Smart Growbox

Jonas Valentin Rose

2020

Inhaltsverzeichnis

1 Einleitung **3**
1.1 Idee . . . 3

2 Ziel und Abgrenzung **3**
2.1 Das Konstrukt . . . 3
2.2 Schriftlicher Teil . . . 3

3 Motivation **4**

4 An die Arbeit **5**
4.1 Wachstumsbedingungen . . . 5
4.1.1 Wasser . . . 5
4.1.2 Licht . . . 5
4.1.3 Luft . . . 5
4.1.4 Nährstoffe . . . 5
4.2 Wie weiter? . . . 6
4.3 Hornbach, Cannabis-Shops und mein Keller . . . 6
4.4 Trockenes Wasser . . . 7
4.5 Belüftung.. . . . 7
4.6 Es soll hell sein . . . 8
4.7 Das Hirn! . . . 9
4.8 Meiers Tageshit: Universaldünger, leicht säuerlich. . . . 9
4.9 Löcher im Boden . . . 9

5 Arbeitsprozess **11**
5.1 Recherche . . . 11
5.2 Löten, testen, löten . . . 11
5.2.1 Meine Hassliebe zu PWM . . . 12
5.2.2 Relays als nette Abwechslung . . . 13
5.3 Oszilloskopisches Wunder . . . 13
5.4 Code . . . 13
5.4.1 C++/Arduino . . . 13
5.4.2 Python . . . 14
5.5 GitHub, the beast . . . 16
5.6 Bash . . . 17

6 Technische Hintergründe **17**
6.1 Ganz von vorn. . . . 17
6.2 Luft rein, Luft raus. . . . 17
6.3 Klick, klick, klick . . . 17

7 Perspektiven **18**
7.1 Technische Einsichten . . . 18
7.2 Mögliche Weltverbesserung . . . 18
7.3 Warum *genau* baue ich das eigentlich? . . . 18

8 Fazit **18**

9 Danke sagen **18**

10 SWAP **18**

Glossar **18**

1 Einleitung

Dies ist der schriftliche Teil meiner Maturarbeit, bei welcher ich mich mit der Konstruktion einer möglichst smarten Growbox auseinandersetze. Ich stelle hier so viel möglich und so wenig wie nötig alle meine Gedanken vor, welche mich während dieser Zeit motivierten oder plagten. Angefangen bei der Idee selbst, über die pflanzlichen Anforderungen, weiter zu den drei Kernelementen des Konstrukts bis hin zum esten Webserver, ich verhoffe mir, Ihnen einen möglichst nahen Einblick in meine letzten paar Monate zu geben.

1.1 Idee

Mögen Sie Tomaten? Ich tue es, im Sommer 2019 habe ich meine eigenen gezüchtet, direkt auf meinem Balkon. Sie stellten sich als recht gut gereift heraus. Allerdings plagte es mich, während des vierwöchigen Korsika-Urlaubs die Verantwortung bezüglich der Bewässerung abzugeben. Nicht dass ich gerne selbst Pflanzen bewässere, klar es hat durchaus etwas magisches an sich, doch nein, ich in faul. Magisch faul. Faul lässt es sich am besten clever leben, ergo: Es muss eine richtige Lösung her. Ich will in den Urlaub fahren können, ohne die Verantwortung meiner Stecklinge abgeben zu müssen, aber ich will mir auch nicht die ganze Zeit Sorgen machen müssen. Ich will sozusagen den *Föifer* und *s'Weggli*. Zur Motivation komme ich erst im nächsten Kapitel, doch lassen Sie mich Ihnen eins versichern: Das Internet kann sehr inspirierend wirken. Lange Rede, kurzer Sinn - Ich will Tomaten essen, ohne mich um Sie kümmern zu müssen. Sie sollen wachsen, aber ohne meine Bemühungen. Klar doch, eine Maschine muss her! Eine solche, mit der man dem Tomätchen die perfekten Wachstumsbedingunen geben kann, ohne sich auch nur darum zu bemühen.
Das mit der Bemühung ist so ein Thema, denn zuerst will die Maschine gebaut werden. Ich stelle sie mir so vor: Eine Growbox in meinem Zimmer mit elektronischer Beleuchtung, automatischer Bewässerung und einem sich selbstregulierenden Belüftungssystem. Klingt gewagt, aber nicht unmachbar. Ich beweise es Ihnen.

2 Ziel und Abgrenzung

2.1 Das Konstrukt

Im praktischen Teil konstruiere ich eine Maschine, welche mir Esspflanzen züchtet. Sie soll die drei Hauptaufgaben übernehmen, welche ich als Belüftung, Beleuchtung und Bewässerung identifizere. Sie soll selbst gebaut sein, d.h. keine fremde Regelungstechnik wie Zeitstromschalter oder Thermostaten. Die finanzierung erfolgt selbst. Ich werde die Bauteile selbst kaufen/zusammenbauen/konstruieren/programmieren und das dazugehörige Wissen beziehe ich grösstenteils aus dem Internet. Ich mache auch von der Erfahrung meines Vaters gebrauch, er steht mir in besonders abstrakten Problemstellungen bei.

2.2 Schriftlicher Teil

Hier im schriftlichen Teil versuche ich, wie in der Einleitung bereits erwähnt, dem Leser einen möglichst direkten Einblick in die Lösungsfindung der Teilprojekte zu bieten. Was ich nicht vorhabe, ist es, Ihnen Python, C++ oder Löten beizubringen. Sie werden hin und wieder auf Codesnippets treffen, bei denen es um die Funktion des jeweiligen Ausschnitts geht. Aus all diesen Gründen ist es von Vorteil, wenn Sie schon etwas mit Technik zu tun hatten. Nebst dem

Arbeitsprozess schildere ich im nächsten Kapitel auch meine Motivation, dieses Projekt nicht *nur* als Maturarbeit zu verfolgen. Das Kapitel *Arbeitsprozess* ist als Hauptteil des Kommentars gedacht. Anschliessend werde Ich mit möglichst wenigen aber klaren Sätzen das *Konstrukt* selbst noch kurz erklären, wie es funktioniert, wo die Tücken liegen und worauf ich besonders Stolz bin. Im Kapitel *Perspektiven* gehe ich auf alles ein, was mich rund um die Arbeit beschäftigte. Zu guter Letzt ziehe ich ein Fazit und bedanke mich bei allen Menschen, welche mir geholfen haben, mein Vorhaben in die Realität umzusetzen.

3 Motivation

Ich verbringe (zu) viel Zeit auf YouTube. Eine mich inspirierende Gattung sind die Art Videos, welche sich mit Offgrid-Versorgungssystemen beschäftigen, also der Nahrungs-/Energieversorgung eines Lebensraums ohne Teil des Hauptnahrungs-/energienetzes zu sein. Dieses Autarksein fasziniert mich. Des weiteren stiess ich vor etwa 10 Monaten auf den Kanal von *Jeb Gardener*. Ich fasse die Gemeinsamkeiten seiner Videos im folgenden Satz zusammen: Verrückter Gärtner züchtet mit komplett unkonventionellen und (laut einigen Stimmen seitens des *traditionellen Gärtnermilieus*) ethisch kritischen Methoden Pflanzen. Seine Bewässerungsmethoden sind überaus selten traditioneller Art, bedeutet, dass er fast immer auf Erde verzichtet. Er stellt die Pflanzen direkt ins Wasser. Beleuchtet wird künstlich, mit pinken LEDs und die Nährstoffe sind synthetisiert und werden einfach ins Wasser geworfen, wo sie sich auflösen. Konventionalität interessiert ihn nicht. Dazu kommt noch, und das ist fast das Beste, dass seine Videos durch die musikalische Begleitung von epischer Orchestermusik eine fast zerreissende Spannung erreichen. Stellen Sie sich einen durchgeknallten Gärtner vor, der von einem Streichorchester begleitet, Wurzeln filmt. Seine Unverdrossenheit im Umgang mit Anbaumethoden ist sehr ansteckend. *Whatever gets the job done.*
Segelvideos sind seit ein paar Jahren ebenfalls sehr *in*, sie verdanken ihren Erfolg wohl den Lebensumständen, in denen alle Landratten leben, welche gerne zur See fahren würden: Der landrattige YouTube-Zuschauer sieht durch seinen Bildschirm den unendlichen Horizont, den Sonnenuntergang und versucht sich den Wind in den Haaren vorzustellen, der Segler lebt diesen Moment und fängt es mit der Kamera ein. Es ist ein Zusammenspiel und ich bin ein Teil davon. Die Freiheit welche man durch das Segeln erlangt wirkt auf mich ansteckend. Was aber hat das mit meiner Maturarbeit zu tun? Hier der Link: Bei einer Ozeanüberquerung hat man nach ein paar Wochen keine frischen Lebensmittel mehr, die Rede ist von Früchten, frischen Kräutern und Gemüse. Was aber, wenn man sich seine Lieblingstomaten einfach selbst auf dem Boot züchtet? In einer Miniaturversion meines Konstrukts, mit Solarpanels als Energieversorgung. Wäre das nicht einfach genial? Wenn ich das als Startup-Projekt gut verkaufen kann, lässt es sich markttechnisch sogar verkaufen. Ich könnte DIY-Anleitungen machen, sodass mehr Segler mehr frische Sachen essen können. Durch *Der verschenkte Sieg* von *Bernard Moitessier* habe ich gelernt, wie sehr frisches Gemüse oder Früchte die Lebensqualität auf einer Pazifiküberquerung heben können. Die oben genannten Erlebnisse motivieren mich, das *Konstrukt* nicht nur als Maturprojekt sondern als Lebensprojekt zu sehen. Wer weiss, vielleicht studiere ich etwas in diese Richtung.

4 An die Arbeit

4.1 Wachstumsbedingungen

Das Hauptziel in dieser Arbeit ist die Lebenserhaltung von Pflanzen durch eine künstliche Umgebung. Pflanzen bergen gewisse Ansprüche und um die geht es jetzt kurz. Ich erkläre den Anspruch und weswegen er besteht - mit einer für die Arbeit tragbaren Tiefe.

4.1.1 Wasser

Pflanzen brauchen Wasser für den Transport von Nährstoffen, für die Photosynthese und um den Zelldruck und die damit verbundene Stabilität zu erhalten - unbewässerte Pflanzen lassen bekanntlich ihre Köpfchen hängen.

4.1.2 Licht

Licht ist die Energieform, welche sich die Pflanze bei der Photosynthese zunutze macht, um energiearme Baustoffe in Energiereiche zu verwandeln. Diese werden für den Bau und die Energiespeicherung verwendet.

4.1.3 Luft

Luft, auch wenn eigentlich überall vorhanden, muss einen genügend grossen CO_2-Anteil bergen, um der Pflanze die Photosynthese zu ermöglichen. Sie wird wie das Licht grösstenteils über die Blätter aufgenommen.

4.1.4 Nährstoffe

Nebst der selbst hergestellten Glucose verwendet die Pflanze auch noch andere Bausteine um sich Struktur zu geben und den Stoffwechsel zu betreiben. Diese können nicht aus der Luft gefischt werden, sondern werden in gelöster Form von den Wurzeln aufgenommen. Dazu zählen Elemente wie Phosphor, Kalium und Stickstoff.

4.2 Wie weiter?

Ich kenne nun die Ansprüche meiner Pflanzen, aber wie baue ich eine Maschine, die denen Genüge tut? Dies ist der Kern der Arbeit, alles dreht sich darum. In diesem Kapitel stelle ich vor, wie ich mir die Konstruktion vorgestellt habe, während ich im Internet recherchiert habe. Zuerst brauche ich Bauteile, doch von wo nu- Ach ja genau, das Internet, es ist voller Möglichkeit, Wissen und unnötigem Zeugs, wussten Sie, dass es eine Bewässerungsmöglichkeit gibt, welche komplett auf Substrate wie Erde oder Tonpellets verzichtet? Nennt sich Hydroponik und ist recht simpel: Eimer nehmen, Wasser rein, Nährstoffe rein, Pflanze rein. Alles ganz fein. Doch es geht besser. Mehr dazu im nächsten Kapitel ;)
Materialsuche ist ein anstrengendes Unterfangen, ich bin aber nochmal glimpflich davongekommen, denn ich fand eine gute Menge Material in unserer Kellerwerkstatt. Mein Vater ist ein *Techie*, der gerne Elektroteile sammelt, zu meinem Glück.

4.3 Hornbach, Cannabis-Shops und mein Keller

Hier sind die essenziellen Bauteile des *Konstrukts*.

Grow Box Die Hülle stellt meine Grow Box dar, sie besteht aus einem Plastikgewebe, das an der Innenseite mit Mylar beschichtet ist - es reflektiert zu einem hohen Bestandteil Licht. Sie gibt Form und ermöglicht es den Pflanzen, ein eigenes kleines Klima zu errichten, eine eigene, kleine Ökosphäre sozusagen.

LEDs Ich beleuchte meine Pflanzen künstlich mit zwei 50 Watt LEDs. Beide sind auf die für die Pflanze essentiellen Lichtspektren eingestellt, was sich für das menschliche Auge als ein pinkes Helligkeitswunder herausstellt. Es schmerzt, ich hätte besser nicht reingeschaut..

Fogger Ein Wundergerät, es macht aus flüssigem Wasser trockenes Wasser in Nebelform. Damit kann ich die Wurzeln über die Luft mit Wasser und Nährstoffen versorgen, ich habe es vorhin schon angedeutet, mehr dazu gleich.

Wasserbecken Hornbach hat ein beachtliches Sortiment an *Zementmischeimern*, einen davon habe ich erworben.

Sensoren Um zu wissen, wie fest ich Lüften muss, brauche ich Sensoren. Zu meinem Erfreunis sind sie äusserst günstig.

Microcontroller Die meisten Microcontroller, namentlich Arduinos und einen Raspberry Pi hatte ich dank meinem *Geekvater* schon zuhause.

Sonstige Elektronik Kleinere Bauteile für die verschiedenen Schaltungen hat ebenfalls die Kellerwerkstatt meines Vaters beigesteuert. Darunter befinden sich vor allem Drähte, Kabel, Widerstände, Transistoren und Dioden.

Lüfter Aus der edlen CPU-Lüftersammlung meines Vaters konnte ich vier Gleichstrom-Lüfter ergattern.

Pflanze Das ganze Projekt würde ohne Pflanzen nicht funktionieren. Beim *Gartenmeier Center* bescherte ich mich mit Tomatenstecklingen, zweimal, denn das Projekt hat länger gedauert als angenommen, wodurch der erste Satz Stecklinge leider dran glauben musste.

Nährstofflösung/Dünger Mit einer Universalnährstofflösung (die sogar den richtigen pH-Wert mit sich bringt) kann ich nichts falsch machen. Auch darauf komme ich nochmal zu sprechen.

4.4 Trockenes Wasser

Durch Jeb Gardner wusste ich von der Hydroponik schon vorher und durch Zufall stiess ich auf Fogponics, was sich als recht eleganten Ansatz herausstellt. Man zählt Fog-, respektive Aeroponik als eine Unterkategorie der Hydroponik. Konkret geht es dabei um die Bewässerung der Pflanze, während sich ihre Wurzeln *in der Luft befinden*, also weder im Wasser, noch in einem Substrat. Düsen werden gerne in grossen, industriellen Gewächshäusern verwendet, doch nach endloser Recherche musste ich mir eingestehen, dass Düsenn einfach keine Lösung sind, ich bekam es einfach nicht hin, in absehbarer Zeit die richtigen Düsen und das Röhrenset dafür zu bestellen. Ich erweiterte also meinen Horzizont. Fogponics sind genial, man nimmt den Fogger (ein recht schweres, wasserfestes Metalding mit fünf Porzellanscheiben), setzt ihn ins Wasser und schaltet ein. Die Porzellanscheiben vibrieren mit 1.7MHz, also etwa etwa alle 0.6 Microsekunden einmal, und spicken somit feinste Wassertröpchen in die Luft - Ein Nebel entsteht. Wenn im Wasser dann noch die essentiellen Nährstoffe gelöst sind, hat man einen Nährstoffnebel! Die Tröpfchen sind ca. ein Zweihundertstel Millimeter, was es ihnen ermöglicht, direkt in die Wurzeln einzudringen, ohne dabei an der Wurzelwand zu kondensieren. Eine so direkte Nährstoffaufnahme ist *giga-effizient* und nur bei solch kleinen Tröpfchen möglich.
Kleiner Seitenfakt: Beim Einatmen von solch kleinen Tröpfchen muss man Husten, denn sie ähneln mehr einer Art Staub als Wasser. Das ist wahrscheinlich auf ihre eingeschränkte Kondensfähigkeit zurückzuführen, was sie für uns *nass* machen würde. Diese Technologie wurde durch die NASA markttauglich gemacht und wird deshalb heutzutage auch in der Raumfahrt eingesetzt.Ich verwende es nun um Pflanzen *intrawurzulös* mit trockenem Wasser und Nährstoffen zu versorgen, auch gut.

4.5 Belüftung..

Die Lüfter waren ebenfalls ein lustiges (*trockenes, kaltes Lachen...*) Teilprojekt:
Um ein Lüftungssystem zu bauen, braucht man offensichtlich einen Lüfter, am besten gleich mehrere. Es gilt zwischen EC- und DC-Lüftern zu unterscheiden, wobei EC-Lüfter generell leiser und energieeffizienter sind. Naheliegenderweise hätte ich gerne einen Lüfter der EC-Gattung, doch da mein Vater wie schon erwähnt ein leidenschaftlicher Sammler jeglichen Elektroschrotts ist, konnte ich mich einfach im Keller bedienen. Um es jetzt kurz zu fassen: Bei der Belüftungskonstruktion habe ich meine Seele verloren, es ist das Teilprojekt mit dem meisten Aufwand. Das mit der Seele ist sinnbildlich gemeint, versteht sich. Im Nachhinein habe ich mich wirklich gefragt, ob es dieser Aufwand nicht die 120Fr. wert gewesen wäre, was so ein EC-Lüfterchen kostet. Ich erläutere das mal: Einen DC-Lüfter (Gleichstrom) runterzuregeln, so dass er nicht volle Pulle lüftet, ist eine Wissenschaft für sich, ich bin recht gut geworden, doch zu welchem Preis? Einfach die Spannung runterzudrehen geht nicht wirklich, denn sie ist nicht proportional zur Umdrehungszahl. Was bleibt mir also? Ganz einfach - *Pulsweitenmodulation.* Hier das Prinzip: Man versorgt den Lüfter entweder mit 12V oder mit 0V, aus dem zeitliche Verhältnis zwischen diesen beiden Zuständen erfolgt die effektive Lüfterleistung. Ist der *Puls* also 75% eines ganzen Zyklus', läuft der Lüfter auch mit 75%, klingt einfach, ist es aber nicht. Die Krux liegt in der Frequenz dieser Zyklen, es macht nämlich einen Unterschied, ob man ihn für 7.5 mit 12V versorgt, dann 2.5s Pause macht oder ob diese Zyklen mit einer Frequenz von

40'000Hertz aufgetischt werden. Beim ersten Beispiel stockt der Lüfter und beim Zweiten wird der Saft so unglaublich schnell an und abgestellt, dass der Motor gar nicht mehr zwischen An und Aus unterscheiden kann. Voilà, Pulweitenmodulation.
Die Theorie an sich ist einfach, denn hier kommt die technische Umsetzung: Die von mir verwendeten Microcontroller besitzen eine PWM-Funktion, jedoch hat diese eine Maximalfrequenz von 1kHz, also 1000Hertz und das ist zu langsam, denn damit *kann* der DC-Motor noch zwischen An und Aus unterscheiden und es kommt zu hörbaren Vibrationen, was einem nach 2min recht auf den Wecker schlägt. Es muss demnach eine andere Lösung her. Der Arduino (der Microcontroller) ist rein technisch gesehen dafür ausgerüstet, höhere Frequenzen zu takten, ein gutes Beispiel dafür ist der Prozessor, der, wenn er auf 1kHz laufen würde, *schnarchend* langsam wäre. Dem ist natürlich nicht so. Und auch wenn der eingebaute ATmega32U4 Prozessor mit seinen 16MHz das weitaus schnellste Glied auf der Platine ist, weilen die internen Timer-Register nicht viel hinten. Die C++-Bibliothek *TimerOne.h* beherbergt die nötigen Werkzeuge. Man kann sich einen eigenen Timer stellen, sagen wir auf 25 Mikrosekunden, und wenn immer dieser Timer abläuft, schalten wir entweder an oder aus. So ungefähr lautet das Prinzip, effektiv existiert aber schon eine eigens dafür entwickelte Funktion. Man muss nur den Pin angeben, die Frequenz vorher eingestellt haben und sagen, welches Verhältnis man wünscht. Perfekt für meine Ansprüche.
Soviel zum Softwareteil, der Hardwareteil ist auch eine rechte Herausforderung. Ich brauche für jeweils vier Lüfter eine Schaltung, welche den Strom auf einer so hohen Frequenz an- und ausschalten kann. Also gut, Platine her, Lötkolben und Oszilloskop raus und runter in die Werkstatt. Ich habe für diese Schaltung etwa einen Monat gebraucht, natürlich war ich nicht jeden Abend dran, doch die alleinige Dauer, welche ich vor dem Oszilloskop verbracht habe, um die richtigen Verhältnisse zu sehen, hat mir die Seele wahrlich etwas betäubt. Im Kapitel Perspektiven gehe ich näher auf Gelerntes ein, deshalb verrate ich hier einfach, dass am Ende der sogenannte *PWM-Controller* geboren war.

4.6 Es soll hell sein

Die Belichtung ist um unser allen Dankbarkeit um einfaches simpler. Nichtsdestotrotz muss ich kurz ausholen:
Um sowohl die Belichtung, als auch die Bewässerung zu steuern, brauche ich eine Schaltung, welche mal an und mal aus geht - und das bitte in einem von mir vorgegebenen Takt. Swoosh, geboren ist der *Relay-Controller*! Auf den Bauprozess dieses Babys wird später eingegangen, Sie sollen einfach wissen, dass es recht gut funktioniert. Die ist Belichtung echt keine Hexerei, die LEDs gehören einfach eingesteckt und der Relay-Controller vorprogrammiert und es läuft. Ein solcher LED-Schweinwerfer kostet 65Fr und ist rein von der Leistung gesehen (50W) für meine zu belichtende Grundfläche von 0.36m^2 hell genug. Warum habe ich also zwei davon? Erstens sehen diese LEDs total schön aus - sie haben auch so ein cooles Gehäuse, sollten Sie mal sehen, und andererseits hat mich - ob relevant oder nicht - die Frage beschäftigt, wie lang diese Babies denn so halten. Das Internet gibt keine genaue Angabe, deshalb habe ich mir, so genial wie ich bin, überlegt, dass ich, wenn ich den Arbeitsaufwand auf zwei verschiedene Lampen aufteile, die Haltbarkeit von diesem Teil der Machine wohl doppelt so hoch sein muss. Nebenbei ergibt sich die Möglichtkeit, beide aufs Mal laufen zu lassen, wovon ich mir einen Wachstumsboost der Pflanzen versprechen, wir werden sehen. Und falls eine den Geist aufgibt, habe ich Ersatz da. Ich habe in der Einleitung schon von Sorgenfreiheit gesprochen, Ersatzteile sind ein Aspekt davon.

4.7 Das Hirn!

Irgendjemand, oder besser gesagt irgend*was* muss diesen gutmütigen Controllern eine Richtung weisen, ihnen sagen, was sie überhaupt zu tun haben. An diesem Punkt würde ich mich gerne bei Herrn Upton, dem Gründer der Raspberrypi-Foundation bedanken, ohne ihn wäre dieses ganze Vorhaben nämlich recht *hirnlos*, im wahrsten Sinne des Wortes: Ich verwende einen RaspberryPi als Zentrum für alle Steuerprozesse. Der Raspi (umgangssprachlich auch *Pi*) beherbergt alle Funktionen welche man von einem normalen Computer erwarten würde, nur kleiner, denn er passt Problemlos in die Hosentasche. Die Leistung ist logischerweise geriner als bei Computer mit normalen Dimensionen, doch Rechenleistung steht bei vielen Projekten erst an zweiter Stelle. Ich z.B. brauche einfach ein kleines Linux-Compüterchen und genau das ist der Pi. Mit Linux bin ich gleichermassen aufgewachsen wie mit einem Fahrrad, es war einfach da. Deshalb ist es für mich kein Problem, dieses Schätzchen auf meine Bedürfnisse anzupassen. Da es in diesem Kapitel generell nur um die erste Herangehensweise geht, fasse ich mich kurz, denn der Pi kommt zeitlich erst recht am Schluss ins Projekt: Er steuert die Sachen, welche andere Sachen steuern und nimmt Inputs vom Benutzer - das bin Ich - entgegen.

4.8 Meiers Tageshit: Universaldünger, leicht säuerlich.

Dieses Unterkapitel mag Sie etwas überraschen, doch es sei angemerkt, dass sich mir dieses Thema recht lange zur Last machte. Die Frage, *wie* und mit welcher Nährstofflösung (eng. *Nutrientsolution* oder einfach *Solution*) ich meine Pflänzchen füttere, hat mich lange geplagt. Eine einfache Antwort darauf lässt sich im Internet **nicht** finden. Auch Jeb hat darauf keine wirklich einfache Antwort, denn er misst die Menge an Kaliumsubstraten mit einer Küchenwage und schmeisst sie dann in den Wasseareimer. Da ich ein Laie in Sachen Substratmixtur bin, brauche ich also eine simplere Lösung, welche mir das Internet nicht gab. Bei einem meiner Besuche in Meiers Gartenmeiercenter in Dürnten stiess ich auf *the man himself*, Herr Meier.
Wahrlich ein Engel; Er stand hinter dem Kundenberatungsthresen und als ich endlich dran war, ging es keine halbe Minute und er wusste *welche Solution* ich *wo* zu suchen hatte. Gesagt, getan, gefunden, in den Händen hielt ich eine grüne 1L Flasche mit Wunderzeugs. Sie enthält, soweit mich meine eigenen Sinne und Kenntnisse nicht trügen, um weitaus mehr Nährstoffe, als alle anderen Dünger aus unserem Gartenschuppen. Das Zeug mit Leitungswasservermischt soll angeblich auch direkt denn perfekten pH-Werte haben, 5.5, also leicht auf der sauren Seite. Zu so etwas sage ich nicht Nein.

4.9 Löcher im Boden

Die Pflanzen finden Halt in sogenanannten Steinwolleblöckchen, diese widerrum stehen in kleinen Netzbechern/Netcups und *diese* hängen durch ein Brett durch, sodass die Würzelchen im Nährnebel hängen. Das Brett ist aus beschichtetem Holz. Ich habe es zusammen mit dem Zementeimer im Hornbach gefunden. Zum Leid jeder Nase stank diese Brett so wahnsinnig bestialisch nach dieser chemischen Beschichtung, dass ich mehr als einmal keinen Appetit verspürte, nachdem ich die Growbox aufmachte, wo ich alles Material verstaute. Es stank, aber es nicht unhygienisch wie gewisse andere Sachen, doch es stank. Eines Tages wusste ich, dass ich Löcher in das Brett machen musste. Logisch, das ist ja der Sinn des Brettes. Also ab in den Keller und - Hmm, wie macht ohne es je einmal gemacht zu haben 8cm breite Löcher in chemisch verstärktes Holz? Die Werkstatt offerierte mir einen Bohrer und ein Löcherbohrset, eines mit einer dieser Scheiben, vo man verschiedene, kreisrunde Sägeblätter einstecken kann. Das Problem dabei war jedoch die vorhandenen Durchmesser, 6cm war das grösstmögliche. Mir fehlten

also 2cm. Haben sie schon einmal von einer *Fräse* gehört? Nun, mein Vater hat so ein Ding und ich habe es gefunden. Dieses Stück Maschine funktioniert im Prinzip wie ein normaler Bohrer, aber seitwärts. Anstatt nach unten zu bohren, ist der Fräsenkopf an der Seite mit Klingen bewaffnet, welche sich sterbenshungrig ins Holz fressen. So fest, dass man das ganze Konstrukt leicht verlieren kann, wenn sich die Fräse erst mal festgebissen hat, gibt es kein entkommen. Das ganze Gerät, welches man an zwei (!) massiven Haltern hält, rennt praktisch durchs Holz und es stäubt, oh wie es stäubt. Für die Atemmaske war ich dankbar. Diese Fräse ist ein Erlebnis, das sieht man an der tiefen Kerbe in einem der Löcher, der Netzbecher fällt fast raus.. Den Fräsenabend werde ich nicht so schnell vergessen, ich bin noch nie mit so viel Sägestaub aus dem Keller gekommen, meine Augen enthielten mehr Sägemehl als Tränenflüssigkeit.

5 Arbeitsprozess

Wie genau bin ich wann vorgegangen? Auf diese Frage gibt es viele Antworten, hier sind die wichtigsten. Ich unterteile den Arbeitsprozess in Kategorien, welche widerrum Beispiele enthalten.

5.1 Recherche

Am Anfang jeder Maturarbeit steht die Recherche. Ich habe, vom Interne ja schon inspiriert, vorallem im Web recherchiert. Mein Wissen, sei es Bewässerung, Optimaltemperatur, Nährstoffe, Belichtungsmethoden, Bewässerungssysteme oder einfach Pflanzen selbst, kam zu einem *sehr* grossen Anteil von Cannabis-Blogs, -shops und -foren. Jeder, der meinen Suchverlauf inspiziert, wird direkt glauben, dass ich vor habe, ganz gross ins Business einzusteigen. Ich kann nicht verleugnen, dass das *Konstrukt* cannabis-bezüglich durchaus Potenzial hätte, aber ich segle lieber satt als high. Deshalb spezialisiere ich es bewusst *nicht* auf sensible Marihuana-Pflänzchen, denn es sei auch angemerkt, dass diese Praxis eine Wissenschaft für sich ist.
Recherche fand eigentlich konstant statt, wenn auch nicht immer aufgrund der gleichen Motivation: Am Anfang wollte ich wissen, wie ich so eine Growbox baue, die Bauteile standen also im Zentrum. Am Ende verbrachte ich 80% meiner Web-Zeit auf *stackoverflow.com*, wem diese Webseite ein Name ist, schmunzelt jetzt womöglich :) - Wem nicht, hier ist die Erklärung: Wenn ein Programmierer auf eine Fehlermeldung stösst, welche er nicht versteht, gibt es nur eine Website, welche bei der darauffolgenden Google-Suche *praktisch immer* zuoberst erscheint, es ist *stackoverflow.com*. In der Humorszene (richtig, ich meine Memes) der Coding-Communities hat sich dieser Witz schon recht gefestigt. Hin und wieder musste ich auch am Ende nicht-programmiertechnische Sachen googeln, aber das sind wie gesagt nur etwa ein fünftel, wenn nicht *noch* weniger. Recherche, so trist es auch klingen mag, ist anstrengend und nur das Mittel zum Zweck, jedenfalls für mich. Es inspiriert vielleicht, aber ehrlich gesagt ist das auch schon alles - Mit der Suchleiste baue ich keinen Relay-Controller. Dafür brauche ich Kabel, Steckdosen, Relays und einen Lötkolben - und um den geht es jetzt.

5.2 Löten, testen, löten

Löten ist, wenn man mit einem heissen Metalkolben Lötzinn, das ist ein Zinndraht mit Flussmittel in der Mitte, zwischen zwei Kontaktstellen verteilt, sodass anschliessen Strom durchfliessen kann. Ich habe zwei eigene Schaltungen gebaut, den PWM- und den Relay-Controller. Beide besitzen als Grundbaustein eine Platine (Eine Plastikscheibe), auf welche ich die Stromleitungen drauflötete. Das an sich ist super cool; Sich selbst einen Schaltplan zu zeichnen und diesen dann in die Realität umzusetzen macht einfach Spass. Mein Vater ist ehemaliger Elektroniker und ich habe mich wie ein Lehrlich gefühlt, als er wie ein Prüfer mit dem Finger versucht hat, meine Lötstellen zu durchtrennen, wenn sie hielten war er zufrieden, wenn sie nachgaben und aufsprangen, bedeutete das, dass sie auch keinen Strom durchlassen würden - oder nur begrenzt. Es war die härteste aber effizienteste Art, meine Kreationen zu testen. Er vertritt generell die Arbeitsmoral, etwas *einmal, dann aber richtig* zu machen. Und das hat er mir beigebracht, es gibt keine patzige Lötstelle in keinem Controller, jedes Tröpfchen Zinn sitzt. Der Prozess bestand meistens darin, dass ich im Keller sass, lötete, testete, wenns gut kam, den Fehler selbst fand, nochmal neu lötete und es ihm dann zeigte. Oft musste er mir aber helfen. Ich muss zugeben, die Souveränität und die Art, in welcher ein erfahrener Elektroniker an eine nicht sauber funktionierende Schaltung heran geht, ist beeindruckend. *Ist die Masse überall gleich? Ist die Diode verkehrt drin? Schau mal diese Lötstelle hier.* Das alles sind oft

gehörte Sätze und wenn ich dann mit einer funktionierenden Schaltung hochschritt um sie ihm zu zeigen, war das Gefühl ein besonders schönes. Das Lob eines stolzen Vaters. Ich kann mich wirklich glücklich schätzen, einen Pro im Haus zu haben.
Was auch recht half, war die ungeheuer grosse Sammlung an Elektroteilen, welche er noch aus alten Tagen hat. Er hat alle Kisten sortiert, ich konnte ihn einfach fragen, wo die Transistoren sind und er zeigte mir den richtigen. Von Transistoren habe ich vor diesem Projekt zwar schon gewusst, allerdings blieb mir ihre Bedeutung für die Menschheit verborgen. Das änderte sich eines Tages, recht am Anfang, als er mir gefühlt den ganzen Keller erklärte, angefangen von Dioden, über Transistoren, bis hin zu Spulen. Nicht dass ich jetzt noch besonders viel weiss, dass muss ich ehrlich gestehen, doch es ist superspannend und recht nützlich zu wissen. Ich weiss jetzt zum Beispiel, wie der Transport via Hochspannungsleitungen funktioniert, respektive *warum* es Hochspannung ist - weniger Verluste, so einfach ist das.
Ich habe noch etwas anderes gelernt, etwas schmerzhaftes. Aber ganz von vorn. Ich sass eines Nachmittags im Keller, lötete und testete den PWM-Controller und brauchte eine Stromversorgung für die 12V-Lüfter, also brauchte ich ein Netzteil. Netzteile sind oft schwer, was bedeutet, dass es welche sind, welche mit Spulen funktionieren. Ich hatte ein solches rumliegen und prüfte die Ausgangspannung, 12V, passte also. Einmal eingesteckt wollte es anschalten, es hatte einen Schalter dran, welcher jedoch nicht *im* Gehäuse, sondern ausserhalb war. Das Netzteil hatte nämlich kein Gehäuse, es war einfach ein Paket Spulen, auf jeder Seite ein Kabel, eins mit 230V und das andere mit 12V und dazu kam der Schalter. Nun ist es so, dass sich der Schalter auf der 230V-Seite befand, also da, wo es richtig weh tut - und das tat es. Der Schalter ist nicht ganz abisoliert worden, beide 230V-Pins waren offen, ich wollte den Schalter mit einer Hand anmachen, musste ihn also gleichzeitig stabiliseren, damit ich draufdrücken konnte und ich berührte dadurch direkt die beiden Pins. Es war nicht lebensgefährlich, der Strom floss nur durch meinen Finder, aber es zuckte trotzdem recht schnell. Ich bin noch nie so schnell aufgestanden, wie ein Blitz. Den Adrenalinflash den ich nachher empfand können sich nur andere 230V-Opfer vorstellen; *Huii, das zuckt!* Danach lacht man hysterisch und ist ganz aufgeputscht. Ich muss jetzt noch lachen, einfach irre dieses *Zuck*.

5.2.1 Meine Hassliebe zu PWM

Um nun etwas konkreter zu werden, was ich im Keller genau so trieb, werde ich Ihnen nun kurz den Frust beschreiben, welchen meine PWM-Odyssey in mir auslöste. Die Situation ist simpel: Man sitzt im dunklen Keller, vor sich der Laptop, der ganze Tisch voller Drähte, Kabel und winzigen Elektroteilchen, es ist *voll*, eine richtige Unordnung. Das ist normal. Ich teste die Funktionalität der neuen PWM-Schaltung und es geht nicht. Ich sende also genau die Bytes über die serielle Schnittstelle zum Arduino, welche er in PWM-Signale für einen bestimmten Lüfter umwandeln soll. Aber der Lüfter dreht nicht, oder immer noch gleich. Das System hat also nicht funktioniert. Die Frustration ergibt sich nicht aus dem simplen *Nicht-Funktionieren*, sondern aus dem *Wo-ist-der-Fehler?*-Problem. Denn mit einer seriellen Schnittstelle, die sind heikel, glauben Sie mir, einem Arduino, der ist auch heikel, und vier Transistoren hat man reichlich potenzielle Fehlerquellen. Das serielle Signal, also ein paar Bytes müssen vom Arduino registriert, gelesen, geparsed (mehrere Werte voneinander unterscheiden), ins richtige Format konvertiert und dann an die PWM-Funktion weitergegeben werden. Diese an sich ist recht schlicht, doch dann erst beginnt der elektronische Teil, also die Verteilung der vier PWM-Signale auf die Transistoren, welche den Hauptstrom für die Ventilatoren regeln. Wenn eine Lötstelle nicht richtig verbindet, ist die ganze Leitung unterbrochen, das heisst ich muss neu löten. Und wenn ein ganzer Transistor aufgrund einer zu hoch angesetzten Spannung durchbrennt (ich nenne es *frittieren*), darf man alle drei Pins des Transistors ablöten und einen neuen dranlöten.

Puh, das sind eine Menge potenzieller Fehler. Aber ich habe es geschafft, der PWM-Controller tut genau das, was ich ihm jetzt sage. Das ist äusserst erfüllend, auch wenn ich jedesmal noch ein Bisschen Angst habe, auf ein unidentifizierbares Problem zu stossen. Eine technisch genauere Beschreibung dieser Schaltung finden Sie im nächsten Kapitel.

5.2.2 Relays als nette Abwechslung

Im Vergleich zum PWM-Karussell war der der Relay-Controller um einiges einfacher, wenn man die Software ignoriert... Die Relays, für alle non-Techies, sind wie Lichtschalter, sie lassen entweder Strom durch oder nicht, je nachdem, ob man an einem dritten Pin eine Spannung anlegt oder nicht - bekanntlich 5V. Der Lötaufwand war hier um einiges kleiner. Denn ich verwende, weil ich mit 230V arbeite, dickere Kabel, also normale Haushaltskabel, genau die, welche in ganz neuen, noch lampenlosen Häusern immer aus der Decke hängen. Sie sind um einiges steifer als meine bisherigen *5V-Drähtchen*. Für den Arduino brauchte ich eine eigene Stromversorgung, welche nicht 230V sondern 5V entspricht. Eigens dafür bestückte ich die Platine mit einem kleinen Netzteil, es ist wirklich niedlich. So richtige Schwiriegkeiten gab es bezüglich der Löt- und Elektronikpraxis beim Relay keine. Was für eine nette Abwechslung.

5.3 Oszilloskopisches Wunder

Das, lieber Leser, ist eine eigens von mir benannte Emotion, welche sich ergibt, wenn man das erste Mal einen Fehler findet, indem man sich hinsetzt und versucht, die Stromsignale *selbst* zu verstehen. Wenn man bedarf für ein solches Gerät hat, weiss man, dass man ganz unten angekommen ist, bezüglich den elektronischen Sphären - Mental ist man nämlich ganz oben. Ein *Oszi* ist ein Gerät mit einem Bildschirm und sogenannten *Klemmen*, welche man an bestimmten Punkten einer Schaltung ansetzt, um anschliessend den Verlauf der Spannung beobachten zu können. Es ist ein Wundergerät, welches jedem richtigen Elektroniker das Herz schneller schlägen lässt. Das beste ist wohl, dass ich jetzt weiss, wie man es bedient. Es hat mir womöglich den Rest meiner Seele gerettet, nachdem ich einen grossen Teil schon verloren hatte und mein Vater mir sagte, wo diese Wundermaschine versteckt ist. Ich benutzte das Oszi für die Veranschaulichung der PWM-Signale. Das funktioniert sehr gut, denn die Verhältnisse lassen sich direkt ablesen. So wusste ich immer, wie stark das PWM-Signal an einem Punkt ist. Bei der Fehlerfindung in einem Labyrinth wie meinem PWM-Controller schätze ich das Oszi besonders, es wirkt wie eine Leiter, mit welcher man auf die Mauern draufklettert und über die Gänge in Richtung Ausgang balanciert. Das Oszilloskop ist eine Art des *Debuggings*, einer Praxis, welche einen grossen Teil der Arbeit ausmacht.

5.4 Code

Der Softwareteil des *Konstrukts* ist nicht zu unterschtzen. Ich habe 95% der Programmierherausforderungen mit C++ und Python erledigt.

5.4.1 C++/Arduino

Der Titel mag etwas verwirren, denn die Low-Level-Programmiersprache für alle Arduinos ist eine leichte Abänderung der Sprache C++, welche wiederrum eine objektorientierte Weiterentwicklung des Klassikers *C* ist. Mit dieser Sprache habe ich die *Backends* programmiert, namentlich die Arduinos, ich verwende zwei, einen für jeden Controller. Beide müssen Signale entgegennehmen können und dann dementsprechend handeln. Am Anfang der Arbeit, ab

dem Moment, wo ich den ersten Arduino herausnahm, lag meine Hauptbeschäftigung fast nur darin, herauszufinden, was ich alles erreichen kann *und wieviel meiner Seele ich dafür opfern muss*. Spass, aber ich glaube Sie verstehen langsam, dass mir die Komplexität von Backend-Problemen leicht zuwider ist. Ich erläutere das mal: *C* und das davon abstammende *C++*, respektive *Arduino* sind sehr *lowelevel*, d.h. sie arbeiten nah am Prozessor, für mich als Programmierer bedeutet das, dass ich mich um mehr Probleme sorgen muss, als ich es mir von *highlevel* Sprachen wie Python gewohnt mit. Ein kleines Beispiel habe ich hier. Während man bei Python die Deklarierung einer Variable nicht mehr nötig ist,

```
number = 5
```

muss man bei C++ höllisch aufpassen, nichts falsch zu machen, sei es die Deklaration mit `int` oder das `;` am Ende jeder Zeile.

```
int number = 5;
```

Das ist nur einer vieler kleiner Syntaxfinessen, welche einem C++ abverlangt, für mich aus der Python-Ecke ist das kompliziert. Nichtsdestotrotz ist die MA nicht das erste Mal, bei dem ich C++ programmiere, ich kannte auch Arduinos schon davor und habe mit ihnen herumgespielt, z.B. um LED-Streifen anzusteuern. Komplett verloren war ich also nicht. Zudem kommt noch, dass mein Vater jahrelange Erfahrung mit *C* hat. Mich beeindruckt besonders der Fakt, dass er seinen eigenen C-Compiler geschrieben hat. *Wie macht das denn?* Wie baut man einen kompletten Compiler nur in Assembly, einer Sprache, die den Namen *Sprache* nichtmal mehr richtig verdient.
Den kompletten C++-Code für Arduinos schreibt man in der eigens dafür vorgesehenen Arduino-IDE. Diese hat den Vorteil, dass man das Script, wenn der Arduino per USB mit dem Gerät verbunden ist, direkt hochladen kann. Der Workflow an sich ist somit sehr schnell, was ich schätze. Das grösste Stück Code was ich schrieb, waren die sogenannten *Relay-Timer*, sie steuern jedes Relay unabhängig voneinander. Der Kniff bei der Sache, wenn man das Problem mithilfe von C++ löst, sind die dynamischen Updates. Ein Beispiel: Angenommen ein Relay ist für 20 Sekunden an, lässt also Strom durch, und für 20s aus, lässt keinen Strom durch, und ich verspüre das Verlangen, die Auszeit auf 10s zu ändern; Dann muss die Software, welche den Relays mitteilt, wann sie zu schalten haben, das mitbekommen und seine Konfiguration ändern. Mit C++ ein Protokoll zu erstellen, womit man über die serielle Schnittstelle Kommandos absetzt, in diesem Beispiel eine neue Konfiguration für ein bestimmtes Relay, ist durchaus machbar, hab es ja selbst gemacht. Allderdings, und das tut mir in der Seele leid, denn ich verbrachte noch so einige Stunden vor der Arduino-IDE, lässt sich ein so komplexes Problem auf Python einfach besser angehen. Nicht nur sind die Debugging-Möglichkeiten, also das Fehlerfinden, grösser, sondern ich bin schlicht einfach ein besserer Python- als C++-Coder. Was ich also tat, war, das komplette Relay-Timer-Script auf Python zu übersetzen.

5.4.2 Python

Nicht, dass das Relay-Timer-Script lange gehalten hätte, ich hab's zwei Tage nach der Übersetzung von C++ wieder verworfen und ein komplett neues Konzept zu realisieren versucht. Zu solch code-lastigen Beispielen wie dem Relay-Timer und dem vorher erwähnten PWM-Controller erfahren sie mehr im nächsten Kapitel *Technische Hintergründe*.
Python ist bekanntlich eine, wenn nicht *die* einfachste Einsteigerprogrammiersrpache, vorausgesetzt man will auch wirklich etwas anstellen. Gerade in Felder wie *künstlichen Intelligenzen*

und *Big Data* wurde Python in den letzten Jahren immer stärker. Mittlerweile lässt sich fast alles damit machen. Eine Opensource-Bewegung namens *pySerial* hat es sich zum Ziel gemacht, auch noch in den Lowlevel-Sektor vorzudringen, was für mich bedeutet, dass pySerial vielleicht sogar eine ernste Alternative zu C++ dargestellt hätte. Effektiv weiss ich das nicht, denn ich kam mit C++ genug gut klar, als das ich mich hätte umgewöhnen wollen. Python hat also, wie schon mitgeteilt, einen grossen Teil der Anforderungen meines Vorhabens erfüllt. Der Workflow ist etwas anders, das Debugging ist einfacher, vorallem mit IDEs wie *PyCharm*. Für alle die das nicht wissen, das ist der Tesla unter allen Python-Entwicklungsumgebungen und für meine Vorhaben noch fast zu gross. Das bedeutet, dass ich den letzten grossen Teil Python nicht mit PyCharm sondern mit Sublime bestritt. Sublime ist eine gute Mittellösung, er hat einen eingebauten Compiler, was bedeutet, dass der Code direkt in der IDE ausgeführt werden kann, und ausserdem sind die Editorfunktionen genau nach meinem Geschmack. Jeder Coder entwickelt nach ein paar Jahren so seinse Präferenzen, welche sich natürlich immer wieder ändern können, aber *mein Gott*, das Sublime-Team hat da ein paar feine Sachen eingebaut! Die intensiven letzten Tage vor der Abgabe habe ich mit Sublime verbracht - Unter Zeitstress muss eine IDE *alles* raushauen können, vorallem *Refactoring*: Wenn man dann einfach mal schnell den Namen einer bestimmten Variabel oder Funktion im *gesamten* Projekt flux ändern kann, dann, lieber Leser, *ist das Premium.*

Eine rechte Lobeshymne an Sublime, Sie merken es schon. Was aber *genau* habe ich mit Sublime genau gecodet? Einen Webserver. Schonmal von *Flask* gehört? Das ist eine ganz feine Kreation: Ein Pythonmodul, mit dem man, wie erwartet, einen kompletten Webserver aufsetzen kann. Serversprachen wie PHP verlieren damit recht ihre Wichtigkeit. Das trifft sich in meiner Situation recht gut. Ich muss wohl etwas ausholen, denn Sie wundern sich womöglich, warum genau mein Projekt bedarf einer Website hat. *Will er etwa schon sein Smart-Growbox-Business eröffnen?* Nein, das habe ich nocht nicht vor, dafür brauchts noch etwas Forschung. Ich könnte die Belüftungsregelung von einer KI übernehmen...
Eine Website brauche ich ganz einfach, um einen direkteren Einfluss in die ganze Regelungstechnik zu haben. Beispiele dafür sind die Relays oder manuelle Steuerung des Lüftungsystem. Mit einer Website kann ich von überall der Welt - auch von Korsika aus - auf die Zentralsteuerung meiner Pflanzenlebenserhaltung Einfluss ausüben. *Aber Jonas, was ist mit SSH?* (SSH ermöglicht es mir remote Textbefehle auf einer Maschine abzusetzen) Ja, liebe Bedenkstimme, SSH, wäre auch eine Möglichkeit, aber damit lassen sich keine Graphen darstellen und ein anständiges, grafisches Benutzerinterface ist einfach was feines. Ausserdem setzt *Flask* auch nur auf Python auf! Eine Website ist einfach cooler und die eben erwähnte Funktion von einer Graphendarstellung *wäre* auch golden. Ich muss Sie jedoch enttäuschen, bis jetzt haben es die Graphen noch nicht auf die Webseite geschafft. Der alleinige Fakt, dass ich mit Python einfach eine Webseite aus dem Boden stampfe ist schon ansehnlich, vorallem weil das ich das vorher noch nie gemacht habe. Und *nur* Python ist es dann eben doch nicht; Das Frontend, also das, was der Browser darstellt, ist ganz traditionell in HTML und CSS geschrieben - und es ist lange her, seit ich das letzte mal Webseiten designed habe. Es wird ja auch immer seltener, dass sich eine Person hinsetzt und echte HTML-Tags in ein Dokument *töckelt*, wenn man die ungeheuere Menge an *Mach-deine-eigene-Webseite*-Angeboten beachtet. *Genau so eine Lösung wäre doch perfekt gewesen, ich hätte mir die Webseite einfach zusammengeklickt.* In erster Linie ja, das Frontend hätte ich so innerhalb eines Tages am laufen gehabt. Den Server allerdings mit den Arduinos zu verbinden, wäre irre schwierig bis unmöglich. Ich brauche meinen eigenen Server. YouTube enthält genug Tutorials, um in zwei Tagen eine solche Web-Instanz ins Leben zu rufen. Ich tat genau das. Zack, hatte ich einen Ordner mit allen nötigen Scripts, um mir selbst

einen Seite zu hosten. Das Interface hatte ich auch in *absehbarer* Zeit am laufen. Naja, da fängt eben dann das Backend des Server an, welches die *wirklich* grossen Aufgaben des *Konstrukts* übernimmt

Die Website, wer es noch nicht erraten hat, wird vom Raspi gehostet. Dieses kleine Linux-Gerät kann so einiges, einen Webserver zu hosten ist, wenn man keine ressourcen-lastigen Webapps hostet, ein gutes Beispiel dafür. Jetzt habe ich den Webserver und ein Python-Script, welches die Relays ansteuert. Was fehlt noch?
Klar doch, der *Daemon*. Nicht erschrecken, das ist kein echter Dämon, das ist nur ein Hintergrundprozess der, deshalb der Name, *nie stirbt*. Er läuft 24/7. Der Webserver ist auf eine Art auch ein *Daemon*, er läuft auch immer, sonst wäre es eine seltsame Webseite. Eine Website mit Öffnungszeiten, komische Vorstellug... Der Daemon ist das Herz, das Hirn, nennen Sie es wie Sie wollen, des *Konstrukts*. Er läuft immer, weiss und steuert alles. Ich habe den natürlich auch in Python geschrieben. Der Webserver und das Relay-Script waren im Vergleich zum Daemon kleine Engel. Das ganze Ding ist so komplex aufgebaut, dass ich einiges meiner Seele opfern musste, bis ich den Dämon bezwang... Im Techik-Kapitel erfahren Sie genauer, wie er funktioniert, jetzt geht es weiter mit Git - dem nächsten Biest.

5.5 GitHub, the Beast

Der Titel soll nicht lügen. Wenn Sie jemals mit Git gearbeitet haben, dann stimmen Sie dem Titel zu, wenn nicht, erkläre ich es kurz. GitHub ist ein Onlinedienst, mit welchem man seine Code-Projekte hochlädt um mit seinem Team oder der Opensource-Community daran zu arbeiten. Es erlaubt einem eine recht fortgeschrittene Kontrolle über verschiedene Versionen des Projekts. Ich habe den kompletten Datenteil des *Konstrukts* mit Git verwaltet, also jedes Stück Code, Bilder, Konzepte und auch alle LaTeX-Dateien, respektive die Buchstaben welche Sie *jetzt* gerade lesen. Es funktioniert wie eine grosse Cloud, einfach besser und mit einer Menge mehr Funktionen. So viele, dass ich einen guten Teil meiner Zeit damit verbrachte, `git` *zu lernen*. Man achte auf die Kleinschreibung, denn ich rede vom *Bashprogramm* `git`. Was *Bash* bedeutet? Das ist die *born again shell*, eines der mächtigsten Stücke Technologie des Planeten, man braucht es, um in einer textbasierten Umgebung Befehle abzusetzen. Die *Bash* ist das Zuhause jedes richtigen Linux-Users. Das gesamte Thema *GitHub-Versioncontrol* ist eine Welt für sich. Eine Wissenschaft so wie die Materie der Elektronik, die Pulsweitenmodulation(PWM), die erwähnte Marihuana-Praxis derer, die es ernst damit meinen, dem Coden von Python, C++ oder HTML. Auf die erlernten Fähigkeiten wird im Kapitel *Perspektiven* noch eingegangen. Das es in diesem Kapitel um den Arbeitsprozess und meines Erachtens deshalb auch um den *Workflow* geht, sind hier die wichtigstes `git`-Befehle.

```
git clone
```

Gecloned wird ein Repo (eine Code-Sammlung, wenn man zum ersten Mal runterlädt.

```
git commit
```

Committen tut der Programmierer, bevor er die vorgenommenen Änderungen am Projekt hochlädt. Das Hochladen erfolgt mit

```
git push
```

Und falls es aus der Cloud gespeicherte Änderungen lokal zu synchronisieren, benutzt man

`git pull`

Um zwischen verschiedenen Versionen, auch *Branches/Äste* genannt, umherzuswitchen, benutzt man

`git checkout`

Dies sind die wichtigsten Befehle, welche ich während meiner gesamten Developement-Phase verwendet habe. Natürlich ist das nur die Spitze des Eisbergs, Git birgt noch so manche, deshalb nenne ich es auch *the Beast*. GitHub ist einer, wenn nicht *der* grösste Förderer der Opensource-Community, ich würde es sogar wagen zu behaupten, dass ein Code-Projekt, welches ich nicht auf GitHub finde, kein Opensource-Projekt ist. Sie merken, bezüglich OpenSource bin ich nicht minder leidenschaftlich wie mit der Refactoring-Funktion von Sublime: Es gibt einfach gewisse Dinge, welche einem Developer das Leben ungemein erleichtern. Das nächste ist ebenfalls eines davon.

5.6 Bash

Nebst all den Programmiersprachen, dem Lötkolben und dem Oszilloskop verdanke ich der Bash einen ungemein grossen Teil meiner Effizienz.

`jonas@pop-os:~$`

Jedes Mal, wenn ich das Terminal öffne, grüsst mich dieser Prompt, was dann folgt sind meistens Kommandos wie `cd` oder `ls`. Unter Zeitdruck werden sie so schnell in die Tasten genagelt, dass ich fast Mitleid mit meiner Tastatur habe. Auch hier wird mir jeder geneigte *Linux-Pinguin* einfach nur zunicken und kaum merklich schmunzeln. Die oben genannten Commands, für alle welche nicht vom Südpol stammen, sind die Augen und Beine in der Bash, mit `cd` läuft man durchs Dateinsystem und mit `ls` verschafft man sich einen Überblick über alle vorhandenen Dateien in einem bestimmten Ordner. Die Bash hat eine *so* grosse Präsenz in meinem Leben, wie kein anderes Programm, naja YouTube vielleicht: Jeden `git`-Command setzt man über die Bash ab. Man navigiert damit durch den Computer, tötet überflüssige Prozesse (aber ja nicht die Dämonen, versteht sich) oder programmiert mit `vim` oder `nano` kurz ein kleines Bash-Script. Ich habe genau einmal ein Bash-Script gebaut und zwar, als es mir zu blöd wurde, via GitHub meinen Daemon-Code auf den Raspi zu laden.

5.6.1 rsync

Rsync ist Dropbox für die Kommandozeile, also Bash, aber besser: Es kann verschlüsselt und rekursiv Daten von einer Quelle zum Ziel kopieren, alles was schon da ist, wird nicht doppelt kopiert. Es ist äussert flink und einfach zu bedienen, wenn man sich dann mit den *SSH-Keys* richtig anstellt, muss man nichtmal das Passwort des Zielrechners angeben. Das von mir vorher erwähnte Bash-Script hat alle 5 Sekunden meine Änderung, welche ich mit Sublime, Gott hab es seelig, am Dämonen (ich verwende absichtlich das böse Wort) vorgenommen hab, auf den Raspi kopiert. Was habe ich mir damit gespart? Einen kompletten `git commit -a && git push` Befehl, zusammen mit dem *Commit-Kommentar*, den `git` jedes Mal von einem verlangt, wenn man die Änderungen committet. Rundherum sind das grob 10s und eine Menge unnötiger Tastenanschläge, denn ein GitHub-Commit für lediglich die Änderung eines Variabelnamens war in meiner Situation einfach ein *Overkill*.

6 Technische Hintergründe

6.1 Ganz von vorn.

Arduinos.

6.2 Luft rein, Luft raus.

6.3 Klick, klick, klick

Diese kleine, graue Box, ist in der Lage, jede ihrer oben angebrachten Steckdosen individuell anzusteuern, die Zeitdaten teile ich ihr ebenfalls über ein USB-Kabel mit. Darin eingesteckt sind die zwei Stecker der beiden LED-*Laser* (Kleiner Witz, Sie erinnern sich an den versteckten Warnhinweis bezüglich dem pinken Helligkeitswunder?) und der des Foggers/der Nebelmaschine. Dieses Stück Technik hat mir natürlich auch einen guten Teil der Seel abverlangt, doch ehrlich gesagt ging es bei diesem Teilprojekt um einiges fröhlicher zur Sache als beim PWM-Controller. Ich denke, es liegt am den vier kleinen Relay, welche jedesmal ein kleines Klickgeräusch machen, wenn sie umschalten.

7 Perspektiven

Einhergehend mit den Arbeitsprozessen, bekam ich auch neue, mir zuvor völlig undenkbare Einsichten und Perspektiven auf mehreren Ebenen

7.1 Technische Einsichten

Oszilloskop Flask C++ Loeten Elektronik

7.2 Mögliche Weltverbesserung

asdf

7.3 Warum *genau* baue ich das eigentlich?

asdf

8 Fazit

9 Danke sagen

Vielen Dank Papa für deine unendlich ansteckende Begeisterung für die Materie der Technik.

10 SWAP

Glossar: Oszilloskop CPU Timer-Register Relay Transistor

```
for i in range(19):
```

Alle haben es in dieser Schule gelernt, einige schon im U1; Pflanzen brauchen Wasser, Luft, Licht und Nährstoffe für richtiges Wachstum. Die ersten drei sind notwendig für die Photosyntese. Ausserdem ist auch ein ädequates Klima von grosser Bedeutung; Eine Temperatur von 25-35°C und nicht zu trockene oder zu feuchte Luft.

Das ist äusserst nützlich, denn somit absorbieren die Wände kein Licht und heizen sich auf, sondern das Licht, das nicht direkt auf die Blätter trifft, wird weiter umherreflektiert, bis es schliesslich beim Blatt ankommt.