# 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

# Einleitung

## Projektbeschreibung / Aufgabenstellung

Im Rahmen der Projektarbeit sollte ein drahtloses Türschild mit einem E-Paper-Display auf einem „SparkFun ESP32 Thing“ programmiert werden. Es soll ein Waveshare 7.5inch e-Paper HAT als Anzeige benutzt werden. Der Bildschirm soll nicht einfach nur ein statisches Bild anzeigen, sondern es soll auch einen Ankündigungstext, ähnlich dem Info-Monitor auf der 4. Etage, angezeigt werden. Dieser Ankündigungstext soll sich über WLAN jederzeit und schnell ändern lassen. Sowohl der ESP32, als auch der E-Paper Display sollen über eine Batterie betrieben werden und müssen dementsprechend stromsparend betrieben werden. Alle angesprochenen Einzelteile sollen sicher an der Wand vor dem Raum 1.365 befestigt werden. Zur mit einer Wandhalterung, welche mit einem 3D-Drucker hergestellt werden soll.

Als Referenz konnten wir den Artikel „Ausdauernde Infotafel“ aus der Zeitschrift c’t im Helft 2/2018 nutzen. Diesem lag ein Türschildprogramm bei, welches sich gut auf unser Projekt anwenden lies. Dieses musste jedoch noch personalisiert und erweitert werden um unsere Aufgabenstellung zu erfüllen.

# Projektplanung

## Projektressourcen

### ESP32-Microcontroller

Für unser Projekt brauchten wir einen Microcontroller, der Wifi Funktionalitäten bietet. Außerdem muss er batteriebetrieben arbeiten können. Das Programm der c’t an dem wir uns orientiert hatten nutzte den ESP32. Der ESP32 „Thing“ von der Firma SparkFun ist ein Microcontroller aus der gleichen Familie, wie der, welcher von der c’t benutzt wurde und erfüllt alle erweiterten Anforderungen, die wir zusätzlich zur c’t hatten.

Der Controller besitzt einen integrierten 802.11 BGN WiFi Transceiver, mit dem die geforderten Wifi Funktionalitäten problemlos abgedeckt werden können. Der ESP32 kann seinen eigenen Wifi Access Point aufbauen, über den man eine Verbindung direkt zum Controller herstellen kann, als auch sich mit anderen Routern verbinden, wodurch alle Geräte im gleichen Netzwerk sich mit dem Controller verbinden können.

Durch einen integrierten „LiPo Battery Charger“ ist der langfristige Batteriebetrieb möglich, vor allem da sich der Controller die meiste Zeit im „Deep Sleep-Modus“ befindet, in welchem er nur auf einer Stromstärke von 2.5µA läuft.

Das einzige Problem, welches wir mit dem ESP32 „Thing“ hatten, war der zu geringe Arbeitsspeicher. Mit 520kB internem SRAM, besitzt der Controller eigentlich sogar recht viel Speicher im vergleich zu anderen Controllern seiner Größe, jedoch sind es leider etwas zu wenig um all die Daten zu speichern, die das Waveshare Display braucht um Farbe anzuzeigen. Das c’t Programm selbst war auf ein schwarz-weißes Display ausgelegt und das hinzufügen einer weiteren Farbe verdoppelt die Anzahl an Daten, die an das Waveshare Display gesendet werden müssen, mehr dazu in der Beschreibung des Waveshare Displays. Hier muss noch hin, wie die Sache mit dem paged print ausgegangen ist.

### Waveshare E-Paper-Display

Zur Anzeige des Türschilds brauchten wir ein Display, welches Stromsparend arbeiten kann. E-Paper Displays erfüllen dieses Kriterium, da sie nur beim erstmaligen „zeichnen“ eines Bildes Strom verbrauchen und dieses Bild ohne weiteren Stromverbrauch „halten“ können.

Der “7.5inch e-Paper HAT” der Firma Waveshare benutzt die Anzeigetechnik Microencapsulated Electrophoretic Display (MED). Diese Technologie lässt Micro kapseln gefüllt mit elektrisch geladenen weiße Partikel in gefärbten Öl schweben. Für jeden einzelne Kapsel wird elektronisch festgelegt, ob die weißen Partikel oben oder unten in der Kapsel schweben sollen, so dass der Betrachter des Displays entweder die weißen Partikel oder das farbige Öl wahrnimmt. Sobald dieser Zustand einmal festgelegt ist, ändert sich der Zustand nicht von selbst, weswegen das angezeigte Bild für eine lange Zeit angezeigt werden kann, ohne weiter Strom zu benötigen.

Für unser Projekt benutzen wir die Version des Displays, die neben schwarz und weiß noch eine Farbe anzeigen kann. Die dritte Farbe ist normalerweise Rot, weswegen oft zwischen Schwarz-Weiß und Schwarz-Weiß-Rot unterschieden wird. Unser Display kann jedoch gelb als dritte Farbe, jedoch wird Gelb genauso angesprochen, wie Rot.

## Gemeinsames Arbeiten

### Backlogverwaltung über Google Sheets

Neben GitHub für die Versionsverwaltung, wollten wir noch einen Überblick darüber haben, was wir bereits erledigt haben, und was noch zu tun ist. Wir haben uns für eine tabellarische Übersicht entscheidend, welche wir dann über Googles Cloudbasiertes Tabellenbearbeitungsprogramm „Google Sheets“ umgesetzt haben.

Wir haben unsere Backlogs in vier Oberkategorien unterteilt: Netzwerk, Hardware, Software und Dokumentation. Jede der Oberkategorien hat zur besseren Übersicht eine eigene Tabelle bekommen, in welcher wiederrum mehrere einzelne Backlogs verwaltet werden.

Ein Backlog besteht aus mehreren Punkten:

* Problem: Name des Backlogs in wenigen Worten zusammengefasst.
* Kategorie: Kategorie des Backlogs. Dient der Übersichtlichkeit.
* Beschreibung: Ausführlichere Beschreibung des Backlogs.
* Aufwand: Geschätzter Aufwand, der für die Bearbeitung benötigt wird
* Kommentar: Platz für Kommentare.
* Status: Momentane Status de Backlogs
* Bearbeiter: Wer diesen Backlog bearbeitet

Der Status wird basierend auf den Eingaben in den anderen Feldern des Backlogs berechnet. Sobald etwas in der Spalte „Problem“ eingetragen wird, wird der Status auf „erstellt“ gesetzt. Der Status wird automatisch auf „zugewiesen“ gesetzt, wenn das Feld „Bearbeiter“ gefüllt wird. Zwei Checkboxen sollen nun die übrigen Status verwalten. Der erste Haken soll gesetzt werden, wenn der Bearbeiter die Bearbeitung abgeschlossen hat. Dieser setzt den Status auf “bearbeitet“. Nach einer Überprüfung durch die zweite Person, darf diese den zweiten haken setzen, und somit den Status auf „erledigt setzen.

Die Backlogverwaltung auf diese Art hat sehr gut funktioniert. Da Google Sheets cloudbasiert ist und auf die gleichzeitige Bearbeitung durch mehrere Leute ausgelegt ist, hatten wir nie Probleme mit verlorengegangenen Änderungen und konnten zu jeder Zeit und Ortsunabhängig auf die Backlogs zugreifen. Das Design war übersichtlich genug, damit man problemlos überblicken konnte, was noch zu tun war und wie viel schon erledigt wurde.

# Entwurfsphase

## Entwurf des Anzeigebilds

Für unseren ersten Entwurf des Anzeigebilds wollten wir uns an dem bereits bestehenden nicht-digitalen Türschild orientieren. Da dieses auch nur aus den Farben Schwarz, Weiß und Gelb besteht, war dieses auch ohne weiteres möglich. Mit dem GUI prototyping Programm „Pencil“ erstellten wir unseren ersten Entwurf.

Wir wollten das Türschild in drei Bereiche Aufteilen. Etwas Platz am oberen Ende nach dem Vorbild des bestehenden Türschilds, einen Infomonitor in der Mitte und einen Bereich am unteren Ende für den Anwesenheitsstatus und Informationen über den Professor. Wie in dem bestehenden Türschild, wollten wir am oberen Ende die Raumnummer durch einen gelben Strich vom Rest des Türschilds getrennt haben. Anstatt des Fachbereiches wollten wir oberhalb der Linie auch noch den Raumnamen ausgeben um mehr Platz für den Infomonitor zu haben. Am unteren Ende des E-Papers sollen Infos zum Professor an der linken Seite angezeigt werden. Der Name, die E-Mail-Adresse und die Telefonnummer waren unsere ersten Ideen für Professor-spezifische Informationen. Gegenüber den Professor-Informationen soll der aktuelle Anwesenheitsstatus angezeigt werden. Nachdem genug Platz für den oberen und unteren Bereich geplant haben, war der bleibende Bereich für den Infomonitor vorgesehen.

Für unseren Entwurf haben wir die Informationen von Professor Korte eingetragen und ein paar Beispielmeldungen für den Infomonitor angegeben, sowie einen beispielhaften Anwesenheitsstatus.

## Entwurf des Netzwerks

In dem Netzwerk was wir uns bei unserem ersten Entwurf vorgestellt haben sollte Orts- und Geräteunabhängig zu jeder Zeit Informationen, wie der Anwesenheitsstatus und Nachrichten für den Infomonitor aktualisiert werden. Die aktualisierten sollen so gespeichert werden, dass die PHP-Skripte das aktualisierte Türschild „zeichnen“ können, wenn der ESP32 auf den Server zugreift.

Da des E-Paper direkt über ein Kabel mit dem ESP32 verbunden ist, war unsere Erste Hürde die Frage, wie wir des ESP32 mit dem PHP-Server verbinden Das Programm der c’t hat die Netzwerkfunktionalitäten über die „Basecamp“-Library gelöst. Hier noch mal mit Jani klären, wie viel im Bereich c’t Programmerklärung schon geschrieben wurde.

Da wir nicht über unsere eigene Weboberfläche in der Basecamp Weboberfläche Werte ändern können. Mussten wir das aktualisieren des Bildes unabhängig von den Basecamp Funktionalitäten designen. Da Basecamp auf eine spezifizierte .php-Datei zugreift, müssen wir diesen Pfad statisch belassen, jedoch können wir ändern, was in der Datei steht. Da die Anzeigebilder durch PHP-Skripte „gezeichnet“ werden und diese Skripte wiederrum auf csv-Dateien zugreife, soll die Weboberfläche die csv-Dateien verändern, so dass beim neuen „zeichnen“ die aktualisierten Werte genutzt werden.

# Implementierungsphase

## PHP-Skript zur Anzeige eines Bildes

# Fazit

## Ausblick