftführer	Dokumentation, Modellierung, Diagramme, Präsentation
Rezapour, Armin	Programmierer	Android App-Entwicklung, Skript-Programmierung, Dokumentation und Präsentation
Wiemer, Simon	Programmierer	Konfiguration des Raspberry Pi, Einrichtung der benötigten Server und Dienste, Dokumentation und Präsentation

2.3.2 Material

Hardware:

-Raspberry Pi inkl. Gehäuse
-SD-Karte
-Y-Kabel
-Netzteil
-Externe Festplatte (USB) inkl. Verbindungskabel
-Monitor inkl. HDMI-Kabel
-Android Smartphone
-Computer zur Programmierung/Dokumentation
-Hardware zur Internetanbindung

Software:

-Microsoft Visio
-Microsoft-Office
-verschiedene Bildbearbeitungssoftware
-Apache, PHP 5
-Eclipse + Android SDK
-Raspian inklusive Installationspaket NOOBS
-Notepad++
-Dropbox
-WhatsApp

3 Projektorganisation und –planung

3.1 Projektstrukturplan

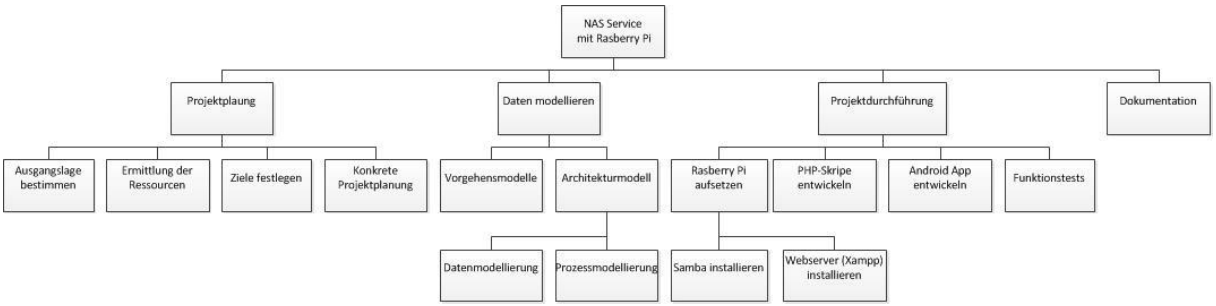


Abbildung 2: Projektstrukturplan

Nachdem die Projektziele festgelegt waren, konnte mit der Planung des Projektes begonnen werden. Als ersten Schritt wurde somit ein Projektstrukturplan entwickelt, anhand dessen die einzelnen Aspekte des Projekts in Meilensteine und Unterpunkte dieser Meilensteine aufgeführt wurden. Mithilfe dieser Struktur konnte anschießend ein klarer und übersichtlicher Projektplan ausgearbeitet werden.

3.2 Projektplan

3.2.1 Tabellarischer Projektplan

ID	Vorgang	Dauer	Vorgänger	Bearbeiter
0	Meilenstein Projektstart	0	-	
1	Meilenstein Projektplanung	0	-	
2	Ausgangslage bestimmen	0,5h	-	Frommen
3	Ziele festlegen	1h	2	Alle
4	Ermittlung der Ressourcen	0,5h	3	Alle
5	Material	0,5h	3	Alle
6	Konkrete Projektplanung	3h	4,5	Dahmen, Frommen
7	Meilenstein Daten modellieren	0	-	
8	Vorgehensmodelle	1h	6	Frommen
9	Architekturmodell	0,5h	8	Frommen
10	Datenmodellierung	1,5h	9	Frommen
11	Prozessmodellierung	5h	10	Dahmen, Frommen
12	Meilenstein Projektdurchführung	0	-	
13	Raspberry Pi aufsetzen	8h	11	Wiemer
14	Samba-Server installieren	2h	13	Wiemer
15	Apache Webserver installieren	5h	13	Wiemer
16	PHP-Skripte schreiben	18h	11	Rezapour
17	Dateien auf Webserver legen(Server Abts)	0,5h	16	Rezapour
18	Android App programmieren	9h	11	Rezapour
19	Tests	20h	15,17,18	Wiemer, Rezapour
20	Überarbeitung/Korrektur Dokumentation	10h	19	Alle
21	Meilenstein Projektende	0	-	

Für die Dokumentation der Projektplanung wurde zunächst ein tabellarischer Projektplan aufgestellt, der einen einfachen Überblick gibt.

Die tatsächliche Abarbeitung im Projektverlauf wurde in kleineren Sprints (max. 14 Tage) mit kleineren Aufgabenpaketen durchgeführt. Jedoch gab es durch äußere Einflüsse Re-Priorisierungen der abzuarbeitenden Aufgaben und nicht geschaffte Arbeiten, die daraufhin in den folgenden Sprints nachgeholt werden mussten. Dies verleitete zur häufigen Neudefinition des kommenden Sprints, welches die Dynamik und fortschreitende Treibkraft von Scrum nur bestätigt hat.

3.2.2 Gantt-Diagramm

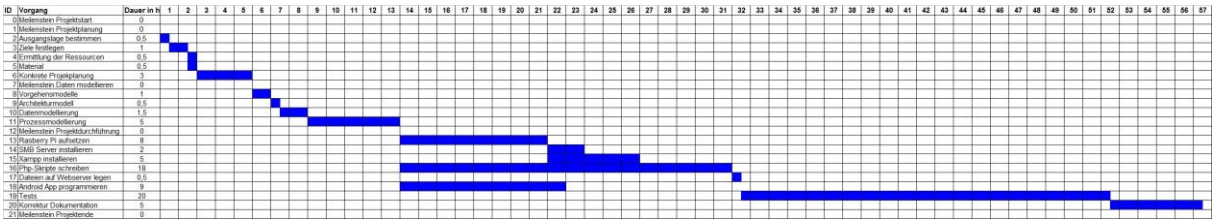


Abbildung 3: Gantt-Diagramm

3.3 Projektrisiken

Aufgrund der Tatsache, dass einige Hardware-Komponenten verwendet werden, stellt ein Geräteausfall immer ein potentielles Risiko dar.

Darüber hinaus ist dieses Projekt eine gänzlich neue Erfahrung für das Projektteam, weshalb jegliche Kalkulationen bezüglich Zeit- und Aufgabenplanung von der Realität abweichen können. Denn zum einen gab es keine Vorkenntnisse zum Umgang mit dem Raspberry Pi, zum anderen hatten die Projektmitglieder in dieser Teambelegung vorher noch keine gemeinsamen Projekte bearbeitet.

Zudem ist der tatsächliche Programmieraufwand zu Beginn noch nicht konkret abschätzbar, ebenso wie die damit verbundenen Risiken. Schließlich birgt jede Programmierzeile das potentielle Risiko einen Teilaspekt des Projektzieles aufgrund von (unentdeckten und) nicht behobenen Fehlern zumindest nicht in Gänze zu erreichen.

Überdies kann bei Gruppenprojekten – insbesondere bei größeren Teams – immer ein Problem der Absprache bestehen, welchem jedoch von vornherein bestmöglich entgegengewirkt werden sollte. Daher wurde entsprechend auch als Vorgehensmodell Scrum ausgewählt und eine Online-Absprache (WhatsApp) sowie –Unterlagenspeicherung bzw. -verteilung via Dropbox eingerichtet.

4 Anforderungen aus Kunden- bzw. Benutzersicht

Die Anforderungen eines kleinen bis mittelständischen Unternehmens an diese NAS-Lösung sind die einwandfreie Funktion, welche im Projektziel definiert worden ist, sowie die Einhaltung des geplanten Budgets.

Die genauen Anforderungen wurden bereits im Kapitel 2.2 „Ziele des Projektes“ dargestellt.

4.1 Vorgehensmodell

4.1.1 Auswahl des Vorgehensmodells

Das Projektteam hat sich für Scrum als Vorgehensmodell entschieden.

Scrum ist ein iteratives, inkrementelles und empirisches Vorgehensmodell welches seine Ursprünge in der Softwaretechnik hat. Es zeichnet sich durch ein dynamisches Vorgehen durch ständige Re-Priorisierung von Aufgabenpaketen zur bestmöglichen Zielerreichung aus.

Aufgrund der geringen Größe des Projektteams (4 Personen) kann jedoch keine übliche Rollenverteilung wie in Scrum vorgesehen stattfinden. Zudem existiert in diesem Gruppenprojekt, anders als bei Scrum angedacht, die Rolle des Gruppenleiters.

Jedoch gefällt die Idee der Aufteilung des Projektendziels in einzelne Sprints und die grafische Darstellung der Abarbeitung mittels eines simplen Burndown-Charts verhilft bei der Priorisierung und Organisation der anstehenden Aufgaben.

Als Beispiel der Burndown-Chart für die Tests am Ende der Programmierung.

Es wurde ein Aufwand von 20h geschätzt und für die Abarbeitung dieses zeitlichen Aufwands eine Sprint-Dauer von 13 Tagen angesetzt.

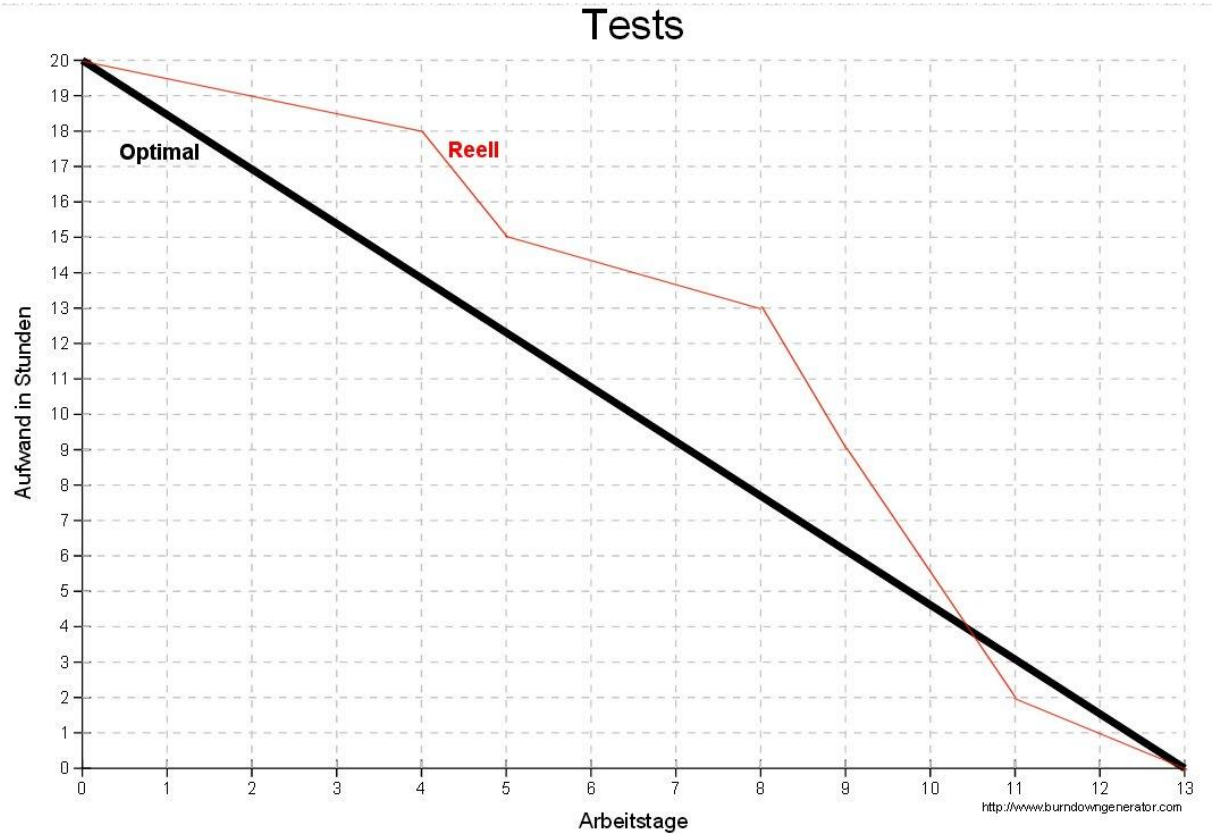


Abbildung 4: Burndown-Chart Tests

Aufgrund der geografischen Entfernungen zwischen den Projektmitgliedern haben sich eine Online-Besprechung und ein Online-Datenaustausch (via WhatsApp und Dropbox) als optimale Lösung herauskristallisiert. Auf diesem Wege konnte auch ein Daily Scrum-Meeting (wenn auch nicht immer zeitlich synchron) stattfinden.

Hauptsächlich für die Sprint Planungsmeetings, Reviews und Tests wurden Treffen in Persona als unerlässlich angesehen, da die weiteren Arbeiten durch organisierte Aufgabenteilung (und Überprüfung bzw. Absprache in Daily Meetings) möglich war.

Für weitere Informationen über das Vorgehensmodell Scrum in seiner Reinform befindet sich am Ende des Anhangs (7.8) eine Ausarbeitung dieses Themas, welches im Rahmen des Studiums von dem Projektleiter in der Vergangenheit erstellt worden ist.

4.1.2 Konkrete Durchführung

Bei der Vorgehensweise im Projekt wurde sich strikt an das Scrum-Modell gehalten.

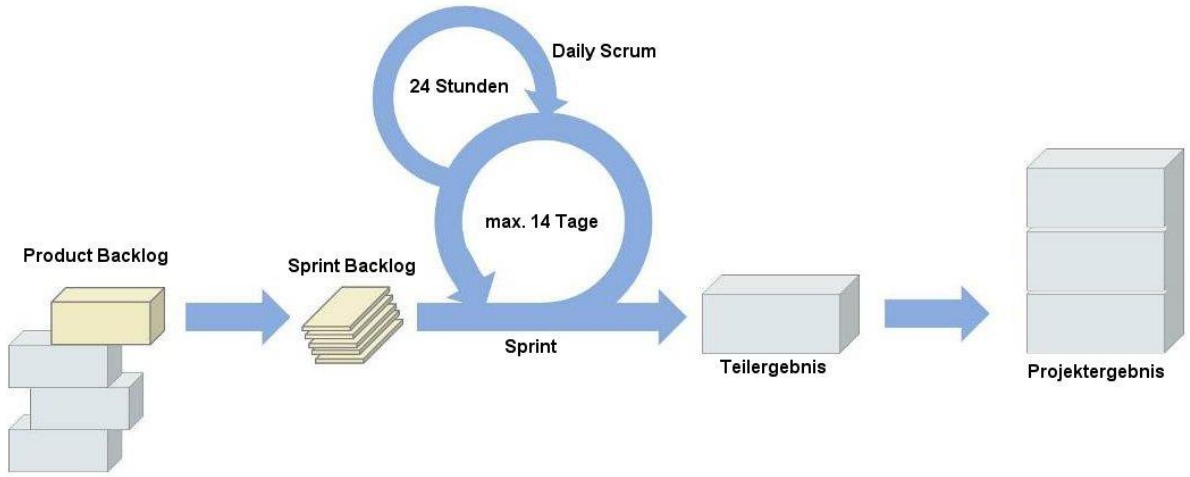


Abbildung 5: Scrum-Modell

Zunächst einmal folgt aus der Zielbeschreibung die Auflistung der gewünschten Ergebnisse.

Aus den Ergebnissen (oder Zielen) lassen sich Aufgaben ableiten, die erforderlich sind um diese zu erreichen. Diese Aufgaben werden im so genannten *Product Backlog* festgehalten. Aus dem Product Backlog (also den gesamten Aufgaben-Pool) werden nun einzelne *Sprint Backlogs* abgeleitet, also kleinere Aufgabenpakete die innerhalb eines gewissen Zeitraumes (dem so genannten Sprint) abzuarbeiten sind. Darüber hinaus gibt es nach gewissen Sprints (welche in Umfang und zeitlicher Dauer variieren können) so genannten Milestones, also Meilensteine die im Projektverlauf erreicht werden sollen; sozusagen größere Passagen innerhalb des Projektes (z.B. Einrichtung/Vorkonfiguration des Raspberry Pi; vgl. hierzu Abbildung 3: Gantt-Diagramm).

Während die Sprints durchlaufen werden (welche in diesem Projekt eine maximale Länge von 14 Tage haben), finden täglich Meetings (die *Daily Scrum* Meetings via WhatsApp) zur Status-Überprüfung statt, bei denen die folgenden Fragen von jedem Projektmitglied beantwortet werden:

-Woran arbeite ich gerade?

-Was habe will ich bis zum Sprintende (bzw. zum Ende des Tages) erreicht haben?

-Was habe ich für Hindernisse? / Gibt es Probleme?

Nach jedem Sprint soll ein (weiterer) Teil des Projektergebnisses erledigt sein, welche in Summe genommen am Projektende das Gesamtergebnis (in diesem Fall die lauffähige NAS-Lösung) darstellen.

Die Tests zur Lauffähigkeit sowie Demonstration innerhalb der Gruppe wurden jeweils am Ende des jeweiligen Sprints, nahe dem Projektende, durchgeführt.

4.2 Architekturmodell

4.2.1 Auswahl des Modells und der Notationen

Die Wahl des Architekturmodells fiel auf ARIS.

Dieses Konzept der Architektur integrierter Informationssysteme nach Scheer, besteht aus fünf Beschreibungssichten aufgrund derer es jeglichen Anforderungen gerecht werden soll.

Die fünf ARIS-Sichten im vereinfachten, so genannten ARIS-Haus zusammengefasst:

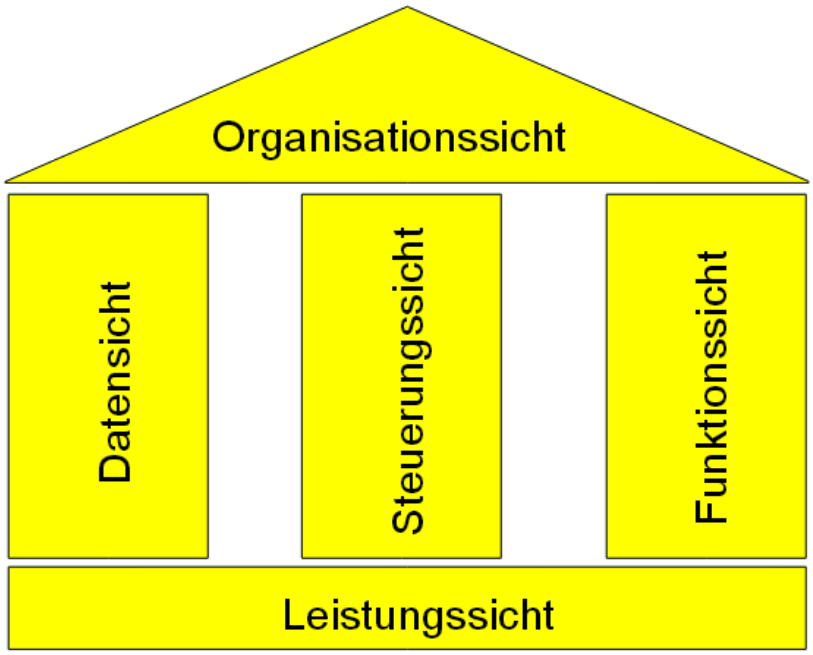


Abbildung 6: ARIS-Haus

Das Dach dieses Hauses bildet die Organisationssicht. Sie beschreibt die Aufbauorganisation eines Unternehmens mit ihren Beziehungen untereinander. Als Modellierung eignet sich hierfür ein Organigramm.

Die erste Säule stellt die Datensicht dar. Meist wird in einem Entity Relationship-Modell die Beziehung zwischen sämtlichen Ereignissen aufgezeigt.

In der Funktionssicht werden alle relevanten betrieblichen Prozesse dargestellt, die das Projekt betreffen. Für die Modellierung eignet sich in den meisten Fällen ein Funktionsbaum.

Die Leistungssicht, im ARIS-Haus als Fundament anzusehen, wird häufig mittels eines Produktbaumes abgebildet und fasst sämtliche materiellen (Güter) und immateriellen Leistungen eines Unternehmens zusammen.

Die Steuerungssicht beinhaltet die vier vorher genannten Sichten und stellt sie, in der Regel durch eine ereignisgesteuerte Prozesskette, in einen logischen und zeitlichen Plan.

4.2.2 Datenmodellierung

1. Funktionssicht:

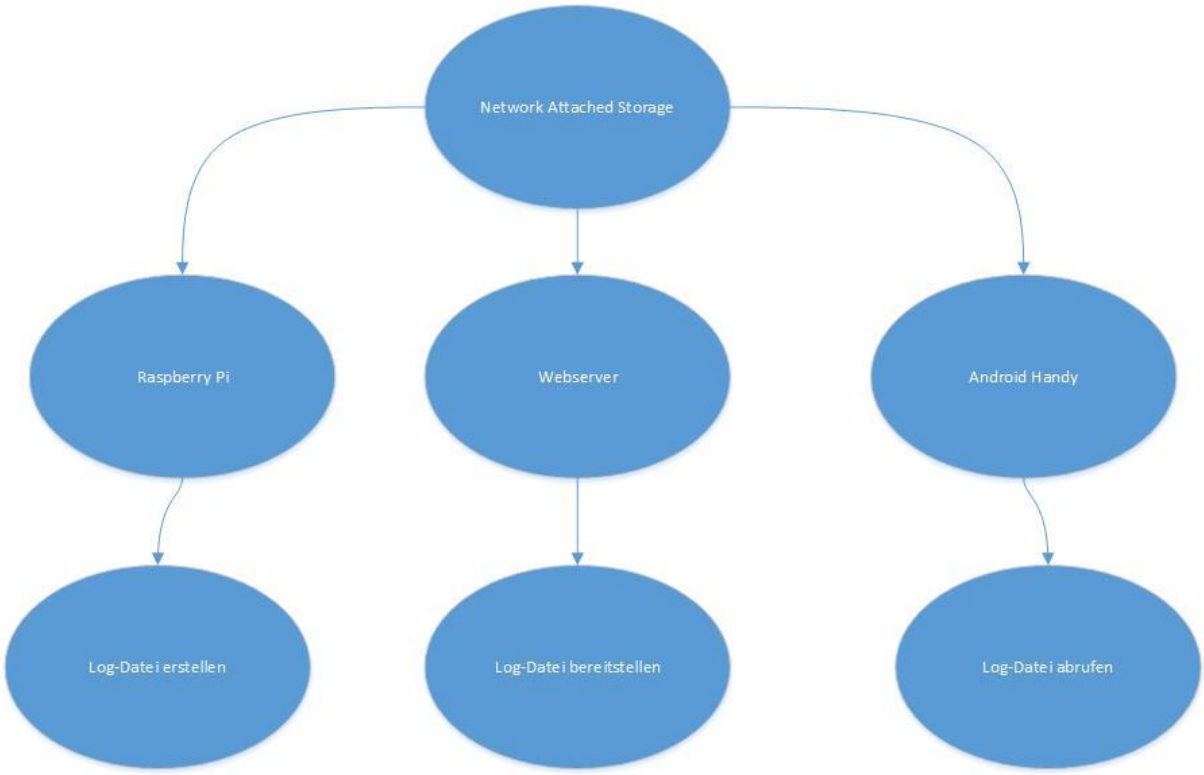


Abbildung 7: Funktionsbaum NAS

In diesem Funktionsbaum sind die drei Komponenten des Network Attached Storages zu sehen.

Zum einen der Hauptakteur Raspberry Pi, auf dem sich das meiste abspielt, wie das PHP-Skript zur Erstellung der Log-Datei. Zum anderen der Webserver, auf dem u.a. für das Android Gerät, welches separat auch aufgelistet ist, die Log-Datei bereitgestellt wird.

2. Organisationssicht:

Da im Rahmen dieser Projektarbeit kein tatsächlicher Kunde involviert ist, entfällt diese Sicht.

3. Datensicht:

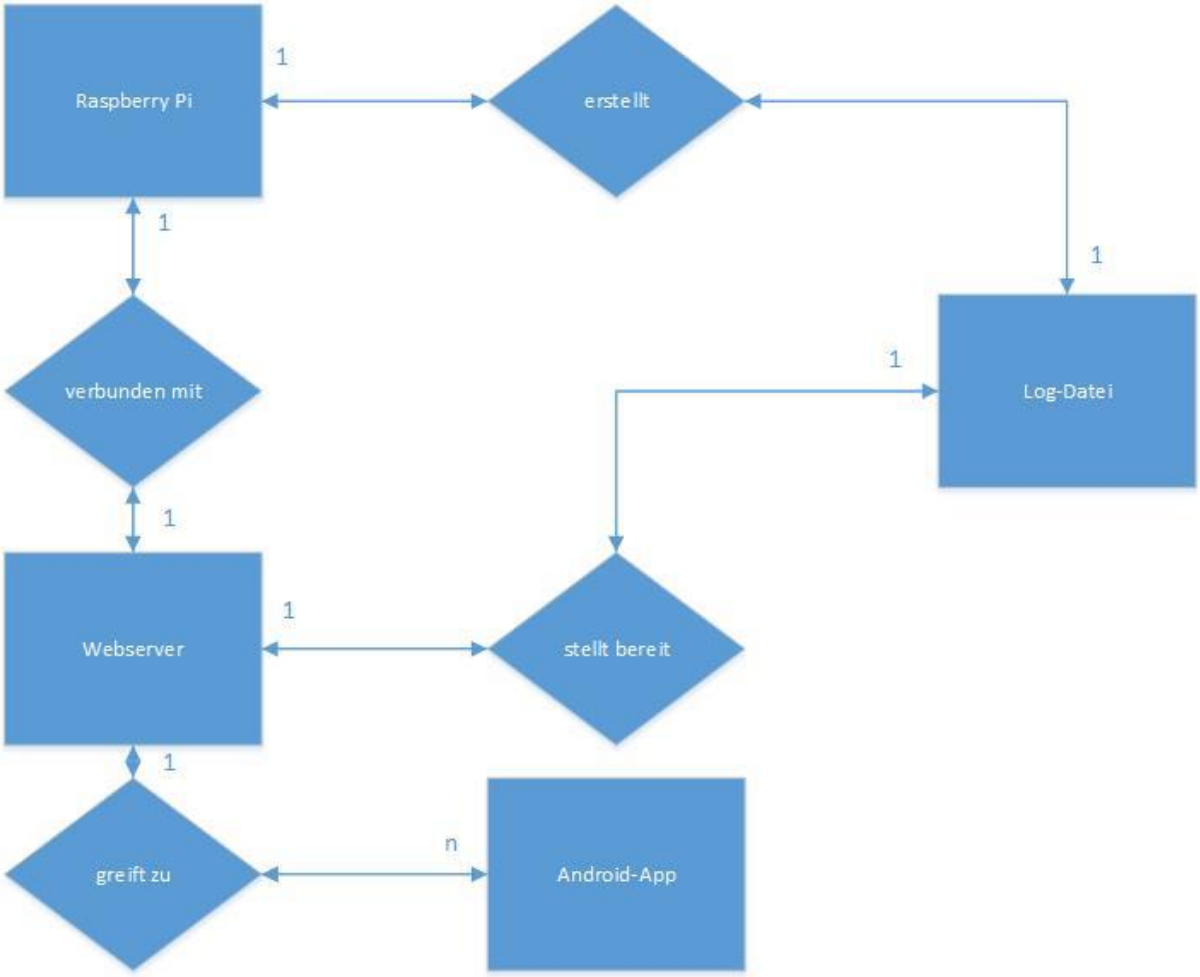


Abbildung 8: ER-Modell Grundkonzept

Eine vereinfachte Darstellung aus der Datensicht des Projektes mittels ER-Modell.

Der Raspberry Pi erstellt zum einen die Log-Datei und ist des Weiteren mit dem Webserver verbunden. Der Webserver kann, durch die Verbindung zum Raspberry Pi die Log-Datei für die Android-App – welche auf eben diese zugreifen möchte – bereitstellen.

4. Leistungssicht:

In dem vorliegenden Projekt ist die Erstellung der Leistungssicht nicht möglich bzw. nötig.

5. Steuerungssicht:

Die einzelnen Diagramme zur Steuersicht sind der nachfolgenden Prozessanalyse und -modellierung (4.2.3) zu entnehmen.

4.2.3 Prozessanalyse und – modellierung

Modell 1: Raspberry Pi aufsetzen und vorkonfigurieren

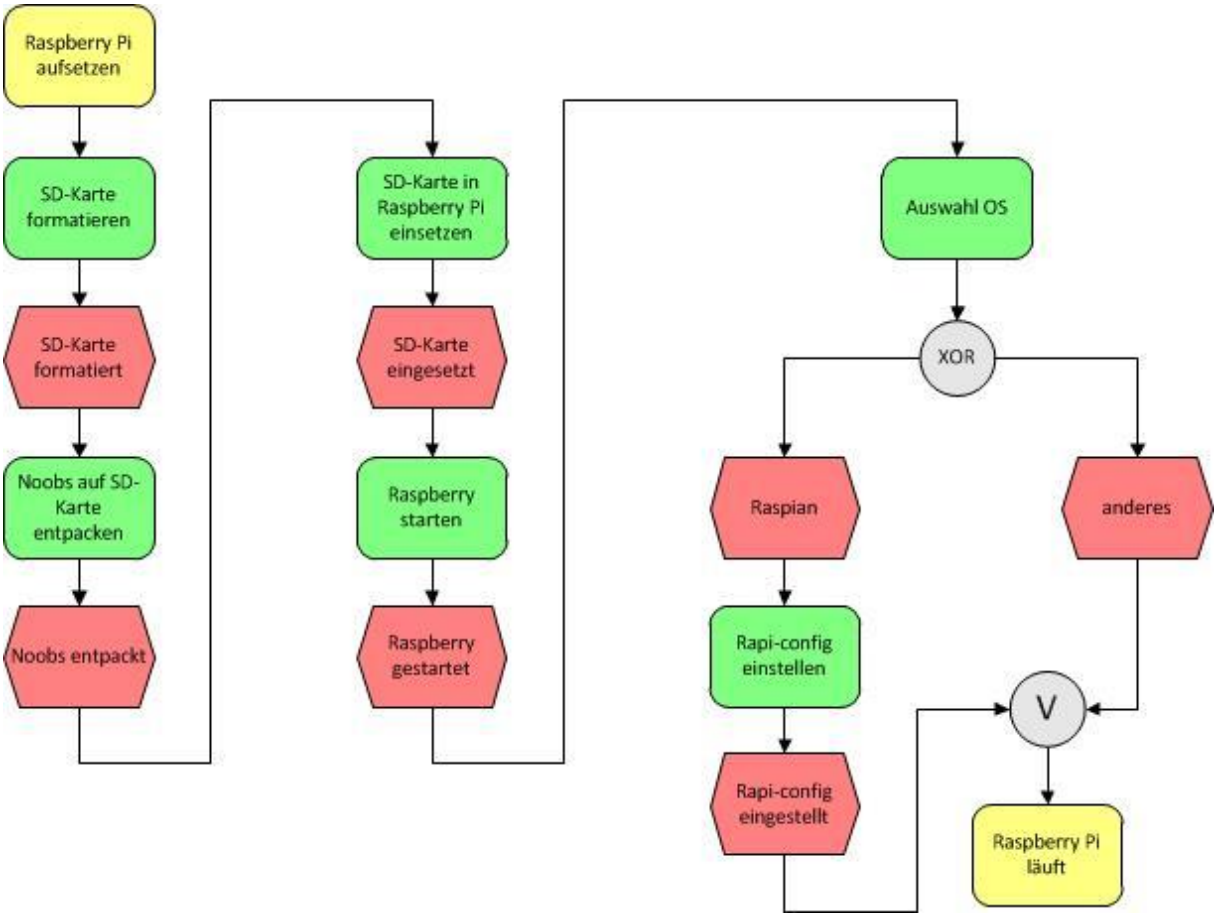


Abbildung 9: Raspberry Pi aufsetzen

Um den Raspberry Pi aufzusetzen muss zunächst das jeweilige Betriebssystem auf eine SD-Karte gespielt werden, welche später in diesen eingesetzt wird. Bevor dies jedoch getan werden kann muss die SD-Karte formatiert werden. Anschließend wird der Installer NOOBS (New Out Of Box Software) auf die SD-Karte entpackt. Sobald diese nun in den Raspberry Pi eingesetzt wird, besteht die Auswahl aus mehreren verschiedenen Betriebssystemen. Im Fall dieser Projektarbeit haben wir uns für Raspian entschieden.

Nach der Installation muss noch vor dem ersten Start die Raspi-config eingestellt werden. Jedoch muss bis auf die Änderung des Tastaturlayouts hier vor dem ersten Start nichts weiter geändert werden.

Modell 2: Der Dateizähler

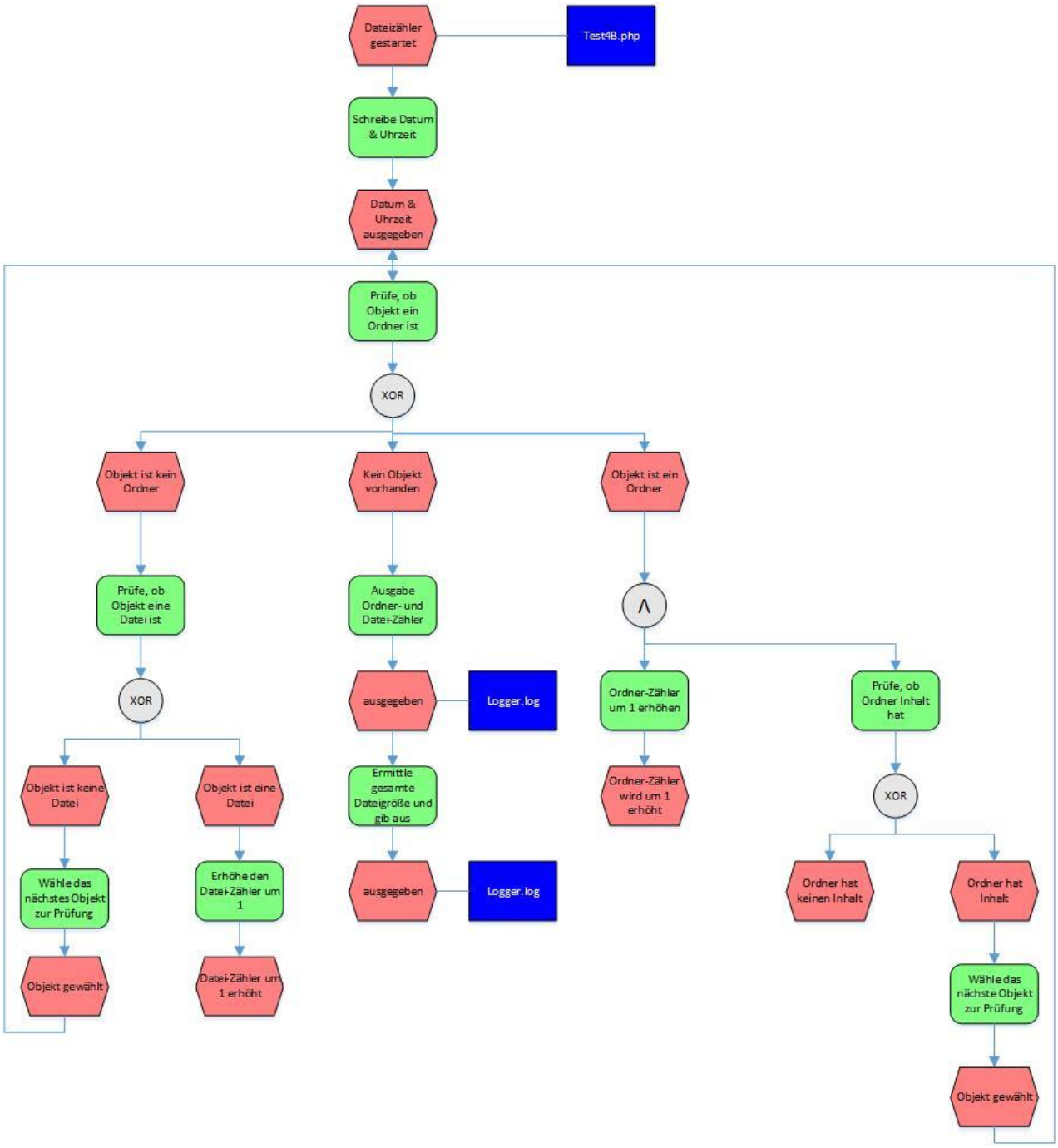


Abbildung 10: Dateizähler

Sobald der Dateizähler gestartet wird, gibt er das Datum und die Uhrzeit aus. Anschließend beginnt die Prüfung von Objekten nacheinander.

Wenn das Objekt ein Ordner ist, wird der Ordner-Zähler um den Wert 1 erhöht. Zudem wird geprüft, ob der Ordner einen Inhalt hat. Besteht kein Inhalt, geschieht nichts. Wenn der Ordner einen Inhalt hat, wird dieser geöffnet und geprüft. Weiter geht es dann mit der erneuten Prüfung, ob es sich um einen Ordner handelt (ergo eine Schleife).

Handelt es sich hingegen um keinen Ordner, wird geprüft, ob es sich um ein eine gewöhnliche Datei (File) handelt. Ist dies nicht der Fall, wird das nächste Objekt zur Prüfung gewählt und die Schleife beginnt wieder mit der Prüfung, ob ein Ordner vorliegt. Wenn die Datei ein regulärer File ist, wird der Datei-Zähler hingegen um 1 erhöht.

Bei der Entscheidung, ob eine Datei ein Ordner ist, oder nicht, besteht auch die Möglichkeit, dass gar kein Objekt mehr vorhanden ist. Dies tritt ein, wenn alle Objekte geprüft worden sind. Dies ist auch stets der Weg wie das Skript des Dateizählers endet: er gibt nun die erworbenen Informationen zu Anzahl der Ordner und Dateien aus, überprüft den Speicherplatz und gibt diesen ebenfalls aus.

Modell 3: Auf den Raspberry Pi von außen zugreifen

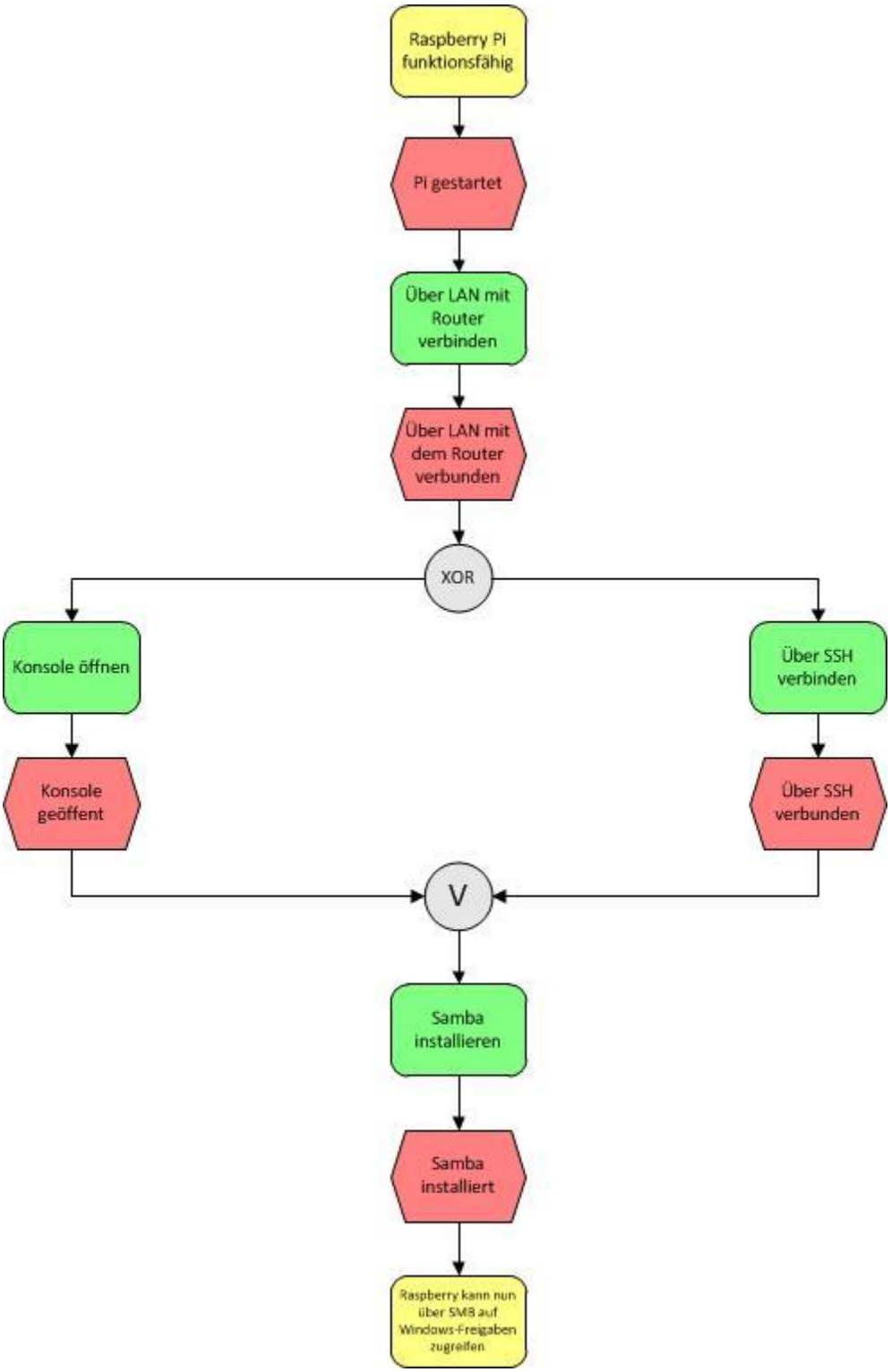


Abbildung 11: Zugriff auf Raspberry Pi

Der Raspberry Pi wird über ein LAN-Kabel mit dem Router verbunden. Nun kann entweder die Konsole des Raspberry gestartet oder von einem gewöhnlichen Computer über SSH auf den Raspberry Pi zugegriffen werden. Hier kann nun der von Raspian mitgelieferte SMB-Client Samba installiert werden. Sobald dieser installiert wurde kann mit dem Raspberry Pi auf Windows-Freigaben zugegriffen werden und umgekehrt.

Modell 4: Android-App

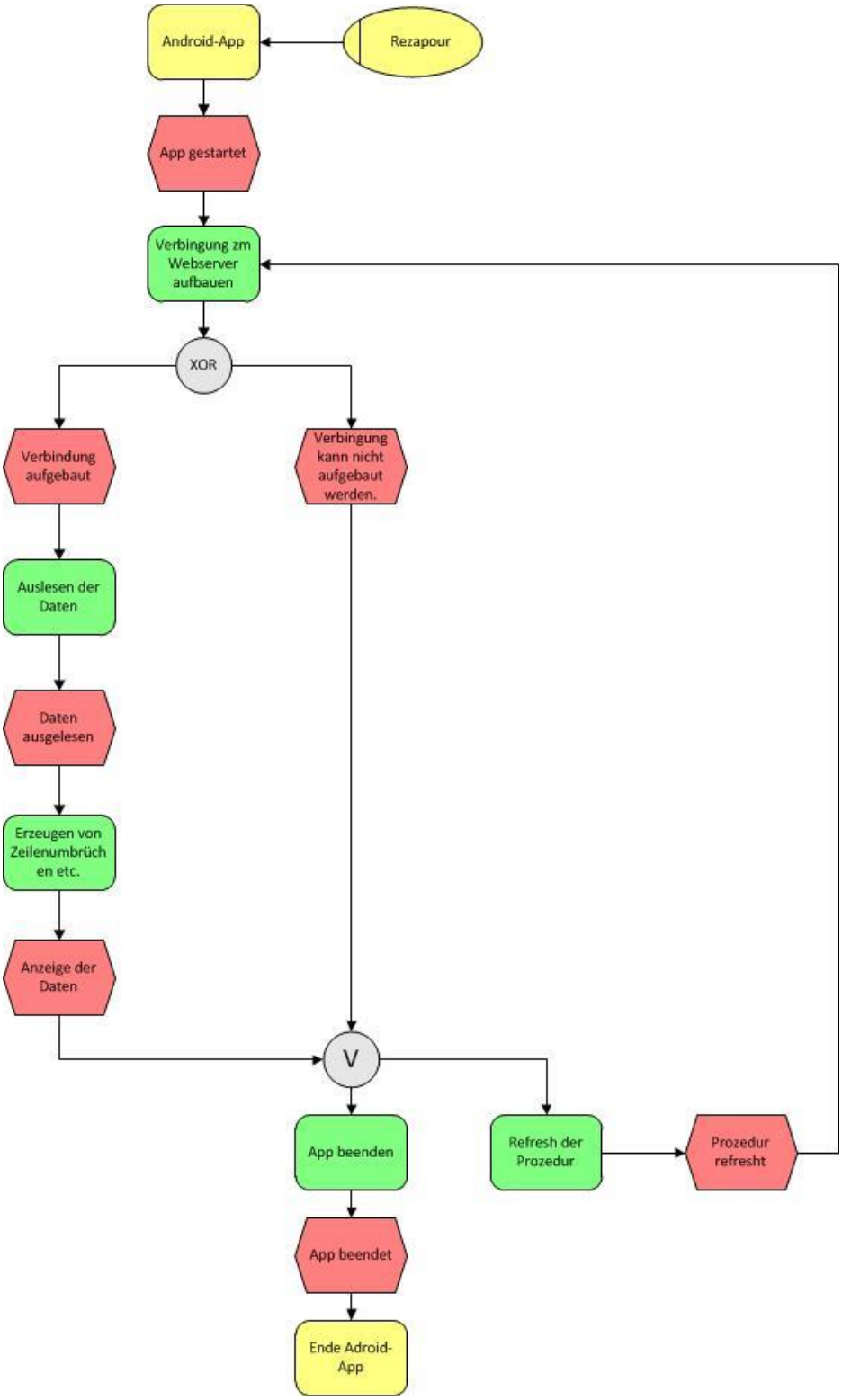


Abbildung 12: Android-App

Nach Start der App baut diese eine Verbindung zum Webserver auf. Sobald die Verbindung aufgebaut wurde, liest die App die Daten vom Webserver aus. Die Daten werden nun optisch angepasst und auf dem Display ausgegeben.

Sollte keine Verbindung zustanden kommen, bleibt die Anzeige leer.

Über einen Refresh-Button besteht die Möglichkeit den Verbindungsaufbau zum Server erneut zu starten und somit auch die Daten vom letzten bzw. aktuellsten Scan abzurufen.

5 Realisierung

Nachdem wir nun die Projektbeschreibung, die Projektorganisation und die Anforderungen aus Kunden – und Benutzersicht behandelt haben, kommen wir nun zu der eigentlichen Umsetzung unseres Projektes. Diesen Abschnitt haben wir, wie nachfolgend zu erkennen ist, in die Realisierung der Skript-Programmierung und des Raspberry Pi-Servers unterteilt.

5.1 Server

Der erste Schritt bestand hier aus der Formatierung der SD-Karte, damit wir diese in den Raspberry Pi einsetzen können. Mithilfe des „Win 32 Disk Imager“ wird das Image des Installers „NOOBS“ (New Out Of Box Software), welches wir vorher aus dem Internet heruntergeladen haben, auf die SD-Karte gespielt. Anschließend lässt sich das Installationsprogramm starten und das gewünschte Betriebssystem auswählen. In unserem Fall haben wir uns für Raspian entschieden, haben also eine bootfähige SD-Karte erzeugt, mit der nachfolgend die Einrichtung des Raspberry Pi beginnen kann.

Das Tool „Win 32 Disk Imager“ ist übrigens auch geeignet um ein Backup des bestehenden Systems zu machen, damit im Falle eines Hardwaredefekts auch eine Datensicherung vorhanden ist.

Nachdem die grundlegende Installation nun abgeschlossen ist, müssen jedoch vor dem ersten Start des Rapberrys noch einige Einstellungen in der „Raspi-Config“ vorgenommen werden. Hier haben wir jedoch vorerst nur das Tastaturlayout auf Deutsch gestellt.

Da wir nach dem ersten Start des Raspberrys gemerkt haben, dass noch weitere Einstellungen in der „Raspi-Config“ vorgenommen werden können, haben wir anschließend unter der Option „change_pass“ das Passwort für den User „Pi“ geändert, die Sprache des Systems mit dem Befehl „*de_DE.UTF-8 UTF-8*“ auf Deutsch gesetzt und die Zeitzone ebenfalls auf Berlin gesetzt. Da wir eine neuere Version von Raspian verwenden, ist der SSH-Server bereits vorinstalliert und wir mussten ihn lediglich in der „Raspi-Config“ aktivieren und anschließend starten.

Damit wir jetzt den Raspberry Remote ohne Bildschirm steuern können, müssen wir die IP-Adresse des Servers mithilfe des Befehls „ifconfig“ herausfinden. Nun können wir über einen Windows Computer im lokalen Netzwerk und der Software „PuTTY“ auf unseren Server zugreifen.

Der nächste Punkt der Konfiguration bestand aus der Einrichtung des Samba-Servers, damit mit einem Windows Client auf die externe Festplatte des Raspberry zugegriffen werden kann.

Der Samba-Server lässt sich relativ einfach mit folgender Befehlszeile installieren:

„*sudo apt-get install samba samba-common-bin*“.

Nachdem dieser Installationsvorgang durchgelaufen ist, müssen jedoch noch ein paar Anpassungen in der Samba Konfigurationsdatei vorgenommen werden, diese rufen wir mit dem Befehl

„*sudo nano /etc/samba/smb.conf*“

auf.

Wir kommentieren hier unter dem Punkt „Authentication“ den Punkt

„security = user“

aus, löschen also das Doppelkreuz (#), damit später der Zugriff auf die Dateifreigabe nur über die Authentifizierung vollzogen werden kann. Weiterhin müssen die Freigaben am Ende der Datei erstellt werden und dem Benutzer „Pi“ die notwendigen Rechte vergeben werden. Die genauen Befehle und der dazugehörige Quellcode kann in dem Screenshot in der Implementierungsanleitung nachgesehen werden.

Nach erfolgreicher Änderung der folgenden Schritte muss der Samba-Server mit dem Befehl

„*sudo /etc/init.d/samba restart*“

neugestartet werden und die Dateifreigabe funktioniert.

Der nächste Schritt bestand aus der Installation des Apache Webservers, welcher eine spezielle Benutzergruppe benötigt, die mit dem folgenden Befehl erstellt werden konnte.

„*groupadd www-data usermod –a –G www-data www-data*“

Sobald diese Gruppe erstellt ist, können wir die Pakete des Raspberrys mit dem Befehl

“*apt-get update*”

aktualisieren und anschließen den Apache mithilfe des Befehls

„*apt-get install apache2*“

installieren.

Um die Installation des Apache Webservers erfolgreich zu testen, rufen wir nun die IP-Adresse des Servers im Web-Browser auf und erhalten nun die Meldung „It works!“.

Da nun jetzt auch der Webserver erfolgreich läuft, müssen wir nun noch PHP 5 auf dem Raspberry installieren, damit die benötigten Skripte auch laufen. Hierfür starten wir die Installation mit dem Befehl

„*sudo apt-get install php5*“.

Damit keine Probleme mit dem Webserver auftreten haben wir danach laut Anleitung aus dem Internet (Internetquelle im Quellenverzeichnis) noch ein paar optionale Pakete installiert. Unter anderem werden hierbei häufig benötigte Bibliotheken und der alternative PHP Cache installiert, der uns eine schnellere Verarbeitung der Skripte bietet, da diese schon vorkompiliert werden. Der Befehl hierfür lautete:

„*sudo apt-get install libapache2-mod-php5 libapache2-mod-perl2 php5 php5-cli php5-common php5-curl php5-dev php5-gd php5-imap php5-ldap php5-mhash php5-mysql php5-odbc php-pear php-apc*“.

Auch diese Installation wollen wir natürlich testen und wechseln mit

„*cd /var/www*“

in das Verzeichnis des Webservers. Hier können wir per Befehl

„*sudo gedit phpinfo6.php*“

eine Testdatei erstellen und schreiben folgenden Inhalt in diese Datei:

„*<? php phpinfo (); ?>*“.

Sobald PHP erfolgreich installiert ist und der Webserver läuft, erhalten wir eine PHP Infoseite, wenn wir

„*192.168.xxx.xxx/phpinfo6.php*“

in unserem Web-Browser aufrufen.

(Wir haben den Texteditor gedit zum bearbeiten von Textdateien verwendet, da wir mit dem Standardeditor nano bei dieser Installation massive Probleme hatten. Zu diesem Problem kommen wir aber später noch einmal bei den Lessons Learned.)

Abschließend und nachdem der Raspberry vollständig für unsere Ansprüche konfiguriert ist, müssen wir noch die zu verwendenden Skripte einbinden und das bestehende System umfangreich testen.

5.2 Skript-Programmierung

Der gesamte Quellcode wird separat in ausgedruckter Form und digital (per E-Mail) übergeben.

Für die Bestimmung der Anzahl, der Ordner und Dateien im PHP Skript test4B.php, wurde ein rekursiver Aufruf verwendet, der alle Verzeichnisse auf dem NAS-Server durchforstet und die Dateien mitzählt.

Hierbei ist zu beachten, dass das aktuelle Verzeichnis durch einen Punkt "." und das hierarchisch übergeordnete Verzeichnis durch zwei Punkte ".." deklariert wird.

```php
// Ueberprueft ob das angegebene Verzeichnis existiert
if(is_dir($dir))
  {
  $afile["directory"]=0;
  $afile["file"]=0;

  chdir($dir);
  $handle=opendir(".");
  while($file=readdir($handle))
    {

    // Handelt es sich bei $file um ein Unterverzeichnis?
    if(is_dir($file) && $file!="." && $file!="..")
      {

      // directory um eins erhoehen
      $afile["directory"]++;

      // rekursiver Aufruf mit aktuellem Verzeichnis
      $y=count_file($file);

      // Ergebnisse des rekursiven Aufrufs der Funktion zu
      // directory und file aufaddieren
      $afile["directory"]+=$y["directory"];
      $afile["file"]+=$y["file"];
      }

    // Handelt es sich bei $file um eine Datei?
    if(is_file($file))
      {

      // file um eins erhoehen
      $afile["file"]++;
      }
    }

  // Verzeichnisstruktur zurueck gehen
  if(stristr($dir,"../")) chdir($_SERVER["DOCUMENT_ROOT"].substr($_SERVER["REQUEST_URI"],0,strrpos($_SERVER["REQUEST_URI"],"/")));
  elseif($dir!=".") chdir("../");

  closedir($handle);
  }
```

Abbildung 13: Test4B.php

6 Lessons Learned

Während des Projektverlaufes wurde immer deutlicher, dass eine Gruppenarbeit mit vier Personen, welche täglich auch andere Verpflichtungen (durch Studium und Arbeit) hatten, sich nicht einfach gestaltet. Innerhalb eines Unternehmens werden daher während Projektarbeiten in der Regel die Leute von ihren Arbeiten sozusagen abkommandiert und arbeiten mit vollem Fokus an besagtem Projekt.

Aufgrund anderer anfallender Aufgaben mussten daher einige Mal die Aufgaben während eines Sprints re-priorisiert und teilweise in den nächsten Sprint mit übernommen werden. Bei einem weiteren Projekt solchen Umfanges sollte daher ein noch größerer Puffer eingeplant werden.

Die Abstimmung via WhatsApp hingegen, welche sich aufgrund geografischer Distanz angeboten hatte, hat hingegen gut geklappt.

Dadurch bestand trotzdem die Möglichkeit täglich ein Daily Scrum Meeting abzuhalten, auch wenn gelegentlich Projektmitglieder nicht zum richtigen Zeitpunkt teilnehmen konnten. Die Asynchronität dieses Informations-austausches war hingegen kein Problem. Einziger Nachteil war, dass die in WhatsApp ausgetauschten Informationen und Überlegungen nicht ohne weiteren Aufwand auf den Computer übertragen und dort weiter bearbeitet bzw. genutzt werden konnten. Für die Zukunft sollten in dem Fall Systeme wie Google Wave genutzt werden, welche beispielsweise auch mobil angefangen und dann an einem Desktop-Rechner aufgegriffen und fortgeführt werden können.

Weiterhin gab es vorübergehend erhebliche Probleme bei der Realisierung unseres Projektes, nämlich genauer bei der Installation und Konfiguration des Apache Webservers auf dem Raspberry Pi. Dieses Problem äußerte sich in der fehlerhaften Speicherung der PHP-Testdatei, welche eine erfolgreiche Konfiguration des Apache Webservers darstellen sollte. Diese Darstellung des Skripts wurde uns auch nach mehrstündigen Tests nicht angezeigt, und Dateien im Homeverzeichnis des Webservers wurden auf dem Raspberry Pi fehlerhaft abgespeichert.

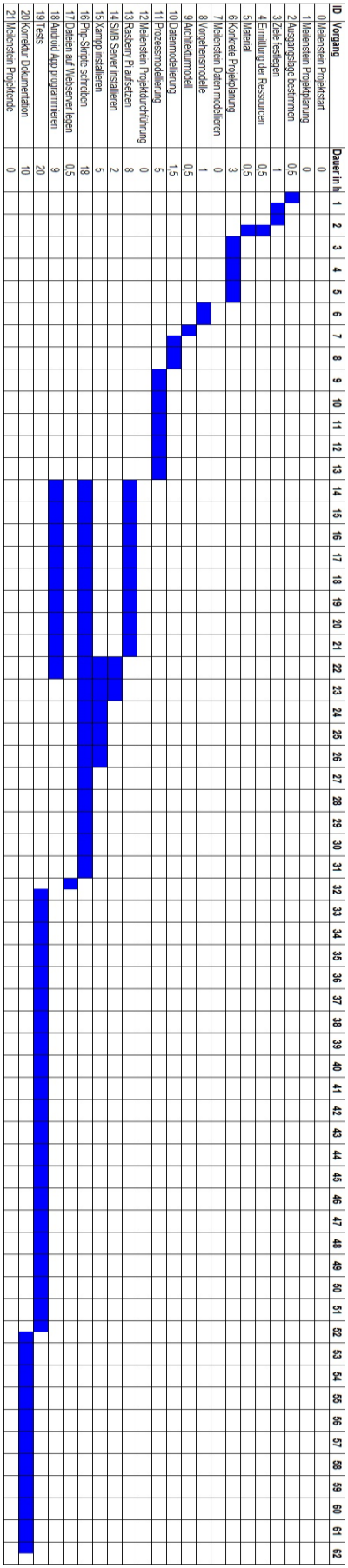
Da wir auch von anderen Quellen (Internet & Herr Brell persönlich) kein positives Feedback über mögliche Erfahrungen mit diesem Problem bekamen, geriet die Konfiguration des Raspberry Pi vorübergehend ins Stocken.

Schließlich kamen wir auf die Idee, anstatt des Standard-Texteditors „nano“ einen anderen Editor auszuprobieren und hatten schließlich unsere Lösung.

Mithilfe des Editors „gedit“ gelang es uns, die PHP-Testdatei erfolgreich und richtig im dazugehörigen Verzeichnis zu speichern.

Diese Schwierigkeiten haben unsere Zeitplanung ein wenig durcheinandergebracht und unseren Ablauf verzögert, deswegen ist es ratsam und notwendig, für kommende Projekte ausreichend Puffer für mögliche ähnliche Probleme einzuplanen.

7 Anhang



ID	Vorgang	Dauer in h
0	Meilenstein Projektstart	0
1	Meilenstein Projektplanung	0
2	Ausgangslage bestimmen	0,5
3	Ziele festlegen	1
4	Ermittlung der Ressourcen	0,5
5	Material	0,5
6	Konkrete Projektplanung	3
7	Meilenstein Daten modellieren	0
8	Vorgehensmodelle	1
9	Architekturmodell	0,5
10	Datenmodellierung	1,5
11	Prozessmodellierung	5
12	Meilenstein Projektdurchführung	0
13	Raspberry Pi aufsetzen	8
14	SMB Server installieren	2
15	Xampp installieren	5
16	Php Skripte schreiben	18
17	Dateien auf Webserver legen	0,5
18	Android App programmieren	9
19	Tests	20
20	Korrektur Dokumentation	10
21	Meilenstein Projektende	0

7.1 Implementierungsanleitung

In den folgenden Absätzen soll kurz und knapp erläutert werden, wie der Raspberry Pi installiert, konfiguriert und genutzt werden kann.

7.1.1 Benötigte Ressourcen

Zur Realisierung unseres Projektes wurden folgende Ressourcen benötigt:

1. Raspberry Pi Server mit folgenden Eigenschaften:
    - SSH Server
    - Samba Server zur Dateifreigabe
    - Apache Webserver
    - PHP ab Version 5

2. Windows Computer mit folgenden Eigenschaften:
    - Betriebssystem mindestens XP, besser 7 oder 8
    - SSH-Client (für Remotesteuerung)
    - aktueller Web-Browser

3. Smartphone mit folgenden Eigenschaften:
    - Android ab Version 3.2
    - WLAN / 3G Funktion

7.1.2 Notwendige Schritte

Für die komplette Installation sind jeweils mehrere Schritte am Server sowie auch am Skript, bzw. Client durchzuführen.

1. Raspbian Installation
    - „Raspbian wheezy“ aus dem Internet herunterladen
    - Image Datei mit dem „Win 32 Disk Imager“ auf die SD-Karte spielen
    - Raspbian von SD-Karte booten → Konfigurations-Tool erscheint

2. Raspberry Grundkonfiguration
    - Tastaturbelegung, Zeitzone und Sprache auf Deutsch stellen
    - Root-Partition erweitern (Speicherplatz SD-Karte voll ausnutzen)
    - Desktop direkt nach dem Booten starten
    - Passwort für den Benutzer „Pi“ ändern
    - Netzwerk konfigurieren (IP-Adresse dynamisch vergeben)

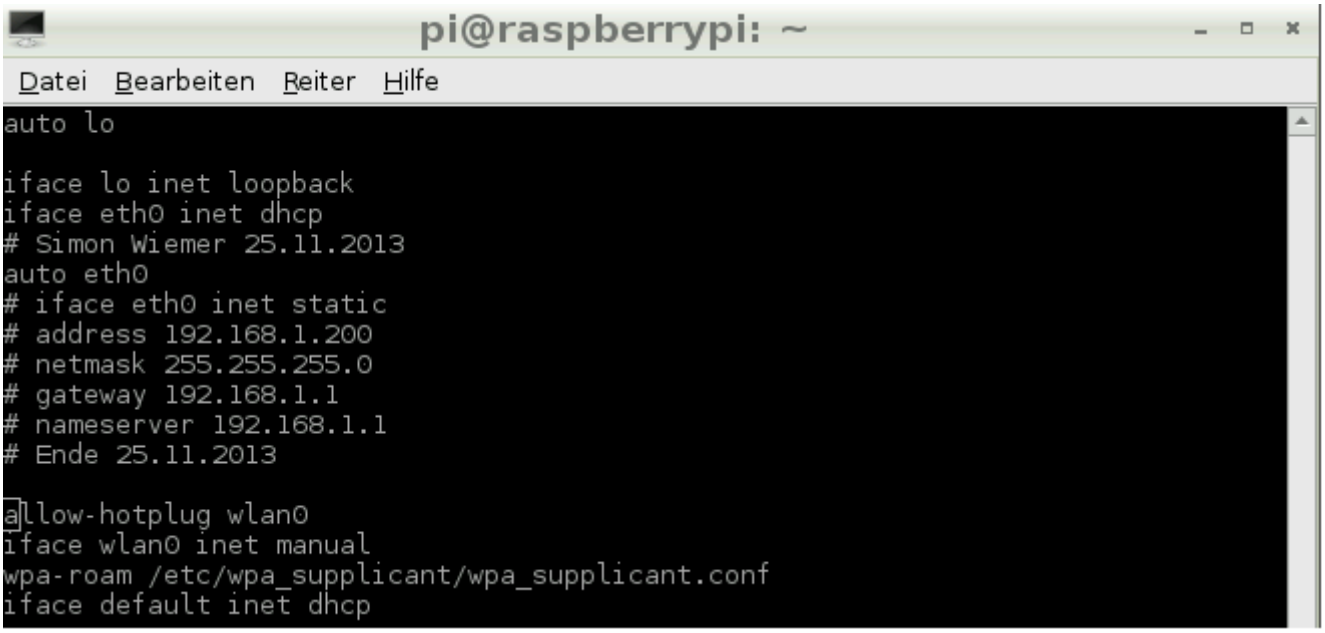


Abbildung 14: Screenshot 1

- Verbindung testen (Windows-Client kann Rasperry per Ping erreichen)
- SSH-Server auf Raspberry aktivieren
- Zugriff auf den Raspberry über Windows-Client aktiv (Software Putty)
- Update des RasPi („*sudo apt-get update & sudo apt-get upgrade*“)

3. Konfiguration Samba-Server

- Terminal starten und vorhandene Updates laden
- angeschlossene Laufwerke anzeigen lassen „*sudo fdisk –l*“

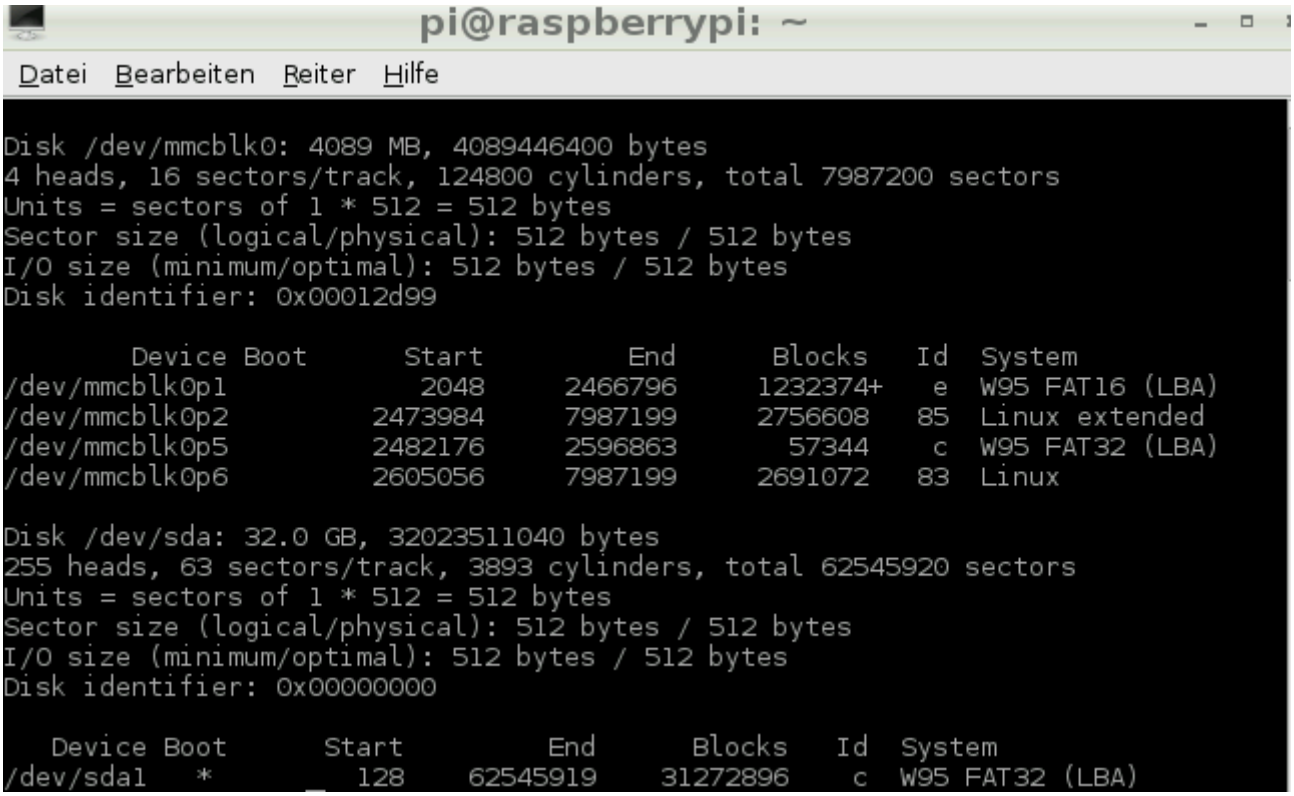


Abbildung 15: Screenshot 2

- USB-Stick mit 32 GB im unteren Teil zu erkennen (Disk /dev/sda)
- Samba installieren („*sudo apt-get install samba samba-common-bin*“)
- Samba-Konfiguration aufrufen (“*sudo nano /etc/samba/smb.conf*”)
- Sicherheitsmaßnahmen anpassen (security = user)
- Nutzer für Samba hinzufügen (“Pi”, Passwort: “Brell”)
- Freigaben definieren (siehe Screenshot unten)

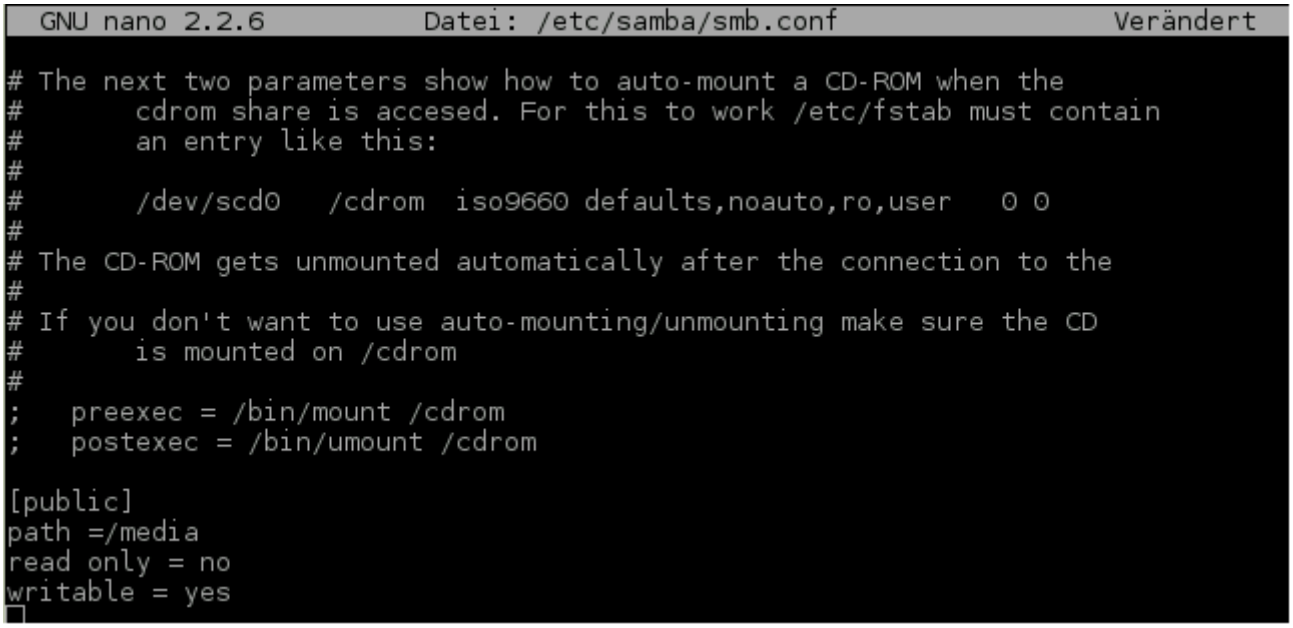


Abbildung 16: Screenshot 3

- Samba-Server neu starten („*sudo /etc/init.d/samba restart*“)
- Einhängepunkte der Laufwerke bestimmen („*sudo nano /etc/fstab*“)
- Dort werden dann die Partition hinzugefügt, die eingehängt werden sollen
- Testen der Freigaben im Netzwerk auf einem Windows-Client
-

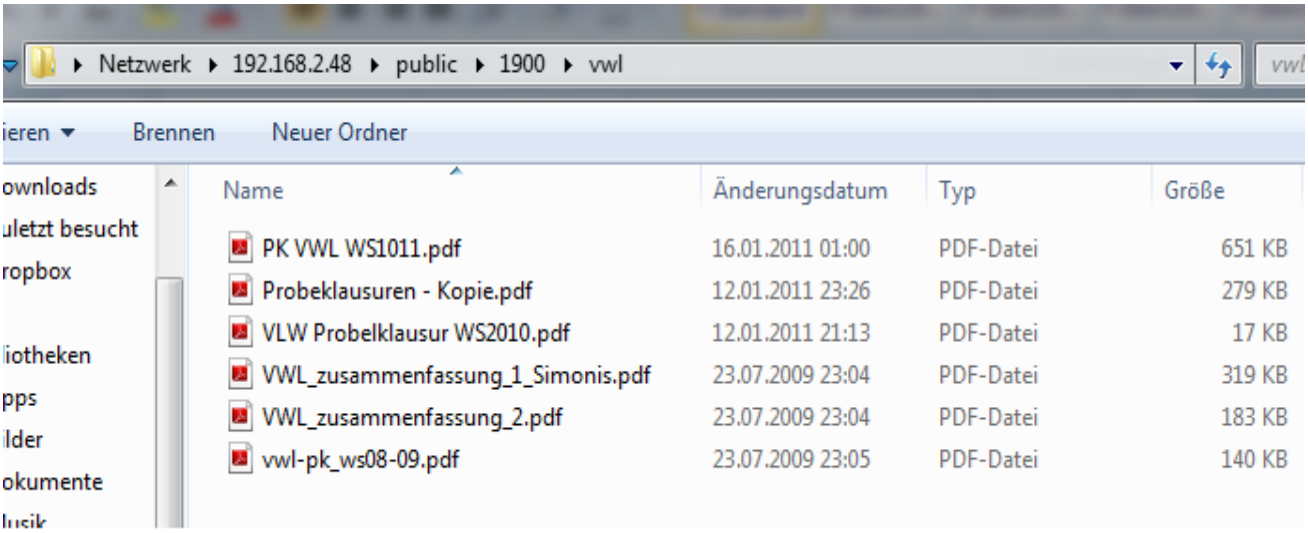


Abbildung 17: Screenshot 4

4. Konfiguration Apache Webserver

- Benutzergruppe erstellen („*groupadd www-data usermod –a –G www-data www-data“)*
- Update des RasPi („*sudo apt-get update & sudo apt-get upgrade*“)
- Installation Apache (“*sudo apt-get install apache2*”)
- Installation des Webservers testen

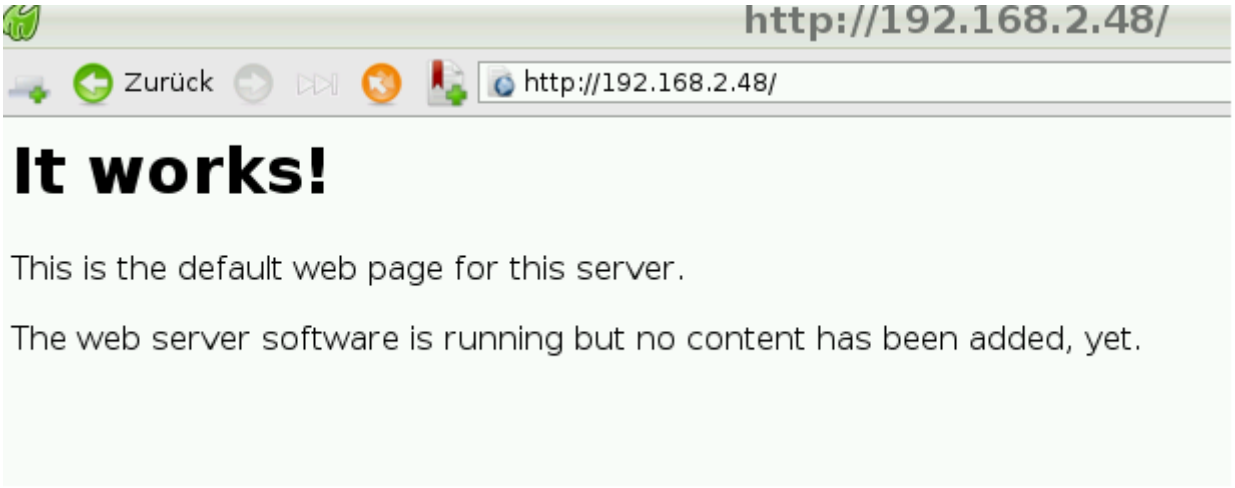


Abbildung 18: Screenshot 5

5. Installation PHP 5

- PHP 5 benötigt für erfolgreich laufenden Webserver
- Installation PHP 5 („*sudo apt-get install php5*“)
- optionale Pakete installieren (“*sudo apt-get install libapache2-mod-php5 libapache2-mod-perl2 php5 php5-cli php5-common php5-curl php5-dev php5-gd php5-imap php5-ldap php5-mhash php5-mysql php5-odbc php-pear php-apc*”)
- im Verzeichnis des Webservers PHP-Testdatei erstellen (/var/www/test)

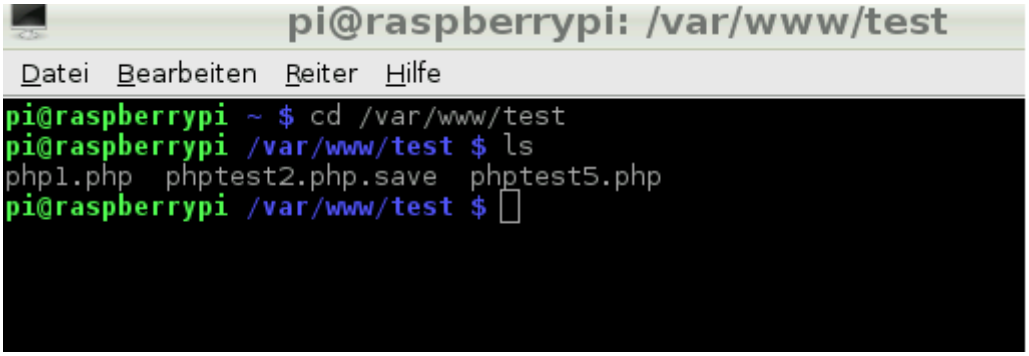


Abbildung 19: Screenshot 6

- Testdatei „phpinfo6.php“ erstellen und testen

Abbildung 20: Screenshot 7

7.1.3 Implementierungsaufwand

Die gesamte Implementierung sowohl des Servers mit allen benötigten Diensten als auch der App als Client sollte aufgrund der oben stehenden Administratoranleitung und der relativ simplen Handhabung unserer Android-App nicht länger als 20 Stunden dauern.

7.2 Benutzeranleitung

Die Dauer des Aktualisierungsintervalls lässt sich durch das Ändern des Befehls

„header("refresh: 10;");“

Anfang der Skripte test4B.php und logger2.php anpassen. Hierbei muss die Zahl „10“ durch die gewünschte Dauer (in Sekunden) bei beiden Dateien ausgetauscht werden.

Es lässt sich außerdem noch bestimmen, ob der übergeordnete Ordner von test4B.php mit in die Zählung der Dateien einbezogen werden soll oder nicht. Wenn man den übergeordneten Ordner nicht mit einbeziehen will, entfernt man die Zwei Schrägstriche "//" in Zeile 64 und 65 und setzt diese stattdessen in Zeile 70 und 71.

```
63   // Aufruf der Funktion
64   //$dir=".";
65   //$atest=count_file($dir);
66   //echo"<br />DIRECTORIES: ".$atest["directory"];
67   //echo"<br />FILES: ".$atest["file"];
68   //echo"<br />";
69
70   $dir="../";
71   $atest=count_file($dir);
72  //  echo"\\n DIRECTORIES: ".$atest["directory"];
73  //  echo"\\n FILES: ".$atest["file"];
74
```

Abbildung 21: Screenshot 8

Sollte test4B.php beispielsweise im Verzeichnis ../htdocs/www liegen, lässt sich somit bestimmen ob der Ordner htdocs ebenfalls durchforstet werden, oder ob die Zählung im Ordner www beginnen soll.

Um das gesamte Programm und damit zugleich unseren Projekterfolg testen zu können, kopiert man die PHP-Datei „test4B.php“ auf den Rasberry-Pi in das Homeverzeichnis des Apache Webservers. Dieser war in unserem Fall „ var/www/test“.

Das andere Skript, welche Veränderungen auf dem Server ausliest („logger2.php“) wird auf einen Webserver geladen und kann dann anschließend von jedem Gerät aus dem Internet bedient werden.
(Für unser Projekt haben wir den von der Hochschule zur Verfügung gestellten Webserver http://www08.mg.hs-niederrhein.de/studenten/studentwi/bwi50205/p/Ra836531/adm/ verwendet.)

Im folgenden Screenshot ist die Ausgabe des PHP-Skripts „test4B.php“ zu sehen.

Abbildung 22: Screenshot 9

7.3 Administratoranleitung

Eine separate Administrationsanleitung ist aufgrund der in der Benutzeranleitung definierten Schritte nicht notwendig.

7.4 Tabellen und Abbildungsverzeichnis

7.5 Abkürzungsverzeichnis

App	Applikation
ARIS	Architektur Integrierter Informationssysteme
bzw.	beziehungsweise
Dr. rer. nat.	Doktor der Naturwissenschaften
E-Mail	Elektronische Mail
ER(M)	Entity Relationship (Model)
GB	Gigabyte

H	Stunde(n)
HDMI	High Definition Multimedia Interface
LAN	Local Area Network
max.	maximal
NAS	Network Attached Storage
NOOBS	New Out Of Box Software
PHP	Hypertext Reprocessor
Prof.	Professor
SDK	Software Development Kit
SMB	Samba
USB	Universal Serial Bus

7.6 Literatur und Internetverweise

Bücher:

Engelhardt, E.: Coole Projekte mit dem Raspberry Pi, München 2013.

Richardson, M.; Wallace S.: Raspberry Pi für Einsteiger, Köln 2013.

Schmidt, M.: Raspberry Pi – Einstieg Optimierung Projekte, 1. Auflage, München 2013.

Internetquellen:

http://www.selfphp.info/tipps_tricks/verzeichnisse/count_file.php (Stand 11.01.2014)

http://www.selfphp.de/kochbuch/kochbuch.php?code=45 Stand (10.01.2014)

http://www.forum-raspberrypi.de (Stand 13.01.2014)

http://kampis-elektroecke.de/?page_id=1626 (Stand 13.01.2014)

http://www.roboternetz.de/community/threads/58522-Kleiner-Webserver-auf-dem-Raspberry-Pi (Stand 13.01.2104)

7.7 Quellcode

Logger2.php

```
<?php
header("refresh:10;");
//logger.php
if (isset($_GET['text']))
        $text = $_GET['text'];
$Ausgabe="$text";
$fh=fopen("logger2.log","w");
fwrite($fh,$Ausgabe);
fclose($fh);
echo "$text";
?>
```

Test4B.php

```
<?PHP
header("refresh: 10;");

$datum = date("d.m.Y");
$uhrzeit = date("H:i:s");
// echo $datum," - ",$uhrzeit," Uhr";



 function count_file($dir)
  {

  // Ueberprueft ob das angegebene Verzeichnis existiert
  if(is_dir($dir))
   {
   $afile["directory"]=0;
   $afile["file"]=0;

   chdir($dir);
   $handle=opendir(".");
   while($file=readdir($handle))
    {

    // Handelt es sich bei $file um ein Unterverzeichnis?
    if(is_dir($file) && $file!="." && $file!="..")
     {

     // directory um eins erhoehen
     $afile["directory"]++;
```

```
        // rekursiver Aufruf mit aktuellem Verzeichnis
        $y=count_file($file);

        // Ergebnisse des rekursiven Aufrufs der Funktion zu
        // directory und file aufaddieren
        $afile["directory"]+=$y["directory"];
        $afile["file"]+=$y["file"];
        }

       // Handelt es sich bei $file um eine Datei?
       if(is_file($file))
        {

        // file um eins erhoehen
        $afile["file"]++;
        }
       }

     // Verzeichnisstruktur zurueck gehen
     if(stristr($dir,"../"))
chdir($_SERVER["DOCUMENT_ROOT"].substr($_SERVER["REQUEST_URI"],0,strrpos($_SERVER[
"REQUEST_URI"],"/")));
     elseif($dir!=".") chdir("../");

     closedir($handle);
     }

   return $afile;
   }

 // Aufruf der Funktion
 //$dir=".";
 //$atest=count_file($dir);
 //echo"<br />DIRECTORIES: ".$atest["directory"];
 //echo"<br />FILES: ".$atest["file"];
 //echo"<br />";

 $dir="../";
 $atest=count_file($dir);
//  echo"\\n DIRECTORIES: ".$atest["directory"];
//  echo"\\n FILES: ".$atest["file"];


 // quelle http://www.selfphp.info/tipps_tricks/verzeichnisse/count_file.php
 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////


function dir_size($dir, &$size, $recursive=TRUE) {
```

```
    $handle = @opendir($dir);

    if(!$handle)
        return false;

    while ($file = @readdir ($handle)) {
        if (eregi("^\.{1,2}$",$file)) {
            continue;
        }

        if(!$recursive && $dir != $dir.$file."/") {
            if(is_dir($dir.$file))
                continue;
        }

        if(is_dir($dir.$file)) {
            dir_size($dir.$file."/", $size, $recursive);
        } else {
            $size += filesize($dir.$file);
        }
    }

    @closedir($handle);

}


function sizeMath($size) {

    if($size >= 1000000) {
        $size = $size / 1000000;
        $size = sprintf("%01.4f", $size) . 'MegaByte';
      } else if ($size <1000) {
                        $size = $size;
        $size = sprintf("%01.4f", $size) . 'Byte';
                        } else {
        $size = $size / 1000;
        $size = sprintf("%01.4f", $size) . 'KiloByte';
            }

    return $size;

}

$size = 0;

dir_size("../", $size, True);
```

```
$sizeD = sizeMath($size);

// echo "\\n";
// echo $sizeD;

        // quelle http://www.selfphp.de/kochbuch/kochbuch.php?code=45
 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

$text = $datum . "-" . $uhrzeit . "Uhr" . "\\nDIRECTORIES:" . $atest["directory"] . "\\nFILES:" .
$atest["file"] . "\\n" . $sizeD;

$server = "http://www08.mg.hs-
niederrhein.de/studenten/studentwi/bwi50205/p/Ra836531/logger2.php?text=$text";

 $fp = fopen ( $server, 'r' );
 $string = fread ( $fp,500);
 fclose ( $fp );
echo $string;

?>
```

7.8 Hausarbeit Scrum