# 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 CT-Programms (Janfi)

## 3.2) Beschreibung des CT-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) Quellen

# 8) Anhang

## 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.