# Inhaltsverzeichnis

# 1) Einleitung

## 1.1) Projektbeschreibung / Aufgabenstellung (Niclas)

## 1.2) Projektumfeld (Janfi)

# 2) Projektplanung

## 2.1) Projektphasen (Janfi)

## 2.2) Projektressourcen

### 2.2.1) ESP32-Microcontroller (Niclas)

### 2.2.2) Waveshare E-Paper-Display (Niclas)

## 2.3) Gemeinsames Arbeiten

### 2.3.1) Versionsverwaltung über GitHub (Janfi)

### 2.3.2) Backlogverwaltung über Google Sheets (Niclas)

# 3) Analysephase

## 3.1) Beschreibung des c‘t-Programms (Janfi)

## 3.2) Beschreibung des c‘t-Webservers (Janfi)

## 3.3) Anforderungsanalyse (Janfi)

# 4) Entwurfsphase

## 4.1) Entwurf des Anzeigebilds (Niclas)

## 4.2) Entwurf des Webinterfaces (Janfi)

## 4.3) Entwurf des Netzwerks (Niclas)

# 5) Implementierungsphase

## 5.1) PHP-Skript zur Anzeige eines Bildes (Niclas)

## 5.2) Hinzufügen von Texten über Webinterface (Janfi)

# 6) Fazit

## 6.1) Soll/Ist-Vergleich (Janfi)

## 6.2) Ausblick (Niclas)

# 7) Anhang

# 8) Quellen

# Einleitung

## Projektumfeld

Dieses Projekt ist Teil des Moduls „Projektarbeit“ im Bachelor-Studiengang „Technische Informatik“ an der Hochschule Ostwestfalen-Lippe in Lemgo. Bei diesem Modul handelt es sich um ein Pflichtmodul des vierten Semesters, welches von den Studierenden fordert die Inhalte ihres bisherigen Studienverlaufs in einem gewählten Themengebiet anzuwenden und zu vertiefen. Dabei steht vor allem eine praxisnahe und konzentrierte Bearbeitung der gestellten Aufgabenstellung im Vordergrund.

Dieses Projekt ist in der Projektgruppe von Professor Doktor Thomas Korte entstanden, welcher Projekte für ein bis zwei Personen zum Thema „Physical Computing“ anbietet. So wurde ein zweites Projekt, mit einem ähnlichen Thema angeboten, welches von Mathis Mohr bearbeitet wurde und sich mit der Programmierung des Türschildes mit einem anderen Microcontroller auseinandersetzte. Des Weiteren wurde ein Projekt für die Entwicklung einer Todo-Liste für den Schreibtisch von Ka Yung Cheng umgesetzt, welches eine kleinere Version des hier genutzten e-Paper Displays verwendet hat.

Eine weitere Schnittstelle dieses Projekt ist, wie in Abschnitt „Projektbeschreibung“ bereits angesprochen, das Projekt aus der Computerzeitschrift c’t (Ausgabe 02/2018), welches eine Grundlage für die Ansteuerung / Darstellung des Türschilds mit Hilfe des ESP32-Controllers bildet. Dieses Projekt setzt selbst auf der sogenannten „Basecamp“-Bibliothek, welche von der c’t entwickelt wurde und genutzt wird, um auf leichte und modifizierbare Art und Weise aus Microcontrollern IoT-Geräte zu machen, auf. Die letzte Bibliothek, die von diesem Projekt verwendet wird, ist die sogenannte „Espressif“-Bibliothek, welche dazu verwendet wird, den Sparkfun ESP32 Thing über die Arduino-DIE verwenden zu können.

# Projektplanung

## Projektphasen

## Gemeinsames Arbeiten

### Versionsverwaltung über GitHub

Da bei Softwareprojekten an denen mehr als eine Person beteiligt ist, sehr schnell Probleme bezüglich der Versionsverwaltung auftreten, haben wir uns entschieden ein Tool für die Versionsverwaltung zu verwenden. Unsere Entscheidung fiel auf Grund von persönlichen Erfahrungen auf GitHub welches von der GitHub Inc. entwickelt wird, welche im Laufe des Projekts von Microsoft aufgekauft wurden.

Der Name GitHub leitet sich von dem Versionsverwaltungssystem Git ab, für welches GitHub eine Online-Plattform mit den Aspekten der sozialen Medien verknüpft. So kann jeder Nutzer hier eigene, sogenannte Repositories *(zu Deutsch: Ablage / Archiv)* erstellen um seine Projekte zu verwalten. Dabei können die einzelnen Projekte sehr einfach mit anderen geteilt oder von anderen favorisiert werden. Dieser soziale Aspekt erleichtert das Finden von interessanten Projekten, an welchen man mitarbeiten kann. Außerdem behält man so stets den Überblick über die Projekte von befreundeten Entwicklern und kann diesen bei eventuellen Problemen leicht aushelfen.

Dabei verwendet GitHub die, oben bereits genannten, Repositories. Dabei handelt es sich um eine Art Ordner, welcher alle Dateien eines bestimmten Software-Projekts zentriert sammelt und immer einem Benutzerkonto zugeordnet ist. Dieser Ordner kann nun lokal kopiert werden um Dateien hinzuzufügen oder zu ändern. Sobald Änderungen vorgenommen wurden, kann der Nutzer diese „committen“. Dabei werden seine Änderungen mit dem aktuellen Stand auf dem Server verglichen und danach dort hochgeladen. Jedem Commit wird eine bestimmte Beschreibung und eine eindeutige Nummer zugeordnet, was dafür sorgt, dass jeder Commit als eine Version der Software gesehen werden kann. GitHub gibt dem Entwickler nun die Möglichkeit, bei Problemen oder Bugs alte Versionen der Software wieder zu laden um mit diesen weiterzuarbeiten.

Wenn nun mehrere Entwickler an einem Projekt arbeiten wollen, so muss der Besitzer des Repositories die anderen einladen, bevor sie als sogenannte „Contributors“ *(zu Deutsch: Mitwirkender)* den vollen Zugriff auf den gesammelten Quellcode haben. Um allen Entwicklern die Möglichkeit zu geben, nicht ständig die Arbeit der anderen während der Entwicklung durch regelmäßige Commits zu behindern, können sogenannte „Branches“ (zu Deutsch: Äste) angelegt werden. Diese erlauben die parallele Entwicklung ab einem bestimmten Startpunkt, bis zu welchem alle Branches auf dem gleichen Stand sind. Nach der Trennung kann ein Entwickler seinen Ast bearbeiten, bis er sein gewünschtes Feature umgesetzt hat. Daraufhin kann er seinen Ast wieder mit dem Hauptast (hier: master, häufig aber auch „Trunk“ genannt) verknüpfen, um allen anderen Entwicklern sein Ergebnis zur Verfügung zu stellen.

Wenn jedoch zwei Entwickler so zeitnah Änderungen an der gleichen Datei vornehmen, dass sich beide Änderungen überschneiden würden, so gilt das Prinzip „Wer zuerst kommt, malt zuerst“. Das heißt der Entwickler, welcher zuerst committed, kann seine Änderungen problemlos hochladen. Wenn der andere Entwickler dann seine Ergebnisse committen möchte, so wird er von GitHub darauf hingewiesen, dass seine Datei von der Server-Datei abweicht. Dann muss ein manuelles Abgleichen der Änderungen geschehen, um zu entscheiden, wie beide Dateien verknüpft werden sollen.

# Analysephase

Da diese Projektarbeit auf einem, bereits abgeschlossenem, Projekt der Computerzeitschrift c’t basiert, wurde dieser Quellcode in der Analysephase analysiert und nachvollzogen, um auf optimale Art und Weise darauf aufsetzen zu können.

## Beschreibung des c’t-Programms

## Beschreibung des c’t-Webservers

## Anforderungsanalyse

# Entwurfsphase

## Entwurf des Webinterface

Bei der Entwicklung des Webinterface war anfangs gar nicht klar, wie die eigentliche Implementierung aussehen sollte. Auch die genutzte Programmiersprache und die Software standen noch nicht fest. So wurden erst einmal die Anforderungen an die Schnittstelle zum Webserver formuliert:

* Unabhängig vom Benutzersystem
* Zugriff von mehr als einem Rechner / Gerät
* Umsetzbar in C oder Java
* Möglichkeit CSV-Dateien zu bearbeiten

Innerhalb des Teams wurden dann mehrere mögliche Umsetzungen gesammelt und ihre Vor- und Nachteile abgeglichen:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Möglichkeit | Unabhängig? | Mehrfachzugriff? | C oder Java? | CSV-Dateien |
| HTML Webinterface | Ja, nur Browser benötigt | Ja, wenn Internetzugriff eingerichtet | HTML, JavaScript und PHP | Nicht direkt über HTML aber mit PHP |
| Java Dektopanwendung | Jein, benötigt JRE | Nein, nur Zugriff von einem Rechner auf lokale Dateien | Java | Ja |

Wie man der obigen Tabelle entnehmen kann, bietet die Entwicklung eines Webinterfaces mit HTML, Javascript und PHP die Umsetzung der meisten geforderten Aspekte, weshalb dieser Ansatz verfolgt wurde. Bevor jedoch mit der eigentlichen Programmierung und Implementierung begonnen wurde, wurde ein Entwurf des Interface erstellt, welcher als Vorlage für die nachfolgende Umsetzung verwendet wurde. Dieser wurde, nach ein paar ersten Tests, iterativ überarbeitet und den Anforderungen noch besser angepasst.

So besteht der erste Entwurf aus zwei Hälften, welche die zwei Fragestellungen repräsentieren, welche der Anwender beantworten kann um seine Eingabe zu tätigen. Im ersten Teil soll der Anwender seinen aktuellen Zustand auswählen. Dabei hat er per RadioButton die Auswahl zwischen fünf unterschiedlichen Zuständen, welche mal mehr, mal weniger häufig Verwendung finden. Die fünf Zuständen handelt es sich um anwesend, abwesend, krank, im Urlaub und in der Pause.

Nachdem der Anwender sich für einen dieser fünf Zustände entschieden hat, kann im unteren Bereich der Seite in einem Textfeld eine Meldung eingegeben werden, welche dann, bei der nächsten Aktualisierung auf dem Türschild angezeigt werden soll. Zur Unterstützung des Anwenders werden die zwei aktuell angezeigten Meldungen, mitsamt dem Datum und der Uhrzeit der Erstellung, unter dem Textfeld angezeigt. Diese Anordnung verhindert, dass der Nutzer manuell auf seinem Türschild nachsehen muss um zu wissen, was die aktuellen Meldungen sind. Zu guter Letzt finden sich auf dem Entwurf noch zwei reguläre Buttons, welche einmal die Änderungen verwerfen („Abbrechen“) oder die Änderungen bestätigen und die Hintergrundprozesse starten können.

Der zweite Entwurf führt kleine, aber sinnvolle Verbesserungen am bisherigen Design ein. So wird ein DropDown-Menü hinzugefügt, welches in der oberen rechten Ecke des Bildschirms angeordnet ist und die Auswahl zwischen mehreren Professoren bzw. Anwendern des Türschilds ermöglicht. Über diesen Umstand können mehrere Anwender das gleiche Webinterface zur Einstellung ihrer jeweiligen Türschilder nutzen. Die anderen Änderungen sind mehrere kleine Textfelder, welche hinzugefügt werden um dem Anwender eine Hilfestellung bei der Bedienung des Webinterface zu bieten.

# Implementierungsphase

## Hinzufügen von Texten über ein Webinterface

# Fazit

## Soll-/Ist-Vergleich