Hochschule Augsburg University of Applied Sciences

Fakultät für Informatik

Bachelorarbeit

Studienrichtung
Informatik

Bau einer automatisierten Klimakammer zur Anzucht von Pflanzen mit dem Raspberry Pi

Prüfer: Prof. Dr. Hubert Högl

Verfasser:
Benjamin Ochocki
Kurze Wertachstraße 5
86153 Augsburg
+49 176 62850248
benjamin.ochocki@proton.me
Matrikelnr.: 2048974

Hochschule für angewandte Wissenschaften Augsburg
An der Hochschule 1
86161 Augsburg
Telefon: +49 (0)821-5586-0
Fax: +49 (0)821-5586-3222
info@hs-augsburg.de

Überblick

Das Aufrechterhalten der bestmöglichen Bedingungen zum Keimen von Samen mit einem üblichen Plastikgewächshaus erwies sich als sehr aufwändig. Das Fehlen von Zeit führte zu vielen fehlgeschlagenen Anzucht-Versuchen. Deshalb wird der Bau einer automatisierten Klimakammer von Grund auf in einer Bachelorarbeit zusammengefasst. Dabei werden die theoretischen Grundlagen als auch der gesamte Bau und die Programmierung erläutert. Damit der Endnutzer die Kammer leicht bedienen kann, wird eine Benutzerschnittstelle implementiert. Abschließend kann die fertige Klimakammer genutzt und nachgebaut werden.

Diese Arbeit soll als Anleitung zum Bau einer solchen Kammer dienen und wird inklusive aller Source Code Dateien unter folgender Adresse der Öffentlichkeit zur Verfügung gestellt:

https://github.com/BenjaminOchocki/rpi-growfridge

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

CO2	Kohlenstoffdioxid
COB	Chip on Board
DLI	Daylight Integral
HTTPS	Hypertext Transfer Protocol Secure
LED	Light Emitting Diode
MP	Megapixel
PAR	Photosynthetic Active Radiation
PHP	Hypertext Preprocessor
PPF	Photosynthetic Photon Flux
PPFD	Photosynthetically Active Photon Flux Density
SD	Secure Digital
SMARTS	Simple Model of the Atmospheric Radiative Transfer of Sunshine
SSH	Secure Shell Protocol
UV	Ultraviolette-Strahlung
UVA	Ultraviolette-Strahlung aus dem Bereich 315 nm - 400 nm
UVB	Ultraviolette-Strahlung aus dem Bereich 280 nm - 315 nm

Verzeichnis der Listings

1 Einleitung

1.1 Motivation

Nach dem Abschluss des allgemeinen Abiturs bewarb ich mich direkt um einen Platz an der Fachhochschule Augsburg für den Bachelor in Informatik. Durch die starke Beliebtheit dieses Studienganges wurde mir leider mitgeteilt, dass es zu diesem Zeitpunkt keinen freien Platz gäbe. Mir wurde ein Platz auf der Warteliste gegeben und mit Glück könnte ich noch zu einem späteren Zeitpunkt im Semester nachrücken. Damit die Wartezeit sinnvoll genutzt wurde, machte ich aus meiner Situation das Beste und fing an, mich mit Pflanzen und ihrer Anzucht auseinander zu setzen.

Als ich Mitte 2012 in meine Studentenwohnung zog, gab es vor meiner Tür einen schmalen Streifen Erde, in dem bisher nur 4 Bäume Kirschlorbeer wuchsen. Also fragte ich nun meinen Vermieter, ob ich auf diesem Stück etwas anpflanzen dürfe. Ich bekam die Erlaubnis, dort zu machen, was ich wolle, jedoch unter der Bedingung, dass der Kirschlorbeer nur zurückgeschnitten und nicht komplett entfernt würde. Dies stellte kein Problem dar.

Die Wahl der Pflanzen fiel mir nicht leicht. Fest stand jedoch, dass ich Chilis anpflanzen wollte, denn ich esse selber liebend gerne scharf und wollte somit noch einen weiteren Nutzen aus meinem neuen Hobby ziehen. Die ersten Samen waren schnell bestellt, dazu gab es ein kleines Plastikgewächshaus und ein paar Kokos-Quell Tabs. Die Sache schien simpel. Die Quell Tabs werden in die Wanne des Plastikgewächshauses gestellt und die Wanne wird dann mit ein wenig Wasser aufgefüllt. Nach kurzer Zeit saugen die Quell Tabs das Wasser auf und quellen an. Danach steckt man ein bis zwei Samen in das kleine Loch. Sind alle Quell Tabs bestückt, wird der Deckel auf die Gewächshaus-Wanne aufgelegt und das Gewächshaus an einen hellen, warmen Platz gestellt.

Schnell zeigten sich die ersten Auffälligkeiten. Gerade bei Chilis gibt es viele exotische Sorten. Teilweise kann es bei Samen exotischer Chili-Arten bis zu 6 Wochen dauern, bis die Keimung eintritt. Während der gesamten Zeit sollten die Bedingungen der Situation angepasst werden. Leider war dies mit dem kleinen Plastikgewächshaus nicht so einfach wie gedacht. Die Luftfeuchtigkeit verringerte sich schnell, da der Deckel auf der Wanne nur auflag und nicht dicht war. Damit es dauerhaft zur

gewünschten relativen Luftfeuchtigkeit von durchschnittlich ca. 85% kam, musste ständig Wasser nachgefüllt werden. Vor allem junge Pflanzen und noch ungekeimte Samen brauchen Sauerstoff, um zu überleben. Wurde das Gewächshaus zu oft gegossen, bildeten sich Algen, wodurch eine undurchlässige Schicht entstand und jeglichen Sauerstoff blockierte. Dadurch erstickten die Samen, bevor sie keimen konnten. Fand ein zu geringer Luftaustausch im Gewächshaus statt, so bildete sich auf den Quell Tabs unerwünschter Schimmel.

All diese Bedingungen bis zur Keimung der Samen im optimalen Bereich zu halten, war mit viel mehr Arbeit als Spaß am Hobby verbunden. Ich hatte noch wenig Erfahrung mit Chilis und ihren unterschiedlichen Wachstumsstadien, sodass sämtliche Anzuchtversuche ohne Erfolg blieben. Frustriert von den ersten Ergebnissen meines neuen Hobbys, dachte ich mir, dass es doch einen besseren Weg geben müsse, wie ich an gesunde junge Chilipflanzen komme. Die ersten Versuche mit einem Pflanzenzelt aus dem Internet waren nicht zufriedenstellend. Die Luftfeuchtigkeit konnte nur in Verbindung einer zu hohen Temperatur eingehalten werden. Um die Temperatur zu senken, musste konstant frische, trockene Luft in das Zelt gepumpt werden. Dabei wurde viel Luftfeuchtigkeit aus dem Zelt transportiert. Die Pflanzen fingen an, Wasser zu transpirieren und die Verdunstung von Feuchtigkeit aus der Erde wurde beschleunigt. Letztendlich musste ich wieder öfters gießen. Durch die trockene Luft in der Winterzeit vertrockneten mir dadurch beinahe meine ungekeimten Samen. Dieser Aufbau war somit auch keine Lösung für mich. Bei einer gemütlichen Runde mit Freunden kam mir die Idee, einen Kühlschrank als Gewächshaus zu benutzen. Ein anwesender Freund hatte zufällig einen alten defekten Kühlschrank bei sich zu Hause und bot an, mir diesen baldmöglichst zu überlassen.

Sobald der Kühlschrank ankam wurde dieser über die nächsten Wochen modifiziert. Um eine größere Anzahl an Samen ansetzen zu können, wurde der Kühlschrank auf die Seite gelegt. Mit einer 36 mm durchmessenden *Bohrkrone*, wurden zwei Löcher in die ehemalige Decke des Kühlschranks gebohrt. Ein Loch in die obere Hälfte und eines in die untere Hälfte der ehemaligen Decke. Um frische Luft in den Kühlschrank befördern zu können, wurde das untere Loch mit einem *Radiallüfter* versehen. An die Rückwand des Kühlschranks wurde eine Heizmatte geklebt. Der Lüfter und die Heizmatte wurden beide an einen *Controller* angeschlossen, der vorher über AliExpress bestellt wurde. Dieser *Controller* hat ein Sensormodul zur Messung der Temperatur und der relativen Luftfeuchtigkeit. Es können Sollwerte und deren *Schalthysterese* eingestellt werden.

Als Lichtquelle wurden mehrere *LEDs* verschiedener Farben mit Heißkleber an die neue Decke des Kühlschranks geklebt. Der Innenraum sah wie in Abbildung 1.1 aus.

Abbildung 1.1: Innenraum des ersten *Growfridge*-Prototyps

Die ersten Tests zeigten, dass sich die Temperatur relativ gut halten ließ. Lediglich die Luftfeuchtigkeit konnte nicht konstant gehalten werden. Aus diesem Grund wurde als Dampfsperre eine mit Watte gefüllte Socke in das obere Loch gesteckt, um die Feuchtigkeit im Kühlschrank zu halten. Dadurch verringerte sich der Luftdurchsatz und der Lüfter lief kontinuierlich. Mit diesen Ergebnissen ging es an die Planung des nächsten Prototyps. Diesmal wurde der neue Ansatz mit einem aufrecht stehenden, funktionierenden Kühlschrank umgesetzt.

Der Aufbau des zweiten Prototyps wurde deutlich komplexer und konnte auf seine Wirksamkeit hin getestet werden. Durch ein Sensormodul zur Messung der relativen Luftfeuchtigkeit inklusive der Temperatur und ein 8-Kanal Relaisboard, sollten alle benötigten Funktionen sichergestellt werden. In einer Datenbank wurden alle Messwerte und Einstellungen der Peripherie gespeichert. Nach mehreren Versuchen ergab sich ein funktionierendes System. Der zweite, deutlich komplexere Prototyp konnte seinen ersten Funktionstests unterzogen werden. Abbildung 1.2 zeigt dabei den Innenraum des zweiten Prototyps. Die Unzugänglichkeiten des ersten Prototyps waren beseitigt, jedoch zeigten sich neue Herausforderungen im Umgang mit dem zweiten Prototyp.

Bei der Planung war nicht bedacht worden, dass sich an der Rückwand des Kühlschranks bei laufendem Betrieb Kondenswasser bilden würde. Dies führte dazu, dass die Abtropfwanne über dem Kompressor des Kühlschranks langsam voll und letztendlich über lief. Unter dem Kühlschrank bildete sich folglich eine Wasserpfütze. Als Notlösung wurde eine Plastikfolie an die Rückwand geklebt, wodurch das Wasser zurück in den Luftbefeuchter laufen konnte. Dieser war allerdings nur für den

Abbildung 1.2: Innenraum des zweiten *Growfridge*-Prototyps

Betrieb mit einer geschlossenen Flasche ausgelegt, weshalb der Reiseluftbefeuchter daraufhin über lief. Um dies zu verhindern, wurde ein Loch in das Reservoir des Luftbefeuchters gebohrt, damit überschüssiges Wasser in den Überlaufcontainer ablaufen konnte. Jetzt waren die Grundfunktionen des Prototyps sichergestellt. Durch Einloggen in die Datenbank des *Growfridges* und geschicktes Verändern von Parametern, konnte das gewünschte Verhalten des *Growfridges* produziert werden. Mit dem zweiten Prototyp war es ohne großen Aufwand möglich, eine festgelegte Temperatur und relative Luftfeuchtigkeit in den jeweiligen Grenzenbereichen einzuhalten. Dabei wurde die Peripherie vom Raspberry Pi gesteuert. Durch diese Verbesserung des Aufbaus erhöhte sich die Keimrate auf nahezu 100 %. Eine Garantie, dass jeder Samen keimt, gibt es nicht.

Mit der verbesserten Lichtquelle des zweiten Aufbaus erhöhte sich nicht nur die Anzahl der steuerbaren Lichtfarben, sondern es erhöhte sich auch die Komplexität der Parametrierung. Jeder dieser Farben sollte einzeln ansteuerbar sein und um dies zu realisieren, war es notwendig, ein 16-kanaliges Relaisboard einzubauen. Da ein *Proof of Concept* wichtig war, endete diese Steuerung vorerst in einem unschönen Chaos. In Abbildung 1.3 sieht man die 16-Kanalsteuerung in Ihrer vollen Pracht.

Die neuen Herausforderungen des zweiten Prototyps und das Chaos seiner Schaltung mussten unbedingt beseitigt werden. Nach mehreren Monaten und einigen Tests waren genug Ergebnisse gesammelt, um daraus ein ansehnliches Projekt zu machen. Als der Zeitpunkt kam, die Bachelorarbeit anzumelden, wurde dieses Projekt bei Herrn Prof. Dr. Hubert Högl vorgestellt. Dieser nahm den Vorschlag voller Begeisterung an.

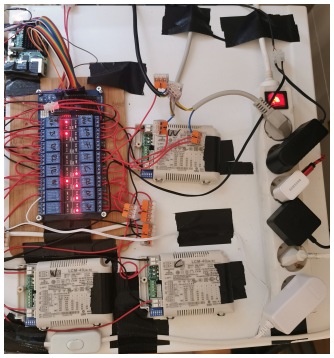


Abbildung 1.3: Chaotische Steuerung des zweiten *Growfridge*-Prototyps

1.2 Ziel der Arbeit

Das Projektziel ist eine automatisierte Klimakammer, im folgenden *Growfridge* genannt, die über eine einfache grafische Benutzerschnittstelle bedient werden soll. Über diese Schnittstelle soll ein aktuelles Bild des Innenraums zu sehen sein. Zu diesem Bild soll die aktuelle Temperatur und Luftfeuchtigkeit der Kammer angezeigt werden. Ebenfalls sollen Informationen der aktuellen Einstellungen angezeigt werden. In einem geschützten Bereich der Benutzerschnittstelle soll der Benutzer die Konfiguration der unterschiedlichen Konditionen vornehmen können. Der zeitliche Verlauf dieser Konditionen soll ebenfalls dort einstellbar sein. Zur grafischen Interpretation aller Messwerte soll eine extra Schnittstelle hinter einem Login verwendbar sein. Dort soll der Benutzer die Möglichkeit haben, ein eigenes Dashboard mit allen Messewerten zusammenstellen zu können. Die Anleitung zum Bau eines *Growfridges* soll inklusive *Source Code* und dieser Bachelorarbeit zur Verfügung gestellt werden. Die Daten werden durch ein *Github Repository* bereitgestellt.

1.3 Aufbau der Arbeit

Damit ein allgemeines Grundverständnis darüber besteht, wie der *Growfridge* funktioniert, wird zuerst auf die Grundlagen eingegangen. Danach wird der generelle Aufbau des *Growfridges* beschrieben.

Nach Abschluss der Theorie wird systematisch gezeigt, wie ein *Growfridge* nachgebaut werden kann. Zuerst wird das Raspberry Pi installiert. Danach wird die gesamte Hardware in den Kühlschrank eingebaut und der *Growfridge* das erste Mal softwaremäßig eingerichtet. Dabei wird auf die verschiedenen Skripte des Systems genauer eingegangen. Nach einer Übersicht zur allgemeinen Bedienung des *Growfridges*, wird genauer auf die Benutzerschnittstelle eingegangen.

Mit einem Fazit und einem Ausblick in die Zukunft dieses Projektes wird diese Bachelorarbeit abgerundet.

1.4 Typographische Konventionen

Zum besseren Verständnis dieser Arbeit werden einige typografische Konventionen festgelegt.

Fachbegriffe, Dateien und Verzeichnisse werden *kursiv* formatiert.

Genutzte Bash-Kommandos, Nutzernamen, Passwörter oder einzeilige Codefragmente werden in `Proportionalschrift mit grauem Hintergrund` dargestellt. Längerer Quelltext wird in Form von Codeblöcken, die als Listings bezeichnet werden, dargestellt. Um die Lesbarkeit der Listings zu verbessern, ist die Größe der seitlichen Ränder auf ein Minimum reduziert.

Liegt eine besondere Betonung auf einem Wort, so wird dieses **fettgedruckt** dargestellt. Sonstige Hervorhebungen werden ebenfalls **fettgedruckt**.

Abkürzungen werden bei erster Nennung kurz erläutert und können zudem im Abkürzungsverzeichnis auf Seite IX nachgeschlagen werden.

2 Grundlagen

Vor dem Bau, sollten als Erstes die Konditionen bekannt sein, die später hergestellt werden müssen. Diese Konditionen bilden die Grundlagen und somit das Fundament, auf dem die Funktionsweise des *Growfridges* basiert. Je nach Pflanze und Vorhaben können die optimalen Konditionen stark schwanken. Der primäre Zweck dieses Projekts ist das Heranziehen junger Chilipflanzen verschiedener ausgewählter Samen.

2.1 Der Samen

Der Samen ist der Ursprung neuer Nachkommen einer Pflanze. Damit sich eine neue Pflanze aus dem Samen bilden kann, muss dieser zur Keimung gebracht werden. Unter Keimung versteht man allgemein, die Sprossung eines Samens.

Es gibt zwei Arten, die *epigäische* und die *hypogäische* Keimung. Bei der *epigäischen* Keimung, keimt der Samen unter der Erde und schiebt seine *Keimblätter* an die Erdoberfläche. Meistens hängt dabei der Samen noch an den *Keimblätter*n und fällt später ab. Beispiele für epigäische Keimer sind die Sonnenblume, die Chili oder der Raps.

Bei der *hypogäischen* Keimung bleibt der Samen mit den *Keimblättern* unter der Erde. Erst die Primärblätter treiben aus der Erde hinaus und werden fotosynthetisch aktiv. Beispiele hierfür sind die Mango oder die Avocado.

Unabhängig vom Typ der Keimung, gibt es unterschiedliche Faktoren, welche zur Keimungsauslösung eines Samens führen. Neben Wasser und Sauerstoff spielt die Temperatur eine wichtige Rolle. Abhängig vom Samen können auch bestimmte Lichtverhältnisse oder besondere Einwirkungen notwendig sein.

Der sogenannte *Lichtkeimer* braucht Helligkeit, um keimen zu können. Dabei werden die Samen meist nur leicht an die Erde gedrückt, um feucht genug zu bleiben. Es kann eine sehr dünne Deckschicht aus Erde darüber gelegt werden. Die Samen brauchen kein direktes Sonnenlicht, sondern nur eine gewisse Helligkeit in ihrer Umgebung. Beispiele für Lichtkeimer sind der Lavendel, der Basilikum oder der Mohn.

Das Gegenteil bilden die *Dunkelkeimer*. Diese beginnen auch ohne Licht zu keimen und können deshalb einige Zentimeter in der Erde vergraben werden. Ist der Samen zu tief vergraben oder die Erde über ihm zu dicht gepackt, bekommt er schlecht Luft und kann ersticken. Beispielsweise die Tomate, die Chili oder die Zucchini zählen zu den *Dunkelkeimern*.

Einige Samen benötigen eine Kälteperiode, bevor sie keimen können. Diese nennt man *Kaltkeimer*. Dabei quillt der Samen durch die Feuchtigkeit an und verändert innerhalb der Kälteperiode sein Verhältnis von keimhemmenden zu keimfördernden Substanzen und treibt letztendlich aus. Die Temperaturen der Kälteperiode variieren je nach Samen zwischen 0° und 10° C. Typische Kaltkeimer sind der Mohn, der Bärlauch oder der Roggen.

Die *Warmkeimer* bilden zu den *Kaltkeimern* das Gegenteil. Diese benötigen eine Mindesttemperatur von 5° C. Es gibt einige frostresistente *Warmkeimer* welche bereits im Februar ausgesät werden können. Die restlichen *Warmkeimer* werden erst im Mai ausgesät, um die Frostperiode zu umgehen. Beispiele für Warmkeimer sind der Mais, der Basilikum oder die Gurke.

Unter *Feuerkeimern* versteht man Samen, die nur durch ein Brandereignis zur Keimung gelangen. Zu ihnen gehören der australische Zylinderputzer oder die echte Akazie. Experimente haben gezeigt, dass Temperaturen von bis zu 100° C über eine Minute den Samen des böhmischen Storchschnabels zum Schwellen und späterer Keimung brachten (Dahlgren (1923)).

2.2 Die Temperatur

Die *Fotosynthese* kann im Allgemeinen zwischen 0° und 30° C ohne Probleme stattfinden. Pflanzen, die in warmen Sommerzonen wachsen, vertragen Temperaturen von 7° bis 40° C. Einige Wüstenpflanzen sind sogar in der Lage, bei Temperaturen zwischen 15° und 45° C weiter *Fotosynthese* zu betreiben (Sage u. Kubien (2007)). Steigt die Temperatur weiter an, so sinkt die *Fotosyntheserate* rapide ab. Des Weiteren wurde beobachtet, dass Pflanzen deutlich mehr CO2 assimilieren, wenn sie hohen Temperaturen ausgesetzt sind. Ist der Samen gekeimt, sollten ihm die bestmöglichen Konditionen für seine Art geboten werden. Sollte der Samen mit dem *Growfridge* herangezogen werden und später einmal nicht in seiner heimischen Zone wachsen, so sollte die Temperatur innerhalb des *Growfridges* dem finalen Standort der Pflanze angepasst werden. Ist das Licht im *Growfridge* eingeschaltet (es ist Tag), so sollte sich die Temperatur nahe der maximalen Temperatur des finalen Standortes befinden. Dabei ist zu beachten, dass die Erde, in der die Pflanze heranwächst, ebenfalls

auf diese Temperatur erwärmt wird und eine zu hohe Bodentemperatur der Pflanze schaden kann. Ist das Licht ausgeschaltet (es ist Nacht), sollte sich die Temperatur auf das zu erwartende Minimum bei Nacht am finalen Standort einstellen. Somit ist der spätere Übergang vom *Growfridge* in die freie Natur deutlich stressfreier für die noch jungen Pflanzen und sie müssen sich nicht erst an die neuen Konditionen anpassen.

2.3 Die Luftfeuchtigkeit

Der Samen darf vom Setzen bis zur Keimung nicht austrocknen. Damit die Erde, in der sich der Samen befindet, nicht zu schnell austrocknet, wird in den kleinen Gewächshäusern eine hohe Luftfeuchtigkeit eingehalten. Die Luft innerhalb des Gewächshauses kann nur einen bestimmten Anteil an Wasser aufnehmen. Dieser Anteil verdunstet entweder von der Erde oder wird bereits von den frischen Pflanzen transpiriert. Kann die Luft kein weiteres Wasser mehr aufnehmen (maximale Sättigung erreicht), so verliert die Erde bei einem komplett luftdichten Raum kein weiteres Wasser.

Sobald der Samen gekeimt ist, sollte die Luftfeuchtigkeit auf 50 bis 65%, je nach Pflanzenart, gesenkt werden. Bei zu hoher Luftfeuchtigkeit bilden sich schnell ungewollte Pilze, welche die jungen Pflanzen befallen und zerstören können. Zusätzlich sollte die Luft innerhalb eines Gewächshauses immer umgewälzt werden. Dies hilft nicht nur dabei, die Bildung von Pilzen zu verhindern, sondern regt auch ein stabiles Stängelwachstum an. Werden junge Pflanzen nicht genug Wind ausgesetzt, kann dies in einem unzureichenden Wachstum des Stängels resultieren, der somit im weiteren Verlauf nicht in der Lage ist, das Gewicht der Pflanze zu tragen. Senkt man die Luftfeuchtigkeit zu weit, wird die Pflanze zu stark zur Transpiration gebracht. Dieser Vorgang kostet Energie, die der Pflanze dann zum Wachsen fehlt.

Wird der Samen mit dem *Growfridge* gekeimt, so sollten sich die Tag- und Nacht-Werte der relativen Luftfeuchtigkeit an ihr Maximum bzw. Minimum des finalen Standortes anpassen. Sollten diese Werte nicht bekannt sein, können diese in den Klimadiagrammen der jeweiligen Stadt nachgesehen oder mit einer günstigen Wetterstation ermittelt werden.

2.4 Das Licht

Sobald ein Samen gekeimt ist wachsen seine *Keimblätter*. Durch diese bekommt der *Keimling* seine Energie, um sich weiterzuentwickeln. Deshalb ist es wichtig, dass aus-

reichend Licht für die bestmögliche Entwicklung der Pflanzen zur Verfügung gestellt wird. Die künstliche Beleuchtung von Pflanzen ist ein komplexes Thema. Im Folgenden wird deshalb nur grundlegend auf die wichtigsten Punkte eingegangen. Fachbegriffe sollen genauer erklärt und dadurch ein solides Grundverständnis für Pflanzenlampen erzeugt werden. Sind die Grundlagen abgeschlossen, wird genauer auf die eingebaute Lichtquelle des *Growfridges* eingegangen.

2.4.1 Das menschliche Auge

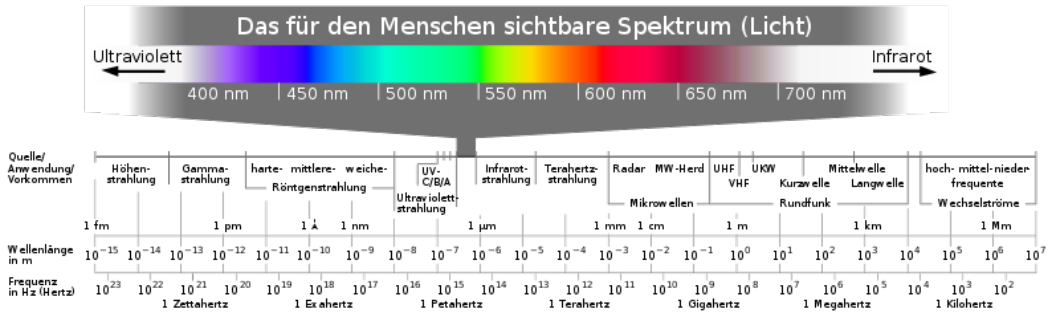


Abbildung 2.1: Das für den Menschen sichtbare Spektrum (Licht) (Horst Frank / Phrood / Anony (2008))

Wenn man allgemein von Licht spricht, handelt es sich meist um elektromagnetische Wellen mit *Wellenlängen* von etwa 380 bis 780 Nanometern. In Abbildung 2.1 ist eine Übersicht der verschiedenen Farben innerhalb dieses Bereiches zu sehen. Photonen dieser *Wellenlängen* lösen im menschlichen Auge eine Helligkeitsempfindung aus. Zusätzlich hat das menschliche Auge zwei unterschiedliche Empfindlichkeiten. Die Helligkeitsempfindung bei Tag (*photopic*) hat seine maximale Empfindlichkeit bei 555 nm *Wellenlänge*. Dies entspricht der Farbe Grün. Die Helligkeitsempfindung bei Nacht (*scotopic*) hat seine maximale Empfindlichkeit bei 507 nm *Wellenlänge*, was der Farbe Türkis nahekommt. Ebenfalls erwähnenswert ist, dass das menschliche Auge beim *Nachtsehen* etwa 2,5-mal so empfindlich ist, wie beim *Tagsehen*. Die Einheit, in welcher Helligkeit für das menschliche Auge gemessen wird, ist Lumen (lm). Diese Einheit berücksichtigt die Empfindlichkeit des menschlichen Auges beim *Tagsehen*.

Gibt ein Lampenhersteller diesen Wert bei einer seiner Pflanzenlampen an, so ist dieser Wert nur bedingt nützlich. Normiert man die *Hellempfindlichkeitskurve* (engl. *Luminous efficiency function*), so hat diese ihr Maximum von 1 bei der *Wellenlänge* von 555 nm. Vergleichsweise hat die Farbe Blau mit 470 nm einen Wert von ca. 0,091 und die Farbe Rot bei 660 nm einen Wert von ca. 0,061 (Sharpe u. a. (2005)). Demnach braucht eine Lampe nur eine hohe Lichtabgabe an Grün nahe der

555 nm und ihr Lumen-Wert ist ebenfalls sehr hoch. Da moderne Pflanzenlampen meist nicht mehr aus einfarbigen *LEDs* bestehen, kann mit der Lumen-Angabe des Herstellers ein nützlicher Wert abgeschätzt werden. In Abbildung 2.2 sieht man einen Vergleich der beiden relativen Hellempfindlichkeiten des menschlichen Auges bei Tag und Nacht.

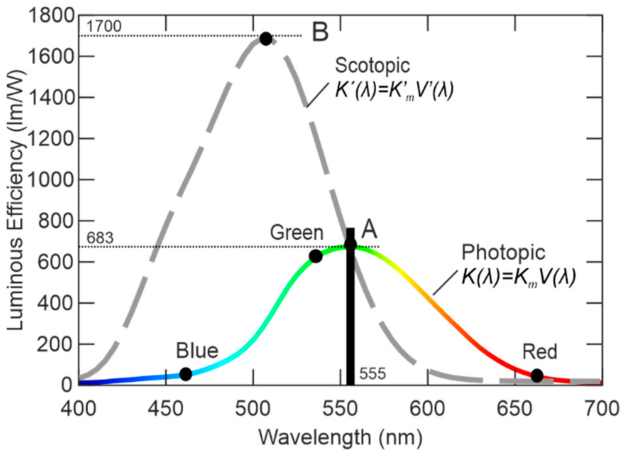


Abbildung 2.2: Die sensorischen Eigenschaften des menschlichen Sehvermögen (Leuchter u. a. (2021))

2.4.2 Das Sonnenspektrum

Das auf der Erde eintreffende Licht wird von der Sonne in unserem Sonnensystem produziert. Trifft das Sonnenlicht auf die *Atmosphäre* der Erde, so werden die unterschiedlichen *Wellenlängen* des Lichts verschieden stark gefiltert. Der Grund dafür liegt in der Zusammensetzung der Luft. Je nach Element werden unterschiedliche *Wellenlängen* absorbiert. Hat das Licht die gesamte Luftmasse (engl. *air mass*, kurz AM) durchquert, trifft das gefilterte Licht auf der Erdoberfläche ein. In Abbildung 2.3 wurden die Daten der letzten spektralen Messung der Internationalen Raumstation ISS vom 6. Juni 2022 genutzt (Laboratory for Atmospheric and Space Physics (2022)), um das Spektrum der Einstrahlung auf dem Boden mit einer Luftmasse von 1.5 zu berechnen. Der Wert AM1.5 wurde 1980 als Durchschnitt aller Staaten der USA festgelegt (Gonzalez u. Ross (1980)). Je nach Dicke der *Atmosphäre* und Einstrahlwinkel des eintreffenden Lichts, werden bestimmte *Wellenlängenbereiche* des Lichtes mehr oder weniger gefiltert. Dabei sind die berechneten Strahlungsintensitäten im *Wellenlängenbereich* von 400 nm bis 700 nm besonders interessant.

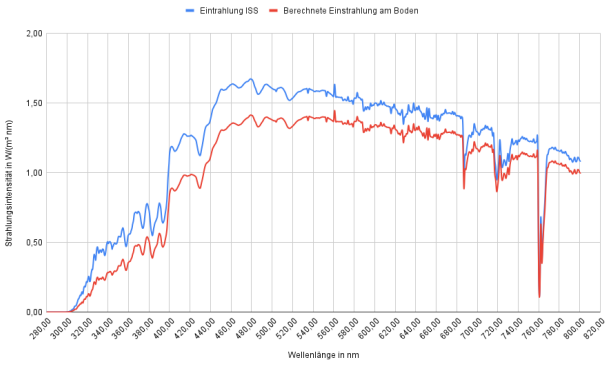


Abbildung 2.3: Einstrahlung der Sonne auf der Internationalen Raumstation ISS und auf dem Boden mit einer Luftmasse (AM) von 1,5 im *Wellenlängenbereich* von 280 nm bis 800 nm

Diese werden nicht nur für die *Fotosynthese* benötigt, sondern liegen auch im sichtbaren Bereich des menschlichen Auges. Für die Berechnung des Spektrums wurde das *Simple Model of the Atmospheric Radiative Transfer of Sunshine*, kurz SMARTS, des nationalen Labors für erneuerbare Energien der USA mit seinen Standardparametern genutzt (Gueymard (2001)).

2.4.3 Die Farbtemperatur

Das auf der Internationalen Raumstation ISS eintreffende Spektrum der Sonne im *Wellenlängenbereich* von ca. 140 nm bis ca. 10 cm kann dabei näherungsweise mit dem Spektrum eines idealen Schwarzen Strahlers der Temperatur von etwa 5900 Kelvin (K) beschrieben werden. Allgemein wird dieses Spektrum auch Farbtemperatur genannt. Farbtemperaturen kleiner 3500 K nennt man Warmweiß, da sie einen typischen gelb-orange bis roten Stich haben. Ein neutrales Weiß hat Farbtemperaturen von 3500 K bis 4500 K. Ab Farbtemperaturen von 4500 K spricht man von einem Kaltweiß, da sich der Blauanteil erhöht. Die Sonne hat somit ein kaltweißes Spektrum.

Besonders in den Abendstunden hat jeder schon einmal den roten Himmel bewundert. Diese Färbung kommt durch die sogenannte *Rayleigh-Streuung* zustande. Je dicker die Luftschicht ist, durch die das Sonnenlicht passieren muss, desto mehr streut sich das Licht. Dabei werden die blauen Farben des Lichtes stärker gestreut als die roten Farben. Das Resultat ist die typische Rotfärbung des Himmels am Morgen und Abend, da das Licht zu diesen Zeiten durch eine besonders dicke Luftschicht passieren muss. Trifft das Sonnenlicht senkrecht auf der Oberfläche ein, so hat die Luftschicht ihre minimale Dicke von AM1.0 erreicht. Das Spektrum des eintreffen-

den Lichts hat zu diesem Zeitpunkt die geringste Filterung durch die *Atmosphäre* erfahren.

2.4.4 Die fotosynthetisch aktive Strahlung

McCree hat die Reaktion von Nutzpflanzen auf Licht unterschiedlicher *Wellenlängen* studiert. Dabei hat er 22 verschiedene Pflanzen in Gewächshäusern zu seinen Tests im fotosynthetisch aktiven Strahlungsbereich (engl. Photosynthetic Active Radiation, kurz PAR) von 400 nm bis 700 nm unterzogen. Am Ende seiner Studie hat er den Durchschnitt aller Ergebnisse als die sogenannte McCree Kurve publiziert (McCree (1971)). Seine Ergebnisse waren die Grundlage vieler weiterer Studien im Bereich Pflanzen und der *Fotosynthese*. Auch das Deutsche Institut für Normung hat viele Jahre später die Norm DIN 5031-10 veröffentlicht und zuletzt 2018 aktualisiert. Aus ihr gehen, neben dem Aktionsspektrum, auch die unterschiedlichen Auswirkungen von Licht verschiedenster *Wellenlängen* auf Pflanzen hervor. Eine Liste dieser Auswirkungen ist in Tabelle 2.1 zu finden. Der *fotosynthetisch aktive Photonenfluss* (engl. *Photosynthetic Photon Flux*, kurz PPF) wird in der Einheit Mikromol pro Sekunde (μmol/s) gemessen. Um die fotosynthetisch aktive Photonenstromdichte (engl. *Photosynthetically Active Photon Flux Density*, kurz PPFD) zu beschreiben, wird die eingestrahlte Menge an Licht über die Fläche von einem Quadratmeter innerhalb einer Sekunde gemessen. Das bedeutet, ihre Einheit ist Mikromol pro Sekunde je Quadratmeter (μmol/m^{2}s).

Tabelle 2.1: Näherungsweise Zuordnung der verschiedenen optischen Strahlungsbereiche zu den wesentlichen physiologischen Wirkungen auf Pflanzen (DIN-Normenausschuss Lichttechnik, 2018)

Wellenlänge	Physiologische Wirkungen
$>$ 1000 nm	ausschließlich thermische Effekte
1000 - 700nm	vornehmlich Stammwachstum
700 nm - 610 nm	höchster fotosynthetischer Effekt, *Chlorophyllsynthese*, Blütenbildung (*Fotoperiodismus*)
610 nm - 510 nm	geringste physiologische Wirkung im Bereich der sichtbaren Strahlung
510 nm - 400 nm	Absorption durch *Carotinoide*, zweiter Höchstpunkt der *Chlorophyllabsorption* und *Fotosynthese*, Wachstumseffekte und formende Wirkungen
400 nm - 315 nm	*Fotosynthese*, Wachstumseffekte und formende Wirkungen
315 nm - 280 nm	Schädigende Wirkung auf die meisten Pflanzen
$<$ 280 nm	rasches Absterben

Beim Vergleich der beiden *Aktionsspektren* in Abbildung 2.4 fällt vor allem auf, dass es zu einer besonders hohen fotosynthetischen Reaktion im *Wellenlängenbereich* von

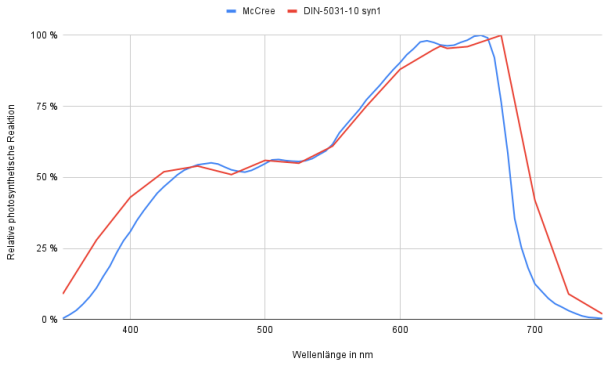


Abbildung 2.4: Vergleich der *Aktionsspektren* von McCree (1971) und DIN 5031-10 (2018) zur relativen fotosynthetischen Reaktion von Pflanzen auf Licht verschiedener *Wellenlängen*

600 nm (orange) bis 660 nm (rot) kommt. Der Irrglaube, Pflanzen würden kein grünes Licht zur *Fotosynthese* nutzen, wird dadurch ebenfalls widerlegt.

2.4.5 Die künstlichen Lichtquellen

In den 1960er Jahren kamen die ersten kommerziellen *Natriumdampf Hochdrucklampen* auf den Markt. Diese waren bereits in der Lage, Licht mit einer Effizienz von bis zu 100 Lumen pro Watt zu produzieren. Das Spektrum dieser Lampen hatte ihr Maximum zwischen ca. 560 nm und 570 nm, wie in Abbildung 2.5 zu erkennen ist. Ihre Farbtemperatur liegt dabei zwischen 2200 K und 2700 K. Sie sind charakteristisch für ihr oranges Licht in Straßenlaternen. Die Leistungsaufnahme der Lampen beträgt zwischen 250 und 1000 Watt. Mit diesen Lampen war es bereits möglich Pflanzen komplett ohne Sonne vom Samen bis zur Ernte in einer Kammer wachsen zu lassen. Aufgrund der hohen Leistungsaufnahme der Lampen mussten diese mit extra Zubehör abgeschirmt und gekühlt werden. Da diese Lampen in 360 Grad abstrahlen, brauchten sie zusätzlich einen Reflektor, um so viel Licht wie möglich in Richtung der Pflanzen strahlen zu lassen.

Die kleinere Variante der *Natriumdampf Hochdrucklampen* sind die *Leuchtstofflampen*. Ihre Effizienz reicht von 50 bis 100 Lumen pro Watt. Das Spektrum dieser Lichtquelle ist etwas ausgeglichener, wie in der Abbildung 2.6 zu sehen ist. Die Leistung reicht von 4 bis 100 Watt. Auch diese Lampen haben einen Abstrahlwinkel von 360 Grad. Aufgrund ihrer geringen Leistung werden diese Lampen eher selten als einzige Lichtquelle zur Pflanzenzucht genutzt. Da *Leuchtstofflampen* nicht gekühlt werden müssen, eignen sie sich hervorragend zur Anzucht von *Keimlingen*.

Da die meisten Pflanzen schon relativ früh hohe Strahlungsintensitäten vertragen, müssten mehrere *Leuchtstofflampen* genutzt werden, um das volle Potenzial der jun-

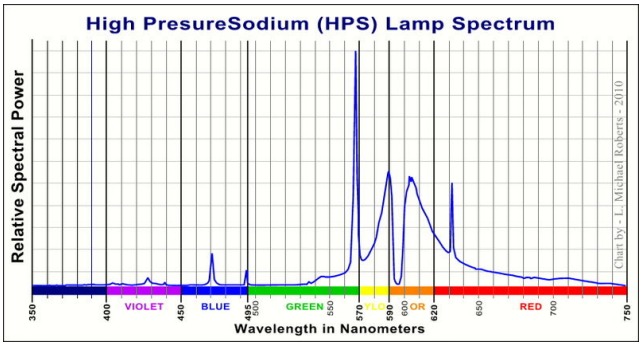


Abbildung 2.5: Lichtspektrum einer typischen Natriumdampf Hochdrucklampe LMRoberts (2010)

gen Pflanzen in der vegetativen Phase zu nutzen. Früchtetragende Pflanzen würden zwar unter diesen Lampen wachsen, jedoch wären die Früchte klein und unausgereift. Hinzu kommt, dass das Spektrum der Leuchtstofflampe durch ihren verwendeten Leuchtstoff festgelegt ist. Sollten mehrere Lichtfarben gewünscht sein, wird der Einbau mehrerer *Leuchtstofflampen* aufwändig.

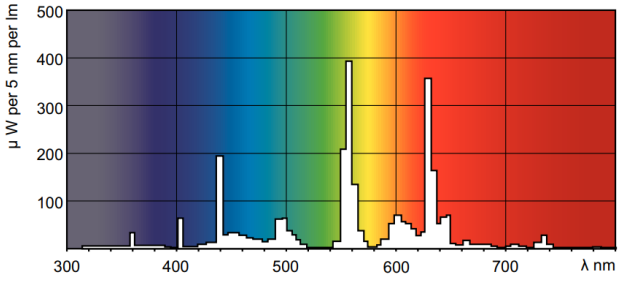


Abbildung 2.6: Lichtspektrum aus dem Datenblatt der Philips MASTER TL5 HO Xtra 80W/840 SLV/20 der Farbtemperatur 4000 K Philips Lighting (2022)

Im Jahr 2022 brachte Samsung die bisher leistungsstärkste *Mid-Power Horticulture LED* mit 3,14 μmol/J auf den Markt. Ihr verbessertes *LED*-Modell namens LM301H EVO produziert dabei ein weißes Vollspektrum. Dieses Vollspektrum beinhaltet eine gleichmäßigere Kombination von *Wellenlängen* im Bereich 380 nm bis 780 nm. Das verbesserte Spektrum des neuen Modells unterdrückt sogar, im Vergleich zu herkömmlichen *LED*-Vollspektren, ungewolltes mikrobielles Wachstum auf den Pflanzen (Samsung (2022)). Diese *LEDs* haben sich aufgrund ihrer Vorteile in der Pflanzenzucht weltweit etabliert. Das Modell LM301H EVO gibt es in unterschiedlichen Farbtemperaturen. Aufgrund der hohen Effizienz und der hohen Konfigurierbarkeit von *LEDs*, wird die Lichtquelle dieses Projekts aus mehreren dieser *LEDs* in unterschiedlichen Farbtemperaturen bestehen. Die *LEDs* sind in verschiedene Kanäle (engl. Channels) aufgeteilt. So ist es möglich, unterschiedliche Farben

des Lichtes einzeln zu steuern. In Abbildung 2.7 sieht man das typische Spektrum einer Samsung LM301H EVO *LED* der Farbtemperatur 3000 K.

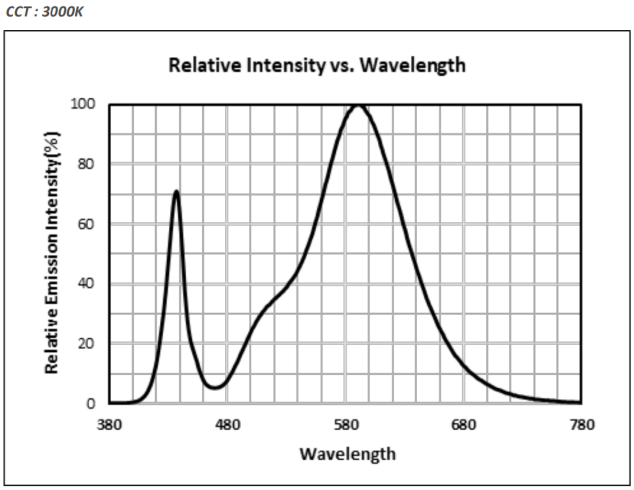


Abbildung 2.7: Spektrum aus dem Datenblatt der Samsung LM301H EVO in der Farbtemperatur 3000 K Samsung (2020)

2.4.6 Der Emerson-Effekt

In vielen neuartigen *LED*-Pflanzenlampen wird zum weißen Vollspektrum immer öfter einfarbiges (*monochromes*) Rot (660 nm) und Infrarot (730 nm) hinzugefügt. Auch die Lichtquelle in diesem Projekt besitzt zusätzliches Rot und Infrarot. Die Grundlage dafür bildet neben der höchsten fotosynthetischen Reaktion bei 660 nm, der sogenannte Emerson Effekt. Dieser wurde von Robert Emerson 1957 das erste Mal beschrieben (NAS (1957)). Robert Emerson fand bei seinen Experimenten heraus, dass für die *Fotosynthese* zwei *Fotosysteme* zuständig sind. Er testete die *Fotosyntheserate* beider Systeme einzeln und auch in Kombination. Dabei fiel ihm auf, dass die *Fotosyntheserate* bei einer Kombination der beiden *Fotosysteme* höher ist, als die Summe der einzelnen *Fotosyntheseraten*. Die *Wellenlänge*, welche die maximale Reaktion im *Fotosystem* 1 auslöst, liegt bei ca. 680 nm und für das *Fotosystem* 2 liegt die *Wellenlänge* bei ca. 750 nm (Kitajima u. Butler (1975)). Als wichtiger Hinweis an dieser Stelle bleibt zu sagen, dass Experimente im Jahr 1954 gezeigt haben, dass die Licht-keimenden Samen von Salat bei einer Bestrahlung von rotem Licht (640 - 670 nm) besser und schneller keimten, eine Bestrahlung mit infrarotem Licht (720 - 750 nm) die Keimung jedoch verhinderte (Borthwick u. a. (1954)). Aus diesem Grund sollte die Möglichkeit gegeben sein, die infraroten *Wellenlängen* der Lichtquelle je nach Anwendungsfall manuell zu steuern.

2.4.7 Die ultraviolette Strahlung

Ein weiterer Effekt, den jeder schon einmal gesehen hat, ist das Schwächeln junger Pflanzen, die das erste Mal in die direkte Sonne gestellt werden. Der Grund dafür ist die ultraviolette Strahlung (kurz UV-Strahlung) im Sonnenlicht. Jeder Mensch wird bei genug Sonnenschein braun. Dafür sind Pigmente in der Haut verantwortlich, die produziert werden, um auf die Beschädigung der Haut zu reagieren. Das Gleiche passiert bei Pflanzen. Hat eine Pflanze keine Pigmente, werden ihre Blätter beschädigt und sie transpiriert übermäßig viel. Dabei kann sie den Wasserhaushalt in den Blättern nicht aufrechterhalten und fängt an auszutrocknen und schließlich zu verwelken. Als Abwehrreaktion werden von der Pflanze Pigmente produziert.

Wie in Abbildung 2.3 zu sehen ist, hat das eintreffende Sonnenlicht einen nicht unbeachtlichen Anteil an ultravioletter Strahlung im Bereich von kleiner 400 nm. Bestrahlt man junge Pflanzen vor dem Hinaussetzen einige Zeit mit UV-Strahlung, so haben sie genügend Zeit Pigmente aufzubauen und sich auf direktes Sonnenlicht vorzubereiten. Bei diesem Prozess ist die *Wellenlänge* und Intensität der UV-Strahlung besonders zu beachten. Je kürzer die *Wellenlänge* des einstrahlenden Lichts ist, desto höher ist die Energie des Photons und desto höher ist auch der verursachte Schaden. UV-Strahlung wird generell in drei Arten eingeteilt. Die UVA-Strahlung hat einen *Wellenlängenbereich* von ca. 400 nm bis 315 nm. UVB-Strahlung beginnt ab ca. 315 nm und reicht bis etwa 280 nm *Wellenlänge*. Ab 280 nm bis ca. 200 nm nennt man die Strahlung UVC-Strahlung. Durch eigene Tests kann der Bereich von 365 nm bis 400 nm besonders empfohlen werden, um junge Pflanzen auf das Sonnenlicht vorzubereiten. Kürzere *Wellenlängen* verursachten Probleme für die jungen Pflanzen und sollten deshalb vermieden werden. Für jede Pflanze muss die optimale Menge an UV-Strahlung durch Tests herausgefunden werden.

Es gibt spezielle Gasentladungslampen, die entweder für die UV-Therapie beim Menschen (Phplips Lighting (2015)) oder für spezielle Reaktionen bei Pflanzen (Agro-Max (2015)) hergestellt wurden. Beide Lampen strahlen einen hohen Anteil an UV-Strahlung aus. Im Bereich der *LEDs* gibt es ebenfalls immer mehr Produkte, die einen sehr schmalen *Wellenlängenbereich* mit UVA bzw. UVB-Strahlung abgeben. Diese sind allerdings hochpreisig und haben nur eine geringe Lebensdauer. Da, wie bereits erwähnt, kein positiver Effekt für Pflanzen in diesem jungen Stadium gefunden werden konnte, wurden die Experimente eingestellt.

Der Schaden durch UV-Strahlung am Menschen ist nicht unerheblich. Wird mit ultravioletter Strahlung gearbeitet, ist aus diesem Grund zu jedem Zeitpunkt Schutzkleidung in Kombination mit einer Schutzbrille zu tragen. Besonders im Bereich der

Pflanzenzucht gibt es Lichtquellen, die in der Lage sind, extrem hohe Strahlungsmengen im Bereich UVB und UVA abzugeben. Bei zu geringem Abstand können schon wenige Sekunden Bestrahlung ausreichen, um irreparable Schäden an Haut und Augen zu verursachen.

2.4.8 Die Lichtintensität

Die Intensität des Lichtes spielt für die Entwicklung der Pflanze ebenfalls eine wichtige Rolle. Es gibt Schattenpflanzen die nur wenig Licht benötigen. Ein Beispiel sind Fensterblätter (*Monstera*) oder Flamingoblumen (*Anthurium*). Als Gegenbeispiel kann man beispielsweise den Mais betrachten. Dieser kann sehr viel direktes Sonnenlicht vertragen und zur Produktion von Früchten nutzen. Wird eine Pflanze zu viel Licht ausgesetzt, tritt die sogenannte *Fotoinhibition* in Kraft. Dabei leitet die Pflanze Schutzmaßnahmen ein, um sich vor der erhöhten Strahlung zu schützen. Dadurch sinkt die *Fotosyntheserate* und das Wachstum rapide ab.

Vor allem angehende Hobbygärtner machen gerne den Fehler, ihren jungen Pflanzen zu viel Licht zu geben. Meistens ist der Gedankengang der, dass die Pflanze unter echtem Sonnenlicht ja auch hohen Intensitäten ausgesetzt ist und dies somit kein Problem darstellen sollte. Dabei wird vergessen, dass das Sonnenlicht durch die Tageszeit, die Dicke der *Atmosphäre*, den Einstrahlungswinkel und einigen weiteren Faktoren beeinflusst wird. In Abbildung 2.8 ist die Einstrahlungsintensität der Sonne im Verlauf eines Tages zu sehen.

Ist die Pflanze bereits in der vegetativen Phase angekommen, hat sie schon einige Wurzeln ausgebildet, welche die Pflanze mit ausreichend Flüssigkeit versorgen, um bei hohen Einstrahlungswerten ausreichend transpirieren zu können. Sind diese Wurzeln noch nicht ausreichend ausgebildet, resultiert zu viel Licht in einem raschen Austrocknen junger Pflanzen. Der *fotosynthetisch aktive Photonenfluss* (PPF) verliert mit zunehmendem Abstand zu seiner Lichtquelle an Intensität. Aus diesem Grund wird in der Pflanzenzucht die Oberfläche betrachtet, auf der das Licht eintrifft. Die *fotosynthetisch aktive Photonenflussdichte* (PPFD) beschreibt die eintreffende Menge an Licht aus dem PAR-Bereich, die pro Sekunde je Quadratmeter einstrahlt. Dieser Wert ist die Grundlage zur Einschätzung der Lichtintensität an einer bestimmten Stelle. Direktes Sonnenlicht kann unter klarem Himmel eine Intensität von über 2000 μmol/m^{2}s erreichen (Varella u. a. (2011)).

Wird bei einer Pflanzenlampe der PPF Wert angegeben, so existieren meist auch PPFD Übersichten zu verschiedenen Abständen. Durch diese Angaben zeigt der Hersteller dem Nutzer die Intensitäten auf, die unter der Lampe bei bestimmten Abständen zu erwarten sind. Diese Angaben wurden überwiegend unter optimalen

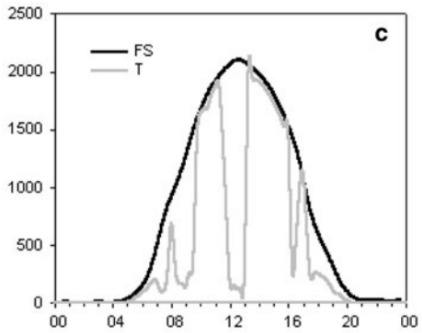


Abbildung 2.8: PPFD Tagesverlauf des Sonnenlichts, gemessen bei klarem Himmel unter vollem Sonnenlicht (FS) und unter Bäumen (T) im Mittsommer in Canterbury, Neuseeland Varella u. a. (2011)

Bedingungen ermittelt. Bei unterschiedlicher Umgebungstemperatur oder altersbedingter Degradierung kommt es zu Abweichungen dieser Angaben. Da sich die Intensität nicht nur durch den Abstand, sondern auch durch den Winkel zur Strahlungsquelle verändert, wird die betrachtete Oberfläche in ein Gitternetz vieler gleich großer Quadrate aufgeteilt. Im Zentrum jedes dieser Quadrate wird mit einem Sensor ein Wert zu einem festgelegten Abstand zur Lampe ermittelt. Die Werte werden mit unterschiedlichen Lampenabständen zur Oberfläche ermittelt. Die Ergebnisse werden in sogenannten PPFD Übersichten (engl. PPFD Maps) dargestellt. Abbildung 2.9 zeigt Beispiele von PPFD Maps.

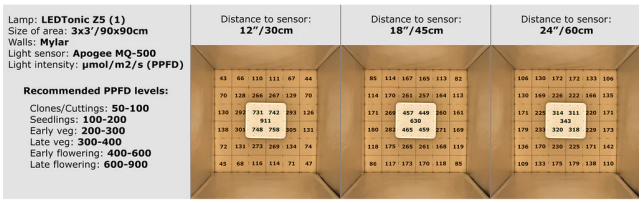


Abbildung 2.9: Typische Übersicht der PPFD Werte einer Pflanzenlampe LEDTonic (2020)

Die Angaben zu den unterschiedlichen Intensitäten bezog der Author dieser Grafik auf unterschiedliche Pflanzen. Werte ab 200 μmol/m²s reichen für die frühe vegetative Phase von Tomaten, Chilis und Cannabis.

2.4.9 Das Daylight Integral

Die Menge an Licht, die auf einer Fläche von einem Quadratmeter über eine Zeitspanne von 24 Stunden eintrifft, wird als das sogenannte *Day Light Integral*, kurz DLI, bezeichnet. Seine Einheit ist Mol pro Quadratmeter je Tag (mol/m²d, engl. *mol per square meter per day*). Im Südwesten der USA wurden im Sommer 2008

DLI-Werte von bis zu 65,2 mol/m²d gemessen (Ciolkosz (2008)). Dabei lag der Durchschnitt aller 201 Messpunkte über das Jahr hinweg bei 29,6 mol/m²d. Diese Werte repräsentieren nur den Durchschnitt der USA, jedoch geben ihre Längen- und Breitengrade einen Eindruck davon, wie sich die Sonneneinstrahlung von Spanien bis Norddeutschland verhält. In Abbildung 2.10 wurde eine Tabelle erstellt, die die jeweiligen DLI Werte zu unterschiedlichen Intensitäten bei unterschiedlicher Einstrahlungsdauer anzeigt.

DLI Tabelle. DLI Werte in mol/m²d

PPFD in µmol/m²s		50	100	200	300	400	500	600	700	800	900	1000
Zeit in Stunden												
8		1,4	2,9	5,8	8,6	11,5	14,4	17,3	20,2	23,0	25,9	28,8
9		1,6	3,2	6,5	9,7	13,0	16,2	19,4	22,7	25,9	29,2	32,4
10		1,8	3,6	7,2	10,8	14,4	18,0	21,6	25,2	28,8	32,4	36,0
11		2,0	4,0	7,9	11,9	15,8	19,8	23,8	27,7	31,7	35,6	39,6
12		2,2	4,3	8,6	13,0	17,3	21,6	25,9	30,2	34,6	38,9	43,2
13		2,3	4,7	9,4	14,0	18,7	23,4	28,1	32,8	37,4	42,1	46,8
14		2,5	5,0	10,1	15,1	20,2	25,2	30,2	35,3	40,3	45,4	50,4
15		2,7	5,4	10,8	16,2	21,6	27,0	32,4	37,8	43,2	48,6	54,0
16		2,9	5,8	11,5	17,3	23,0	28,8	34,6	40,3	46,1	51,8	57,6
17		3,1	6,1	12,2	18,4	24,5	30,6	36,7	42,8	49,0	55,1	61,2
18		3,2	6,5	13,0	19,4	25,9	32,4	38,9	45,4	51,8	58,3	64,8
	DLI Werte	< 5	5 - 10	10 - 15	15 - 20	20 - 25	25 - 30	30 - 35	35 - 40	40 <		

Abbildung 2.10: DLI Tabelle verschiedener Intensitäten mit unterschiedlicher Einstrahlungsdauer

2.4.10 Das eingebaute Led-Board

Die Lichtquelle aus diesem Projekt hat laut Hersteller bei einer Bestromung aller Kanäle mit 350mA eine Gesamtleistung von ca. 39 Watt. Dabei hat sie einen *Licht-Output* von ca. 100 µmol/s. Der Abstand der Lampe zur Pflanze kann im *Growfridge* maximal 40 cm betragen. Die bestrahlte Fläche beträgt 0,2 m² (40 × 50 cm). Geräte zur Messung der fotosynthetischen Photonenflussdichte (PPFD) sind mit einem aktuellen Preis von rund 700 Euro leider immer noch sehr hochpreisig (Growmart (2022)). Theoretisch ist eine Berechnung möglich, jedoch sind dies nur grobe Näherungswerte.

Der Hersteller gibt die jeweiligen PPF Werte bei einem bestimmten Strom an. Sind alle Werte bekannt, kann ein grober Näherungswert unter folgenden Bedingungen berechnet werden:

- der gesamte *Licht-Output* des Boards wird durch eine einzige *LED* verursacht
- diese *LED* hat einen Abstrahlwinkel von 120 Grad
- die Reflektionen an der Kühlschrankwand werden vernachlässig
- die Verringerung der Intensität durch Entfernung zum Strahlungsmittelpunkt wird vernachlässigt
- die Strahlungsintensität wird für einen Abstand von 40 cm berechnet

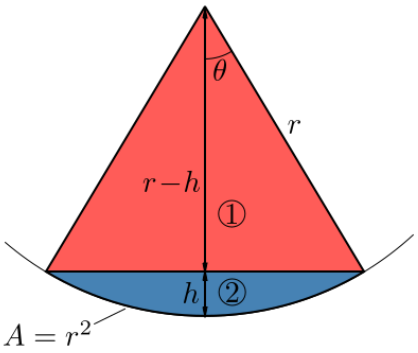


Abbildung 2.11: Zeigt den Schnitt durch das Zentrum eines Kegels, der einen Winkel von einem Steradiant in einer Sphäre mit Radius r hat Inductiveload (2007).

Dabei entsteht ein Lichtkegel mit der Höhe des Abstands, wie in der Abbildung 2.11 zu sehen ist. Der zentrale Mittelpunkt der Fläche unter dem Lichtkegel bekommt dabei die höchste Intensität ab. Erhöht sich der Abstand vom Mittelpunkt des Kegels, verringert sich die Intensität des Lichts. Aus diesem Grund sind die Boards auch aus vielen einzelnen kleinen *LEDs* mit einigem Abstand aufgebaut. Dadurch entsteht eine relativ gleichmäßige Ausstrahlung über die Fläche des *LED*-Boards. Mit der folgenden Formel kann der PPFD-Wert der Lichtquelle im Zentrum der Ausleuchtungsfläche für einen bestimmten Abstand berechnet werden:

$$PPFD = \frac{PFD}{r^2 4\Pi(\sin(\frac{\Omega}{4}))^2}$$

$PPFD$	Menge an fotosynthetisch aktiver Strahlung die pro Quadratmeter je Sekunde einstrahlt
PPF	Menge an fotosynthetisch aktiver Strahlung die pro Sekunde einstrahlt
r	Abstand von der *LED* zur Pflanze
Ω	Abstrahlwinkel der *LED*

Das Daylight-Integral kann demnach wie folgt berechnet werden:

$$DLI = PPFD * 10^{-6} * h * 3,6 * 10^{3}$$

$$=> DLI = PPFD * h * 3,6 * 10^{-3}$$

DLI	Menge an fotosynthetisch aktiver Strahlung die pro Quadratmeter innerhalb eines Tages einstrahlt
$PPFD$	Menge an fotosynthetisch aktiver Strahlung die pro Quadratmeter je Sekunde einstrahlt
h	Dauer des einstrahlenden Lichtes in Stunden

Werden die Werte für die unterschiedlichen Intensitäten der Lichtquelle über eine Dauer von 18 Stunden pro Tag berechnet, so erhält man die Tabelle 2.2. Bei einer Bestromung von mehr als 500 mA sollte aus wärmetechnischen Gründen eine Kühlung für das *LED*-Board eingebaut werden.

Tabelle 2.2: Theoretische Näherungswerte der Einstrahlung im Zentrum des Lichtkegels bei einem Abstand von 40 cm

Bestromung in A	PPF in µmol/s	PPFD in µmol/m²s	DLI in mol/m²d
Rot: 0,35 Weiss: 0,35	105	209	13,5
Rot: 0,7 Weiss: 0,7	206	410	26,6
Rot: 1 Weiss: 1	212	422	27,3
Rot: 1 Weiss: 1,4	262	521	33,8
Rot: 1 Weiss: 1,7	309	615	39,8

Um junge Pflanzen nicht zu großem Lichtstress auszusetzen, sollte die Intensität des eintreffenden Lichts geringer als das pflanzentypische Maximum des aktuellen Pflanzenstadiums sein. Die notwendige Menge Licht wird durch eine erhöhte Bestrahlungsdauer gewonnen. Je nach Pflanze muss der bestmögliche Wert durch sukzessives Herantasten empirisch ermittelt werden. Bei eigenen Experimenten wurden für junge Chili-Pflanzen über 18 Stunden Lichtphase mit einem DLI-Wert von maximal 30 mol/m²d die besten Ergebnisse erzielt. Größere Werte führten dazu, dass die noch jungen Pflanzen Anzeichen von Lichtstress zeigten. Da die Pflanzen relativ früh aus der Klimakammer genommen werden, wird das *LED*-Board mit 350 mA bestromt.

2.5 Die Led Energieversorgung

Eine Light Emitting Diode, kurz *LED*, ist ein Halbleiter, der Licht produziert, sobald Spannung anliegt und Strom durch ihn hindurch fließt. Nutzt man eine Reihenschaltung mehrerer *LEDs*, so muss die Mindestspannung der Schaltung erreicht werden, damit Strom hindurchfließt und Licht erzeugt wird. *LEDs* Arrays gibt es in unterschiedlichen Konfigurationen. So kann beispielsweise eine einzelne *LED* genutzt

werden, die manuell angeschlossen wird. Meistens hat ein Anbieter diese *LEDs* bereits auf einem Aluminium Board mit integrierten Leiterbahnen verbaut und bietet diese dann als lange Streifen unterschiedlicher Breite und Länge oder in rechteckigen Boards in Größen bis zum DIN A3 Format an. Eine weitere Art der Anordnung ist der sogenannte *Chip on Board*, kurz COB genannt. Bei dieser Art werden hunderte *LEDs* auf einer kleinen Fläche von ca. 30 × 30 mm angebracht. Je nach Konfiguration besitzt ein Produkt eine andere Gesamtspannung, die von der Stromquelle (Treiber) geliefert werden muss. Dabei sollte die Wahl des Treibers gut bedacht sein. Im Folgenden wird auf zwei Arten von Treibern genauer eingegangen.

2.5.1 Der Konstantspannungstreiber

Der Konstantspannungstreiber erzeugt eine konstante Spannung. Dabei variiert der Strom, welcher durch die angeschlossene Schaltung fließt, solange, bis die festgelegte Spannung erreicht ist. Bei Erreichen der Nennleistung, kann der Treiber den Strom nicht weiter erhöhen. So kann beispielsweise ein Treiber mit einer konstanten Spannung von 42 V und einer Nennleistung von 100 W maximal 2,38 A liefern. Ein Konstantspannungstreiber wird meist für die Kombination mehrerer Boards mit gleicher Spannung genutzt, diese werden parallel an den Treiber angeschlossen. Bei einer Parallelschaltung ist die Spannung in allen Boards gleich und der Gesamtstrom teilt sich entsprechend an den parallel geschalteten Boards auf. Bei baugleichen Boards fließt durch jedes Board der gleiche Strom. Abbildung 2.12 zeigt eine Parallelschaltung von drei Widerständen.

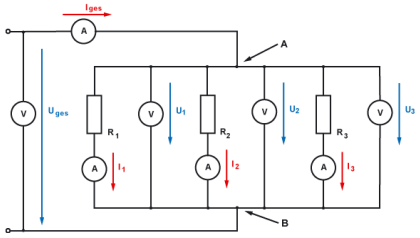


Abbildung 2.12: Parallelschaltung von Widerständen Elektronik Kompendium (2020a).

Bei der Parallelschaltung wird jeder Pluspol eines Bauteils mit dem Pluspol des Treibers verbunden. Gleiches passiert mit jedem Minuspol eines Bauteils und dem Minuspol des Treibers.

2.5.2 Der Konstantstromtreiber

Der Konstantstromtreiber erzeugt einen konstanten Strom und variiert dabei die Spannung so lang, bis der festgelegte Strom erreicht wird. Hat der Treiber seine

Nennleistung erreicht, so kann er die Spannung nicht weiter erhöhen. So kann beispielsweise ein Treiber mit einem Konstantstrom von 350 mA und einer Nennleistung von 50 W seinen Strom bis maximalen Spannung von 142,86 V aufrechterhalten. Ein Konstantstromtreiber wird meist für die Reihenschaltung mehrerer *LEDs* genutzt, damit durch alle *LEDs* der gleiche Strom fließt. Bei einer Reihenschaltung ist der Strom in allen *LEDs* gleich und die Gesamtspannung addiert sich aus den Einzelspannungen der *LEDs*.

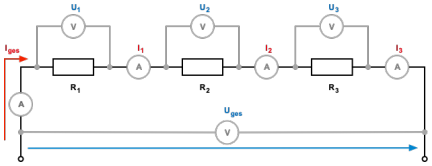


Abbildung 2.13: Reihenschaltung von Widerständen Elektronik Kompendium (2020b).

Bei einer Reihenschaltung wird der Pluspol des Treibers mit dem Pluspol der ersten *LED* verbunden. Danach dient der Minuspol der ersten *LED* als Pluspol der zweiten *LED*. Diese Anreihung geschieht bis zur letzten *LED*. Bei der letzten *LED* wird der Minuspol der *LED* mit dem Minuspol des Treibers verbunden.

Der Anschluss der Bauelemente sollte nur von Fachpersonal durchgeführt werden. Die Treiber bilden den Übergang vom *Wechselstromnetz* zum geregelten *Gleichstromkreisnetz*. Beides kann für den Menschen sehr gefährlich werden. Die Schaltung sollte vor dem Betrieb ausführlich geprüft werden. Der Treiber sollte auf seine Art und Angaben geprüft werden. Je nach Fehlerart sind mehrere *LEDs* betroffen und können schnell zu umfangreichen Schäden führen.

2.6 Die Peripherie

Zur Peripherie gehören neben der Klimakammer alle Geräte, die benötigt werden, um die Konditionen innerhalb dieser Kammer zu steuern und zu regeln. Als Klimakammer wird ein handelsüblicher Standkühlschrank ohne Gefrierfach genutzt. Um die Klimakammer beheizen zu können, wird in den Kühlschrank eine Heizmatte eingebaut. Damit die Luftfeuchtigkeit erhöht werden kann, wird in die Kühlschranktür ein Reiseluftbefeuchter eingesetzt. Dieser kann mit einem mitgelieferten Adapter mit nahezu jeder PET-Flasche genutzt werden. Um die Luftfeuchtigkeit senken zu können, werden zwei 54 mm große Öffnungen höhenversetzt in die Kühlschranktür gebohrt. Die untere Öffnung wird mit einem *Radiallüfter* ausgestattet, damit Luft in den Kühlschrank befördert werden kann. Um das Luftgemisch innerhalb des Kühlschranks homogen zu halten, wird ein kontinuierlich laufender 12V Lüfter innerhalb

der Kammer montiert. Zur Beleuchtung wird das benötigte *LED*-Board an die Decke des Kühlschranks montiert.

2.7 Das Raspberry Pi

Das Raspberry Pi wird zur Steuerung des 8-Channel Relaisboards genutzt. Jedes Peripheriegerät wird von einem der Channel des Relaisboards gesteuert. Die notwendigen Sensorwerte aus dem Kühlschrank, welche das Raspberry Pi benötigt, um zu entscheiden, welches Endgerät aktiviert werden soll, werden über einen Temperatur- und einen Luftfeuchte-Sensor ermittelt. Da die angeschlossenen Endgeräte teilweise mit 230V Wechselspannung betrieben werden, ist beim Anschluss auf die notwendigen Sicherheitsbestimmungen zu achten.

2.8 Die Software

Die Software muss generell drei Aufgaben erfüllen. Es müssen die Sensoren und das Relaisboard betrieben werden. Des Weiteren sollen die notwendigen Dienste, die zur Steuerung und Darstellung benötigt werden, durch eine zentrale Stelle bedienbar werden. Mithilfe dieser Dienste kann der Benutzer eine Kondition erstellen, die automatisch vom *Growfridge* gehalten wird. Diese Konditionen werden in einer Datenbank abgespeichert und ihr Ablauf in einem Ablaufplan dargestellt. Wurde das Ende eines Eintrages erreicht, wird der Start- und Endzeitpunkt um einen Tag erhöht. Auf diese Weise werden alle durchlaufenen Einträge täglich wiederholt. Die Messwerte werden in einer Datenbank gespeichert und können mittels eines Dashboards übersichtlich dargestellt werden.

3 Bau der Kammer

Der Bau der Kammer findet in zwei Schritten statt. Der erste Schritt ist die Installation des Raspberry Pis. Als Zweites findet der Einbau bzw. die Modifikation der Hardware statt. Anschließend kann mit der Einrichtung des Systems und des Benutzerinterfaces fortgefahren werden.

3.1 Die Stückliste

Damit der *Growfridge* gebaut werden kann, werden einige zusätzliche Bauteile und verschiedene Werkzeuge benötigt.

Tabelle 3.1: Liste aller benötigten Werkzeuge

Kreuz- und Schlitzschraubendreher verschiedener Grössen
Seitenschneider groß und für Feinelektronik
Abisolierzange (stellte sich als sehr wichtig heraus)
Lötkolben mit Spitzen unterschiedlicher Dicke
Haushaltsschere, Haushaltsmesser
Heißklebepistole mit mehreren Stiften
Akkubohrer mit verschiedenen Bohrern / Bohrkrone 54 mm und 80 mm
Sechskantschlüssel verschiedener Größen
Metall Bogensäge

In der Tabelle A.1 werden alle Bauteile des *Growfridges* genannt und beschrieben.

3.2 Die Installation des Raspberry Pi

Augrund der fehlenden Treiberkompatibilität zwischen der neuesten Firmware-Version und der verwendeten 64 MP Kamera, wird in diesem Projekt auf ein Softwareupdate und ein Update der Firmware verzichtet. Für die Installation des Betriebssystems muss als Erstes eine aktuelle Version des *Raspberry Pi OS* heruntergeladen werden. Zum Zeitpunkt dieser Arbeit wird auf der Homepage das Raspberry Pi OS in 64 Bit mit der Kernel Version 5.15 und der Debian Version 11 angeboten. Der Download kann mit dem Browser oder dem *Raspberry Pi Imager* erledigt werden. Diesen

gibt es für Linux, MacOS und Windows und kann zum Zeitpunkt dieser Arbeit in der Version 1.7.2 auf der Homepage heruntergeladen werden (RaspberryPi Ltd (2022b)). Mit dem Imager wird danach das Image auf die SD-Karte geschrieben. Nachdem das entsprechende Betriebssystem und die dazugehörige SD-Karte ausgewählt wurde, erscheint im Bereich rechts unten ein Zahnrad, welches seit der Version 1.7.2 zusätzliche Optionen zur Konfiguration der Installation anbietet.

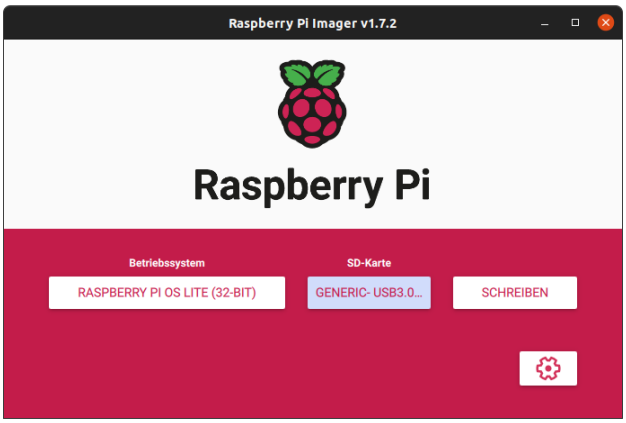


Abbildung 3.1: *Raspberry Pi Imager* v1.7.2 nach der Auswahl des Betriebssystems und der SD-Karte

In früheren Versionen des Betriebssystems war der Standard Benutzername immer `pi` und das dazugehörige Passwort lautete `raspberry`. Diese Einstellung wurde aus sicherheitstechnischen Gründen geändert (RaspberryPi Ltd (2022a)) und muss nun bei aktuellen Installationen vor dem Beschreiben der SD-Karte gesetzt werden. Alle weiteren Einstellungen im Imager sind optional, es empfiehlt sich jedoch, alle Optionen zu nutzen, um die Einrichtung des Raspberry Pis zu beschleunigen. Neben dem Benutzernamen und dem Passwort kann direkt die Wireless LAN Verbindung eingerichtet und der Zugriff per *SSH* aktiviert werden. Zusätzlich kann dem Raspberry Pi ein Hostname zugeteilt und die Standard-Spracheinstellungen vorgenommen werden. Aus sicherheitstechnischen Gründen empfiehlt es sich, auf den Standard-Benutzernamen `pi` mit dem Standardpasswort `raspberry` zu verzichten.

Nachdem der Schreibvorgang beendet ist, wird die SD-Karte in das Raspberry Pi eingesetzt und mit dem ersten Bootvorgang begonnen. Je nach Modell kann dieser Prozess einige Minuten dauern. Nachdem die SD-Karte in das Raspberry Pi gesteckt wurde und dieses hochgefahren ist, sollte direkt ein *SSH Key* mit dem Befehl `ssh-keygen -t ed25519` erzeugt werden. Dieser muss vom Nutzer in seinem *Github-Account* hinterlegt werden um später das *Git-Repository* zu klonen.

3.3 Die Hardware

Bevor mit dem Bau des *Growfridges* begonnen wird, müssen alle benötigten Elemente auf ihre Funktion geprüft werden. Defekte, die nach Einbau festgestellt werden, führen zu unnötiger Mehrarbeit. Besonders das Sensormodul und das Relaisboard sollten auf ihre einwandfreie Funktion überprüft werden. Die dazu notwendigen Programme können auf der jeweiligen Treiberseite der Hardware gefunden werden. Sämtliche Montagearbeiten sind im spannungsfreien Zustand durchzuführen.

Als Erstes sollte der Kühlschrank modifiziert werden. Es werden alle Einschübe bis auf den Untersten herausgenommen. Die Getränkehalter der Tür werden bis auf den Untersten und Obersten entfernt. Danach wird die Abdeckung der Kühlschrankbeleuchtung entfernt. Dahinter befindet sich der Drehknopf zur Regulierung der Temperatur. Der Regler schaltet den Kompressor je nach Temperatur ein oder aus. Nachdem alle angeschlossenen Leiter des Temperaturschalters entfernt wurden, wird der Schalter mit seinem Temperaturfühler vorsichtig aus dem Kühlschrank gezogen. In Abbildung 3.2 ist der ausgebaute Temperaturschalter samt Fühler zu sehen.

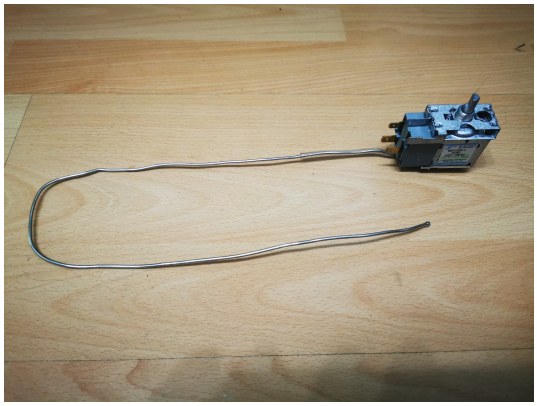


Abbildung 3.2: Bild des ausgebauten Temperaturschalters

Von den entfernten Leitern wurden zwei verwendet, um zu erkennen, ob die Tür geöffnet wurde. Die anderen schalten die Stromversorgung des Kühlschranks und die der Beleuchtung. Um den Betrieb des Kompressors zu gewährleisten, werden der schwarze und der braune Leiter elektrisch miteinander verbunden. Die restlichen Leiter werden alle einzeln isoliert. Die Öffnung, aus dem der Fühler des Temperatursensors gezogen wurde, wird abgedichtet. Dies ist wichtig, damit keine Feuchtigkeit entsteht und zur Schimmelbildung führt. Sind diese Schritte abgeschlossen, werden alle Leiter an der Kühlschrankwand mit Klebeband befestigt. Jetzt sollte der Kühlschrank noch einmal auf seine Funktion geprüft werden. Beim Einstecken des Netzsteckers läuft der Kompressor an. Ist die Funktion gewährleistet, wird der Stecker aus der Steckdose gezogen und wird so abgeschnitten, dass das verbleiben-

de Kabel am Kühlschrank möglichst lang bleibt. Dieses Kabel wird später an das Relaisboard angeschlossen.

Das *LED*-Board wird am Ende der Montagearbeiten mittig im Kühlschrank befestigt. Dazu werden in die Decke der Innenseite des Kühlschrankes vier Metallhaken eingedreht. Dabei ist darauf zu achten, dass die Löcher, die dabei entstehen abgedichtet werden. In Abbildung 3.5 kann das montierte *LED*-Board gesehen werden.

Als Nächstes sollten die Öffnungen in die Tür des Kühlschranks gebohrt werden. Dazu wird eine 54 mm Bohrkrone verwendet. Je nach Kühlschrank kann die Tür ausgebaut werden. Da für das Bohren der Öffnungen unter Umständen ein großer Druck erzeugt werden muss, empfiehlt es sich nach Möglichkeit, die Tür auszubauen und mit der Innenseite auf eine feste Unterlage zu legen. Das unterste Einhängefach legt die Position der unteren Öffnung für den *Radiallüfter* fest. Dort sollte sich später der Lüfter neben dem Luftbefeuchter befinden. Das obere Loch wird innerhalb des obersten Einhängefachs der Tür gebohrt. Anschließend wird die Tür wieder eingebaut.

Der Lüfter, der frische Luft in den Kühlschrank befördert, wird dann mit einem 2 mm dicken, doppelseitigen Schaumstoffklebeband befestigt. Das Schaumstoffklebeband erfüllt den Zweck, die Vibrationen des Lüfters im Betrieb zu dämpfen und einen leisen Betrieb zu ermöglichen. Zusätzlich wird ein 120 mm 12 V Lüfter an die Unterseite des oberen Einhängefachs der Tür geklebt, der zur konstanten Umwälzung der Luft im *Growfridge* genutzt wird. Beide Öffnungen werden auf der Außenseite der Tür mit einer Filtermatte versehen. Die Filtermatte ist aus dem Filtermaterial für Dunstabzugshauben gefertigt. Diese filtert zum einen beim Einlass den Staub aus der Luft und zum anderen verhindert sie das Eindringen von Schädlingen durch die Öffnungen. In Abbildung 3.3 sind die befestigten Filtermatten zu sehen. Der Lüfter wird erst gegen Ende der Arbeiten angeschlossen.

Der Reiseluftbefeuchter wird direkt neben dem Lüfter im Einhängefach befestigt. Die mitgelieferten Flaschenadapter können genutzt werden, um beinahe jede PET-Flasche kompatibel zu machen. Die Flasche sollte erst am Ende der Arbeiten aufgefüllt und eingesetzt werden. Die Anschlussleitung des Luftbefeuchters kann in seiner aktuellen Form nicht genutzt werden. Das Netzteil muss entfernt und aus seinem Gehäuse ausgebaut werden. Der Anschluss erfolgt ebenfalls erst in den finalen Schritten des Aufbaus über das Relaisboard zu seinem modifizierten Netzteil. Die Heizmatte wird auf dem untersten Einschub platziert und ihr Stromstecker abgeschnitten.

Nun werden die Leitungen verlegt. Dazu wird an der Decke des Kühlschranks eine passende Stelle für die Kabeldurchführung markiert. Das *LED*-Board wird provisorisch eingehängt, um die Position der Öffnung für die Leitungen des Boards zu be-

Abbildung 3.3: Befestigte Filtermatten an der Außenseite der *Growfridgetür*

stimmen. Ist die Markierung gesetzt, wird das Board wieder entfernt. Falls noch nicht geschehen, wird der Deckel des Kühlschranks demontiert. Jetzt werden an den Markierungen die notwendigen Öffnungen gebohrt. Dabei sollte die Größe der Öffnung anhand der Stärke der Leitungen bemessen werden, sodass es bei der Durchführung der Leitungen zu keiner Beschädigung der Isolierung kommt. Sind alle notwendigen Öffnungen gebohrt, werden die Leitungen der Lüfter und des Reiseluftbefeuchters an der Kühlschrankwand auf der Innenseite verlegt und mit weißem Gewebeklebeband fixiert, aber noch nicht durch die Öffnungen geführt.

Um ein Überlaufen des *Growfridges* durch Kondenswasser an der Rückwand zu vermeiden, wird ein Aluminiumstreifen über nahezu der gesamten Breite der Rückwand befestigt. Dieser Streifen in der Größe 10 × 50 cm wurde um 2 cm auf 10 × 48 cm gekürzt. Anschließend wurde er mittig entlang seiner langen Seite gebogen und an der Rückwand mit doppelseitigem Klebeband befestigt. Als Schale für die Töpfe wird ein Backblech verwendet. Diese Verformung des Aluminiumstreifens sorgt dafür, dass herunterlaufendes Kondenswasser nicht in die Abtropfwanne des *Growfridges*, sondern in das Backblech mit den Pflanztöpfen geleitet wird. Das Backblech ist äußerst stabil und knick bei hoher Belastung nicht ab. In Abbildung 3.5 ist der befestigte Aluminiumstreifen und das Backblech zu sehen. Mit dieser Konstruktion wird die kondensierte Luftfeuchtigkeit zur Bewässerung genutzt und das System kann mithilfe des Reiseluftbefeuchters indirekt mit Wasser versorgt werden. Damit das Backblech als ebene Fläche im *Growfridge* steht, wurden Unterlegscheiben aus dem

Baumarkt übereinander zusammengeklebt und so zwei Abstandshalter erzeugt, die das Backblech gerade halten.

Anschließend muss die Hartschaumisolierung oberhalb der Decke so weit entfernt werden, sodass die gesamte Peripherie des *Growfridges* dort eingesetzt werden kann. Die Plastikwand des Kühlschranks ist sehr empfindlich und kann bei Unachtsamkeit irreparabel beschädigt werden. In Abbildung 3.4 ist zu sehen, wie die Hartschaumisolierung nach der teilweisen Entfernung aussieht. Es wird genug Raum für den Einbau der *LED*-Treiber, des Raspberry Pis, des Relaisboards und der gesamten Stromversorgung inklusive Kühlung gelassen. Da später der Deckel des Kühlschranks erneut montiert wird und dieser nicht flächendeckend abdichtet, wird nach Fertigstellung des Aufbaus eine passende Plexiglasplatte mit doppelseitigem Schaumklebeband darüber geklebt. Dieser Schritt sollte erst nach Abschluss aller Installationsschritte durchgeführt werden. Somit wird ein Luftkanal erzeugt, der die gesamte Elektronik des *Growfridges* kühlt. Die warme, austretende Luft am Ende des Kanals wird unterhalb des Deckels nach außen abgeführt. In Abbildung 3.4 wurde die Kühlfunktion getestet. Dazu wurden die ehemaligen Glaseinschübe des Kühlschranks zur provisorischen Abdichtung des Luftkanals genutzt, da die Plexiglasscheiben noch nicht eingetroffen waren.

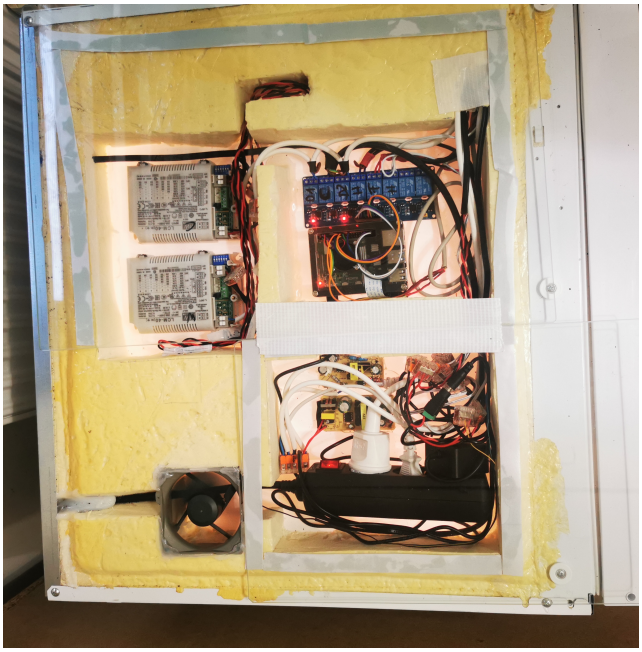


Abbildung 3.4: Test der Kühlung nach eingebauter Schaltung

Im nächsten Schritt wird die Hardware in den freien Raum eingebaut und angeschlossen. Die verlegten Leitungen werden durch die jeweilige Öffnung geführt und mit dem Relaisboard verbunden. Das *LED*-Board wird in den *Growfridge* eingebaut und die Stromleitungen durch den Deckel verlegt. Das Sensormodul wird mit

seiner Datenleitung im *Growfridge* an der Seitenwand befestigt. Die Datenleitung wird durch seine Öffnung in der Decke geleitet. Sind alle Leitungen verlegt, kann das sensible Datenkabel der Kamera verlegt und angeschlossen werden. Dazu wird mit einem scharfen Haushaltsmesser ein möglichst kleiner Streifen aus der dünnen Decke des *Growfridges* entfernt, sodass das Kamera-Datenkabel in den Innenraum geführt werden kann. Der abgetrennte 2 × 10 cm breite Aluminiumstreifen wird bis zur Hälfte bei 5 cm auf die Rückseite des *LED*-Boards geklebt. Die über das *LED*-Board stehenden 5 cm werden als Halterung für die Kamera genutzt. Die Kamera wird am Aluminiumstreifen befestigt. Abschließend wird die Kamera in ihren finalen Blickwinkel gedrückt und der Aluminiumstreifen langsam in diese Position gebogen.

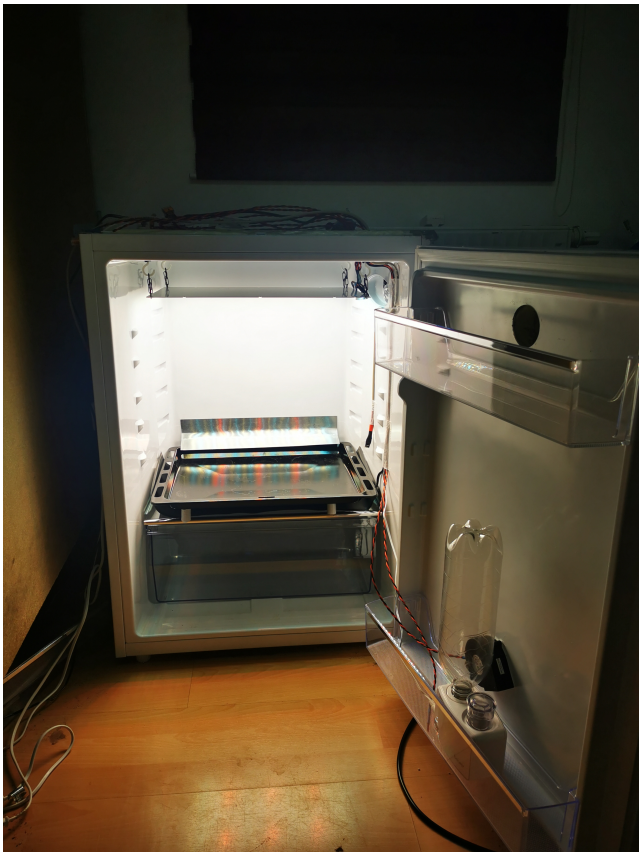


Abbildung 3.5: Erster Funktionstest nach Anschluss der Hardware

Das *LED*-Board, die Lüfter und der Reiseluftbefeuchter werden ebenfalls an ihre jeweiligen Treiber angeschlossen. Der Kühlschrank und die Heizmatte werden mit jeweils einem der Wechselstromleiter vom Relaisboard gesteuert und nach diesem an das *Wechselstromnetz* angeschlossen. Alle Lüfter des *Growfridges* werden parallel an die 12 V Stromversorgung angeschlossen. Dabei wird der Gehäuselüfter und der Umwälz-Lüfter direkt an die Stromquelle angeschlossen. Der Frischluft-Lüfter wird über einen Leiter vom Relaisboard geschaltet und ebenfalls an die 12 V Stromver-

sorgung angeschlossen. Der Reiseluftbefeuchter wird an die 24V Stromquelle angeschlossen und mit einem Leiter über das Relaisboard geschaltet.

Nun wird noch eine 80 mm Öffnung in den Deckel gebohrt, damit der Lüfter zur Peripheriekühlung Luft ansaugen kann. Der Deckel wird auf dem Kühlschrank montiert und der Ansaugkanal des Gehäuselüfters wird mit einer 75 mm Überschiebmuffe für Abflussrohre vor dem erneuten Einsaugen der warmen Luft isoliert. Damit sich im Laufe der Zeit nicht zu viel Staub in der Steuerung des *Growfridges* ansammelt, wird der entstandene Schornstein mit einem Stück Filtermatte abgedeckt.

(a) *Growfridge* von Außen **(b)** *Growfridge* von Innen

Abbildung 3.6: Fertiger *Growfridge* im Betrieb von Außen und Innen

In Abbildung 3.7 kann man den Schaltplan des *Growfridges* sehen. Hilfe beim Anschluss an die richtigen GPIO-Pins kann die Seite https://pinout.xyz/ liefern (Gadgetoid (2022)). Nachdem ist der Einbau der Hardware abgeschlossen und es kann mit der Einrichtung der Benutzerschnittstelle begonnen werden. In Abbildung 3.6 ist der fertige *Growfridge* von Außen und Innen zu sehen.

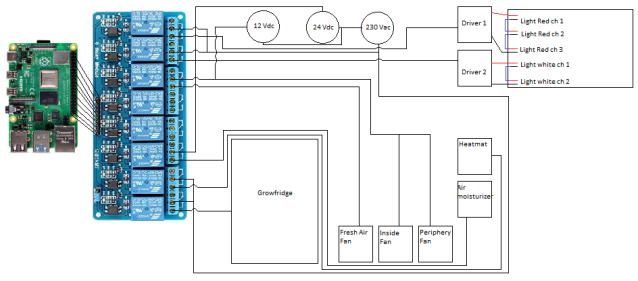


Abbildung 3.7: Schaltplan des *Growfridges*

4 Programmierung des *Growfridges*

Die Programmierung des *Growfridges* kann in vier Schritte unterteilt werden. Zuerst wird ein Grundsystem entwickelt, das in der Lage ist alle Anforderungen zu erfüllen. Anschließend wird ein Prozess entwickelt, der die Initialisierung des *Growfridges* übernimmt. Der dritte Schritt ist der generelle Betrieb des *Growfridges*. Abschließend wird die Benutzerschnittstelle installiert und konfiguriert.

4.1 Das Grundsystem

Damit das Raspberry Pi alle seine Aufgaben erledigen kann, wird Zusatzsoftware benötigt. Diese muss vor ihrer Benutzung installiert und konfiguriert werden. In der Vergangenheit wurde diese direkt auf dem System installiert. Im Laufe der Zeit hat sich das Prinzip der *Containerisierung* immer weiter durchgesetzt. So wird die benötigte Software nicht mehr direkt auf dem System installiert, sondern kann sich über *Docker* und das jeweilige *Image* einfach ausführen lassen.

Mit `git clone git@github.com:BenjaminOchocki/rpi-growfridge.git` wird ein Ordner namens *rpi-growfridge* im *Homeverzeichnis* des Nutzers erstellt. Der folgende Teil bezieht sich auf den Inhalt dieses Ordners. Das bedeutet, für jedes auszuführende Kommando wird davon ausgegangen, dass es innerhalb dieses Ordners ausgeführt wird. Um das Projekt ausführen zu können, wird zusätzliche Software wie *Docker* und *Docker-Compose* benötigt.

Docker bietet zur Konfiguration einer Umgebung mit mehreren *Containern* eine *docker-compose.yml* Datei an. In ihr werden alle benötigten Services definiert. Auf diese Weise wird jeder Service innerhalb eines eigenen *Containers* gestartet und mit anderen *Containern* verknüpft. Je nach Belieben kann so ein neuer Service hinzugefügt oder editiert werden. Dabei bietet *Docker* die Nutzung einer .env Datei an, um sensible Daten aus der Konfiguration der Services herauszuhalten. Manche Service-Images können für ihren benötigten Zweck nicht unverändert genutzt werden. Für diesen Fall hat der jeweilige Service dann ein eigenes Dockerfile. In diesem Dockerfile sind die Änderungen aufgeführt, welche vom Standard abweichen und später benötigt werden. Ist das Image gebaut, wird dieses lokal zwischengespeichert

und zur Ausführung des *Containers* genutzt. Wie die *docker-compose.yml* Datei des *Growfridges* aussieht, ist in Listing A.1 zu sehen.

Als Erstes wird ein Netzwerk mit Namen *grownet* erstellt, in welchem alle Mitglieder eine IP aus dem Subnet Bereich von 172.42.0.0/24 besitzen. Danach werden die jeweiligen Services erstellt. Jeder Service bringt spezifische Einstellungen für seine Nutzung mit. Diese werden durch konfigurierbare Parameter an den Service weitergereicht.

Damit der Benutzer den *Growfridge* über sein lokales Netzwerk steuern kann, wird der Webserver *Nginx* in einem *Container* ausgeführt. Die Konfiguration des Webservers erfolgt durch die Datei *env/nginx/default.conf* in Listing A.2. Diese Datei wird mit der Option *Volume* in den *Container* des Webservers verlinkt. Der Webserver benötigt PHP, um den Code des Frameworks Laravel ausführen zu können.

Der PHP-Service wird ebenfalls über einen eigenen *Container* ausgeführt und mit den Daten des Webservers **verknüpft**. Die Konfiguration des Services findet mit der Datei *env/php-fpm/php.ini* statt, die in Listing A.4 zu sehen ist. Laravel benötigt einige PHP-Erweiterungen, die im Standard-Image von PHP nicht enthalten sind. Deshalb wird das Image mit einem Dockerfile angepasst. In diesem Schritt werden alle benötigten Systempakete installiert. Einige dieser Pakete dienen der leichteren Bedienung, sollte eine Fehlersuche notwendig sein. Dieses Dockerfile sieht wie in Listing A.3 aus.

Damit PHP Zugriff auf die Dateien von Laravel hat, wird der Ordner, in dem sich die Dateien befinden, in das */var/www/html* Verzeichnis des *Nginx Containers* eingehängt. Somit haben sowohl der Webserver, als auch *PHP* Zugriff auf die Daten von Laravel.

Das Framework Laravel wurde wegen der leichten Bedienbarkeit genwählt. Die Entwicklung mit diesem Framework geht leicht von der Hand und bietet eine sehr gute Dokumentation. Die Community ist ebenfalls im starken Wachstum, weshalb der Support im Internet relativ gut ist.

Damit Laravel ausgeführt werden kann, wird eine Datenbank benötigt. Mit dem *MariaDB System* kann dieser Bedarf gedeckt werden. Um *MariaDB* mit allen notwendigen Datenbanken und den dazugehörigen Tabellen zu versorgen, werden bei der Initialisierung alle *SQL* Dateien des Ordners *env/mariadb/initdb* ausgeführt. In den Listings der Dateien *01_create_databases.example.sql* (A.5) und *02_create_tables.example.sql* (A.6) kann ihr Inhalt gesehen werden.

Mit dem Service phpMyAdmin kann mittels des Internetbrowsers der Inhalt der MySQL Datenbank angezeigt werden. Dieser benötigt lediglich valide Zugangsdaten

und ist sofort in der Lage, jede MySQL Datenbank mit entsprechendem Zugriff zu lesen und zu modifizieren.

Da das Sensormodul immer zwei Werte und einen Zeitpunkt dazu liefert, bietet sich eine Zeitreihen-Datenbank an. *InfluxDB* bietet neben der zeitreihenbasierten Datenbank auch weitere nützliche Funktionen. So können beispielsweise eigene Dashboards mit maximal konfigurierbarem Inhalt erzeugt werden. Eine automatische Komprimierung von alten Messwerten wird ebenfalls unterstützt.

Neben den aktuellen Sensorwerten soll ebenfalls der jeweilige Zustand der Peripherie dokumentiert werden. Wird der Zustand eines Peripheriegerätes geändert, so wird diese Veränderung ebenfalls gespeichert. Dadurch ist es später im *InfluxDB* Dashboard möglich die Entscheidungen der Schaltlogik nachzuverfolgen.

Am Raspberry Pi sind ein Sensormodul und das *Relaisboard* angesteckt. Um diese auszulesen und zu steuern, wird ein Python *Docker* Image benötigt. In diesem werden die Treiber der jeweiligen Hardware installiert. Dies geschieht durch ein Dockerfile in *env/python3/Dockerfile*, welches in Listing A.7 zu sehen ist. Da beide Skripte auf der gleichen Python Version laufen, bietet es sich an, ein einziges Image zu bauen. In diesem Image sind die Treiber für beide Skripte installiert. Jedes Skript bekommt seinen eigenen Service, über welchen das Skript in den *Docker Container* gelinkt und ausgeführt wird. Zur Ausführung des Skriptes wird die jeweilige Hardware an den *Container* weitergereicht.

Das Pythonskript *env/python3/sensor/sensor_reader.example.py* für das Sensormodul ist in Listing A.9 einsehbar. In Listing A.10 ist das Skript des Relaisboards (*env/python3/relayboard/relays_switcher.example.py*) zu sehen. Das Skript *env/python3/relayboard/relays_off.py.example.py* wird zur Deaktivierung aller Relais genutzt und ist in Listing A.8 einsehbar.

Um die Benutzung für den späteren Nutzer zu vereinfachen, wurde ein *Makefile* erstellt, um mehrere Tasks zu definieren. Mithilfe dieser Tasks braucht der Benutzer keinerlei Wissen über Befehle oder Skripte zu haben, sondern kann einfach einen vordefinierten Task verwenden. Führt man den Befehl `make help` aus, so erhält man eine Übersicht aller verfügbaren Tasks.

In Listing A.11 wird der Inhalt des Makefiles gezeigt.

4.2 Die Initialisierung

Damit der *Growfridge* bei seiner erstmaligen Einrichtung alles hat, was benötigt wird, gibt es in der Übersicht aller Tasks eine Rubrik namens

`[Install the Growfridge. Run in this order]`, in der alle benötigten Tasks stehen, um den *Growfridge* einzurichten. All diese Tasks wurden in einem extra *Makefile* festgehalten. Der Inhalt der Datei *.make/01_Setup.mk* wird in Listing A.12 gezeigt.

Die folgende Auflistung beinhaltet alle Tasks in ihrer entsprechenden Reihenfolge, welche zur initialen Installation des *Growfridges* benötigt werden. Zu jedem Task wird erklärt, was dieser erledigt.

`setup-login`: startet das Skript *config-copy.sh* aus dem Ordner *install*. Dieses Skript kopiert die .env.example Datei zur .env Datei, in welcher der Nutzer seine jeweiligen Anpassungen wie Passwörter, Benutzernamen und vieles mehr vornehmen kann. Diese sollte der Nutzer nach der Ausführung des Befehls vornehmen, jedoch spätestens vor Ausführung des Befehls `setup-config`. Der Inhalt der .env.example Datei wird in Listing A.13 gezeigt. Der Inhalt des Skripts *install/config-copy.sh* kann in Listing A.14 gesehen werden.

`setup-software`: dieser Task erfordert Root-Rechte! Dieser Task startet das Skript *setup.sh* aus dem Ordner *install*. Das Skript aktiviert alle benötigten Schnittstellen für angeschlossene Hardware und installiert die entsprechende Software, die zur Ausführung des *Growfridges* benötigt wird. Nach Abschluss startet das Raspberry Pi automatisch neu. Der Inhalt von *install/setup.sh* wird in Listing A.15 angezeigt.

`setup-drivers`: dieser Task erfordert Root-Rechte! Dieser Task startet das Skript *arducam-drivers.sh* aus dem Ordner *install* aus. Dieses Skript installiert alle notwendigen Treiber für die angeschlossene 64 Megapixel (kurz MP) Kamera. Nach Abschluss wird das Raspberry Pi neu gestartet. Der Inhalt des Skripts *arducam-drivers.sh* ist in Listing A.16 zu sehen.

`setup-cron`: dieser Task installiert den Kamera-Cronjob, welcher zu jeder viertel Stunde ein Bild aus dem *Growfridge* aufnimmt und das vorherige Bild überschreibt. Das Skript zur Installation des Cronjobs ist in Listing A.17 und das der Kamera *env/python3/camera.sh* in Listing A.18 zu sehen.

`setup-config`: dieser Task startet das Skript *config-init.sh* aus dem Ordner *install*. Dabei werden die Platzhalterwerte aus der .env Datei in die jeweiligen Dateien und Skripte eingefügt. Durch das Setzen dieser Werte wird der Zugriff jedes einzelnen Services geregelt. Der Inhalt des *config-init.sh* Skipts wird in Listing A.19 gezeigt.

`setup-container`: mit diesem Task werden alle *Docker Container* initialisiert, sodass der *Growfridge* mit allen benötigten Services startet. Als Erstes werden gezielt die Datenbank *Container* gestartet, um ihnen genug Zeit zur initialen Einrichtung zu

geben. Anschließend werden die restlichen *Container* initialisiert. Sobald alle *Container* initiliasiert wurden, wird mit der Installation von Laravel begonnen. Dafür startet dieser Task das Skript *install/setup-laravel.sh*. Der Inhalt des Skripts kann in Listing A.20 eingesehen werden. Sobald Laravel vollständig installiert ist, werden alle *Container* heruntergefahren und die erstmalige Einrichtung des *Growfridges* ist abgeschlossen.

Anschließend muss ein Benutzerkonto erstellt werden. Damit dies möglich ist, wird der *Growfridge* mit dem Befehl `make growfridge-start` gestartet. Danach wird die IP-Adresse des *Growfridges* in einem Browser eingegeben. Die angezeigte Willkommensseite bietet oben rechts die Möglichkeit, einen neuen Benutzer zu registrieren. Sobald der Benuzter registriert ist, wird mit dem Befehl `make growfridge-registration-off` die Registrierung deaktiviert. Sollte es zu einem späteren Zeitpunkt einmal notwendig sein, die Registrierung erneut zu aktivieren, so ist dies mit folgendem Befehl `make growfridge-registration-on` möglich. Das Skript, welches dazu aufgerufen wird, ist in Listing A.21 zu sehen.

Damit die Messwerte des *Growfridges* dargestellt werden können, bietet es sich an, ein Dashboard zu erstellen. Dazu wird die IP-Adresse des *Growfridges* mit dem Port 8086 im Browser aufgerufen (beispielsweise 192.168.178.42:8086). Im darauffolgenden Login werden die Daten genutzt, die in der *.env* Datei gesetzt wurden. In Abbildung 4.1 ist der Aufbau eines Beispieldashboards direkt nach der Installation zu sehen.

Abbildung 4.1: Beispieldashboard des *InfluxDB* Systems

4.3 Der Betrieb

Der spätere Betrieb des *Growfridges* wird ebenfalls mithilfe eines *Makefiles* gesteuert. In der Übersicht aller Befehle befinden sich diese Tasks in der Rubrik *[Growfridge]*. Der Inhalt des *Makefiles .make/02_Growfridge.mk* wird in Listing A.22 gezeigt.

Die folgende Auflistung beinhaltet alle Tasks, die zum Betrieb des *Growfridges* nötig sind. Die Aufgabe jedes Tasks wird erklärt.

`growfridge-start`: dieser Task startet alle *Container* des *Growfridges* in der benötigten Reihenfolge.

`growfridge-stop`: dieser Task stoppt alle *Container* des *Growfridges*. Um sicherzustellen, dass alle Relais ausgeschaltet wurden, wird ein extra *Container* gestartet, der alle Relais deaktiviert. Dieser wird nach Abschluss ebenfalls gestoppt.

`growfridge-restart`: dieser Task ruft lediglich `make growfridge-stop` und `make growfridge-start` nacheinander auf

`growfridge-registration-off`: dieser Task ruft das Skript *install/laravel-toggle-registration.sh* auf, um die Registrierfunktion zu deaktivieren. Ist die Registrierung aktiv, kann sich jeder Besucher ein neues Konto erstellen und die Einstellungen des *Growfridges* bearbeiten. Aus diesem Grund sollte die Registrierung nach der Erstellung des Benutzerkontos deaktiviert werden.

`growfridge-registration-on`: dieser Task ruft das Skript *install/laravel-toggle-registration.sh* auf, um die Registrierfunktion erneut zu aktivieren.

`generate-passwords`: dieser Task kann vom Benutzer dazu verwendet werden, neue Passwörter für den *Growfridge* zu generieren. Dabei wird das Skript *install/passwords.sh* aufgerufen, dessen Inhalt in Listing A.23 angezeigt wird:

4.4 Die Benutzerschnittstelle

Da dieses Projekt bei Abschluss bereits mit einer fertigen Benutzerschnittstelle zur Verfügung gestellt wird, dient der folgende Teil nur zum Verständnis des Aufbaus und muss nicht erneut durchgeführt werden. Es soll dabei die Vorgehensweise bei der Erstellung der Benutzerschnittstelle gezeigt werden.

Damit die Benutzerschnittstelle erstellt werden kann, wird ein laufender *Growfridge* vorausgesetzt. Mit dem Befehl `docker exec -it php-fpm bash` verbindet sich die Shell mit dem *php-fpm Container*. Die *bash* des *Containers* wird im Verzeichnis */var/www/html/* aufgerufen. Als PHP Framework wird Laravel genutzt. Laravel

wird über Composer installiert, indem ein neues Projekt erstellt wird. Dabei lädt Composer alle Dateien herunter, die zur Installation benötigt werden. Das Verzeichnis */var/www/html/* wird als Speicherort der Laravel-Dateien genutzt. Der Inhalt dieses Verzeichnisses ist im Service von *php-fpm* mit dem *src/* Verzeichnis per Volume-Eintrag verlinkt.

Mit dem Befehl `composer create-project laravel/laravel ./` wird ein neues Projekt im aktuellen Verzeichnis erstellt. Um die korrekten Berechtigungen auf alle Dateien zu setzen, werden auf dem Raspberry Pi, aber nicht im *Container*, nachfolgende Befehle ausgeführt:

```
sudo chown -R www-data:www-data ./src
sudo find ./src -type f -exec chmod 644
sudo find ./src -type d -exec chmod 755
```

Diese Reihenfolge an Befehlen ändert dabei den Besitzer und die Gruppe aller Dateien auf *www-data* und setzt die entsprechenden Berechtigungen für Verzeichnisse und Dateien.

Sind alle Dateien heruntergeladen, wird mit folgendem Befehl *Laravel Breeze* installiert: `composer require laravel/breeze`. *Laravel Breeze* ist eine einfache und minimale Implementierung einer Authentifizierung mit eigener Anmeldung, Registrierung und E-Mail-Verifikation. Mit dem Befehl `php artisan breeze:install` wird *Laravel Breeze* installiert. Danach muss die .env Datei von Laravel angepasst werden. Sind die neuen Einstellungen gesetzt, werden die Befehle `php artisan migrate` und `nvm install 14 && npm install && npm run build` ausgeführt. Anschließend hat die Standard-Willkommens-Seite von Laravel bereits einen Login und Registrationsbereich. Damit Laravel mit der Datenbank von *InfluxDB* kommunizieren kann, werden innerhalb des *php-fpm Containers* mit folgendem Befehl die benötigten Dateien per Composer heruntergeladen:

`composer require influxdata/influxdb-client-php`.

Der Aufbau der Schnittstelle wird dabei in 4 einzelne Schritte unterteilt. Die Willkommensseite, die für jeden Besucher des Webservers sichtbar ist und die hinter einem Login verborgenen Seiten des Dashboards, der Konditionen und des Ablaufplans. Die Willkommensseite wird durch ein Bild des Innenraums und einer Kurzinformation über Temperatur und Luftfeuchtigkeit dargestellt. Des Weiteren ist ein Loginbereich oben rechts verlinkt. Wie die Willkommensseite aussieht, kann in Abbildung 4.2 gesehen werden. Der notwendige *Controller* ist in Listing A.25 zu sehen. In der Datei *routes/web.php* wird der Route auf das Wurzelverzeichnis / eine Funktion des *GrowfridgeControllers* zugeordnet. Dies ist in Listing A.24 einsehbar.

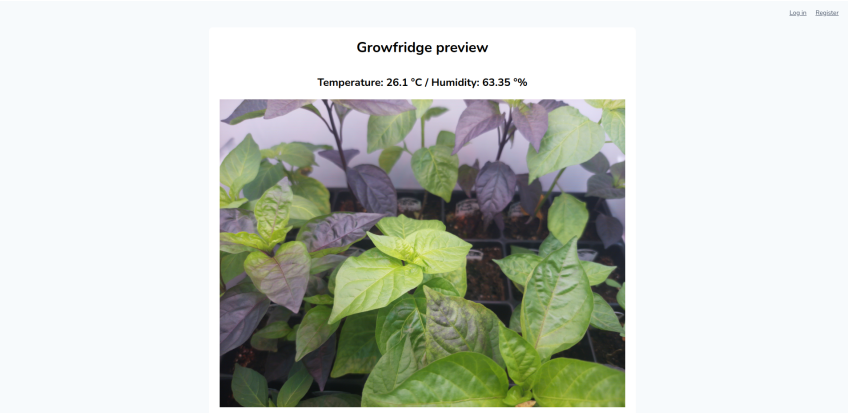


Abbildung 4.2: Öffentliche Willkommensseite des *Growfridges*

Der Inhalt des Dashboards nach dem Nutzer-Login wurde nach der Installation von *Laravel Breeze* angepasst, damit dieser ebenfalls ein Bild des Innenraums und einige Zusatzinformationen anzeigt. Dabei werden in dieser Ansicht zusätzlich Informationen zur aktuellen Kondition ausgegeben. Abbildung 4.3 zeigt dabei das Dashboard nach dem Login durch einen Benutzer. Der Inhalt der Datei *src/resources/views/dashboard.blade.php* ist in Listing A.26 zu sehen und die Änderungen der Route */dashboard* sind in Listing A.24 einsehbar.

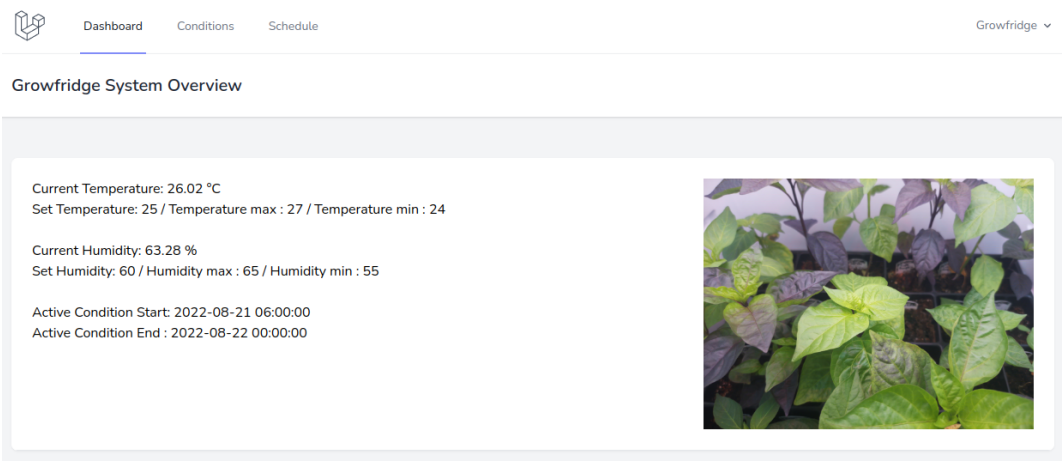
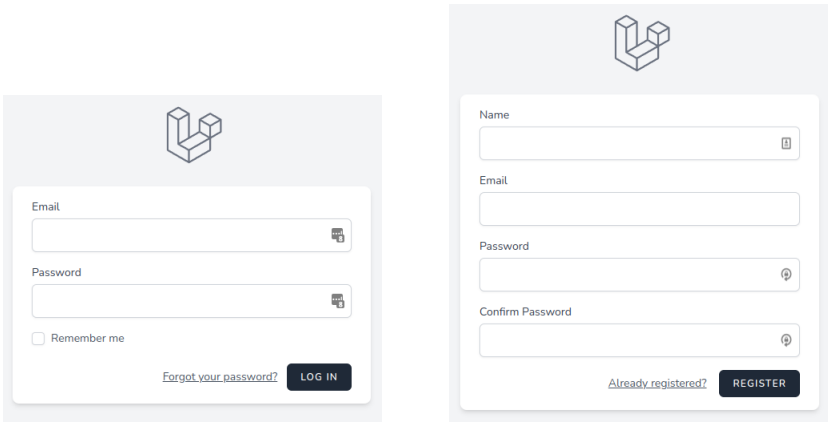


Abbildung 4.3: Dashboard des *Growfridges* nach dem Login

Die Seiten, um sich als Nutzer einzuloggen oder zu registrieren, wurden nach der Installation von *Laravel Breeze* nicht angepasst. Die Standardseiten von *Laravel Breeze* zur Anmeldung und Registration von Nutzern sehen wie in Abbildung 4.4 aus:

Wie in Abbildung 4.3 zu sehen ist, hat der eingeloggte Nutzer die Möglichkeit, in der Navigation des Dashboards, die Seiten *Conditions* und *Schedule* zu besuchen. Die Seite *Conditions* listet alle eingetragenen Konditionen auf. Des Weiteren kann der



(a) Login Seite **(b)** Registrierung

Abbildung 4.4: Login und Registrierungsseiten von *Laravel Breeze*

Nutzer weitere Konditionen erstellen oder bestehende Konditionen bearbeiten oder löschen. Damit diese Funktionen zur Verfügung stehen, muss ein *Controller* dafür existieren. Das Anzeigen, Erstellen und Bearbeiten der eingetragenen Konditionen findet jeweils auf einer eigenen Seite statt. Die Aktionen des Löschens geschieht durch das Mülltonnen Icon in der letzten Spalte des Eintrags bei seiner Auflistung. Der Inhalt der Datei *src/app/Http/Controllers/ConditionController.php* ist in Listing A.27 zu sehen. In den Listings der Dateien *src/resources/views/conditions/create.blade.php* (A.28), *src/resources/views/conditions/edit.blade.php* (A.29) und *src/resources/views/conditions/index.blade.php* (A.30) ist ihr Inhalt einsehbar. In Abbildung 4.5 ist die Übersicht aller Konditionen zu sehen.

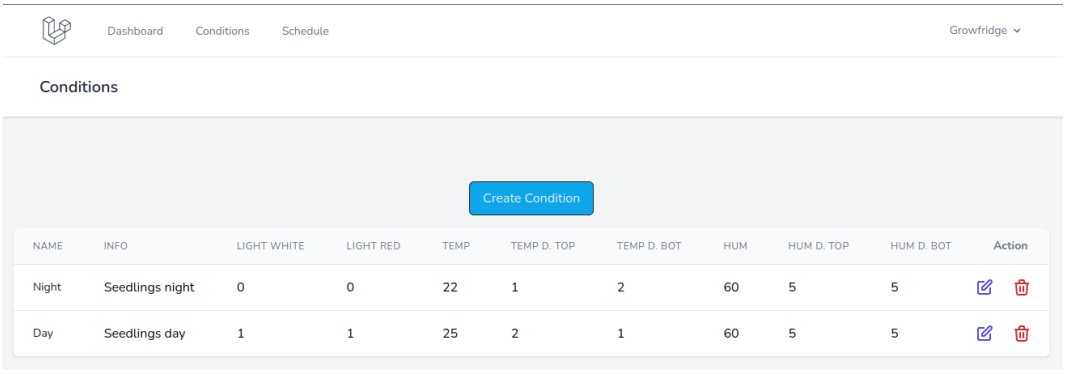


Abbildung 4.5: Übersicht aller gespeicherter Konditionen des *Growfridges*

Da sich die Felder zur Bearbeitung einer Kondition nicht ändern, ist die Seite zur Bearbeitung eines Eintrages identisch mit der Seite der Erstellung. Die Abbildung 4.6 zeigt die Seite zur Erstellung und Bearbeitung einer Kondition.

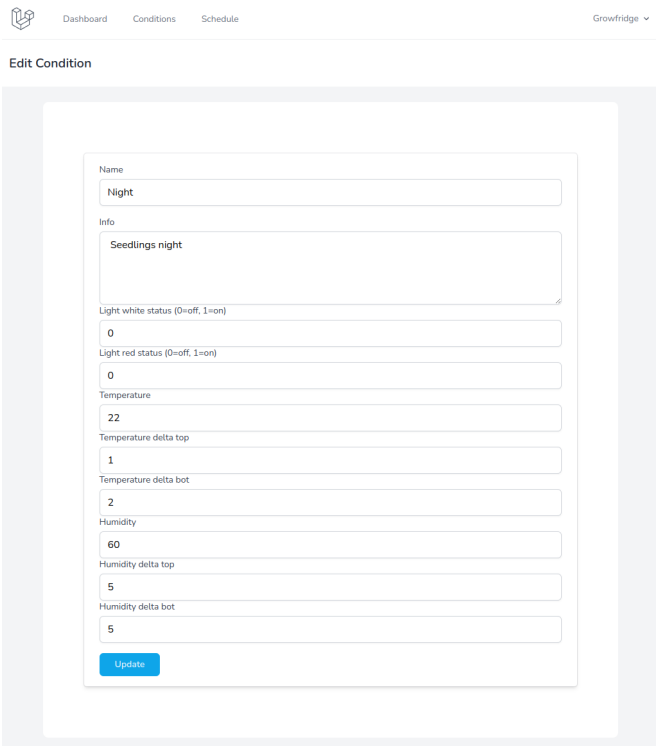


Abbildung 4.6: Seite zur Erstellung und Bearbeitung einer Kondition

Die Seite *Schedule* bietet eine Übersicht aller eingetragenen Konditionen im Ablaufplan des *Growfridges*. Alle Einträge werden täglich wiederholt. Dabei wird nach Ablauf einer Kondition das Startdatum und Enddatum um einen Tag erhöht. Sollten einmal keine Einträge vorhanden sein, wird die Peripherie des *Growfridges* ausgeschaltet. Zur Darstellung aller Einträge ist ein *Controller* notwendig. Die Darstellung, Erzeugung und Bearbeitung eines Eintrages findet jeweils auf einer eigenen Seite statt. Das Löschen eines Eintrages kann wieder durch das Mülltonnen Icon durchgeführt werden. Der Inhalt der Datei *src/app/Http/Controllers/ScheduleController.php* ist in Listing A.31 zu sehen. Die Listings der Dateien *src/resources/views/schedule/create.blade.php* (A.32), *src/resources/views/schedule/edit.blade.php* (A.33) und *src/resources/views/schedule/index.blade.php* (A.34) zeigen ihren jeweiligen Quellcode.

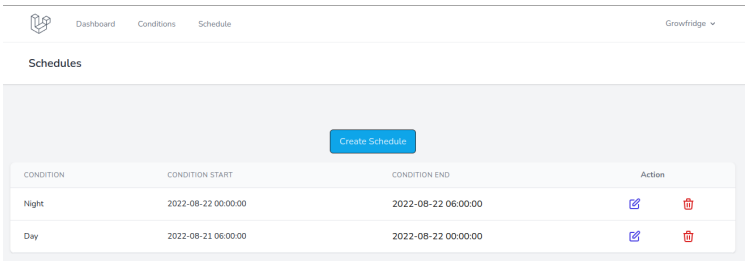


Abbildung 4.7: Ablaufplan aller Konditionen eines Tages

Dabei zeigt Abbildung 4.7 den Ablaufplan des *Growfridges*. In Abbildung 4.8 ist die Seite zur Erstellung und Bearbeitung eines Eintrages zu sehen. Sind alle Funktionen erfolgreich getestet, ist die Benutzerschnittstelle fertig und kann vom Endnutzer zur Einstellung des *Growfridges* genutzt werden.

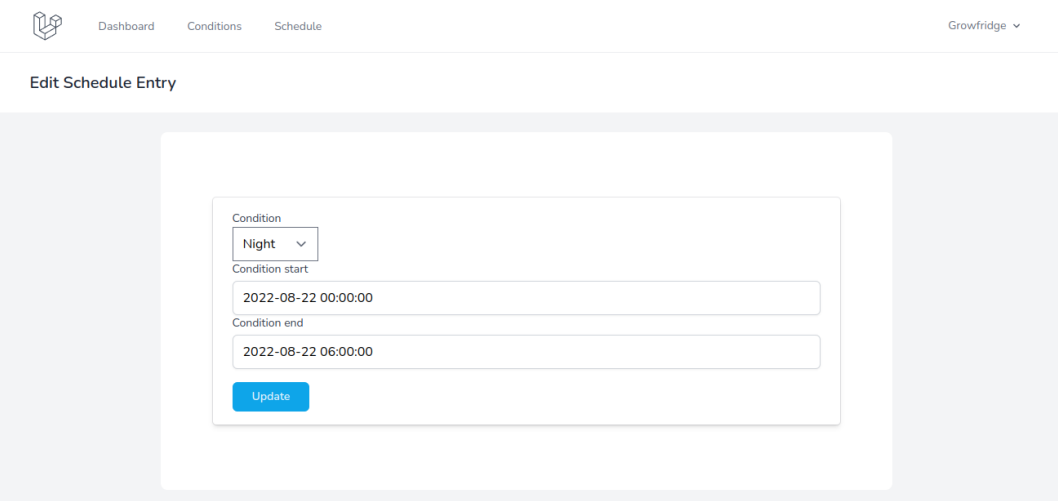


Abbildung 4.8: Seite zur Erstellung und Bearbeitung eines Eintrags im Ablaufplan

5 Fazit und Ausblick

5.1 Fazit

Sieht man sich den Werdegang dieses Projekts an, so fällt auf, dass es viele verschiedene Herausforderungen gab. Um diese zu bewältigen durfte ich mich in viele spannende, mir unbekannte Themen einarbeiten. Dadurch habe ich sehr viel gelernt.

Als unerfahrener Gärtner wurde als Erstes ein Grundwissen im Bereich der Pflanzenanzucht angeeignet. Dabei war jeder Teilbereich vom Samen bis hin zum Licht ein wichtiger Baustein. Die unterschiedlichen Arten von Samen legten einige Rahmenbedingungen fest, die später von der Klimakammer umsetzbar sein müssen. Je nach Pflanze und ihrem aktuellen Stadium, gilt es unterschiedliche Temperaturen und relative Luftfeuchtigkeit aufrechtzuerhalten. Des Weiteren mussten die unterschiedlichen Anzeichen von gestressten Pflanzen jeglicher Ursache kennengelernt werden. Mit diesen Kenntnissen der Biologie galt es nun herauszufinden, wie diese Bedingungen innerhalb einer Kammer hergestellt und gehalten werden können.

Damit unterschiedlichste Bedingungen erreicht und eingehalten werden können, müssen die Werte von Temperatur und relativer Luftfeuchtigkeit beeinflussbar sein. Die Hardware muss entsprechend dem Luftvolumen dimensioniert werden. Die Wahl der Hardware muss aufeinander abgestimmt werden. Da bis zu diesem Zeitpunkt keinerlei Kenntnisse über die Leistung der einzelnen Geräte in diesem speziellen Versuchsaufbau bestand, musste dies durch zeitintensive Tests erlernt werden. Hatte ein Bauteil zu viel oder zu wenig Leistung, so wurde dies entsprechend angepasst. Bei diesen Tests zeigte sich auch, dass weitere Kenntnisse im Bereich der Physik notwendig waren. Dabei fiel auf, dass die relative Luftfeuchtigkeit und die aktuelle Temperatur zusammenhängen. Da die Temperatur und relative Luftfeuchtigkeit vom System reguliert werden, müssen diese Zusammenhänge verstanden werden.

Die Auswahl des Sensors und des Relaisboards war eine weitere Herausforderung. Das Angebot verschiedener Sensoren ist groß. Es gibt Sensoren und Relaisboards in unterschiedlichen Qualitäts- und Preisklassen. Dabei musste nicht qualifizierte Hardware durch einige Tests herausgefiltert werden. Dies kostete vor allem Geduld und Zeit, resultierte jedoch in der angemessenen Auswahl der benötigten Hardware.

Damit das gesamte System von einer Zentrale bedient werden kann, musste ein Konzept der digitalen Struktur entwickelt werden. Dabei lag der Fokus vor allem auf der einfachen Bedienbarkeit durch den Endbenutzer. Die Entwicklung dieses Systems bestand aus mehreren Herausforderungen. Dabei wurde das System immer wieder verbessert und nach neuesten *State of the Art* Prinzipen angepasst. Um dies zu bewerkstelligen, wurden neue Technologien erlernt und angewendet.

Der Zusammenbau des *Growfridges* brachte einige lehrreiche Erfahrungen mit sich. Das fehlende Wissen über einige Bauteile resultierte einige Male in ihrer Zerstörung. Diese mussten erneut bestellt und eingebaut werden, jedoch verbesserte sich dadurch das allgemeine Verständnis über diese Bauteile.

Dieses Projekt hat neben wertvollen Erfahrungen und neuem Wissen auch dafür gesorgt, dass vor meiner Haustür ein kleiner Heimgarten entstanden ist. In diesem verbringe ich gerne Zeit und pflanze einige verschiedene Pflanzen an. In Abbildung 5.1 ist ein aktuelles Bild meines Gartens zu sehen.

Abbildung 5.1: Bild meines voll bepflanzten Grünstreifens vor der Haustür

5.2 Ausblick

Da ich dieses Projekt in meiner Freizeit begonnen habe, wird es nach Abgabe der Bachelorarbeit weiterentwickelt. Dabei werden alle weiteren Updates innerhalb des gleichen Git-Repositorys gespeichert.

In meinem zweiten Prototyp hatte ich bereits mehrere verschiedene, einzeln steuerbare *Wellenlängen*. Damit war es möglich, unterschiedlichste Spektren zu erzeugen. Wie ich bereits im Bereich Licht erwähnt habe, kann am Morgen und am Abend die Rotfärbung des Himmels wegen der Rayleigh-Streuung beobachtet werden. Diesen Effekt möchte ich in Zukunft im *Growfridge* simulieren. Dazu sollen die jeweiligen Kanäle des Boards langsam gedimmt werden und die Änderung des Spektrums fließend geschehen.

Eine weitere Besonderheit des vorherigen Prototyps war seine Bandbreite an ultravioletter Strahlungsquellen. Es gab UV-Strahlung der *Wellenlängen* von 400 nm bis 280 nm. Wie bereits erwähnt, hatte ich in meinen Tests besonders gute Ergebnisse mit UV-Strahlung von 400 nm bis 365 nm, um Pflanzen auf das reale Sonnenlicht vorzubereiten. Diesen Bereich werde ich wieder zum Spektrum des *Growfridges* hinzufügen.

Die Idee, ein Tag-Nacht-Zeitraffer zu erstellen, finde ich sehr reizvoll. Leider habe ich bisher keine geeignete Kamera gefunden, mit der dies inklusive Autofokus und akzeptabler Qualität möglich wäre. Die Zeitraffer-Funktion soll in Zukunft hinzugefügt werden. Dies könnte in Kombination mit dem aktuell ungenutzten Türsensor des *Growfridges* geschehen. Dadurch wäre ein Zeitraffer von einem Öffnen der Tür zum nächsten erstellbar.

Da im aktuellen Aufbau dieser Bachelorarbeit noch zwei Kanäle auf dem Relaisboard zur Verfügung stehen, können weitere Funktionen hinzugefügt werden. Im vorherigen Prototyp kam die Idee auf, einen Wasseranschluss an den *Growfridge* anzubringen, um eine dauerhafte Versorgung mit Wasser zu garantieren. Dieser könnte in Kombination mit einem Sprinkler genutzt werden, um Regen zu simulieren. Bedauerlicherweise ist das *LED*-Board bisher nicht vor Spritzwasser geschützt und auch die Verträglichkeit der anderen Elektronik, vor allem der Sensoren, ist eine Herausforderung.

Dadurch, dass das System des *Growfridges* allein durch ein Sensormodul und ein Relaisboard genutzt werden kann, bietet sich die Möglichkeit, dieses System deutlich größer zu skalieren. Dazu muss lediglich die Hardware zur Größe des Raumes neu dimensioniert werden und der Nutzung des *Growfridges* steht nichts im Wege.

Literaturverzeichnis

[AgroMax 2015] AgroMax: *F54T5HO PURE UV*. https://www.growagromax.com/products/t5-grow-lamps/t5-pure-series/816-2/. Version: 2015. – [Online; accessed 24 July, 2022] 2.4.7

[Borthwick u. a. 1954] Borthwick, H. A. ; Hendricks, S. B. ; Toole, E. H. ; Toole, V. K.: Action of Light on Lettuce-Seed Germination. In: *Botanical Gazette* 115 (1954), Nr. 3, 205-225. http://dx.doi.org/10.1086/335817. – DOI 10.1086/335817 2.4.6

[Ciolkosz 2008] Ciolkosz, Daniel: Design daylight availability for greenhouses using supplementary lighting. In: *Biosystems Engineering* 100 (2008), Nr. 4, 571-580. http://dx.doi.org/https://doi.org/10.1016/j.biosystemseng.2008.04.010. – DOI https://doi.org/10.1016/j.biosystemseng.2008.04.010. – ISSN 1537–5110 2.4.9

[Dahlgren 1923] Dahlgren, K. V. O.: GERANIUM BOHEMICUM L.XG. BOHEMICUM DEPREHENSUM ERIK ALMQ., EIN GRüNWEISS-MARMORIERTER BASTARD. In: *Hereditas* 4 (1923), Nr. 1-2, 239-250. http://dx.doi.org/https://doi.org/10.1111/j.1601-5223.1923.tb02962.x. – DOI https://doi.org/10.1111/j.1601–5223.1923.tb02962.x 2.1

[DIN-Normenausschuss Lichttechnik 2018] DIN-Normenausschuss Lichttechnik: Strahlungsphysik im optischen Bereich und Lichttechnik - Teil 10: Photobiologisch wirksame Strahlung, Größen, Kurzzeichen und Wirkungsspektren / DIN-Normenausschuss Lichttechnik. Version: 2018. https://www.beuth.de/de/norm/din-5031-10/276889937. 2018 (DIN 5031-10:2018-03). – Standard. – [Online; accessed 24 July, 2022] (document), 2.1

[Elektronik Kompendium 2020a] Elektronik Kompendium: *Parallelschaltung von Widerständen*. https://www.elektronik-kompendium.de/sites/slt/0110192.htm. Version: 2020. – [Online; accessed 11 August, 2022] 2.12

[Elektronik Kompendium 2020b] Elektronik Kompendium: *Reihenschaltung von Widerständen*. https://www.elektronik-kompendium.de/sites/slt/0110191.htm. Version: 2020. – [Online; accessed 11 August, 2022] 2.13

[Gadgetoid 2022] Gadgetoid: *Pinout.xyz is the successor to the popular Pi pinout website originally hosted on http://pi.gadgetoid.com/pinout.* https://pinout.xyz/. Version: 2022. – [Online; accessed 11 August, 2022] 3.3

[Gonzalez u. Ross 1980] Gonzalez ; Ross: Performance measurement reference conditions for terrestrial photovoltaics / Jet Propulsion Lab., Pasadena, CA. Version: 1980. https://www.osti.gov/biblio/6427954. 1980 (CONF-800604-P3). – Standard. – [Online; accessed 1 August, 2022] 2.4.2

[Growmart 2022] Growmart: *Apogee MQ-500: Full-Spectrum Quantum Meter 699 Euro.* https://www.growmart.eu/MQ-500-Full-Spectrum-Quantum-Meter.html. Version: 2022. – [Online; accessed 24 July, 2022] 2.4.10

[Gueymard 2001] Gueymard, Christian A.: Parameterized transmittance model for direct beam and circumsolar spectral irradiance. In: *Solar Energy* 71 (2001), Nr. 5, 325-346. http://dx.doi.org/https://doi.org/10.1016/S0038-092X(01)00054-8. – DOI https://doi.org/10.1016/S0038–092X(01)00054–8. – ISSN 0038–092X 2.4.2

[Horst Frank / Phrood / Anony 2008] Horst Frank / Phrood / Anony: *Electromagnetic Wave Spectrum. Colorimetrically more correct version of Image:Electromagnetic spectrum.svg, based on Image:Spectrum-sRGB-low.svg.* https://commons.wikimedia.org/wiki/File:Electromagnetic_spectrum_-de_c.svg. Version: 2008. – [Online; accessed 1 June, 2022] 2.1

[Inductiveload 2007] Inductiveload: *Diagram showing a section through the centre of a cone (1) subtending a solid angle of 1 steradian in a sphere of radius r, along with the spherical "cap"(2). The fact that the external surface of the cap has an area of r2 is shown. Note that this applies only if the solid angle of the cone is exactly 1 steradian.* https://commons.wikimedia.org/wiki/File:Steradian_cone_and_cap.svg. Version: 2007. – [Online; accessed 14 August, 2022] 2.11

[Kitajima u. Butler 1975] Kitajima, M. ; Butler, W.L.: Excitation spectra for Photosystem I and Photosystem II in chloroplasts and the spectral characteristics of the distribution of quanta between the two photosystems. In: *Biochimica et Biophysica Acta (BBA) - Bioenergetics* 408 (1975), Nr. 3, 297-305. http://dx.doi.org/https://doi.org/10.1016/0005-2728(75)90131-0. – DOI https://doi.org/10.1016/0005–2728(75)90131–0. – ISSN 0005–2728 2.4.6

[Laboratory for Atmospheric and Space Physics 2022] Laboratory for Atmospheric and Space Physics: *TSIS Solar Spectral Irradiance - Daily Average, Spectrum.* https://lasp.colorado.edu/lisird/latis/dap/tsis_ssi_24hr.csv?&wavelength>=200&wavelength<=2400&time~2022-06-06T22:00:00.000Z. Version: 2022. – [Online; accessed 6 June, 2022] 2.4.2

[LEDTonic 2020] LEDTonic: *LED Grow Lamps & Light Terms: PPFD (PAR) & Light Footprint Maps*. https://www.ledtonic.com/blogs/guides/ppfd. Version: 2020. – [Online; accessed 11 June, 2022] 2.9

[Leuchter u. a. 2021] Leuchter, Jan ; Hon, Lukas ; Bloudicek, Radim ; Balaz, Teodor ; Blasch, Erik: The Study of Aviation Safe Incapacitating Device Based on LED Technology with a Smart-Illumination Sensor Unit. In: *Sensors* 21 (2021), Nr. 1. http://dx.doi.org/10.3390/s21010081. – DOI 10.3390/s21010081. – ISSN 1424–8220 2.2

[LMRoberts 2010] LMRoberts: *Spectrum of a typical High Pressure Sodium (HPS) lamp*. https://commons.wikimedia.org/w/index.php?curid=16990016. Version: 2010. – [Online; accessed 5 June, 2022] 2.5

[McCree 1971] McCree, K.J.: The action spectrum, absorptance and quantum yield of photosynthesis in crop plants. In: *Agricultural Meteorology* 9 (1971), 191-216. http://dx.doi.org/https://doi.org/10.1016/0002-1571(71)90022-7. – DOI https://doi.org/10.1016/0002–1571(71)90022–7. – ISSN 0002–1571 2.4.4

[NAS 1957] NAS: National Academy of Sciences: Abstracts of Papers To Be Presented at the Annual Meeting, 22-24 April 1957, Washington, D.C. In: *Science* 125 (1957), Nr. 3251, 746-752. http://dx.doi.org/10.1126/science.125.3251.746. – DOI 10.1126/science.125.3251.746 2.4.6

[Philips Lighting 2022] Philips Lighting: *MASTER TL5 HO Xtra 54W/840 1SL/20 Lichtfarbe/840*. https://www.lighting.philips.de/prof/konventionelle-lampen-und-leuchtstofflampen/leuchtstofflampen-und-starter/tl5/master-tl5-ho-xtra. Version: 2022. – [Online; accessed 11 June, 2022] 2.6

[Phplips Lighting 2015] Phplips Lighting: *PL-L 36W/09/4P*. https://www.lighting.philips.com/main/prof/conventional-lamps-and-tubes/special-lamps/various-uv-applications/uv-apuva/uv-a-1-pl-l. Version: 2015. – [Online; accessed 24 July, 2022] 2.4.7

[RaspberryPi Ltd 2022a] RaspberryPi Ltd: *Configuration*. https://www.raspberrypi.com/documentation/computers/configuration.html. Version: 2022. – [Online; accessed 11 August, 2022] 3.2

[RaspberryPi Ltd 2022b] RaspberryPi Ltd: *Getting Started*. https://www.raspberrypi.com/documentation/computers/getting-started.html. Version: 2022. – [Online; accessed 11 August, 2022] 3.2

[Sage u. Kubien 2007] Sage, Rowan F. ; Kubien, David S.: The temperature response of C3 and C4 photosynthesis. In: *Plant, Cell & Environment* 30 (2007), Nr.

9, 1086-1106. http://dx.doi.org/https://doi.org/10.1111/j.1365-3040.2007.01682.x. – DOI https://doi.org/10.1111/j.1365–3040.2007.01682.x 2.2

[Samsung 2020] Samsung: *LM301H EVO CCT: 3000K*. https://www.samsung.com/led/lighting/mid-power-leds/3030-leds/lm301h-evo/. Version: 2020. – [Online; accessed 11 June, 2022] 2.7

[Samsung 2022] Samsung: *Samsung Introduces "Plant-Centric Spectrum LEDs, LM301H EVO" for Most Effective Indoor Farming*. https://www.samsung.com/led/about-us/news-events/news/news-detail-65/. Version: 2022. – [Online; accessed 11 June, 2022] 2.4.5

[Sharpe u. a. 2005] Sharpe, Lindsay T. ; Stockman, Andrew ; Jagla, Wolfgang ; Jägle, Herbert: A luminous efficiency function, V*(λ), for daylight adaptation. In: *Journal of Vision* 5 (2005), 12, Nr. 11, 3-3. http://dx.doi.org/10.1167/5.11.3. – DOI 10.1167/5.11.3. – ISSN 1534–7362 2.4.1

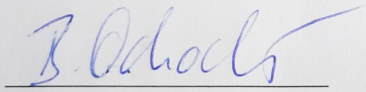
[Varella u. a. 2011] Varella, A.C. ; Moot, D. ; Pollock, Keith ; Peri, Pablo ; Lucas, R.: Do light and alfalfa responses to cloth and slatted shade represent those measured under an agroforestry system? Agrofor Syst. In: *Agroforestry Systems* 81 (2011), 02, S. 157–173. http://dx.doi.org/10.1007/s10457-010-9319-6. – DOI 10.1007/s10457–010–9319–6 2.4.8, 2.8

Ich, Benjamin Ochocki, Matrikel-Nr. 2048974, versichere hiermit, dass ich die vorliegende Arbeit mit dem Thema

> *Bau einer automatisierten Klimakammer zur Anzucht von Pflanzen mit dem Raspberry Pi*

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Augsburg, den 22. August 2022



Benjamin Ochocki

A Anhang

Um die Lesbarkeit der Liste aller Bauteile und der Listings zu verbessern, ist die Größe der seitlichen Ränder auf ein Minimum reduziert.

A.1 Liste aller Bauelemente

Tabelle A.1: Liste aller benötigten Bauelemente

Name	**Bauteilinformation**
Amica VK S15917 W Standkühlschrank	60 cm breit, 156 L, Platte abnehmbar
LERWAY Heizmatte	52 × 25 cm, 21 Watt
Beurer LB 12	Luftbefeuchter, 24 Volt
Titan TFD-B9733M12C	Frischluft-Lüfter, 12 Volt
Noctua NF-R8 redux-1800	Gehäuselüfter, 80 mm, 12 Volt
Noctua NF-F12 iPPC-2000	Umwälz-Lüfter, 120 mm, 12 Volt
EFISH DC 12V 2A 24W Netzteil	Netzteil für 12 Volt Endgeräte
LED-Tech 326 XT TW3	Fünfkanal LED-Board
2 × Netzteil MeanWell LCM-40	Treiber für die einzelnen LED-Board Kanäle
Vier Hakenschrauben	Befestigung des LED-Boards
Mini-Karabiner	Befestigung des LED-Boards an den Hakenschrauben
Backblech 46,5 × 37,5 cm	Stabile Wanne für die Töpfe der Pflanzen
Mehrere Unterlagscheiben	Werden genutzt um das Backblech gerade auszurichten
Aluplatte 10 × 50 cm	Umleitung des Kondenswassers in das Backblech
Montageklebeband Doppelseitig 2 mm dick	Montage des Frischluft-Lüfters und der Plexiglasscheiben
Doppelseitiges Klebeband Extra Stark Transparent 2,4 cm breit	Montage der Aluplatte, Treiber, Mehrfachsteckdose, Raspberry Pi und des Relaisboards
Panzertape weiss	Befestigung aller Leitungen innerhalb des Kühlschranks
Verschiedene Verbindungskabel	Ein, zwei und drei adrige Verbindungskabel
Arducam 64 MP Kamera	Innenaufnahmen der *Growfridges*
Extra langes 50 cm Kamerakabel	Verbindung der Kamera
Abzugshaubenwatte 1 Packung	Filterung
Plexiglasscheiben 0,5 mm dick	Abdichtung der Steuerung sodass ein Luftkanal zur Kühlung entsteht
Dreifachsteckdose	Stromversorung
Schrumpfschlauch verschiedener Größen	Isolierung von Leitern und Lötstellen
Silikon	Zur Abdichtung von Löchern
Raspberry Pi 4	Herz der Steuerung
SD Karte 64 GB	Festplatte des Raspberry Pis
Gehäuse Raspberry Pi	Gehäuse des Raspberry Pis
Netzteil Raspberry Pi	Stromversorung
Sensormodul mit einem abgeschirmten Kabel	Vier-adrig 1 meter
GPIO Kabel	Verbindung des Sensormoduls und des Relaisboards
Acht Kanal Relaisboard	Schaltung der Peripherie
Waago Klemmen	Lötfreie Verbindung von Stromleitern
Übersteckmuffe 75 mm Abflussrohr	Frischluft Öffnung des Gehäuselüfters

A.2 Listing aller Quellcode Dateien

Listing A.1: Inhalt: docker-compose.yml

```
version: '3.9'

networks:
  grownet:
    ipam:
      config:
        - subnet: "172.42.0.0/24"

services:
  php-fpm:
    build:
      context: "./env/php-fpm/"
      dockerfile: "Dockerfile"
      args:
        PHP_VERSION: ${PHP_VERSION?}
    container_name: "php-fpm"
    restart: "always"
    networks:
      grownet:
        ipv4_address: 172.42.0.2
    volumes:
      - "${APP_DIR}:/var/www/html"
      - "${PHP_CONFIG_LOCATION}:/usr/local/etc/php/conf.d/php.ini"

  nginx:
    image: nginx:${NGINX_VERSION}
    container_name: "nginx"
    restart: "always"
    networks:
      grownet:
        ipv4_address: 172.42.0.3
    depends_on:
      - "php-fpm"
    links:
      - "php-fpm"
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - "${APP_DIR}:/var/www/html"
      - "${NGINX_CONFIG_LOCATION}:/etc/nginx/conf.d/default.conf"

  mariadb:
    image: mariadb:${MARIADB_VERSION}
    container_name: "mariadb"
    restart: "always"
    networks:
      grownet:
        ipv4_address: 172.42.0.4
    command: "--default-authentication-plugin=mysql_native_password"
    volumes:
```

```
52      - "./env/mariadb/initdb:/docker-entrypoint-initdb.d"
53      - "${MARIADB_DATA_LOCATION}:/var/lib/mysql"
54      - "${MARIADB_CONFIG_FILE}:/etc/mysql/my.cnf"
55    environment:
56      MARIADB_ROOT_PASSWORD: "${MARIADB_ROOT_PASSWORD}"
57
58  phpmyadmin:
59    image: phpmyadmin:${PHPMYADMIN_VERSION}
60    container_name: "phpmyadmin"
61    restart: "always"
62    depends_on:
63      - "mariadb"
64    networks:
65      grownet:
66        ipv4_address: 172.42.0.5
67    ports:
68      - "8080:80"
69    environment:
70      PMA_HOST: "mariadb"
71      PMA_USER: "${PHPMYADMIN_DATABASE_USER}"
72      PMA_PASSWORD: "${PHPMYADMIN_DATABASE_PASSWORD}"
73
74  influxdb:
75    image: influxdb:${INFLUXDB_VERSION}
76    container_name: "influxdb"
77    restart: "always"
78    networks:
79      grownet:
80        ipv4_address: 172.42.0.6
81    ports:
82      - "8086:8086"
83    volumes:
84      - "${INFLUXDB_DATA_LOCATION}:/var/lib/influxdb2:rw"
85    environment:
86      DOCKER_INFLUXDB_INIT_MODE: "${INFLUXDB_MODE}"
87      DOCKER_INFLUXDB_INIT_USERNAME: "${INFLUXDB_USERNAME}"
88      DOCKER_INFLUXDB_INIT_PASSWORD: "${INFLUXDB_PASSWORD}"
89      DOCKER_INFLUXDB_INIT_ORG: "${INFLUXDB_ORGANISATION}"
90      DOCKER_INFLUXDB_INIT_BUCKET: "${INFLUXDB_BUCKET}"
91      DOCKER_INFLUXDB_INIT_ADMIN_TOKEN: "${INFLUXDB_TOKEN}"
92      DOCKER_INFLUXDB_INIT_RETENTION: "${INFLUXDB_RETENTION}"
93
94  sensor-reader:
95    container_name: sensor-reader
96    restart: "always"
97    networks:
98      grownet:
99        ipv4_address: 172.42.0.7
100   depends_on:
101     - "influxdb"
102   build:
103     context: "./env/python3/"
104     dockerfile: "Dockerfile"
105     args:
```

```
106         PYTHON_VERSION: "${PYTHON_VERSION?}"
107     volumes:
108       - "./env/python3/sensor/sensor_reader.py:/sensor_reader.py"
109     devices:
110       - "/dev/i2c-1:/dev/i2c-1"
111     command: "sh -c 'python3 /sensor_reader.py'"
112
113
114   relays-switcher:
115     container_name: "relays-switcher"
116     restart: "always"
117     networks:
118       grownet:
119         ipv4_address: 172.42.0.8
120     build:
121       context: "./env/python3/"
122       dockerfile: "Dockerfile"
123       args:
124         PYTHON_VERSION: "${PYTHON_VERSION?}"
125     links:
126       - "mariadb"
127     depends_on:
128       - "mariadb"
129       - "influxdb"
130     volumes:
131       - "./env/python3/relayboard/relays_switcher.py:/relays_switcher.py"
132     devices:
133       - "/dev/gpiomem:/dev/gpiomem"
134       - "/dev/i2c-1:/dev/i2c-1"
135     command: "sh -c 'python3 /relays_switcher.py'"
136
137   relays-off:
138     container_name: "relays-off"
139     networks:
140       grownet:
141         ipv4_address: 172.42.0.9
142     build:
143       context: "./env/python3/"
144       dockerfile: "Dockerfile"
145       args:
146         PYTHON_VERSION: "${PYTHON_VERSION?}"
147     volumes:
148       - "./env/python3/relayboard/relays_off.py:/relays_off.py"
149     devices:
150       - "/dev/gpiomem:/dev/gpiomem"
151       - "/dev/i2c-1:/dev/i2c-1"
152     command: "sh -c 'python3 /relays_off.py'"
```

Listing A.2: Inhalt: env/nginx/default.conf

```
server {
    index index.php index.html;
    server_name phpfpm.local;
    error_log  /var/log/nginx/error.log;
    access_log /var/log/nginx/access.log;
    root /var/www/html/public;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~ \.php$ {
        try_files $uri =404;
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
        fastcgi_pass php-fpm:9000;
        fastcgi_index index.php;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_path_info;
    }
}
```

Listing A.3: Inhalt: env/php-fpm/Dockerfile

```
ARG PHP_VERSION
FROM php:${PHP_VERSION}

# Installs the latest composer
RUN php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');" && \
    php composer-setup.php && \
    php -r "unlink('composer-setup.php');" && \
    mv composer.phar /usr/local/bin/composer

# Installs growfridge required packages
RUN apt-get update && apt-get install -y bash coreutils libxml2-dev libzip-dev libcurl4-
    openssl-dev unzip vim zip wget git nodejs openssh-client python3 yarn

# Installs the latest node version manager
RUN wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/$(curl -sL https://api.github.com/
    repos/nvm-sh/nvm/releases/latest | grep "tag_name" | cut -c16-22)/install.sh | bash

# Installs laravel php extension requirements
RUN docker-php-ext-install -j$(nproc) pdo_mysql pcntl xml zip curl

CMD ["php-fpm"]
```

Listing A.4: Inhalt: env/php-fpm/php.ini

```
memory_limit = 1G
max_execution_time = 800
date.timezone = "Europe/Berlin"
```

Listing A.5: Inhalt: env/mariadb/initdb/01_create_databases.sql.example

```
    /* Creates the databases and users. Also sets permissions. */

SET old_passwords=0;

CREATE DATABASE PHPMYADMIN_DATABASE_NAME;
CREATE DATABASE LARAVEL_DATABASE_NAME;

CREATE USER 'PHPMYADMIN_DATABASE_USER'@'%' IDENTIFIED BY 'PHPMYADMIN_DATABASE_PASSWORD';
CREATE USER 'GROWFRIDGE_DATABASE_USER'@'%' IDENTIFIED BY 'GROWFRIDGE_DATABASE_PASSWORD';
CREATE USER 'LARAVEL_DATABASE_USER'@'%' IDENTIFIED BY 'LARAVEL_DATABASE_PASSWORD';

GRANT ALL PRIVILEGES ON PHPMYADMIN_DATABASE_NAME.* TO 'PHPMYADMIN_DATABASE_USER'@'%';
GRANT ALL PRIVILEGES ON PHPMYADMIN_DATABASE_NAME.* TO 'LARAVEL_DATABASE_USER'@'%';
GRANT ALL PRIVILEGES ON LARAVEL_DATABASE_NAME.* TO 'PHPMYADMIN_DATABASE_USER'@'%';
GRANT ALL PRIVILEGES ON LARAVEL_DATABASE_NAME.* TO 'LARAVEL_DATABASE_USER'@'%';
GRANT ALL PRIVILEGES ON LARAVEL_DATABASE_NAME.conditions TO 'GROWFRIDGE_DATABASE_USER'@'%';
GRANT ALL PRIVILEGES ON LARAVEL_DATABASE_NAME.schedule TO 'GROWFRIDGE_DATABASE_USER'@'%';

FLUSH PRIVILEGES;
```

Listing A.6: Inhalt: env/mariadb/initdb/02_create_tables.sql.example

```
/* Creates the conditions and schedule table. Also inserts dummy entries. */

USE LARAVEL_DATABASE_NAME;

CREATE TABLE LARAVEL_DATABASE_NAME.conditions (
id INT(5) NOT NULL AUTO_INCREMENT,
name VARCHAR(128) NOT NULL,
info VARCHAR(1024) NOT NULL,
light_white INT(1) NOT NULL,
light_red INT(1) NOT NULL,
temperature FLOAT(5,2) NOT NULL,
temp_delta_top FLOAT(5,2) NOT NULL,
temp_delta_bot FLOAT(5,2) NOT NULL,
humidity FLOAT(5,2) NOT NULL,
hum_delta_top FLOAT(5,2) NOT NULL,
hum_delta_bot FLOAT(5,2) NOT NULL,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ,
updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (id)
) ENGINE = InnoDB;

INSERT INTO conditions (id, name, info, light_white, light_red, temperature, temp_delta_top,
    temp_delta_bot, humidity, hum_delta_top, hum_delta_bot)
VALUES ('1', 'default',
'This setting is an example and fallback entry.\r\n\r\nTemperature set to 20 with range of 0
    to 100\r\nHumidity is set to 50 with range of 0 to 100',
'0','0','20','80', '20', '50', '50', '50');

CREATE TABLE LARAVEL_DATABASE_NAME.schedules (
id INT(5) NOT NULL AUTO_INCREMENT,
condition_id INT(5) NOT NULL,
CONSTRAINT condition_id FOREIGN KEY (condition_id) REFERENCES conditions(id) ON DELETE
    RESTRICT ON UPDATE RESTRICT,
condition_start DATETIME NOT NULL,
condition_end DATETIME NOT NULL,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ,
updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (id)
) ENGINE = InnoDB;

INSERT INTO schedules (id, condition_id, condition_start, condition_end)
VALUES ('1', '1', '1989-09-01 00:00:00', CURRENT_TIMESTAMP)
```

Listing A.7: Inhalt: env/python3/Dockerfile

```
1 ARG PYTHON_VERSION
2 FROM python:${PYTHON_VERSION}
3
4 # Install updates and additional packages.
5 # Upgrade pip and install hardware required python libraries
6 RUN apt update && apt upgrade -y && apt install -y i2c-tools \
7   && /usr/local/bin/python -m pip install --upgrade pip \
8   && pip3 install --no-cache-dir RPi.GPIO RPi.bme280 smbus2 influxdb-client mysql-connector-
      python==8.0.29
```

Listing A.8: Inhalt: env/python3/relayboard/relays_off.py.example

```
    # This script has the sole purpose to turn off all relays

import smbus2
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
RELAY_OFF = 1
RELAY_ON = 0

# Represents all channels of the relay board with a name and its channel number
dict_relay_channel = {
    "RELAY_1_NAME": RELAY_1_CHANNEL,
    "RELAY_2_NAME": RELAY_2_CHANNEL,
    "RELAY_3_NAME": RELAY_3_CHANNEL,
    "RELAY_4_NAME": RELAY_4_CHANNEL,
    "RELAY_5_NAME": RELAY_5_CHANNEL,
    "RELAY_6_NAME": RELAY_6_CHANNEL,
    "RELAY_7_NAME": RELAY_7_CHANNEL,
    "RELAY_8_NAME": RELAY_8_CHANNEL,
}

for relay in dict_relay_channel:
    GPIO.setup(dict_relay_channel[relay], GPIO.OUT, initial=RELAY_OFF)
```

Listing A.9: Inhalt: env/python3/sensor/sensor_reader.py.example

```
    # Required imports
import bme280
import smbus2
import time
from datetime import datetime
from influxdb_client import InfluxDBClient, Point, WritePrecision
from influxdb_client.client.write_api import SYNCHRONOUS

## Initialize the sensor
port = I2C_PORT
address = I2C_ADDRESS
bus = smbus2.SMBus(port)
calibration_params = bme280.load_calibration_params(bus, address)

## Initialize variables for influxdb
# You can generate an API token from the API Tokens Tab in the UI
token = "INFLUXDB_TOKEN"
org = "INFLUXDB_ORGANISATION"
bucket = "INFLUXDB_BUCKET"

values_per_minute = SENSOR_VALUES_PER_MINUTE
value_amount_to_write_to_db = SENSOR_VALUES_SAVED_BEFORE_WRITE_TO_DB

value_counter = 0
data_sequence = []

# Wait for influxdb
time.sleep(10)

while True:
    if datetime.now().second % (60/values_per_minute) == 0:
        # If it is time to take a sensor reading, append the data to an array
        sensor_data = bme280.sample(bus, address, calibration_params)

        data_sequence.append(["sensor,host=growfridge temperature="+str(round(sensor_data.
            temperature, 2))+" "+str(time.time_ns()),
                              "sensor,host=growfridge humidity="+str(round(sensor_data.
                                  humidity, 2))+" "+str(time.time_ns())])
        value_counter += 1

        if value_counter != value_amount_to_write_to_db:
            # Wait until next reading
            time.sleep((60/values_per_minute)*0.9)

    if value_counter == value_amount_to_write_to_db:
        # If the amount of values in the array reached the limit, write all data to database
            and clear the array
        with InfluxDBClient(url="http://influxdb:8086", token=token, org=org) as client:
            write_api = client.write_api(write_options=SYNCHRONOUS)
            write_api.write(bucket, org, data_sequence)
            client.close()

        data_sequence = []
```

```
51          value_counter = 0
```

Listing A.10: Inhalt: env/python3/relayboard/relays_switcher.py.example

```
    # Required imports
import mysql.connector
import bme280
import smbus2
import time
import RPi.GPIO as GPIO
from datetime import datetime
from datetime import timedelta
from influxdb_client import InfluxDBClient, Point, WritePrecision
from influxdb_client.client.write_api import SYNCHRONOUS

# Initialize the GPIO's with turned off status
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
RELAY_OFF = 1
RELAY_ON = 0
TURN_RELAY_ON = 1
TURN_RELAY_OFF = 0
UTC_OFFSET = 2

# Represents all channels of the relay board with a name and its channel number
dict_relay_channel = {
    "RELAY_1_NAME": RELAY_1_CHANNEL,
    "RELAY_2_NAME": RELAY_2_CHANNEL,
    "RELAY_3_NAME": RELAY_3_CHANNEL,
    "RELAY_4_NAME": RELAY_4_CHANNEL,
    "RELAY_5_NAME": RELAY_5_CHANNEL,
    "RELAY_6_NAME": RELAY_6_CHANNEL,
    "RELAY_7_NAME": RELAY_7_CHANNEL,
    "RELAY_8_NAME": RELAY_8_CHANNEL,
}

for relay in dict_relay_channel:
    GPIO.setup(dict_relay_channel[relay], GPIO.OUT, initial=RELAY_OFF)

## Initialize the sensor
port = I2C_PORT
address = I2C_ADDRESS
bus = smbus2.SMBus(port)
calibration_params = bme280.load_calibration_params(bus, address)

## Initialize variables for influxdb
# You can generate an API token from the API Tokens Tab in the UI
token = "INFLUXDB_TOKEN"
org = "INFLUXDB_ORGANISATION"
bucket = "INFLUXDB_BUCKET"

def get_database_connection():
    """
    Returns a mysql connection
    """
    return mysql.connector.connect(
        user="GROWFRIDGE_DATABASE_USER",
```

```
        password="GROWFRIDGE_DATABASE_PASSWORD",
        host="mariadb",
        database="LARAVEL_DATABASE_NAME",
        raise_on_warnings=True)

# Get a database connection
database_connection = get_database_connection()

# Get a database cursor
db_cursor = database_connection.cursor(dictionary=True)

# Time to wait before checking current conditions again
seconds_to_wait_till_next_check = RELAY_WAIT

def get_schedule_table():
    """
    Returns all entries from the growfridge.schedule table
    """
    query = "SELECT * FROM schedules"
    db_cursor.execute(query)
    result = db_cursor.fetchall()
    database_connection.commit()
    return result

def get_active_schedule_entry():
    """
    Returns the active schedule entry or none
    """
    schedules = get_schedule_table()
    if schedules:
        now = datetime.now() + timedelta(hours=UTC_OFFSET)
        for entry in schedules:
            if (entry['condition_start'] < now) and (now < entry['condition_end']):
                return entry

        return None

def get_active_condition():
    """
    Returns the active condition entry
    """
    schedules = get_schedule_table()

    # For the case that there are no active entries in the schedule table
    # this ensures that the dummy entry will be returned
    get_active_condition_query = "SELECT * FROM conditions WHERE id=1"

    if schedules:
        # Check all entries and change the query if there is an active entry
        now = datetime.now() + timedelta(hours=UTC_OFFSET)
        for entry in schedules:
            if (entry['condition_start'] < now) and (now < entry['condition_end']):
                get_active_condition_query = "SELECT * FROM conditions WHERE id=" + str(entry
                    ['condition_id'])
```

```
107                 break
108
109     # Fetch active condition
110     db_cursor.execute(get_active_condition_query)
111     result = db_cursor.fetchone()
112     database_connection.commit()
113
114     # Return active condition
115     return result
116
117 def get_current_sensor_data():
118     """
119     Returns the sensor reading data
120     """
121     return bme280.sample(bus, address, calibration_params)
122
123
124 def change_device_state(sensor_data, condition):
125     # Initialize variables for condition checking
126     current_temp = round(sensor_data.temperature, 2)
127     current_hum = round(sensor_data.humidity, 2)
128
129     """
130     ## Limit values before intervening
131     Depending on your hardware and the conditions in which the growfridge stands
132     these values must be changed accordingly. I recommend doing tests on how
133     fast your hardware changes the current conditions and tweaking the multipliers
134     until you reach the sweet spot of not going over max or under min values.
135     """
136     min_temp = condition['temperature'] - condition['temp_delta_bot']*0.75
137     max_temp = condition['temperature'] + condition['temp_delta_top']*0.75
138     min_hum = condition['humidity'] - condition['hum_delta_bot']*0.75
139     max_hum = condition['humidity'] + condition['hum_delta_top']*0.75
140     driver1_status = condition['light_white']
141     driver2_status = condition['light_red']
142
143     """
144     Create a periphery array and add GPIO status changes depending on the current sensor
            values.
145     Once all values have been checked and the array is completed, execute all changes. For
            every
146     check the state of each relay is logged.
147     """
148     periphery = []
149     if  current_temp < min_temp:
150         GPIO.output(dict_relay_channel['heating'], RELAY_ON)
151         GPIO.output(dict_relay_channel['cooling'], RELAY_OFF)
152         periphery.append(["periphery,host=growfridge heating=1" + " " + str(time.time_ns()),
153                           "periphery,host=growfridge cooling=0" + " " + str(time.time_ns())])
154     elif current_temp > max_temp:
155         GPIO.output(dict_relay_channel['cooling'], RELAY_ON)
156         GPIO.output(dict_relay_channel['heating'], RELAY_OFF)
157         periphery.append(["periphery,host=growfridge cooling=1" + " " + str(time.time_ns()),
158                           "periphery,host=growfridge heating=0" + " " + str(time.time_ns())])
```

```
159     elif min_temp < current_temp < max_temp:
160         GPIO.output(dict_relay_channel['heating'], RELAY_OFF)
161         GPIO.output(dict_relay_channel['cooling'], RELAY_OFF)
162         periphery.append(["periphery,host=growfridge cooling=0" + " " + str(time.time_ns()),
163                           "periphery,host=growfridge heating=0" + " " + str(time.time_ns())])
164 
165     if  current_hum < min_hum:
166         GPIO.output(dict_relay_channel['humidifier'], RELAY_ON)
167         GPIO.output(dict_relay_channel['fan'], RELAY_OFF)
168         periphery.append(["periphery,host=growfridge humidifier=1" + " " + str(time.time_ns()
                ),
169                           "periphery,host=growfridge fan=0" + " " + str(time.time_ns())])
170     elif current_hum > max_hum:
171         GPIO.output(dict_relay_channel['fan'], RELAY_ON)
172         GPIO.output(dict_relay_channel['humidifier'], RELAY_OFF)
173         periphery.append(["periphery,host=growfridge fan=1" + " " + str(time.time_ns()),
174                           "periphery,host=growfridge humidifier=0" + " " + str(time.time_ns()
                              )])
175     elif min_hum < current_hum < max_hum:
176         GPIO.output(dict_relay_channel['humidifier'], RELAY_OFF)
177         GPIO.output(dict_relay_channel['fan'], RELAY_OFF)
178         periphery.append(["periphery,host=growfridge humidifier=0" + " " + str(time.time_ns()
                ),
179                           "periphery,host=growfridge fan=0" + " " + str(time.time_ns())])
180 
181     if driver1_status == GPIO.input(dict_relay_channel['driver1']):
182         if driver1_status == TURN_RELAY_ON:
183             GPIO.output(dict_relay_channel['driver1'], RELAY_ON)
184             periphery.append(["periphery,host=growfridge driver1=1" + " " + str(time.time_ns
                    ())])
185         else:
186             GPIO.output(dict_relay_channel['driver1'], RELAY_OFF)
187             periphery.append(["periphery,host=growfridge driver1=0" + " " + str(time.time_ns
                    ())])
188     else:
189         periphery.append(["periphery,host=growfridge driver1=" + str(driver1_status) + " " +
                str(time.time_ns())])
190 
191     if driver2_status == GPIO.input(dict_relay_channel['driver2']):
192         if driver2_status == TURN_RELAY_ON:
193             GPIO.output(dict_relay_channel['driver2'], RELAY_ON)
194             periphery.append(["periphery,host=growfridge driver2=1" + " " + str(time.time_ns
                    ())])
195         else:
196             GPIO.output(dict_relay_channel['driver2'], RELAY_OFF)
197             periphery.append(["periphery,host=growfridge driver2=0" + " " + str(time.time_ns
                    ())])
198     else:
199         periphery.append(["periphery,host=growfridge driver2=" + str(driver2_status) + " " +
                str(time.time_ns())])
200 
201     # Write data to influxdb
202     with InfluxDBClient(url="http://influxdb:8086", token=token, org=org) as client:
203         write_api = client.write_api(write_options=SYNCHRONOUS)
```

```
204         write_api.write(bucket, org, periphery)
205         client.close()
206 
207 condition_before = []
208 schedule_before = []
209 
210 while True:
211     # Get active condition
212     active_condition = get_active_condition()
213 
214     if not condition_before and not schedule_before:
215         # Init variables on first run
216         condition_before = active_condition
217         condition_query = "SELECT * FROM schedules WHERE condition_id=" + str(
                condition_before['id'])
218         db_cursor.execute(condition_query)
219         schedule = db_cursor.fetchall()
220         database_connection.commit()
221         schedule_before = schedule[0]
222 
223     if active_condition['id'] != condition_before['id']:
224         # The conditions have been changed
225 
226         # Don't change the schedule start and end times if it is the dummy schedule
227         if schedule_before['id'] != 1:
228             # Add one day to start and end times
229             time_delta_one_day = timedelta(days=1)
230             schedule_before['condition_start'] += time_delta_one_day
231             schedule_before['condition_end'] += time_delta_one_day
232             write_query = "UPDATE `schedules` " + \
233                    "SET " + \
234                    "`condition_start` = \'" + str(schedule_before['condition_start']) + "\',
                           " + \
235                    "`condition_end` = \'" + str(schedule_before['condition_end']) + "\' " + \
236                    "WHERE " + \
237                    "`schedules`.`id` = " + str(schedule_before['id'])
238             # Write changed to the database
239             db_cursor.execute(write_query)
240             database_connection.commit()
241 
242             # Update condition and schedule before entries
243             condition_before = active_condition
244             schedule_before = get_active_schedule_entry()
245 
246     # Change relay states depending on the active condition
247     change_device_state(get_current_sensor_data(), get_active_condition())
248     time.sleep(5)
```

Listing A.11: Inhalt: Makefile

```
    # Source @see https://tech.davis-hansson.com/p/make/
SHELL := /bin/bash
.SHELLFLAGS := -eu -o pipefail -c
MAKEFLAGS += --warn-undefined-variables
MAKEFLAGS += --no-builtin-rules

# Source @see https://www.thapaliya.com/en/writings/well-documented-makefiles/
DEFAULT_GOAL := help
help:
  @awk 'BEGIN {FS = ":.*##"; printf "\nUsage:\n  make \033[36m<target>\033[0m\n"} /^[a-zA-Z0
      -9_-]+:.*?##/ { printf "  \033[36m%-40s\033[0m %s\n", $$1, $$2 } /^##@/ { printf "\n
      \033[1m%s\033[0m\n", substr($$0, 5) } ' $(MAKEFILE_LIST)

include .make/*.mk
```

Listing A.12: Inhalt: .make/01_Setup.mk

```
    ##@ [Install the Growfridge. Run in this order]

# .PHONY is needed in case of having a file with the same name in the same directory.
# .PHONY will prevent using the local same name file and instead calls the defined task
.PHONY: setup-login
setup-login: ## Initialize credentials. Modify the .env file for your needs before you proceed!
  @bash ./install/config-copy.sh

.PHONY: setup-software
setup-software: ## Needs root! Execute this to install system required software. Reboots afterwards!
  @bash ./install/setup.sh

.PHONY: setup-drivers
setup-drivers: ## Needs root! Install required hardware drivers. Reboots afterwards!
  @bash ./install/arducam-drivers.sh

.PHONY: setup-cron
setup-cron: ## Install required camera cronjob.
  @bash ./install/arducam-cron.sh

.PHONY: setup-config
setup-config: ## Execute this after you updated your .env file
  @bash ./install/config-init.sh

.PHONY: setup-container
setup-container: ## Setup Growfridge. This will shutdown by itself
  # Start mariadb and influxdb before starting others so they have time to initialize
  @docker-compose up -d mariadb influxdb
  @docker-compose up -d php-fpm nginx phpmyadmin sensor-reader relays-switcher
  # Installing laravel components and set permissions
  @bash ./install/setup-laravel.sh
  # At this point all containers have been initialized.
  @docker-compose down
  # Start the relays-off container to shut down all relays just in case of weird behavior
  @docker-compose up -d relays-off
  # Execute docker-compose down again to remove relays-off container.
  @docker-compose down
```

Listing A.13: Inhalt: .env.example

```
##### Please modify for your personal needs before you go on and start the system the first time
##### NO SPACES AFTER '=' !!!
### Passwords and tokens
### Run bash ./install/passwords.sh for some passwords

MARIADB_ROOT_PASSWORD=
GROWFRIDGE_DATABASE_PASSWORD=
PHPMYADMIN_DATABASE_PASSWORD=
LARAVEL_DATABASE_PASSWORD=
INFLUXDB_PASSWORD=

# Influxdb creates tokens with 90 characters
INFLUXDB_TOKEN=



# This setup is made for a 8 channel relay board.
# Update names and channel for your needs. I let my settings stay as default.
RELAY_1_NAME=driver1
RELAY_1_CHANNEL=26
RELAY_2_NAME=driver2
RELAY_2_CHANNEL=25
RELAY_3_NAME=cooling
RELAY_3_CHANNEL=16
RELAY_4_NAME=heating
RELAY_4_CHANNEL=24
RELAY_5_NAME=fan
RELAY_5_CHANNEL=23
RELAY_6_NAME=humidifier
RELAY_6_CHANNEL=22
RELAY_7_NAME=dummy1
RELAY_7_CHANNEL=27
RELAY_8_NAME=dummy2
RELAY_8_CHANNEL=17

# Seconds to wait till next condition check
RELAY_WAIT=5

# If sudo i2cdetect -y 1 shows a device on address 0x76 let it as it is, otherwise adjust
I2C_PORT=1
I2C_ADDRESS=0x76

SENSOR_VALUES_PER_MINUTE=6
SENSOR_VALUES_SAVED_BEFORE_WRITE_TO_DB=6

### Can be edited but default values are given

# When using a timezone the slash needs to be escaped with a backslash like seen in the current setting
TIMEZONE=Europe\/Berlin

GROWFRIDGE_DATABASE_USER=growfridge
```

```
52 PHPMYADMIN_DATABASE_USER=pma_user
53 PHPMYADMIN_DATABASE_NAME=phpmyadmin
54 LARAVEL_DATABASE_USER=laravel
55 LARAVEL_DATABASE_NAME=laravel
56 INFLUXDB_USERNAME=dashboard
57 INFLUXDB_ORGANISATION=RPi
58 INFLUXDB_BUCKET=growfridge_data
59
60 ### Can be edited but only change if you know what you are doing
61
62 ### Service docker images to use
63 # Example: NGINX_VERSION=1.23.1 -> nginx:NGINX_VERSION -> nginx:1.23.1 <- resulting docker
       image to use
64 NGINX_VERSION=1.23.1
65 PHP_VERSION=8.1.9-fpm
66 PYTHON_VERSION=3.9.13
67 MARIADB_VERSION=10.8.3
68 PHPMYADMIN_VERSION=5.2.0
69 INFLUXDB_VERSION=2.3.0-alpine
70
71 NGINX_CONFIG_LOCATION=./env/nginx/default.conf
72 PHP_CONFIG_LOCATION=./env/php-fpm/php.ini
73 MARIADB_CONFIG_FILE=./env/mariadb/my.cnf
74 MARIADB_SERVER_CONFIG_FILE=./env/mariadb/50-server.cnf
75 MARIADB_DATA_LOCATION=./data/mariadb
76 INFLUXDB_DATA_LOCATION=./data/influxdb2/database
77 INFLUXDB_MODE=setup
78 INFLUXDB_RETENTION=28d
79 APP_DIR=./src
```

Listing A.14: Inhalt: install/config-copy.sh

```
#!/bin/bash
# Copy all example files in their own directory without the .example in the filename

cp ./.env.example ./.env
cp ./src/.env.example ./src/.env
cp ./env/mariadb/initdb/01_create_databases.sql.example ./env/mariadb/initdb/01
    _create_databases.sql
cp ./env/mariadb/initdb/02_create_tables.sql.example ./env/mariadb/initdb/02_create_tables.
    sql
cp ./env/python3/relayboard/relays_off.py.example ./env/python3/relayboard/relays_off.py
cp ./env/python3/relayboard/relays_switcher.py.example ./env/python3/relayboard/
    relays_switcher.py
cp ./env/python3/sensor/sensor_reader.py.example ./env/python3/sensor/sensor_reader.py

sed -i "s/REPLACE_USER/$(echo $USER)/g" ./env/python3/camera.sh

echo "Please update your .env file"
echo "Insert your login credentials and create secure passwords!"
```

Listing A.15: Inhalt: install/setup.sh

```
#!/bin/bash
echo -e "\n\n\tMUST BE RUN AS ROOT!\n\n"

echo -e "\n\n\tEnable I2C Interface\n\n"

# Activating the I2C interface and updating the boot config
raspi-config nonint do_i2c 1
sed -i "s/dtparam=i2c_arm=off/dtparam=i2c_arm=on/g" /boot/config.txt

echo -e "\n\n\tEnable Camera in /boot/config.txt\n\n"

# Check, if memory already has been reserved for the camera
if ! grep -rnw '/boot/config.txt' -e 'dtoverlay=vc4-kms-v3d,cma-512'
then
  # This config update is specially for the raspberry pi 4.
  sed -i "s/arm_boost=1/arm_boost=1\ndtoverlay=vc4-kms-v3d,cma-512/g" /boot/config.txt

  # The position can be different depending on the raspberry pi model (eg. rpi 3)
  # This project runs on a rpi 4 but if you modify the setup and somehow use a rpi 3
  # with the arducam 64MP, then add this line on the end of you config (without the # and '')
  # 'dtoverlay=vc4-kms-v3d,cma-512'
fi

# PErform a full update, upgrade and install system required packages
apt update && apt install -y git curl wget libffi-dev libssl-dev python3 python3-dev python3-
    pip

# Install docker
curl -sSL https://get.docker.com | sh

# Add current user to the docker group so it has permissions on the docker socket
usermod -aG docker $(id -nu 1000)

# Install latest docker-compose
curl -L "https://github.com/docker/compose/releases/download/$(curl https://github.com/docker
    /compose/releases | grep -m1 '<a href="/docker/compose/releases/download/' | grep -o 'v
    [0-9]*.[0-9]*.[0-9]*')/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-
    compose

# Add execute permissions on docker-compose
chmod +x /usr/local/bin/docker-compose

# Enable the docker service
systemctl enable docker

echo -e "\n\n\tA REBOOT IS NEEDED\n\n"
echo -e "\n\n\tRUN 'sudo make setup-drivers' NEXT\n\n"

reboot
```

Listing A.16: Inhalt: install/arducam-drivers.sh

```
#!/bin/bash
echo -e "\n\n\tMUST BE RUN AS ROOT!\n\n"

# Download the current camera driver installer
# This is right from their documentation
# https://www.arducam.com/docs/cameras-for-raspberry-pi/64mp-camera-for-raspberry-pi/64mp-camera-troubleshooting/
wget -O install_pivariety_pkgs.sh https://github.com/ArduCAM/Arducam-Pivariety-V4L2-Driver/releases/download/install_script/install_pivariety_pkgs.sh
chmod +x install_pivariety_pkgs.sh

# Run the install script for the camera drivers and automatically decline reboot now question at the end of it
yes n | ./install_pivariety_pkgs.sh -p 64mp_pi_hawk_eye_kernel_driver
./install_pivariety_pkgs.sh -p libcamera_dev
./install_pivariety_pkgs.sh -p libcamera_apps

# Do some cleanup afterwards
rm -rf \
  64mp_pi_hawk_eye_kernel_driver_links.txt \
  arducam_64mp_kernel_driver_* \
  install_pivariety_pkgs.sh \
  libcamera-apps-* \
  libcamera_apps_links.txt \
  libcamera-dev-* \
  libcamera_dev_links.txt \
  packages.txt \
  Release

echo -e "\n\n\tA REBOOT IS NEEDED\n\n"
echo -e "\n\n\tRUN 'sudo make setup-cron' NEXT\n\n"

reboot
```

Listing A.17: Inhalt: install/arducam-cron.sh

```
#!/bin/bash
# Installs the cronjob which takes a picture and
# overrides the older one in the same location

# It might be the case, that there is no crontab for the user.
if crontab -l
then
  # Check if cronjob has already been added
  if ! crontab -l | grep 'camera.sh'
  then
    # Cronjob not found, backing up old cronjobs
    crontab -l > crontabs

    # Adding new cronjobs at the and of the backed up file
    # Adding the PATH variable before adding the cronjob enables access to it inside the cronjob
    echo 'PATH="/usr/local/bin:/usr/bin:/bin"' >> crontabs
    echo '*/15 * * * * sh /home/'$USER'/rpi-growfridge/env/python3/camera.sh' >> crontabs

    # Replace existing cronjobs with updated file
    crontab crontabs

    # Remove updated backup file
    rm crontabs
    echo -e "\n\n\tA REBOOT IS NEEDED\n\n"
    echo -e "\n\n\tIf you haven't modified your .env file yet, do it before the next step!\n\n"
    echo -e "\n\n\tRUN make setup-config NEXT\n\n"

  else
    # Cronjob has been found, nothing will be done
    echo -e "\n\n\tCronjob already installed\n\n"
  fi
else
  # There is no crontab. Initialize with default template
  echo "# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
```

```
52 # 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
53 #
54 # For more information see the manual pages of crontab(5) and cron(8)
55 #
56 # m h  dom mon dow   command" > crontabs
57
58   # Adding new cronjobs at the and of the backed up file
59   # Adding the PATH variable before adding the cronjob enables access to it inside the
       cronjob
60   echo 'PATH="/usr/local/bin:/usr/bin:/bin"' >> crontabs
61   echo '*/15 * * * * sh /home/'$USER'/rpi-growfridge/env/python3/camera.sh' >> crontabs
62
63   # Replace existing cronjobs with updated file
64   crontab crontabs
65
66   # Remove updated backup file
67   rm crontabs
68   echo -e "\n\n\tIf you haven't modified your .env file yet, do it before the next step!\n\n"
69   echo -e "\n\n\tRUN 'make setup-config' NEXT\n\n"
70 fi
```

Listing A.18: Inhalt: env/python3/camera.sh

```
#!/bin/bash

# Executes libcamera with autofocus.
# Because of the way the camera is mounted it creates upside down pictures.
# Luckily for us, libcamera supports flipping images
# Also we want to always override the older picture in the same location
libcamera-still --autofocus --hflip --vflip --output '/home/REPLACE_USER/rpi-growfridge/src/
    public/pic.jpg'
```

Listing A.19: Inhalt: install/config-init.sh

```
#!/bin/bash
# Preparing the copied example files with values from the .env file

# Array of all placeholders
declare -a placeholder=(GROWFRIDGE_DATABASE_USER GROWFRIDGE_DATABASE_PASSWORD
    PHPMYADMIN_DATABASE_NAME PHPMYADMIN_DATABASE_USER PHPMYADMIN_DATABASE_PASSWORD
    LARAVEL_DATABASE_NAME LARAVEL_DATABASE_USER LARAVEL_DATABASE_PASSWORD INFLUXDB_TOKEN
    INFLUXDB_ORGANISATION INFLUXDB_BUCKET PHPMYADMIN_DATABASE_USER
    PHPMYADMIN_DATABASE_PASSWORD GROWFRIDGE_DATABASE_NAME RELAY_1_NAME RELAY_2_NAME
    RELAY_3_NAME RELAY_4_NAME RELAY_5_NAME RELAY_6_NAME RELAY_7_NAME RELAY_8_NAME
    RELAY_1_CHANNEL RELAY_2_CHANNEL RELAY_3_CHANNEL RELAY_4_CHANNEL RELAY_5_CHANNEL
    RELAY_6_CHANNEL RELAY_7_CHANNEL RELAY_8_CHANNEL RELAY_WAIT I2C_PORT I2C_ADDRESS
    SENSOR_VALUES_PER_MINUTE SENSOR_VALUES_SAVED_BEFORE_WRITE_TO_DB)

# For each placeholder replace its value if found in file
for i in "${placeholder[@]}"
do
    :
    sed -i "s/$(echo $i)/$(grep "$i" ./.env | cut -d "=" -f2)/g" ./env/mariadb/initdb/*.sql
    sed -i "s/$(echo $i)/$(grep "$i" ./.env | cut -d "=" -f2)/g" ./env/python3/relayboard/*.py
    sed -i "s/$(echo $i)/$(grep "$i" ./.env | cut -d "=" -f2)/g" ./env/python3/sensor/*.py
done

sed -i "s/INFLUXDB_ORGANISATION_REPLACE/$(grep "INFLUXDB_ORGANISATION" ./.env | cut -d "=" -
    f2)/g" ./src/.env
sed -i "s/INFLUXDB_TOKEN_REPLACE/$(grep "INFLUXDB_TOKEN" ./.env | cut -d "=" -f2)/g" ./src/.
    env
sed -i "s/INFLUXDB_DBNAME_REPLACE/$(grep "INFLUXDB_ORGANISATION" ./.env | cut -d "=" -f2)/g"
    ./src/.env
sed -i "s/INFLUXDB_USER_REPLACE/$(grep "INFLUXDB_USERNAME" ./.env | cut -d "=" -f2)/g" ./src
    /.env
sed -i "s/INFLUXDB_PASSWORD_REPLACE/$(grep "INFLUXDB_PASSWORD" ./.env | cut -d "=" -f2)/g" ./
    src/.env
sed -i "s/LARAVEL_DATABASE_PASSWORD_REPLACE/$(grep "LARAVEL_DATABASE_PASSWORD" ./.env | cut -
    d "=" -f2)/g" ./src/.env
sed -i "s/INFLUXDB_BUCKET_REPLACE/$(grep "INFLUXDB_BUCKET" ./.env | cut -d "=" -f2)/g" ./src
    /.env
sed -i "s/TIMEZONE_REPLACE/$(grep "TIMEZONE" ./.env | cut -d "=" -f2)/g" ./src/.env

echo -e "\n\n\tRUN 'make setup-container' NEXT\n\n"
```

Listing A.20: Inhalt: install/setup-laravel.sh

```
#!/bin/bash

# Execute commands within the docker container  to install laravel
docker exec php-fpm sh -c 'composer install'
docker exec php-fpm sh -c 'php artisan key:generate'
docker exec php-fpm sh -c 'php artisan migrate'
docker exec php-fpm bash -c '[[ -s $HOME/.nvm/nvm.sh ]] && . $HOME/.nvm/nvm.sh && nvm install 14'
docker exec php-fpm bash -c '[[ -s $HOME/.nvm/nvm.sh ]] && . $HOME/.nvm/nvm.sh && npm install'
docker exec php-fpm bash -c '[[ -s $HOME/.nvm/nvm.sh ]] && . $HOME/.nvm/nvm.sh && npm run build'

# Set correct ownership and permissions for laravel files and directories
sudo chown -R www-data:www-data ./src
sudo find ./src -type f -exec chmod 644 {} \;
sudo find ./src -type d -exec chmod 755 {} \;
sudo chmod 666 ./src/public/pic.jpg
```

Listing A.21: Inhalt: install/laravel-toggle-registration.sh

```
#!/bin/bash

if [ $1 == 'OFF' ]; then
    sed -i "s/Route::post('register',\s\[RegisteredUserController::class,\s'store']);/\/\/
        Route::post('register', \[RegisteredUserController::class, 'store']);/g" src/routes/
        auth.php
    sed -i "s/Route::get('register',\s\[RegisteredUserController::class,\s'create'])/\/\/
        Route::get('register', \[RegisteredUserController::class, 'create'])/g" src/routes/
        auth.php
    sed -i "s/->name('register');/\/\/->name('register');/g" src/routes/auth.php
fi

if [ $1 == 'ON' ]; then
    sed -i "s/\/\/Route::post('register',\s\[RegisteredUserController::class,\s'store']);/
        Route::post('register', \[RegisteredUserController::class, 'store']);/g" src/routes/
        auth.php
    sed -i "s/\/\/Route::get('register',\s\[RegisteredUserController::class,\s'create'])/
        Route::get('register', \[RegisteredUserController::class, 'create'])/g" src/routes/
        auth.php
    sed -i "s/\/\/->name('register');/->name('register');/g" src/routes/auth.php
fi
```

Listing A.22: Inhalt: .make/02_Growfridge.mk

```
##@ [Growfridge]

.PHONY: growfridge-start
growfridge-start: ## Start the Growfridge
  # Start all services except the relays-off service
  @docker-compose up -d mariadb influxdb
  @docker-compose up -d php-fpm nginx phpmyadmin sensor-reader relays-switcher

.PHONY: growfridge-stop
growfridge-stop: ## Stop the Growfridge
  @docker-compose down
  # Start the relays-off container to shut down all relays.
  @docker-compose up -d relays-off
  # Execute docker-compose down again to remove relays-off container.
  @docker-compose down

.PHONY: growfridge-restart
growfridge-restart: ## Restart the Growfridge
  @make growfridge-stop
  @make growfridge-start

.PHONY: growfridge-registration-off
growfridge-registration-off: ## Needs root! Removes the user registration option
  @bash ./install/laravel-toggle-registration.sh OFF

.PHONY: growfridge-registration-on
growfridge-registration-on: ## Needs root! Activates the user registration option
  @bash ./install/laravel-toggle-registration.sh ON

.PHONY: generate-passwords
generate-passwords: ## If you have to generate new secure passwords run this
  @bash ./install/passwords.sh
```

Listing A.23: Inhalt: install/passwords.sh

```
#!/bin/bash
# Generates the correct amount of passwords needed for this setup

echo -e "\nHere are some passwords:"

for i in {1..5}
do
  # Using translate (tr) to filter all alphanumeric characters (alnum)
  # from a random number generator (/dev/urandom) piping the output (|)
  # into head to cut it after its first 64 characters (head -c 64)
  #
  # The reason this is done with alphanumeric characters is because
  # adding special characters might break the script or config files with
  # characters like " or ' etc.
  #
  # To still be secure i just increased the length to 64 characters
  tr -cd </dev/urandom "[:alnum:]" | head -c 64
  echo ""
done
echo ""

echo -e "Here is your influxdb token:"
tr -cd </dev/urandom "[:alnum:]" | head -c 90
echo -e "\n"
```

Listing A.24: Inhalt: src/routes/web.php

```php
<?php

use App\Http\Controllers\ConditionController;
use App\Http\Controllers\ScheduleController;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\GrowfridgeController;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/',
    [GrowfridgeController::class, 'getLastValuesWelcome']
);

Route::get('/dashboard',
    [GrowfridgeController::class, 'getLastValues'])
    ->middleware(['auth'])->name('dashboard');

Route::resource('conditions',ConditionController::class);

Route::resource('schedule',ScheduleController::class);

require __DIR__.'/auth.php';
```

Listing A.25: Inhalt: src/app/Http/Controllers/GrowfridgeController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Contracts\Foundation\Application;
use Illuminate\Contracts\View\Factory;
use Illuminate\Contracts\View\View;
use Illuminate\Support\Facades\DB;
use InfluxDB2\Client as Client;

class GrowfridgeController extends Controller
{
    /**
     * Returns a client for the InfluxDB database
     *
     * @return Client
     */
    private function getInfluxdbClient(): Client
    {
        return new Client([
            "url" => "172.42.0.6:8086",
            "token" => env('INFLUXDB_TOKEN'),
            "bucket" => env('INFLUXDB_DBNAME'),
            "org" => env('INFLUXDB_ORGANISATION'),
        ]);
    }

    /**
     * Returns all bucket data for sensor measurements
     * @return array
     */
    public function getInfluxDbBucketData(): array
    {
        $client = $this->getInfluxdbClient();
        $query = "from(bucket: \"".env('INFLUXDB_BUCKET', null)."\") |> range(start: -120s)";
        $tables = $client->createQueryApi()->query($query);
        $client->close();

        $result = [];
        foreach ($tables as $table) {
            foreach ($table->records as $record) {
                if ($record->getMeasurement() === "sensor") {
                    $result[] = [
                        "field" => $record->getField(),
                        "value" => $record->getValue(),
                    ];
                }
            }
        }

        return $result;
    }

```

```
54      /**
55       * Returns last temperature value from the bucket
56       * @return string
57       */
58      public function getInfluxDbTemperatureLastEntry(): string
59      {
60          if (!$this->getInfluxDbBucketData()) {
61              return '';
62          } else {
63              $temperatures = [];
64              foreach ($this->getInfluxDbBucketData() as $entry) {
65                  if ($entry['field'] === 'temperature') {
66                      $temperatures[] = $entry['value'];
67                  }
68              }
69
70              if (!$temperatures) {
71                  return '';
72              } else {
73                  return end($temperatures);
74              }
75          }
76      }
77
78      /**
79       * Returns last humidity value from the bucket
80       * @return string
81       */
82      public function getInfluxDbHumidityLastEntry(): string
83      {
84          if (!$this->getInfluxDbBucketData()) {
85              return '';
86          } else {
87              $humidities = [];
88              foreach ($this->getInfluxDbBucketData() as $entry) {
89                  if ($entry['field'] === 'humidity') {
90                      $humidities[] = $entry['value'];
91                  }
92              }
93
94              if (!$humidities) {
95                  return '';
96              } else {
97                  return end($humidities);
98              }
99          }
100     }
101
102     /**
103      * Returns last values for the dashboard
104      * @return Application|Factory|View
105      */
106     public function getLastValues(): View|Factory|Application
107     {
```

```
108         $lastValues = [
109             'temperature' => $this->getInfluxDbTemperatureLastEntry(),
110             'humidity' => $this->getInfluxDbHumidityLastEntry(),
111             'condition_start' => $this->getSqlActiveScheduleEntry()['condition_start'],
112             'condition_end' => $this->getSqlActiveScheduleEntry()['condition_end'],
113             'setTemp' => $this->getSqlActiveConditionEntry()['temperature'],
114             'tem_delta_top' => $this->getSqlActiveConditionEntry()['temp_delta_top'],
115             'tem_delta_bot' => $this->getSqlActiveConditionEntry()['temp_delta_bot'],
116             'setHum' => $this->getSqlActiveConditionEntry()['humidity'],
117             'hum_delta_top' => $this->getSqlActiveConditionEntry()['hum_delta_top'],
118             'hum_delta_bot' => $this->getSqlActiveConditionEntry()['hum_delta_bot'],
119             'light_white' => $this->getSqlActiveConditionEntry()['light_white'],
120             'light_red' => $this->getSqlActiveConditionEntry()['light_red'],
121         ];
122 
123         return view(
124             'dashboard',
125             $lastValues
126         );
127     }
128 
129     /**
130      * Returns last values for the welcome site
131      * @return Application|Factory|View
132      */
133     public function getLastValuesWelcome(): View|Factory|Application
134     {
135         $lastValues = [
136             'temperature' => $this->getInfluxDbTemperatureLastEntry(),
137             'humidity' => $this->getInfluxDbHumidityLastEntry(),
138         ];
139 
140         return view(
141             'welcome',
142             $lastValues
143         );
144     }
145 
146     /**
147      * Returns the active schedule entry
148      * @return array|mixed|string[]
149      */
150     public function getSqlActiveScheduleEntry(): mixed
151     {
152         $scheduleDB = DB::table('schedules')->select('*')->get();
153         $schedule = json_decode(json_encode($scheduleDB), true);
154 
155         $result = [];
156 
157         foreach ($schedule as $entry) {
158             $start = $entry['condition_start'];
159             $end = $entry['condition_end'];
160             $now = date('Y-m-d H:i:s');
161 
```

```
162             if ($start < $now && $now < $end) {
163                 $result = $entry;
164             }
165         }
166 
167         if($result == null)
168         {
169             return [
170                 'condition_id' => 'No schedule entry yet!',
171                 'condition_start' => 'No schedule entry yet!',
172                 'condition_end' => 'No schedule entry yet!'
173             ];
174         }
175 
176         return $result;
177     }
178 
179     /**
180      * Returns the active condition entry
181      * @return array|mixed
182      */
183     public function getSqlActiveConditionEntry(): mixed
184     {
185         $activeCondition = $this->getSqlActiveScheduleEntry();
186         $allConditionsDB = DB::table('conditions')->select('*')->get();
187         $allConditions = json_decode(json_encode($allConditionsDB), true);
188 
189         $result = [];
190 
191         foreach ($allConditions as $condition) {
192             if($condition['id'] == $activeCondition['condition_id']) {
193                 $result = $condition;
194             }
195         }
196 
197         if ($result == null) {
198             $result = $allConditions[0];
199         }
200 
201         return $result;
202     }
203 
204     /**
205      * Returns all conditions except the default entry
206      * @return mixed
207      */
208     public function getSqlAllConditions(): mixed
209     {
210         $allConditionsDB = DB::connection('growfridge')->table('conditions')->select('*')->
                get();
211         $conditionArray = json_decode(json_encode($allConditionsDB), true);
212 
213         return array_slice($conditionArray, 1);
214     }
```

```
215
216     /**
217      * Returns all schedule entries except the default entry
218      * @return Application|Factory|View
219      */
220     public function getSqlSchedule(): View|Factory|Application
221     {
222         $allConditionsDB = DB::connection('growfridge')->table('schedules')->select('*')->get
                ();
223         $schedule = json_decode(json_encode($allConditionsDB), true);
224
225         return view(
226             'schedule',
227             [
228                 'scheduleEntries' => array_slice($schedule, 1),
229             ],
230         );
231     }
232 }
```

Listing A.26: Inhalt: src/resources/views/dashboard.blade.php

```
<x-app-layout>
    <x-slot name="header">
        <h2 class="font-semibold text-xl text-gray-800 leading-tight">
            {{ __('Growfridge System Overview') }}
        </h2>
    </x-slot>

    <div class="py-12">
        <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
            <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">
                <div class="p-6 bg-white border-b border-gray-200">
                    <div style="display: flex; flex-direction: row">
                        <div>
                            <p class="left">Current Temperature: {{ $temperature }} °C</p>
                            <p class="left">Set Temperature: {{ $setTemp }} / Temperature max
                                : {{ $setTemp + $tem_delta_top }} / Temperature min : {{
                                $setTemp - $tem_delta_bot }}</p>
                            <br>
                            <p class="left">Current Humidity: {{ $humidity }} %</p>
                            <p class="left">Set Humidity: {{ $setHum }} / Humidity max : {{
                                $setHum + $hum_delta_top }} / Humidity min : {{ $setHum -
                                $hum_delta_bot }}</p>
                            <br>
                            <p class="left">Active Condition Start: {{ $condition_start }}</p
                                >
                            <p class="left">Active Condition End  : {{ $condition_end }}</p>
                        </div>
                        <div style="max-width: 33%; max-height: 33%; float: right; margin-
                            left: auto;">
                            <img src="{{URL('/pic.jpg')}}" alt="Growfridge Preview">
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</x-app-layout>
```

Listing A.27: Inhalt: src/app/Http/Controllers/ConditionController.php

```php
<?php

namespace App\Http\Controllers;

use App\Http\Requests\ConditionRequest;
use App\Models\Condition;
use Illuminate\Contracts\Foundation\Application;
use Illuminate\Contracts\View\Factory;
use Illuminate\Contracts\View\View;
use Illuminate\Http\RedirectResponse;

class ConditionController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return Application|Factory|View
     */
    public function index(): View|Factory|Application
    {
        $conditions = Condition::latest()->paginate(10);
        return view('conditions.index', compact('conditions'));
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return Application|Factory|View
     */
    public function create(): View|Factory|Application
    {
        return view('conditions.create');
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param ConditionRequest $request
     * @return RedirectResponse
     */
    public function store(ConditionRequest $request): RedirectResponse
    {
        Condition::create($request->validated());
        return redirect()->route('conditions.index')->with('message', 'Condition Created
            Successfully');
    }

    /**
     * Display the specified resource.
     *
     * @param Condition $condition
     * @return View|Factory|Application
     */
```

```
53      public function show(Condition $condition): View|Factory|Application
54      {
55          return view('conditions.show', compact('condition'));
56      }
57
58      /**
59       * Show the form for editing the specified resource.
60       *
61       * @param Condition $condition
62       * @return View|Factory|Application
63       */
64      public function edit(Condition $condition): View|Factory|Application
65      {
66          return view('conditions.edit', compact('condition'));
67      }
68
69      /**
70       * Update the specified resource in storage.
71       *
72       * @param ConditionRequest $request
73       * @param Condition $condition
74       * @return RedirectResponse
75       */
76      public function update(ConditionRequest $request, Condition $condition): RedirectResponse
77      {
78          $condition->update([
79              'name' => $request->name,
80              'info' => $request->info,
81              'light_white' => $request->light_white,
82              'light_red' => $request->light_red,
83              'temperature' => $request->temperature,
84              'temp_delta_top' => $request->temp_delta_top,
85              'temp_delta_bot' => $request->temp_delta_bot,
86              'humidity' => $request->humidity,
87              'hum_delta_top' => $request->hum_delta_top,
88              'hum_delta_bot' => $request->hum_delta_bot,
89          ]);
90
91          return redirect()->route('conditions.index')->with('message', 'Condition Updated
                Successfully');
92      }
93
94      /**
95       * Remove the specified resource from storage.
96       *
97       * @param Condition $condition
98       * @return RedirectResponse
99       */
100     public function destroy(Condition $condition): RedirectResponse
101     {
102         $condition->delete();
103         return redirect()->route('conditions.index')->with('message', 'Condition Deleted
                Successfully');
104     }
```

```
105 }
```

Listing A.28: Inhalt: src/resources/views/conditions/create.blade.php

```
<x-app-layout>
    <x-slot name="header">
        <h2 class="text-xl font-semibold leading-tight text-gray-800">
            {{ __('Create Condition') }}
        </h2>
    </x-slot>

    <div class="font-sans antialiased">
        <div class="flex flex-col items-center min-h-screen pt-6 bg-gray-100 sm:pt-0">

            <div class="max-w-7xl px-16 py-20 mt-6 overflow-hidden bg-white rounded-lg lg:max-w-4xl"style="min-width: 768px">

                <div class="max-w-7xl px-6 py-4 bg-white rounded shadow-md ring-1 ring-gray-900/10">
                    <form method="POST" action="{{ route('conditions.store') }}">
                        @csrf
                        <!-- name -->
                        <div>
                            <label class="block text-sm font-medium text-gray-700" for="name">
                                name
                            </label>

                            <input
                                class="block w-full mt-1 border-gray-300 rounded-md shadow-sm placeholder:text-gray-400 placeholder:text-right focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-50"
                                type="text" name="name" placeholder="256" value="{{old('name')}}">
                            @error('name')
                            <span class="text-red-600 text-sm">
                                {{ $message }}
                            </span>
                            @enderror
                        </div>

                        <!-- Info -->
                        <div class="mt-4">
                            <label class="block text-sm font-medium text-gray-700" for="info">
                                Info
                            </label>
                            <textarea name="info"
                                      class="block w-full mt-1 border-gray-300 rounded-md shadow-sm placeholder:text-gray-400 placeholder:text-right focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-50"
                                      rows="4" placeholder="400"> {{old('info')}}</textarea>
                            @error('info')
                            <span class="text-red-600 text-sm">
                                {{ $message }}
```

```
43                          </span>
44                          @enderror
45                      </div>
46
47                      <!-- light_white -->
48                      <div>
49                          <label class="block text-sm font-medium text-gray-700" for="
                                light_white">
50                              Light white status (0=off, 1=on)
51                          </label>
52
53                          <input
54                              class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                     placeholder:text-gray-400 placeholder:text-right focus:
                                    border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                    ring-opacity-50"
55                              type="text" name="light_white" placeholder="1" value="{{old('
                                    light_white')}}">
56                          @error('light_white')
57                          <span class="text-red-600 text-sm">
58                              {{ $message }}
59                          </span>
60                          @enderror
61                      </div>
62
63                      <!-- light_red -->
64                      <div>
65                          <label class="block text-sm font-medium text-gray-700" for="
                                light_red">
66                              Light red status (0=off, 1=on)
67                          </label>
68
69                          <input
70                              class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                     placeholder:text-gray-400 placeholder:text-right focus:
                                    border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                    ring-opacity-50"
71                              type="text" name="light_red" placeholder="1" value="{{old('
                                    light_red')}}">
72                          @error('light_red')
73                          <span class="text-red-600 text-sm">
74                              {{ $message }}
75                          </span>
76                          @enderror
77                      </div>
78
79                      <!-- temperature -->
80                      <div>
81                          <label class="block text-sm font-medium text-gray-700" for="
                                temperature">
82                              Temperature
83                          </label>
84
85                          <input
```

```
86                                  class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                        placeholder:text-gray-400 placeholder:text-right focus:
                                        border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                        ring-opacity-50"
87                                  type="text" name="temperature" placeholder="5" value="{{old('
                                        temperature')}}">
88                              @error('temperature')
89                              <span class="text-red-600 text-sm">
90                                  {{ $message }}
91                              </span>
92                              @enderror
93                          </div>
94
95                          <!-- temp_delta_top -->
96                          <div>
97                              <label class="block text-sm font-medium text-gray-700" for="
                                    temp_delta_top">
98                                  Temperature delta top
99                              </label>
100
101                             <input
102                                 class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                        placeholder:text-gray-400 placeholder:text-right focus:
                                        border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                        ring-opacity-50"
103                                 type="text" name="temp_delta_top" placeholder="5" value="{{
                                        old('temp_delta_top')}}">
104                             @error('temp_delta_top')
105                             <span class="text-red-600 text-sm">
106                                 {{ $message }}
107                             </span>
108                             @enderror
109                         </div>
110
111                         <!-- temp_delta_bot -->
112                         <div>
113                             <label class="block text-sm font-medium text-gray-700" for="
                                    temp_delta_bot">
114                                 Temperature delta bot
115                             </label>
116
117                             <input
118                                 class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                        placeholder:text-gray-400 placeholder:text-right focus:
                                        border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                        ring-opacity-50"
119                                 type="text" name="temp_delta_bot" placeholder="5" value="{{
                                        old('temp_delta_bot')}}">
120                             @error('temp_delta_bot')
121                             <span class="text-red-600 text-sm">
122                                 {{ $message }}
123                             </span>
124                             @enderror
125                         </div>
```

```
126
127                         <!-- humidity -->
128                         <div>
129                             <label class="block text-sm font-medium text-gray-700" for="
                                    humidity">
130                                 Humidity
131                             </label>
132
133                             <input
134                                 class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                        placeholder:text-gray-400 placeholder:text-right focus:
                                        border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                        ring-opacity-50"
135                                 type="text" name="humidity" placeholder="5" value="{{old('
                                        humidity')}}">
136                             @error('humidity')
137                             <span class="text-red-600 text-sm">
138                                 {{ $message }}
139                             </span>
140                             @enderror
141                         </div>
142
143                         <!-- hum_delta_top -->
144                         <div>
145                             <label class="block text-sm font-medium text-gray-700" for="
                                    hum_delta_top">
146                                 Humidity delta top
147                             </label>
148
149                             <input
150                                 class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                        placeholder:text-gray-400 placeholder:text-right focus:
                                        border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                        ring-opacity-50"
151                                 type="text" name="hum_delta_top" placeholder="5" value="{{old
                                        ('hum_delta_top')}}">
152                             @error('hum_delta_top')
153                             <span class="text-red-600 text-sm">
154                                 {{ $message }}
155                             </span>
156                             @enderror
157                         </div>
158
159                         <!-- hum_delta_bot -->
160                         <div>
161                             <label class="block text-sm font-medium text-gray-700" for="
                                    hum_delta_bot">
162                                 Humidity delta bot
163                             </label>
164
165                             <input
166                                 class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                        placeholder:text-gray-400 placeholder:text-right focus:
```

```
                                    border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                    ring-opacity-50"
167                                 type="text" name="hum_delta_bot" placeholder="5" value="{{old
                                    ('hum_delta_bot')}}">
168                             @error('hum_delta_bot')
169                             <span class="text-red-600 text-sm">
170                                 {{ $message }}
171                             </span>
172                             @enderror
173                         </div>
174
175                         <div class="flex items-center justify-start mt-4">
176                             <button type="submit"
177                                     class="inline-flex items-center px-6 py-2 text-sm font-
                                        semibold rounded-md text-sky-100 bg-sky-500 hover:bg-
                                        sky-700 focus:outline-none focus:border-gray-900
                                        focus:ring ring-gray-300"
178                                     style="border: 1px solid #212529">
179                                 Save
180                             </button>
181                         </div>
182                     </form>
183                 </div>
184             </div>
185         </div>
186     </div>
187
188 </x-app-layout>
```

Listing A.29: Inhalt: src/resources/views/conditions/edit.blade.php

```
<x-app-layout>
    <x-slot name="header">
        <h2 class="text-xl font-semibold leading-tight text-gray-800">
            {{ __('Edit Condition') }}
        </h2>
    </x-slot>

    <div class="font-sans antialiased">
        <div class="flex flex-col items-center min-h-screen pt-6 bg-gray-100 sm:pt-0">

            <div class="w-full px-16 py-20 mt-6 overflow-hidden bg-white rounded-lg lg:max-w-4xl">

                <div class="w-full px-6 py-4 bg-white rounded shadow-md ring-1 ring-gray-900/10">
                    <form method="POST" action="{{ route('conditions.update',$condition->id) }}">
                        @csrf
                        @method('PUT')
                        <!-- name -->
                        <div>
                            <label class="block text-sm font-medium text-gray-700" for="name">
                                Name
                            </label>

                            <input
                                class="block w-full mt-1 border-gray-300 rounded-md shadow-sm placeholder:text-gray-400 placeholder:text-right focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-50"
                                type="text" name="name" placeholder="256" value="{{old('name',$condition->name)}}">
                            @error('name')
                            <span class="text-red-600 text-sm">
                                {{ $message }}
                            </span>
                            @enderror
                        </div>

                        <!-- Info -->
                        <div class="mt-4">
                            <label class="block text-sm font-medium text-gray-700" for="info">
                                Info
                            </label>
                            <textarea name="info"
                                      class="block w-full mt-1 border-gray-300 rounded-md shadow-sm placeholder:text-gray-400 placeholder:text-right focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-50"
                                      rows="4" placeholder="400"> {{old('info',$condition->info)}}</textarea>
```

```
41                          @error('info')
42                          <span class="text-red-600 text-sm">
43                              {{ $message }}
44                          </span>
45                          @enderror
46                      </div>
47
48                      <div>
49                          <label class="block text-sm font-medium text-gray-700" for="
                                light_white">
50                              Light white status (0=off, 1=on)
51                          </label>
52
53                          <input
54                              class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                     placeholder:text-gray-400 placeholder:text-right focus:
                                    border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                    ring-opacity-50"
55                              type="text" name="light_white" placeholder="256" value="{{old
                                    ('light_white',$condition->light_white)}}">
56                          @error('light_white')
57                          <span class="text-red-600 text-sm">
58                              {{ $message }}
59                          </span>
60                          @enderror
61                      </div>
62
63                      <div>
64                          <label class="block text-sm font-medium text-gray-700" for="
                                light_red">
65                              Light red status (0=off, 1=on)
66                          </label>
67
68                          <input
69                              class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                     placeholder:text-gray-400 placeholder:text-right focus:
                                    border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                    ring-opacity-50"
70                              type="text" name="light_red" placeholder="256" value="{{old('
                                    light_red',$condition->light_red)}}">
71                          @error('light_red')
72                          <span class="text-red-600 text-sm">
73                              {{ $message }}
74                          </span>
75                          @enderror
76                      </div>
77
78                      <div>
79                          <label class="block text-sm font-medium text-gray-700" for="
                                temperature">
80                              Temperature
81                          </label>
82
83                          <input
```

```
84                                  class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                        placeholder:text-gray-400 placeholder:text-right focus:
                                        border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                        ring-opacity-50"
85                                  type="text" name="temperature" placeholder="256" value="{{old
                                        ('temperature',$condition->temperature)}}">
86                              @error('temperature')
87                              <span class="text-red-600 text-sm">
88                                  {{ $message }}
89                              </span>
90                              @enderror
91                          </div>
92
93                          <div>
94                              <label class="block text-sm font-medium text-gray-700" for="
                                    temp_delta_top">
95                                  Temperature delta top
96                              </label>
97
98                              <input
99                                  class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                        placeholder:text-gray-400 placeholder:text-right focus:
                                        border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                        ring-opacity-50"
100                                 type="text" name="temp_delta_top" placeholder="256" value="{{
                                        old('temp_delta_top',$condition->temp_delta_top)}}">
101                             @error('temp_delta_top')
102                             <span class="text-red-600 text-sm">
103                                 {{ $message }}
104                             </span>
105                             @enderror
106                         </div>
107
108                         <div>
109                             <label class="block text-sm font-medium text-gray-700" for="
                                    temp_delta_bot">
110                                 Temperature delta bot
111                             </label>
112
113                             <input
114                                 class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                        placeholder:text-gray-400 placeholder:text-right focus:
                                        border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                        ring-opacity-50"
115                                 type="text" name="temp_delta_bot" placeholder="256" value="{{
                                        old('temp_delta_bot',$condition->temp_delta_bot)}}">
116                             @error('temp_delta_bot')
117                             <span class="text-red-600 text-sm">
118                                 {{ $message }}
119                             </span>
120                             @enderror
121                         </div>
122
123                         <div>
```

```
124                         <label class="block text-sm font-medium text-gray-700" for="
                                humidity">
125                             Humidity
126                         </label>
127
128                         <input
129                             class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                     placeholder:text-gray-400 placeholder:text-right focus:
                                    border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                    ring-opacity-50"
130                             type="text" name="humidity" placeholder="256" value="{{old('
                                humidity',$condition->humidity)}}">
131                         @error('humidity')
132                         <span class="text-red-600 text-sm">
133                             {{ $message }}
134                         </span>
135                         @enderror
136                     </div>
137
138                     <div>
139                         <label class="block text-sm font-medium text-gray-700" for="
                                hum_delta_top">
140                             Humidity delta top
141                         </label>
142
143                         <input
144                             class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                     placeholder:text-gray-400 placeholder:text-right focus:
                                    border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                    ring-opacity-50"
145                             type="text" name="hum_delta_top" placeholder="256" value="{{
                                old('hum_delta_top',$condition->hum_delta_top)}}">
146                         @error('hum_delta_top')
147                         <span class="text-red-600 text-sm">
148                             {{ $message }}
149                         </span>
150                         @enderror
151                     </div>
152
153                     <div>
154                         <label class="block text-sm font-medium text-gray-700" for="
                                hum_delta_bot">
155                             Humidity delta bot
156                         </label>
157
158                         <input
159                             class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                     placeholder:text-gray-400 placeholder:text-right focus:
                                    border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                    ring-opacity-50"
160                             type="text" name="hum_delta_bot" placeholder="256" value="{{
                                old('hum_delta_bot',$condition->hum_delta_bot)}}">
161                         @error('hum_delta_bot')
162                         <span class="text-red-600 text-sm">
```

```
163                                 {{ $message }}
164                             </span>
165                             @enderror
166                         </div>
167
168                         <div class="flex items-center justify-start mt-4">
169                             <button type="submit"
170                                     class="inline-flex items-center px-6 py-2 text-sm font-
                                        semibold rounded-md text-sky-100 bg-sky-500 hover:bg-
                                        sky-700 focus:outline-none focus:border-gray-900
                                        focus:ring ring-gray-300">
171                                 Update
172                             </button>
173                         </div>
174                     </form>
175                 </div>
176             </div>
177         </div>
178     </div>
179
180 </x-app-layout>
```

Listing A.30: Inhalt: src/resources/views/conditions/index.blade.php

```
<x-app-layout>
    <x-slot name="header">
        <h2 class="text-xl font-semibold leading-tight text-gray-800">
            {{ __('Conditions') }}
        </h2>
    </x-slot>
    <div class="container max-w-7xl mx-auto mt-20">
        <div class="mb-4 mt-4">
            @if (session()->has('message'))
                <div class="p-3 rounded bg-green-500 text-green-100 my-2">
                    {{ session('message') }}
                </div>
            @endif

            <div class="flex justify-center mt-4">
                <a href="{{ route('conditions.create')}}"
                   class="px-4 py-2 rounded-md bg-sky-500 text-sky-100 hover:bg-sky-600"
                   style="border: 1px solid #212529">Create Condition</a>
            </div>
        </div>
        <div class="flex flex-col">
            <div class="overflow-x-auto sm:-mx-6 sm:px-6 lg:-mx-8 lg:px-8">
                <div
                    class="inline-block min-w-full overflow-hidden align-middle border-b
                       border-gray-200 shadow sm:rounded-lg">
                    <table class="min-w-full">
                        <thead>
                        <tr>
                            <th
                                class="px-6 py-3 text-xs font-medium leading-4 tracking-wider
                                    text-left text-gray-500 uppercase border-b border-gray
                                   -200 bg-gray-50"
                                style="display: none">
                                ID</th>
                            <th
                                class="px-6 py-3 text-xs font-medium leading-4 tracking-wider
                                    text-left text-gray-500 uppercase border-b border-gray
                                   -200 bg-gray-50">
                                Name</th>
                            <th
                                class="px-6 py-3 text-xs font-medium leading-4 tracking-wider
                                    text-left text-gray-500 uppercase border-b border-gray
                                   -200 bg-gray-50">
                                Info</th>
                            <th
                                class="px-6 py-3 text-xs font-medium leading-4 tracking-wider
                                    text-left text-gray-500 uppercase border-b border-gray
                                   -200 bg-gray-50">
                                Light white</th>
                            <th
                                class="px-6 py-3 text-xs font-medium leading-4 tracking-wider
                                    text-left text-gray-500 uppercase border-b border-gray
                                   -200 bg-gray-50">
```

```
43                                  light red</th>
44                              <th
45                                  class="px-6 py-3 text-xs font-medium leading-4 tracking-wider
                                        text-left text-gray-500 uppercase border-b border-gray
                                       -200 bg-gray-50">
46                                  temp</th>
47                              <th
48                                  class="px-6 py-3 text-xs font-medium leading-4 tracking-wider
                                        text-left text-gray-500 uppercase border-b border-gray
                                       -200 bg-gray-50">
49                                  temp d. top</th>
50                              <th
51                                  class="px-6 py-3 text-xs font-medium leading-4 tracking-wider
                                        text-left text-gray-500 uppercase border-b border-gray
                                       -200 bg-gray-50">
52                                  temp d. bot</th>
53                              <th
54                                  class="px-6 py-3 text-xs font-medium leading-4 tracking-wider
                                        text-left text-gray-500 uppercase border-b border-gray
                                       -200 bg-gray-50">
55                                  hum</th>
56                              <th
57                                  class="px-6 py-3 text-xs font-medium leading-4 tracking-wider
                                        text-left text-gray-500 uppercase border-b border-gray
                                       -200 bg-gray-50">
58                                  hum d. top</th>
59                              <th
60                                  class="px-6 py-3 text-xs font-medium leading-4 tracking-wider
                                        text-left text-gray-500 uppercase border-b border-gray
                                       -200 bg-gray-50">
61                                  hum d. bot</th>
62                              <th class="px-6 py-3 text-sm text-left text-gray-500 border-b
                                   border-gray-200 bg-gray-50"
63                                  colspan="2">
64                                  Action</th>
65                          </tr>
66                          </thead>
67
68                          <tbody class="bg-white">
69                          @foreach ($conditions as $condition)
70                              @if($condition->name !== 'default')
71                                  <tr>
72                                      <td class="px-6 py-4 whitespace-no-wrap border-b border-
                                            gray-200" style="display: none">
73                                          <div class="flex items-center">
74                                              {{ $condition->id }}
75                                          </div>
76
77                                      </td>
78
79                                      <td class="px-6 py-4 whitespace-no-wrap border-b border-
                                            gray-200">
80                                          <div class="text-sm leading-5 text-gray-900">
81                                              {{ $condition->name }}
```

```
82                                      </div>
83                                  </td>
84
85                                  <td class="px-6 py-4 whitespace-no-wrap border-b border-
                                        gray-200">
86                                      {{ $condition->info }}
87                                  </td>
88
89                                  <td class="px-6 py-4 whitespace-no-wrap border-b border-
                                        gray-200">
90                                      {{ $condition->light_white }}
91                                  </td>
92
93                                  <td class="px-6 py-4 whitespace-no-wrap border-b border-
                                        gray-200">
94                                      {{ $condition->light_red }}
95                                  </td>
96
97                                  <td class="px-6 py-4 whitespace-no-wrap border-b border-
                                        gray-200">
98                                      {{ $condition->temperature }}
99                                  </td>
100
101                                 <td class="px-6 py-4 whitespace-no-wrap border-b border-
                                        gray-200">
102                                     {{ $condition->temp_delta_top }}
103                                 </td>
104
105                                 <td class="px-6 py-4 whitespace-no-wrap border-b border-
                                        gray-200">
106                                     {{ $condition->temp_delta_bot }}
107                                 </td>
108
109                                 <td class="px-6 py-4 whitespace-no-wrap border-b border-
                                        gray-200">
110                                     {{ $condition->humidity }}
111                                 </td>
112
113                                 <td class="px-6 py-4 whitespace-no-wrap border-b border-
                                        gray-200">
114                                     {{ $condition->hum_delta_top }}
115                                 </td>
116
117                                 <td class="px-6 py-4 whitespace-no-wrap border-b border-
                                        gray-200">
118                                     {{ $condition->hum_delta_bot }}
119                                 </td>
120
121                                 <td
122                                     class="text-sm font-medium leading-5 text-center
                                            whitespace-no-wrap border-b border-gray-200 ">
123                                     <a href="{{ route('conditions.edit', $condition->id)
                                            }}"
124                                        class="text-indigo-600 hover:text-indigo-900">
```

```
125                                         <svg xmlns="http://www.w3.org/2000/svg" class="w
                                                -6 h-6" fill="none"
126                                              viewBox="0 0 24 24" stroke="currentColor">
127                                             <path stroke-linecap="round" stroke-linejoin=
                                                    "round" stroke-width="2"
128                                                   d="M11 5H6a2 2 0 00-2 2v11a2 2 0 002 2
                                                       h11a2 2 0 002-2v-5m-1.414-9.414a2 2
                                                        0 112.828 2.828L11.828 15H9v-2.828
                                                       l8.586-8.586z" />
129                                         </svg>
130                                     </a>
131
132                                 </td>
133                                 <td class="text-sm font-medium leading-5 whitespace-no-
                                        wrap border-b border-gray-200 ">
134                                     <form action="{{ route('conditions.destroy',
                                            $condition->id) }}" method="POST" onsubmit="
                                            return confirm('{{ trans('are You Sure ? ') }}');
                                            ">
135
136                                         <input type="hidden" name="_method" value="DELETE
                                                ">
137                                         <input type="hidden" name="_token" value="{{
                                                csrf_token() }}">
138                                         <button type="submit" class="flex items-center">
139                                             <svg xmlns="http://www.w3.org/2000/svg"
140                                                  class="w-6 h-6 text-red-600 hover:text-
                                                     red-800 cursor-pointer" fill="none"
141                                                  viewBox="0 0 24 24" stroke="currentColor
                                                     ">
142                                                 <path stroke-linecap="round" stroke-
                                                    linejoin="round" stroke-width="2"
143                                                       d="M19 7l-.867 12.142A2 2 0
                                                           0116.138 21H7.862a2 2 0
                                                           01-1.995-1.858L5 7m5 4v6m4-6
                                                           v6m1-10V4a1 1 0 00-1-1h-4a1 1 0
                                                            00-1 1v3M4 7h16" />
144                                             </svg>
145                                         </button>
146                                     </form>
147                                 </td>
148                             </tr>
149                         @endif
150                     @endforeach
151                     </tbody>
152                 </table>
153             </div>
154         </div>
155     </div>
156 </div>
157 </x-app-layout>
```

Listing A.31: Inhalt: src/app/Http/Controllers/ScheduleController.php

```php
<?php

namespace App\Http\Controllers;

use App\Http\Requests\ScheduleRequest;
use App\Models\Condition;
use App\Models\Schedule;
use Illuminate\Contracts\Foundation\Application;
use Illuminate\Contracts\View\Factory;
use Illuminate\Http\RedirectResponse;
use Illuminate\View\View;

class ScheduleController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return View|Factory|Application
     */
    public function index(): View|Factory|Application
    {
        $schedule = Schedule::latest()->paginate(10);
        $conditions = Condition::all(['id','name']);
        return view('schedule.index', compact('schedule'), compact('conditions'));
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return View|Factory|Application
     */
    public function create(): View|Factory|Application
    {
        $conditions = Condition::all(['id','name']);
        return view('schedule.create', compact('conditions'));
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param ScheduleRequest $request
     * @return RedirectResponse
     */
    public function store(ScheduleRequest $request): RedirectResponse
    {
        Schedule::create($request->validated());
        return redirect()->route('schedule.index')->with('message', 'Schedule Entry Created
            Successfully');
    }

    /**
     * Display the specified resource.
     *
```

```
53      * @param Schedule $schedule
54      * @return View|Factory|Application
55      */
56     public function show(Schedule $schedule): View|Factory|Application
57     {
58         return view('schedule.show', compact('schedule'));
59     }
60
61     /**
62      * Show the form for editing the specified resource.
63      *
64      * @param Schedule $schedule
65      * @return View|Factory|Application
66      */
67     public function edit(Schedule $schedule): View|Factory|Application
68     {
69         $conditions = Condition::all(['id','name']);
70         return view('schedule.edit', compact('schedule'), compact('conditions'));
71     }
72
73     /**
74      * Update the specified resource in storage.
75      *
76      * @param ScheduleRequest $request
77      * @param Schedule $schedule
78      * @return RedirectResponse
79      */
80     public function update(ScheduleRequest $request, Schedule $schedule): RedirectResponse
81     {
82         $schedule->update([
83             'condition_id' => $request->condition_id,
84             'condition_start' => $request->condition_start,
85             'condition_end' => $request->condition_end
86         ]);
87
88         return redirect()->route('schedule.index')->with('message', 'Schedule Entry Updated
                Successfully');
89     }
90
91     /**
92      * Remove the specified resource from storage.
93      *
94      * @param Schedule $schedule
95      * @return RedirectResponse
96      */
97     public function destroy(Schedule $schedule): RedirectResponse
98     {
99         $schedule->delete();
100        return redirect()->route('schedule.index')->with('message', 'Schedule Entry Deleted
                Successfully');
101    }
102 }
```

Listing A.32: Inhalt: src/resources/views/schedule/create.blade.php

```
<x-app-layout>
    <x-slot name="header">
        <h2 class="text-xl font-semibold leading-tight text-gray-800">
            {{ __('Create Schedule') }}
        </h2>
    </x-slot>

    <div class="font-sans antialiased">
        <div class="flex flex-col items-center min-h-screen pt-6 bg-gray-100 sm:pt-0">

            <div class="max-w-7xl px-16 py-20 mt-6 overflow-hidden bg-white rounded-lg lg:max-w-4xl"style="min-width: 768px">

                <div class="max-w-7xl px-6 py-4 bg-white rounded shadow-md ring-1 ring-gray-900/10">
                    <form method="POST" action="{{ route('schedule.store') }}">
                        @csrf
                        <!-- condition_id -->
                        <div>
                            <label class="block text-sm font-medium text-gray-700" for="condition_id">
                                Condition
                            </label>

                            <select name="condition_id" class="form-control">
                                @foreach($conditions as $condition)
                                    <option value="{{$condition->id}}">{{$condition->name}}</option>
                                @endforeach
                            </select>
                            <!-- <input
                                class="block w-full mt-1 border-gray-300 rounded-md shadow-sm placeholder:text-gray-400 placeholder:text-right focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-50"
                                type="text" name="condition_id" placeholder="256" value="{{old('condition_id')}}"> -->
                            @error('condition_id')
                            <span class="text-red-600 text-sm">
                                {{ $message }}
                            </span>
                            @enderror
                        </div>

                        <!-- condition_start -->
                        <div>
                            <label class="block text-sm font-medium text-gray-700" for="condition_start">
                                Condition start
                            </label>

                            <input
```

```
44                                  class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                        placeholder:text-gray-400 placeholder:text-right focus:
                                       border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                       ring-opacity-50"
45                                  type="text" name="condition_start" placeholder="" value="{{
                                       old('condition_start')}}">
46                              @error('condition_start')
47                              <span class="text-red-600 text-sm">
48                                  {{ $message }}
49                              </span>
50                              @enderror
51                          </div>
52
53                          <!-- condition_end -->
54                          <div>
55                              <label class="block text-sm font-medium text-gray-700" for="
                                    condition_end">
56                                  Condition end
57                              </label>
58
59                              <input
60                                  class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                        placeholder:text-gray-400 placeholder:text-right focus:
                                       border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                       ring-opacity-50"
61                                  type="text" name="condition_end" placeholder="" value="{{old
                                       ('condition_end')}}">
62                              @error('condition_end')
63                              <span class="text-red-600 text-sm">
64                                  {{ $message }}
65                              </span>
66                              @enderror
67                          </div>
68
69                          <div class="flex items-center justify-start mt-4">
70                              <button type="submit"
71                                      class="inline-flex items-center px-6 py-2 text-sm font-
                                           semibold rounded-md text-sky-100 bg-sky-500 hover:bg-
                                           sky-700 focus:outline-none focus:border-gray-900
                                           focus:ring ring-gray-300"
72                                      style="border: 1px solid #212529">
73                                  Save
74                              </button>
75                          </div>
76                      </form>
77                  </div>
78              </div>
79          </div>
80      </div>
81
82  </x-app-layout>
```

Listing A.33: Inhalt: src/resources/views/schedule/edit.blade.php

```
<x-app-layout>
    <x-slot name="header">
        <h2 class="text-xl font-semibold leading-tight text-gray-800">
            {{ __('Edit Schedule Entry') }}
        </h2>
    </x-slot>

    <div class="font-sans antialiased">
        <div class="flex flex-col items-center min-h-screen pt-6 bg-gray-100 sm:pt-0">

            <div class="w-full px-16 py-20 mt-6 overflow-hidden bg-white rounded-lg lg:max-w-4xl">

                <div class="w-full px-6 py-4 bg-white rounded shadow-md ring-1 ring-gray-900/10">
                    <form method="POST" action="{{ route('schedule.update',$schedule->id) }}">
                        @csrf
                        @method('PUT')
                        <!-- condition_id -->
                        <div>
                            <label class="block text-sm font-medium text-gray-700" for="condition_id">
                                Condition
                            </label>

                            <select name="condition_id" class="form-control">
                                @foreach($conditions as $condition)
                                    @if($condition->id == $schedule->condition_id)
                                        <option value="{{$condition->id}}" selected>{{$condition->name}}</option>
                                    @else
                                        <option value="{{$condition->id}}">{{$condition->name}}</option>
                                    @endif
                                @endforeach
                            </select>
                            @error('condition_id')
                            <span class="text-red-600 text-sm">
                                {{ $message }}
                            </span>
                            @enderror
                        </div>

                        <div>
                            <label class="block text-sm font-medium text-gray-700" for="condition_start">
                                Condition start
                            </label>

                            <input
                                class="block w-full mt-1 border-gray-300 rounded-md shadow-sm placeholder:text-gray-400 placeholder:text-right focus:
```

```
                                    border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                    ring-opacity-50"
46                                  type="text" name="condition_start" placeholder="256" value="
                                    {{old('condition_start',$schedule->condition_start)}}">
47                                  @error('condition_start')
48                                  <span class="text-red-600 text-sm">
49                                      {{ $message }}
50                                  </span>
51                                  @enderror
52                              </div>
53
54                              <div>
55                                  <label class="block text-sm font-medium text-gray-700" for="
                                    condition_end">
56                                      Condition end
57                                  </label>
58
59                                  <input
60                                      class="block w-full mt-1 border-gray-300 rounded-md shadow-sm
                                       placeholder:text-gray-400 placeholder:text-right focus:
                                      border-indigo-300 focus:ring focus:ring-indigo-200 focus:
                                      ring-opacity-50"
61                                      type="text" name="condition_end" placeholder="256" value="{{
                                      old('condition_end',$schedule->condition_end)}}">
62                                  @error('condition_end')
63                                  <span class="text-red-600 text-sm">
64                                      {{ $message }}
65                                  </span>
66                                  @enderror
67                              </div>
68
69                              <div class="flex items-center justify-start mt-4">
70                                  <button type="submit"
71                                          class="inline-flex items-center px-6 py-2 text-sm font-
                                              semibold rounded-md text-sky-100 bg-sky-500 hover:bg-
                                              sky-700 focus:outline-none focus:border-gray-900
                                              focus:ring ring-gray-300">
72                                      Update
73                                  </button>
74                              </div>
75                          </form>
76                      </div>
77                  </div>
78              </div>
79          </div>
80
81  </x-app-layout>
```

Listing A.34: Inhalt: src/resources/views/schedule/index.blade.php

```
<x-app-layout>
    <x-slot name="header">
        <h2 class="text-xl font-semibold leading-tight text-gray-800">
            {{ __('Schedules') }}
        </h2>
    </x-slot>
    <div class="container max-w-7xl mx-auto mt-20">
        <div class="mb-4 mt-4">
            @if (session()->has('message'))
                <div class="p-3 rounded bg-green-500 text-green-100 my-2">
                    {{ session('message') }}
                </div>
            @endif

            <div class="flex justify-center mt-4">
                <a href="{{ route('schedule.create')}}"
                   class="px-4 py-2 rounded-md bg-sky-500 text-sky-100 hover:bg-sky-600"
                   style="border: 1px solid #212529">Create Schedule</a>
            </div>
        </div>
        <div class="flex flex-col">
            <div class="overflow-x-auto sm:-mx-6 sm:px-6 lg:-mx-8 lg:px-8">
                <div
                    class="inline-block min-w-full overflow-hidden align-middle border-b
                       border-gray-200 shadow sm:rounded-lg">
                    <table class="min-w-full">
                        <thead>
                        <tr>
                            <th
                                class="px-6 py-3 text-xs font-medium leading-4 tracking-wider
                                    text-left text-gray-500 uppercase border-b border-gray
                                   -200 bg-gray-50"
                                style="display: none">
                                ID</th>
                            <th
                                class="px-6 py-3 text-xs font-medium leading-4 tracking-wider
                                    text-left text-gray-500 uppercase border-b border-gray
                                   -200 bg-gray-50">
                                Condition</th>
                            <th
                                class="px-6 py-3 text-xs font-medium leading-4 tracking-wider
                                    text-left text-gray-500 uppercase border-b border-gray
                                   -200 bg-gray-50">
                                Condition Start</th>
                            <th
                                class="px-6 py-3 text-xs font-medium leading-4 tracking-wider
                                    text-left text-gray-500 uppercase border-b border-gray
                                   -200 bg-gray-50">
                                Condition End</th>

                            <th class="px-6 py-3 text-sm text-left text-gray-500 border-b
                               border-gray-200 bg-gray-50"
                                colspan="2">
```

```
44                          Action</th>
45                  </tr>
46                  </thead>
47
48                  <tbody class="bg-white">
49                  @foreach ($schedule as $entry)
50                      @if($entry->id != 1)
51                          <tr>
52                              <td class="px-6 py-4 whitespace-no-wrap border-b border-
                                    gray-200"
53                                  style="display: none">
54                                  <div class="flex items-center">
55                                      {{ $entry->id }}
56                                  </div>
57
58                              </td>
59
60                              <td class="px-6 py-4 whitespace-no-wrap border-b border-
                                    gray-200">
61                                  <div class="text-sm leading-5 text-gray-900">
62                                      @foreach($conditions as $condition)
63                                          @if($condition->id == $entry->condition_id)
64                                              {{$condition->name}}
65                                          @endif
66                                      @endforeach
67                                  </div>
68                              </td>
69
70                              <td class="px-6 py-4 whitespace-no-wrap border-b border-
                                    gray-200">
71                                  <div class="text-sm leading-5 text-gray-900">
72                                      {{ $entry->condition_start }}
73                                  </div>
74                              </td>
75
76                              <td class="px-6 py-4 whitespace-no-wrap border-b border-
                                    gray-200">
77                                  {{ $entry->condition_end }}
78                              </td>
79
80                              <td
81                                  class="text-sm font-medium leading-5 text-center
                                        whitespace-no-wrap border-b border-gray-200 ">
82                                  <a href="{{ route('schedule.edit', $entry->id) }}"
83                                     class="text-indigo-600 hover:text-indigo-900">
84                                      <svg xmlns="http://www.w3.org/2000/svg" class="w
                                            -6 h-6" fill="none"
85                                           viewBox="0 0 24 24" stroke="currentColor">
86                                          <path stroke-linecap="round" stroke-linejoin=
                                                "round" stroke-width="2"
87                                              d="M11 5H6a2 2 0 00-2 2v11a2 2 0 002 2
                                                    h11a2 2 0 002-2v-5m-1.414-9.414a2 2
                                                     0 112.828 2.828L11.828 15H9v-2.828
                                                    l8.586-8.586z" />
```

```
88                                          </svg>
89                                      </a>
90
91                                  </td>
92                                  <td class="text-sm font-medium leading-5 whitespace-no-
                                        wrap border-b border-gray-200 ">
93                                      <form action="{{ route('schedule.destroy',$entry->id)
                                            }}" method="POST" onsubmit="return confirm('{{
                                            trans('are You Sure ? ') }}');">
94
95                                          <input type="hidden" name="_method" value="DELETE
                                                ">
96                                          <input type="hidden" name="_token" value="{{
                                                csrf_token() }}">
97                                          <button type="submit" class="flex items-center">
98                                              <svg xmlns="http://www.w3.org/2000/svg"
99                                                   class="w-6 h-6 text-red-600 hover:text-
                                                       red-800 cursor-pointer" fill="none"
100                                                  viewBox="0 0 24 24" stroke="currentColor
                                                      ">
101                                                 <path stroke-linecap="round" stroke-
                                                        linejoin="round" stroke-width="2"
102                                                       d="M19 7l-.867 12.142A2 2 0
                                                           0116.138 21H7.862a2 2 0
                                                           01-1.995-1.858L5 7m5 4v6m4-6
                                                           v6m1-10V4a1 1 0 00-1-1h-4a1 1 0
                                                            00-1 1v3M4 7h16" />
103                                             </svg>
104                                         </button>
105                                     </form>
106                                 </td>
107                             </tr>
108                         @endif
109                     @endforeach
110                     </tbody>
111                 </table>
112             </div>
113         </div>
114     </div>
115 </div>
116 </x-app-layout>
```

A.3 Inhalt des Datenträgers

Der dieser Arbeit beigelegte Datenträger beinhaltet zusätzliche Materialen. Neben der Arbeit selbst im Portable Document Format (PDF) befinden sich auch die Sources der Implementierungen.

`./rpi-growfridge/`
: Alle Daten zur Ausführung des *Growfridges*

`./Thesis_Benjamin_Ochocki_2048974.pdf`
: PDF Version dieser Arbeit