**HTL Saalfelden**

**Systemplanung und Projektentwicklung**

****

**Projektdokumentation**

**2022 / 2023**

|  |  |
| --- | --- |
| **Projektbezeichnung** | TableCast |
| **Projektteam** | Cedric Broukx, Alessandro Davare, Jan Grassegger, Sarah Hagenhofer |
| **Erstellt am** | 11.01.2023 |
| **Letzte Änderung am** | 24.05.2023 |
| **Status** | In Bearbeitung [/fertiggestellt/pausiert/abgebrochen/Prüfung] |
| **Aktuelle Version** | 1.0 |

**Änderungsverlauf**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr.** | **Datum** | **Version** | **Geänderte Kapitel** | **Art der Änderung** | **Autor** |
| 1 | 11.01.2023 | 1.0 | Deckblatt, Projektübersicht | Grundlegende Dinge anpassen und erste Informationen eintragen | Sarah Hagenhofer |
| 2 | 18.01.2023 20.01.2023 | 1.0 | Funktionale Anforderungen, Nichtfunktionale Anforderungen, Projektplanung | Ersteintragung der Anforderungen, Eintragung der Variantenbildung | Sarah Hagenhofer |
| 3 | 25.01.2023 | 1.0 | Projektdurchführung – Sprintplanung | Eintragen der Informationen zu Sprint 1 | Sarah Hagenhofer |
| 4 | 08.02.2023 | 1.0 | Projektdurchführung – Sprint Demo, Sprint Retrospektive, Sprint Zusammenfassung | Ergänzung der Daten zum Sprintende | Sarah Hagenhofer |
| 5 | 22.02.2023 | 1.0 | Projektdurchführung | Eintragen der Informationen zu Sprint 2 | Sarah Hagenhofer |
| 6 | 01.03.2023  02.03.2023 | 1.0 | Projektdurchführung | Dokumentation Sprint 2 | Sarah Hagenhofer |
| 7 | 08.03.2023 | 1.0 | Projektdurchführung | Abschluss Sprint 2 | Sarah Hagenhofer |
| 8 | 15.03.2023  22.03.2023 | 1.0 | Projektdurchführung | Planung Sprint 3 | Sarah Hagenhofer |
| 9 | 12.04.2023 | 1.0 | Projektdurchführung | Abschluss Sprint 3 | Sarah Hagenhofer |
| 10 | 19.04.2023 | 1.0 | Projektdurchführung | Eintragung der Informationen zu Sprint 4 | Sarah Hagenhofer |
| 11 | 11.05.2023 | 1.0 | Projektplanung | Eintragung UML Diagramme | Sarah Hagenhofer |
| 12 | 24.05.2023 | 1.0 | Softwarearchitektur | Dokumentation Website | Alessandro Davare |
| 13 | 31.05.2023 | 1.0 | Softwarearchitektur & Installation | Cronjobs & Installation | Alessandro Davare |
| 14 | 31.05.2023 | 1.0 | Projektdurchführung | Abschluss 4, Planung 5 | Sarah Hagenhofer |
| 15 | 13.06.2023 | 9.9 | Design | Figma / Website | Jan Grassegger |

**Inhalt**

[1. Allgemeines / Projektübersicht 4](#_Toc136420094)

[1.1 Projektbeschreibung 4](#_Toc136420095)

[1.2 Projektteam und Schnittstellen 4](#_Toc136420096)

[2. Funktionale Anforderungen 5](#_Toc136420097)

[2.1 Grundlegende Funktionalität 5](#_Toc136420098)

[2.1.1 Teller drehen, Text auf dem Display, LED Helligkeit 5](#_Toc136420099)

[2.1.2 SSH Server 5](#_Toc136420100)

[2.2 Sämtliche Steuerung durch Website, Datenspeicherung 5](#_Toc136420101)

[3. Nichtfunktionale Anforderungen 5](#_Toc136420102)

[4. Projektplanung 6](#_Toc136420103)

[4.1 Variantenbildung 6](#_Toc136420104)

[4.2 Machbarkeitsstudie 6](#_Toc136420105)

[4.3 Allgemeine Planungsinformationen 6](#_Toc136420106)

[4.4 Projektumfeldanalyse 6](#_Toc136420107)

[5. Softwarearchitektur 6](#_Toc136420108)

[5.0.1. Main.py 7](#_Toc136420109)

[5.0.2. Drehteller.py 10](#_Toc136420110)

[5.0.3. Bashscripts 14](#_Toc136420111)

[5.0.3.1 ForwardPort 14](#_Toc136420112)

[5.0.3.2 Start 14](#_Toc136420113)

[5.0.3.3 RunScript 14](#_Toc136420114)

[5.0.4. Cronjobs 15](#_Toc136420115)

[5.1 Aktivitätsdiagramme 15](#_Toc136420116)

[5.1.1 Aktivitätsdiagramm 1 Name 15](#_Toc136420117)

[5.1.2 Aktivitätsdiagramm n Name 15](#_Toc136420118)

[5.2 Sequenzdiagramme 16](#_Toc136420119)

[5.2.1 Sequenzdiagramm Erstellung neuer User 16](#_Toc136420120)

[5.2.2 Sequenzdiagramm Bedienung Drehteller 17](#_Toc136420121)

[5.3 Komponentendiagramme 17](#_Toc136420122)

[5.4 Verteilungsdiagramme 18](#_Toc136420123)

[5.5 Anwendungsfalldiagramm 18](#_Toc136420124)

[5.6 Softwarekomponenten / Programme 19](#_Toc136420125)

[5.6.1 SW Programme 19](#_Toc136420126)

[5.6.2 SW Komponenten 19](#_Toc136420127)

[6. Projektdurchführung 20](#_Toc136420128)

[6.1 Sprint 1 20](#_Toc136420129)

[6.1.1 Sprintplanung 20](#_Toc136420130)

[6.1.2 Sprint Demo 20](#_Toc136420131)

[6.1.3 Sprint Retrospektive 21](#_Toc136420132)

[6.1.4 Sprint Zusammenfassung 21](#_Toc136420133)

[6.2 Sprint 2 23](#_Toc136420134)

[6.2.1 Sprintplanung 23](#_Toc136420135)

[6.2.2 Sprint Demo 23](#_Toc136420136)

[6.2.3 Sprint Retrospektive 24](#_Toc136420137)

[6.2.4 Sprint Zusammenfassung 24](#_Toc136420138)

[6.3 Sprint 3 26](#_Toc136420139)

[6.3.1 Sprintplanung 26](#_Toc136420140)

[6.3.2 Sprint Demo 26](#_Toc136420141)

[6.3.3 Sprint Retrospektive 26](#_Toc136420142)

[6.3.4 Sprint Zusammenfassung 27](#_Toc136420143)

[6.4 Sprint 4 28](#_Toc136420144)

[6.4.1 Sprintplanung 28](#_Toc136420145)

[6.4.2 Sprint Demo 28](#_Toc136420146)

[6.4.3 Sprint Retrospektive 28](#_Toc136420147)

[6.4.4 Sprint Zusammenfassung 29](#_Toc136420148)

[6.5 Sprint 5 29](#_Toc136420149)

[6.5.1 Sprintplanung 29](#_Toc136420150)

[6.5.2 Sprint Demo 29](#_Toc136420151)

[6.5.3 Sprint Retrospektive 29](#_Toc136420152)

[6.5.4 Sprint Zusammenfassung 30](#_Toc136420153)

[7. Installation / Software deployment 32](#_Toc136420154)

[8. Projektabschluss 32](#_Toc136420155)

[8.1 Projektzusammenfassung 32](#_Toc136420156)

[8.2 Attachments 32](#_Toc136420157)

# Allgemeines / Projektübersicht

## Projektbeschreibung

Die HTL Saalfelden hat im Werkstätten-Unterricht der Mechatronikabteilung einen Präsentationsteller entworfen. Das Gerät wird von vier Seiten beleuchtet, ein Display zeigt an der Vorderseite einen Schriftzug und der Teller dreht sich durch einen Gleichstrommotor.

Bisher wurde eine Android App zur Steuerung verwendet, nun soll die bestehende Elektroniklösung durch einen Raspberry Pi Zero inklusive WLAN ersetzt werden, während die Steuerung nun durch eine Website erfolgen soll, sodass das Produkt mit beliebigen Geräten gesteuert werden kann. Die genauen Anforderungen an das Projekt werden im Kapitel Funktionale Anforderungen festgehalten.

## Projektteam und Schnittstellen

|  |  |  |
| --- | --- | --- |
| **Rolle(n)** | **Name** | **E-Mail** |
| Product Owner/Scrum Master | Sarah Hagenhofer | [Sarah.hagenhofer@htl-saalfelden.at](mailto:Sarah.hagenhofer@htl-saalfelden.at) |
| Developer | Alessandro Davare | [Alessandro.davare@htl-saalfelden.at](mailto:Alessandro.davare@htl-saalfelden.at) |
| Developer | Jan Grassegger | [Jan.grassegger@htl-saalfelden.at](mailto:Jan.grassegger@htl-saalfelden.at) |
| Developer | Cedric Broukx | [Cedric.broukx@htl-saalfelden.at](mailto:Cedric.broukx@htl-saalfelden.at) |

# Funktionale Anforderungen

## Grundlegende Funktionalität

### Teller drehen, Text auf dem Display, LED Helligkeit

Der Teller soll sich drehen können, auf dem Display soll ein Text sichtbar sein und die LED Helligkeit soll sich einstellen lassen. Die Werte hierfür (Drehgeschwindigkeit, Text und Helligkeit) werden vom Benutzer gegeben und können jederzeit geändert werden. Außerdem soll der mögliche Text bis zu 10 Zeilen lang sein können und zwischen den einzelnen Zeilen hin- und herwechseln.

### 2.1.2 SSH Server

Der Zugriff auf den Raspberry Pi soll auch über einen SSH Server möglich sein.

### Sämtliche Steuerung durch Website, Datenspeicherung

Sämtliche Steuerung soll für den Benutzer über eine Website möglich sein. Die Website soll passwortgeschützt sein, um ungewollten Zugriff zu verhindern.

* 1. **Netzwerkverbindung**

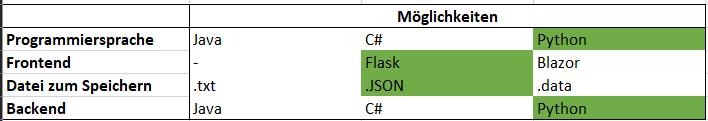
Der Teller soll einen eigenen Hotspot darstellen, auf den sich andere Geräte einloggen können. Auch soll ein Zugang in ein bestehendes WLAN möglich sein. Eine Änderung zwischen bestehendem und neuem WLAN ist jederzeit möglich.

# Nichtfunktionale Anforderungen

Der Teller soll darauf vorbereitet werden, in Zukunft mit mehreren anderen Tellern zusammenzuarbeiten – hierbei soll eine Verbindung und Steuerung untereinander möglich sein. Der gesamte Code ist übersichtlich zu kommentieren, die Website übersichtlich und passend zu gestalten.

# Projektplanung

## Variantenbildung



Die Variantenbildung wurde mit dem gesamten Team durchgeführt, gewählt wurde der grün markierte Pfad.

# Softwarearchitektur

Um das Projekt zum Laufen zu bringen, werden nur die unten genannten SW-Komponenten benötigt. Das Projekt besteht aus 2 großen Scripts: der „main.py“ welche die Flask-Anwendung beinhaltet und der „drehteller.py“, welche für die Steuerung der Drehtellers ist. Ebenfalls gibt es einige kleinere Dateien, die nur zur Kommunikation zwischen den Skripts dienen. Es gibt eine zweite Version von dem Drehteller Script: „drehteller2.py“. Diese ist lediglich zum Stoppen des Tellers, falls dieser gerade läuft. Diese funktioniert auf dieselbe Weise wie drehteller.py, mit dem Unterschied, dass sie eine JSON-Datei lädt, bei der die Daten auf null gesetzt sind und wird deshalb nicht ausführlich dokumentiert.

Außerdem noch wichtig ist das „forwardPort.sh“ Skript, welches dafür zuständig ist, dass der Port weitergeleitet wird. Dies bedeutet, dass man von einem anderen Gerät aus auf den localhost mit dem jeweiligen Port vom Raspberry Pi zugreifen kann.

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung

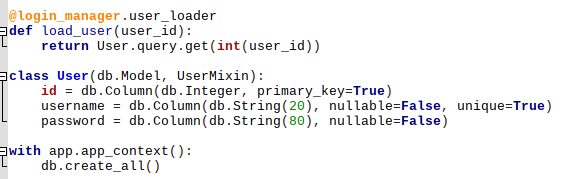
## 5.0.1. Main.py

Am Anfang wird die „app“ Variable initialisiert. In app wird der Flask Server gestartet und wird in dem gesamten Script für beispielsweise Paths benutzt. Ebenfalls werden Variablen für beispielsweise die SQLite Verbindung gesetzt oder auch für den Login Manager

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Für den Login Manager ist es notwendig, den User zu laden. Dafür gibt es die Funktion „load\_user“ mit der „@login\_manager“ Annotation, welche die User\_id zurückgibt. Der User besteht aus einer ID, die von der Datenbank automatisch vergeben wird, einem Username und einem Passwort. Ebenfalls wird falls noch nicht vorhanden, die Datenbank erstellt.



Für das Login und für die Registrierung der Nutzer wird jeweils eine neue Klasse erzeugt. Für das Login werden nur zwei Felder für die Eingabe von Benutzername und Passwort erstellt. Außerdem noch ein Button, um die eingegebenen Daten zu überprüfen.

Für Registerform zählt das gleiche, wobei es bei dieser Klasse noch eine Funktion gibt, um zu prüfen, ob es den Benutzernamen schon gibt. Falls beim Registrierungsvorgang ein Benutzername eingegeben, der schon existiert, bekommt man einen Validation Error mit der Message „Username already Exists“.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Bei der Settings Route, wird am Anfang abgefragt, ob ein POST Request gemacht wurde. Dies ist für die Unterscheidung, welche Website angezeigt werden soll. Am Anfang wird das Forms angezeigt, in dem man die Einstellung tätigen kann und erst dann, wenn man auf den Button drückt, wird ein POST geschickt. Nach dem Buttonclick werden die Eingabe in die Variablen gespeichert. Für den zu speichernden Text muss noch etwas getan werden, da dieser als ein langer String gespeichert ist. Der String muss noch an den Zeilenumbrüchen gesplittet werden, sodass ein Array an Strings entsteht. Nachdem man alle Daten hat, wird ein json-string erzeugt, um diesen in einem JSON-File zu speichern. Mit fo.write(…) wird direkt in die Datei geschrieben. Zum Abschluss bekommt der User eine Auflistung der gespeicherten Daten im HTML-Format.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Bei der Start Route wird am Anfang geprüft, ob der Teller schon dreht, oder nicht. Falls der Teller schon dreht, wird eine Nachricht zurückgegeben, dass der Teller bereits läuft. Falls er nicht läuft, wird die variable auf 1 gesetzt, und das drehteller.py Script ausgeführt, dass die bereits gespeicherten Daten benutzt.

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Die beiden Routen / und /home sind für dasselbe, sie geben die index.html zurück. Wenn die Route / aufgerufen wird, wird direkt auf die /home Route umgeleitet.

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung

Für die Login Route wird am Anfang geprüft, ob der User bereits eingeloggt ist. Falls dies der Fall ist, gibt diese Seite „Already logged in“ zurück. Wenn nicht dann wird das Login Form erstellt und angezeigt. Wenn der User in diesem die Daten eingibt und bestätigt, wird geprüft, ob diese stimmen, falls nicht wird die Seite neugeladen und falls sie stimmen, wird der User auf die Home Seite umgeleitet.

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Beim Logout ist es ziemlich Simpel, der User wird auf Knopfdruck abgemeldet und direkt auf die Home Seite weitergeleitet.

Ein Bild, das Text, Schrift, Screenshot, Reihe enthält.

Automatisch generierte Beschreibung

Register gibt bei Aufruf das Forms zurück, um sich zu registrieren. Bei Bestätigung der Daten werden diese direkt in die Datenbank geschrieben.

Die Main Funktion in dieser Datei führt die App aus.

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

## 5.0.2. Drehteller.py

Der erste Teil dieser Datei besteht aus den vorgegebenen Skripts, welche benötigt sind, um die einzelnen Teile des Tellers anzusprechen.

Ein Bild, das Text, Screenshot, Zahl, Schrift enthält.

Automatisch generierte Beschreibung

In der Main wird am Anfang die JSON-Datei geladen und der Inhalt in die Variable „data“ gespeichert. Als nächstes wird bei Zeile 106 der Key „Helligkeit“ in Data gesucht und die mit dem Value davon die LED angesteuert. Dasselbe passiert auf Zeile 113 mit dem Key Geschwindigkeit und dem Motor des Drehtellers.

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung

Als nächstes wird der Text in eine Variable gespeichert und am Ende noch ein Beistrich mit einem Leerzeichen hinzugefügt. Der extra Beistrich ist nur dafür da, um eine Nahtlose Schleife zu erzeugen, da der Text am Bildschirm später Zeichen für Zeichen aktualisiert wird. Der lange Text wird an den Beistrichen in ein Array gesplittet und beispielsweise in der Konsole ausgegeben. Danach wird ein extra Array erzeugt, dieses ist dafür da die einzelnen Zeichen zu speichern

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Es wird Zeichen für Zeichen durch den Input durchgegangen und es wird an das gerade erstellte Array angehängt.

Ein Bild, das Text, Schrift, weiß enthält.

Automatisch generierte Beschreibung

Bevor die Endlosschleife für den Text beginnt, wird noch eine Zählervariable ‚x‘ gesetzt und eine weitere Variable „eight“, die im Endeffekt dafür ist, Text auf die zwei Bildschirme aufzuteilen.

Ein Bild, das Text, Schrift, Screenshot enthält.

Automatisch generierte Beschreibung

Die Schleife läuft, solange der Text nicht null ist. Als erstes werden die ersten 8 Zeichen in der „eight“ Variable angehängt. Wenn der Zähler dann 8 erreicht, werden die 8 Zeichen auf den Linken Bildschirm gesetzt und „eight“ wieder zurückgesetzt. Der Zähler zählt bei 8 dann weiter. Wenn die nächsten 8 Zeichen hinzugefügt worden sind und der Zähler 16 erreicht, werden die 8 Zeichen auf den rechten Bildschirm gesetzt. Außerdem wird der erste Buchstabe des Arrays am Ende hinzugefügt, und am Anfang weggegeben, sodass ein Effekt entsteht, als würde der Text sich nach Links bewegen. Am Ende werden dann die Variablen wieder zurückgesetzt und das Zählen der Zeichen fängt wieder von vorne an, mit einem Zeichen nach Links gesetzt.

Die Exception „KeyboardInterrupt“ ist, falls jemand das Skript manuell durch einen Keyboard Interrupt beendet. Hier wird dann eine Kurze Nachricht ausgegeben und die GPIO Pins des Raspberry Pi’s gecleared.

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

## 5.0.3. Bashscripts

## 5.0.3.1 ForwardPort

Bei diesem Script gibt es nicht viel zu sagen. Durch Socat sagen wir dem Raspberry Pi, dass alle Anfragen, die auf den Port 5001 außerhalb des Lokalen Netzwerkes geschickt werden, an den Localhost mit dem Port 5000 weiterschickt werden. Auf diesem Port läuft unser Projekt am Raspberry Pi.



## 5.0.3.2 Start

Dieses Skript ist für den Cronjob und für das Manuelle Starten da. Hiermit sagen wir, dass unsere App mit Gunicorn starten. Mann muss die Pfade angeben, da das System nicht weiß, woher es die Dateien nehmen sollen. Ebenfalls wird in diesem Command gesagt, dass es in das Flasksite Verzeichnis wechseln soll, in der es gunicorn mit den Argumenten ausführen soll. Mit dem Argument -b und der Adresse sagt man, wo der Server gestartet werden soll. Am Ende sagt man welche Datei gestartet werden soll und was in dieser Funktion. In unserem Fall die main.py und es soll app gestartet werden.



## 5.0.3.3 RunScript

Wieder ein einfacheres Script, ist nur da um von der Main.py aus das drehteller.py Skript auszuführen.



## 5.0.4. Cronjobs

Folgende Cronjobs sind notwendig, um das Projekt beim Start des Raspberry Pis automatisch zu starten.

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung

## 

## 5.0.5. Website Design

Für das Design wurde ursprünglich ein Entwurf mit Figma erstellt und daran wurde sich erstmals auch noch gehalten.

Ein Bild, das Text, Screenshot, Rechteck, Design enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Multimedia, Software enthält.

Automatisch generierte Beschreibung

Ein Bild, das Elektronik, Text, Screenshot, Gerät enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift, Design enthält.

Automatisch generierte Beschreibung

Dieses Design wurde jedoch schnell durch ein neueres „moderneres“ bzw. passenderes ersetzt und die Seiten Dashboard und Wlan wurden ebenfalls verworfen.

Die Loginseite sieht mittlerweile folgendermaßen aus:  
Ein Bild, das Text, Screenshot, Software, Betriebssystem enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Software, Zahl enthält.

Automatisch generierte Beschreibung

Die Navbar ist außerdem Responsive, dies wurde mithilfe von Bootstrap und dem Framework Lit ermöglicht.  
Eigentlich benutzt man dafür Bootstrap und JQuery oder noch andere Möglichkeiten mit Javascript.  
Ich habe mich jedoch dafür entschieden das Framework Lit zu benutzen und die Navbar in einen Lit Component zu geben welche dann auf jeder Html Seite gerendert wird.

Ein Bild, das Text, Screenshot, Software, Multimedia enthält.

Automatisch generierte Beschreibung

Für den Lit Component erstellt man ein einfaches Javascript file in welchem man Lit importiert und eine Klasse erstellt welche ein LitElement erweitert.  
Zuvor muss das Lit Framework noch mit „npm i lit“ im Terminal des Projektes installiert werden.  
Danach kann man wie bei dem Beispiel unserer Navbar css Code hinzufügen als auch eigene Funktionen erstellen welche im HTML Abschnitt darunter benutzt werden.  
Bei unserer Navbar haben wir eine Funktion für das Responsive Benutzen des Navbar Buttons.  
Alternativ könnte man Jquery benutzen aber da es mit Lit leicht möglich ist diese Collapse/Toggle Funktion für die Navbar zu erstellen wurde diese Methode benutzt.  
Nach der ToggleNavbar Funktion kommt der Render Html Teil mit dem gesamten Navbar Code in welchem ein zusätzliches CSS Stylesheet eingebunden wurde um die Navbar zu stylen.  
Der Endpart des Skripts ist noch besonders wichtig da dieser das Skript erst benutzbar macht.  
Man definiert einen Namen mit welchem der Component aufgerufen bzw eingebunden werden kann.

Ein Bild, das Text, Screenshot, Software, Zahl enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Zahl, Schrift enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Zahl, Schrift enthält.

Automatisch generierte Beschreibung

## Aktivitätsdiagramme

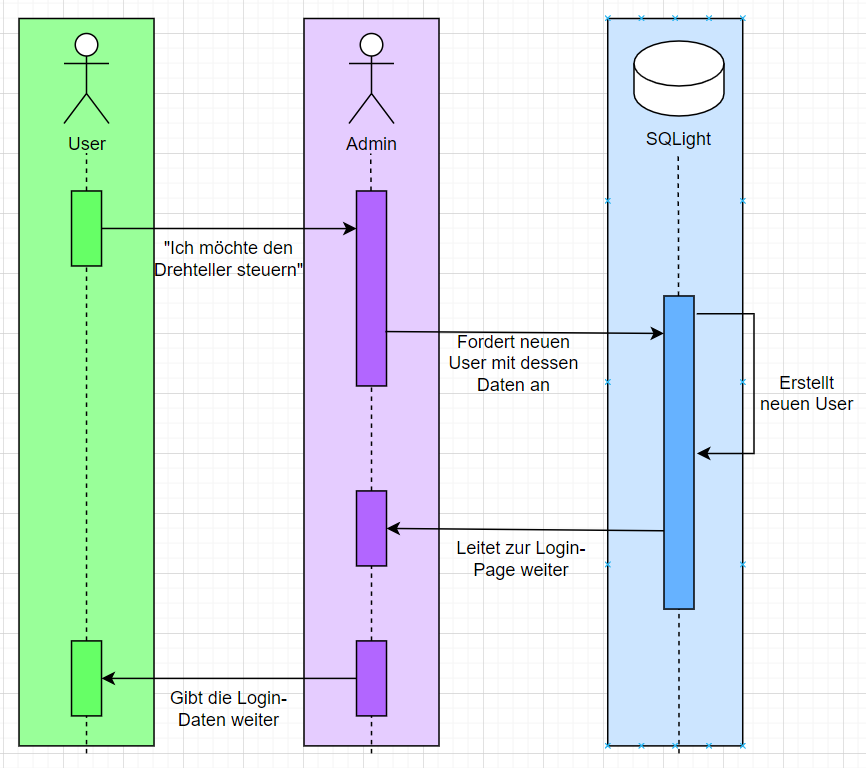
### Aktivitätsdiagramm Daten bearbeiten

Ein Bild, das Text, Diagramm, Screenshot, Plan enthält.

Automatisch generierte Beschreibung

## Sequenzdiagramme

### Sequenzdiagramm Erstellung neuer User



### Sequenzdiagramm Bedienung Drehteller

Ein Bild, das Text, Diagramm, Zahl, parallel enthält.

Automatisch generierte Beschreibung

## Komponentendiagramme

Ein Bild, das Text, Diagramm, Rechteck, Plan enthält.

Automatisch generierte Beschreibung

## Verteilungsdiagramme

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

## Anwendungsfalldiagramm

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

## Softwarekomponenten / Programme

### SW Programme

|  |  |  |  |
| --- | --- | --- | --- |
| Programm | Version | Download | Anmerkung |
| Visual Studio Code | 1.78.2 | https://code.visualstudio.com/ | IDE um Python Code zu schreiben |
| Raspberry Pi Imager | 1.7.4 | <https://www.raspberrypi.com/software/> | Benötigt um auf die SD Karte ein Betriebssystem zu installieren |
| HeidiSQL | 12.5 | <https://www.heidisql.com/> | Um die User SQLite DB zu erstellen bzw. zu editieren |
| Draw IO | 20.8.16 | <https://www.drawio.com/> | Für die Diagramme |
| VNC Viewer | 6.22.286 | https://www.realvnc.com/de/connect/download/viewer/ | Um den PI Bildschirm zu streamen |
| Github Desktop | 3.2.3 | https://desktop.github.com/ | Um Daten auf das Github Repo zu pushen |

### 5.6.2 SW Komponenten

|  |  |  |  |
| --- | --- | --- | --- |
| Komponente | Version | Download | Anmerkung |
| Python3 | 3.9.2 | Apt install python3 |  |
| flask | 2.2.3 | pip install flask | Webframework |
| flask\_sqlalchemy | - | Pip install flask\_sqlalchemy | Die folgenden Komponenten sind alle für die Authentifizierung des Users |
| flask\_login | - | Pip install flask\_login | . |
| flask\_wtf | - | Pip install flask\_wtf | . |
| wtforms | - | Pip install wtforms | . |
| flask\_bcrypt | - | Pip install flask\_bcrypt | . |
| Gunicorn | 20.1 | Pip install gunicorn | Server für Flaskanwendung |
| socat | - | Apt install socat | Für das Portforwarding |
| Lit Framework | - | Npm i Lit | Für die Navbar im Design |

# Projektdurchführung

## Sprint 1

### Sprintplanung

Dauer: 25.01.2023 – 08.02.2023

Ausgewählte User Stories: Teller drehen – Steuerung durch Benutzer, Displaytext – Steuerung durch Benutzer, LED-Helligkeit – Steuerung durch Benutzer, Nachträgliche Änderungen an der Software – Zugang für den Administrator, Hotspot

Anzahl Story points: 14

Ausgewählte Punkte aus der Impediment Liste: 5

### Sprint Demo

Erledigt wurden:

* Teller drehen – Steuerung durch Benutzer
* Displaytext – Steuerung durch Benutzer
* LED-Helligkeit – Steuerung durch Benutzer
* Nachträgliche Änderungen an der Software – Zugang für den Administrator

Anzahl erledigte Story Points: 6

Nicht erledigt wurden:

* Hotspot

Anzahl nicht erledigte Story Points: 8

Grund der Nichterledigung:  
Es wäre ein Gerät benötigt worden, das noch nicht zur Verfügung gestellt wurde. Ab kommendem Sprint ist dieses voraussichtlich benutzbar.

### Sprint Retrospektive

(Bewertung des Sprints auf reetro.app)

Gut lief laut Team während Sprint1 folgendes:

* Pi aufsetzen
* Erster Eindruck von Python
* Flask anschauen und erste Versionen schreiben

Zu verbessern ist laut Team:

* Python Skills
* User-Story Storypointschätzung
* Team Spirit
* Kommunikation
* Genaue Sprintplanung und Arbeitszuteilung

Nötige Items/Aktionen laut Team:

* Drehteller
* User-Story Points sollen neu geschätzt werden (vor allem bei großen)

### Sprint Zusammenfassung

Die Grundlagen für das Projekt wurden gelegt. Das Team hat sich über die wichtigen Funktionen und dessen Benutzung informiert und bereits mit den ersten Punkten begonnen.   
Der Punkt „Hotspot“ konnte nicht begonnen werden, da hierfür der Drehteller fehlt, der erst zur Verfügung gestellt werden muss.

Es wurde die neue User Story „Websitedesign“ erstellt, in der die Website gestaltet werden soll – diese soll gleich im nächsten Sprint abgearbeitet werden. Auch wurde die User Story „Zeitabhängiges Ein- und Ausschalten“ während dem Sprint auf Befehl vom Product Owner hinzugefügt und mit 40 Storypoints bewertet – die Story wird voraussichtlich erst im letzten Sprint bearbeitet. Es wurden keine User Stories entfernt. Die Storypoints des Punktes „Hotspot“ wurden angepasst, da sich das Team informiert hat und den Punkt zügig bearbeiten kann, sobald das Gerät zur Verfügung gestellt wird.   
**Storypointverteilung: 129 offen, 8 werden in nächsten übernommen, 6 geschafft**

Laut Team verlief der Sprint okay. Zu verbessern gilt die generelle Kommunikation und die Planung samt Einteilung. Wichtig ist es, möglichst bald das Drehteller zur Verfügung gestellt zu bekommen.

Zeitplanung:

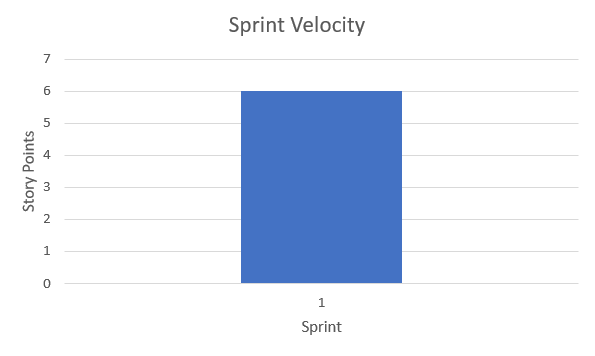
Aktuell keine Terminverschiebung nach hinten.

Burndown-Chart:

Ein Bild, das Text, Monitor, Screenshot, Bildschirm enthält.

Automatisch generierte Beschreibung

Velocity – aktuell bei 6:



## Sprint 2

### Sprintplanung

Dauer: 22.02.2023 – 08.03.2023

Ausgewählte User Stories: Displaytext soll länger als 20 Zeichen sein dürfen – Steuerung durch Benutzer, Websitesteuerung, Websitedesign, Eingestellte Werte speichern und einsehen können;  
  
Verbleibende User Stories aus vorherigem Sprint: Hotspot;

Anzahl Story points: 44

Ausgewählte Punkte aus der Impediment Liste: 5 🡪 4 neue und 1 aus vorherigem Sprint

### Sprint Demo

Erledigt wurden:

* Hotspot
* Eingestellte Werte speichern/einsehen
* Displaytext länger als 20 Zeichen
* Websitesteuerung

Anzahl erledigte Story Points: 36

Nicht erledigt wurden:

* Websitedesign

Anzahl nicht erledigte Story Points: 8

Grund der Nichterledigung: Nicht genug Zeit 🡪 wurde angefangen, aber nicht beendet.

### Sprint Retrospektive

(Bewertung des Sprints auf reetro.app)

Gut lief laut Team während Sprint2 folgendes:

* Designkritik
* Visual Studio öffnen
* Backend
* Flask Seite

Zu verbessern ist laut Team:

* Kommunikation
* Website

Nötige Items/Aktionen laut Team:

### Sprint Zusammenfassung

Der offengebliebene Punkt „Hotspot“ aus dem vergangenen Sprint wurde abgeschlossen.

Es wurde an einer Website gearbeitet, über die dann die gesamte Steuerung erfolgen soll. Die Website wurde fertig abgeschlossen, sämtliche Funktionen funktionieren bereits über diese. Die Werte werden richtig gespeichert und können eingesehen werden. Zudem wurde die Möglichkeit gegeben, den Displaytext auf bis zu 10 Zeilen Text verlängern.  
Angefangen wurde mit dem Websitedesign, dazu wurde bereits Feedback eingeholt – dieser Punkt wurde nicht abgeschlossen und wird in den nächsten Sprint gezogen.

Es wurden keine User Stories hinzugefügt oder entfernt. Alle Schätzungen sind gleichgeblieben.  
**Storypointverteilung: 101 offen, 8 werden in nächsten übernommen, 36 geschafft**

Laut Team verlief der Sprint gut, es wurden viele Punkte bearbeitet und abgeschlossen. Das Websitedesign bringt Schwierigkeiten, die im nächsten Sprint noch abgearbeitet werden sollen. Nötige Aktionen gibt es keine.

Ab nächstem Sprint wird die SCRUM-Planung in Trello vorgenommen – Velocity und Burndown-Chart werden selbst dokumentiert. Das Trello-Chart ist hier zu finden: <https://trello.com/b/lc6CWpEH/t1tablecast>

Zeitplanung:

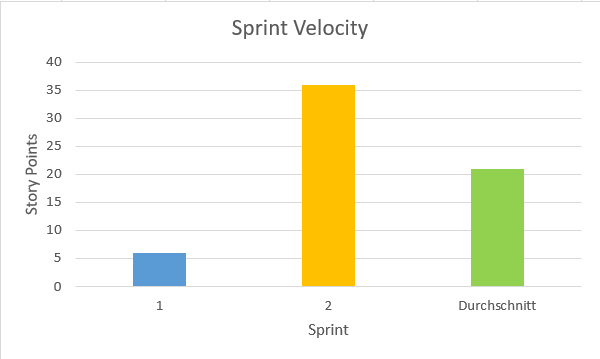
Aktuell keine Terminverschiebung.

Burndown-Chart:

Ein Bild, das Text, Monitor, drinnen, Screenshot enthält.

Automatisch generierte Beschreibung

Velocity – aktuell bei: 21



## Sprint 3

### Sprintplanung

Dauer: 15.03.2023 – 12.04.2023 (zwei Wochen länger wegen Ferienverzögerung)

Ausgewählte User Stories: Website passwortgeschützt, Zeitabhängiges Ein- und Ausschalten  
  
Verbleibende User Stories aus vorherigem Sprint: Websitedesign

Anzahl Story points: 51

Ausgewählte Punkte aus der Impediment Liste: 3 ausgewählt 🡪 1 davon ist aus vorherigem Sprint

### Sprint Demo

Erledigt wurden:

* Website passwortgeschützt

Anzahl erledigte Story Points: 3

Nicht erledigt wurden:

* Websitedesign
* Zeitabhängiges Ein- und Ausschalten

Anzahl nicht erledigte Story Points: 48

Grund der Nichterledigung: Zu kompliziert für die Zeit

### Sprint Retrospektive

(Bewertung des Sprints auf reetro.app)

Gut lief laut Team während Sprint1 folgendes:

* Lit hinzufügen
* Flask Tutorials

Zu verbessern ist laut Team:

* Website
* Mentale Gesundheit
* Krankheitsbedingte Ausfälle

### Sprint Zusammenfassung

Burndown-Chart:

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

Velocity – aktuell bei : 15

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

## Sprint 4

### Sprintplanung

Dauer: 26.04.2023 – 24.05.2023 (länger wegen Verzögerungen und Ausfällen)

Ausgewählte User Stories: Wechseln WLAN / Hotspot  
  
Verbleibende User Stories aus vorherigem Sprint: Websitedesign, Ein- und Ausschalten

Anzahl Story points: 56

Ausgewählte Punkte aus der Impediment Liste: 3 🡪 2 davon sind aus vorherigem Sprint

### Sprint Demo

Erledigt wurden:

* Websitedesign
* Ein- und Ausschalten

Anzahl erledigte Story Points: 48

Nicht erledigt wurden:

* Wechseln WLAN / Hotspot

Anzahl nicht erledigte Story Points: 8

Grund der Nichterledigung: Schwierigkeiten bei der Ausführung

### Sprint Retrospektive

(Bewertung des Sprints auf reetro.app)

Gut lief laut Team während Sprint4 folgendes:

* Zusammenfügen

Nötige Items/Aktionen laut Team:

* Mehr Organisation in der Dokumentationsschreibung und -zusammenfügung

### Sprint Zusammenfassung

Burndown-Chart:

Ein Bild, das Text, Reihe, Diagramm, Screenshot enthält.

Automatisch generierte Beschreibung

Velocity – aktuell bei : 23,25

Ein Bild, das Text, Screenshot, Zahl, Diagramm enthält.

Automatisch generierte Beschreibung

## Sprint 5

### Sprintplanung

Dauer: 31.05.2023 – 14.06.2023 🡪 Abschlusssprint

Ausgewählte User Stories: Code abschließen, Website ausbauen, Möglichkeit mehrere Teller  
  
Verbleibende User Stories aus vorherigem Sprint: Wechseln Wlan / Hotspot

Anzahl Story points: 50

Ausgewählte Punkte aus der Impediment Liste: 4 ausgewählt 🡪 1 davon ist aus vorherigem Sprint

### Sprint Demo

Erledigt wurden:

* Code abschließen
* Website ausbauen

Anzahl erledigte Story Points: 2

Nicht erledigt wurden:

* Möglichkeit mehrere Teller
* Wechseln vom Wlan zum Hotspot

Anzahl nicht erledigte Story Points: 48

Grund der Nichterledigung: Zu kompliziert für die Zeit

### Sprint Retrospektive

(Bewertung des Sprints auf reetro.app)

Gut lief laut Team während Sprint5 folgendes:

* Design und Funktionalität verbinden
* Abschluss

### Sprint Zusammenfassung

Burndown-Chart:

Ein Bild, das Text, Screenshot, Diagramm, Reihe enthält.

Automatisch generierte Beschreibung

Velocity – aktuell bei : 19

Ein Bild, das Text, Screenshot, Zahl, Diagramm enthält.

Automatisch generierte Beschreibung

# Installation / Software deployment

Als erstes wird das Projekt von unserem Github Repository heruntergeladen, und auf dem Raspberry Pi auf dem Desktop entpackt. Als nächstes werden die unter Kapitel 5.5.2 genannten SW-Komponenten installiert. Mit „Apt install Python3“ wird python installiert, falls nicht vorhanden. Ebenfalls muss mit „apt install socat“ das socat Paket heruntergeladen werden, das für das Port forwarding benötigt wird. Mit „pip install flask flask\_sqlalchemy flask\_login flask\_bcrypt flask\_wtf wtforms email\_validator gunicorn” werden alle pip Abhängigkeiten heruntergeladen und installiert.

Ein Bild, das Screenshot, Text, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung

Falls gewollt sind nun ebenfalls unter „crontab -e“ die in 5.0.4 gezeigten Cornjobs zu erstellen. Diese werden beim Systemstart ausgeführt und starten sowohl das Script um den Port weiterzuleiten sowieso das eigentliche Script um den Gunicorn Webserver zu starten. Von hier an funktioniert das Projekt und auf den Webserver man angesteuert werden.

# Projektabschluss

## Projektzusammenfassung

Abschließend lässt sich sagen, dass das Projekt zum Großteil gut gelaufen ist. Die Hauptfunktionen der Website funktionieren und wir haben ein ansprechendes Design. Die Seite ist durch ein Login geschützt und es können weitere Accounts hinzugefügt werden, sobald man einmal mit einem Account eingeloggt ist. Man kann Einstellungen für den Drehteller über eine eigene Seite speichern die lokal am Raspberry PI in einer JSON-Datei gespeichert werden, welche dann über einen Start Button abgerufen werden, und über ein anderes Skript dann angewandt werden. Ebenfalls gibt es einen Knopf, um den Teller wieder zu stoppen.

Einige Anforderungen haben es zeitlich bedingt nicht in das Projekt geschafft. Zum einen Teil das Wechseln von WLAN zu einem eigenen Hotspot oder das Zeitabhängige Ein- bzw. Ausschalten.

Durch das Projekt wurden viele neue Sachen gelernt. Von der Webprogrammierung mit dem Flask-Python Framework bis zu einigen Kleinigkeiten in Linux. Des Weiteren wurden erste Schritte mit Figma gelernt, um ansprechende Mockups zu erstellen.

Es gibt nicht viele Punkte, die man anders machen würde. Es war eine gute Idee, Flask zu benutzen, da es bereits viele Tutorials im Internet dazu gibt, aber auch weil die gegebenen Scripts, um die einzelnen Teile des Tellers anzusteuern, auch schon in Python geschrieben sind. Ein Punkt, der anders gemacht werden würde, wäre die Teameinteilung. Einige Tasks haben viel zu lange gedauert.

## Attachments

|  |  |
| --- | --- |
| Beschreibung | Link |
| Github Repo mit allen Dateien | https://github.com/DavareDavare/T1\_TableCast |