

Hochschule Ostwestfalen-Lippe

University of Applied Sciences

Drahtloses Türschild mit E-Paper-Display

Projektarbeit

Autoren:

Niclas Muss (15365020) &
Jan-Philipp Töberg (15363063)

Studiengang:
Technische Informatik

Projektbetreuer:
Prof. Dr. Thomas Korte

Abgabedatum:
02.07.2018

Inhaltsverzeichnis

Inhaltsverzeichnis	I
Tabellenverzeichnis	II
Abbildungsverzeichnis	II
1. Einleitung	1
1.1 Aufgabenstellung (Muss)	1
1.2 Projektbeschreibung (Muss)	1
1.3 Projektumfeld (Töberg)	1
2. Projektplanung	2
2.1 Projektphasen (Töberg)	2
2.2 Projekttressourcen (Muss)	2
2.2.1 ESP32-Microcontroller	2
2.2.2 Waveshare E-Paper-Display	3
2.3 Gemeinsames Arbeiten	3
2.3.1 Versionsverwaltung über GitHub (Töberg)	3
2.3.2 Backlogverwaltung über Google Sheets (Muss)	4
3. Analysephase (Töberg)	5
3.1 Der c't-Micro-Controller	5
3.2 Der c't-Webserver	6
3.3 Anforderungsanalyse	7
4. Entwurfsphase	7
4.1 Entwurf des Anzeigebilds (Muss)	7
4.2 Entwurf des Webinterface (Töberg)	8
4.3 Entwurf des Netzwerks (Muss)	10
5. Implementierungsphase	11
5.1 Bildanzeige über PHP (Muss)	11
5.1.1 index.php	11
5.1.2 türschild.php	12
5.2 Das Webinterface (Töberg)	15
5.2.1 HTML und JavaScript auf Client-Seite	16
5.2.2 PHP auf Server-Seite	17
6. Fazit	18
6.1 Soll-/Ist-Vergleich (Töberg)	18
6.2 Ausblick (Muss)	18

7. Quellen	20
A Anhang.....	A
A1 – Netzwerkdiagramme.....	A
A2 – Das Webinterface.....	A
A3 – Fotos der Türschilder (vorher)	B
A4 – Fotos der Türschilder (nachher)	C
A5 – Abkürzungsverzeichnis.....	D
A6 – QR-Code zum GitHub-Repository.....	D
A7 – Detaillierte Ansicht der Backlogverwaltung.....	E
A8 – Artikel aus der c’t	F
A9 – Die „index.php“-Datei	I
A10 – Die „turschild_V1.php“-Datei.....	L
A11 – Die „webinterface.html“-Datei	O
A12 – Die “getDataFromCSV.php“-Datei.....	R
A13 – Die “saveDataToCSV.php“-Datei.....	S
A13 – Die Anschlüsse des SparkFun ESP32 Thing.....	U

Tabellenverzeichnis

Tabelle 1 - Vergleich der Webserver-Schnittstellen.....	9
Tabelle 2 - Aufbau der professor.csv mit zwei Beispieleintragungen	19

Abbildungsverzeichnis

Abbildung 1 - Überblick über die Backlogverwaltung (detaillierte Ansicht im Anhang A7).....	5
Abbildung 2 - Entwurf des Anzeigebildes (Version 01)	8
Abbildung 3 - Entwurf des Webinterface (Version 01)	9
Abbildung 4 - Entwurf des Webinterface (Version 02)	10
Abbildung 5 - Veranschaulichung des Zusammenhangs von csv-Datei und Anzeige	10
Abbildung 6 - Entwicklung des Anzeigebildes über den Projektzeitraum.....	13
Abbildung 7 - Kommunikation zwischen Server und Client	16
Abbildung 8 - Entwurf des Netzwerks (Version 01)	A
Abbildung 9 - Entwurf des Netzwerks (Version 02)	A
Abbildung 10 - Das aktuelle Webinterface (Version 02)	A
Abbildung 11 - Türschild von Prof. Hausdörfer (vorher)	B
Abbildung 12 - Türschild von Prof. Korte (vorher)	B
Abbildung 13 - Türschild von Prof. Hausdörfer (nachher)	C
Abbildung 14 - Türschild von Prof. Korte (nachher).....	C
Abbildung 15 - QR-Code für GitHub	D
Abbildung 16 - komplette Backlogverwaltung (Stand: 27.06.2018)	E
Abbildung 17 - Detailansicht der Backlogverwaltung	E

1. Einleitung

1.1 Aufgabenstellung (Muss)

Im Heft 2/2018 der Zeitschrift c't wird unter dem Titel Ausdauernde Infotafel der Aufbau und Programmierung eines stromsparenden drahtlosen Türschildes mit E-Paper-Display beschrieben. So ein Türschild soll für das Büro im Raum 1.365 aufgebaut und mit erweiterten Funktionsumfang programmiert werden. So soll es neben der Übertragung von Bildern auch möglich sein, schnell einen Ankündigungstext - so wie auf dem Info-Monitor auf der 4. Etage - aus dem Büro heraus auf das Türschild via WLAN zu übertragen. Es muss dazu nicht nur ein Mikrocontroller und ein Web Server programmiert, sondern auch eine sichere Befestigung für Display, Controller und Batterien konstruiert werden. In der c't ist dazu ein Beispiel angegeben. Dies ist ein Team-Projekt für zwei Personen. Alle benötigten Komponenten bis auf die Befestigungsmaterialien sind bereits vorhanden.

1.2 Projektbeschreibung (Muss)

Im Rahmen der Projektarbeit im Sommersemester 2018 soll das in der Aufgabenstellung beschriebene Türschild implementiert werden. Dafür wurde mehrere Wochen lang mit dem Mikrocontroller „SparkFun ESP32 Thing“ und dem „Waveshare 7.5inch e-Paper HAT“ gearbeitet.

Das Ziel des Projekts war es, alle in der Aufgabenstellung genannten Anforderungen zu erfüllen und, wenn möglich, noch einige weitere Features hinzuzufügen. Neben dem geforderten Ankündigungstext soll auch ein Anwesenheitsstatus und einige Informationen über den Professor, welchem das Büro gehört, angezeigt werden. Sowohl der ESP32, als auch der E-Paper Display sollen außerdem 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 mit einer Wandhalterung, welche mit einem 3D-Drucker hergestellt werden soll.

Als Referenz wurde der Artikel „Auszdauernde Infotafel“ aus der Zeitschrift c't im Heft 2/2018 benutzt, das dort vorgegebene Projekt ließ sich gut für die Ansprüche des Projekts erweitern.

1.3 Projektumfeld (Töberg)

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.¹

Dieses Projekt ist in der Projektgruppe von Professor Doktor Thomas Korte entstanden, welcher mehrere Projekte für ein bis zwei Personen zum Thema „Physical Computing“ anbot. 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 „1.2 Projektbeschreibung“ bereits angesprochen, das Projekt aus der Computerzeitschrift c't (Ausgabe 02/2018), welches eine Grundlage für die Ansteuerung / Darstellung des Türschildes mit Hilfe des ESP32-Controllers bildet².

¹ Siehe Quelle #1

² Siehe Quelle #2

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, auf³. 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-IDE verwenden zu können⁴.

2. Projektplanung

2.1 Projektphasen (Töberg)

Die einzelnen Abschnitte des Projekts lassen sich in vier unterschiedliche Phasen einteilen: Planung, Analyse, Entwurf und die Umsetzung. Begonnen wurde dabei logischerweise mit der Planungsphase, in welcher das weitere Vorgehen während des Projektes überlegt und geplant wurde. Dabei wurde sich einmal mit der Hardware vertraut gemacht, aber auch Lösungen für die Probleme des Versionsmanagements und der Aufgabenverfolgung gesucht & gefunden.

In der darauffolgenden Analysephase wurde sich intensiv mit dem Programm der Zeitschrift c't auseinandergesetzt. Dabei wurde dessen Ablauf und grundlegende Struktur verinnerlicht, sowie notwendige Anpassungen vorgenommen. Insgesamt wurde während dieser Phase ein gutes Verständnis der möglichen Ansatzpunkte für dieses Projekt gesammelt, was das Entwerfen von eigenen, ergänzenden Funktionalitäten, vereinfacht.

Nachdem das grundlegende Programm analysiert und verstanden wurde, konnte sich mit dem Entwurf der neuen Möglichkeiten, welche das Projekt in Zukunft bieten soll, auseinandergesetzt werden. Dabei wurden Prototypen der graphischen Oberflächen angefertigt und auch im Nachhinein iterativ überarbeitet.

Zu guter Letzt wurden die geplanten Funktionen gemäß den Entwürfen umgesetzt. Die Vorgehensweise und die daraus resultierenden Ergebnisse werden im Abschnitt „5. Implementierungsphase“ detaillierter beschrieben.

Insgesamt wurde mit einer angepassten Form des Wasserfallmodells gearbeitet, da, im Gegensatz zum herkömmlichen Wasserfallmodell, Prozessphasen mehrmals durchlaufen wurden. So wurden die Planungs- und die Entwurfsphase im Nachhinein noch angepasst, wenn Änderungen in der Implementierungsphase aufgetreten sind. Generell wurden die einzelnen Phasen aber erst komplett abgearbeitet, bevor mit der Arbeit in der nächsten Phase begonnen wurde.

2.2 Projektressourcen (Muss)

2.2.1 ESP32-Microcontroller

Für das Projekt wurde ein Microcontroller verwendet, der Wifi Funktionalitäten bietet und außerdem batteriebetrieben arbeiten kann. Das Programm der c't, welches als Grundlage dient, 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, auf dem ersten Blick, 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. Um diesen WiFi Access Point jedoch zielführend nutzend zu können, muss ein

³ Siehe Quelle #3

⁴ Siehe Quelle #4

Programm diesen öffnen und eine Benutzeroberfläche anbieten. Für die geforderten Zwecke konnte die Basecamp Library genutzt werden, welche auch von der c't genutzt wurde.

Durch einen integrierten „LiPo Battery Charger“ ist der langfristige Batteriebetrieb möglich, vor allem da sich der Controller, die meiste Zeit im Tiefschlafmodus befinden soll, in welchem er nur auf einer Stromstärke von 2.5 µA läuft. Wenn sich der Controller im Tiefschlafmodus befindet, gibt es mehrere Möglichkeiten ihn aufzuwecken, welche vorher definiert werden müssen. Unter anderem auch eine Möglichkeit, in welcher der Controller seine Wifi Funktionalitäten aufrechterhält und auf einen Wakeup-Call über sein Wifi-Modul reagieren kann, welche sich für das Projekt anbietet. Bei einem aktiven Betrieb kann der Battery Charger jedoch nicht mit dem Betriebsverbrauch mithalten. Leider konnte während der Projektarbeit nicht getestet werden, ob der Betrieb über eine Batterie mit dem endgültigen Programm funktionieren würde, da keine Batterie zur Verfügung gestanden hat.

Das einzige Problem, welches sich im Umgang mit dem ESP32 „Thing“ gezeigt hat, war der zu geringe Arbeitsspeicher. Mit 520 kB internem SRAM besitzt der Controller recht viel Speicher im Vergleich zu anderen Controllern seiner Größe, jedoch sind es leider zu wenig um all die Daten zu bearbeiten, 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. Dieses Problem ließ sich innerhalb des Projektzeitraums nicht lösen, da die vom Controller und der GD-Bibliothek unterstützte Möglichkeit einer paged-Speicherung und Anzeige nicht mit dem gegebenen Programm der c't kompatibel war und daher eine sehr aufwändige Änderung wäre, welche im Projektzeitraum nicht mehr implementiert werden konnte.

2.2.2 Waveshare E-Paper-Display

Zur Anzeige des Türschilds wurde ein Display, welches stromsparend arbeiten kann benötigt. E-Paper Displays erfüllen dieses Kriterium, da sie nur beim erstmaligen „Zeichnen“ eines Bildes Strom verbrauchen und dieses Bild ohne weiteren Stromverbrauch „halten“ können. Für den spezifischen Fall dieses Projekts bedeutet das, dass ein Türschild, welches einmalig auf das E-Paper Display gezeichnet wurde, von diesem ohne jeglichen weiteren Stromverbrauch bis zum nächsten Zeichnen des Türschildes angezeigt werden kann.

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 weißen Partikel in gefärbten Öl schweben. Für jede 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 das Projekt wird die Version des Displays benutzt, 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. Die dritte Farbe unsers Displays ist jedoch Gelb. Für die Software macht dies jedoch keinen Unterschied, da der Display genau die gleichen Werte übergeben bekommt, wie an einem Display mit Rot als dritter Farbe. Aus diesem Grund sind viele der Variablen innerhalb des Skriptes auf die Farbe Rot bezogen, obwohl die endgültige Ausgabe Gelb sein soll.

2.3 Gemeinsames Arbeiten

2.3.1 Versionsverwaltung über GitHub (Töberg)

Da bei Softwareprojekten, an denen mehr als eine Person beteiligt ist, sehr schnell Probleme bezüglich der Versionsverwaltung auftreten, wurde im Team die Entscheidung getroffen, ein Tool für

die Versionsverwaltung zu verwenden. Die 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.

2.3.2 Backlogverwaltung über Google Sheets (Muss)

Neben GitHub für die Versionsverwaltung sollte ein Überblick darüber existieren, was bereits erledigt wurde und was noch zu tun ist. Im Team wurde sich auf eine tabellarische Übersicht geeinigt, welche dann über Googles cloudbasiertes Tabellenbearbeitungsprogramm „Google Sheets“ umgesetzt wurde.

Die sogenannten Backlogs lassen sich in vier Oberkategorien unterteilen: 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.
Etwas weiter eingeschränkt als die Oberkategorie.
- Beschreibung:
ausführliche Beschreibung des Backlogs.
- Aufwand:
Geschätzter Aufwand, der für die Bearbeitung benötigt wird.
- Kommentar:
Platz für Kommentare. Um den Status etwas genauer zu beschreiben.
- Status: Momentaner Status des Backlogs (automatisch generiert).
- Bearbeiter: Name der Person, die diesen Backlog bearbeitet.

PROBLEME FÜR DIE ZUKUNFT								
NETZWERK PROBLEME								
ID	Kategorie	Beschreibung	Aufwand (min)	Kommentar	Status	Bearbeiter	Bearbeitet	Erledigt
1	Keine Verbindung in bestehende Netzwerke möglich	Verbindung Das Board kann sich wieder mit dem Eduroam noch mit Kortes Router verbinden	5	Erstellen	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	recht-anteriorer Server	Server Langfristig brauchen wir einen echten php Server	10	Bearbeitet	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	Webinterface braucht Internet	Server Das Webinterface muss auf lange Sicht über das Internet erreicht werden können	30	Erledigt	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	Netzwerk-Diagramm aktualisieren	Entwurf Das Netzwerk-Diagramm muss auf den neueren Stand gebracht werden	15	Erledigt	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5								
6								
7								
8								
9								
10								

HARDWARE PROBLEME								
SOFTWARE PROBLEME								
ID	Kategorie	Beschreibung	Aufwand (min)	Kommentar	Status	Bearbeiter	Bearbeitet	Erledigt
1	Farbe anzeigen	E-Röper Die Farbe lässt sich nicht über das onlinefüige Programm anzeigen	5	Erstellen	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	Flackern	E-Röper Überprüfen ob das Flackern umgehbar ist	10	Erstellen	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	Stromversorgung	Controller Der Controller wird über eine Batterie betrieben	0	Zugewiesen	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	Cope	E-Röper Es soll ein Gehäuse für das Display 3d gedruckt werden	5	Zugewiesen	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5								
6								
7								
8								
9								
10								

DOKUMENTATION								
ID	Kategorie	Beschreibung	Aufwand (min)	Kommentar	Status	Bearbeiter	Bearbeitet	Erledigt
1	Kolloquium vorbereiten	Präsentation Erstellen der Präsentationen für das Kolloquium am 05.07	10	Zugewiesen	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	Kapitel 1 schreiben	Ausarbeitung Projektbeschreibung / Aufgabenstellung	20	Zugewiesen	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	Kapitel 1-2 schreiben	Ausarbeitung Projektanfeld	45	Zugewiesen	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	Kapitel 2-4 schreiben	Ausarbeitung Projektphasen	60	Zugewiesen	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5	Kapitel 2-3 schreiben	Ausarbeitung Projektressourcen - ESR32-Mikrocontroller	50	Zugewiesen	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Abbildung 1 - Überblick über die Backlogverwaltung (detaillierte Ansicht im Anhang A7)

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 Vorgang 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 Statuspalte an sich ist jedoch zu eingeschränkt um alleine anzeigen zu können, wie der Backlog bearbeitet wurde oder wird. Zur beschreibbaren Erweiterung der Statuspalte soll die Beschreibung genutzt werden um, wenn nötig, Einzelheiten zu Bearbeitung des Backlogs angeben zu können.

Die Backlogverwaltung auf diese Art hat sehr gut funktioniert. Da Google Sheets cloudbasiert ist und auf die gleichzeitige Bearbeitung durch mehrere Leute ausgelegt ist, sind keine Probleme wie verlorengegangene Änderungen oder ähnliches aufgetreten und es konnte zu jeder Zeit auch ortsunabhängig auf die Backlogs zugegriffen werden. Das Design ist übersichtlich genug, damit man problemlos überblicken kann, was noch zu tun ist und wie viel schon erledigt wurde.

3. Analysephase (Töberg)

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

3.1 Der c't-Micro-Controller

Der grundlegende Teil der c't-Unterlagen, welcher in diesem Projekt auch nur minimal angepasst werden musste, besteht aus der „tuerschild.ino“-Datei. Dabei handelt es sich um eine

Quellcodedatei, welche in der Programmiersprache C++ in der Arduino-IDE geschrieben wurde und auf dem ESP32 Controller ausgeführt wird.

Das Programm beginnt dabei mit dem einbinden von unterschiedlichen Bibliotheken und definieren von diversen Konstanten. Dabei wird auch die Größe des verwendeten e-Papers festgelegt damit die dazugehörige Bibliothek des Herstellers geladen werden kann. Daraufhin wird die begin()-Methode gestartet, welche einmalig beim Start des Controllers ausgeführt wird und ein paar grundlegende Einstellungen vornimmt. So wird die Kommunikation mit dem Webserver über TCP/IP eingerichtet und getestet.

Daraufhin beginnt die loop()-Methode, welche von dem Controller in Endlosschleife ausgeführt wird, bis dieser ausgeschaltet wird oder ein neues Programm übertragen bekommt. Da wird zu Beginn einmal das aktuelle Bild vom Webserver angefragt und ausgedruckt bevor der Controller entweder in den Tiefschlaf übergeht oder, wenn er noch nicht im Produktionsmodus betrieben wird, eine vom Anwender definierte Zeit wartet. Nach dieser Wartezeit wird mit der Ausführung der loop()-Methode weitergemacht.

Zusätzlich zu diesen beiden Methoden, welche als grundlegende Methoden bei einem Großteil von Microcontrollern eingesetzt werden, besteht dieses Programm noch aus sieben, teilweise sehr kurzen, Methoden, welche für das Senden und Empfangen http-Anfragen über TCP/IP verantwortlich sind und einer Methoden, welche für das Darstellen auf dem e-Paper verantwortlich sind. Dieser Methode wird ein Array von Character-Variablen übergeben, welche die einzelnen Pixel des Displays darstellen. Bei der Auswertung dieses Arrays werden nun einzelne Bytes anstatt Bits ausgewertet und zu Beginn wird überprüft, ob das genutzte e-Paper nur zwei Farben (schwarz & weiß) oder noch eine dritte Farbe (rot oder gelb) anzeigen kann. Abhängig davon finden unterschiedliche Auswertungen statt.

Bei der ersten, zweifarbigen Auswertung wird jeder Pixel entsprechend der Stelle in jedem Byte geschaltet. So bedeutet eine 1, dass der Pixel schwarz und eine 0 das er weiß gefärbt wird. Bei der zweiten Abfrage gestaltet sich diese Abfrage nicht mehr so leicht, da eine dritte Möglichkeit eingeführt wird. So werden bei dieser Auswertung immer ein Bit und sein Vorgänger betrachtet. Wenn beide den Wert 1 besitzen, dann wird der Pixel schwarz. Wenn nur der Vorgänger den Wert 1 besitzt, so wird der Pixel rot (bzw. gelb, je nach E-Paper) geschaltet. Sollte keines der beiden Pixel den Wert 1 besitzen, so wird der Pixel weiß gefärbt. Im Gegensatz zu der anderen Auswertung werden hier auch immer zwei Bits in der Zählschleife weitergesprungen, da immer zwei Bits zum Übertragen der Farbinformation für einen Pixel gebraucht werden.

Wenn der Controller nun gestartet wird, so lässt sich dieser über ein eigenes Webinterface einrichten. Diese Funktionalität kommt aus der (in Abschnitt „1.3 Projektumfeld“ bereits erwähnten) Basecamp-Bibliothek. Dabei lässt sich einstellen, welchen Server der Controller anfragen soll, welche Datei und nach welcher Zeit der Controller neue Anfragen senden soll.

3.2 Der c't-Webserver

Mit dem c't-Programm mitgeliefert werden mehrere PHP-Skripte, welche die Funktionalität des Webservers abbilden. So besitzt jedes Bild, was das Türschild theoretisch anzeigen kann, ein eigenes PHP-Skript, welche jedoch alle von einem zentralen Skript, der „index.php“-Datei, gestartet werden.

Diese Datei beginnt mit dem Laden von mehreren Schriftarten, welche von den anderen Skripten danach verwendet werden können, und dem Einstellen der Displaygröße. Diese wird dabei aus den Parametern beim Aufruf der Datei ausgelesen. Über diese Parameter wird auch der Inhalt angegeben, der angezeigt werden soll. Dieser wird, nachdem das Skript überprüft, ob der Server eine bestimmte Erweiterung namens GD, welche als Bibliothek für graphische Bearbeitung fungiert,

geladen hat, in einen Dateinamen übersetzt und die zugehörige Datei wird eingebettet und ausgeführt.

Nach der Ausführung dieser Datei wird mit der „index.php“ weitergemacht. Diese übersetzt zu guter Letzt dass erzeugte PNG-Bild der anderen Datei in ein Array aus einzelnen Byte, welches dann an den Microcontroller gesendet wird, damit dieser es ausgeben kann.

Bei den Beispieldateien aus der c’t wurden die naheliegendsten Anwendungsfälle des Projektes bereits berücksichtigt. So fanden sich unter den Beispielen eine Wetteranzeige, ein Tagesplan für einen Gesprächsraum und ein Türschild. Da dieses Türschild jedoch nicht die gewünschten Anforderungen erfüllt hat, wurde in diesem Projekt ein eigenes Skript für die Darstellung erstellt, welches in Abschnitt „5.1 Bildanzeige über PHP“ genauer behandelt wird.

Ein weiteres, letztes PHP-Skript erlaubte die Anzeige eines statischen Bildes aus dem „static images“ Ordner. Dieses Bild muss dabei im PNG-Dateiformat vorhanden sein. Sollten in dem Ordner mehrere Bilder liegen, so wählt das Skript bei jeder Anfrage des Controllers zufällig eines der passenden aus und nimmt die Umwandlung vor.

3.3 Anforderungsanalyse

Zum Abschluss der Analysephase wurden die Anforderungen des Anwenders analysiert um die wichtigsten Anforderungen an das fertige Projekt herauszuarbeiten. Dabei konnte zum einen auf mündliche Aussagen, aber auch auf die formulierte Aufgabenstellung zurückgegriffen werden (siehe Abschnitt „1.1 Aufgabenstellung“). Insgesamt wurden folgende Anforderungen ausgearbeitet:

- Projekt aus der c’t lauffähig machen
- eigene Texte hinzufügen können (ähnlich dem Infomonitor des FB5)
- schnelle Verarbeitung der eingegebenen Texte
- umgesetzt in C oder Java (optional)

Während der Planung und der Analyse des bisherigen Projekts wurden weitere, interne Anforderungen spezifiziert. Diese sollen vor allem klare technische Anforderungen beschreiben, da die Anforderungen des Anwenders bewusst technisch unabhängig formuliert sind:

- eigenes PHP-Skript für Türschild-Anzeige schreiben
- Schnittstelle zum Webserver schreiben
- Anzeigen von Bildern und Text gemeinsam
- Programme möglichst modular entwickeln
- ausführliche Kommentierung des Quellcodes

Nachdem alle Anforderungen definiert wurden, wird im Folgenden mit der Umsetzung dieser begonnen. Wie erfolgreich diese Umsetzung verlaufen ist, wird im Abschnitt „6.1 Soll-/Ist-Vergleich“ dargestellt.

4. Entwurfsphase

4.1 Entwurf des Anzeigebilds (Muss)

Für den ersten Entwurf des Anzeigebilds wurde sich an dem bereits bestehenden, nicht-digitalen Türschild orientiert. Da dieses auch nur aus den Farben Schwarz, Weiß und Gelb besteht, war dies auch ohne weiteres möglich. Mit dem kostenlosen GUI-Prototyping Programm „Pencil“ wurde der erste Entwurf erstellt.

Das Türschild sollte in drei Bereiche aufgeteilt werden: Etwas Platz am oberen Ende nach dem Vorbild des bestehenden Türschildes, einen Infomonitor in der Mitte und einen Bereich am unteren

Ende für den Anwesenheitsstatus und Informationen über den Professor. Wie in dem bestehenden Türschild (siehe Anhang A3), wollten wir am oberen Ende die Raumnummer durch einen gelben Strich vom Rest des Türschilds getrennt haben. Neben der Raumnummer soll auch noch der Raumname ausgegeben werden. Um mehr Platz für den Infomonitor zu haben, wurden die Informationen über den Fachbereich entfernt. 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 wurden als erste Ideen für Professor-spezifische Informationen umgesetzt. Am anderen Ende des Kastens soll der aktuelle Anwesenheitsstatus angezeigt werden. Außerdem ist dort noch ein Bild eingefügt um das Türschild etwas anschaulicher zu gestalten. Nachdem genug Platz für den oberen und unteren Bereich geplant wurde, war der bleibende Bereich für den Infomonitor vorgesehen.

Für den Entwurf wurden die Informationen von Professor Korte eingetragen und ein paar Beispielmeldungen für den Infomonitor angegeben, sowie einen beispielhaften Anwesenheitsstatus und ein Bild, welches aus seinen Vorlesungen zu Software-Design entnommen wurde.

<p style="font-size: 1.2em; font-weight: bold;">Raum 365</p> <ul style="list-style-type: none"> ■ 24.05.2018 - 08:50 Uhr: Die Vorlesung "Programmiersprachen 2" fällt diese Woche aus! ■ 21.05.2018 - 11:37 Uhr: Die Klausureinsicht im Fach "Software-Design" findet am 12.06. um 12:30 Uhr statt! Sie geht bis 13:30. ■ 10.05.2018 - 15:12 Uhr: Ab nächstem Montag bin ich für 2 Wochen im Urlaub! Ich bin ab dem 28.05.2018 wieder anzutreffen. Bei dringenden Anliegen melden Sie sich bitte im Büro des FB5. 	<p style="font-size: 1.2em; font-weight: bold;">Labor für Informationstechnologie</p>
<div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"> <p>Prof. Dr.-Ing. Thomas Korte @ thomas.korte@hs-owl.de T (05261) 702-5839</p> </div> <div style="flex: 1; text-align: right;"> <p>Ich bin... ... anwesend</p>  </div> </div>	

Abbildung 2 - Entwurf des Anzeigebildes (Version 01)

4.2 Entwurf des Webinterface (Töberg)

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?	Programmiersprachen	CSV-Dateien
HTML Webinterface	benötigt nur Browser	Ja	HTML, JavaScript und PHP	Ja, über PHP
Java Desktopanwendung	benötigt JRE	Nein, nur Zugriff von einem Rechner auf lokale Dateien	Java	Ja
Java-Webserver (Java Servlet)	Server als auch Client benötigen JRE	Muss programmiert werden	Java	Ja

Tabelle 1 - Vergleich der Webserver-Schnittstellen

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.

Einrichtung des Türschilds

1) Anpassung des Zustands für das Feld "Ich bin..."

- anwesend
- abwesend
- krank
- im Urlaub
- in der Pause

2) Freitext erstellen

Aktuell werden folgende zwei Meldungen angezeigt:

10.05.2018 - 15:12 Uhr: Ab nächstem Montag bin ich für 2 Wochen im Urlaub! Ich bin ab dem 28.05.2018 wieder anzutreffen.
Bei dringenden Anliegen melden sie sich bitte im Büro des FB5.

21.05.2018 - 11:37 Uhr: Die Klausureinsicht im Fach "Software-Design" findet am 12.06. um 12:30 Uhr statt! Sie geht bis 13:30.

Abbildung 3 - Entwurf des Webinterface (Version 01)

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.

Einrichtung des Türschilds

Prof. Dr. Thomas Korte

1) Anpassung des Zustands für das Feld "Ich bin..."

- anwesend
- abwesend
- krank
- im Urlaub
- in der Pause

2) Freitext erstellen

Geben Sie hier die neue Meldung ein

Anmerkung: Dieses Feld darf nicht leer gelassen werden

Aktuell werden folgende Meldungen angezeigt:

> 10.05.2018 - 15:12 Uhr: Ab nächstem Montag bin ich für 2 Wochen im Urlaub! Ich bin ab dem 28.05.2018 wieder anzutreffen. Bei dringenden Anliegen melden sie sich bitte im Büro des FB5.

> 21.05.2018 - 11:37 Uhr: Die Klausureinsicht im Fach "Software-Design" findet am 12.06. um 12:30 Uhr statt! Sie geht bis 13:30.

Bestätigen Abbrechen

Projekt by Niclas Muss & Jan-Philipp Töberg

[Link zum Projekt auf GitHub](#)

Abbildung 4 - Entwurf des Webinterface (Version 02)

4.3 Entwurf des Netzwerks (Muss)

In dem ersten Entwurf des Netzwerks sollte Orts- und Geräteunabhängig zu jeder Zeit Informationen, wie der Anwesenheitsstatus und Meldungen für den Infomonitor aktualisiert werden. Die aktualisierten Meldungen sollen so gespeichert werden, dass die PHP-Skripte das aktualisierte Türschild „zeichnen“ können, wenn der ESP32 auf den Server zugreift.

Da das E-Paper direkt über ein Kabel mit dem ESP32 verbunden ist, war die erste Hürde die Frage, wie der ESP32 mit dem PHP-Server verbunden wird. Das Programm der c't hat die Netzwerkfunktionalitäten über die „Basecamp“-Library gelöst. Mit dieser lässt sich problemlos eine Verbindung zu einem Netzwerk und einem Server innerhalb dieses Netzwerks herstellen. Da der Controller alle benötigten Werte lokal speichert, kann er, nach einmaliger Anmeldung in diesem Netzwerk, sich nach einem Neustart direkt wieder mit diesem Netzwerk verbinden, wenn sich die Einstellungen des Netzwerks nicht geändert haben.

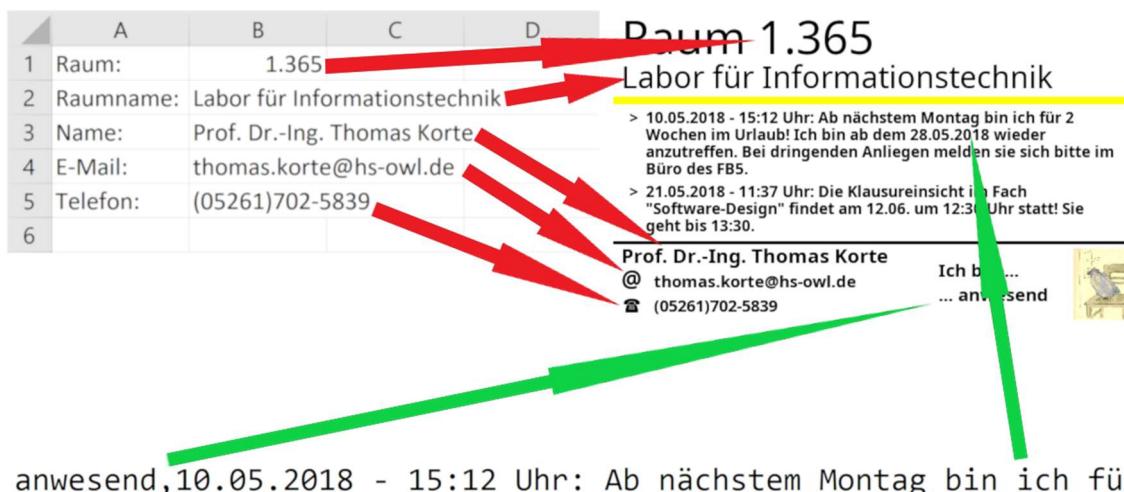


Abbildung 5 - Veranschaulichung des Zusammenhangs von csv-Datei und Anzeige

Da nicht über die entworfene Weboberfläche in der Basecamp Weboberfläche Werte geändert werden können, musste das Aktualisieren des Bildes unabhängig von den Basecamp Funktionalitäten designet werden. Da Basecamp auf eine spezifizierte PHP-Datei zugreift, muss diesen Pfad statisch belassen werden, jedoch kann geändert werden, 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.

5. Implementierungsphase

5.1 Bildanzeige über PHP (Muss)

Das Bild, welches auf dem Bildschirm angezeigt werden soll, wird über zwei PHP-Skripte erstellt. Das Index Skript wird direkt vom Controller aufgerufen. Durch Spezifikation in der angegebenen Serveradresse wird das zweite Skript angegeben, welches von der index.php aufgerufen wird um das auszugebende Bild zu „zeichnen“.

5.1.1 index.php

Die index.php ist die Schnittstelle zwischen dem ESP32 und dem Anzeigeskript. Es ruft ein anderes Skript auf und gibt dessen Inhalt, in einer für den Controller verständlichen Form, zurück.

Wenn versucht wird, das Skript aufzurufen, müssen einige Parameter übergeben werden, damit das Skript richtig funktionieren kann.

- Debug: Ein boolescher Wert, der festlegt, wie das Bild zurückgegeben werden soll:
 - True: Gibt das Bild als Bilddatei zurück. Lesbar von Webbrowsern und daher gut um zu testen, wie das gezeichnete Bild aussehen wird.
 - False: Gibt das Bild als Byte-String zurück. Lesbar von dem Controller und daher nötig um das Bild auf dem E-Paper anzeigen zu können.
- Display: Bei diesem Wert wird die Größe des Displays in Inches und die Information, ob er Farben anzeigen kann oder nicht übergeben. Die hier möglichen Werte werden in einem Array definiert und betragen 7.5, 7.5bwr, 4.2, 4.2bwr, 2.9 & 1.5.
 - Die Anzeigeskripte, die im Rahmen dieser Projektarbeit entstanden sind, sind nur für die Bildschirmarten 7.5 und 7.5bwr ausgelegt und skalieren schlecht bis gar nicht auf andere Displaygrößen.
- Content: Name des Anzeigeskriptes. Hier muss das Anzeigeskript angegeben werden, welches ausgegeben werden soll. Die Datei muss die Endung .php besitzen und im Ordner „contents“ liegen, damit es akzeptiert wird.
- Professor: Auswahl welche Daten über das Anzeigeskript ausgegeben werden. Genauere Erläuterung folgt im nächsten Kapitel.

Wenn kein Wert für Debug eingegeben wird, dann wird dieser automatisch auf false gesetzt. Display, Content und Professor müssen jedoch eingegeben werden, da das Skript ansonsten frühzeitig abbricht und eine Fehlermeldung ausgibt. Außerdem überprüft es, ob alle wichtigen Bibliotheken installiert sind und bricht die Bearbeitung ab, falls sie es nicht sind. Im nächsten Schritt werden, basierend auf den Parametern, einige Variablen definiert, die auch für das Anzeigeskript relevant sind. Diese Variablen sind die Dimensionen des Displays in Pixeln, eine Bildressource mit den Ausmaßen des Displays, die zuerst mit der Farbe Weiß gefüllt wird, und ein boolescher Wert, der angibt ob der Bildschirm rot, oder in diesem Projekt gelb, anzeigen kann oder nicht, welcher sowohl im Anzeigeskript als auch bei der Übergabe der Daten an den ESP32 benutzt wird. Neben den Variablen, die von den Parametern abhängig sind, werden auch noch parameterunabhängige Variablen festgelegt. Dazu gehören die Farbcodes für Schwarz, Rot und, zu Testzwecken, gelb.

Außerdem werden die Pfade zu den verschiedenen unterstützten Schriftarten in einem Array definiert.

Als nächstes wird das Bild durch das Anzeigeskript gezeichnet. Durch den „include“-Befehl kann eine andere PHP-Datei eingebunden werden. Die eingebundene Datei ist das über den content-Parameter angegebene Skript. Da das andere Skript innerhalb des index-Skriptes ausgeführt wird, teilen sich die Skripte die Variablen. Das andere Skript kann also in die gleiche Bildressource, die vorher bereits mit der Größe des Bildschirms initialisiert wurde, hineinschreiben.

Nachdem das Anzeigeskript, welches im nächsten Kapitel genauer erklärt wird, nun die geteilte Bildressource so beschrieben hat, dass das Türschild durch die Bildressource darstellt wird, liegt es nun an dem Index-Skript diese Ressource angemessen weiterzugeben. Je nach Einstellung der Debug-Variable wird die Ressource anders übergeben. Wenn Debug auf „True“ gesetzt ist, muss die Ressource für die Wiedergabe über einen Webbrowser aufbereitet und letztendlich angezeigt werden. Sollte Debug jedoch auf „False“ eingestellt sein, wird die Funktion rawImage() aufgerufen um die Ressource in eine Form zu bringen, die der Controller verarbeiten kann, um das gezeichnete Bild auf dem E-Paper Display anzeigen zu können. Diese Funktion überprüft für jeden Pixel einzeln die vorhandene Farbe und versucht, den vorhandenen Wert in Schwarz, Weiß oder Rot einzuteilen. So würde er also ein dunkles Grau als Schwarz ausgeben und ein helles Grau als Weiß. Für ein schwarz-weißes Display steht ein Bit stellvertretend für Schwarz oder Weiß, wobei eine 1 für Schwarz und eine 0 für Weiß steht. Für ein mehrfarbiges Display werden zwei Bit für einen Pixel benötigt, wobei „11“ für Schwarz, „00“ für Weiß und „01“ für Rot steht. Die einzelnen Bits werden noch in Bytes verpackt und am Ende des Skriptes wird der Byte-String zurückgegeben.

5.1.2 türschild.php

Das Türschild-Skript dient für unsere Zwecke als das Anzeigeskript, welches von dem Index-Skript eingebunden wird. In dem Index-Skript wurde bereits eine Bildressource initialisiert, die jetzt von dem Anzeigeskript über Funktionen aus der GD-Library „bemalt“ werden soll. Die GD-Library ist eine Open-Source-Programmbibliothek, zur dynamischen Erzeugung und Manipulation von Grafiken.

Um für eine gute Erweiterbarkeit des Türschildes zu sorgen, werden fast alle der angezeigten Werte aus externen CSV-Dateien eingelesen. In der Datei „Professor.csv“ stehen Werte, die voraussichtlich für längere Zeit konstant bleiben, wie die Raumnummer und Informationen über den Professor, welchem das Büro gehört. In der Datei „Professor_status.csv“ hingegen werden die Werte abgelegt, die häufiger durch die Weboberfläche geändert werden können, wie der Anwesenheitsstatus und die Infomonitormeldungen. Diese Daten werden am Anfang des Skriptes eingelesen und in zwei verschiedene Arrays geschrieben.

Außerdem soll es möglich sein, zu den Informationen eines anderen Professors zu wechseln, wenn diese bereits eingetragen wurden. Für diesen Zweck gibt es den Parameter „professor“, in welchem man den Namen des Professors eingeben kann, dessen Türschild man anzeigen möchte. Die csv-Dateien müssen Namenskonventionen folgen, damit dieser Parameter richtig funktionieren kann. Die Datei, die im vorherigen Kapitel „professor.csv“ genannt wurde, muss den gleichen Namen haben wie der Parameter und die „status.csv“ Muss einen Namen der Form „Parameter_status.csv“ haben.

Neben dem Parameter für den Professor ist auch der im vorherigen Kapitel erwähnte Parameter des Displays wichtig, um genau zu sein, die Variable \$hasred, welche aussagt, ob der Display Farben oder nur schwarz anzeigen kann. Die Variable wurde bereits im Index-Skript definiert und ist abhängig von dem Parameter Display. Wenn man also einen Bildschirm als Parameter übergibt, der keine Farben anzeigen kann, soll auch in der Webansicht die Farbe Gelb durch Schwarz ersetzt werden.

Wie schon erwähnt kennt das E-Paper Display softwareseitig nur die Farbe Rot und möchte diese übergeben bekommen. Da aber die endgültige Anzeigefarbe des Displays gelb ist, wäre es schön, wenn die Webansicht gelb anzeigt, obwohl der Controller rot übergeben bekommt. Dafür benutzen wir die bereits definierte Variable „debug“, welche, wie im letzten Kapitel erläutert, angibt, ob die fertige Bilddatei in einem Webbrower ausgegeben werden soll, oder ob sie für den Controller in Pixelarrays aufbereitet werden soll. In dem Skript wird nur die Variable \$red benutzt, die je nach Wert des debug-Parameters entweder den Farbcode der Farbe Rot enthält, wenn die Bildressource an den Controller übergeben werden soll, oder den Farbcode der Farbe Gelb, falls die Webansicht benutzt werden soll.

Unser Türschild besteht aus drei Bereichen, wie bereits in dem Kapitel „4.1 Entwurf des Anzeigebilds“ beschrieben. Im oberen Teil soll die Raumnummer und der Name des Büros stehen, welche daraufhin mit einer gelben Linie vom Rest des Schildes getrennt werden. Zuerst werden die Texte mit der Funktion „imagettfttext“ generiert, welche einen Text auf eine Imageressource schreibt. Dazu müssen die x und y-Koordinaten angegeben werden, sowie den Pfad zu einer Schriftart, die Schriftfarbe und eine Schriftgröße. Sowohl die Schriftgröße und die y-Koordinate werden über Variablen realisiert und die Schriftart kann über ein Array, welches in dem Index-Skript initialisiert wurde, ausgewählt werden. Die x-Koordinate jedoch hat keine eigene Variable und muss manuell für jeden Text eingegeben werden. Das liegt daran, dass die Variable für die y-Koordinate hochgezählt werden kann, da das Skript die Bildressource von oben nach unten bearbeiten wird. Daher kann eine Variable für die Texthöhe während des Skriptes hochgezählt werden, während die x-Koordinaten für jeden Text

unabhängig von der Stelle im Skript sind. Die beiden ersten Werte des Arrays, welches aus der Professor.csv erzeugt wurde, sind die Raumnummer und der Raumname, die beide als Text auf die Bildressource gezeichnet werden. Für die gelbe Trennlinie wird die Funktion „imagine“ genutzt. Auch diese Funktion „zeichnet“ auf die Bildressource aus dem Index-Skript. Diese Funktion zeichnet jedoch keinen Text, sondern eine Linie. Für diese Funktion müssen der Start und der Endpunkt der

Raum 365 Labor für Informationstechnologie

- > 10.05.2018 - 15:12 Uhr: Ab nächstem Montag bin ich für 2 Wochen im Urlaub! Ich bin ab dem 28.05.2018 wieder anzutreffen. Bei dringenden Anliegen melden sie sich bitte im Büro des FB5.
- > 21.05.2018 - 11:37 Uhr: Die Klausureinsicht im Fach "Software-Design" findet am 12.06. um 12:30 Uhr statt! Sie geht bis 13:30.
- > 24.05.2018 - 08:50 Uhr: Die Vorlesung "Programmiersprachen 2" fällt diese Woche aus!

Prof. Dr.-Ing. Thomas Korte

E-Mail: thomas.korte@hs-owl.de

Telefon: (05261)702-5839

Ich bin ...
... anwesend



Raum 365 Labor für Informationstechnologie

- > 10.05.2018 - 15:12 Uhr: Ab nächstem Montag bin ich für 2 Wochen im Urlaub! Ich bin ab dem 28.05.2018 wieder anzutreffen. Bei dringenden Anliegen melden sie sich bitte im Büro des FB5.
- > 21.05.2018 - 11:37 Uhr: Die Klausureinsicht im Fach Software-Design" findet am 12.06. um 12:30 Uhr statt! Sie geht bis 13:30."

Prof. Dr.-Ing. Thomas Korte

E-Mail: thomas.korte@hs-owl.de

Telefon: (05261)702-5839

Ich bin ...
... im Urlaub



Raum 1.365 Labor für Informationstechnik

- > 10.05.2018 - 15:12 Uhr: Ab nächstem Montag bin ich für 2 Wochen im Urlaub! Ich bin ab dem 28.05.2018 wieder anzutreffen. Bei dringenden Anliegen melden sie sich bitte im Büro des FB5.
- > 21.05.2018 - 11:37 Uhr: Die Klausureinsicht im Fach "Software-Design" findet am 12.06. um 12:30 Uhr statt! Sie geht bis 13:30.

Prof. Dr.-Ing. Thomas Korte

@ thomas.korte@hs-owl.de

☎ (05261)702-5839

Ich bin ...
... anwesend



Abbildung 6 - Entwicklung des Anzeigebildes über den Projektzeitraum

Linie festgelegt werden. Damit also eine horizontale Linie über den gesamten Bildschirm angezeigt wird, müssen die y-Koordinaten am Startpunkt und Endpunkt gleichbleiben. Die y-Koordinaten des Startpunktes muss 0 sein und für den Endpunkt der Linie bietet sich die bereits definierte Variable über die Bildschirmweite in Pixeln an. Um festzulegen, wie Fett die Linie sein soll muss vorher die Funktion „imagesetthickness“ aufgegeben werden, welcher man eine Zahl übergibt, die die Dicke alle weiteren Linien, die durch GD-Library-Funktionen gezeichnet werden, angibt. Dabei ist zu beachten, dass wenn wir die Zahl vier übergeben die Linie acht Pixel breit ist, da sich die Dicke in beide Seiten erstreckt.

Der Bereich in der Mitte ist reserviert für den Infomonitor. Dieser war die größte Herausforderung in der Implementierung des Anzeigeskriptes. Schon in der ersten Implementierung wurde klar, dass die Ausgabe über „imagettfttext“ keine automatischen Zeilenumbrüche unterstützt, da sie nur Text auf eine Bildressource „zeichnet“ ohne dessen Aufmaße zu kennen. Dadurch wurde der Text einfach in einer Zeile gedruckt, bis er am rechten Bildschirmrand abgeschnitten wurde. Zeilenumbrüche mussten also manuell in das Anzeigeskript implementiert werden. Da die Länge eines Textes nicht von der Anzahl Buchstaben abhängig ist, weil Buchstaben unterschiedliche Breiten haben können, musste die Länge des Textes dynamisch bestimmt werden. Die GD-Library bietet mit „imagettbbox“ eine Funktion an, die genau das ermöglicht. Wenn man ihr den Text zusammen mit der benutzen Schriftart und Größe übergibt, wird ein Array zurückgegeben, welches die Eckpunkte einer box angibt, welches den Text umgeben könnte. Die Differenz zwischen einem Punkt am rechten Rand und einem Punkt am linken Rand ergibt die Länge des Textes in Pixeln, die wir mit der Variable \$displayWidth, welche die Breite des Displays in Pixeln angibt, vergleichen können. Um zu wissen an welcher Stelle Zeilenumbrüche eingefügt werden müssen muss die Textlänge jedoch nach jedem einzelnen Wort überprüft werden.

Dazu muss im ersten Schritt eine Schleife gestartet werden, die über die Anzahl der Nachrichten iteriert, damit jede Nachricht einzeln bearbeitet und ausgegeben werden kann. Die Anzahl der Nachrichten ist die Größe des Arrays welche aus der Status-Datei ausgelesen wurde, abzüglich des Anwesenheitsstatus. Dazu müssen die Nachrichten zuerst durch die „explode“-Funktion in Arrays aus Wörtern aufgebrochen werden. Nun wird mit einer Zählschleife über die Anzahl an Wörtern in dem Array iteriert und in jedem Schritt wird ein weiteres Wort dem Teststring \$zeile hinzugefügt. Mit „imagettbbox“ wird die Länge des Strings zu diesem Zeitpunkt, also immer nach dem Hinzufügen eines neuen Wortes zu dem Teststring, überprüft und mit der Bildschirmweite abgeglichen. Falls die Breitenbegrenzung noch nicht überschritten wurde, wird das Wort was zuletzt dem \$zeile hinzugefügt wurde auch dem String \$text hinzugefügt, der später ausgegeben werden soll. Falls die Breitenbegrenzung überschritten wurde, wird das letzte Wort zuerst zurückgehalten. In dem String \$text steht nun genau so viel Text, wie in eine Zeile passt, daher kann dieser ausgegeben werden. Da jedoch eine Nachricht nur als Ganzes ausgegeben werden soll und noch nicht bekannt ist, ob die Nachricht nicht vielleicht noch die Höhenbegrenzung überschreitet, muss der \$text String zuerst noch einmal zwischengespeichert werden. Dies geschieht im Array \$nachricht in dem jedes Element eine Zeile einer Nachricht sein soll. Nachdem die Schleife abgearbeitet ist und die Nachricht zeilenweise in ein Array verpackt wurde, wird einmal überprüft, ob die Nachricht die Höhenbeschränkung des Infomonitor übersteigt. Falls sie zu groß ist und in den dritten Teil der Anzeige hineinschreiben würde, wird diese Nachricht nicht mehr ausgegeben und auch keine weiteren. Die Schleife über die Anzahl der Nachrichten wird daraufhin mit einem break-Befehl vorzeitig beendet.

Falls die Höhenbegrenzung nicht überschritten wurde, kann die Nachricht ganz normal ausgegeben werden. Dazu wird zuerst das Zeichen „>“ über „imagettfttext“ gedruckt werden, um den Beginn einer neuen Nachricht zu kennzeichnen. Daraufhin werden alle Zeilen aus dem \$nachricht Array

untereinander mit „`imagedtttext`“ ausgegeben. Danach springt das Skript wieder zum Anfang der äußeren Zählschleife, die über die Anzahl der Nachrichten iteriert, falls noch weitere Nachrichten vorliegen. Dabei behält es jedoch die Variable, welche die y-Koordinate angibt, bei um testen zu können, ob die nächste Nachricht die Höhenbeschränkung überschreitet, da man ab der zweiten Nachricht einen Startpunkt der Nachricht hat, die von der Größe der vorherigen Nachricht abhängt. Die weiteren Nachrichten werden dabei auf dieselbe Art bearbeitet, wie die erste. Sobald es keine Nachrichten mehr gibt, oder die Schleife aufgrund der Höhenbegrenzung frühzeitig abbricht, ist die Bearbeitung des Infomonitors beendet und das Skript kann zum nächsten Teil der Anzeige übergehen.

Am unteren Ende des Bildschirms soll eine Box angezeigt werden, in der der Anwesenheitsstatus und einige Informationen zum Professor stehen. Um die Box von dem Infomonitor abzugrenzen, soll das Skript zuerst eine Linie mit „`imageline`“ zeichnen, wobei auch hier vorher mit „`imagesetthickness`“ die Dicke der Linie angegeben werden. Zuerst wird der Name des Professors ausgegeben, welchem dieses Büro gehört. Diese Information kann man aus der Einrichtungs-Datei lesen. Direkt darunter sollen die Telefonnummer und die E-Mailadresse des Professors angegeben werden. Vor diesen Informationen soll ein einzelnes Zeichen gedruckt werden, das anzeigen soll, ob es sich um die E-Mailadresse oder die Telefonnummer handelt. Für die E-Mailadresse wird ein einzelnes „@“ mit einer leicht erhöhten Schriftgröße über „`imagedtttext`“ gedruckt werden und für die Telefonnummer wird durch das Telefonzeichen aus der Schriftart „NotoEmoji-Regular“ dargestellt. Jeweils hinter diesen Zeichen sollen die dazugehörigen Werte geschrieben werden, die auch aus der zugehörigen csv-Datei eingelesen werden. Auch in der Box am unteren Ende soll der Anwesenheitsstatus stehen für welchen wir erst den festen String „Ich bin ...“ mit Hilfe von „`imagedtttext`“ drucken und darunter den Anwesenheitsstatus zeichnen, welcher das erste Element in dem status-Array ist. Als letztes angezeigtes Element gibt es noch ein Bild, welches rechts-unten in der Ecke angezeigt werden soll. Der Pfad des Bildes ist abhängig von dem Professor-Parameter. Um ein Bild anzeigen zu können muss ein Bild mit dem Namen des Parameters im Ordner Bilder auf dem Webserver liegen. An dieser Stelle wird auch noch unterschieden ob der Bildschirm Farbe angeben kann. Falls er es kann muss noch „_bwr“ an den Namen der Bilddatei angehängen werden. So gibt es ein Bild für farbfähige Bildschirme und ein schwarz-weißes Bild für Bildschirme, die keine Farben anzeigen können. Damit das Bild richtig angezeigt werden kann, müssen die Dimensionen so geändert werden, dass es in die Ecke passt, dabei kann es jedoch das ursprüngliches Seitenverhältnis des Bildes auf 3:4 gestreckt oder gestaucht wird. Das Bild kann vom Typ PNG oder vom Typ JPEG sein, falls kein Bild angegeben ist oder die Datei von einem nicht unterstützten Typ sein sollte, wird kein Bild angezeigt.

Am Ende des Anzeigeskripts sollte die Bildressource so bemalt worden sein, dass sie das Türschild darstellt.

5.2 Das Webinterface (Töberg)

Parallel zu der Entwicklung des PHP-Skriptes, welches sich mit der Anzeige des Türschildes auf dem e-Paper befasst, wurde das Webinterface entwickelt. Dabei besteht das Webinterface aus mehreren Abschnitten und Dateien, welche dem Nutzer ermöglichen sollen, über eine Webschnittstelle seinen Zustand einzustellen und Meldungen hinzuzufügen. Der Austausch zwischen diesen unterschiedlichen Dateien und den Skripten der Anzeige erfolgt dabei über eine CSV-Datei, welche beschrieben und gelesen werden kann.

Bei der Entwicklung des Webinterface wurde mit dem Erstellen eines HTML-Skriptes begonnen, welches die Darstellung der verschiedenen graphischen Elemente ermöglicht. Innerhalb dieses HTML-Skriptes findet sich ein Abschnitt in welchem die dahinterliegende Funktionalität mit Hilfe von JavaScript entwickelt wurde. Beide Aspekte, welche im Folgenden noch genauer beschrieben werden, werden jedoch auf Clientseite ausgeführt, weshalb es einen Gegenpart auf der Serverseite geben muss, welcher Anfragen bearbeiten und beantworten kann. Diese Funktionalität wurde mit zwei unterschiedlichen PHP-Skripten realisiert, welche in Abschnitt „5.2.2 PHP auf Server-Seite“ genauer beschrieben werden.

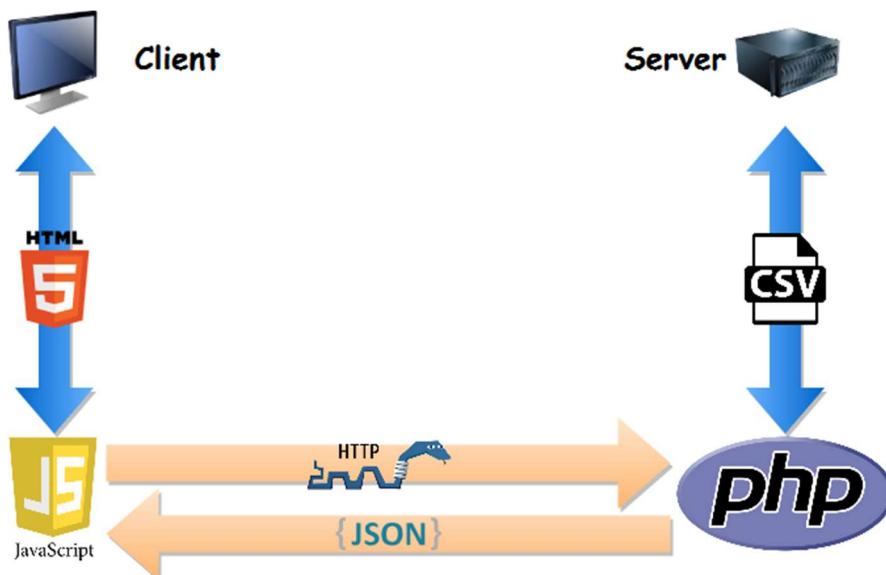


Abbildung 7 - Kommunikation zwischen Server und Client

5.2.1 HTML und JavaScript auf Client-Seite

Im Folgenden wird die clientseitige Implementierung dargestellt werden. So beginnt das Dokument mit ein paar allgemeinen Festlegungen (z.B.: Sprache des Dokuments ist deutsch, Zeichensatz ist UTF-8, usw.) gefolgt von der Definition der unterschiedlichen Interaktionselemente mit ihren jeweiligen Attributen. Besonders hervorheben möchte ich dabei die Attribute „id“ und „style“, welche bei vielen (jedoch nicht allen) Elementen verwendet werden. Über das id-Attribut wird dem Element eine eindeutige Identifikation zugewiesen, welche das Auffinden und Bearbeiten des Elements über JavaScript stark vereinfacht. Mit Hilfe des style-Attribut können unterschiedliche Einstellungen bezüglich des Designs des Elements gemacht werden. Dabei wird die gleiche Schreibweise wie bei CSS-Dateien verwendet, dies jedoch nicht über den expliziten Import einer passenden Datei, sondern über eine eingebettete Art und Weise.

Nachdem alle Elemente definiert wurden, werden bestimmte Elemente mit dazugehörigen JavaScript-Funktionen verknüpft, sodass die Funktion bei einem bestimmten Ereignis ausgeführt wird. Die drei Ereignisse auf die reagiert wird sind der Klick auf jeweils einen der Buttons, der Wechsel des Professors über das Dropdown-Menü und das Aktualisieren der Website.

Die Funktion mit dem geringsten Umfang wird aufgerufen, wenn entweder der Professor gewechselt wird oder der „Abbrechen“-Button betätigt wird. Die sogenannte „reloadTheWholePage()“-Funktion tut was der Name verspricht und lädt einmal die Seite neu, was automatisch die „reloadAndLoadData()“-Funktion ausführt. Diese Funktion erstellt eine neue http-Request an den Webserver, um genau zu sein an die „GetDataFromCSV.php“-Datei. In dieser Anfrage sendet der Client die Bezeichnung des aktuell ausgewählten Professors mit, damit der Webserver weiß welche

Daten er laden & zurücksenden muss. Nachdem die Anfrage gesendet wurde, wartet die Funktion auf eine Antwort (Anhang A10, ab Zeile 57) und wertet, bei Erhalt dieser, die Antwort im JSON-Format aus. Die Antwort enthält dabei den aktuell gespeicherten Zustand und die zwei aktuellsten Meldungen des Professors. Diese werden direkt auf den Inhalt der zugehörigen Elemente der Weboberfläche geschrieben. Dadurch wird garantiert, dass nach jeder Aktualisierung der Seite die aktuellsten Daten angezeigt werden.

Die letzte Funktion („confirmAndStartSending()“) befasst sich mit dem Senden der neuen Daten an den Webserver. Diese Funktion wird immer ausgeführt, wenn der „Bestätigen“-Button geklickt wird. Die Funktion beginnt damit, die unterschiedlichen Eingaben des Anwenders auszuwerten. Dafür werden die oben bereits angesprochenen eindeutigen Bezeichner der Elemente verwendet, um mit Hilfe des `document.getElementById("ID").innerHTML`-Aufrufs den Inhalt des Elements zu bekommen. Mit dem Inhalt ist in diesem Kontext der Text gemeint, welchen das Element anzeigt. Innerhalb dieser Funktion werden auch die unterschiedlichen RadioButtons innerhalb einer Zählschleife ausgewertet, um zu bestimmen, welcher der fünf Buttons ausgewählt ist. Nachdem alle relevanten Informationen ausgewertet wurden, werden diese per Methodenaufruf an die „`sendToPHPAndSave()`“-Methode übertragen, welche ebenfalls eine http-Request formuliert. In diesem Fall wird jedoch die „`saveDataToCSV.php`“-Datei angesprochen und es werden auch mehr Parameter übergeben. Diese werden in der Funktion vor dem Senden zu einem String mit passenden Stichworten konkateniert, was die Auswertung durch das PHP-Skript erleichtert. Danach wird dieser String gesendet und die Antwort des Servers auf der Konsole ausgegeben.

5.2.2 PHP auf Server-Seite

Nachdem die clientseitige Kommunikation beschrieben wurde, werden im Folgenden die beiden, bereits angesprochenen, PHP-Dateien behandelt, welche den Zugriff auf die CSV-Datei auf dem Webserver steuern. Beide Dateien besitzen dabei gewisse Gemeinsamkeiten. Zum einen nutzen beide einen vordefinierten Header, welcher die aktuellen Informationen zum Dokument bündelt. Des Weiteren legen beide über den Befehl „`mb_internal_encoding("UTF-8")`“ ihre interne Kodierung auf UTF-8 fest, was das Verarbeiten von Sonderzeichen in den Strings, welche per http-Anfrage gesendet werden, ermöglicht. Zu guter Letzt teilen sich beide Dateien eine Mehrfachverzweigung, welche, abhängig vom gesendeten Professor, den Dateinamen festlegt, aus welcher gelesen oder in welche geschrieben werden soll. Dadurch wird garantiert, dass mehrere Professoren unabhängig voneinander ihre Meldungen bearbeiten können.

Die „`getDataFromCSV.php`“-Datei verwaltet den Lesezugriff auf die CSV-Datei. Dabei wird, nachdem der passende Dateiname festgelegt ist, aus dieser Datei alles ausgelesen und in ein Array geschrieben. Die Elemente dieses Arrays werden dann ins JSON-Format überführt und als Antwort an den Client gesendet.

Wenn der Client jedoch Daten sendet, welche in einer Datei gespeichert werden sollen, so wird die „`saveDataToCSV.php`“-Datei angefragt, welche mit den oben bereits beschriebenen Formatierungen und der Dateinamen-Abzweigung beginnt. Nachdem diese Aspekte festgelegt wurden, werden die gesendeten Daten des Clients ausgelesen. Diese werden bei einer http-Anfrage in ein Array auf dem Server geschrieben, welches dieser einfach auslesen kann. Lediglich die Formatierung muss bei einigen Werten angepasst werden, da die alten Meldungen, welche mitgeschickt werden, ein vorangestelltes Datum mitsamt Uhrzeit besitzen. Dieses wird am Anfang entfernt um spätere Vergleiche zu erleichtern. Daraufhin wird in einer verschachtelten bedingten Anweisung festgelegt, welche Daten in die Datei geschrieben werden sollen. Dabei werden neue, vom Anwender geschriebene Meldungen nicht gespeichert, wenn Sie entweder leer sind, aus der Standard-Nachricht bestehen oder die aktuell angezeigte Meldung sind. Nachdem diese Festlegung passiert, werden die Daten über einen einfachen Methodenaufruf in die passende CSV-Datei geschrieben und über den

„echo“-Befehl wird eine Antwort an den Client gesendet, welche kurz zusammenfasst, was geschrieben wurde.

6. Fazit

6.1 Soll-/Ist-Vergleich (Töberg)

Bevor im nächsten Abschnitt beschrieben wird, welche möglichen Erweiterungen der aktuelle Stand dieses Projekts bietet, wird im Folgenden darauf eingegangen ob die in Abschnitt „3.3 Anforderungsanalyse“ beschriebenen Anforderungen umgesetzt werden konnten.

So wurden die grundsätzlichen Anforderungen des Anwenders erfolgreich umgesetzt. Das Türschild erlaubt das flexible Hinzufügen von Texten, welche auf dem Display angezeigt werden. Lediglich die Geschwindigkeit bietet Verbesserungspotential, da das Türschild aktuell nicht auf Veränderungen durch den Anwender reagiert, sondern nach jeder Aktualisierung in einen Leerlauf übergeht. Sobald es aus diesem erwacht, überprüft es, ob andere Meldungen angezeigt werden müssen. Ist dies nicht der Fall, so ändert sich nichts und es wird wieder in den Leerlauf gegangen. Dieser Leerlauf existiert nur im Produktionsmodus und sobald dieser verlassen wird, wird der Controller in den Tiefschlafmodus gehen und so wenig Strom verbrauchen. Da die Tiefschlafzeit aktuell jedoch zehn Minuten beträgt, werden im schlimmsten Fall neue Meldungen erst nach zehn Minuten angezeigt.

Leider konnte die optionale Anforderung das Projekt in der Programmiersprache Java oder C umzusetzen nicht umgesetzt werden. Das lag zum einen daran, dass der C/C++-Teil des Projekts bereits abgeschlossen war und keiner Erweiterung bedurfte, aber auch daran, dass es bessere Alternativen gab (wie in Abschnitt „4.2 Entwurf des Webinterface“ dargestellt).

Die Anforderungen, die das Projektteam ergänzend formuliert hat, konnten überzeugend umgesetzt werden. Insbesondere auf die Kommentierung und die modulare Entwicklung wurde geachtet. Durch die modulare Entwicklung ist ein Grundstein für die Weiterentwicklung des Projektergebnisses eventuell sogar durch ein anderes Projektteam gelegt.

6.2 Ausblick (Muss)

Bei der Implementierung des Anzeigeskriptes wurde viel Wert auf die Erweiterbarkeit gelegt. Zusammen mit dem guten Grundgerüst, welches von der c't übernommen werden konnte, ist es sehr leicht das Türschild mit den Werten eines anderen Professors zu füllen. Alles was getan werden muss um dies zu ermöglichen, ist das Anlegen von zwei csv-Dateien nach dem Vorbild der bereits vorhandenen und gegebenenfalls das Hinzufügen eines Bildes an die richtige Stelle.

Die erste csv-Datei, die angelegt werden muss, ist die Datei, die Konstanten über den Professor und den Raum beinhalten soll. Diese muss den Namen tragen, welcher später als Parameter übergeben werden muss um sich die Daten anzeigen zu lassen. Um diese Dateien auseinanderhalten zu können, haben wir in unseren Beispielen den Namen des Professors als Parameter benutzt. Die csv-Datei ist benutzerfreundlich aufgebaut, da diese von Benutzern geändert werden soll. Alle Werte der Tabelle müssen gefüllt sein, damit das das Anzeigeskript funktioniert.

professor.csv	Eintrag	korte.csv	hausdoerfer.csv
Raum:	Nummer des Raumes	1.365	1.328
Raumname:	Name oder Titel des Raumes	Labor für Informationstechnik	Echtzeitsysteme
Name:	Name des Professors	Prof. Dr.-Ing. Thomas Korte	Prof. Dr.-Ing. Rolf Hausdörfer
E-Mail:	E-Mail-Adresse des Professors	thomas.korte@hs-owl.de	rolf.hausdoerfer@hs-owl.de

Telefon	Telefonnummer des Professors	(05261)702-5839	(05261) 702-3562
---------	------------------------------	-----------------	------------------

Tabelle 2 - Aufbau der professor.csv mit zwei Beispieleintragungen

Die zweite csv-Datei ist die Datei, die regelmäßig von der Weboberfläche geändert wird. Da unterschiedliche Professoren auch unterschiedliche Infomonitormeldungen haben können, muss das Skript neben einer anderen professor.csv auch eine andere status.csv benutzen. Auch dieser lässt sich durch den Parameter „professor“ auswählen, wenn die csv-Datei nach dem Muster professorname_status.csv benannt wurde. Die momentane Weboberfläche kann auch zwischen mehreren verschiedenen Professoren differenzieren und in die zugehörigen csv-Dateien schreiben.

Auch das Bild, welches rechts-unten in der Ecke angezeigt wird kann von Professor zu Professor unterschiedlich sein. Auch hier müssen die Bilder einen Namen tragen, der später durch den Parameter zugeordnet werden kann, nach dem Muster „parameter.png“ und „parameter_bwr.png“. Die Endung muss dabei nicht zwangsläufig „.png“ sein, da „.jpeg“-Dateien auch unterstützt werden

Durch Professor-abhängige csv-Dateien, und die Auswahl dieser über Parameter ist gewährleistet, dass mehrere Professoren das gleiche Skript und den gleichen Server benutzen können, wenn alle ihren eigenen Controller und ihr eigenes E-Paper Display besitzen. Es muss nur der richtige Parameter bei der Ersteinrichtung in der Basecamp Weboberfläche übergeben werden.

Da die drei Bereiche des Türschildes im Skript komplett unabhängig voneinander sind, lassen sich Änderungen, die nur einen der drei Teile betreffen gut durchführen, ohne die anderen zu beeinflussen. Falls ein anderer Professor anstelle des Infomonitors beispielsweise eine Art Kalender haben möchte. Könnte man den Infomonitor einfach ersetzen, ohne dass es einen Einfluss auf den oberen oder den unteren Abschnitt hat.

7. Quellen

1) Modulhandbuch Fachbereich 5

Letzter Aufruf am: 28.06.2018

[Modulhandbuch FB5](#)

2) Doorsign EPD Repository by jamct on GitHub

Letzter Aufruf am: 28.06.2018

[DoorsignEPD](#)

3) Basecamp by merlinschuhmacher on GitHub

Letzter Aufruf am: 29.06.2018

[Basecamp](#)

4) arduino-esp32 by espressif

Letzter Aufruf am: 30.06.2018

[Espressif](#)

5) ESP32 „Thing“ der Firma SparkFun

Letzter Aufruf am: 01.07.2018

[ESP32](#)

5) E-Paper Display der Firma Waveshare

Letzter Aufruf am: 01.07.2018)

[7.5inch e-Paper HAT](#)

A Anhang

A1 – Netzwerkdiagramme

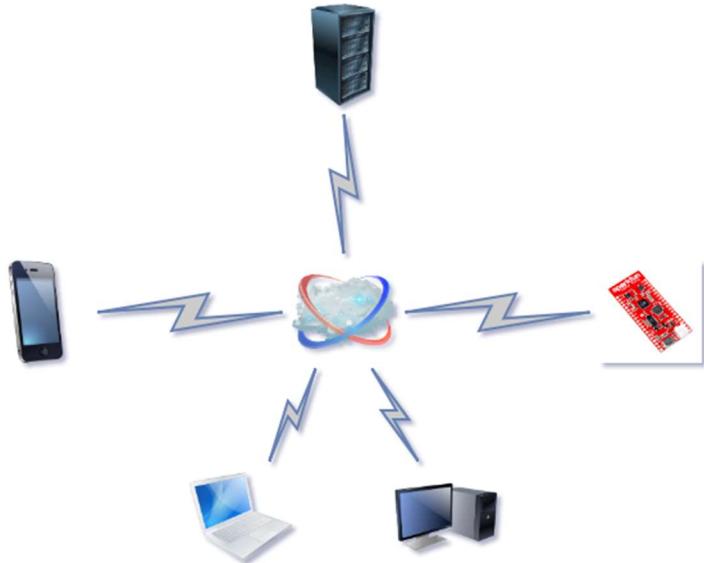


Abbildung 8 - Entwurf des Netzwerks (Version 01)

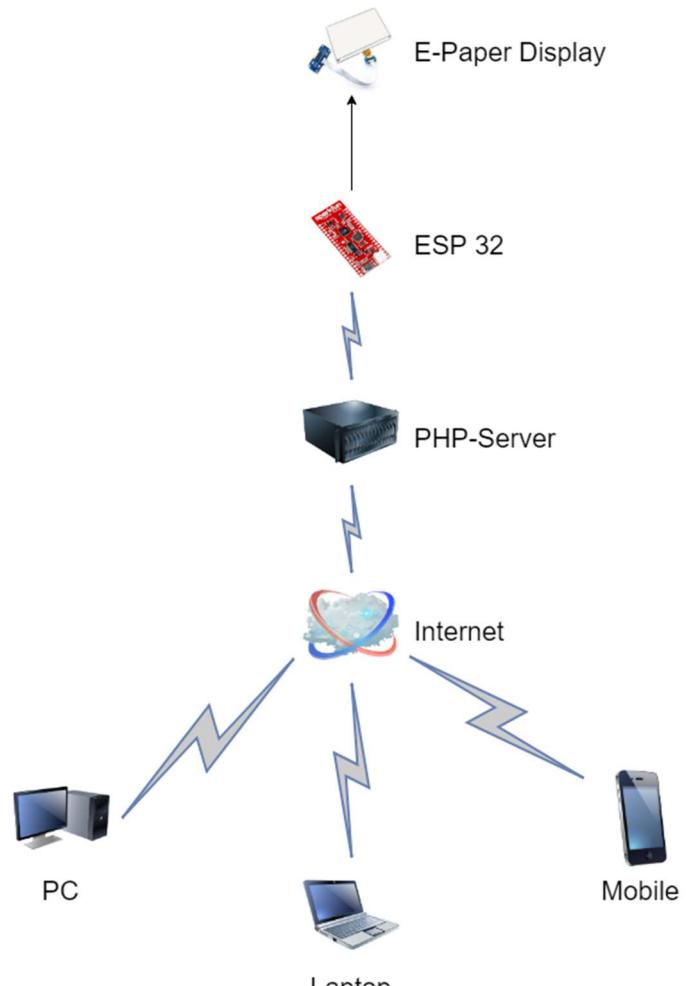


Abbildung 9 - Entwurf des Netzwerks (Version 02)

A2 – Das Webinterface

Einrichtung des Türschildes

1) Anpassung des Zustands für das Feld "Ich bin..."

- anwesend
- abwesend
- krank
- im Urlaub
- in der Pause

Prof. Dr. Thomas Korte ▾

2) Freitext erstellen

Geben Sie hier die neue Meldung ein

Hinweis: Dieses Feld leer lassen um keine Änderungen vorzunehmen!

Aktuell werden folgende Meldungen angezeigt:

28.06.18 - 15:12 Die Klausureinsicht im Fach "Objektorientierte Analyse und Design" findet am 19.07. um 14 Uhr in Raum 1.365 statt!

28.06.18 - 15:12 Am 12.07. findet keine Übung im Fach "Programmiersprachen 2" statt!

Project by Niclas Muss & Jan-Philipp Töberg
[Link zum Projekt auf GitHub](#)

Abbildung 10 - Das aktuelle Webinterface (Version 02)

A3 – Fotos der Türschilder (vorher)

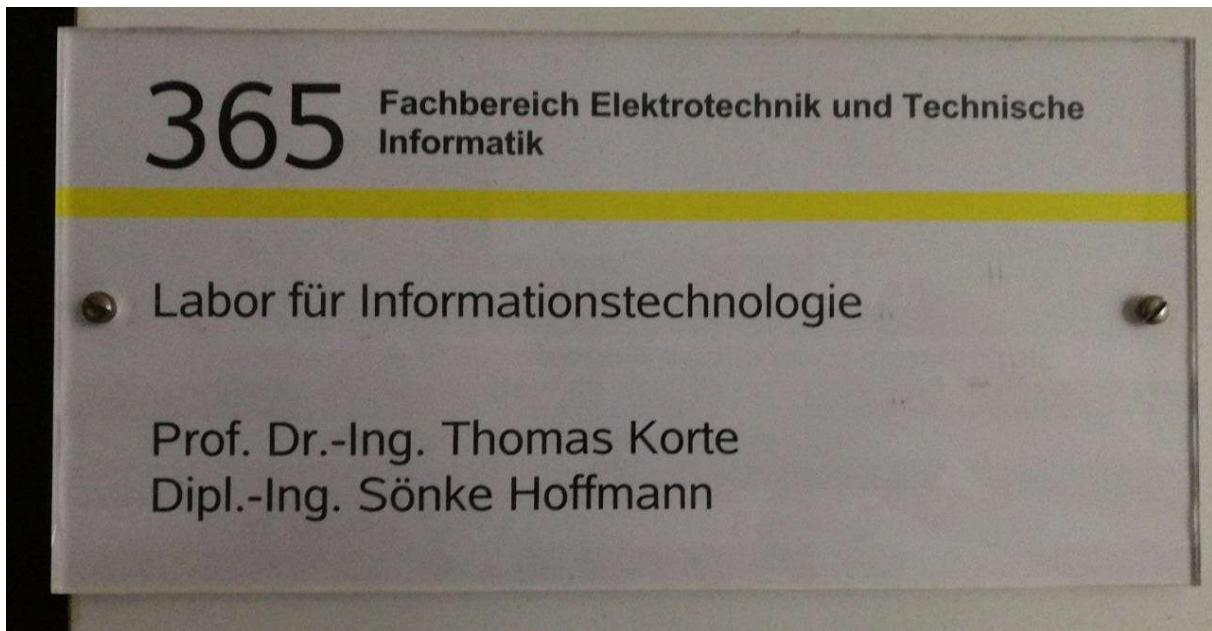


Abbildung 12 - Türschild von Prof. Korte (vorher)

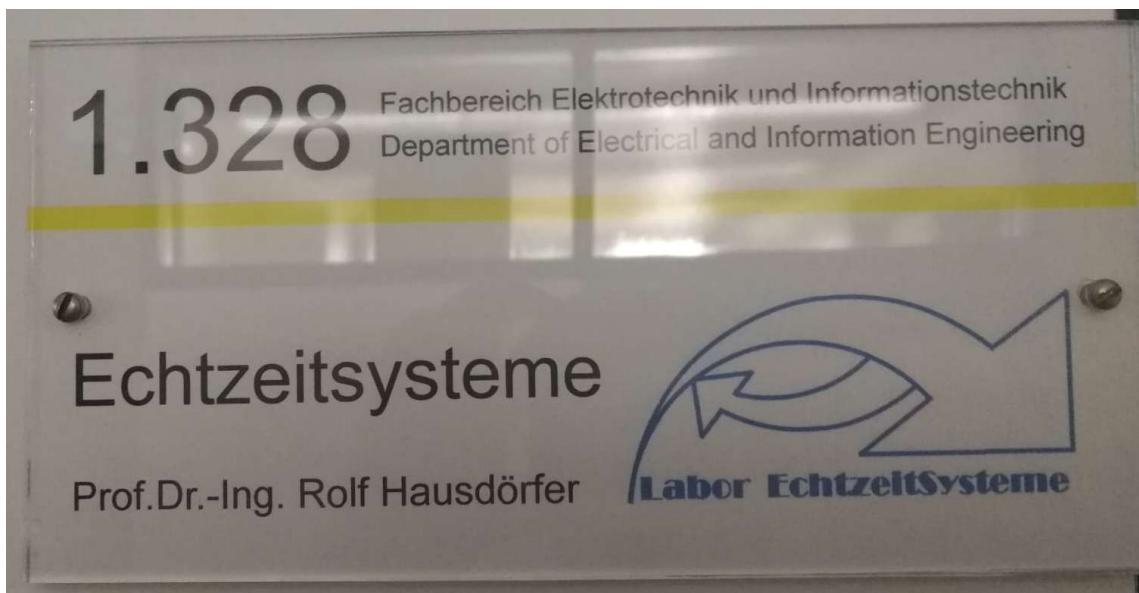


Abbildung 11 - Türschild von Prof. Hausdörfer (vorher)

A4 – Fotos der Türschilder (nachher)

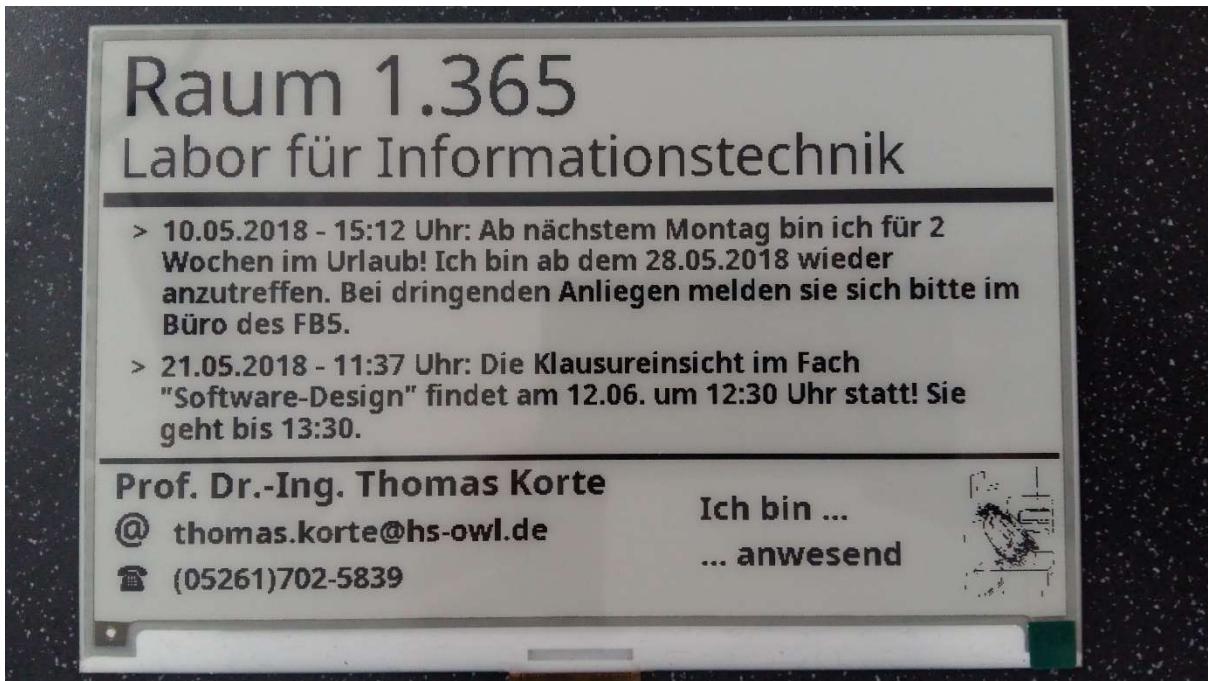


Abbildung 14 - Türschild von Prof. Korte (nachher)

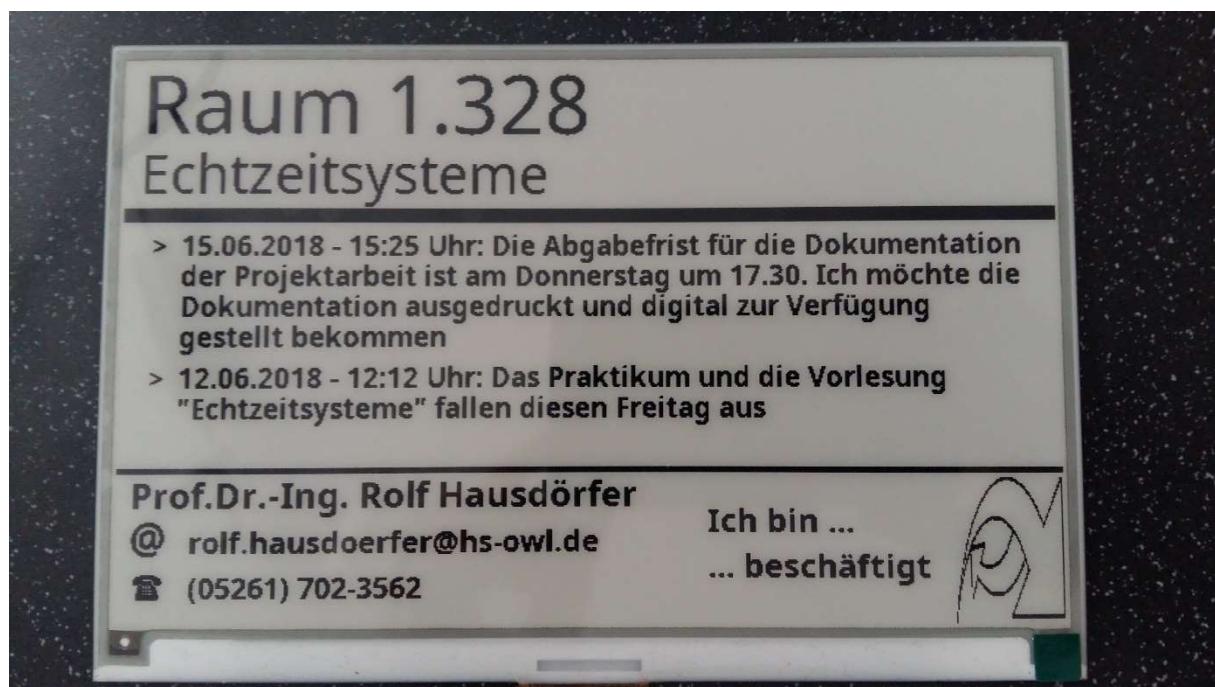


Abbildung 13 - Türschild von Prof. Hausdörfer (nachher)

A5 – Abkürzungsverzeichnis

WLAN	Wireless Local Area Network
HAT	Hardware Attached on Top
IoT	Internet of Things
IDE	Integrated Development Environment
LiPo	Lithium-Polymer
SRAM	Static Random-Access Memory
TCP/IP	Transmission Control Protocol/Internet Protocol
http	Hypertext Transfer Protocol
PHP	Hypertext Preprocessor
GD	Gif Draw
PNG	Portable Network Graphics
HTML	Hypertext Markup Language
CSV	Comma-separated values
UTF	Unicode Transformation Format
JSON	JavaScript Object Notation
JPEG	Joint Photographic Experts Group

A6 – QR-Code zum GitHub-Repository



Abbildung 15 - QR-Code für GitHub

A7 – Detaillierte Ansicht der Backlogverwaltung

Aufwand [min]	Kommentar	Status	Bearbeiter	Bearbeitet	Erledigt
		Erledigt	Niclas	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	LineHeight verbessern	Bearbeitet	Niclas	<input checked="" type="checkbox"/>	<input type="checkbox"/>
120		Bearbeitet	Janfi	<input checked="" type="checkbox"/>	<input type="checkbox"/>
240		Bearbeitet	Janfi	<input checked="" type="checkbox"/>	<input type="checkbox"/>
80		Erledigt	Janfi	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
30		Erledigt	Niclas & Janfi	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
0		Erledigt	Niclas	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
		Erstellt		<input type="checkbox"/>	<input type="checkbox"/>
150		Erledigt	Janfi	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Abbildung 17 - Detailansicht der Backlogverwaltung

PROBLEME FÜR DIE ZUKUNFT

NETZWERK PROBLEME

ID	Problem	Kategorie	Beschreibung	Aufwand [min]	Kommentar	Status	Bearbeiter	Bearbeitet	Erledigt
1	Keine Verbindung in bestimmte Netzwerke möglich	Verbindung	Das Board kann sich weder mit dem Eduroam noch mit Kortes Router verbinden	60	nicht mehr unser Problem	Erledigt	Korte	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
2	nicht-emulierter Server	Server	Langfristig brauchen wir einen echten Webserver	60	nicht mehr unser Problem	Erledigt	Korte	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
3	Webinterface braucht Internet	Server	Das Webinterface muss auf lange Sicht über das Internet erreicht werden können	30		Bearbeitet	Janfi	<input type="checkbox"/>	<input type="checkbox"/>
4	Netzwerk-Diagramm aktualisieren	Entwurf	Das Netzwerk-Diagramm muss auf den neusten Stand gebracht werden	15		Erledigt	Niclas	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
5								<input type="checkbox"/>	<input type="checkbox"/>
6								<input type="checkbox"/>	<input type="checkbox"/>
7								<input type="checkbox"/>	<input type="checkbox"/>
8								<input type="checkbox"/>	<input type="checkbox"/>
9								<input type="checkbox"/>	<input type="checkbox"/>
10								<input type="checkbox"/>	<input type="checkbox"/>

HARDWARE PROBLEME

ID	Problem	Kategorie	Beschreibung	Aufwand [min]	Kommentar	Status	Bearbeiter	Bearbeitet	Erledigt
1	Farbe anzeigen	E-Paper	Die Farbe lässt sich nicht über das onlinefähige Programm anzeigen	0	Verm. nicht fixbar, da zu wenig RAM	Erledigt	Mathis?	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
2	Flackern	E-Paper	Überprüfen ob das Flackern umgehbar ist	45		Erstellt		<input type="checkbox"/>	<input type="checkbox"/>
3	Stromversorgung	Controller	Der Controller wird über eine Batterie betrieben	0	nicht mehr unser Problem	Erledigt	Korte	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4	Case	E-Paper	Es soll ein Gehäuse für das Display 3D gedruckt werden	0	nicht mehr unser Problem	Erledigt	Korte	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
5								<input type="checkbox"/>	<input type="checkbox"/>
6								<input type="checkbox"/>	<input type="checkbox"/>
7								<input type="checkbox"/>	<input type="checkbox"/>
8								<input type="checkbox"/>	<input type="checkbox"/>
9								<input type="checkbox"/>	<input type="checkbox"/>
10								<input type="checkbox"/>	<input type="checkbox"/>

SOFTWARE PROBLEME

ID	Problem	Kategorie	Beschreibung	Aufwand [min]	Kommentar	Status	Bearbeiter	Bearbeitet	Erledigt
1	Bilder in php	PHP	Bilder, wie die Wettersymbole durch php Skripte anzeigen lassen	0		Erledigt	Niclas	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
2	Erstellung der php Skripte	PHP	Erstellung des Türschildes in einem php Skript		LineHeight verbessern	Bearbeitet	Niclas	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3	Webinterface - Ideen sammeln	Webinterface	Wie soll dem Türschild gesagt werden, was er anzeigen soll?	120		Bearbeitet	Janfi	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4	Webinterface - Funktionalität	Webinterface	Sobald die Oberfläche fertig ist, soll die Funktionalität implementiert werden	240		Bearbeitet	Janfi	<input checked="" type="checkbox"/>	<input type="checkbox"/>
5	Webinterface - Entwurf	Entwurf	Entwurf Sobald eine konkrete Idee da ist, soll die UI entworfen werden	80		Erledigt	Janfi	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6	Anforderungsanalyse / UI-Entwurf vorstellen	Gespräch	Was soll alles dargestellt werden? Passst unser Entwurf aus letzter Stunde?	30		Erledigt	Niclas & Janfi	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
7	UI-Entwurf überarbeiten	Entwurf	Nach Gespräch mit Korte Darstellung des Türschildes evtl. überarbeiten	0		Erledigt	Niclas	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
8	Schlafmodus des Controllers umgehen	ESP32	Der Controller soll auf Änderungen reagieren. Momentan benutzt er Schlafmodus			Erstellt		<input type="checkbox"/>	<input type="checkbox"/>
9	Webinterface - Oberfläche	Webinterface	Die Oberfläche soll, ohne dahinterliegende Funktionalität, angezeigt werden	150		Erledigt	Janfi	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
10								<input type="checkbox"/>	<input type="checkbox"/>

DOKUMENTATION

ID	Problem	Kategorie	Beschreibung	Aufwand [min]	Kommentar	Status	Bearbeiter	Bearbeitet	Erledigt
1	Kolloquium vorbereiten	Presentation	Erstellen der Präsentationen für das Kolloquium am 05.07.	150		Zugewiesen	Niclas & Janfi	<input type="checkbox"/>	<input type="checkbox"/>
2	Kapitel 1-1 schreiben	Ausarbeitung	Projektbeschreibung / Aufgabenstellung	20		Bearbeitet	Niclas	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3	Kapitel 1-2 schreiben	Ausarbeitung	Projektkontext	45		Bearbeitet	Janfi	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4	Kapitel 2-1 schreiben	Ausarbeitung	Projektophasen	60		Bearbeitet	Janfi	<input checked="" type="checkbox"/>	<input type="checkbox"/>
5	Kapitel 2-1-1 schreiben	Ausarbeitung	Projekttressourcen - ESP32 Microcontroller	50		Bearbeitet	Niclas	<input checked="" type="checkbox"/>	<input type="checkbox"/>
6	Kapitel 2-2-2 schreiben	Ausarbeitung	Projekttressourcen - Waveshare eKindle Display	50		Bearbeitet	Niclas	<input checked="" type="checkbox"/>	<input type="checkbox"/>
7	Kapitel 2-3-1 schreiben	Ausarbeitung	Gemeinsames Arbeiten - Versionsverwaltung über GitHub	60		Bearbeitet	Janfi	<input checked="" type="checkbox"/>	<input type="checkbox"/>
8	Kapitel 2-3-2 schreiben	Ausarbeitung	Gemeinsames Arbeiten - Backlogverwaltung über Google Spreadsheets	45		Zugewiesen	Niclas	<input type="checkbox"/>	<input type="checkbox"/>
9	Kapitel 3-1 schreiben	Ausarbeitung	Beschreibung des CT-Programms	80		Zugewiesen	Janfi	<input type="checkbox"/>	<input type="checkbox"/>
10	Kapitel 3-2 schreiben	Ausarbeitung	Beschreibung des CT-Webservers	80		Zugewiesen	Janfi	<input type="checkbox"/>	<input type="checkbox"/>
11	Kapitel 3-3 schreiben	Ausarbeitung	Anforderungsanalyse	45		Zugewiesen	Janfi	<input type="checkbox"/>	<input type="checkbox"/>
12	Kapitel 4-1 schreiben	Ausarbeitung	Entwurf des Anzeigebilds	45		Zugewiesen	Niclas	<input type="checkbox"/>	<input type="checkbox"/>
13	Kapitel 4-2 schreiben	Ausarbeitung	Entwurf des Webinterfaces	120		Bearbeitet	Janfi	<input checked="" type="checkbox"/>	<input type="checkbox"/>
14	Kapitel 4-3 schreiben	Ausarbeitung	Entwurf des Netzwerks	60		Bearbeitet	Niclas	<input checked="" type="checkbox"/>	<input type="checkbox"/>
15	Kapitel 5-1 schreiben	Ausarbeitung	PHP Skript zur Anzeige eines Bildes	200		Zugewiesen	Niclas	<input type="checkbox"/>	<input type="checkbox"/>
16	Kapitel 5-2 schreiben	Ausarbeitung	Hinzufügen von Texten über Webinterface	120		Zugewiesen	Janfi	<input type="checkbox"/>	<input type="checkbox"/>
17	Kapitel 6-1 schreiben	Ausarbeitung	Self/Ist-Vergleich	75		Zugewiesen	Janfi	<input type="checkbox"/>	<input type="checkbox"/>
18	Kapitel 6-2 schreiben	Ausarbeitung	Ausblick	45		Zugewiesen	Niclas	<input type="checkbox"/>	<input type="checkbox"/>
19	Ausarbeitungen zusammenführen	Ausarbeitung	Ausarbeitungen müssen zusammengeführt & Formate angepasst werden	80		Zugewiesen	Janfi	<input type="checkbox"/>	<input type="checkbox"/>
20								<input type="checkbox"/>	<input type="checkbox"/>

Abbildung 16 - komplette Backlogverwaltung (Stand: 27.06.2018)

A8 – Artikel aus der c't



Ausdauer! Infotafel

Drahtloses Türschild mit E-Paper-Display

Ein Belegungsplan für den Konferenzraum, die Anwesenheit von Kollegen im Büro oder der Putzplan für die WG: Ein digitales Türschild ersetzt Zettelwirtschaft und Papieraushänge.

Von Jan Mahn



In digitales Türschild mit einem LCD ist schnell gebaut: eine Wandhalterung mit einem alten Tablet, darauf eine App oder eine Webseite im Vollbild. Das Problem ist der Stromverbrauch, denn selbst mit einem neuen Akku wird die Lösung nur wenige Tage durchhalten. Schließlich dürstet der Bildschirm ununterbrochen nach Strom. Wer ein Türschild plant, das mehrere Monate mit einer Akkuladung hält, kommt an E-Paper-Displays nicht vorbei. Diese benötigen nur dann Energie, wenn der Bildinhalt wechselt. Kombiniert mit dem Tiefschlafmodus des ESP32 entsteht ein digitales Türschild mit langer Akkulaufzeit.

Für das E-Paper-Türschild haben wir ein 7,5-Zoll großes Display verwendet, das der Hersteller Waveshare mit einem Aufsteckmodul für den Raspberry Pi anbietet. Das Panel kostet beim deutschen Händler 58 Euro, hat eine Auflösung von 640 × 384 Pixel und läuft, wie der ESP32, mit 3,3 V. Vom selben Hersteller gibt es auch Varianten mit 2,9 und 4,2 Zoll.

Um Strom zu sparen, wollen wir dem Mikrocontroller möglichst wenig Arbeit überlassen: Es soll einmal stündlich aus dem Tiefschlaf aufwachen, sich mit dem WLAN verbinden, per HTTP-Anfrage von einem Webserver das anzugebende Bild als Rohdaten herunterladen, dieses auf dem Display anzeigen und sich wieder in den Tiefschlaf begeben. Da die Gestaltung des Bildes auf dem Webserver erledigt wird, können Sie die Inhalte jederzeit austauschen, ohne neuen Code auf den ESP32 zu flashen. Ein Webserver kann auch mehrere Schilder mit Bildern versorgen und beispielsweise für jedes Konferenzraumschild ein Bild mit der aktuellen Belegung zusammenbauen.

Prototyp

Den ersten Aufbau sollten Sie mit einem ESP32-Entwicklerboard mit integriertem Programmieradapter zusammenstecken. Waveshare liefert eine Kabelpeitsche mit Jumper-Kabeln mit, die Belegung finden Sie in der Tabelle rechts oben.

Ist die Arduino-IDE mit der ESP32-Erweiterung und unserer Bibliothek Bascamp eingerichtet (siehe S. 64), kann die Arbeit am Code für das Türschild beginnen. Installieren Sie zu Beginn die beiden externen Bibliotheken „GxEPD“ und „Adafruit_GFX“. Die Links dafür finden Sie unter ct.de/yrzv. Laden Sie den Inhalt der Repositories als zip-Dateien herunter und entpacken sie im Ordner „libraries“ im Arduino-Verzeichnis. Starten Sie anschließend die IDE neu. Die Bibliotheken erledigen die Kommunikation mit dem Display. Laden Sie zunächst ein Beispielprojekt, das die Entwickler von „GxEPD“ mitgeliefert haben, um die Verkabelung zu prüfen. Sie finden es in der Arduino-IDE unter „Datei/Beispiele/GxEPD/GxEPD_SPI_TestExample“. In den Zeilen 46 bis 56 (Zeilennummern aktivieren Sie unter „Datei/Voreinstellungen“) binden Sie die für das Display passende Bibliothek ein und kommentieren Sie alle anderen mit „//“ aus. Für das 7,5-Zoll-Display entfernen Sie die Kommentarzeichen für `#include <GxGDEW075T8/GxGDEW075T8.cpp>`.

Pinbelegung E-Paper-Display	
Display	ESP32
BUSY	4
RST	16
DC	17
CS	5
CLK	18
DIN	23
GND	GND
3.3V	3.3V

Schließen Sie das ESP-Entwicklerboard per USB an und flashen Sie das Beispiel auf den Mikrocontroller - stimmt die Verkabelung, zeigt das Display ein Testprogramm mit Bildern und Texten an. Die Funktion zur Bildausgabe, die in der Biethel mitgeliefert wird, verlangt als Bildformat ein Array mit Hexadezimalwerten - solche Bilder können Sie zwar problemlos auf den ESP32 flashen. Um es per HTTP zu übertragen, ist es aber ungeeignet.

Pixelstrom

Das E-Paper-Display stellt bauartbedingt nur monochrome Bilder dar, kennt also nur zwei Zustände für einen Pixel - ein oder aus. Das Display hat 640 Pixel in der Breite und 384 in der Höhe, insgesamt also 245.760 Bildpunkte. Unser Bildformat ist auf das Wesentliche reduziert: Beginnend in der linken oberen Ecke werden die Schaltzustände der Pixel zu einem String zusammengefügt - je acht Bits werden zu einem Byte zusammengefasst. Aus

Pixelfolge 00101011 wird beispielsweise das ASCII-Zeichen +. Am Ende der Zeile folgen kommentarlos die Pixel der nächsten Zeile.

Bevor die Hardware Bilddaten empfangen kann, müssen diese erst einmal in ein geeignetes Datenformat umgewandelt werden. Auf dem Webserver haben wir das mit PHP realisiert (siehe Listing auf S. 70).

Eingelesen wird das Bild „bild.png“, das von der Funktion `createMonochromeImage()` Zeile für Zeile zerlegt wird. Für jedes Pixel prüft das Programm, ob in einem der Farbkanäle Rot, Grün oder Blau ein Farbwert gesetzt ist und hängt das Zeichen 0 oder 1 an den String an. Dieser Code ersetzt bei aufwendigen farbigen Bildern keine Umwandlung durch ein Bildbearbeitungsprogramm - jeder nicht weiße Pixel wird auf dem Display schwarz. Ist die Länge des Strings durch acht teilbar, wird das nächste Byte erzeugt und an den Ausgabe-String angehängt.

Aus diesem Material muss der ESP32 jetzt wieder ein Bild zusammensetzen. Nachdem er sich den String vom Webserver geholt hat, liest er ihn Byte für Byte ein, zerlegt die Bytes wieder in Bits und schaltet die Pixel des E-Paper-Displays einzeln ein oder aus. Innerhalb der Schleife ist das aktuelle Byte in der Variable `byte`:

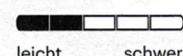
```
for (int b = 7; b >= 0; b--) {
    int bit = bitRead(byte, b);
    if (bit == 1) {
        display.drawPixel(x, y,GxEPD_BLACK);
    } else {
        display.drawPixel(x, y,GxEPD_WHITE);
    }
    x++;
    if(x == 640) {
        y++;
        x = 0;
    }
}
```

Einkaufsliste Steckdose

4 Stunden (mit Gehäusebau)

80 €

- ESP32
- Waveshare 7.5inch E-Ink display HAT for Raspberry Pi
- Multiplex
- Lötkolben
- Stichsäge



leicht schwer

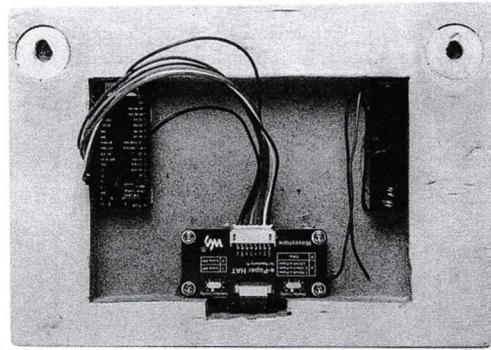
Aus den Bytes werden stückweise wieder acht Bildpunkte, die direkt aufs Display geschrieben werden. Nach jeweils 640 Spalten beginnt die nächste Zeile. Sollten Sie ein kleineres Modell einsetzen, müssen Sie die Breite aus dem jeweiligen Datenblatt des Herstellers entnehmen und den Wert eintragen. Sind alle Bytes eingelesen, folgt der Befehl `display.update()`, damit das Display die neuen Bildpunkte anzeigt.

Tiefschlaf

Die HTTP-Anfrage und der Bildaufbau haben nur wenige Sekunden gedauert. Jetzt ist es für den ESP32 Zeit, sich wieder schlafen zu legen. Soll er das nächste Bild in 60 Minuten abrufen, bekommt er eine Wartezeit von 3.600.000.000 Mikroskunden (3600 Sekunden): `esp_sleep_`



Damit das digitale Türschild an der Wand seiner Arbeit nachgehen kann, bekommt es ein Gehäuse aus Holz.



Im Hohlräum auf der Rückseite finden ESP32 und Batteriefach Platz.

Praxis | Smarte Helfer: Türschild

```
$im = imagecreatefrompng("bild.png");
echo createMonochromeImage($im);
function createMonochromeImage($im) {
    $bits = "";
    $bytes = "";
    $pixelcount = 0;
    for ($y = 0; $y < imagesy($im); $y++) {
        for ($x = 0; $x < imagesx($im); $x++) {
            $rgb = imagecolorat($im, $x, $y);
            $r = ($rgb >> 16) & 0xFF;
            $g = ($rgb >> 8) & 0xFF;
            $b = $rgb & 0xFF;
            $gray = ($r + $g + $b) / 3;
            if ($gray < 0xFF) {
                $bits .= "1";
            } else {
                $bits .= "0";
            }
            $pixelcount++;
            if ($pixelcount % 8 == 0) {
                $bytes .= pack("H*", str_pad(base_convert($bits, 2, 16), 2, "0", STR_PAD_LEFT));
                $bits = "";
            }
        }
    }
    return $bytes;
}
```

Für das E-Paper-Display berechnet das PHP-Skript aus einer PNG-Datei einen String aus Bytes mit je acht monochromen Pixeln.

`enable_timer_wakeup(3600000000)` und verabschiedet sich anschließend mit `esp_deep_sleep_start()` in den Tiefschlaf.

Rahmenbedingungen

Den Code zum Bildaufbau haben wir zusammen mit unserer Bibliothek Basecamp bereitgestellt, sodass Ihr erstes Türschild schnell einsatzbereit ist. Das Repository enthält im Ordner „Server“ Anregungen, was der Webserver darstellen könnte. Das gesamte Projekt finden Sie unter ct.de/yrzv. Flashen Sie das Programm auf den ESP32, verbinden Sie sich mit dem WLAN, das dieser bereitstellt, und öffnen Sie die Konfigurationsoberfläche (siehe S. 67). In der Weboberfläche geben Sie jetzt die Adresse des Servers und den Pfad zum Bild sowie die gewünschte Schlafzeit ein. Anschließend startet der Mikrocontroller neu und beginnt mit der Arbeit.

Sandwich-Gehäuse

Beim Bau des Gehäuses für das 7,5 Zoll große Display haben wir ein Design ent-

wickelt, das Sie ohne 3D-Drucker und CNC-Fräse nachbauen können. Es ist schichtweise zusammengeleimt und besteht aus einer Schicht Sperrholz (3 mm), zwei Lagen Hartfaserplatte (4 mm) und einer Schicht Multiplex (18 mm). Die Zeichnung mit allen Maßen finden Sie im Projekt-Repository im Ordner „case“, die Verarbeitung funktioniert mit Stich- und Kreissäge (oder Zuschnitten aus dem Baumarkt) auch mit wenig Werkzeug. Legen Sie das Display in seine Aussparung in der zweiten Ebene und verbinden Sie die Schichten mit Holzleim. Das Flachbandkabel führen Sie bis in die letzte Ebene durch. Dort finden ein ESP32 und ein Batteriefach Platz. Mit eingelassenen Schlüssellochblechen befestigen Sie das Schild elegant an der Wand. Um das Projekt über einen Akku mit Strom zu versorgen, finden Sie zwei Ansätze im Projekt „Türsensör“ auf Seite 76.

Luftpost-Update

Das Überspielen der Firmware auf den ESP32 ist kein Problem, solange Sie ein

ESP-Entwicklerboard mit integriertem USB-Adapter verwenden und die Hardware neben dem Computer liegt. Hängt das Schild erst einmal fertig montiert an der Wand, sind Veränderungen am Code aber sehr aufwendig. Wenn Sie das Entwicklerboard durch einen einzelnen ESP32 ersetzt haben, müssten Sie für jedes Update einen Programmieradapter anklammern. Dankenswerterweise haben die Entwickler die Möglichkeit integriert, Updates per WLAN einzuspielen. Im Programm müssen Sie dafür nur wenige Zeilen Code ergänzen:

```
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>

void setup() {
    // [... ] WLAN-Verbindung herstellen
    ArduinoOTA.setPassword("secret");
    ArduinoOTA.begin();
}

void loop() {
    ArduinoOTA.handle();
}
```

Überspielen Sie dieses Programm auf klassischem Weg per USB. Ist der ESP32 mit dem gleichen WLAN verbunden wie der Computer mit der Arduino-IDE, finden Sie ihn im Menü unter „Werkzeuge/Ports“ unterhalb der seriellen Ports – zu erkennen an seiner IP-Adresse. Achten Sie darauf, dass in jeder veränderten Version des Programms die Zeilen für das OTA-Update erhalten bleiben. Fehlen diese, haben Sie sich ausgesperrt und müssen wieder den USB-Programmer bemühen.

In unserer Bibliothek Basecamp sind OTA-Updates bereits integriert und aktiviert. Um ein Kennwort zu setzen, verwenden Sie die Zeile `configuration.set("OTAPassword", "IHR KENNWORD")`. Zum Deaktivieren der Funktion `configuration.set("OTAAactive", "false")`.

Weiterdenken

Mit den veröffentlichten Beispielen und dem Gehäuse haben wir einige Anregungen zusammengestellt – wenn Sie eigene Ideen für Inhalte oder Gehäuse-Baupläne für die kleineren Displays haben, lassen Sie es uns wissen oder erstellen Sie einen Pull-Request bei GitHub. (jam@ct.de)

Repository und Downloads: ct.de/yrzv

A9 – Die „index.php“-Datei

```

1 <?php
2 //To activate productionMode (display entering deep sleep), set http-header
3 X-productionMode: true
4 #header("X-productionMode: true");
5 //To stop productionMode (no deep sleep, web config), set http-header
6 X-productionMode: false
7 #header("X-productionMode: false");
8
9 // Set the sleep interval for the doorsigns via the server
10 #header("X-sleepInterval: 60 ");
11
12 error_reporting(E_ERROR);
13 # Supported displays:
14 # 1.54 inches: https://www.waveshare.com/wiki/1.54inch_e-Paper_Module
15 # 2.9 inches: https://www.waveshare.com/wiki/2.9inch_e-Paper_Module
16 # 4.2 inches: https://www.waveshare.com/wiki/4.2inch_e-Paper_Module
17 # 7.5 inches: https://www.waveshare.com/wiki/7.5inch_e-Paper_HAT
18 const DISPLAYS = array( "7.5"=>array("size"=>"640x384", "rotate"=>"false"),
19                         "7.5bwr"=>array("size"=>"640x384", "rotate"=>"false",
20                                         "red"=>"true"),
21                         "4.2"=>array("size"=>"400x300", "rotate"=>"false"),
22                         "4.2bwr"=>array("size"=>"400x300", "rotate"=>"false",
23                                         "red"=>"true"),
24                         "2.9"=>array("size"=>"296x128", "rotate"=>"true"),
25                         "1.5"=>array("size"=>"200x200", "rotate"=>"true")
26 );
27
28 // Use Googles Noto fonts as the default font face
29 $DEFAULT_FONT = array(
30     "test"=>realpath("./fonts/Wingdings_3.ttf"),
31     "regular"=>realpath("./fonts/noto/NotoSans-Regular.ttf"),
32     "bold"=>realpath("./fonts/noto/NotoSans-Bold.ttf"),
33     "italic"=>realpath("./fonts/noto/NotoSans-Italic.ttf"),
34     "bolditalic"=>realpath("./fonts/noto/NotoSans-BoldItalic.ttf"),
35     "symbols"=>realpath("./fonts/noto/NotoSansSymbols-Regular.ttf"),
36     "emoji"=>realpath("./fonts/noto/NotoEmoji-Regular.ttf"),
37     "weathericons"=>realpath("./fonts/weathericons-regular-webfont.ttf")
38 );
39
40 // To use LiberationSans font, uncomment the following lines
41 /*
42 $DEFAULT_FONT = array(
43     "regular"=>realpath("./fonts/LiberationSans-Regular.ttf"),
44     "bold"=>realpath("./fonts/LiberationSans-Bold.ttf"),
45     "italic"=>realpath("./fonts/LiberationSans-Italic.ttf"),
46     "weathericons"=>realpath("./fonts/weathericons-regular-webfont.ttf")
47 );
48 */
49
50 const THRESHOLDS = array("black" => 150, "red" => 240);
51
52 if (!extension_loaded('gd')) {
53     echo "GD library is not installed. Please install GD on your server
54         (http://php.net/manual/de/image.installation.php);
55     exit;
56 }
57
58 //Function to check if FreeType is installed. Not needed by static_image
59 function checkFreeType() {
60     $gdInfo = gd_info();
61     if($gdInfo['FreeType Support'] != 1){
62         echo "FreeType is not enabled. FreeType is needed for creating text in
63             images (http://php.net/manual/de/function.imagettftext.php);
64         exit;
65     }
66
67 if(strlen($_GET['scale']) AND is_numeric($_GET['scale'])) {
68     $scale = $_GET['scale'];
69 }else{
70     $scale = $GET['scale'] = 32;
71 }
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
737
738
739
739
740
741
742
743
744
745
745
746
747
747
748
749
749
750
751
752
753
754
755
756
756
757
758
758
759
759
760
761
762
763
764
765
766
766
767
767
768
768
769
769
770
771
772
773
774
775
776
776
777
777
778
778
779
779
780
781
782
783
784
785
786
786
787
787
788
788
789
789
790
791
792
793
794
795
796
796
797
797
798
798
799
799
800
801
802
803
804
805
806
806
807
807
808
808
809
809
810
811
812
813
814
815
816
816
817
817
818
818
819
819
820
821
822
823
824
825
826
826
827
827
828
828
829
829
830
831
832
833
834
835
836
836
837
837
838
838
839
839
840
841
842
843
844
845
845
846
846
847
847
848
848
849
849
850
851
852
853
854
855
856
856
857
857
858
858
859
859
860
861
862
863
864
865
866
866
867
867
868
868
869
869
870
871
872
873
874
875
876
876
877
877
878
878
879
879
880
881
882
883
884
885
886
886
887
887
888
888
889
889
890
891
892
893
894
895
895
896
896
897
897
898
898
899
899
900
901
902
903
904
905
905
906
906
907
907
908
908
909
909
910
911
912
913
914
915
915
916
916
917
917
918
918
919
919
920
921
922
923
924
925
925
926
926
927
927
928
928
929
929
930
931
932
933
934
935
935
936
936
937
937
938
938
939
939
940
941
942
943
944
944
945
945
946
946
947
947
948
948
949
949
950
951
952
953
954
954
955
955
956
956
957
957
958
958
959
959
960
961
962
963
964
964
965
965
966
966
967
967
968
968
969
969
970
971
972
973
974
974
975
975
976
976
977
977
978
978
979
979
980
981
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1650
1651
1651
1652
1652
1653
1653
1654
1654
1655
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1660
1661
1661
1662
1662
1663
1663
1664
1664
1665
1665
1666
1666
1667
1667
1668
1668
1669
1669
1670
1670
1671
1671
1672
1672
1673
1673
1674
1674
1675
1675
1676
1676
1677
1677
1678
1678
1679
1679
1680
1680
1681
1681
1682
1682
1683
1683
1684
1684
1685
1685
1686
1686
1687
1687
1688
1688
1689
1689
1690
1690
1691
1691
1692
1692
1693
1693
1694
1694
1695
1695
1696
1696
1697
1697
1698
1698
1699
1699
1700
1700
1701
1701
1702
1702

```

```

67 }
68
69 $displayType = $_GET['display'];
70 if(!isset(DISPLAYS[$displayType])){
71     echo ("Not a valid display size. <br />");
72     echo ("display=");
73     foreach (array_keys(DISPLAYS) as $display_key) {
74         echo ($display_key.", ");
75     }
76     echo ("]");
77     exit;
78 }
79 $hasRed = DISPLAYS[$displayType]['red'];
80
81 $professor = htmlspecialchars($_GET["professor"]);
82
83 //Read existing contents
84 $contents = scandir('contents');
85
86 if(!count($contents)){
87     echo "No content definitions";
88     exit;
89 }
90
91 foreach ($contents as $content) {
92     $contentFile = pathinfo("contents/".$content);
93
94     if($contentFile['extension'] == "php"){
95         $allContents[$contentFile['filename']] = "contents/".$content,
96     }
97 }
98
99 $selectedContent = $allContents[$_GET['content']];
100
101 $displayWidth = explode("x",DISPLAYS[$displayType]['size'])[0];
102 $displayHeight = explode("x",DISPLAYS[$displayType]['size'])[1];
103 $im = imagecreate($displayWidth, $displayHeight);
104 $background_color = ImageColorAllocate ($im, 255, 255, 255);
105 $black = ImageColorAllocate ($im, 0, 0, 0);
106 $red = ImageColorAllocate ($im, 0xFF, 0x00, 0x00);
107
108
109 if(is_file($selectedContent)){
110     include($selectedContent);
111 }else{
112     echo "Not a valid content.";
113     imagedestroy($im);
114     exit;
115 }
116
117
118 if($_GET['debug'] == 'true'){
119     header("Content-type: image/png");
120     imagepng($im);
121 }
122 else{
123     if(DISPLAYS[$displayType]['rotate'] == "true"){
124         $im = imagerotate($im, 90, 0);
125     }
126
127     $im = imagerotate($im, 0, 0);
128     //if you are using an older version of GD library you have to rotate the image
129     //360°. Otherwise you get a white image due to a bug in GD library. Uncomment next
130     //lines:
131     //$im = imagerotate($im, 180, 0);
132     //$im = imagerotate($im, 180, 0);
133
134     echo rawImage($im, $hasRed);
135 }
136 imagedestroy($im);
137

```

```

137
138     function rawImage($im, $hasRed) {
139         $bits = "";
140         $bytes = "";
141         $pixelcount = 0;
142
143         for ($y = 0; $y < imagesy($im); $y++) {
144             for ($x = 0; $x < imagesx($im); $x++) {
145
146                 $rgb = imagecolorat($im, $x, $y);
147                 $r = ($rgb >> 16) & 0xFF;
148                 $g = ($rgb >> 8 ) & 0xFF;
149                 $b = $rgb & 0xFF;
150                 $gray = ($r + $g + $b) / 3;
151
152                 if($hasRed == "true") {
153
154                     if((($r >= THRESHOLDS['red']) && ($g < 50) && ($b < 50)) {
155                         $bits .= "01";
156                     } else {
157                         if ($gray < THRESHOLDS['black']) {
158                             $bits .= "11";
159                         }else {
160                             $bits .= "00";
161                         }
162                     }
163                     $pixelcount = $pixelcount+2;
164                 }else{
165                     if ($gray < THRESHOLDS['black']) {
166                         $bits .= "1";
167                     }else {
168                         $bits .= "0";
169                     }
170                     $pixelcount++;
171                 }
172
173
174                 if ($pixelcount % 8 == 0) {
175                     $bytes .= pack('H*', str_pad(base_convert($bits, 2, 16),2, "0",
176                     STR_PAD_LEFT));
177                     $bits = "";
178                 }
179             }
180         }
181         $size = strlen($bytes);
182
183         header("Content-length: $size");
184         return $bytes;
185     }
186 ?>
187

```

A10 – Die „turschild_V1.php“-Datei

```

1  <?php
2  	/**
3  	*  Projektarbeit Sommersemester2018 - Drahtloses Türschild mit E-Paper-Display
4  	*  Author: Niclas Muss
5  	*  Version: 1.0
6  	*  Letzte Änderung: 28.06.2018
7  	*
8  	*  Dieses Skript zeichnet, wenn es in der index.php aufgerufen wird, auf die
9  	*  Bilddatei $im
10  *  Gezeichnete Zeichnung ist ein Türschild, welches in Abhängigkeit des Parameters
11  "Professor" ein individuelles Türschild
12  */
13  checkFreeType();
14
15 //Variablendefinition
16
17 $schriftGroesse = 42; //Schriftgröße der Überschrift. Anderer Text wird anhand
18 dieser Größe skaliert
19 $cursorY += $schriftGroesse*1.2; //Variable, die eine Höhe in Pixeln angibt
20
21 If($hasRed){
22     $bild = "contents/Bilder/".$professor."_bwr.jpg";
23     if($_GET['debug'] == 'true'){
24         $red = ImageColorAllocate($im, 0xFF, 0xFF, 0x00);
25     }
26 } else{
27     $red = $black;
28     $bild = "contents/Bilder/".$professor.".jpg";
29 }
30
31 //Einlesen der csv Datei mit den Infos zur Person
32 if (($handle = fopen($professor.".csv", "r")) !== FALSE) {
33     while (($reihe = fgetcsv($handle, 1000, ",")) !== FALSE) {
34         if ($reihe[1] !== ''){
35             $konstanten[] = $reihe[1]; //Das 2. Element jeder Reihe ist der
36             //variable Wert
37         }
38     }
39     fclose($handle);
40 } else {
41     die("Problem beim lesen der csv-Datei \"professor\"");
42 }
43
44 //Einlesen der csv Datei mit dem Status und den Meldungen
45 if (($handle = fopen($professor."_status.csv", "r")) !== FALSE) {
46     $status = fgetcsv($handle, 1000, ",");
47     fclose($handle);
48     if ($status[0] == ''){
49         die("Es ist kein Anwesenheitsstatus bekannt");
50     }
51 } else {
52     die("Problem beim lesen der csv-Datei \"professor_status\"");
53 }
54
55
56 /**
57 *  Der obere Teil des Anzeigebildes.
58 *  Zeigt die Raumnummer und den Raumnamen des Büros an.
59 *  eingelesen werden diese datein aus der csv, die zur Einrichtung benutzt wird.
60 *  Wird durch einen gelben oder schwarzen Balken vom Rest der Anzeige getrennt.
61 */
62 imagedttfText($im, $schriftGroesse, 0, 10, $cursorY, $black,
63 $DEFAULT_FONT['regular'], "Raum ".$konstanten[0]);
64 $cursorY += $schriftGroesse;
65 $schriftGroesse = $schriftGroesse/1.5;
66 imagedttfText($im, $schriftGroesse, 0, 10, $cursorY, $black,
$DEFAULT_FONT['regular'], $konstanten[1]);

```

```

67     $cursorY += 15;
68     imagesetthickness($im, 10);
69     imageline ($im , 0 , $cursorY , $displayWidth , $cursorY , $red );
70
71
72 /**
73 *   Der mittlere Teil des Anzeigebildes.
74 *   Zeigt einen kleinen Infomonitor an, auf dem beliebig viele Meldungen
75 angezeigt werden können.
76 *   Zeigt nur so viel Nachrichten an, wie auf den Bildschirm passen, wobei die
77 neuste Immer oben steht.
78 */
79 $schriftGroesse = $schriftGroesse/2;
80 $nachricht = array();
81 $cursorY += $schriftGroesse*1.5;
82 for ($nachrichtCount=1; $nachrichtCount<count($status); $nachrichtCount++) {
83 //Äußere For-Schleife: Wiederholung pro Nachricht
84     $woerter = explode(" ",$status[$nachrichtCount]);//Array der Wörter aus dem
85 eingegabenem Text
86     $woerterAnzahl = count($woerter); //Anzahl der Wörter
87     $zeile = '';//Die momentan beschriebene Zeile (benutzt zum Messen)
88     $text = '';//Die momentan druckbare Zeile
89     $cursorY += $schriftGroesse*0.5;
90     $nachrichtHoehe = $cursorY;
91     for($woerterCount=0; $woerterCount<$woerterAnzahl; $woerterCount++) {
92 //Innere For Schleife: Wiederholung pro Wort
93         $zeile .= $woerter[$woerterCount];
94         $dimensionen = imagegetbbox($schriftGroesse, 0, $DEFAULT_FONT['bold'],
95         $zeile);
96         $zeileLaenge = $dimensionen[2] - $dimensionen[0]; //Länge einer Zeile
97
98         if ($zeileLaenge > $displayWidth-60) { //Wenn die Breitenbeschränkung
99             überschritten wird, soll eine neue Zeile gestartet werden
100             $nachricht[] = $text; //Text dem Nachrichtenarray hinzufügen
101             $nachrichtHoehe +=$schriftGroesse*1.5; //Höhe der Nachricht um die
102             aktuelle Zeile erhöhen
103             $zeile = $woerter[$woerterCount]. ' '; //Zeile und Text zurücksetzen
104             $text = $woerter[$woerterCount]. ' ';
105         }
106         else {
107             $zeile.=' ';
108             $text.=$woerter[$woerterCount]. ' ';
109         }
110     if ($nachrichtHoehe > $displayHeight-105){ //Wenn die Höhenbeschrenkung
111         überschritten wird, sollen keine Nachrichten mehr ausgegeben werden
112         break;
113     }
114     //Ausbabe einer Nachricht
115     imagegettext($im, $schriftGroesse, 0, 20, $cursorY, $black,
116     $DEFAULT_FONT['bold'], '>');
117     $nachricht[] = $text; //letzte Zeile in das Nachrichtenarray speichern
118     for ($woerterCount=0; $woerterCount<count($nachricht); $woerterCount++) {
119         $dimensionen = imagegetbbox($schriftGroesse, 0, $DEFAULT_FONT['bold'],
120         $nachricht[$woerterCount]);
121         imagegettext($im, $schriftGroesse, 0, 40, $cursorY, $black,
122         $DEFAULT_FONT['bold'], $nachricht[$woerterCount]);
123         $cursorY += $schriftGroesse*1.5;
124     }
125     $nachricht = array(); //Nachrichtenarray zurücksetzen
126 }
127
128 /**
129 *   Der unter Teil des Anzeigebildes.
130 *   Zeigt Infos über den Professor an, dem das Büro gehört.
131 *   Zeigt den Namen, die E-Mail Adress und die Telefonnummer aus, die aus der
132 professor.csv gelesen wurden
133 *   Außerdem wird hier der Anwesenheitsstatus und ein kleines Bild angegeben
134 angegeben.
135 */
136 $cursorY = $displayHeight-102;
137 imagesetthickness($im, 4);

```

```

125     imageline ($im , 0 , $cursorY , $displayWidth , $cursorY , $black);
126
127     //Infos zur Person ausgeben
128     $schriftGroesse = 18;    //Schriftgröße des unteren Kastens skaliert nicht mit
129     //der oberen Schriftgröße
130     $cursorY += 25;
131     imagegetfttext($im, $schriftGroesse, 0, 10, $cursorY, $black,
132     $DEFAULT_FONT['bold'], $konstanten[2]);//Name
133     $cursorY += 30;
134     $schriftGroesse = 15;
135     imagegetfttext($im, $schriftGroesse*1.4, 0, 10, $cursorY, $black,
136     $DEFAULT_FONT['bold'], "@");
137     imagegetfttext($im, $schriftGroesse, 0, 50, $cursorY, $black,
138     $DEFAULT_FONT['bold'], $konstanten[3]);//E-Mail Adresse
139     $cursorY += 30;
140     imagegetfttext($im, $schriftGroesse, 0, 10, $cursorY, $black,
141     $DEFAULT_FONT['emoji'], "#\#9742;");
142     imagegetfttext($im, $schriftGroesse, 0, 50, $cursorY, $black,
143     $DEFAULT_FONT['bold'], $konstanten[4]);//Telefonnummer
144
145     //Anwesenheitsstatus ausgeben
146     $schriftGroesse = $schriftGroesse*1.1;
147     $cursorY = $displayHeight-60;
148     imagegetfttext($im, $schriftGroesse, 0, 400, $cursorY, $black,
149     $DEFAULT_FONT['bold'], "Ich bin ...");
150     $cursorY += 30;
151     imagegetfttext($im, $schriftGroesse, 0, 400, $cursorY, $black,
152     $DEFAULT_FONT['bold'], "... ".$status[0]);
153
154     //Bild anzeigen (Das Bild wird auf eine Größe Höhe von 100 pixeln
155     //Runterskaliert. Das Seitenverhältniss kannch dabei ändern)
156     $Datei = pathinfo($bild);
157     $Sendung = $Datei['extension'];
158     if($Sendung == "png" OR $Sendung == "jpg" OR $ndung == "jpeg") { //Überprüfen ob
159     der Pfad wirklich zu einem Bild führt
160         if($Datei['extension'] == "png"){
161             $imageSource = imagecreatefrompng($bild);
162         }
163         if($Datei['extension'] == "jpg" OR $Datei['extension'] == "jpeg" ){
164             $imageSource = imagecreatefromjpeg($bild);
165         }
166         list($originalwidth, $originalheight) = getimagesize($bild); //Höhe und
167         Weite des Bildes einlesen
168         $heigth = 100;
169         $width = 75; ($heigth/$originalheight)*$width;
170         imagecopyresampled($im, $imageSource, $displayWidth-$width,
171         $displayHeight-$heigth, 0, 0, $width, $heigth, $originalwidth,
172         $originalheight); //Bildgröße zu den geforderten neuen Werten verändern
173     }
174
175     ?>
```

A11 – Die „webinterface.html“-Datei

```

1  <!DOCTYPE html>
2  <html lang="de">
3  <head>
4      <meta charset="utf-8"/>
5      <title>Türschild - Config</title>
6  </head>
7  <body onload="reloadAndLoadData()">
8      <div>
9          <h1 style="display:inline-block; margin-right: 50%">Einrichtung des
10         Türschilds</h1>
11         <select id="selectProf" style="display:inline-block"
12             onchange="reloadTheWholePage()">
13             <option value="korte">Prof. Dr. Thomas Korte</option>
14             <option value="hausdoerfer">Prof. Dr. Rolf Hausdörfer</option>
15         </select>
16     </div>
17     <hr>
18
19     <h3>1) Anpassung des Zustands für das Feld "Ich bin..."</h3>
20
21     <form action="">
22         <input id="rbAnwesend" type="radio" name="stateOfProf" value="anwesend"
23             checked="checked">anwesend<br>
24         <input id="rbAbwesend" type="radio" name="stateOfProf"
25             value="abwesend">abwesend<br>
26         <input id="rbKrank" type="radio" name="stateOfProf" value="krank">krank<br>
27         <input id="rbUrlaub" type="radio" name="stateOfProf" value="im Urlaub">im
28             Urlaub<br>
29         <input id="rbPause" type="radio" name="stateOfProf" value="in der Pause">in
30             der Pause
31     </form>
32
33     <hr style="border:dashed #6E6E6E 2px">
34
35     <h3>2) Freitext erstellen</h3>
36     <textarea id="newMessage" rows="5" cols="200" style="resize:none">Geben Sie hier
37         die neue Meldung ein</textarea>
38     <p style="font-size: 11px; font-style: italic">Hinweis: Dieses Feld leer lassen
39         um keine Änderungen vorzunehmen!</p>
40
41     <h4>Aktuell werden folgende Meldungen angezeigt:</h4>
42
43     <p id="oldMessageTop">Ich bin eine alte Meldung</p>
44     <p id="oldMessageBottom">Ich auch</p>
45
46     <button type="submit" value="Submit"
47         onclick="confirmAndStartSending()">Bestätigen</button>
48     <button type="cancel" value="Cancel"
49         onclick="reloadTheWholePage()">Abbrechen</button>
50
51     <p style="font-size: 10px">
52         Project by Niclas Muss & Jan-Philipp Töberg<br>
53         <a style="font-size: 10px"
54             href="https://github.com/Janfiderheld/Drahtloses-Tuerschild-PA">Link zum
55             Projekt auf Github</a>
56     </p>
57 </body>
58
59     <!-- Hier beginnt der Javascript-Teil -->
60     <script type="text/javascript">
61         // lädt Zustand des Profs und beide alten Meldungen aus der CSV-Datei beim
62         // Aktualisieren der Seite
63         function reloadAndLoadData() {
64             var httpReq = new XMLHttpRequest();
65             var url = "getDataFromCSV.php";
66             httpReq.open("POST", url, true);
67
68             httpReq.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
69
70             httpReq.onreadystatechange = function() {
71                 if(httpReq.readyState == XMLHttpRequest.DONE && httpReq.status == 200)
72                 {
73                     var oldMessageTop = document.getElementById("oldMessageTop");
74                     var oldMessageBottom = document.getElementById("oldMessageBottom");
75
76                     oldMessageTop.innerHTML = httpReq.responseText;
77                     oldMessageBottom.innerHTML = httpReq.responseText;
78
79                     var selectProf = document.getElementById("selectProf");
80                     var options = selectProf.options;
81
82                     for (var i = 0; i < options.length; i++) {
83                         if (options[i].value === httpReq.responseText) {
84                             options[i].selected = true;
85                         }
86                     }
87
88                     var form = document.querySelector("form");
89                     form.submit();
90
91                 }
92             }
93         }
94     </script>

```

```

59         // Empfang der Antwort des PHP-Skripts als JSON-Element
60         var dataFromJSON = JSON.parse(httpReq.responseText);
61
62         // Setzt die beiden alten Meldungen mit den Elementen der CSV-Datei
63         document.getElementById("oldMessageTop").innerHTML =
64             dataFromJSON.newsNew;
65         document.getElementById("oldMessageBottom").innerHTML =
66             dataFromJSON.newsOld;
67
68         // Anwählen des passenden RadioButtons (passend zum Zustand aus der
69         // CSV-Datei)
70         var radioButtonTemp;
71         switch(dataFromJSON.status) {
72             case "anwesend":
73                 radioButtonTemp = document.getElementById("rbAnwesend");
74                 break;
75             case "abwesend":
76                 radioButtonTemp = document.getElementById("rbAbwesend");
77                 break;
78             case "krank":
79                 radioButtonTemp = document.getElementById("rbKrank");
80                 break;
81             case "im Urlaub":
82                 radioButtonTemp = document.getElementById("rbUrlaub");
83                 break;
84             case "in der Pause":
85                 radioButtonTemp = document.getElementById("rbPause");
86                 break;
87         }
88         radioButtonTemp.checked = true;
89     }
90
91     var dropDownProfs = document.getElementById("selectProf");
92     var currentProf = dropDownProfs.options[dropDownProfs.selectedIndex].value;
93     httpReq.send("prof=" + currentProf);
94
95     document.getElementById("newMessage").value = "Geben Sie hier die neue
96     Meldung ein";
97
98     // liest die Informationen aus den passenden HTML-Elementen und sendet diese an
99     // das PHP-Skript bei Betätigen des "Bestätigen"-Buttons
100    function confirmAndStartSending() {
101        var dropDownProfs = document.getElementById("selectProf");
102        var currentProf = dropDownProfs.options[dropDownProfs.selectedIndex].value;
103        var lastMessageNew = document.getElementById("oldMessageTop").innerHTML;
104        var lastMessageOld = document.getElementById("oldMessageBottom").innerHTML;
105        var contentOfTextArea = document.getElementById("newMessage").value;
106        var chosenStateOfProf = "undefined";
107
108        // Überprüft welcher der 5 RadioButtons gedrückt ist und speichert den
109        // zugehörigen Zustand
110        var buttons = document.forms[0];
111        for(var i = 0; i < buttons.length; i++) {
112            if(buttons[i].checked) {
113                chosenStateOfProf = buttons[i].value;
114            }
115        }
116
117        // senden der Daten über eine HTTP-Anfrage in einer eigenen Funktion
118        sendToPHPAndSave(currentProf, chosenStateOfProf, contentOfTextArea,
119        lastMessageNew, lastMessageOld);
120
121        // formuliert & sendet eine XMLHttpRequest an das PHP-Skript um diesem die zu
122        // speichernden Werte zu übergeben
123        function sendToPHPAndSave(currentProf, stateOfProf, newsToWrite, lastNewNews,
124        lastOldNews) {
125            var httpReq = new XMLHttpRequest();
126            var url = "saveDataToCSV.php";
127            var dataToSend = "prof=" + currentProf + "&zustand=" + stateOfProf +
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
623
624
625
625
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1
```

```
122 "&meldungWrite=" + newsToWrite + "&meldung1=" + lastNewNews + "&meldung2=" +
123 lastOldNews;
124 httpReq.open("POST", url, true);
125
126 httpReq.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
127
128 httpReq.onreadystatechange = function() {
129     if(httpReq.readyState == XMLHttpRequest.DONE && httpReq.status == 200) {
130         // Bei Antwort des PHP-Skripts:
131         // die Antwort auf der Konsole ausgeben und die Seite neuladen
132         console.log(this.responseText);
133         location.reload();
134     }
135     httpReq.send(dataToSend);
136 }
137
138 // aktualisiert die Seite, wenn der "Abbrechen"-Button gedrückt oder ein anderer
139 // Prof ausgewählt wird
140 function reloadTheWholePage() {
141     location.reload();
142 }
143 </script>
144 </html>
```

A12 – Die “getDataFromCSV.php”-Datei

```

1  <?php
2  	/**
3  	* Projektarbeit Sommersemester 2018 - Drahtloses Türschild mit E-Paper-Display
4  	* Author: Jan-Philipp Töberg
5  	* Version: 1.3
6  	* Letzte Änderung: 28.06.2018
7  	*
8  	* Dieses PHP-Skript regelt den Lesezugriff auf die unterschiedlichen
9  	CSV-Dateien.
10 	* Dabei steuert es, welche Werte geladen werden und überträgt diese an den
11 	Client per XMLHttpRequest.
12 	*/
13
14 	// Einstellen der internen Zeichenkodierung auf UTF-8 (wie bei HTML)
15 	mb_internal_encoding("UTF-8");
16 	// Auslesen des aktuellen Profs aus der Client-Übertragung
17 	$currentProf = htmlentities($_POST['prof']);
18
19 	// wählt die Datei abhängig vom ausgewählten Professor aus
20 	switch($currentProf) {
21  	case 'korte':
22  	// $fileName = 'Korte_status.csv';           // Speicherort im gleichen Ort
23  	$fileName = '../Korte_status.csv';          // Speicherort einen Ordner
24  	darüber
25  	break;
26 	case 'hausdoerfer':
27  	// $fileName = 'Hausdoerfer_status.csv';      // Speicherort im
28  	gleichen Ort
29  	$fileName = '../Hausdoerfer_status.csv';      // Speicherort einen
30  	Ordner darüber
31  	break;
32 	default:
33  	// $fileName = 'status.csv';                 // Speicherort im gleichen Ort
34  	$fileName = '../status.csv';                 // Speicherort einen Ordner darüber
35  	break;
36 }
37
38 // öffnet die CSV-Datei, um Daten in das fileContentOld-Array zu lesen
39 $file = fopen($fileName, 'r');
40
41 /* fileContentOld ist ein Array:
42  * Position 0 - Status des Professors
43  * Position 1 - erste Meldung, die angezeigt wird
44  * Position 2 - zweite Meldung, die angezeigt wird
45  */
46 $fileContentOld = fgetcsv($file, 1000, ",");
47 fclose($file);
48
49 // Übergabe des Array-Inhalts, als JSON formatiert, an das Javascript-Skript
50 echo json_encode( array( 'status' => $fileContentOld[0], 'newsNew' =>
51 $fileContentOld[1], 'newsOld' => $fileContentOld[2] ) );
52
53 // Beendet das PHP-Skript
54 die;
55
56 ?>

```

A13 – Die “`saveDataToCSV.php`“-Datei

```

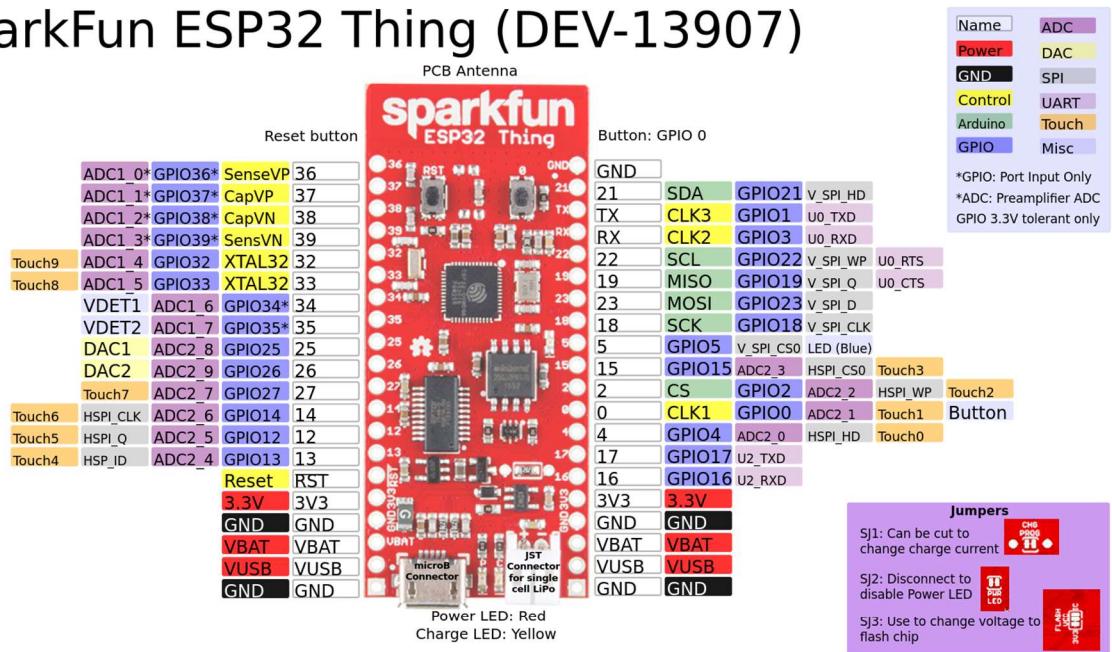
1  <?php
2  	/**
3    * Projektarbeit Sommersemester 2018 - Drahtloses Türschild mit E-Paper-Display
4    * Author: Jan-Philipp Töberg
5    * Version: 1.5
6    * Letzte Änderung: 28.06.2018
7    *
8    * Dieses PHP-Skript regelt den Schreibzugriff auf die unterschiedlichen
9    * CSV-Dateien.
10   * Dabei bekommt es Daten per XMLHttpRequest vom Client gesendet, welche es
11   * überprüft und anschließend
12   * in der passenden Datei speichert.
13 */
14
15 // Einstellen der internen Zeichenkodierung auf UTF-8 (wie bei HTML)
16 mb_internal_encoding("UTF-8");
17
18 // wählt die Datei abhängig vom ausgewählten Professor aus
19 switch($currentProf) {
20 	case 'korte':
21  	// $fileName = 'Korte_status.csv';           // Speicherort im gleichen Ort
22  	$fileName = '../Korte_status.csv';          // Speicherort einen Ordner
23  	darüber
24  	break;
25 	case 'hausdoerfer':
26  	// $fileName = 'Hausdoerfer_status.csv';      // Speicherort im
27  	// gleichen Ort
28  	$fileName = '../Hausdoerfer_status.csv';      // Speicherort einen
29  	// Ordner darüber
30  	break;
31 	default:
32  	// $fileName = 'status.csv';                  // Speicherort im gleichen Ort
33  	$fileName = '../status.csv';                  // Speicherort einen Ordner darüber
34  	break;
35 }
36
37 // Standard-Text in dem Eingabebereich
38 $defaultText = 'Geben Sie hier die neue Meldung ein';
39
40 // die übertragenen Daten von JavaScript
41 $state = htmlentities($_POST['zustand']);
42 $newsWithoutDate = htmlentities($_POST['meldungWrite']);
43 $newsToWrite = date('d.m.y - H:i')." ".$newsWithoutDate;
44 $newsNew = htmlentities($_POST['meldung1']);
45 // die erste alte Meldung, ohne die Datumsangabe vorweg
46 // Da die Datumsangabe genau 16 Zeichen lang ist, wird bei Zeichen 17 getrennt
47 $newsNewWithoutDate = substr($newsNew, 17);
48 $newsOld = htmlentities($_POST['meldung2']);
49 $currentProf = htmlentities($_POST['prof']);
50
51 echo "Übertragene Daten werden gespeichert...\n";
52
53 // Wenn der Zustand nicht übertragen wurde, hat die gesamte Übertragung nicht
54 // geklappt
55 if(!empty($state)) {
56 	// Wenn eine leere neue Meldung ODER die Standard-Meldung übertragen wurde,
57 	// sollen die Meldungen nicht geändert werden
58 	if(empty($newsWithoutDate) || (strcmp($newsWithoutDate, $defaultText) ===
59 	0)) {
60
61 	$dataToWrite = $state.", ".$newsNew.", ".$newsOld;
62 	echo "Keine neue Meldung eingetragen\n";
63 	// Wenn die neue Meldung gleich der ersten alten Meldung ist, dann sollen
64 	// diese ebenfalls nicht geändert werden
65 	} else if(strcmp($newsWithoutDate, $newsNewWithoutDate) === 0) {
66
67 	$dataToWrite = $state.", ".$newsNew.", ".$newsOld;
68 	echo "Die gleiche Meldung wie beim letzten Mal\n";
69 	// Die neue Meldung ist ungleich der alten Meldung, also wird die neue
70 	// Meldung hinzugefügt
71 	} else if(strcmp($newsWithoutDate, $newsNewWithoutDate) !== 0) {
72
73 	$dataToWrite = $state.", ".$newsToWrite.", ".$newsNew;
74 	echo "Neue Meldung!\n";
75 	}
76 }

```

```
63      // Die CSV-Datei wird geöffnet und beschrieben
64      $file = fopen($fileName, 'w');
65      fwrite($file, $dataToWrite);
66      fclose($file);
67      echo "Übertragene Daten wurden gespeichert!\n";
68  } else {
69      echo "Übertragene Daten wurden nicht gespeichert!\nIrgendetwas ist schief
70      gegangen...";
71  }
72
73 // Beendet das PHP-Skript
74 die;
75 ?>
```

A13 – Die Anschlüsse des SparkFun ESP32 Thing

SparkFun ESP32 Thing (DEV-13907)



Power

ESP32 VCC range: 2.2V-3.6V
VBAT: direct to battery (and charger)
VUSB: direct to USB (5V)
VCC: Output of regulator 3.3V/600mA
Up to 250mA during RF transmissions

Wireless

Wifi: 802.11 b/g/n/e/i
WPA/WPA2/WPA2-Enterprise/SPS
Bluetooth: Bluetooth 4.2/BLE

ESP32

Dual-core Xtensa 32-bit LX6
Up to 240MHz
520kB internal SRAM
4MB external flash

- **Multiplexed I/Os allow up to**
- 18 ADC channels
- 3 SPI interfaces
- 3 UART interfaces
- 2 I²C interfaces
- 16 LED PWM outputs
- 2 DACs
- 10 Capacitive Touch Inputs

ADC Preamp

GPIO pins 36,67,38, and 39 are able to be used as a low noise analog pre-amplifier

Other*
Hall Sensor
Temp sensor (-40C to 125C)
SD/SDIO/MMC Host Controller
CAN Bus

*On datasheet, but may not be supported yet

