Entwicklung eines Getränkeautomaten

Maturaarbeit von Felix Graule
Betreuung: R. Riederer und A. Reinhard
Kantonsschule Schaffhausen
2014

Abstract

Im Rahmen meiner Maturaarbeit habe ich einen Getränkeautomaten entwickelt. Der Automat ist in der Lage, aus verschiedenen Zutaten innert kurzer Zeit einen Drink zusammenzustellen. Dabei braucht er relativ wenig Wartung und ist rund um die Uhr einsetzbar.

Die gesamte Dosiermechanik und Steuerungselektronik ist in einem Kühlschrank untergebracht, damit verderbliche Zutaten über lange Zeit gelagert werden können und der Automat möglichst kompakt ist. Um dem Kunden einen Einblick ins spannende Innere zu ermöglichen, ist die Front des Kühlschranks verglast. Die Software des Automaten läuft auf einem demontierten Laptop, welcher als Zentralrechner dient. Die Komponenten der Dosiermechanik und diverse Sensoren werden durch Mikrocontroller der Tinkerforge-Plattform gesteuert. Als Eingabemöglichkeit steht dem User ein 22 Zoll Touchscreen zur Verfügung.

Die Software wurde vollständig mit Python programmiert und umfasst etwa 4000 Zeilen handgeschriebenen Code. Über eine intuitiv bedienbare Benutzeroberfläche kann der Kunde Bestellungen abgeben oder sich über das Projekt informieren. Diese Oberfläche besteht aus vielen verschiedenen Fenstern, zwischen denen der Nutzer hin und her navigieren kann. Die Software ist in der Lage, autonom auf Umwelteinflüsse wie Temperaturschwankungen oder Bewegungen zu reagieren.

Die Zutaten befinden sich in kopfüber hängenden Flaschen. Die Dosierung der Zutaten wird mit Magnetventilen gesteuert, welche jeweils für eine vom Füllstand abhängige, berechnete Zeitdauer geöffnet bleiben. Um diese Öffnungszeit möglichst genau vorherzusagen, wird ein eigens dafür entwickelter Algorithmus verwendet. Die dosierte Flüssigkeit wird von einem beweglichen Trichter unter den Ventilen aufgefangen und über Schlauchleitungen aus dem Kühlschrank in einen Becher geführt.

Inhaltsverzeichnis

1. Einleitung

1.1. Überblick – worum geht es?

Im Rahmen meiner Maturaarbeit habe ich einen Roboter entwickelt, welcher vollautomatisch Cocktails mixen kann. Die zentrale Idee besteht darin, aus verschiedenen Grundzutaten durch geschickte Kombination und Dosierung Cocktails zu mischen. Der Nutzer soll dabei auf einfache Weise einen Drink wählen oder selbst zusammenstellen können. Der gesamte Roboter soll vollautomatisch sein und rund um die Uhr einsetzbar sein, wobei er so wenig Wartungsaufwand wie möglich verursachen sollte.

Das gesamte Projekt ist Open-Source, wer sich also an einem Nachbau oder einer Weiterentwicklung versuchen möchte, darf dies gerne tun. Interessenten können sich jeder Zeit bei mir melden, die E-Mail-Adresse dazu lautet: felix.graule2@gmail.com.

1.2. Ideenfindung

Die Idee für meine Maturaarbeit kam mir zufällig. Ich wollte schon seit längerer Zeit selbst einen Roboter entwickeln, da ich diese sehr spannend finde, unter anderem wegen ihrer Fähigkeit auf Umwelteinflüssen reagieren zu können. Darum habe ich mir überlegt, wo der Mensch im Alltag noch Hilfe gebrauchen könnte. Gemeinsam mit meiner Vorliebe für Mischgetränke aller Art ergab sich so die Idee, einen Cocktailroboter zu entwickeln. Danach fiel mir auf, wie faszinierend und vielseitig das Projekt ist. Ich entschied mich, es als Maturaarbeit durchzuführen.

1.3. Zielsetzung

Ich habe mir für meine Maturaarbeit zwei Hauptziele gesetzt: Erstens einen Roboter zu entwickeln, welcher das vom Benutzer gewünschte Getränk in akzeptablem Zeitraum bereitstellen kann. Zweitens, den Benutzer und den Leser dieser Arbeit zu faszinieren. Ich möchte erklären, wie der Roboter genau funktioniert und ausserdem zeigen, dass man als Schüler komplexe Aufgaben mit Hilfe von Informatik und Elektrotechnik lösen kann.

2. Problemstellung

Zur besseren Übersicht ist das Projekt in verschiedene Hauptaufgaben gegliedert. Als Hauptaufgaben gelten solche, welche für den reibungslosen Betrieb des Geräts unentbehrlich sind. Im Folgenden werde ich die Problemstellung anhand der Hauptaufgaben genau erklären.

2.1. Dosierung und Transport von Flüssigkeiten

Ein fertiges Getränk setzt sich aus verschiedenen Zutaten zusammen. Diese werden jeweils einzeln dosiert und dann gemischt. Das Dosiersystem ist also sehr wichtig, weil Genauigkeit und Effizienz des Mischvorgangs davon abhängig sind. Das Entwickeln einer Dosiermethode ist sehr anspruchsvoll, denn beim Umgang mit Flüssigkeiten und Elektronik muss man sehr vorsichtig sein, da es bei Lecks leicht zu Kurzschlüssen kommt. Ich werde später in der Arbeit einige Ansätze von verschiedenen Dosiersystemen zeigen und erklären.

2.2. Schnittstelle Benutzer – Roboter

Die Anforderungen an die Benutzerschnittstelle des Roboters sind folgende: Der Nutzer muss ohne Lernaufwand eine Bestellung aufgeben können und die Bedienung muss intuitiv sein. In Zeiten der Smartphones und Tablets weiss jeder wie man einen Touchscreen bedient, daher ist dieser das ideale Eingabegerät. Die eigentliche Benutzerschnittstelle ist eine graphische Benutzeroberfläche. Diese ist über den Touchscreen bedienbar und muss entsprechend angepasst sein, etwa über grosse Knöpfe verfügen und auf Texteingabe weitgehend verzichten.

2.3. Dauerhafter Betrieb

Um dauerhaft in Betrieb sein zu können, muss der Stromverbrauch und der Wartungsaufwand des Automaten möglichst gering sein. Das Gerät muss darum möglichst autonom arbeiten können. Eine solche Optimierung ist ausserdem kostensparend und ermöglicht einen kommerziellen Einsatz des Automaten. Aufgrund der notwendigen Kühlung der Zutaten, welche ein schnelles Verderben verhindert, muss sich der Automat in einem gut isolierten Kühlschrank befinden, um den Energieverbrauch niedrig zu halten. Ebenfalls wichtig für den langfristigen Betrieb ist eine Einschränkung des Gesamtkonsums, da ansonsten ständig Zutaten nachgefüllt werden müssen. Dies lässt sich mit einer PIN-Sperre direkt vor der Bestellung des Drinks lösen.

2.4. Ästhetik

Ein ansprechendes Aussehen des Automaten ist wichtig, um Nutzer anzulocken und zufriedenstellend zu bedienen. Vor allem Lichteffekte sind sehr auffällig und sollten daher unbedingt zur Anwendung

kommen. Gerade im Umgang mit Lebensmitteln ist ein hygienisches Gerät wichtig, um das volle Vertrauen des Nutzers zu gewinnen. Der Automat sollte deshalb vorwiegend aus weissen und glänzenden Materialien gefertigt sein. Durch den Einsatz von transparenten Materialien soll es dem Kunden möglich sein, das spannende Innenleben des Automaten zu sehen und den Mischvorgang genau beobachten zu können. Dadurch wird die Neugier des Nutzers geweckt, und er möchte sich über das Projekt informieren, was eines der zwei Hauptziele meiner Arbeit ist.

2.5. Verhältnis zum Kunden

Bei dem Projekt handelt es sich nicht nur um ein Problem aus den Bereichen der Informatik und Elektrotechnik. Schlussendlich geht es auch darum, Kunden vom angebotenen Produkt zu überzeugen und sie als Stammkunden zu binden. Dies soll einerseits durch ein gutes Produkt und ein ansprechendes Design erreicht werden, andererseits auch durch das gezielte Anbieten von bestimmten Zusatzfunktionen für Stammkunden, welche im Folgenden auch als Premium-Kunden bezeichnet werden.

2.6. Ähnliche Projekte

Es gibt heutzutage wahrscheinlich nicht mehr viele Dinge, die es noch nicht gibt beziehungsweise gegeben hat und somit noch erfunden werden können. Das gilt auch für Cocktail-mixende Roboter. Im Internet gibt es unzählige Videos von mehr oder weniger professionellen Robotern. Ich möchte hier die zwei meiner Meinung nach besten Exemplare vorstellen.

2.6.1. Bartendro[1]

Dieser Roboter besteht aus einem stabilen Metallrahmen, unter welche die Zutaten gestellt werden. Das Herzstück des Roboters sind die eigens designten und selbst gefrästen Peristaltikpumpen. Sie befinden sich direkt über den Flaschen und können Flüssigkeiten millimetergenau dosieren. Verbunden sind die einzelnen Pumpen über Ethernet-Schnittstellen mit einem Router. Der Router kommuniziert über ein *Raspberry Pi* [2], welcher die zentrale Steuerung übernimmt. Es können bis zu 15 Pumpen gleichzeitig angesteuert werden.

Bild 1: Der Bartendro

Gedacht ist der Roboter für den Einsatz in Partys und Bars. Eine Kühlung der Zutaten ist nicht vorgesehen, weshalb der Roboter nicht über lange Zeit ohne Wartung benutzt werden kann. Finanziert wurde das Projekt über die Crowd-Funding-Plattform *Kickstarter*. Die Hard- und Software des Bartendros sind Open-Source, auf der Website befinden sich sogar einige Anregungen, wie man den Roboter erweitern könnte. Entwickelt wurde der Bartendro von zwei amerikanischen IT-Profis.

2.6.2. melmacc[3]

Bei der Zubereitung eines Cocktails wandert ein Glas auf einem Förderband von Flasche zu Flasche und holt sich die benötigten Zutaten. Zum Schluss wird dem Glas von einem Roboterarm ein Strohhalm zugefügt. Der melmacc verfügt über eine sehr grosse Drinkrezepte-Datenbank, von welcher aber aufgrund der begrenzten Zutaten nicht ständig alle Cocktails zubereitet werden können.

---

[1] Quelle: http://www.kickstarter.com/projects/partyrobotics/bartendro-a-cocktail-dispensing-robot; http://partyrobotics.com/
Quelle Bild 1 : http://www.wired.com/design/wp-content/uploads/2013/03/bartendro-8-unit-wired-design.jpg

[2] Der Raspberry Pi ist ein sehr kompakter Einplatinencomputer; Quelle: http://www.raspberrypi.org/faqs

[3] Quelle: http://www.melmacc.at/; Quelle Bild 2: http://lh5.ggpht.com/-oSpyRxSjQVc/SvmTaHC7SsI/AAAAAAAAA1c/G9qtze5Pt-k/DSC02778.jpg?imgmax=800

Bild 2: Der melmacc-Drinkroboter

Auch dieser Roboter dient dem Einsatz auf Partys und Festen. Über die verwendete Technik wird nichts berichtet, das Projekt ist leider auch nicht Open-Source. Entwickelt wurde es von drei IT-Profis aus Österreich.

3. Theorie

Ich möchte im folgenden Kapitel einige theoretische Betrachtungen und begriffliche Abgrenzungen erläutern. Dieser Teil soll dem Leser ein bestimmtes Grundwissen über Informatik und Elektrotechnik vermitteln und die Arbeit für jeden verständlich machen.

3.1. Hardware und Software

Als Hardware bezeichnet man alle elektronischen und mechanischen Komponenten eines Systems[4]. Bei einem Computer sind dies beispielsweise das Mainboard, das DVD-Laufwerk, das Netzteil oder der Prozessor. Man unterscheidet zwischen nicht programmierbarer Hardware, zum Beispiel einer Lampe, einem Kondensator oder einem Widerstand, und programmierbarer Hardware, wie etwa dem Prozessor oder einem Mikrocontroller[5].

Als Software bezeichnet man alle nicht-physisch vorhandenen Teile eines Systems[6]. In den meisten Fällen sind das die Programme, die auf der Hardware laufen[7]. Die Software ist die Steuerung der Hardware, kann andererseits aber nicht ohne die Hardware existieren. Software und Hardware stehen darum immer in hoher Abhängigkeit zueinander.

3.2. Schnittstelle Nutzer – Maschine

Um die Kommunikation zwischen Nutzer und Maschine zu ermöglichen verwendet man sogenannte Benutzerschnittstellen. Diese können unterschiedlich aussehen: Knöpfe, Mikrofone, Bildschirme oder ähnliches. Die Schnittstelle ist der Ort im System, an dem die Interaktion zwischen Nutzer und Maschine stattfindet[8]. Sie soll die Bedienung eines Systems erleichtern. Die Schnittstelle kann dabei entweder Informationen des Nutzers aufnehmen, sogenannte Inputs erhalten, oder sie kann Meldungen der Maschine für den Nutzer sichtbar machen, sogenannte Outputs ausgeben[9]. Ein System ohne jegliche Schnittstelle ist selten sinnvoll, da der Nutzer sie weder beeinflussen noch auswerten kann.

[4] Quelle: http://de.wikipedia.org/wiki/Hardware
[5] Quelle: http://pcsupport.about.com/od/termshm/g/hardware.htm
[6] Quelle: http://www.informatik.uni-leipzig.de/lehre/Heyer9900/kap18/sld001.htm
[7] Quelle: http://de.wikipedia.org/wiki/Software
[8] Quelle: http://en.wikipedia.org/wiki/User_interface
[9] Quelle: http://www.itwissen.info/definition/lexikon/Benutzeroberflaeche-UI-user-interface.html

3.3. Roboter oder Automat

Für viele Nutzer des Geräts ist klar, dass sie einen Getränkeautomaten und keinen Roboter bedienen. Die Frage, worum es sich bei dem Projekt handelt, mag zwar trivial erscheinen, ist in Wirklichkeit jedoch nicht eindeutig zu beantworten. Zur genauen Abklärung sollte man die Definitionen des Roboters und des Automaten analysieren. Bereits die Wahl der Definitionen ist jedoch problematisch, da es eine grosse Bandbreite an verschiedenen Abgrenzungen gibt. Ich habe mich für folgende Definitionen entschieden:

> Definition eines Roboters nach der Robotic Industries Association[10]:
>
> *„Ein Roboter ist ein programmierbares Mehrzweck-Handhabungsgerät für das Bewegen von Material, Werkstücken, Werkzeugen oder Spezialgeräten. Der frei programmierbare Bewegungsablauf macht ihn für verschiedenste Aufgaben einsetzbar.“*
>
> Definition eines Automaten nach dem Duden Informatik[11]:
>
> *„Allgemein ist ein Automat ein technisches oder mechanisches Gerät, das zu einer Eingabe ein bestimmtes Ergebnis ausgibt.“*

Beide Definitionen treffen auf das Gerät zu, es handelt sich folglich also um einen Automaten und einen Roboter. Diese Doppelung ist nicht ungewöhnlich, da beinahe jedes technische Gerät als Automat und jedes programmierbare und bewegliche Gerät als Roboter bezeichnet werden kann. Einerseits passen die komplexe Software und die Fähigkeit, auf Umwelteinflüsse wie Benutzereingaben und Temperaturveränderungen reagieren zu können, eher zum Begriff Roboter. Andererseits kann das Gerät nur Drinks zubereiten und ist daher nicht *„für verschiedenste Aufgaben einsetzbar“.* In der Arbeit werde ich daher beide Begriffe gleichwertig verwenden.

3.4. Programmiersprache

Unter einer Programmiersprache versteht man eine erfundene, formale Sprache, die der verständlichen Kommunikation zwischen Mensch und Computer dient[12]. Jede Programmiersprache hat eine spezifische Syntax und semantische Regeln[13]. Mit Programmiersprachen ist es möglich, komplexe Probleme oder Abläufe, sogenannte Algorithmen, in einer für Computer verständlichen Art niederzuschreiben. Der Computer muss diesen Code lediglich noch in Maschinencode umwandeln,

[10] Quelle: http://definitions.uslegal.com/r/robotics/
[11] Quelle: S. 65 in „Duden Informatik“, 2001
[12] Quelle: http://wirtschaftslexikon.gabler.de/Definition/programmiersprache.html
[13] Quelle: http://de.wikipedia.org/wiki/Programmiersprache

der dann direkt auf dem Prozessor läuft. Dieser Maschinencode ist das bekannte Chaos von Nullen und Einsen, sogenannter Binärcode. Bei Programmiersprachen kann man zwischen folgenden Typen unterscheiden:

3.4.1. Low – Level – Programmiersprachen

Unter diesem Begriff fasst man Programmiersprachen zusammen, welche stark an die verwendete Hardware gebunden sind und sehr spezifisch auf diese abgestimmt sind und damit nur selten kompatibel zu anderer Hardware sind[14]. Der Name kommt daher, dass sich diese Sprachen relativ nahe am programmierten Chip befinden. Ein Low-Level-Programmiercode kann leicht in Maschinencode übersetzt werden, sofern er auf dem richtigen Prozessor ausgeführt wird.

3.4.2. High – Level – Programmiersprachen

Dieser Begriff umfasst alle Programmiersprachen, die von der eingesetzten Hardware eher unabhängig sind und darum oft kompatibel zu anderen Systemen sind. High-Level-Programmiersprachen sind deutlich stärker abstrahiert als Low-Level-Programmiersprachen, sie sind dem Maschinencode also viel weniger ähnlich[15]. Ausserdem ähneln sie bezüglich Syntax unseren menschlichen Sprachen und sind somit einfacher zu verstehen[16]. Der Name kommt daher, dass diese Sprachen weit vom Chip und dem Maschinencode entfernt sind. Wegen dieser grossen Unterschiede ist ein externes Programm notwendig, um den Quellcode in Maschinencode umzuwandeln. Dieses Programm nennt man *Compiler*.

3.5. Mikrocontroller

Unter diesem Begriff versteht man ein programmierbares Rechnersystem, das über einen Mikroprozessor und weitere funktionale Komponenten verfügt, beispielsweise Sensoren, Speicherelemente oder andere elektrische Komponenten[17]. Mikrocontroller sind für ihren genauen Einsatz optimiert, sie führen also Spezialaufgaben aus. Häufig sind dies Mess- oder Steuerungsaufgaben, die sie einem übergeordneten System abnehmen und die Ergebnisse dann über eine Schnittstelle an das Zentralsystem weitergeben.

[14] Quelle: http://en.wikipedia.org/wiki/Low-level_programming_language
[15] Quelle: http://en.wikipedia.org/wiki/High-level_programming_language
[16] Quelle: *„High Level Programmiersprachen als Brückensprachen zwischen Mensch und Maschine“* von Ralv Wohlgethan, 1. Auflage, 2007
[17] Quelle: http://www.itwissen.info/definition/lexikon/Mikrocontroller-MCU-micro-controller-unit-microC.html

3.6. Unified Modeling Language (UML)

Unter diesem Begriff, der übersetzt *Vereinheitlichte Modellierungssprache* bedeutet, versteht man eine standardisierte Form der graphischen Darstellung von Software-Systemen. Entwickelt wurde UML in den 90-er Jahren von *Rational Software*. 1997 übernahm die *Object Management Group* das Projekt und sorgt seither für die sinnvolle Weiterentwicklung von UML[18].

Der Zweck einer einheitlichen Charakterisierung eines Programms ist in erster Linie das Erstellen eines strukturierten Plans, um eine klare Auftrennung und Arbeitsteilung zu ermöglichen[19]. Bei grossen Software-Projekten ist ein solcher Plan absolut unverzichtbar, da man teils mehrere Millionen Zeilen Code schreiben und warten muss. Ausserdem ermöglicht man Interessenten mit UML-Diagrammen das Programm zu verstehen, ohne dass sie dafür den Quellcode kennen müssen.

UML-Dokumentationen bestehen hauptsächlich aus verschiedenen Diagrammen. Ich werde im Folgenden einige Diagrammtypen vorstellen und erklären, wie sie aufgebaut sind und wozu sie dienen. Man unterscheidet bei UML-Diagrammen zwischen Verhaltens- und Strukturdiagrammen. Verhaltensdiagramme zeigen den Ablauf von Aktionen in der Software. Strukturdiagramme dienen dazu, den Aufbau der Software übersichtlich darzustellen.

3.6.1. Das Use – Case – Diagramm[20]

Use-Case-Diagramme gehören zu den Verhaltensdiagrammen. Ein Use-Case, zu Deutsch *Anwendungsfall*, ist eine mögliche Aktion des Akteurs. Im Use-Case-Diagramm versucht der Entwickler die groben Anforderungen an die Software zu erfassen und zu ordnen. Festgehalten wird das, was das Programm können muss. Die Art der Implementierung dieser Funktionen ist hier noch kein Thema, das „wie?“ wird komplett ausgeblendet. Der Name des Diagramms kommt daher, dass man sogenannte Use-Cases in Verbindung zum Akteur setzt.

Da eine Software, auch System genannt, von verschiedenen Akteuren auf unterschiedlichen Ebenen verwendet wird, gibt es für jeden möglichen Akteur ein eigenes Use-Case-Diagramm. Der normale User hat also ein anderes Diagramm als etwa der Administrator. Eine Verbindung zwischen zwei Elementen wird in UML Assoziation genannt. Von diesen Assoziationen kommen in allen Diagrammtypen verschiedene Arten vor. Im Use-Case-Diagramm sind es zwei unterschiedliche Typen:

[18] Quelle: http://en.wikipedia.org/wiki/Unified_Modeling_Language
[19] Quelle: https://www.lucidchart.com/pages/uml/what-is-uml
[20] Quelle: http://www.highscore.de/uml/; Quelle Bild 3 bis 6 und 8: http://www.highscore.de/uml/

- Eine durchgezogene Linie zwischen Akteur und Use-Case bedeutet, dass der Akteur den Use-Case ausführen kann.
- Bei einer gestrichelten Linie zwischen zwei Use-Cases unterscheidet man wiederum zwei verschiedene Typen, die man anhand eines Schlüsselwortes über der Linie unterscheidet:
    - Eine „include“-Verbindung von Use-Case A zu Use-Case B bedeutet, dass immer wenn A gestartet wird, auch B gestartet werden muss.
    - Eine „extend“-Verbindung bedeutet, dass Use-Case D unter Umständen Use-Case C erweitern könnte. Die Bedingung für eine Erweiterung ist jeweils neben der Assoziation notiert.

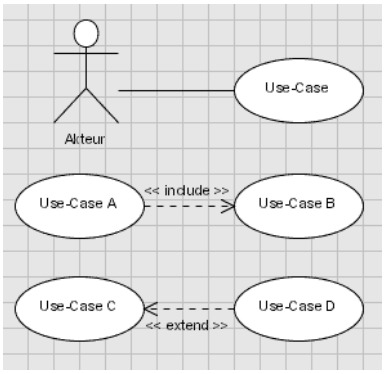


Bild 3: Mögliche Assoziationen

3.6.2. Das Aktivitätendiagramm[21]

Das Aktivitätendiagramm ist ein auch Verhaltensdiagramm. Mit diesem Diagramm stellt man Abläufe übersichtlich und strukturiert dar. Abläufe bestehen aus einer Abfolge von Aktionen und können Use-Cases oder Funktionen eines Programms sein. Das Aktivitätendiagramm wird oft als Allround-UML-Diagramm bezeichnet, weil es während des gesamten Software-Entwicklungsprozesses immer wieder für Klarheit sorgt.

Jedes Aktivitätendiagramm hat einen oder mehrere Anfangs- und Endpunkte, die über Knoten und Kanten verbunden sind. Knoten sind Ereignisse und Kanten die Verbindungen zwischen den Aktionen. Als Token bezeichnet man ein Teilchen, das sich vom Anfangs- zum Endpunkt bewegt und dabei alle Aktivitätsteile ausführt, die auf dem Weg liegen. Der Weg eines Token kann auch verzweigt sein, was man mit Verzweigungen und Gabelungen darstellt. Es gibt zwei Typen von Knoten:

Bild 4: Symbole des Start- und Endknotens

- Aktionen sind die kleinsten Bausteine einer Aktivität. Dargestellt werden sie in abgerundeten Rechtecken.
- Objektknoten sind Datenspeicher. Sie werden als normale Rechtecke dargestellt. Ein Token kann entweder Daten aufrufen oder abspeichern.

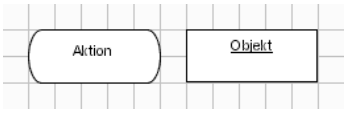


Bild 5: Symbole der Knotentypen

[21] Quelle: http://www.highscore.de/uml/

Es gibt ausserdem zwei Arten von Wegverzweigungen:

- Die Verzweigung wird als leeres Karo dargestellt. Gelangt ein Token zu dieser Verzweigung, wird es direkt an die Ausganslinie weitergegeben.
- Die Gabelung wird als schwarzer Balken gezeichnet. Der Unterschied zur Verzweigung besteht darin, dass ein Token erst dann weitergeleitet wird, wenn von allen Eingangslinien ein Token eingetroffen ist.

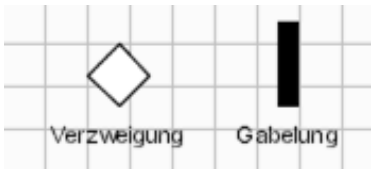


Bild 6: Symbole der Wegverzweigungen

3.6.3. Das Klassendiagramm[22]

Das Klassendiagramm ist ein Strukturdiagramm, in welchem die verschiedenen Klassen eines Programms dargestellt und zueinander in Beziehung gesetzt werden. Ähnliche Klassen werden zu Paketen zusammengefasst. Das Klassendiagramm ist dem Quellcode am ähnlichsten, weil es um die Implementierung und Aufteilung der Funktionen in Klassen geht. Klassen sind ein wichtiges Konzept des objektorientierten Programmierens, ich werde sie darum anhand einer Analogie kurz erklären.

Man kann sich eine Klasse als ein Kuchenrezept vorstellen. Mit Klassen erschafft man Objekte nach bestimmten Regeln, genau wie man mit einem Rezept eine bestimmte Art Kuchen backt. Diesen Kuchen kann man nach dem Backen auch noch verändern, was bei Objekten einer Klasse ebenfalls möglich ist. Auch bezüglich der Struktur haben Klassen und Rezepte einige Gemeinsamkeiten. Beide verfügen über Parameter mit bestimmten Werten, sogenannten Attributen oder Eigenschaften, und über Anweisungen zur Verarbeitung der Eigenschaften, sogenannten Methoden.

Das nebenstehende Beispiel eines Bankkonto-Verwaltungssystems soll verdeutlichen, wie Klassen genau aufgebaut sind[23].

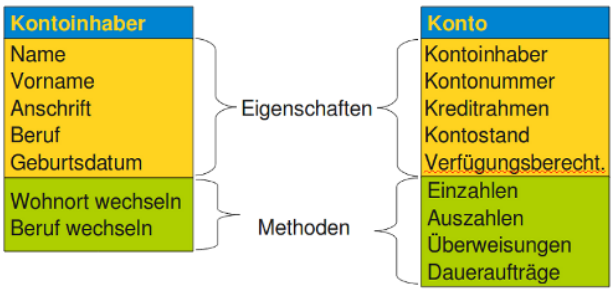


Bild 7: Darstellung von Klassen

[22] Quellen: http://www.highscore.de/uml/
[23] Quelle Bild 7: http://www.python-kurs.eu/klassen.php

Zwischen den Klassen können folgende Assoziationen bestehen:

- Ein Pfeil von Klasse A zu Klasse B bedeutet, dass Klasse A auf Elemente von Klasse B zugreifen kann, man spricht von sogenannter Navigierbarkeit.
- Eine Verbindung mit Pfeilspitze und Raute bedeutet, dass Klasse C nicht ohne Klasse D existieren kann. Mit dieser Verbindung stellt man eine starke Abhängigkeit zwischen zwei Klassen dar. Die Klasse mit der Raute ist jeweils von der anderen Klasse abhängig.
- Ein Pfeil mit leerer Spitze steht für Vererbung. Die Klasse F erbt also alle Attribute und Methoden der Klasse E. Die Klasse F ist die Kindklasse der Elternklasse E. Die leere Pfeilspitze zeigt auf die vererbende Elternklasse.

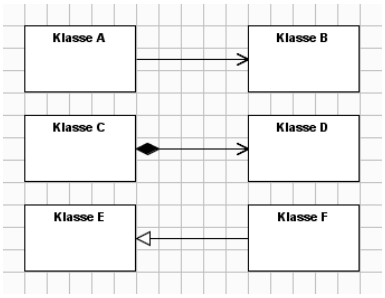


Bild 8: Assoziationen zwischen Klassen

3.7. Dosieren von Flüssigkeiten

Das exakte Abmessen und Abfüllen eines Mediums nennt man Dosieren. Bei verschiedenen Dosierungsmethoden gibt es einige grundlegende Unterschiede. Unter Massenmessungsverfahren versteht man Dosierverfahren, welche das Gewicht messen. Solche Dosiermethoden bezeichnet man auch als *gravimetrisch.* Bei Volumenmessungen wird das Volumen gemessen beziehungsweise begrenzt. Solche Verfahren werden als *volumetrisch* bezeichnet[24].

Bei der Volumenmessung kann man zwischen zwei verschiedenen Methoden unterscheiden: Man kann einen Behälter mit bekanntem Volumen befüllen, überschüssiges Material abschöpfen und anschliessend die übrige Flüssigkeit ausfliessen lassen. Diese Methode erfordert jedoch eine konstruktiv aufwändige Dosiermechanik. Es ist aber auch möglich, eine ruhende Flüssigkeit durch einen Druck- oder

Höhenunterschied in Bewegung zu setzen und den Vorgang nach festgelegter Zeit abzubrechen. Das Volumen der ausgeflossenen Flüssigkeit lässt sich dann mit Hilfe einiger physikalischer Zusammenhänge berechnen, welche hier kurz erklärt werden, da sie in meiner Arbeit eine wichtige Rolle spielen. Dazu sollten erst einige grundlegende Begriffe geklärt werden:

[24] Quelle: „Handbuch Dosieren“ von Gerhard Vetter, 2. Auflage, 2001

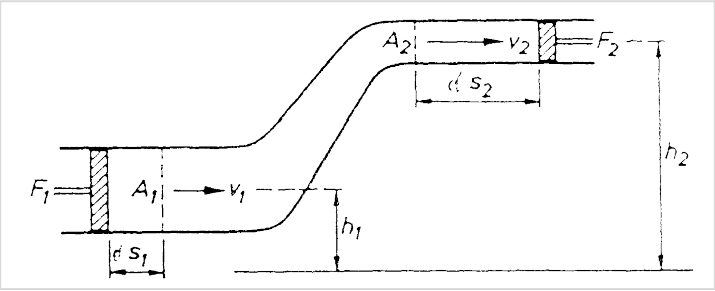
- Als Volumenstrom I_v bezeichnet man das Volumen, das pro Zeitintervall durch einen Querschnitt fliesst[25]. Die SI-Einheit des Volumenstroms ist $\frac{m^3}{s}$. Berechnet wird der Volumenstrom folgendermassen: $I_V = Av$, wobei A der Rohrquerschnitt und v die Strömungsgeschwindigkeit ist.
- Die Ausflussgeschwindigkeit gibt an, wie schnell sich eine Flüssigkeit an einem Ausfluss bewegt. Die SI-Einheit der Ausflussgeschwindigkeit ist $\frac{m}{s}$.
- Bei einer *stationären* Strömung sind die Fliessgeschwindigkeit und die Querschnittsfläche keiner zeitlichen Änderung unterlegen[26]. Das ist gleichbedeutend mit der Aussage, dass der Volumenstrom $I_v = Av$ entlang einer stationären Strömung konstant ist.
- Ist die Flüssigkeit einer stationären Strömung inkompressibel, so ist die Dichte ρ konstant. Für die durchströmende Masse gilt darum: $\Delta m = Avt\rho$ oder $\Delta m = \Delta V \rho$.

Die für mein Projekt relevanten Effekte werden durch das Gesetz von Bernoulli und das Ausflussgesetz von Torricelli beschrieben. Das Gesetz von Bernoulli besagt folgendes:

Ideale Flüssigkeiten bewegen sich bei stationärer Strömung stets so, dass der Ausdruck

$$\rho \frac{v^2}{2} + \rho g h + P$$

konstant bleibt[27]. Dazu müssen folgende Bedingungen erfüllt sein: die betrachtete Flüssigkeit muss inkompressibel sein und es darf keine Reibung auftreten, die Flüssigkeit darf also nicht viskos sein[28]. Für Flüssigkeiten mit geringer Viskosität, wie die vom Automaten verwendeten Getränke, gilt sie in sehr guter Näherung.



Skizze 1: Situationsskizze Bernoulli

ρ:	Dichte	$\frac{kg}{m^3}$
v:	Fliessgeschwindigkeit	$\frac{m}{s}$
g:	Erdbeschleunigung	$\frac{m}{s^2}$
h:	Höhe	m
P:	Hydrostatischer Druck	Pa

[25] Quelle: Kapitel 13 des Buches „Physik“ von Paul A. Tipler und Gene Mosca, 6. Auflage, 2009
[26] Quelle: http://de.wikipedia.org/wiki/Station%C3%A4re_Str%C3%B6mung
[27] Quelle: „Einführung in die Physik, Band 1: Mechanik und Wärme“ von Sexl, Raab und Streeruwitz, 2002
[28] Quelle: http://www.systemdesign.ch/index.php?title=Gesetz_von_Bernoulli

Herleiten lässt es sich folgendermassen: Betrachtet man ein stationäre Strömung in einer Röhre[29] während eines kurzen Zeitintervalls, so ist die durchströmende Masse Δm_1 im ersten Querschnitt A_1 gleich gross wie die durchströmende Masse Δm_2 im zweiten Querschnitt A_2 (siehe Skizze 1). Dies folgt aus der Tatsache, dass bei einer inkompressiblen, stationären Strömung die gesamte Masse im betrachteten Stück nicht ändern kann. Daraus folgt, dass das durchströmende Volumen ΔV in A_1 und A_2 gleich ist.

Da die Fliessgeschwindigkeit in der Stromröhre nicht konstant ist, ändert sich auch die kinetische Energie der Flüssigkeit während dem Durchströmen des Rohres.

$$\Delta E_{kin} = E_{kin2} - E_{kin1} = \frac{1}{2}\Delta m {v_2}^2 - \frac{1}{2}\Delta m {v_1}^2 = \frac{1}{2}\Delta V \rho ({v_2}^2 - {v_1}^2) \quad (3.7.1)$$

Der Energiesatz sagt, dass die Änderung der kinetischen Energie gleich der Arbeit ist, die an der Flüssigkeit verrichtet wurde. In der betrachteten Situation sind dies nur die vom Druck und der Gravitation verrichtete Arbeit W_{Druck} und W_{Grav}.

$$\Delta E_{kin} = W_{Druck} + W_{Grav} = \rho \Delta V + \Delta m g h \quad (3.7.2)$$

Nach einsetzen der Arbeit und umformen ergibt sich:

$$\frac{1}{2}\Delta V \rho ({v_2}^2 - {v_1}^2) = (P_1 - P_2)\Delta V - \rho \Delta V g (h_2 - h_1) \quad (3.7.3)$$

$$\frac{1}{2}\rho ({v_2}^2 - {v_1}^2) = (P_1 - P_2) - \rho g (h_2 - h_1) \quad (3.7.4)$$

Durch umformen erhält man das Gesetz von Bernoulli:

$$\frac{\rho}{2}{v_1}^2 + \rho g h_1 + P_1 = \frac{\rho}{2}{v_2}^2 + \rho g h_2 + P_2 \quad (3.7.5)$$

Ein Spezialfall dieses Gesetzes ist das Ausflussgesetz von Torricelli, welches ich hier an einem Beispiel kurz erklären werde. Dieses Gesetz besagt folgendes:[30]

Die Ausflussgeschwindigkeit einer Flüssigkeit aus einem gefüllten Behälter mit der Höhe *h* ist gleich gross wie die Geschwindigkeit, die ein Wassertropfen nach dem freien Fall aus derselben Höhe *h* erreicht hätte (siehe Skizze 2). In Formeln bedeutet das:

[29] Quelle Skizze 1:http://web.physik.rwth-aachen.de/~fluegge/Vorlesung/PhysIpub/Exscript/9Kapitel/Image119.gif
[30] Quelle: http://www.systemdesign.ch/index.php/Ausflussgesetz_von_Torricelli

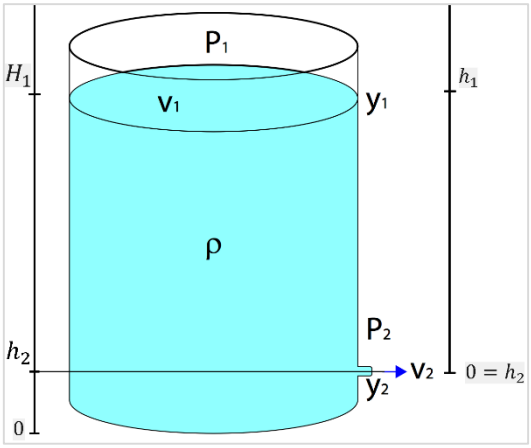
Energieerhaltung im freien Fall: $mgh = \frac{1}{2}mv^2$ (3.7.6)

Nach v aufgelöst: $v = \sqrt{2gh}$ (3.7.7)

Das Gesetz kann ebenfalls als Spezialfall der Bernoulli-Gleichung aufgefasst werden:

Bernoulli-Gleichung: $\frac{\rho}{2}{v_1}^2 + \rho g H_1 + P_1 = \frac{\rho}{2}{v_2}^2 + \rho g h_2 + P_2$ (3.7.8)

Die linke Seite von (3.7.8) beschreibt die Verhältnisse an der Wasseroberfläche, die rechte Seite beschreibt die Zustände im ausfliessenden Strahl.



Skizze 2: Situation
(y_1 und y_2 entsprechen den Höhen h_1 und h_2)

In der betrachteten Situation[31] ist die Geschwindigkeit der Wasseroberfläche v_1 in guter Näherung gleich Null. Die Dichte ρ konstant, da Wasser inkompressibel ist. Die Drücke entsprechen beide dem Luftdruck P_L, da P_1 an der Wasseroberfläche liegt und P_2 im freien Strahl ebenfalls dem Luftdruck entspricht. Die Gleichung vereinfacht sich, wenn man $h_2 = 0$ wählt. H_1 wird dann durch h_1 ersetzt, was der Höhe zwischen der Wasseroberfläche und dem Ausfluss entspricht.

Nach Einsetzen der neuen Werte in (3.7.8) ergibt sich:

$$v_2 = \sqrt{2gh_1} \quad (3.7.9)$$

Mit dieser Formel lässt sich die Ausflussgeschwindigkeit in Abhängigkeit zur Füllhöhe setzen. Daraus lässt sich der Volumenstrom berechnen:

$$I_V = Av \quad (3.7.10)$$

A ist hier der Querschnitt des ausfliessenden Strahls. Dieser unterscheidet sich allerdings vom Rohrquerschnitt, weil der Strahl an einem offenen Ende verzerrt wird. Diese Tatsache ist mir bei Experimenten zur Bernoulli-Gleichung aufgefallen. Der effektive Strahlquerschnitt ist in etwa 50% kleiner als der Querschnitt des Rohres. Exakt lässt sich der Strahlquerschnitt jedoch nicht bestimmen, wodurch eine gewisse Ungenauigkeit beim Dosieren entsteht.

Der Strahlquerschnitt lässt sich näherungsweise mit der Beziehung $A = \frac{I_v}{v}$ bestimmen, wenn der Volumenstrom und die Ausflussgeschwindigkeit bekannt sind. Dazu muss man den Volumenstrom

[31] Quelle Skizze 2: http://aplusphysics.com/courses/honors/fluids/images/water_jug_diagram.png

experimentell bestimmen, indem man das pro Zeitintervall austretende Volumen misst. Die Ausflussgeschwindigkeit lässt sich mit $v = \sqrt{2gh}$ berechnen.

Mit diesen Angaben ist es möglich, das austretende Volumen innerhalb eines Zeitintervalls zu berechnen. Der Volumenstrom $I_V = Av$ multipliziert mit dem Zeitintervall ergibt das ausgeflossene Volumen:

$$V = Avt \tag{3.7.11}$$

Daraus lässt sich die Zeit berechnen, die benötigt wird, um ein bestimmtes Volumen ausfliessen zu lassen. Diese Formel werde ich später in der Arbeit wieder aufgreifen:

$$\boldsymbol{t = \frac{V}{Av}} \tag{3.7.12}$$

3.8. Hygiene im Umgang mit Lebensmitteln

Beim Umgang mit Lebensmitteln muss auf viele wichtige Details geachtet werden. Einige davon möchte ich hier zeigen und erklären.

Da Lebensmittel sehr intensiv mit Menschen in Kontakt kommen, ist es wichtig, diese nicht mit schädlichen Substanzen zu verunreinigen oder verderben zu lassen. Um das für alle käuflichen Produkte sicherzustellen, gibt es viele unabhängige Prüfstellen und Qualitätssiegel. Um Verunreinigungen zu minimieren, sollte man auf lebensmittelechte Materialen setzen.

Es gibt verschiedene Ansätze, das Verderben von Lebensmitteln zu verhindern. Der Einfachste ist das Austauschen von alten Lebensmitteln durch frische. UV-Lampen mit einer Wellenlänge von 254 Nanometer können Bakterien abtöten[32], was zu einer längeren Haltbarkeit der Lebensmittel führt. Durch ständiges Kühlen der Lebensmittel kann das Bakterienwachstum ebenfalls verlangsamt und deren Haltbarkeit gesteigert werden.

Auch eine Verunreinigung durch die Kontaktoberflächen der Lebensmittel kann sehr gefährlich werden. Bei der Wahl der Materialien für Lagerbehälter und Transport ist darum besondere Vorsicht geboten. Insbesondere bei sauren Lebensmitteln kommt es zu chemischen Reaktionen an den Oberflächen der Kontaktflächen. Auf diese Effekte ist besonders zu achten, weil Coca Cola mit einem pH-Wert von 2.5, Fanta mit 3.0 und Fruchtsäfte mit etwa 3.5 sehr saure Flüssigkeiten sind[33].

[32] Quelle: http://www.wasserinbayern.de/uv-anlagen/wissenswertes/
[33] Quelle: http://vsa.labeaux.ch/docs_public/03%20MT%20pH-Wert%20DP.pdf

Bei Kunststoffen besteht die Gefahr, dass sich schädliche Substanzen ablösen oder dass das Material zu quellen[34] beginnt. Bei diesem Vorgang dehnt sich der Kunststoff stark aus, was in beweglichen Teilen - wie etwa bei Ventilen - schwerwiegende Konsequenzen haben kann. Um solche Effekte zu verhindern, eignen sich Kunststoffe wie PE[35] (Polyethylen) und FKM[36] (Fluor-Kautschuk).

Bei Metallteilen ist das Ablösen von giftigen Schwermetallen die grösste Gefahr. Diese meist in Ionenform gelösten Teilchen sind für den menschlichen Organismus sehr schädlich. Die Anfälligkeit für ein Ablösen von Schwermetallen ist bei allen Metallen unterschiedlich hoch. Hier eignet sich rostfreier Edelstahl besonders gut und ist darum auch ein Standard in der Lebensmittelindustrie. Auf den Gebrauch von anderen Metallen sollte wenn möglich verzichtet werden.

Bei Maschinen, die Lebensmittel direkt und für den Kunden ersichtlich verarbeiten, kommt eine weitere Schwierigkeit hinzu. Alles, was sich gegen Bakterien und andere schädliche Vorgänge im Automaten richtet und nicht gut versteckt ist, macht den Kunden auf diese Gefahren aufmerksam. Der Benutzer ist dadurch stark verunsichert und zweifelt die Qualität des Produkts an.

[34] Quelle: S. 53 in „Beständigkeit von Kunststoffen, Band 1“ von Gottfried W. Ehrenstein und Sonja Pongratz, 2007
[35] Quelle: http://de.wikipedia.org/wiki/Polyethylen
[36] Quelle: http://www.freudenberg-process-seals.de/ecomaXL/get_blob.php?name=FKM_de.pdf

4. Planung

In diesem Kapitel wird der Prozess der Planung beschrieben. Eine durchdachte Planung ist bei einem solchen Vorhaben von grosser Bedeutung, da es für viele Probleme unterschiedliche Lösungsansätze gibt, die aufeinander abgestimmt werden müssen.

4.1. Entwicklung des Konzepts

In diesem Abschnitt wird der Verlauf der ersten Planungsphase beschrieben. Es begann mit der Idee einer Bar, die ohne Barkeeper Cocktails zubereitet. Eine Bar, in der ein Glas von Flasche zu Flasche fährt und sich die Zutaten für den bestellten Drink abholt, dann mit Eiswürfeln gefüllt wird und schliesslich, kräftig durchgeschüttelt, einen fertigen Cocktail bereitstellt. Hier möchte ich erklären, wie aus dieser unrealistischen Vorstellung ein ausgereiftes Konzept wurde.

Eine vorgegebene Anforderung an den Automaten ist, ohne Wartung rund um die Uhr benutzbar zu sein. Um dies zu ermöglichen, müssen Zutaten gekühlt werden, damit sie nicht verderben. Eine solche Kühlung lässt sich verschieden realisieren. Sowohl Peltier-Elemente[37], ein Kompressor-Kühlsystem eines Kühlschranks oder flüssiger Stickstoff wären denkbar. Da flüssiger Stickstoff sehr unpraktisch zu handhaben ist, fällt diese Idee weg. Wegen der dauerhaften Kühlung muss der Stromverbrauch möglichst gering sein. Eine Kühlung kann nur bei sehr guter Isolierung, wie man sie in Kühlschränken findet, effizient betrieben werden. Eine ähnlich gute Isolierung selbst zu bauen, wäre sehr aufwendig und teuer. Deshalb sollte sich der Automat in einem Kühlschrank befinden. Um dem Kunden das interessante Innenleben präsentieren zu können, sollte ein Spezial-Kühlschrank mit Sichtscheibe verwendet werden. Als Schnittstelle sollte ein Touchscreen verwendet werden, um eine intuitive Bedienung zu garantieren.

Damit der Automat einen Drink aus dem Kühlschrank befördern kann, werden Schlauchleitungen durch ein kleines Loch in der Kühlschrankwand gelegt. Würde ein grosser Ausgang für ein Glas an der Wand des Kühlschranks verwendet, wäre der Kälteverlust viel zu hoch.

Das Grundkonzept des Automaten ist also: ein verglaster Kühlschrank mit integrierter Dosierung, Schlauchleitungen und einem Touchscreen als Schnittstelle. Die Planung der Dosiermechanik wird später in der Arbeit erläutert, da dieser Prozess sehr kompliziert ist.

[37] Ein Peltier-Element ist eine elektrische Wärmepumpe.
Quelle: http://www.deltron.ch/pdf/produkte/peltier/peltier-element_kurz_erklaert_d.pdf

4.2. Aufbau der Software

Bevor die Programmierarbeiten beginnen konnten, musste der Aufbau der Software geplant sein. Neben der groben Programmstruktur sollte auch das grundlegende Designkonzept des GUI vor dem Programmieren ausgearbeitet sein.

Bereits vorgegeben war der Touchscreen als intuitive Eingabemöglichkeit und der Einsatz einer Software-gesteuerten Dosierung, um die Zutaten der Drinks abzumessen. Aufgrund dieser Vorgaben konnte dann ein ungefähres Bild davon gemacht werden, wie die Struktur des Programms in etwa aussehen könnte.

Zuoberst ist die Schnittstelle zum Nutzer, das sogenannte GUI, welches Informationen über die Bestellung des Kunden sammeln kann. Direkt unter dieser Schnittstelle befindet sich der Rest des Programms. Die vom GUI gelieferten Daten werden zu Befehlen weiterverarbeitet und an die Dosierungsfunktion geleitet. Die Software ist demnach in drei Teile aufgeteilt: das GUI, die Datenverarbeitung und die Dosierung.

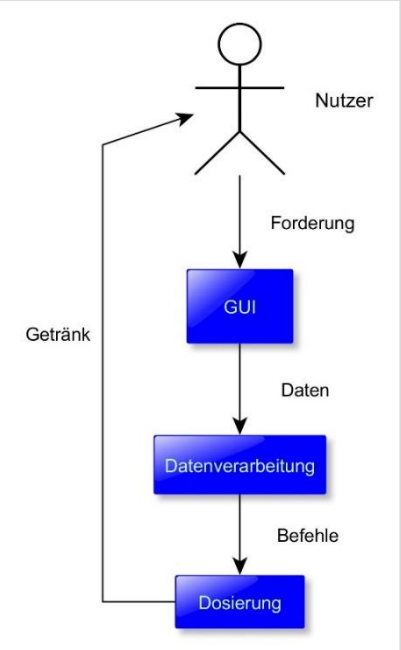


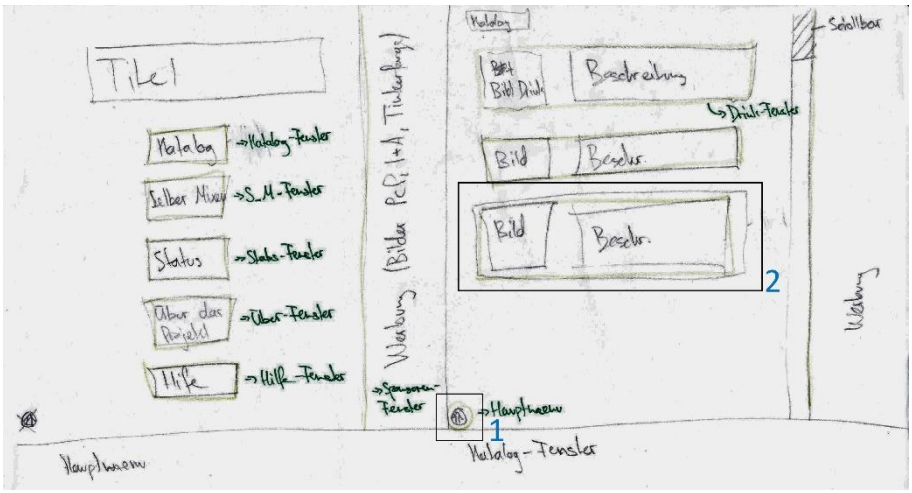
Diagramm 1: Programmstruktur

Um ein GUI zu erstellen, gibt es spezielle Tools, bei denen man die gewünschte Oberfläche mitsamt Verknüpfungen zwischen den Fenstern aufzeichnen kann, ohne dabei programmieren zu müssen. Das Tool erstellt aus der Zeichnung dann selbstständig den Programmcode. Ich habe auf den Einsatz eines solchen Tools aber verzichtet, da der erzeugte Quellcode bis zu achtmal länger ist als der entsprechende handgeschriebene Code.

Da handgeschriebener Code verwendet wird, kann das tiefe Verständnis des GUI-Codes dazu genutzt werden, die Datenverarbeitung direkt darin zu intergieren. Dadurch eröffnen sich völlig neue Möglichkeiten von Funktionen, die man in einem generierten Code nur sehr schwer hätte umsetzten können.

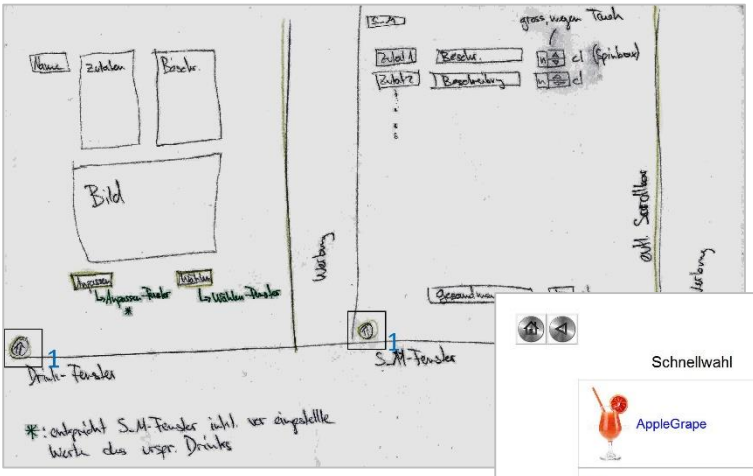
Das Grundkonzept und Aussehen des GUI wurden anschliessend geplant. Da eine grafische Oberfläche möglichst viele Informationen so übersichtlich wie möglich darstellen soll, ist das Verwenden von mehreren Seiten beziehungsweise Fenstern unverzichtbar. Dadurch schafft man mehr Anzeigefläche und die Informationskapazität steigt bei gleichbleibender Informationsdichte erheblich an. Der Wechsel zwischen verschiedenen Fenstern ist programmiertechnisch ziemlich anspruchsvoll. Wie das gelöst wurde, wird später in der Arbeit ausführlich erläutert.

Die nachfolgenden Skizzen zeigen einige der ersten Designentwürfe der grafischen Benutzeroberfläche.



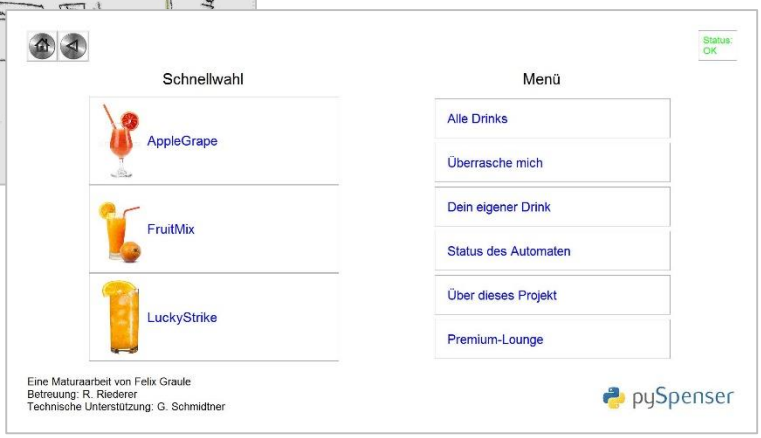
Skizze 3: Hauptmenü (links) und „Drink-Katalog“ (rechts)

Diese Skizze zeigt einen Entwurf des Hauptmenüs und des „Drink-Katalogs“. Spannenderweise haben es viele Konzepte bis in die finale Version der Software geschafft, so etwa die Idee, die Bedienung mit einheitlichen Navigations-Buttons (siehe Markierung 1) zu erleichtern und der Aufbau der Buttons im Katalog (siehe Markierung 2). Dieser besteht wie in der finalen Version aus einem Bild des Drinks und einem Text, der den Drink kurz beschreiben soll.



Skizze 4: Entwurf des „Drink“-Fensters (rechts) und des „Dein eigener Drink“-Fensters (links)

Bild 9: Screenshot des finalen GUI



4.3. Aufbau der Hardware

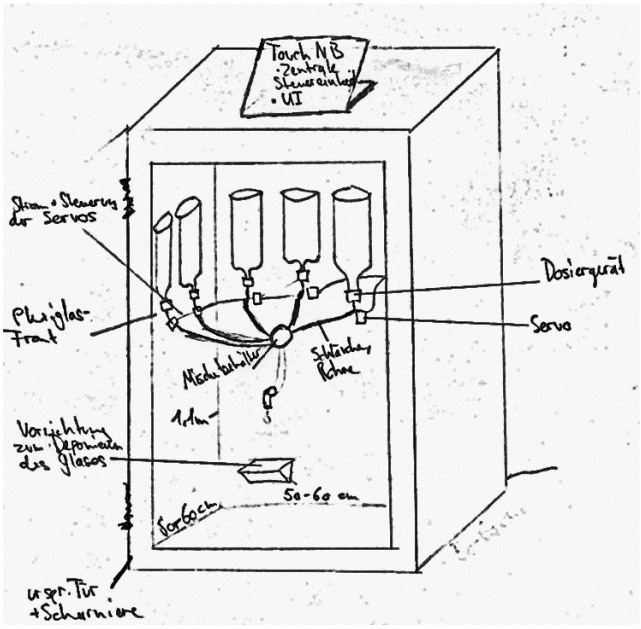
Dieser Abschnitt erklärt die Planung des Aufbaus der Hardware. Dazu werden nachfolgend zusätzlich zum Text einige Skizzen der Hardware und Schaltpläne gezeigt.

Um den Roboter so kompakt wie möglich konstruieren zu können, sollte neben den zu kühlenden Zutaten auch die gesamte Elektronik im Innern untergebracht sein. Der Automat ist grundsätzlich in zwei Teile aufgetrennt: Im oberen Teil sollten sich die Zutaten und die Dosiermechanik befinden, im unteren Teil die elektronische Steuerung. Oben wären demnach die Flaschen, die Ventile und die Trichter-Positionierung eingebaut, unten befänden sich der Zentralrechner und die Mikrocontroller.

Als Trennelement zwischen den Abteilen wird eine Plexiglasscheibe verwendet. Einerseits soll die Aufteilung die Elektronik vor Flüssigkeiten schützen und muss darum mit Silikon abgedichtet sein. Andererseits soll die Abtrennung die von der Elektronik erzeugte Abwärme zurückhalten, um die Zutaten bei möglichst tiefer Temperatur lagern zu können. Darum sollte die Auftrennung möglichst wenig Luftaustausch zulassen. Dies ist auch wichtig, weil die Luftfeuchtigkeit bei tiefen Umgebungstemperaturen kondensieren kann und es so zu einem Kurzschluss an der Steuerungselektronik kommen könnte.

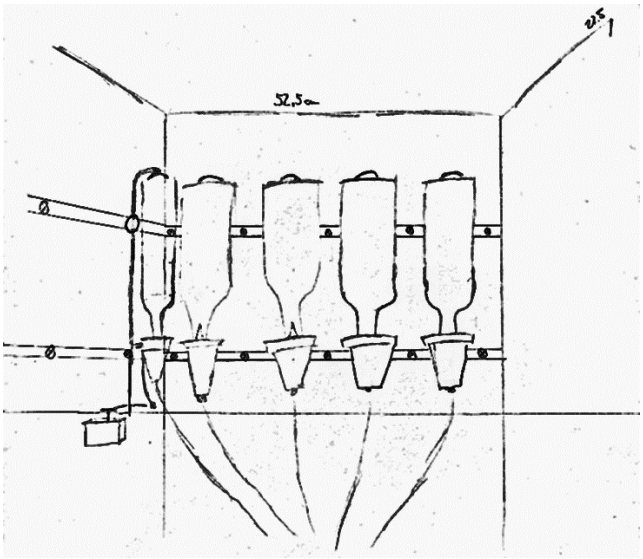
Der Automat sollte über einige Leuchteffekte verfügen, um ästhetisch wertvoll und modern auszusehen. Die Beleuchtung besteht aus folgenden Komponenten: kleine LED sollen den Kunden auf bestimmte Ereignisse im Gerät aufmerksam machen, wie etwa das Öffnen eines Ventils. Etwas grössere und leuchtstärkere Lampen sollten den gesamten Automaten effektvoll ausleuchten.

Nun soll anhand einiger Skizzen die Planung der Hardware aufgezeigt werden. Die ersten Skizzen der Innenkonstruktion sind sehr grobe Darstellungen mit Bleistift, spätere Skizzen wurden mit *Google SketchUp* erstellt. Danach folgen Schaltpläne der beiden benötigten Stromkreise. Diese Pläne wurden mit *CircuitLab* entworfen.



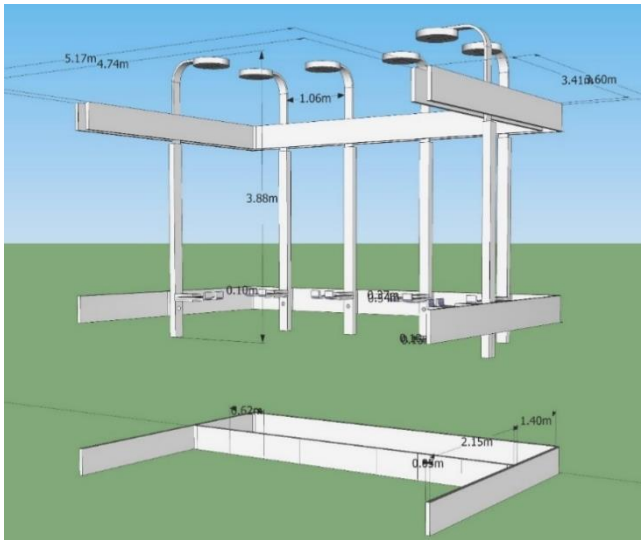
Skizze 5: Frühe Skizze der Gesamtansicht des Roboters

Diese Abbildung zeigt eine der ersten Skizzen des Projekts und unterscheidet sich daher deutlich vom fertigen Roboter. Die Zutaten werden in einem isolierten Quader gekühlt, ein Konvertible-Notebook dient als Zentralrechner und Eingabegerät zugleich. Die Front des Automaten besteht aus Plexiglas.



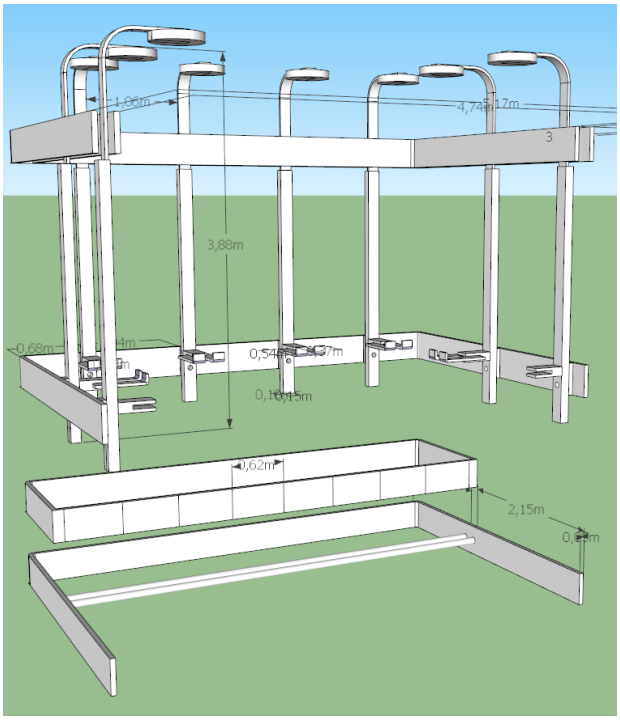
Skizze 6: Bleistiftskizze der Flaschenbefestigung

Diese Bleistiftskizze zeigt bereits das Konzept der Flaschenbefestigung an U-förmigen Metallschienen, wie sie später verwendet wurde. Die Dosiermechanik besteht bei dieser Skizze aus einem Mechanismus von Servos und Portionierern, was sich aber als zu fehleranfällig erwiesen hat.



Skizze 7: 3D-Skizze der Innenkonstruktion

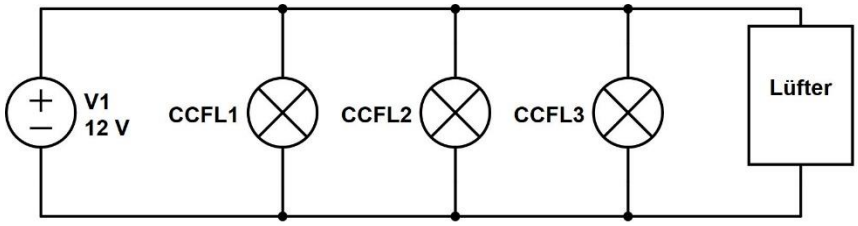
Dieses Bild zeigt die Befestigung der Flaschenhalterungen und der Ventile. Es ist klar ersichtlich, dass die Grundidee der Befestigung ein modulares System aus verschiedenen, U-förmigen Rahmen ist. Diese sollen in den Kühlschrank eingeschoben werden. Die Ventile der Dosiermechanik werden an der Querverstrebung der untersten Schiene befestigt. Bei dieser Skizze fehlt noch die Trichter-Positionierung.



Skizze 8: Skizze der finalen Innenkonstruktion

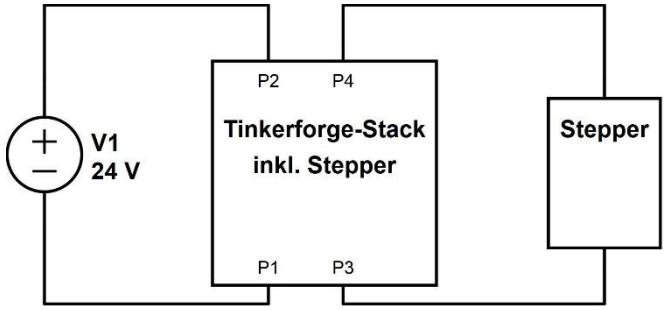
Diese Skizze zeigt den finalen Bauplan der Innenkonstruktion, der beinahe baugleich umgesetzt wurde. In dieser Skizze sind Zutatenlagerung, Dosiermechanik und Trichter-Positionierung eingezeichnet.

Im Folgenden sollen die verschiedenen Stromkreise zur Steuerung und Stromversorgung der Aktoren und Sensoren anhand von Schaltplänen erklärt werden.



Skizze 9: Schaltplan des 12 Volt-Stromkreises

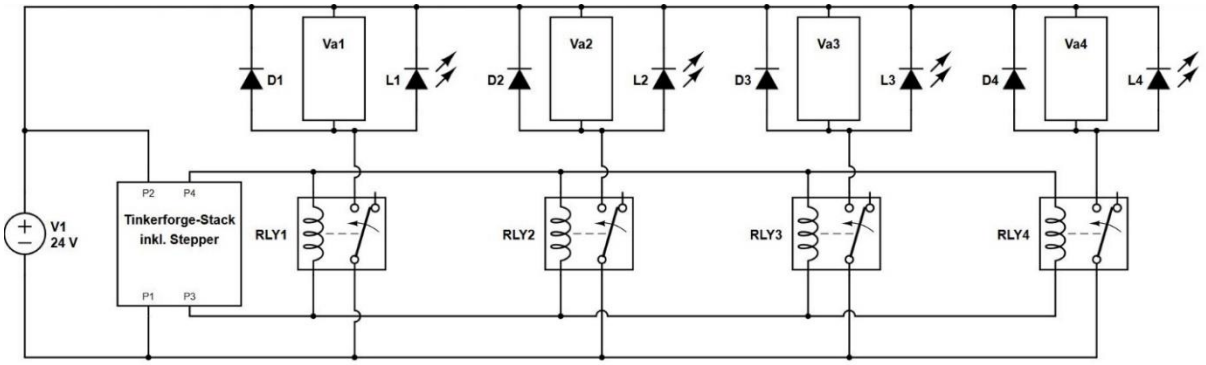
Der Schaltplan zeigt den Versorgungsstromkreis der Leuchtstoffröhren *(CCFL)* und des Lüfters. Die Spannungsquelle *V1* liefert dazu 12 Volt. Alle Elemente sind parallel geschaltet.



Skizze 10: Schaltplan des Stack und Stepper-Stromkreises

In diesem Schaltplan befinden sich genau genommen zwei verschiedene Stromkreise: der linke Stromkreis versorgt den Stack[38] mit Strom, hier liegt die angelegte Spannung konstant bei 24 Volt. Der rechte Stromkreis versorgt vom Stack aus den Stepper-Motor mit Strom, wobei die hier anliegende Spannung von der auszuführenden Fahrbewegung abhängt und vom Mikrocontroller bestimmt wird.

[38] Der Begriff *Stack* wird in Kapitel 4.6.4. erklärt.



Skizze 11: Schaltplan des Ventil-Schaltkreises

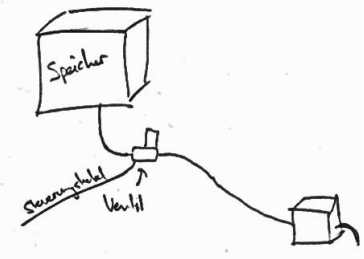
Dieser Schaltplan zeigt wiederum zwei verschieden Stromkreise, wobei der erste im Automat nicht ohne weiteres erkennbar ist. Er versorgt die Relais mit Strom zum Schalten, was beim Tinkerforge-System direkt über dem Stack gemacht wird. Der zweite und wichtigste Stromkreis versorgt die Ventile *Va1* bis *Va8* und die zugehörigen LEDs *D1* bis *D8* mit Strom (zur besseren Übersicht sind hier nur vier der acht Komponenten abgebildet). Dieser Stromkreis kann durch die Relais partiell geschlossen werden, wodurch sich das entsprechende Ventil öffnet und die LED aufleuchtet.

Da es sich bei Magnetventilen um elektrische Spulen und damit um induktive Lasten handelt, ist in diesem Stromkreis noch eine spezielle Schutzbeschaltung verbaut. Die Spulen der Ventile erzeugen beim Ausschalten enorm hohe Spannungsspitzen, diese reichen bis über 100 Volt und können die LEDs und Relais beschädigen. Um dies zu verhindern, ist eine Schutzdiode parallel zur induktiven Last eingebaut. Diese Diode kappt jegliche Spannung oberhalb eines bestimmten Wertes und schütz somit die übrigen Komponenten vor zu hoher Spannung.

4.4. Dosiermechanik

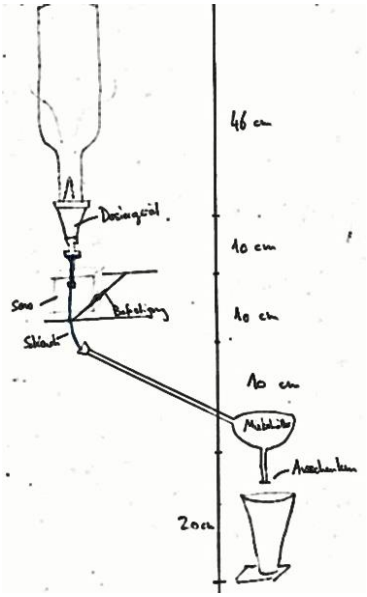
In diesem Abschnitt geht es darum, wie die Dosiermechanik funktioniert und welche anderen Konzepte zur Auswahl standen.

Die Flüssigkeiten werden mit Magnetventilen dosiert, wobei die Öffnungszeit die ausfliessende Menge bestimmt. Die Magnetventile werden mit Relais-Bricklets[39] gesteuert, welche einen 24 Volt-Stromkreis schliessen können und das Ventil folglich öffnen. Der nötige Druckunterschied zwischen Ventileingang und Ventilausgang entsteht allein durch die Gravitation.



Skizze 12: Vereinfachte Skizze der Dosiermechanik mit Magnetventil

Zu Beginn der Planung wurde eine komplizierte Dosiermechanik mit Servomotoren und Dosieraufsätze entworfen. Bei diesem Dosierkonzept aktiviert ein Servomotor durch Ziehen einen mechanischen Dosierer und es fliesst eine festgelegte Menge Flüssigkeit aus. Diese Idee stellte sich aber schnell als fehleranfällig heraus und wäre mit grossem Konstruktionsaufwand verbunden gewesen. Auch Dosierungsmechanismen mit Pumpen wurden ausgearbeitet. Bei diesen wird die dosierte Menge wie bei der finalen Dosiermechanik nur durch die Zeit bestimmt, während der die Pumpen laufen. Pumpen sind jedoch entweder laut oder sehr teuer, weswegen dieser Entwurf verworfen werden musste.



Skizze 13: Dosiermechanismus mit Servomotor und Dosieraufsatz

Ein Überlaufsystem wäre ebenfalls möglich gewesen: ein vorgegebener Raum wird am Ein- und Ausfluss von Ventilen verschlossen. Zum Dosieren wird der Einfluss geöffnet und erst wieder geschlossen, wenn der Raum ganz gefüllt ist. Anschliessend wird das untere Ventil geöffnet und die dosierte Menge fliesst aus.

In einen Drink gehören eigentlich Eiswürfel, weshalb auch dazu Konzepte entworfen wurden. Bei niedrigen Kosten ist das automatische Ausgeben von Eiswürfeln aus folgenden Gründen sehr anspruchsvoll: das Herstellen von Eis mit flüssigem Stickstoff ist zu gefährlich, teuer und aufwendig. Darum hätten bereits gefrorene Eiswürfel in die Drinks gegeben werden müssen. Die Eiswürfel müssen in einem Eisfach gelagert werden, das heisst an einem Ort, an dem Temperaturen unter dem Gefrierpunkt herrschen. Im Zusammenspiel mit der Luftfeuchtigkeit würde aber jeder Mechanismus ständig vereisen. Diese Funktion wurde darum aufgegeben.

[39] Der Begriff *Bricklets* wird im Kapitel 4.6.4. erklärt

4.5. Hygienische Überlegungen

In diesem Abschnitt werden einige Überlegungen zur Hygiene im Automaten beschrieben. „Hygienisch“ bedeutet einerseits, dass keine gesundheitsschädlichen Stoffe oder Organismen vorhanden sind, andererseits beschreibt „hygienisch“ aber auch einen bestimmten optischen Eindruck: ein Automat, der im Kontakt zu Kunden steht, muss darum nicht nur hygienisch sein, sondern auch dementsprechend aussehen.

Damit die Zutaten im Automaten über einen langen Zeitraum geniessbar bleiben, müssen sie gekühlt gelagert werden. Um die Haltbarkeit der Zutaten und die Hygiene im Automaten zu erhöhen, könnten auch Luftfilter, UV-Lampen und Spülmechanismen eingesetzt werden.

Von diesen erweiterten Möglichkeiten wird nur die letzte angewendet, wobei lediglich der Trichter gespült wird. Die Ventile ebenfalls zu spülen, wäre zwar sinnvoll und effektiv, die Umsetzung ist aus folgenden Gründen aber zu teuer: Will man die Ventile spülen, so muss vor jedem Ventil eine T-Verbindung zum Spülkasten vorhanden sein. Der hydrostatische Druck im Zutatenspeicher und Spülkasten variiert abhängig vom Füllstand. Darum ist es möglich, dass Spülflüssigkeit in die Zutaten hochgedrückt wird oder umgekehrt. Um das zu verhindern bräuchte es vor jeder T-Verbindung zum Spülkasten ein zweites Magnetventil, was aus finanziellen Gründen nicht in Frage kam. Ausserdem hätte dieses Spülsystem den Konstruktionsaufwand stark erhöht und viel Platz benötigt.

Luftfilter zu verwenden ist relativ ineffizient, da es beim Wechseln von Flaschen zu einem grossen Austausch mit nicht filtrierter Luft kommt. UV-Lampen sind ebenfalls nicht effektiv, weil die verwendeten Flüssigkeiten teils nicht durchsichtig sind. Das bakterientötende UV-Licht erreicht also nur den Rand der Flaschen und somit nur einen kleinen Anteil der Zutaten.

4.6. Wahl der technischen Komponenten

In diesem Abschnitt wird aufzeigt, für welche technischen Vorgehensweisen und Produkte ich mich entschieden habe. Es wird auch klargemacht, weshalb ich mich gerade für diese Komponenten entschieden habe und wo die Schwierigkeiten lagen.

4.6.1. Python[40]

Python ist eine High-Level-Programmiersprache, bei der besonderen Wert auf einen leserlichen Programmcode gesetzt wird. Deswegen gibt es im Gegensatz zu *C++* oder *Pascal* keine besonderen Zeichen zur Blocktrennung. Blöcke werden durch Einrückungen voneinander getrennt. Ausserdem gibt es vergleichsweise wenige Schlüsselwörter.

Entwickelt wurde Python zu Beginn der 90-er Jahre von Guido van Rossum in Amsterdam. Der Name der Sprache stammt von der englischen Comedygruppe *Monty Python*, nicht wie oft vermutet vom gleichnamigen Schlangen-Taxon. Heute wird diese Sprache oft für Einsteiger empfohlen, weil sie einfach zu erlernen ist. Python wird in Software von *Google* und vom Videoportal *Youtube* verwendet.

Python unterstützt mehrere Programmierparadigmen: man kann objektorientiert, strukturiert, funktional und aspektorientiert programmieren. Python verfügt ausserdem über eine sehr grosse Standardbibliothek, was die Einsatzmöglichkeiten von Python-Programmen sehr vielseitig macht. Es gibt sowohl umfassende Erweiterungen zur Programmierung von Web-Anwendungen, als auch eine integrierte Erweiterung zum Aufbau einer graphischen Oberfläche. Diese trägt den Namen *Tkinter*, setzt auf das *Tk*-Toolkit und lässt sich durch zusätzliche Erweiterungen wie *ttk* und *tix* auch für visuell ansprechende GUIs verwenden.

Als Entwicklungsumgebung stellt Python die sogenannte IDLE zur Verfügung, welche das Verfassen und Ausführen von Code ermöglicht. Die IDLE besteht aus einer Textumgebung und einer *Shell*, in welcher der Programmcode ausgeführt wird. Der benötigte *Compiler* ist ebenfalls direkt im Programm integriert.

Ich habe mich für Python entschieden, weil ich bereits einige Grundkenntnisse der Sprache hatte und Python sehr übersichtlich und einfach zu verstehen ist. Ausserdem gibt es bei Python eine grosse Internet-Communitiy, welcher man in Foren wie *stackoverflow.com* Fragen stellen kann.

[40] Quellen: http://en.wikipedia.org/wiki/Python_(programming_language)

4.6.2. Tkinter[41]

Tkinter ermöglicht das Aufbauen von *Tk*-basierten Oberflächen mit Python und wurde von Steen Lumholt und Guido van Rossum entwickelt. *Tk* ist auf vielen verschieden Betriebssystemen verfügbar, darunter auch *Windows* und *Macintosh*. Entwickelt wurde *Tk* von John Ousterhout.

Ich werde hier nicht weiter auf die programmiertechnischen Details von *Tk* und *Tkinter* eingehen, eine kurze Einführung wäre bereits sehr lang. Als umfassende Einführung empfehle ich das englische Lehrbuch *„An Introduction to Tkinter“* von Fredrik Lundh.

Ich habe Tkinter verwendet, weil die GUI-Bibliothek verständlich ist und man zügig graphische Oberflächen entwickeln kann.

4.6.3. ttk[42] und tix[43]

Mit der *Tkinter*-Erweiterung *ttk* erhält der Entwickler Zugriff auf sogenannte *Themed-Widgets*. Ein Widget ist das kleinste Element einer graphischen Oberfläche, also beispielsweise ein Textfeld oder ein Button. Die Grundidee von *ttk* ist es, die Implementierung der Funktion und des Aussehens eines Widgets soweit wie möglich aufzutrennen. Die sogenannten *Styles* erlauben es, das Design eines Widgettyps einmalig festzulegen und später wiederzuverwenden.

Die Erweiterung *tix* fügt der Tkinter-Bibliothek einige zusätzliche Widgets hinzu. Von besonderen Interesse ist das Control-Widget, welches eine intuitive Eingabe von Zahlen via Touchscreen ermöglicht.

Diese beiden Erweiterungen habe ich verwendet, um den Funktionsumfang und die Designmöglichkeiten von Tkinter zu erhöhen.

---

[41] Quellen: „An Introduction to Tkinter“ von Fredrik Lundh, 1999 ;
http://docs.python.org/3.3/library/tkinter.html
[42] Quelle: http://docs.python.org/3.3/library/tkinter.ttk.html#tkinter.ttk.Widget
[43] Quelle: http://docs.python.org/3.3/library/tkinter.tix.html

4.6.4. Tinkerforge

Tinkerforge™ ist ein 2011 gegründetes Start-Up-Unternehmen aus Deutschland. Unter der Leitung von Bastian Nordmeyer und Olaf Lüke hat die Firma eine Plattform aus Mikro-Controllern aufgebaut[44]. Diese besteht im Wesentlichen aus verschiedenen *Bricks* und *Bricklets.* Das Projekt ist Open-Source[45], es sind also alle Hardware- und Software-Teile quelloffen und für jeden einsehbar.

Mit Tinkerforge-Bausteinen lassen sich sehr vielseitige Projekte realisieren, vom Industrieroboter über Alltagshelfer bis zum Klimamessgerät ist alles machbar. Die Plattform befindet sich in Sachen Komplexität und Modularität in etwa zwischen Lego-Mindstorms® und Arduino®. Das System ist deutlich komplexer und vielseitiger als Lego-Mindstorms, andererseits aber auch einfacher und einschränkender als Arduino.

Bricks sind aufeinander steckbare Platinen (siehe Bild 10)[46], welche mit einem 32-bit ARM™-Mikrocontroller ausgestattet sind. Die Bricks übernehmen das Auswerten von Daten, das Steuern von mechanischen Komponenten und die Kommunikation zwischen PC, Bricks und Bricklets. Ein Stapel von mehreren Bricks wird als *Stack* bezeichnet und kann sehr vielseitige Aufgaben ausführen. Bricklets sind kleine Erweiterungsmodule wie Relais oder Sensoren. Sie werden mit den Bricks verbunden und erweitern deren Funktion.

Bild 10: Stack aus 4 Bricks und der Stromversorgung

Mit sogenannten *Master Extensions* ist es möglich, einen Stack um zusätzliche Schnittstellen wie WLAN oder Ethernet zu erweitern. Diverses anderes Zubehör ermöglicht eine gesicherte Stromversorgung und eine strukturierte Anordnung von Bricks und Bricklets.

Programmierbar sind die Tinkerforge-Bausteine mit nahezu allen gängigen Sprachen. Unterstützt werden neben *C*, *C++*, *Java* und *PHP* unter anderem auch *Ruby* und *Python*. Für das Ausführen von Software müssen die Bausteine ständig per USB, Ethernet oder WLAN mit einem Computer verbunden sein, was wohl der grösste Nachteil gegenüber dem autonomen Arduino ist.

[44] Quellen: http://de.wikipedia.org/wiki/Tinkerforge; http://www.golem.de/specials/tinkerforge/
[45] Quelle: http://de.wikipedia.org/wiki/Open_Source
[46] Quelle Bild 10: http://download.tinkerforge.com/press/media/brick_stack_back.jpg

Für mein Projekt werden die folgenden Tinkerforge-Komponenten verwendet[47]:

Bild 11: Master-Brick, Servo-Brick und Stepper-Brick

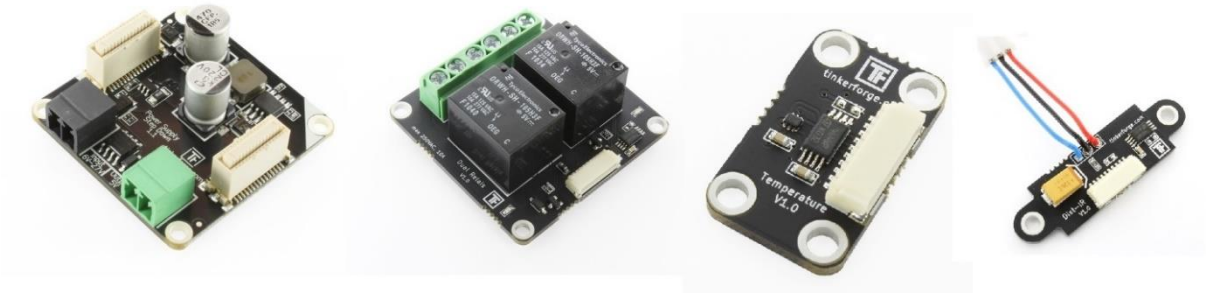


Bild 12: Step-Down Power Supply, Dual-Relay-Bricklet, Temperature-Bricklet und DistanceIR-Bricklet

Ich habe mich für das Tinkerforge-System entschieden, weil ich damit alle meine Ideen verwirklichen konnte und praktisch fand, alle Komponente fixfertig aus einer Hand bestellen zu können. Ausserdem spielten die guten Testberichte von *golem.de*[48] und *Chip.de*[49] eine Rolle bei der Entscheidungsfindung.

4.6.5. Arduino

Die Arduino®-Plattform stellt Soft- und Hardware für die Steuerung von elektrischen Schaltungen zur Verfügung, ist also wie Tinkerforge eine Mikrocontroller-Plattform. Wie bei Tinkerforge sind auch beim Arduino-Projekt alle Komponenten quelloffen.[50]

Die Hardware besteht meist aus einem Board, welches über analoge und digitale Ein- und Ausgänge, sogenannte *I/O-Pins*, in eine Schaltung eingebunden werden kann. Des Weiteren können auch sogenannte *Shields* auf das Board gesteckt werden, um dessen Funktionsumfang zu erweitern. Über die Pins können ausserdem Sensoren aller Art angeschlossen werden.

[47] Quellen Bild 11 und 12: http://www.tinkerforge.com/en/doc/_images/Bricks/; http://www.tinkerforge.com/de/doc/_images/Bricklets/
[48] Referenz: http://www.golem.de/news/tinkerforge-im-test-elektronik-zum-stapeln-1205-91624-4.html
[49] Referenz: http://www.chip.de/news/CHIP-AWARDS-2012-Das-sind-die-Preistraeger_54928401.html
[50] Quelle: http://de.wikipedia.org/wiki/Arduino-Plattform

Programmiert wird mit der Programmiersprache *Processing*. Der Code wird dabei in einer Entwicklungsumgebung geschrieben, anschliessend vom Compiler übersetzt und auf das Board geladen. Ab dann kann das Board ohne Computer verwendet werden, es ist also autonom einsetzbar.

In meinem Projekt kommen ein Arduino Uno-Board und ein Fingerabdruck-Sensor von Adafruit® zum Einsatz:

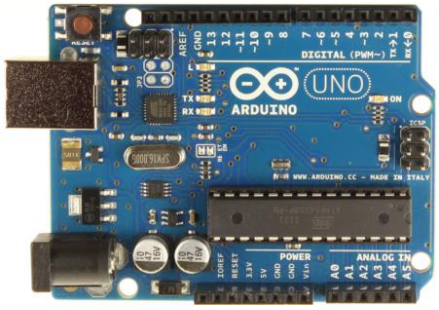


Bild 13: Arduino Uno-Board

Bild 14: Fingerabdruck-Sensor von Adafruit

Ich habe mich für diese Komponenten entschieden, da während der Internet-Recherche keine weiteren Open-Source-Komponenten zum gestellten Problem gefunden wurden und diese Einzelteile einen spannenden und vielversprechenden Lösungsansatz darstellten.

4.6.6. Ventilauswahl[51]

Beim Beschaffen des optimalen Ventils musste die Wahl oft gewechselt werden. Warum das so war und wie die Entscheidung schlussendlich zu Stande kam, wird in diesem Abschnitt erläutert.

Bei Ventilen unterscheidet man grundsätzlich zwischen den Anzahl Wegen, die es besitzt. Diesbezüglich war schnell klar, dass ein sogenanntes 2-Wege-Ventil benötigt wird. Diese Ventile besitzen einen Ein- und Ausgang und verfügen über zwei mögliche Stellungen („offen“ oder „geschlossen“). Ausserdem unterscheidet man die Stellung des Ventils, wenn es nicht betätigt wird. Ein *N.C. (engl.: „normally closed“)* ist in der Ruhestellung geschlossen, ein *N.O. (engl.: „normally opened“)* ist in der Ruhestellung geöffnet. Um den Stromverbrauch zu minimieren, sollten in Ruhestellung geschlossene Ventile (N.C.) verwendet werden.

Eine weitere grundlegende Eigenschaft von Ventilen ist die Art der Ansteuerung. Es gibt elektrisch und pneumatisch gesteuerte Ventile. Eine pneumatische Steuerung benötigt einen zusätzlichen Druckluft-Kreislauf, der neben dem unverzichtbaren elektrischen Kreislauf betrieben wird. Dieser Mehraufwand macht keinen Sinn, weil eine pneumatische Steuerung in meinem Fall keine Vorteile gegenüber einer elektrischen Steuerung bietet.

[51] Quelle: http://content2.smcetech.com/pdf/VDW_1_DE.pdf

Die Auswahl des Materials, aus dem die Ventile gefertigt sein sollten, stellte sich als sehr schwierig heraus. Man unterscheidet hier zwischen dem Gehäuse- und dem Dichtungsmaterial. Das Gehäuse ist meistens aus Metallen wie Messing oder Edelstahl, kann aber auch aus Kunststoffen wie PPS (Polyphenylensulfid) bestehen. Als Dichtungsmaterial wird meist Kunststoff verwendet, etwa NBR (Nitrit-Kautschuk) oder FKM (Fluor-Kautschuk). Zwischen all diesen Materialien gibt es aber grosse Unterschiede bezüglich Beständigkeit, Geschmacksabgabe und Preis, welche im Kapitel 3.8. „Hygiene im Umgang mit Lebensmitteln“ bereits erläutert wurden. Schlussendlich fiel die Wahl auf die teuren Materialien Edelstahl und FKM, da kein gesundheitliches Risiko eingegangen werden sollte.

Nun musste noch entschieden werden, von welchem Produzenten die Ventile stammen sollten. Ich habe dazu verschiedenen Firmen angeschrieben und mich wegen des freundlichen Feedbacks und einem grosszügigen Rabatt für SMC Pneumatik AG entschieden.

Die endgültige Wahl fiel auf das Magnetventil VDW31–5G–4–02–H–Q. Hier eine kurze Übersicht, wofür welche Angaben stehen, und ein Bild des bereits eingebauten Ventils.

Angabe	Bedeutung	Gewählter Wert
VDW	Produktfamilie, Einsatzbereich	Wasser, Luft und Vakuum
3	Nummer der Serie	Serie 30
1	Ventilzustand in Ruhestellung	N.C.
5G	Betriebsspannung und Stromanschluss	24 Volt, eingegossene Kabel
4	Nennweite der Ventilöffnung	4 Millimeter
02	Anschlussgrösse der Gewinde	1/4 Zoll
H	Gehäuse- und Dichtungsmaterial	Edelstahl und FKM
Q	CE-Zertifizierung	Vorhanden

Bild 15: Bereits eingebautes Magnetventil

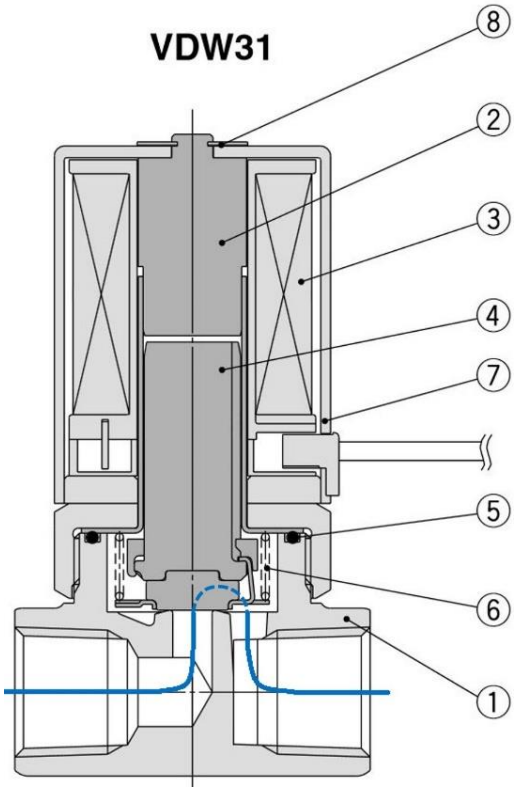
4.6.7. Aufbau und Funktionsweise des Ventils[52]

In diesem Abschnitt wird der Aufbau des verwendeten Ventils und dessen Funktionsweise erklärt.

Das Ventil besteht im Wesentlichen aus zwei Teilen, der Leitung und dem Magneten. Die Leitung verläuft gekurvt durch das Ventil, sie geht durch eine Öffnung in eine Umleitung, welche schliesslich wieder in die Leitung führt. In der nebenstehenden Skizze ist der Weg des durchfliessenden Wassers blau eingezeichnet.

Auf dieser Öffnung sitzt der magnetische Anker und verschliesst die Leitung. Im oberen Teil sitzt eine Spule mit Eisenkern. Fliesst durch die Spule ein elektrischer Strom entsteht ein Magnetfeld. Diese magnetische Anziehung hebt den Anker nach oben und die Öffnung wird frei. Das Medium kann nun durch das Ventil fliessen. Sobald die elektrische Spannung weggenommen wird, fällt das Magnetfeld zusammen und der Anker wird von den Rückstellfedern wieder auf die Öffnung gepresst.

Das Funktionsprinzip eines Magnetventils ist dem eines Relais sehr ähnlich. Beim Relais wird anstatt eines Ankers ein elektrischer Kontakt angezogen und anschliessend, um den Stromkreis wieder zu schliessen, wieder zurück gepresst.



1	Gehäuse	5	O-Ring (Gehäuse)
2	Kern	6	Rückstellfeder
3	Spule	7	Abdeckung
4	Anker	8	Klemmverschluss

Skizze 14: Technische Skizze des Ventilquerschnitts inklusive Legende

[52] Bildquelle Skizze 14: http://content2.smcetech.com/pdf/VDW_1_DE.pdf

4.7. Zusammenarbeit der Hardware

Das folgende Diagramm soll die Zusammenarbeit zwischen PC, Mikro-Controllern, Reglern und Sensoren verdeutlichen. In den Quadraten befinden sich die Komponenten, die Beschriftungen der Pfeile stellen Aktionen dar. Die Pfeilrichtung gibt an, in welche Richtung eine Aktion ausgeführt wird.

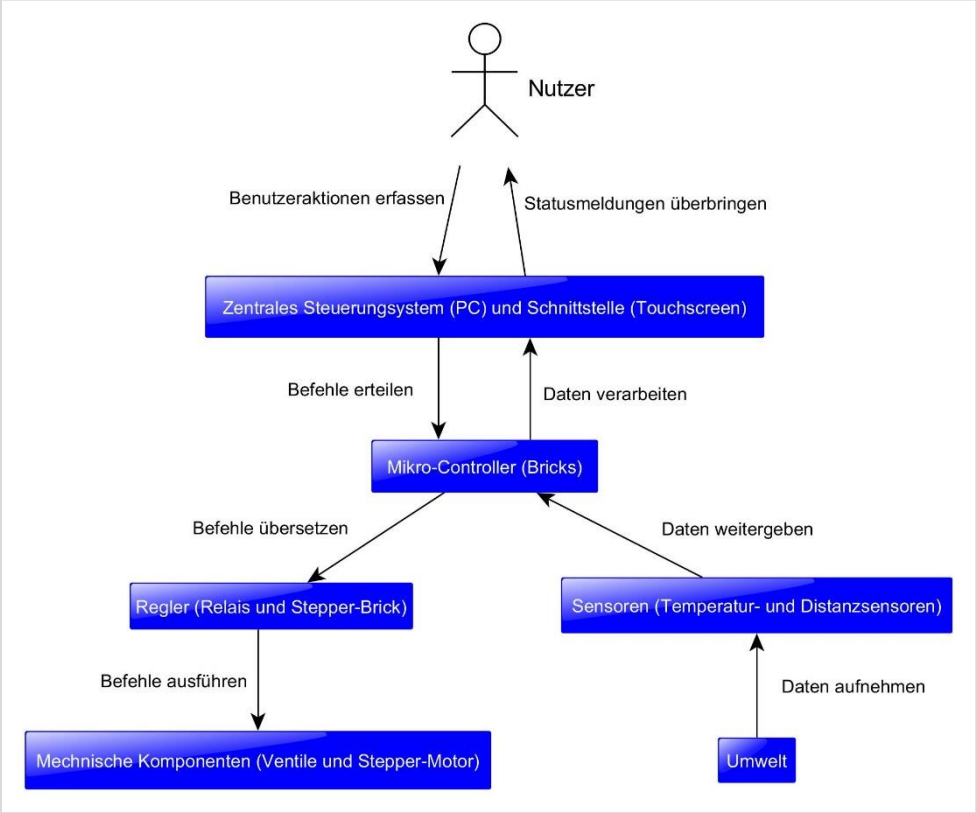


Diagramm 2: Zusammenarbeit der Hardware

4.8. Name des Automaten

Um den Roboter als professionelles Produkt zu präsentieren und dem Nutzer erste Informationen vermitteln zu können, sollte der Getränkeautomaten einen Produktnamen tragen. Der Name sollte direkt auf die Funktion und die verwendete Technik des Automaten hinweisen. Ich habe das Projekt deshalb „*pySpenser*“ genannt. Der Name setzt sich aus der verwendeten Programmiersprache „*Python*“ und dem englischen Wort „*dispenser*“ zusammen, zu Deutsch Ausgeber oder Spender. Darauf aufbauend habe ich ein Logo gestaltet, welches noch deutlicher auf die verwendete Programmiersprache „*Python*“ hinweist.

Bild 16: Logo des Getränkeautomaten

4.9. UML – Diagramme

Die hier folgenden UML-Diagramme sollen die geplante Software so klar wie möglich darstellen. Um die Diagramme verständlicher zu machen, werden einige anhand eines Textes kurz erklärt. Es empfiehlt sich, die Erklärungen zu UML im theoretischen Teil der Arbeit genau zu studieren, bevor man sich diesen teils komplizierten Diagrammen zuwendet.

4.9.1. Use – Case – Diagramme

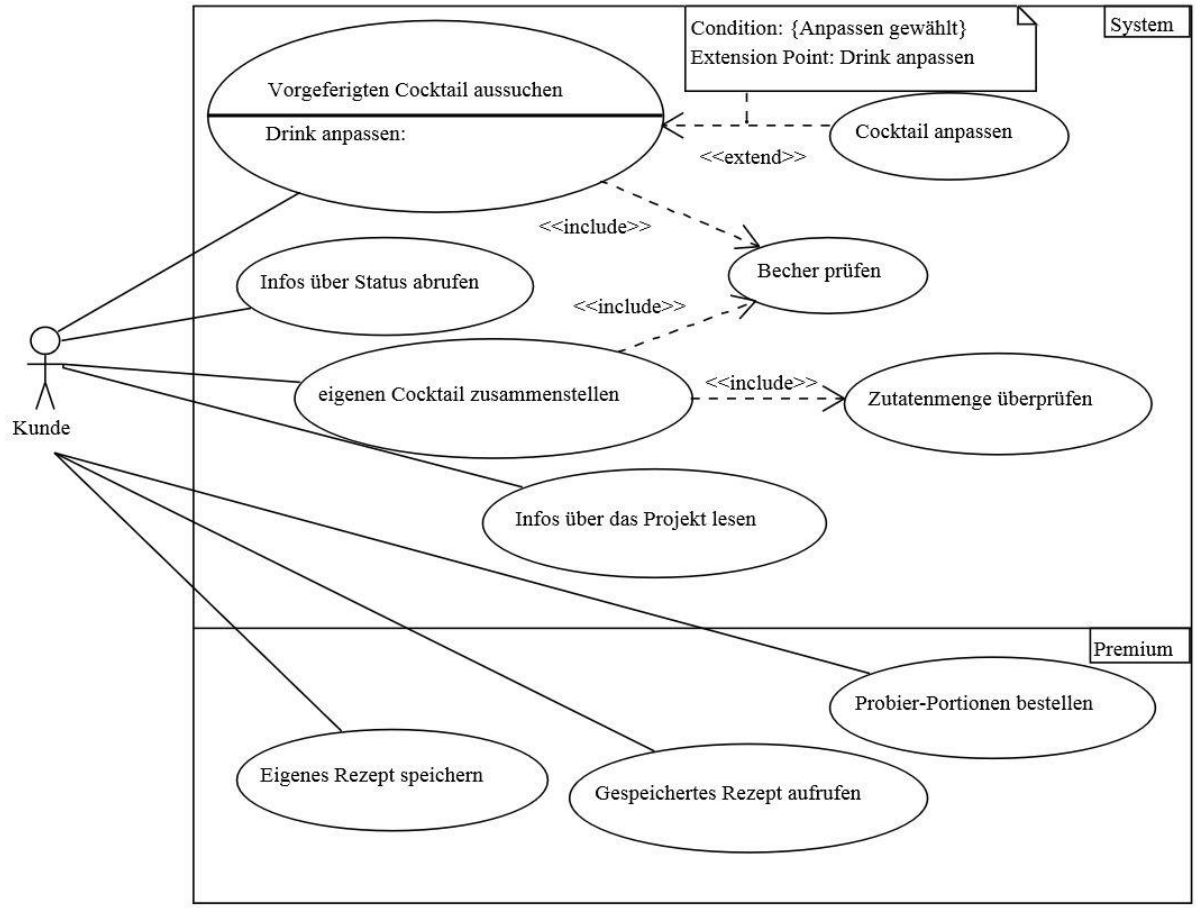


Diagramm 3: Use-Case-Diagramm des Kunden

Dieses Diagramm zeigt, welche Use-Cases der Kunde ausführen kann. Die extend-Assoziation zwischen dem Use-Case „Vorgefertigten Cocktail aussuchen“ und „Cocktail anpassen“ bedeutet, dass wenn der Kunde sich für ein Anpassen des Drinks entscheidet, der Use-Case „Cocktail anpassen“ ausgeführt wird. Die include-Assoziation zwischen dem Use-Case „eigenen Cocktail zusammenstellen“ und „Zutatenmenge überprüfen“ gibt an, dass bei jedem selbsterstellten Drink die Zutatenmenge überprüft wird, um zu verhindern, dass der Becher überläuft. Um sicherzustellen, dass das Getränk aufgefangen wird, prüft die Software bei jeder Bestellung, ob sich unter dem Auslauf ein Becher befindet.

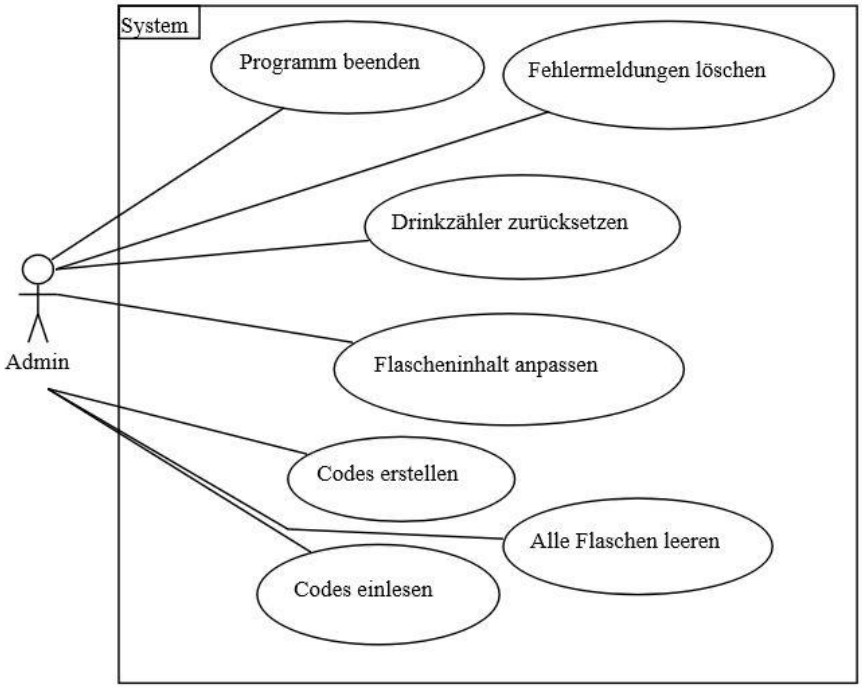


Diagramm 4: Use-Case-Diagramm des Admins

Dieses Diagramm beschreibt die möglichen Use-Cases des Administrators des Automaten. Das Diagramm sollte ohne weitere Beschreibungen klar sein.

4.9.2. Aktivitätendiagramm

Dieses Diagramm ist wegen seiner Grösse in zwei Teile getrennt.

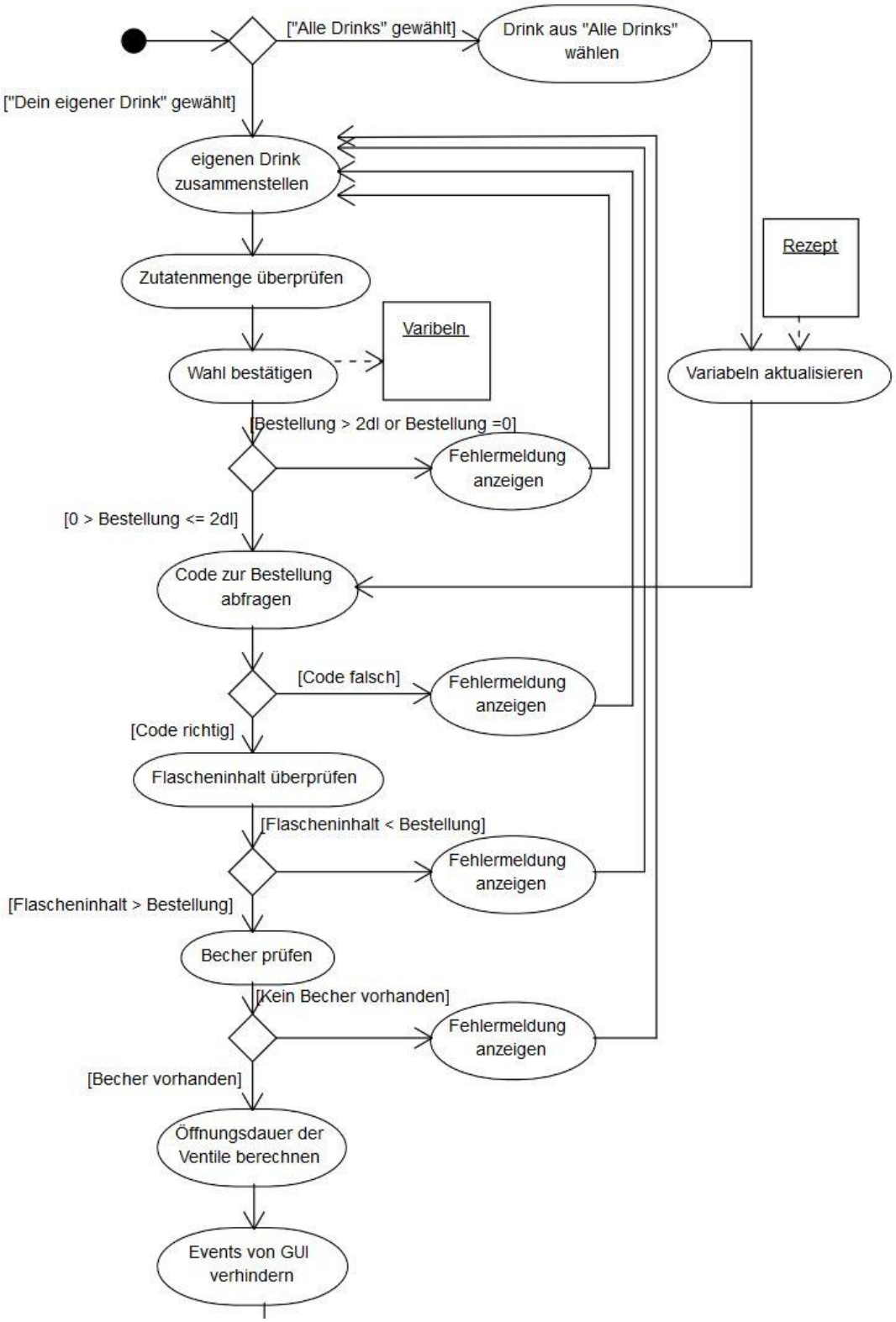


Diagramm 5: Aktivitätendiagramm der Bestellung (oberer Teil)

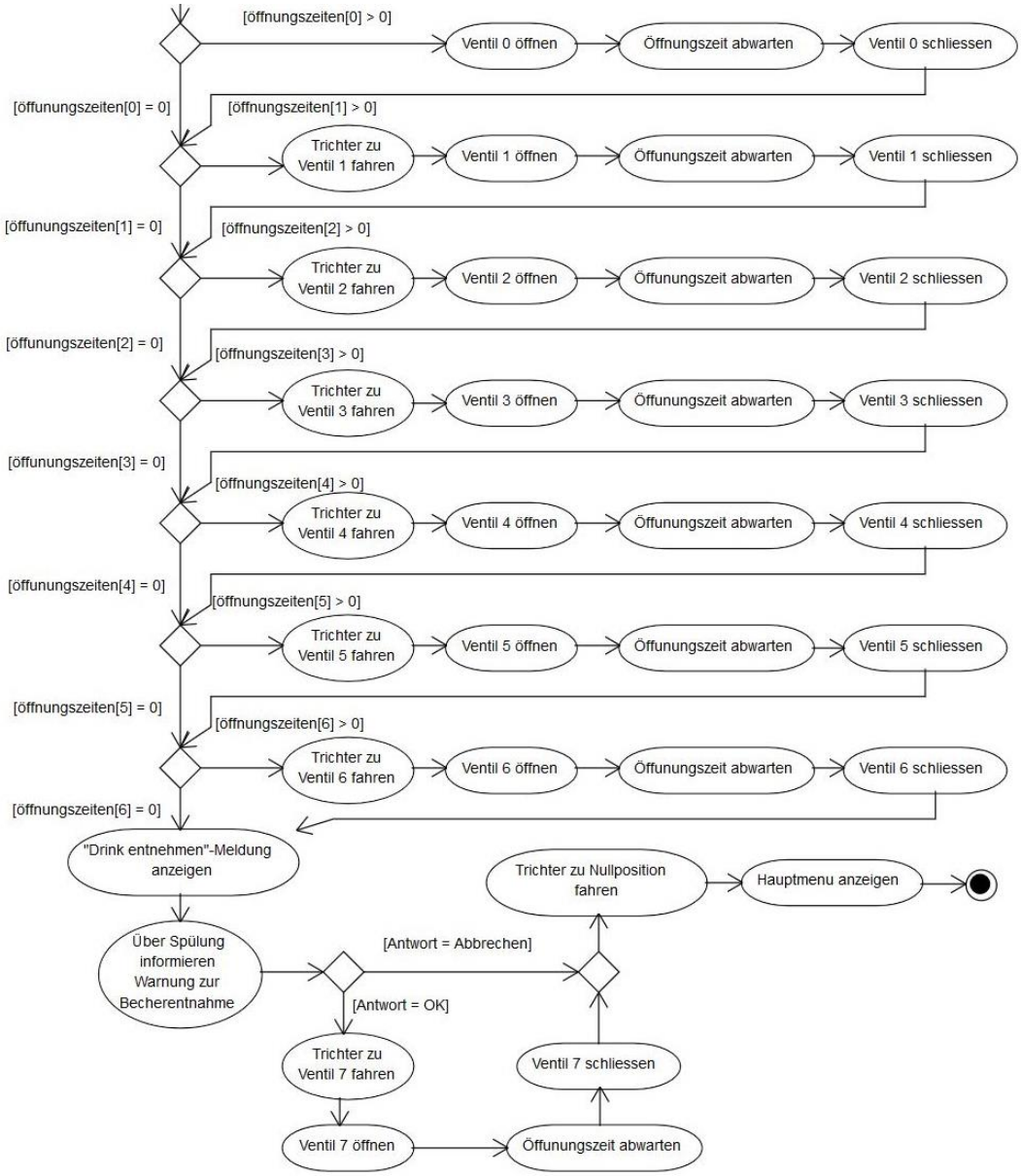


Diagramm 6: Aktivitätendiagramm der Bestellung (unterer Teil)

Dieses Diagramm beschreibt, was im Programm während einer Bestellung abläuft. Im oberen Teil des Diagramms wird hauptsächlich der Prozess der Bestellung dargestellt. Um zu gewährleisten, dass der Kunde eine gültige Bestellung aufgibt, verfügt dieser Bereich über viele if-Konstrukte. Diese geben bei falscher Eingabe, fehlenden Zutaten oder nicht vorhandenem Becher eine Fehlermeldung aus.

Der untere Teil ist stark repetitiv, da hier hauptsächlich die Dosierung dargestellt wird. Man erkennt ein klares Grundmuster, auf welches später noch eingegangen wird. Zusätzlich wird immer überprüft, ob ein Ventil überhaupt angefahren werden muss. Zum Schluss wird der Kunde gefragt, ob er den Automaten spülen möchte. Dies ist nur nötig, wenn der nächste Drink ein anderer ist als der gerade zubereitete.

4.9.3. Klassendiagramm

Dieses Diagramm ist wegen der Grösse ebenfalls in zwei Teile getrennt und zusätzlich quer abgebildet.

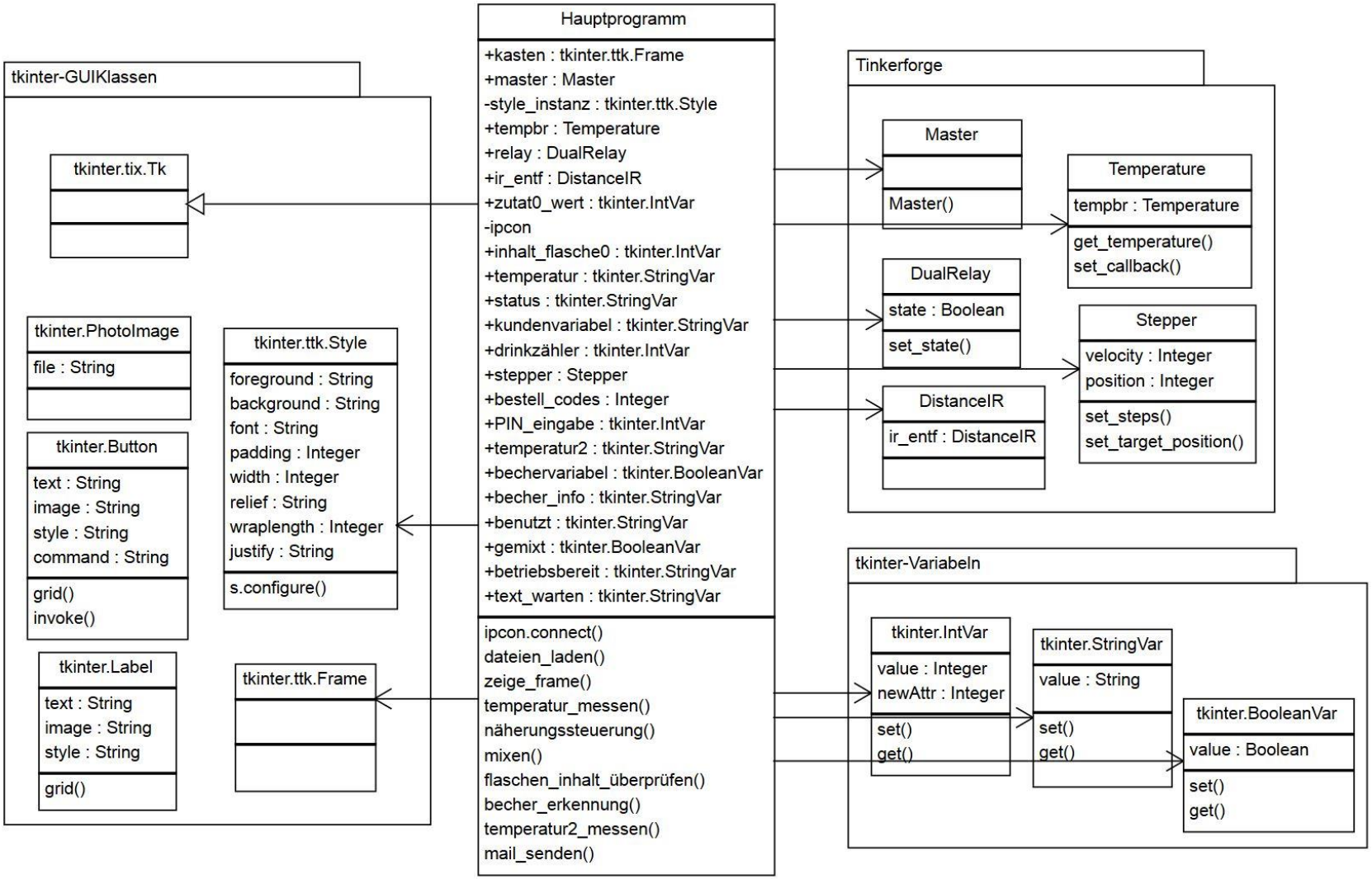


Diagramm 7: Klassendiagramm (oberer Teil)

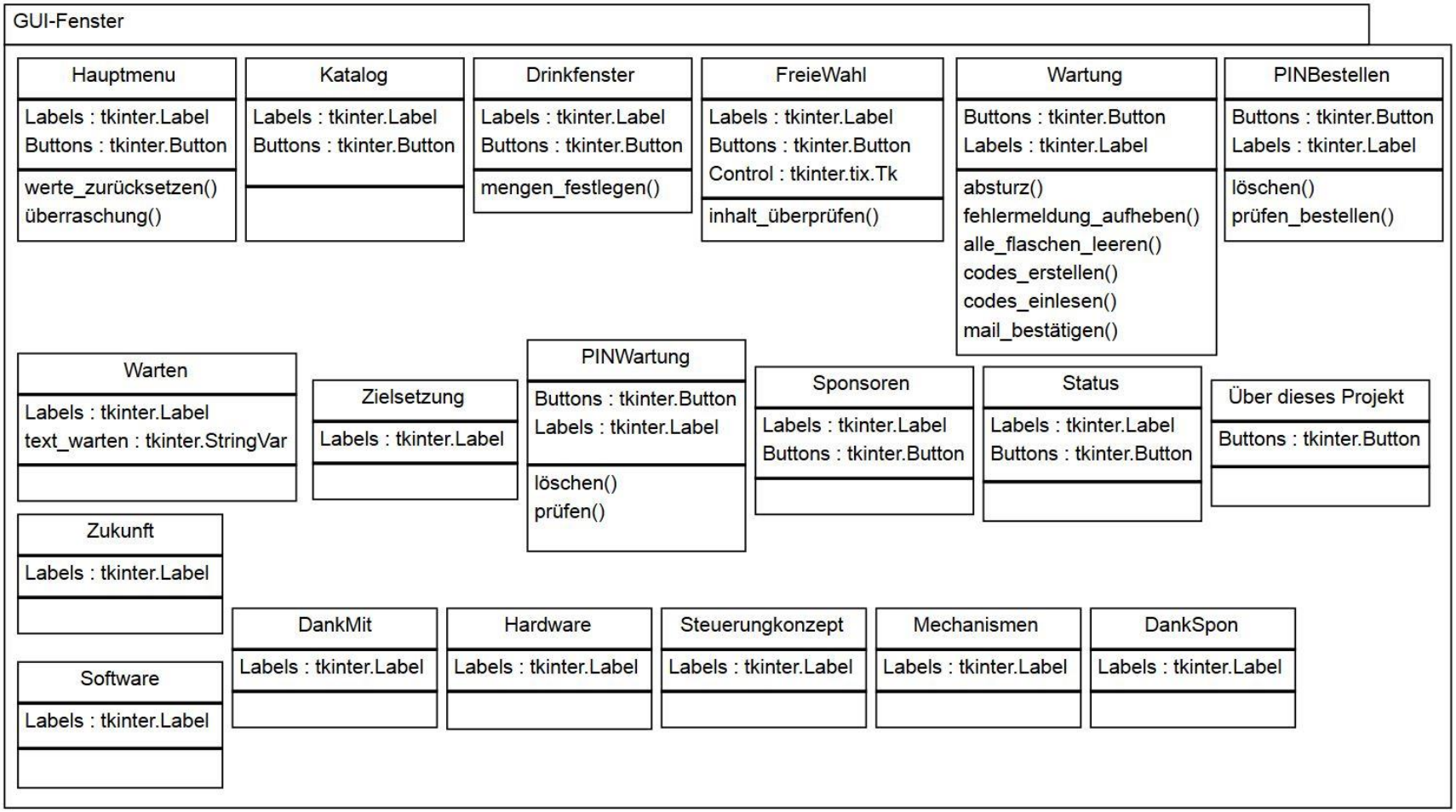


Diagramm 8: Klassendiagramm (unterer Teil)

Dieses Diagramm ist kompliziert und schwer zu verstehen. Es ist nur für Leser mit Programmierkenntnissen interessant. Ergänzend muss noch gesagt werden, dass alle Klassen des Pakets „GUI-Fenster“ von der Klasse „tkinter.ttk.Frame“ erben und in allen oben stehenden Klassen navigieren können. Diese Details wurden zugunsten der Übersichtlichkeit nicht im Diagramm dargestellt.

Das Diagramm zeigt ganz klar die Grundstruktur der Software: Eine Klasse, die sogenannte „Hauptprogramm“-Klasse, übernimmt wie es der Name erahnen lässt, die Koordination des Programms. Die wichtigsten Funktionen wie die Berechnung der Öffnungszeiten, die Steuerung der Dosiermechanik und die Messungen der Sensoren werden alle in dieser Klasse definiert und aufgerufen. Ausserdem werden beinahe alle Variablen in dieser Klasse definiert, was die vielen Attribute der Klasse erklärt. Viele dieser Variablen sind global, das heisst man kann von jedem Ort im Programm auf sie zugreifen. Das macht das Anzeigen von Variablen in GUI-Fenstern deutlich leichter, beispielsweise die Temperatur im „Status“-Fenster. Des Weiteren ist es erwähnenswert dass nur die Klasse „Hauptprogramm“ auf die importierten Klassen der tkinter- und Tinkerforge-Bibliothek zugreifen, obwohl die anderen Klassen theoretisch auch darauf Zugriff hätten.

Die anderen erstellten Klassen bestehen hauptsächlich aus GUI-Elementen, sie bestimmen also wie die Fenster aussehen. Einige von ihnen haben auch eigene Operationen, das beste Beispiel ist hier die PIN-Sperre, welche den eingegebenen Code selbst überprüfen und dementsprechend handeln kann.

5. Umsetzung der Software

In diesem Kapitel wird die Realisierung der Software beschrieben. Zur besseren Übersicht wurde die Software dazu in drei Teile getrennt: die graphische Benutzeroberfläche, die Datenverarbeitung und die Dosierung.

5.1. Eckdaten der Software

Ein ansprechendes und einheitliches Design, eine selbsterklärende Benutzung und hohe Dosierungspräzision: das sind die Stärken der Software des Getränkeautomaten.

Die Software ist in der Lage, auf Nutzereingaben und Umwelteinflüsse autonom zu reagieren. Auf über 4000 Zeilen handgeschriebenem Programmcode verteilt, befinden sich 32 Klassen und 113 verschiedene Funktionen, welche dies ermöglichen. Als Programmiersprache wird hauptsächlich Python verwendet, ein kleiner Teil wurde mit Processing realisiert. Durch 18 Zusatzbibliotheken wurde der Funktionsumfang von Python nochmals deutlich erhöht.

5.2. Die graphische Benutzeroberfläche

In diesem Abschnitt wird die Benutzeroberfläche der entwickelten Software erklärt. Das GUI ist der grösste Teil der Software und ist aufgrund des direkten Kontakts mit dem Kunden sehr wichtig. Das GUI muss eine einfache Bedienung ermöglichen und dabei alle nötigen Informationen vom Kunden aufnehmen können. Ausserdem sollte es ansprechend aussehen und dem Kunden die Möglichkeit bieten, sich genauer über den Getränkeautomaten zu informieren und zu verstehen, welche Abläufe nötig sind, um einen Drink vollautomatisch zuzubereiten.

5.2.1. Struktur des GUI

Das verwendete GUI besteht aus einer Vielzahl verschiedener Fenster, zwischen denen der Nutzer hin und her navigieren kann. Wie das Navigieren genau funktioniert, wird im folgenden Kapitel erklärt. Hier zuerst eine Übersicht über alle Fenster des GUIs und deren Hierarchie. Die Pfeile zwischen den Fenstern geben die Navigierbarkeit untereinander an. Ein einseitiger Pfeil bedeutet, dass man nur in Richtung des Pfeils navigieren kann. Bei einem beidseitigen Pfeil ist es möglich, in beide Richtungen zu navigieren.

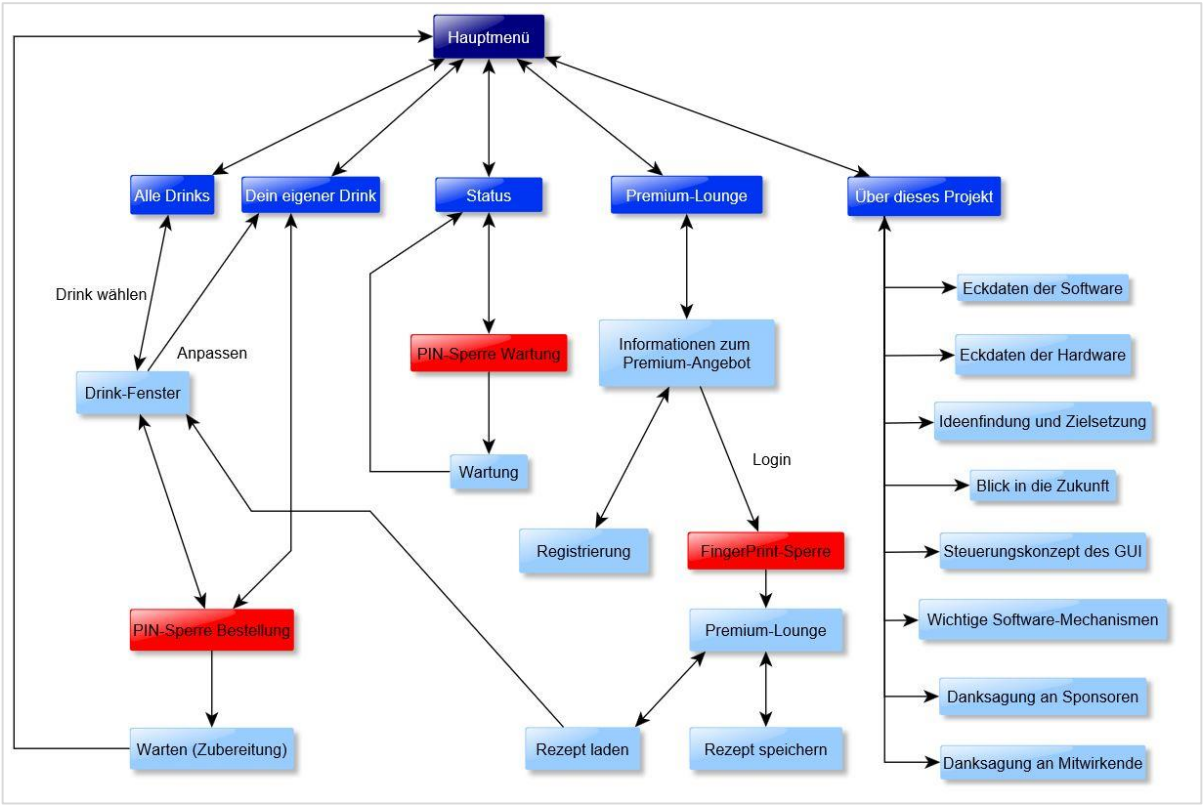


Diagramm 9: Hierarchie der GUI-Fenster

Die verschiedenen Farben der Fenster geben ihre Hierarchie-Ebene an. Je heller der Blauton ist, desto weiter unten befindet sich ein Fenster in der Hierarchie. Die roten Felder sind Sperren, welche nur von autorisierten Nutzern passiert werden können.

5.2.2. Navigation im GUI

Da sich der Nutzer des Automaten durch die verschiedenen Fenster bewegen können muss, bedarf es einiger Navigationstasten. Diese befinden sich immer oben links im GUI und sind in fast jedem Fenster verfügbar. Es gibt einen „Home“-Button und eine „Zurück“-Taste, welche beide als einfach verständliche Symbole dargestellt werden (siehe Bild 15).

Bild 17: "Home"-Button und "Zurück"-Taste

Der „Home“-Button hebt immer das Hauptmenü an die Oberfläche. Die „Zurück“-Taste hebt jeweils den hierarchisch höheren Frame an. Diese Vorgehensweise führt leider zu folgendem Fehler: hat man

beispielsweise vom Hauptmenü direkt eine Schnellwahl getroffen und drückt im „Drink-Fenster“ auf die „Zurück“-Taste, gelangt man nicht ins Hauptmenü sondern in das „Alle Drinks“-Fenster. Dieser Fehler ist aber alles andere als gravierend, weshalb die verwendete Steuerung beibehalten wurde.

Wie der Wechsel zwischen zwei Fenstern genau programmiert wurde kann, soll nun ausführlich erklärt werden. Folgende zwei Grundkonzepte habe ich mir zur Auswahl gestellt:

Ständiges Neuerstellen und Zerstören von Fenstern

Bei diesem Konzept wird je nach Bedarf ein Fenster neu aufgebaut und das alte zerstört. Wenn der Nutzer sich beispielsweise vom Hauptmenü in die Drinkübersicht navigiert, so werden zuerst alle GUI-Elemente (sogenannte *Widgets*) im Hauptmenü zerstört und anschliessend alle Widgets der Drinkübersicht aufgebaut. All das geschieht auf einer Ebene und in einem einzelnen Rahmen, einem sogenannten *Frame*.

Das ständige Auf- und Abbauen bietet viele Aktualisierungsmöglichkeiten der Fenster, da man auf variierende Werte, zum Beispiel die Innentemperatur, beim Aufbauen immer auf den aktuellen Wert zugreifen kann. Das häufige Errichten und Zerstören von Widgets ist aber relativ ineffizient und braucht viel Rechenkapazität und somit einen leistungsfähigen Rechner und viel Strom.

Einmaliges Erstellen von Fenstern und laufendes Aktualisieren

Im Gegensatz zum oben beschriebenen Konzept werden hier beim Starten des Programms auf mehreren Ebenen alle Fenster des GUIs gleichzeitig erstellt und nie zerstört. Man bildet also einen Stapel von Frames, was man als *Stack* bezeichnet. Für den Nutzer ist jeweils nur der oberste Frame sichtbar. Beim Navigieren durch das GUI werden die benötigten Fenster sichtbar, indem sie nach oben gehoben werden.

Das Anheben geschieht mittels einer kleinen, aber sehr wichtigen Funktion, die ich hier ausnahmsweise anhand von Quellcode erklären möchte. Die Funktion lautet folgendermassen:

```
def zeige_frame(self,c):
    frame = self.frames[c]
    frame.tkraise()
```

Bild 18: Quellcode der "zeige_frame"-Funktion

In der ersten Zeile definiert man die Funktion *zeige_frame(self,c)* als das, was nach dem Doppelpunkt folgt. Die Funktion hat die Parameter *„self“* und *„c“*. *„self“* ist dabei ein Standardparameter von Python, der das erste Element vieler Instanzen[53] verkörpert. *„c“* beinhaltet den Namen des Frames, der angehoben werden soll.

[53] Eine Instanz ist ein Objekt, welches während der Laufzeit einer Software durch eine Klasse erschaffen wird.

In der zweiten Zeile legt man dann die lokale Variable *„frame“* als den gewünschten Frame fest, der in *„c“* gespeichert ist, und überprüft gleichzeitig, ob der gewünschte Frame in der Liste aller Frames überhaupt vorkommt.

In der dritten Zeile findet nun das Anheben statt, indem man auf das Objekt *„frame“*, welches der überprüfte, gewünschte Frame ist, die Methode *„tkraise()“* anwendet. Diese Methode gehört zur *tkinter*-Bibliothek und hebt ein GUI-Element ganz nach oben.

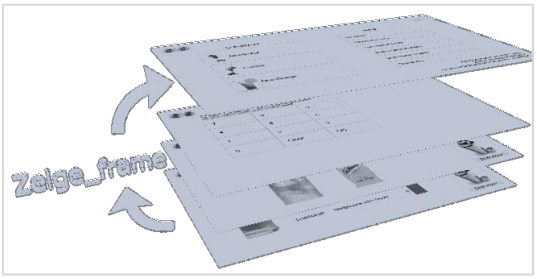


Bild 19: Skizze des Anhebens eines Frames

Der Vorteil des zweiten Navigationskonzepts ist der geringere Rechenaufwand und die somit gesparten Ressourcen. Ein grosser Nachteil ist jedoch das aufwendige Aktualisieren der Fenster. Als Beispiel dient wieder das Anzeigen der Innentemperatur. Diese ändert sich während dem Betrieb des Automaten. Das „Status des Automaten“-Fenster, in welchem die Innentemperatur angegeben wird, wurde aber beim Starten des Programms erstellt. Der angezeigte Wert ist also immer gleich geblieben und nicht dem aktuellen Wert entsprechend angepasst worden. Um dieses Problem zu lösen, verwendet man entweder das sogenannte Event-gesteuerte Abfragen, also einen „Innentemperatur abfragen“-Knopf oder man verwendet spezielle StringVar()-Variablen, mit denen sich automatisch aktualisierende Widgets aufbauen lassen. Aufgrund des geringeren Stromverbrauchs und der begrenzten Rechenleistung des Zentralrechners entschied ich mich für das zweite Konzept.

Anhand dieser kurzen Erklärung des Quellcodes werden einige Dinge sehr deutlich:

1. Es ist oftmals unmöglich, ohne potente High-Level-Methoden wie *„tkraise()“* effizient zu programmieren.
2. Code ist immer komplizierter, als er auf den ersten Blick erscheinen mag.
3. Das verständliche Erklären von Quellcode ist äusserst schwierig und auch für den Leser dieser Arbeit sehr mühsam nachzuvollziehen.

5.2.3. Das Hauptmenü

Dieses Fenster ist der zentrale Punkt im GUI, da von hier alle möglichen Use-Cases beginnen. Das Hauptmenü besteht aus einigen Drink-Verknüpfungen, der sogenannten Schnellwahl, und einem Menü, welches die Navigation zu den restlichen Fenstern ermöglicht. Ausserdem befinden sich im Hauptmenü das Projektlogo und ein Impressum. Um ein einheitliches GUI-Design zu gewährleisten, werden die Navigationstasten ebenfalls angezeigt, obwohl sie im Hauptmenü keine sinnvolle Funktionalität besitzen. Ausserdem wird der Kunde dank einer Anzeige oben rechts über den Status des Automaten informiert.

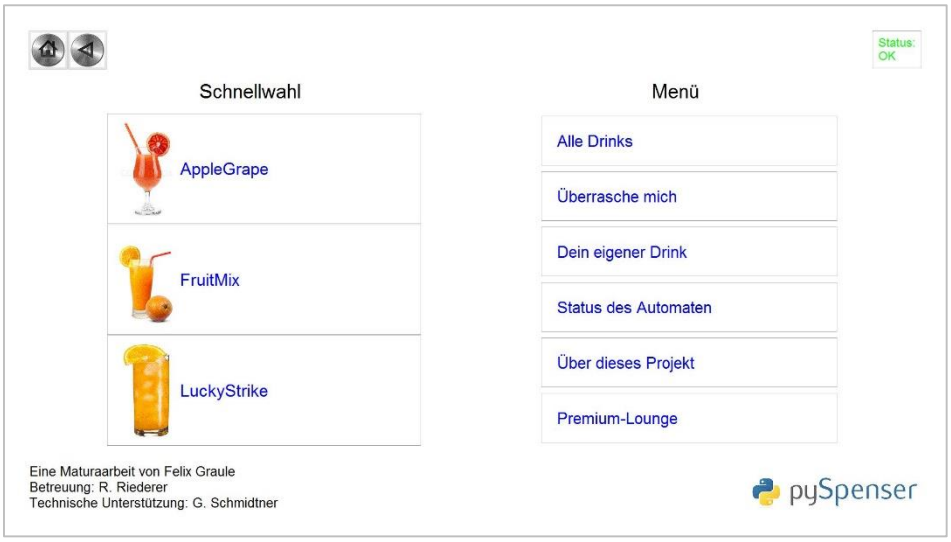


Bild 20: Screenshot des Hauptmenüs

5.2.4. Das „Alle Drinks“-Fenster

Dieses Fenster zeigt eine Übersicht von allen verfügbaren Drinks.

Bild 21: Screenshot des "Alle Drinks"-Fensters

5.2.5. Das „Drink“-Fenster

Von diesem Fenster gibt es insgesamt acht leicht abgeänderte Varianten mit identischer Struktur. Das Fenster besteht aus dem Namen des Drinks, einer kurzen Beschreibung, einem grossen Bild und einer Liste der verwendeten Zutaten. Ausserdem gibt es noch zwei Buttons, um den Drink entweder unverändert zu bestellen oder anzupassen.

Bild 22: Screenshot des "Drink"-Fensters

5.2.6. Das „Dein eigener Drink“-Fenster

In diesem Fenster kann der Kunde sich seinen eigenen Drink zusammenstellen. Das Fenster besteht im Wesentlichen aus einem grossen Bild und sieben ähnlichen Widget-Gruppen. Diese beinhalten jeweils den Namen der Zutat, eine kurze Beschreibung und ein Control-Widget, mit welchem die gewünschte Menge verändert werden kann.

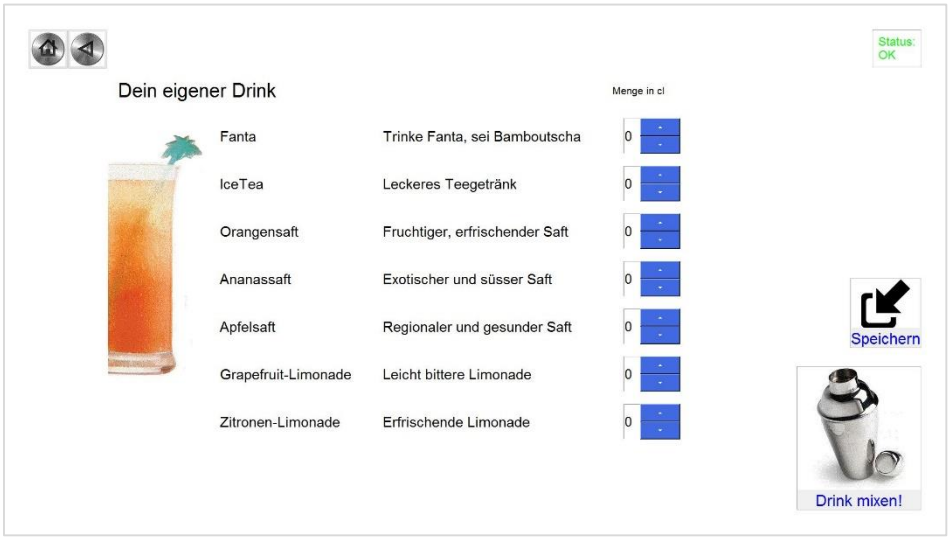


Bild 23: Screenshot des "Dein eigener Drink"-Fensters

5.2.7. Das „Status“-Fenster

In diesem Fenster werden Informationen zum Status des Automaten angezeigt. Ausserdem befindet sich hier der Zugang zum Wartungsbereich der Software.

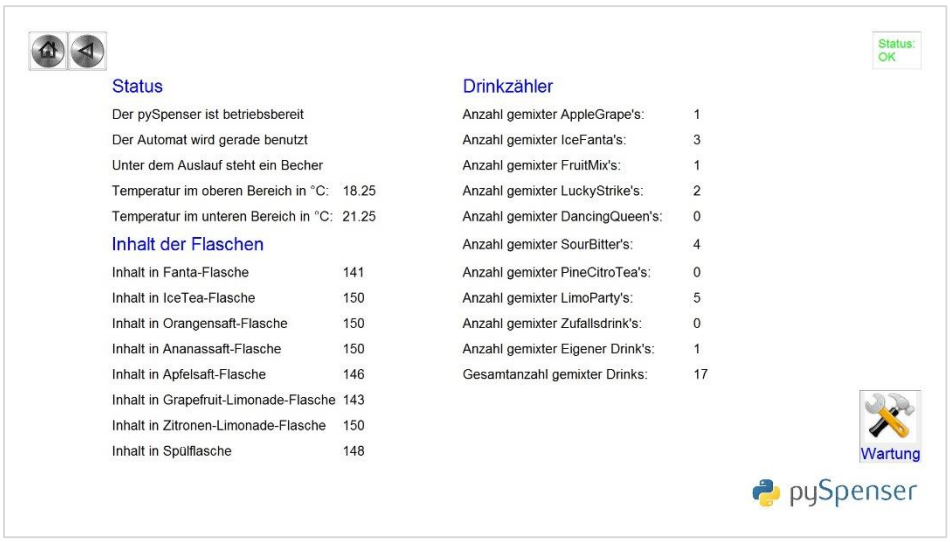


Bild 24: Screenshot des "Status"-Fensters

5.2.8. Das Wartungsfenster

Wenn der Administrator den Automaten warten muss, benutzt er dazu dieses Fenster. Es besteht aus einigen Verknüpfungen zu wichtigen Wartungsfunktionen und der Flaschenverwaltung. Diese wird benötigt, um dem Programm das Auswechseln einer Flasche mitzuteilen.

Bild 25: Screenshot des Wartungsfensters

5.2.9. Das „Über dieses Projekt“-Fenster

In diesem Fenster soll sich der Kunde über das Projekt informieren können. Die eigentlichen Informationen befinden sich wegen fehlender Anzeigefläche jedoch jeweils in einem separaten Fenster.

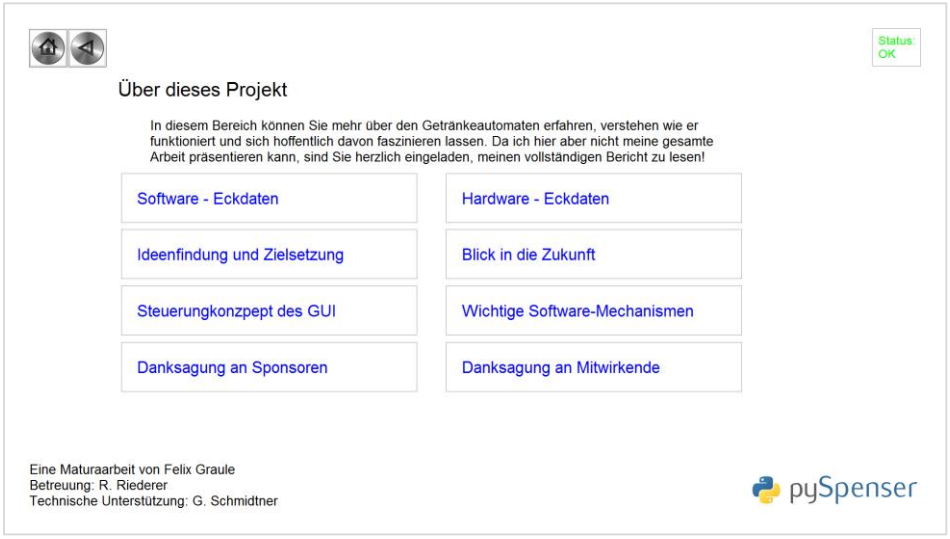


Bild 26: Screenshot des "Über das Projekt"-Fensters

5.2.10. Das „Premium-Login“-Fenster

Dieses Fenster dient den bereits registrierten Premium-Kunden als Eingang zu ihrem eigenen Bereich. Ausserdem werden Kunden, welche noch keinen Premium-Zutritt haben, über das Angebot informiert und ihnen wird die Registrierung ermöglicht. Das Design dieses Fensters ist noch nicht final, wird also noch weiter verfeinert.

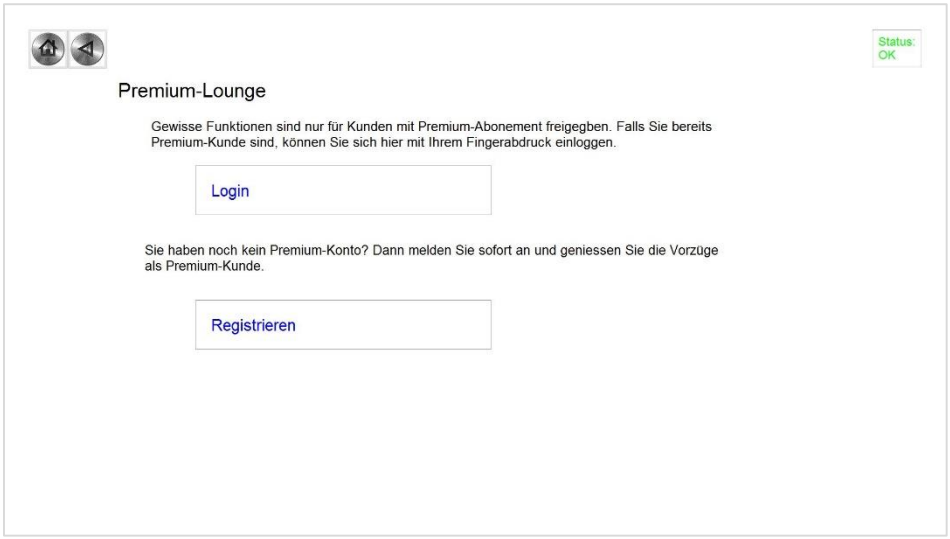


Bild 27: Screenshot des „Premium-Login“-Fensters

5.2.11. Die „Premium-Lounge“

Die Premium-Lounge sieht für jeden Kunden etwas anders aus, sie wird jeweils dem gerade eingeloggten Premium-Kunden entsprechend angepasst. Dabei werden die Fensterüberschrift sowie die tiefergelegenen Fenster abgeändert, insbesondere das Fenster, in dem man eigene Rezepte laden kann. Auch bei diesem Fenster ist das Design noch im Entwicklungsstadium und noch nicht vollständig ausgearbeitet.

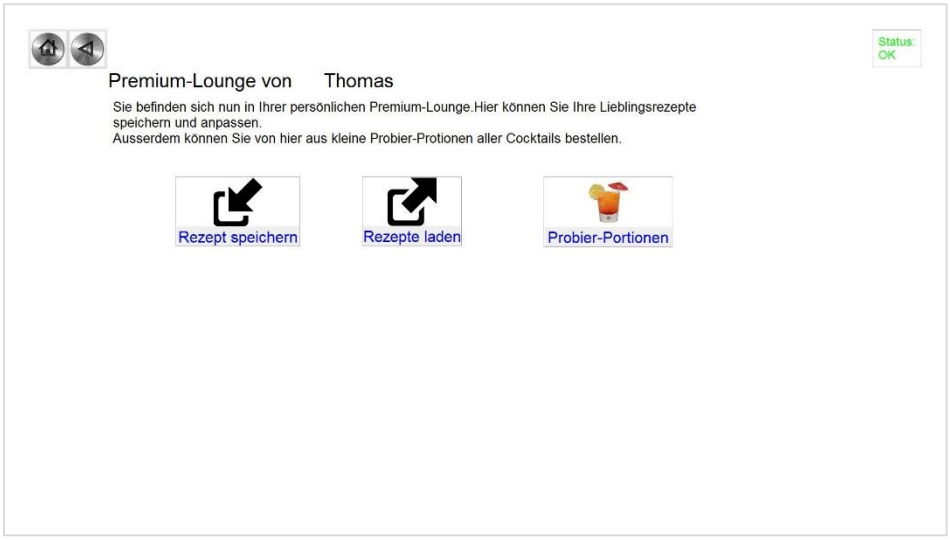


Bild 28: Screenshot der Premium-Lounge

5.2.12. Das „Warten“-Fenster

Dieses Fenster wird während des Mischvorgangs angezeigt und verhindert, dass ein Kunde eine zweite Bestellung aufgibt, bevor die letzte abgeschlossen ist. Das Fenster informiert den Kunden ausserdem über die Sponsoren und die Geschehnisse im Automaten. Ausserdem teilt es dem Kunden mit, wann das bestellte Getränk fertig ist und dass der Automat gespült werden muss.

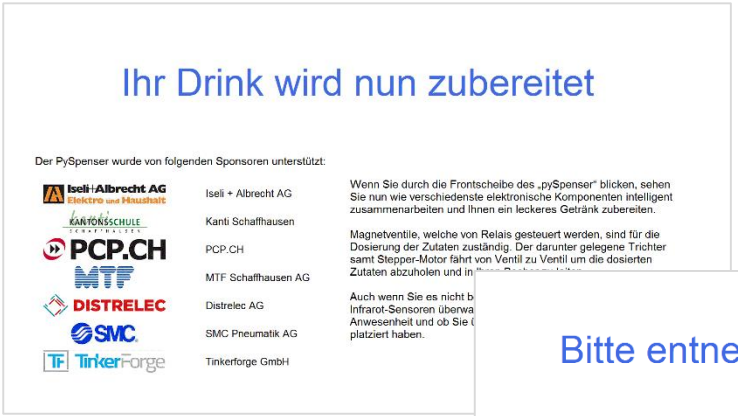


Bild 29: Screenshot des "Warten"-Fensters nach Abschluss des Mischvorgangs

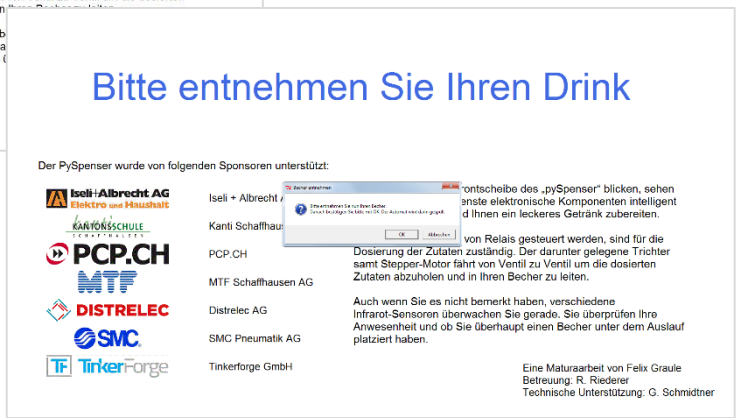


Bild 30: Screenshot des "Warten"-Fensters während dem Mischvorgang

5.3. Dosierung

In diesem Abschnitt werden die Programmfragmente erklärt, welche für die Dosierung der Drinkzutaten zuständig sind.

5.3.1. Steuerung der Dosiermechanik

Diese Funktion steuert die Ventile und die Trichter-Positionierung. Grob vereinfacht führt diese Funktion folgende Schritte aus:

Diagramm 10: Steuerung der Dosiermechanik

Dieser Vorgang muss für jede Getränkeflasche, also insgesamt siebenmal, wiederholt werden. Die einzelnen Schritte sind im Code jeweils nur einige Zeilen lang, die gesamte Funktion ist mit fast 200 Zeilen aber deutlich länger, als man vielleicht annehmen würde. Das liegt vor allem an den Timern, welche dafür sorgen, dass die Schritte zur richtigen Zeit ausgeführt werden. Ausserdem verfügt die Funktion über einige Setup- und Überprüfungsmechanismen. Der ganze Prozess ist im Aktivitätendiagramm in Kapitel 4.7.2. graphisch dargestellt.

5.3.2. Spülen der Dosiermechanik

Beim Zubereiten eines Drinks läuft Flüssigkeit durch Schläuche, wobei es oft zu Rückständen kommt. Damit sich verschiedene Drinks nicht vermischen, kann der Automat den Trichter und den Schlauch zum Ausschank spülen. Der Automat informiert den Kunden nach jedem Mischvorgang, dass der Automaten gespült wird. Der Kunde wird ausserdem darauf hingewiesen, seinen Becher zu entnehmen, weil sonst Spülwasser hineinfliessen würde. Anschliessend wird geprüft, ob der Becher entfernt wurde. Wenn nicht, gibt der Automat eine Warnung aus. Nun beginnt der Spülvorgang. Der Trichter fährt zum Spülventil und es wird Wasser ausgelassen. Danach fährt der Trichter wieder in die Startposition und das GUI wechselt zum Hauptmenü.

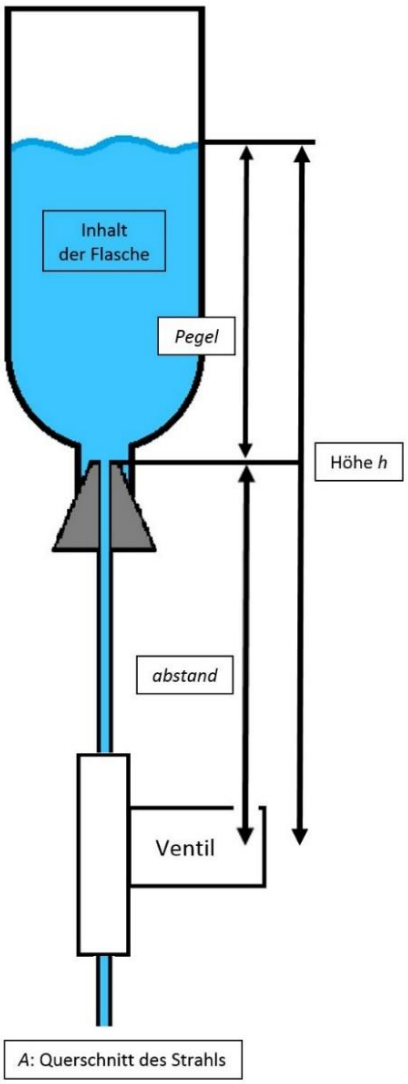
5.3.3. Algorithmus zur Dosierung

In diesem Abschnitt wird der Algorithmus erklärt, welcher die Öffnungszeiten der Ventile berechnet. Der Algorithmus verwendet einige Formeln, die im Theorieteil der Arbeit erklärt werden, es ist darum empfehlenswert, diesen Abschnitt bereits gelesen zu haben.

Die Grundidee der verwendeten Dosiermethode besteht darin, durch zeitlich abgestimmtes Öffnen und Schliessen eines Ventils Flüssigkeit abzumessen. Beim Ausfliessen ändert sich der Flüssigkeitspegel in der Flasche. Dadurch sinkt der Druck auf die Flüssigkeit beim Ausflusspunkt und damit auch die Ausflussgeschwindigkeit. Je nach Füllstand muss das Ventil also unterschiedlich lang geöffnet bleiben, um das gleiche Volumen zu dosieren. Die Öffnungszeiten müssen deshalb vor jedem Dosieren neu berechnet werden.

Um die Öffnungszeiten möglichst genau berechnen zu können, habe ich einen Algorithmus entwickelt. Der Algorithmus arbeitet mit folgenden *Eingangswerten*:

zutat_wert:	Volumen, das dosiert werden soll in $\boldsymbol{cl}$
inhalt_flasche:	Vorhandene Flüssigkeit in der Flasche in cl
abstand:	Distanz zwischen der Ventilöffnung und der Flaschenöffnung in m
g:	Erdbeschleunigung in $\frac{m}{s^2}$
A:	Strahlquerschnitt in m^2



Skizze 15: Situation

Zu jeder im Kühlschrank hängenden Flasche gibt es eine Liste, welche den Pegel in Abhängigkeit zum Inhalt der Flasche ausdrückt. Die Liste besteht bei einer 100 cl Flasche demnach aus 100 Werten, welche den Pegel zu jedem Flascheninhalt angibt, mit deren Hilfe der Algorithmus aus dem Eingangswert *inhalt_flasche* den Pegel in der Flasche bestimmt. Zu diesem Wert wird der *abstand* addiert, was dann die Gesamthöhe *h* zwischen Ventil und Wasseroberfläche ergibt. Nun wird mit der Formel $v = \sqrt{2gh}$ [54] die erwartete Austrittsgeschwindigkeit berechnet, woraus mit dem

[54] Die erwähnten Formeln werden im Kapitel *3.7 Dosieren von Flüssigkeiten* genauer erklärt.

Zusammenhang $t = \frac{V}{Av}$ die benötigte Öffnungszeit berechnet wird. Danach wird die Variable *inhalt_flasche* aktualisiert:

inhalt_flasche = inhalt_flasche – zutat_wert

Die *Ausgangswerte* des Algorithmus sind also die Öffnungszeit *t* und die aktualisierte Variable *inhalt_flasche.* Der Pseudocode für die Berechnung der Öffnungszeit für eine Dosierung von 4 cl lautet:

zutat_wert = 4
inhalt_flasche = 100
abstand = 0.113
g = 9.81
A = 0.0000042

pegel = pegel_liste[100 - inhalt_flasche]
h = abstand + pegel
$v = \sqrt{2gh}$
$t = \frac{zutat\_wert}{Av}$
öffnungszeit = t
inhalt_flasche = inhalt_flasche – zutat_wert

5.3.4. Bestimmen der Füllstände

Die Werte der Listen sind experimentell bestimmt worden, wozu der nebenstehende Versuchsaufbau verwendet wurde. Eine gefüllte Flasche ist in der Flaschenhalterung eingespannt, an ihr ist ein Lineal zur Höhenmessung befestigt. Mit einem drehbaren Hahn kann präzise Flüssigkeit aus der Flasche dosiert werden. Unter dem Hahn steht eine Waage, um das Gewicht der ausgeflossenen Flüssigkeit zu messen. Das gemessene Gewicht entspricht in sehr guter Näherung dem Volumen. Dies folgt aus der Tatsache, dass die Dichten der Getränke sehr nahe bei $\rho = \frac{1g}{cm^3}$ liegen.

Bild 31: Foto des Versuchaufbaus

Zur Messung der Listenwerte wird nun stückweise Flüssigkeit dosiert und der entsprechende Füllstand notiert. Da sich einige Flaschen in ihrer Form sehr ähnlich sind, lassen sich die Listen teilweise mehrfach nutzen. Dadurch verringert sich der Aufwand des experimentellen Bestimmens der Pegelwerte.

5.3.5. Implementierung des Algorithmus

An dieser Stelle soll nun der Algorithmus zur Dosierung anhand von Quellcode besprochen werden.

Der für die Dosierung zuständige Code ist in der Software des Getränkeautomaten sehr verschachtelt eingebaut und daher nur schwer fassbar. Beim Experiment zur Dosiergenauigkeit (siehe *Kapitel 5.3.6.*) kommt der Code in nahezu unveränderter Form vor, ist aber nicht verschachtelt und darum gut verständlich. Aus diesem Grund wird nachfolgend der Code des Experiments erläutert.

```
1  g= 9.81
2  A= 0.0000042
3  höhe_flasche_ventil = 0.133
4
5  def dosieren():
6      inhalt_flasche0 = 150
7      for i in range(30):
8          print(inhalt_flasche0)
9          ln0 = 150 - inhalt_flasche0
10         p0 = füllstand_typ1[ln0]/1000
11         h0 = p0 + höhe_flasche_ventil
12         v0 = math.sqrt(2*g*h0)
13         vol0 = 5/100000
14         t0 = vol0/(v0*A)
15
16         relay1.set_state(None, True)
17         print("Ventil geöffnet")
18         print(t0)
19         time.sleep(t0)
20         relay1.set_state(None, None)
21         print("Ventil geschlossen")
22
23         time.sleep(15)
24         inhalt_flasche0 = inhalt_flasche0 - 5
25
26 if __name__ == '__main__':
27     dosieren()
```

Bild 32: Programm-Code zur Messung der Dosiergenauigkeit

Zeile 1-3: In den ersten drei Zeilen werden die Erdbeschleunigung *g*, die Querschnittsfläche des Schlauches *A* und der Abstand zwischen Ventil und Flasche *höhe_ventil_flasche* als globale Variablen definiert, das heisst sie sind auch innerhalb von Funktionen aufrufbar.

Zeile 5-6: Danach wird eine Funktion *dosieren()* definiert. In dieser Funktion wird der Inhalt zu Beginn des Experiments als 150 cl festgelegt. Dieser Wert ist ein Anfangswert der

folgenden Schleife und muss darum innerhalb der Funktion *dosieren()* definiert werden.

Zeile 7: Die nachfolgende Zeile erstellt einen sogenannten *for-Loop*, also eine Schleife, bei welcher die Anzahl an Iterationen oder Wiederholungen im Voraus bekannt ist. Diese Schleife soll insgesamt 30 Mal wiederholt werden [*in range(30)*].

Zeile 8-9: Die Schleife beginnt damit, dass das Skript mit dem *print()*-Befehl den aktuellen Füllstand ausgibt. Danach wird der aktuelle Inhalt *inhalt_flasche0* mathematisch umgeformt, der dabei entstehende Wert *ln0* beschreibt das folgende: die Liste mit den experimentell ermittelten Füllhöhen besteht aus 150 Werten. Die Zahl *ln0* beinhaltet nun die Position in dieser Liste, an welcher sich die aktuelle Füllhöhe befindet.

Zeile 10-12: Diese Höhe wird in Zeile 10 als *p0* definiert. Zu diesem Wert muss noch der Abstand zwischen Ventil und Flasche addiert werden, dies ergibt die absolute Höhe *h0*. Anhand dieser Höhe wird nun die erwartete Austrittsgeschwindigkeit als *v0* berechnet, wobei die Formel von Torricelli verwendet wird (Zeile 12).

Zeile 13-14: Als *vol0* ist das zu dosierende Volumen in SI-Einheit definiert. Die benötigte Öffnungsdauer ist folglich als $t0 = \frac{vol0}{v0*A}$ definiert.

Zeile 16-18: In dieser Zeile wird das Relais geschalten und damit das Ventil geöffnet. Dies teilt das Skript mit einem *print()*-Befehl mit, auch die errechnete Öffnungszeit wird ausgegeben.

Zeile 19-21: Der Befehl *time.sleep(t0)* friert das Skript für die berechnete Öffnungszeit *t0* ein, wobei das Ventil während dieser Zeit geöffnet bleibt und Flüssigkeit dosiert wird. Nach Ablaufen der Zeit folgt wird das Ventil wieder geschlossen, auch das wird als Nachricht ausgegeben.

Zeile 23-24: Das Skript wird nun für 15 Sekunden eingefroren [*time.sleep(15)*], um das Ablesen und Eintragen des Gewichtes auf der Waage zu ermöglichen. Zum Schluss der Schleife wird der Flascheninhalt aktualisiert. Anschliessend beginnt der Loop von neuem.

Zeile 26-27: Die letzten zwei Zeilen starten die Funktion *dosieren()* sobald das Skript ausgeführt wird.

5.3.6. Experiment zur Dosiergenauigkeit

Um die zubereiteten Getränke auf ihre Qualität untersuchen zu können, habe ich ein Experiment zur Dosiergenauigkeit der Zutaten durchgeführt. Dieser Versuche sowie dessen Resultate sind nachfolgend erklärt.

Das Experiment zur Dosiergenauigkeit ist folgendermassen aufgebaut: ein zuvor gewähltes Volumen $V_{gewählt}$ wird dosiert und das Gewicht der austretenden Flüssigkeit $m_{gemessen}$ mit einer Waage bestimmt. Aus dem Gewicht lässt sich in guter Näherung das tatsächlich dosierte Volumen $V_{gemessen}$ bestimmen, wobei nach Division durch 10 die Einheit Zentiliter (cl) resultiert. Die tatsächlich dosierte Flüssigkeit pro Dosierung wird dann als Differenz zwischen dem aktuellen Wert auf der Waagenanzeige und dem vorhergehende Wert berechnet:

$$\Delta V = V2_{gemessen} - V1_{gemessen}$$

Das Experiment wird drei Mal wiederholt, um Fehler durch Ausreisser zu minimieren. Insgesamt werden also drei Mal 150 cl dosiert und kontrolliert, da eine volle Flasche 150 cl fasst. Als zu dosierendes Volumen wurden 5 cl gewählt: $V_{gewählt} = 5\ cl$. Pro Flasche werden $\frac{150}{5} = 30$ Messungen vorgenommen. Somit erhält man $3\ * 30 = 90$ Messpunkte.

Die drei Messreihen wurden anschliessend zu einer gemittelten Durchschnittsreihe zusammengefasst. Die Messreihen wurden mit Werkzeugen aus der Statistik untersucht, so etwa der Standartabweichung und dem Mittelwert. Nachfolgend werden die Ergebnisse graphisch dargestellt und diskutiert.

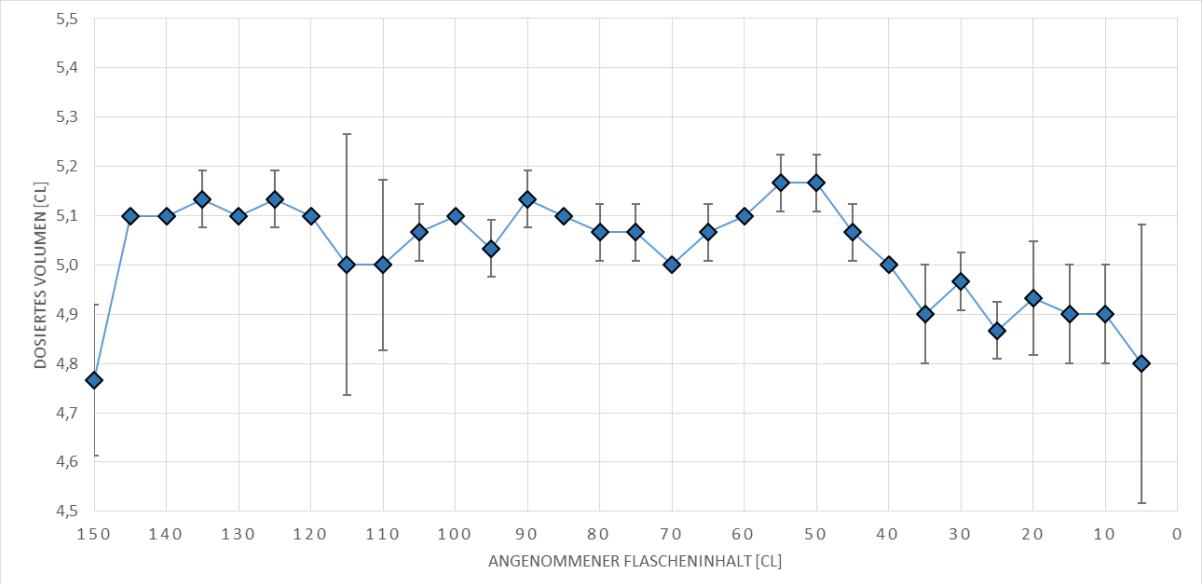


Diagramm 11: Experiment zur Dosiergenauigkeit

Auf der x-Achse des Diagramms befindet sich der Flascheninhalt, welcher vom Automaten vor einem Dosiervorgang angenommen wird. Die y-Achse des Diagramms zeigt das jeweils dosierte Volumen pro Dosiervorgang. Die Standardabweichung je Messpunkt wird als Fehlerindikator angezeigt. Alle Daten sind einheitlich in Zentiliter angegeben.

Der Mittelwert aller Dosiervorgänge liegt bei 5.03 cl, die Standardabweichung aller Messdaten beträgt 0.1273 cl. Unter Annahme der Standardnormalverteilung ergibt sich, dass 95% aller Werte zwischen 4.781 cl und 5.280 cl liegen sollten. Der zu erwartende Fehler bei einer Dosierung ist somit kleiner als 3 ml. Prozentual ausgedrückt beträgt der Fehler also weniger als 6% des zu dosierenden Volumens.

Die Standardabweichung ist beim ersten und letzten Messpunkt auffällig hoch. Dies lässt sich folgendermassen erklären: bei der ersten Messung ist das Schlauchstück, welches Flasche und Ventil verbindet, noch leer. Daher wird im ersten Dosiervorgang weniger Flüssigkeit dosiert. Bei der letzten Dosierung war hingegen die Flasche bei einer Messreihe bereits fast leer, weshalb nicht während der gesamten Öffnungszeit des Ventils Flüssigkeit ausfliessen konnte.

Die Dosiermethode ist äusserst effizient und die Dosiergenauigkeit erfüllt die Kriterien des vorgesehen Einsatzes bezüglich Genauigkeit bei hohem Demonstrationswert.

5.4. Datenverarbeitung

Im folgenden Abschnitt möchte ich die wichtigsten Bestandteile der Datenverarbeitung beschreiben. Die Datenverarbeitung besteht aus vielen verschiedenen Funktionen, welche gesammelte Daten des Kunden verarbeiten und Befehle erteilen. Sie kümmern sich um die Verarbeitung der Nutzereingaben und Umwelteinflüssen.

5.4.1. Speichern der Zutatenwahl

Ein Kunde möchte sich früher oder später für einen Cocktail entscheiden und diesen zubereiten lassen. Er kann dazu ins „Alle Drinks“-Fenster gehen und sich dort einen vorgefertigten Drink bestellen. Es ist aber auch möglich, dass der Kunde mit dem Angebot nicht zufrieden ist und seinen Drink anpassen möchte. Dies kann er im Fenster „Drink anpassen“ bewerkstelligen.

Dieses Fenster basiert auf dem Fenster „Dein eigener Drink“. Da sich der Kunde aber bereits für ein Getränk entschieden hat, müssen die anzuzeigenden Zutatenwerte angepasst werden. Deshalb muss jede durch den bereits gewählten Drink festgelegte Zutatenmenge als Zahl in einer Variablen gespeichert werden. Das Fenster aktualisiert sich dementsprechend selbst und zeigt darum stets die aktuellen Werte.

In der verwendeten GUI-Bibliothek *„tkinter“* heissen die benötigten Variablen *IntegerVariable* (kurz *IntVar*) und können mit dem Befehl *name.set(Wert)* festgelegt werden. Wählt der Nutzer einen Cocktail, werden diese IntVar-Variablen dem Drinkrezept entsprechend angepasst und gespeichert, damit das Programm beim eigentlichen Mischvorgang wieder Zugriff darauf hat. Das Speichern wird ausgelöst, sobald der Nutzer den „Drink-Mixen“-Knopf betätigt (siehe Bild 28 unten rechts). Beim Anpassen im „Dein eigener Drink“-Fenster werden die Werte bei jeder Veränderung automatisch aktualisiert, ein separates Speichern ist hier nicht notwendig.

Bild 33: Screenshot des "Drink"-Fensters

Bild 34: Screenshot des "Dein eigener Drink"-Fensters

5.4.2. Zufällige Auswahlen

Der User kann die Wahl seines Getränks auch dem Zufall überlassen. Es gibt einen „Überrasche mich"-Knopf im Hauptmenü und einen Zufallsdrink.

Wird der „Überrasche mich"-Knopf gedrückt, erzeugt das Programm eine Zufallszahl von null bis acht und navigiert dann der Zahl entsprechend zu einem der neun Drink-Fenster. Hier wird ein zufälliger Drink aus der Drinkrezept-Bibliothek gewählt, dessen Zusammensetzung vor der definitiven Bestellung klar ist.

Beim Zufallsdrink wird bei jeder Bestellung ein zufälliges Rezept erzeugt, indem für jede der sieben möglichen Zutaten eine zufällige Menge gewählt wird. Der Drink muss ebenfalls 20 cl Inhalt haben. Dazu werden sechs Zufallszahlen zwischen 0 und 99 erzeugt und addiert. Danach wird ein Quotient gebildet:

$$Quotient = \frac{Drinkvolumen}{Zutatensumme} = \frac{20}{\sum aller\ Zufallszahlen}$$

Anschliessend werden alle Zufallszahlen mit diesem Quotienten multipliziert, wodurch die Summe aller Zutaten 20 cl entspricht. Zur Sicherheit wird die Gesamtmenge an Zutaten nochmals überprüft, bevor die Bestellung weitergegeben wird. Da der gemischte Drink nicht mit Sicherheit geniessbar ist, wird der Kunde vor der Bestellung ausdrücklich gefragt, ob er dieses Risiko eingehen will.

5.4.3. Kunden- und Bechererkennung

Der Automat ist über Infrarotsensoren in der Lage, sein Umfeld zu erfassen. Es gibt zwei dieser Sensoren, wobei der eine der Kundenerkennung und der andere der Becherüberwachung dient.

Die Kundenerkennung ermöglicht dem Automaten ein energiesparendes Verhalten. Ausserdem soll der Kunde auch bemerken, dass er erkannt wurde.

Zum Messen der Distanz befindet sich an der Front des Automaten ein Infrarotsensor mit einer maximalen Reichweite von 1.5 Metern. Zum Abfragen der Distanz nutzt das Programm sogenannte *Callbacks.* Diese lösen bei Erfüllen einer bestimmten Bedingung eine vorgegebene Funktion aus. In diesem Fall ist die Bedingung, dass die vom Sensor gemessene Distanz unter einem bestimmten Grenzwert liegt, sich also eine Person vor dem Sensor aufhält. Die auszulösende Funktion simuliert einen Mausklick und weckt so den Bildschirm aus dem Schlafmodus. Ausserdem wird die Boolesche[55] Variable mit dem Namen *kundenvariable* auf den Zustand „*True"* geschaltet. Diese Variable verhindert

[55] Eine Variable, welche nur zwei Zustände, meist „*True*" und „*False*", annehmen kann

durch eine Bedingungskonstruktion, dass die oben genannte Funktion dauernd wiederholt wird, solange der Kunde vor dem Automaten steht.

Die Becherüberwachung hat den Zweck, das Bestellen von Drinks ohne einen Becher unter dem Ausschank zu verhindern. Diese Überwachung ist sinnvoll, weil ausgelaufene Zutaten nicht mehr geniessbar wären und damit verschwendet würden.

Die Becherüberwachung funktioniert sehr ähnlich wie die Kundenerkennung, allerdings sind die Grenzwerte tiefer gewählt, weil sich der Sensor hier viel näher am Objekt befindet als bei der Kundenerkennung. Auch bei der Becherüberwachung gibt es eine Boolesche Variable, die *bechervariable*, welche vor jedem Mischvorgang überprüft wird. Befindet sie sich nicht im Zustand „*True*“, weist der Automat den Kunden darauf hin, einen Becher unter dem Auslauf zu positionieren.

5.4.4. Temperaturmessung

Zur Überwachung der Innentemperatur dienen zwei Temperatursensoren. Der Automat ist in zwei Bereiche aufgeteilt, wobei der untere Bereich hauptsächlich Elektronik enthält. Dieser produziert laufend Abwärme, weshalb es im unteren Teil deutlich wärmer ist. Im oberen Teil befinden sich die Zutaten, welche möglichst kühl gelagert werden müssen. Wegen der entstehenden Temperaturdifferenz werden beide Bereiche getrennt überwacht. Der Grenzwert für eine Fehlermeldung liegt im oberen Teil entsprechend tiefer. Im oberen Bereich sollte eine Temperatur von maximal 6° Celsius herrschen, im unteren Bereich darf die Temperatur maximal 20° Celsius betragen.

5.4.5. PIN-Abfrage

Zur Begrenzung des Konsums und zur Absicherung gefährlicher Funktionen des Automaten verfügt die Software an zwei Orten über eine Code-Sperre. Der Code besteht ausschliesslich aus Zahlen, was man als PIN (Persönliche Identifikationsnummer[56]) bezeichnet. Die Sperre funktioniert mit einer einfachen *if*-Bedingung, welche die Eingabe mit den gültigen Codes vergleicht. Zur Eingabe wird ein Zahlenfeld eingeblendet, welches dem eines Taschenrechners nachempfunden ist.

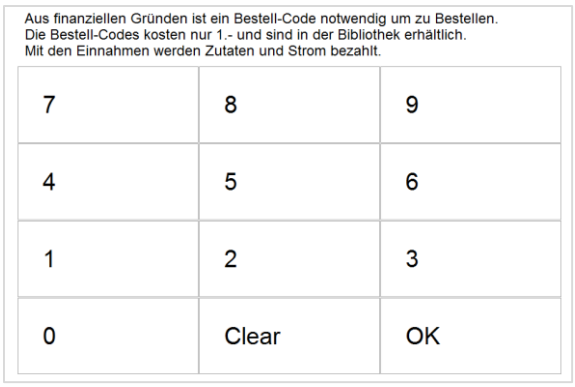


Bild 35: PIN-Sperre vor dem Mischvorgang

Eine der PIN-Sperren soll wie, bereits erwähnt, den Konsum begrenzen. Sie befindet sich daher direkt

[56] Quelle: http://wirtschaftslexikon.gabler.de/Archiv/1598/pin-v9.html

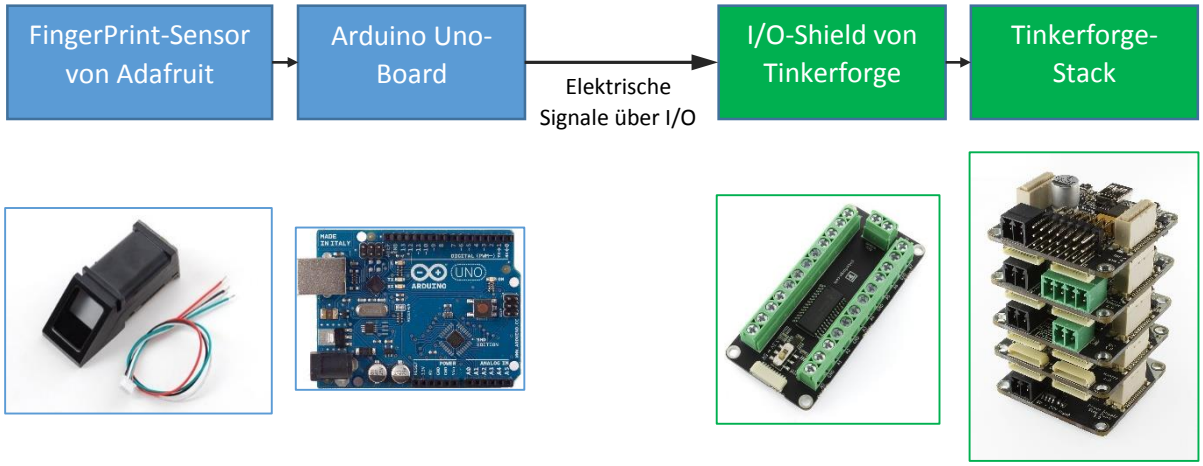
vor dem Beginn des Mischvorgangs, um den Grossteil der Software auch ohne gültigen Code nutzbar zu machen. Es gibt zwei Kategorien von gültigen Codes: Die erste umfasst nur eine Zahlenabfolge, die sogenannte *Admin-PIN* und ist nur dem Administrator des Automaten bekannt. Die zweite Kategorie besteht aus deutlich mehr Codes, welche von Kunden erworben werden können. Die Wahrscheinlichkeit, dass ein Nutzer die Sperre ohne gültige PIN bei einmaliger Eingabe überwinden kann, liegt bei etwa 0.2 %. Dieser Wert dürfte für die Anwendung völlig ausreichend sein.

Eine zweite PIN-Sperre schützt den Wartungsbereich der Software. In diesem Teil des Programms befinden sich heikle Funktionen, wie etwa das Ausleeren aller Flaschen. Des Weiteren befinden sich im Wartungsbereich Werkzeuge, um Fehlermeldungen aufzuheben oder das Programm zu beenden. Die Sperre zum Wartungsbereich besitzt nur einen gültigen Code, die Wahrscheinlichkeit eines erfolgreichen Angriffs bei einmaliger Eingabe liegt hier bei 0.0001 %.

5.4.6. Nutzeridentifikation über Fingerabdruck

Bestimmte Nutzer des Automaten, sogenannte Premiumkunden, können sich mit ihrem Fingerabdruck in einen sonst gesperrt Premium-Bereich der Software einloggen.

Der Fingerabdruck wird dabei von einem Sensor abgelichtet und mit einer internen Datenbank, in welcher die Fingerabdrücke aller zutrittsbewilligten Kunden abgelegt sind, verglichen. Falls eine Übereinstimmung vorliegt, sendet der Sensor die ID-Nummer des erkannten Kunden an das verbundene Arduino® Uno-Board. Dieses leitet die Nummer als elektrisches Signal an den Tinkerforge-Stack weiter. Das elektrische Signal wird folgendermassen erzeugt: es stehen insgesamt vier digitale Pins zur Kommunikation bereit. Die erkannte ID wird dann als Abfolge von Nullen und Einsen beschrieben, also als Binär-Code. Dieser Binär-Code wird an den Pins als Abfolge von logischen *High*- beziehungsweise *Low*-Zuständen angezeigt. Diese Abfolge wird von einem I/0-Bricklet von Tinkerforge ausgelesen und an den Stack weitergeleitet. Als Schema beziehungsweise in Bildern dargestellt sieht der Ablauf der Nutzeridentifikation folgendermassen aus:



Mit Hilfe der gesendeten Kunden-Nummer wird dann ein individuelles Fenster des gerade eingeloggten Kunden aufgebaut, welches der mitgeteilten ID entsprechend angepasst wird. So ergibt sich für jeden Premium-Kunden eine eigene, ganz persönliche Premium-Lounge. Diese Individualität steigert das Gefühl des Kunden, vom Gerät wertgeschätzt und beachtet zu werden.

5.4.7. Speichern von eigenen Rezepten

Premium-Kunden ist es gestattet, ihre eigenen Lieblingsrezepte abzuspeichern und später wieder zu verwenden oder abzuändern. Diese Anpassungen können die Kunden in ihrer persönlichen Premium-Lounge vornehmen. In diesem Bereich befinden sich ein Fenster zum Abspeichern von neuen Rezepten und eines zum Laden von zuvor gespeicherten Rezepten.

Speichert der Kunde ein Rezept, werden die gewählten Zutaten als Zahlenfolge in einer Textdatei auf dem Computer abgespeichert. Der Kunde kann dem Rezept einen eigenen Namen geben, wobei dieser Name dann der Name der Textdatei ist. Für jeden Kunden wird auf dem Zentralrechner ein Rezepte-Ordner erstellt, in welchem die Textdateien abgespeichert werden.

Die so gespeicherten Textdateien werden im Laden-Fenster als Knöpfe aufgelistet. Betätigt der Kunde einen dieser Knöpfe, wird das Rezept geladen und der Kunde wird direkt zum „Dein eigener Drink“-Fenster geleitet. Dort werden die Zutatenmengen dem geladenen Rezept entsprechend angepasst. Damit ist das abgespeicherte Rezept geladen und kann zubereitet werden.

5.4.8. Drinkzähler

Um die Nachfrage und die Beliebtheit der verschiedenen Drinks beurteilen zu können, verfügt die Automaten-Software über Drinkzähler. Diese speichern für jeden Drink die Anzahl Bestellungen. Angezeigt werden diese Werte im „Status des Automaten“-Fenster, sind also öffentlich zugänglich.

Als Zähler kommen, wie bei der Bestellungsaufnahme, *IntVar()* zum Einsatz. Diese werden bei jeder Bestellung entsprechend erhöht. Dazu wird der alte Wert der Variable zwischengespeichert und um 1 erhöht. Die Variable wird dann auf den zwischengespeicherten Wert aktualisiert.

5.4.9. Laden der Dateien

Da im GUI viele Bilder und externer Text angezeigt werden, muss das Programm beim Start auf eine grosse Anzahl von Dateien zugreifen. Solche Zugriffe bergen für ein Programm jedoch einige Risiken, da eine nicht vorhandene Datei einen Fehler verursachen kann. Das Laden der Dateien geschieht deshalb zentral, also an einem einzelnen Ort im Programmcode, was alles übersichtlicher macht und das Implementieren von Fehlerabfangmechanismen vereinfacht.

Die Dateien befinden sich aufgeteilt in Unterordner auf einem lokalen Datenträger. Die Ordnerpfade müssen dem Programm mitgeteilt werden. Das Programm geht dann in die entsprechenden Ordner

und erstellt daraus eine Liste an nutzbaren Dateien, das heisst, es werden nur Dateien aufgelistet, welche den richtigen Dateityp besitzen. Beim Aufbauen des GUIs greift das Programm dann auf die erstellten Listen zu. Alle diese kritischen Vorgänge sind mit sogenannten *try/except*-Mechanismen abgesichert. Bei fehlenden Dateien gibt das Programm eine Fehlermeldung aus, anstatt abzustürzen.

5.4.10. Fehlerübermittlung via E-Mail

Der Automat ist in der Lage, sich bei Problemen oder Fehlern selbstständig bei seinem Administrator zu melden. Diese Fähigkeit verringert den Wartungsaufwand, da der Admin genau Bescheid weiss, wann und weshalb er den Automaten warten muss. Fehlermeldungen werden beispielsweise bei zu hoher Innentemperatur oder leeren Flaschen versendet. Realisiert wurde diese Funktion mit Hilfe der Python-Erweiterung *smtplib (engl.: Simple Mail Transfer Protocol Library).* Diese ermöglicht es, einen Server eines Mail-Providers zur Übertragung von automatisch generierten Nachrichten zu benutzen.

6. Umsetzung der Hardware

In diesem Kapitel wird die Realisierung der Hardware beschrieben. Der fertige Roboter wird anhand zahlreicher Fotos präsentiert, erklärende Texte ergänzen die Bilder.

6.1. Eckdaten der Hardware

Der Getränkeautomat kann bis zu acht Flaschen gekühlt lagern. Gesteuert über drei Bricks betätigen acht Relais ebenso viele Magnetventile. Unter den Ventilen bewegt sich ein Glastrichter hin und her, angetrieben wird er von einem Schrittmotor. Vier Sensoren überwachen laufend die Umgebung des Automaten. Ein demontierter Laptop dient als zentrales Steuerungssystem. Der Automat ist nach vorne hin verglast und die Elektronik nicht in ein Gehäuse eingebaut, damit der Kunde das spannende Innenleben und den Mischprozess genau beobachten kann.

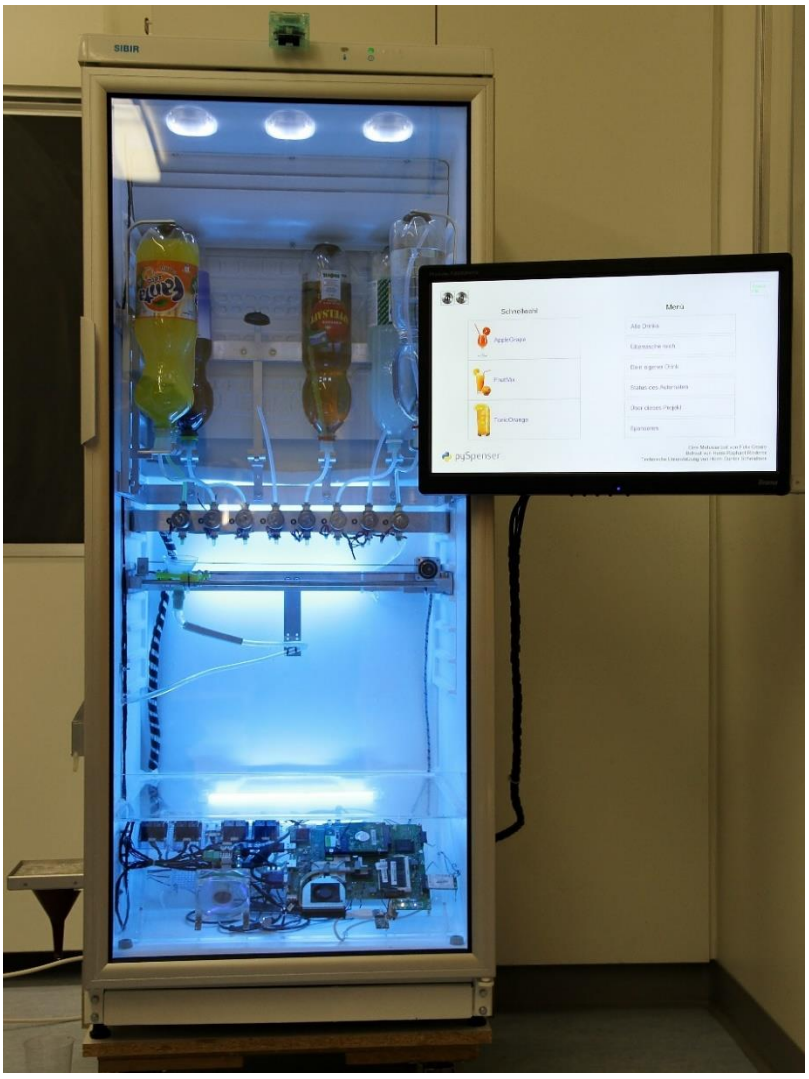


Bild 36: Gesamtansicht des fertigen Automaten

6.2. Komponenten der Hardware

6.2.1. Kühlung

Da die verwendeten Zutaten gekühlt werden müssen, befindet sich der ganze Automat in einem Kühlschrank. Dieser stammt von der Firma Sibir™ und trägt den Produktnamen FKS 292. Er hat ein Nutzvolumen von 275 Litern und eine verglaste Front. Ausserdem verfügt er über eine Abtau-Vollautomatik, welche die Eisbildung im Innern des Automaten minimiert. Die durchschnittliche Leistungsaufnahme des Kühlsystems liegt bei etwa 85 Watt.

6.2.2. Lagerung der Flüssigkeiten

Diese Komponente besteht aus den Flaschenhalterungen, den selbst konstruierten Standflächen für die Flaschen und den stabilen Verbindungsschienen aus Aluminium. Die Funktion der Konstruktion ist selbsterklärend: In den Halterungen können Flaschen mit 0.7 bis 1.5 Litern Inhalt gelagert werden. Insgesamt können also bis zu 10.5 Liter Zutaten gekühlt werden. Neben den Zutaten wird auch die Spülflüssigkeit hier gelagert, von der bis zu 1.5 Liter gespeichert werden können.

Bild 37: Die befüllte Flüssigkeitslagerung

6.2.3. Dosiermechanik

Die Dosiermechanik des Roboters besteht aus folgenden Teilen: den Ventilen, der Trichter-Positionierung und dem Ausschank. Auch die verbindenden Schläuche gehören dazu, spielen aber eine untergeordnete Rolle, da sie keine eigenständige Funktion ausüben, sondern verschiedene Funktionselemente verbinden.

Bild 38: Die Dosiermechanik

Die Steuerung der Ventile übernehmen vier Relais-Bricklets, welche an den Tinkerforge-Stack angeschlossen sind und von der Software gesteuert werden. Befestigt sind die Ventile an einer U-förmigen Aluminiumschiene. Die Ventile werden mit einer Spannung von 24 Volt betrieben. Die Dichtungen zwischen den Schläuchen und den Ventilen bestehen aus Messing und sind mit Teflonband abgedichtet.

Die Schlauchverbindungen bestehen aus PE (Polyethylen) und haben einen Innendurchmesser von 4 Millimetern; der Aussendurchmesser beträgt 6 Millimeter. Die Schläuche sind transparent, um die Fliessbewegung der Zutaten erkennbar zu machen.

Der Glastrichter, der die dosierten Zutaten auffängt, ist in einen Plastiktrichter eingelassen, welcher selbst wiederum an einem beweglichen Wagen aus Kunststoff befestigt ist. Als Fahrschienen dienen zwei Aluminiumstangen, welche parallel zueinander durch den Automaten verlaufen. Der Wagen wird über einen Zahnriemen von einem Schrittmotor bewegt, welcher von einem Stepper-Brick gesteuert wird. Der Wagen läuft auf Kugellagern, um die Reibung zu verringern.

Als Ausschank dient eine Aluminiumkonstruktion mitsamt Auffangplattform. Diese ermöglicht es, beim Spülvorgang oder bei einer Fehlfunktion die auslaufende Flüssigkeit zuverlässig abzuleiten.

Bild 39: Der Ausschank inklusive dem Bechererkennungs-Sensor

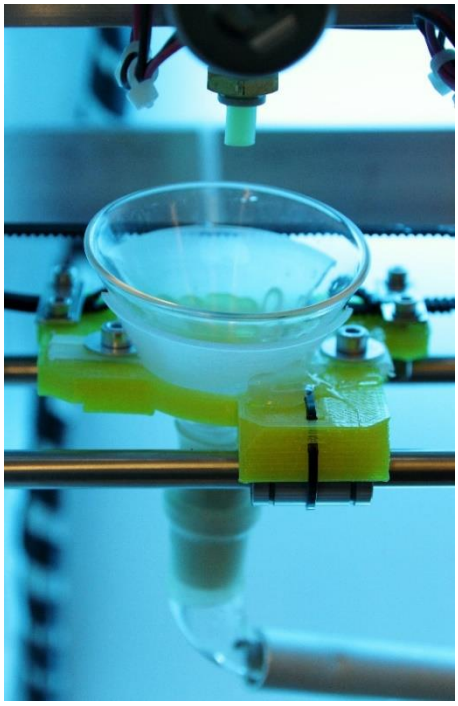


Bild 40: Der Glastrichter inklusive Fahrwagen

6.2.4. Benutzerschnittstelle

Als Benutzerschnittstelle kommt wie in Kapitel *2.2.* beschrieben ein Touchscreen zum Einsatz. Der Bildschirm hat eine Bilddiagonale von 22 Zoll und löst mit 1920 × 1080 Pixeln auf, als Hintergrundbeleuchtung werden CCFL-LED-Lampen verwendet. Die Nutzereingaben werden über ein USB-Kabel an den Zentralrechner geleitet. Sein Gewicht beträgt 4.5 Kilogramm, die Leistungsaufnahme liegt bei 20 Watt. Der Bildschirm ist mit einer beweglichen Bildschirmhalterung am Kühlschrank befestigt.

6.2.5. Zentrales Steuerungssystem

Als Zentralrechner wird ein komplett demontierter Laptop verwendet, auf dem Windows® 7 in der 32-bit Version als Betriebssystem installiert ist. Als Prozessor kommt ein Intel® Pentium® Dual Core zum Einsatz, welcher mit 2.00 Gigahertz taktet. Dem Prozessor stehen 2 Gigabyte Arbeitsspeicher für schnelle Speichervorgänge zur Seite, für langfristiges Abspeichern von Daten steht eine 500 Gigabyte grosse Festplatte zur Verfügung. Der Rechner kann sich dank einer WLAN-Karte kabellos mit dem Internet verbinden. Dies ist wichtig, damit Funktionen wie die Fehlerbenachrichtigung via E-Mail einwandfrei funktionieren.

6.2.6. Mikro-Controller von Tinkerforge

Die drei verwendeten Bricks und die Stromversorgungs-Platine sind als einzelner Stack angeordnet, an diese Platine angeschlossen sind acht Bricklets, das Stromkabel und die USB-Verbindung zum Zentralrechner. Die Relais-Bricklets sind nebeneinander platziert, um das Kabelmanagement möglichst übersichtlich zu gestalten. Die Bricks werden mit einer Spannung von 24 Volt versorgt, damit auch der Schrittmotor fehlerfrei betrieben werden kann. Wird dieser in kurzer Zeit oft beansprucht, kommt es zu einer starken Wärmeentwicklung auf dem steuernden Stepper-Brick. Der verbaute Kühlkörper auf der Platine reicht nicht aus, wodurch der Brick zu heiss wird und den gesamten Stack instabil macht, was mitunter zu Abstürzen führen kann. Um dem vorzubeugen wird der Stack von einem Lüfter gekühlt. Der Kühler ist transparent und mit blauen LEDs versehen, damit er perfekt zum restlichen Design des Automaten passt.

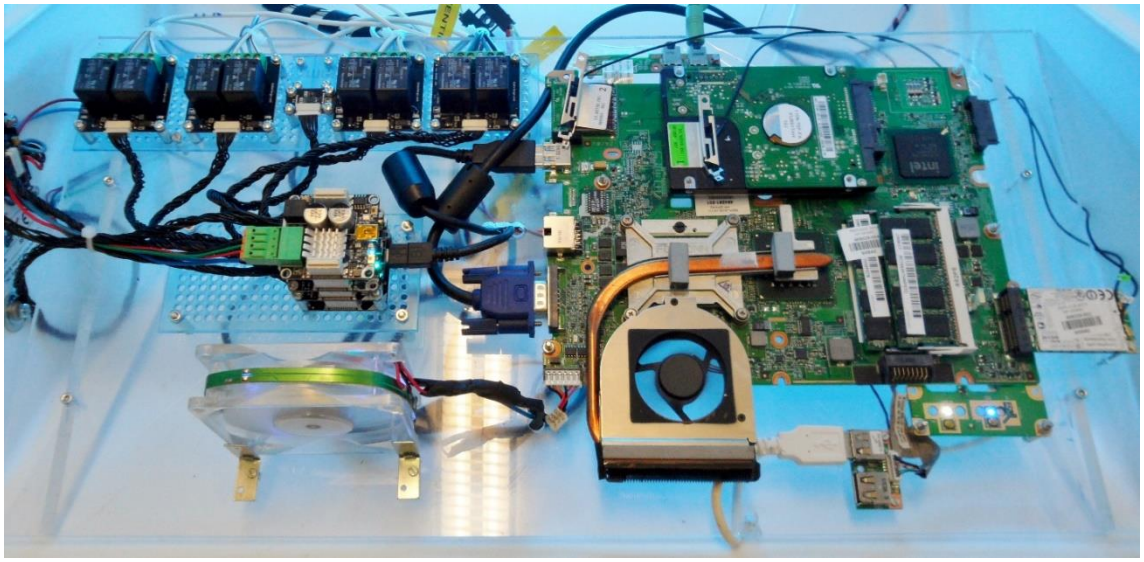


Bild 41: Elektronik im Innern des Automaten

Die relativ tiefen Temperaturen im Kühlschrank können dazu führen, dass das in der Luft enthaltene Wasser kondensiert. Die dadurch entstehenden Wassertröpfchen könnten die Elektronik durch einen Kurzschluss auf einer der Platinen massiv beschädigen. Um dem vorzubeugen, befinden sich im unteren Bereich des Automaten vier Luftentfeuchter. Diese Beutel sind mit einem Silikatgel befüllt und färben sich bei erreichter Aufnahmekapazität rosa. Danach können sie in einer Mikrowelle getrocknet und wiederverwendet werden. Die Beutel haben je eine Aufnahmekapazität von 60 Millilitern, insgesamt können also bis zu 240 Milliliter Wasser gespeichert werden.

6.2.7. Mikro-Controller von Arduino®

Der FingerPrint-Sensor kann nicht direkt mit den Mikro-Controllern der Firma Tinkerforge kommunizieren, sondern funktioniert nur zusammen mit einem Arduino®-Board. Im Automaten befindet sich darum ein Arduino® Uno, welches mit dem Sensor verbunden ist. Das Board arbeitet autonom, wird also im Gegensatz zu den Tinkerforge-Komponenten nicht von einem Zentralrechner gesteuert. Es wird über ein 12 Volt Netzteil mit Strom versorgt.

6.2.8. Sensoren

Der Automaten verfügt über fünf Sensoren: jeweils zwei Temperatur- und Distanzsensoren und einen Fingerabdruck-Sensor. Die Temperatursensoren sind ohne Gehäuse im Innern des Geräts platziert. Wie bereits erwähnt, überwachen die Sensoren jeweils einen der beiden Abteile. Logischerweise ist also in jedem Abteil ein Temperatursensor verbaut.

Die Distanzsensoren sind deutlich aufwendiger montiert. Damit die Benutzererkennung die Umwelt möglichst wahrheitsgetreu erkennen kann, ist der Sensor ganz oben am Automaten befestigt und schaut auf den Kunden herab. Dieser Sensor ist aus ästhetischen Gründen in einem Gehäuse eingeschlossen, ausserdem ist er so einfacher zu montieren und besser geschützt.

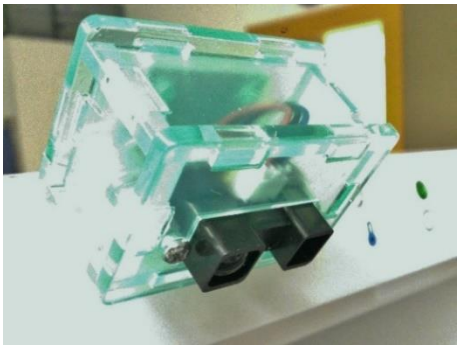


Bild 42: Distanzsensor zur Benutzererkennung

Der Distanzsensor für die Bechererkennung ist direkt beim Auslauf an die Kühlschrankwand geschraubt. Er zeigt genau auf den Ort, an dem der Becher beim Mischvorgang steht, um möglichst gute Resultate zu liefern. Das System ist aber leider überwindbar: man kann den Sensor mit einem anderen Gegenstand täuschen oder den Becher während des Mischvorgangs entfernen. Ich vertraue hier aber auf den guten Willen der Nutzer und gehe nicht von einem mutwilligen Überlistungsversuch aus. Da eine Fehlfunktion des Automaten nicht auszuschliessen ist, befindet sich unter dem Auslauf eine durchlässige Plattform. An der Unterseite dieser Plattform ist ein Trichter befestigt, der in einen Auffangkanister mündet.

Der Fingerabdruck-Sensor dient der Erkennung von Premium-Kunden. Der Sensor stammt von Adafruit und basiert wie bereits beschrieben auf der Arduino-Plattform. Die Fingerabdruck-Erkennung ist wegen des Prototypen-Status noch extern montiert, später könnte sie auch stärker in das Gerät integriert werden und beispielsweise am Touchscreen befestigt werden, um für den Kunden stets gut erreichbar zu sein.

6.2.9. Beleuchtung

Der Automat verfügt über verschiedene Leuchtelemente, welche ihn ästhetisch aufwerten und die Aufmerksamkeit des Kunden wecken.

Neben jedem Magnetventil sitzt jeweils eine kleine, blaue LED. Die LEDs werden mit den Ventilen parallel geschalten, sie leuchten also immer, wenn ihr zugehöriges Ventil geöffnet ist. Vor den LEDs sitzen ausserdem Vorwiderstände, um die 24 Volt Ventilspannung auf 2.7 Volt Spannung zu senken. Eine Schutzdiode schützt die LED zudem vor zu hohen Spannungen, welche beim Schliessen der Ventile entstehen. Dieser Effekt tritt relativ häufig auf, wenn man mit Spulen arbeitet, wie sie in den Magnetventilen verwendet werden.

Nebst den kleinen LEDs sind im Innern des Roboters noch drei *CCFL-Leuchtstoffröhren*[57] montiert. Sie sind zum Teil nach vorne hin abgeschirmt und strahlen so nur die Rückwand des Automaten an, was einen sehr ansprechenden optischen Eindruck erzeugt. Die Elektronik zur Speisung der Beleuchtung ist im ganzen Gerät verteilt, weil die verwendeten Leuchtstoffröhren bei verlängerten Kabeln nicht hell genug leuchten. Die Leuchtstoffröhren werden von *Invertern*[58] gespeist, welche die Eingangsspannung von 12 Volt auf über 220 Volt hochtransformieren. Zum Vorbeugen vor lebensgefährlichen Stromschlägen sind die Inverter in Plastikgehäusen untergebracht.

6.2.10. Sicherung des Automaten

Um den Automaten vor schädigender Fremdeinwirkung zu schützen, kann er zur Lagerung verschlossen werden. Dazu wird der Touchscreen über ein Kensington-Schloss mit einem Metallkabel verbunden. Dieses wird hinten um den Roboter geführt und kann auf der anderen Seite, genauer am Türgriff des Kühlschranks, mit einem Vorhängeschloss befestigt werden. Durch diese Sicherung ist es unmöglich, ans Innere des Roboters zu gelangen oder den Touchscreen zu bewegen.

---

[57] CCFL-Leuchtstoffröhren (*engl.: Cold Cathode Fluorescent Lamp*) sind mit Gas gefüllte Leuchtröhren, welche beim Anlegen einer hohen Wechselspannung in einer vom Füllgas abhängigen Farbe leuchten.
Quelle: http://de.wikipedia.org/wiki/Leuchtr%C3%B6hre

[58] Inverter sind Richter, welche Gleichstrom in Wechselstrom umwandeln.
Quelle: http://de.wikipedia.org/wiki/Inverter_(Energietechnik)

6.3. Getränkeangebot

Wie im Kapitel 6.1. beschrieben, können im Automaten bis zu acht Flaschen gelagert werden, wobei eine davon der Spülung dient. In den sieben anderen Flaschen werden folgende Zutaten gelagert: Fanta, IceTea, Orangensaft, Ananassaft, Apfelsaft, Grapefruit-Limonade und Zitronen-Limonade. Aus diesen Zutaten kann der Automat acht verschiedene vorgegebene Drinks mischen:

Drink	Zutaten	Drink	Zutaten
AppleGrape	12 cl Apfelsaft 8 cl Grapefruit-Limonade	**IceFanta**	6 cl Fanta 14 cl IceTea
PineCitroTea	11 cl IceTea 3 cl Ananassaft 6 cl Zitronen-Limonade	**DancingQueen**	8 cl Orangensaft 9 cl Ananassaft 3 cl Grapefruit-Limonade
FruitMix	9 cl Orangensaft 4 cl Ananassaft 7 cl Apfelsaft	**LimoParty**	8 cl Fanta 5 cl Grapefruit-Limonade 7 cl Zitronen-Limonade
LuckyStrike	6 cl IceTea 4 cl Orangensaft 5 cl Apfelsaft 5 cl Grapefruit-Limonade	**SourBitter**	2 cl Fanta 2 cl Apfelsaft 9 cl Grapefruit-Limonade 6 cl Zitronen-Limonade

Dank des „Dein eigener Drink“-Fensters ist der Anzahl mischbarer Drinks aber keine Grenze gesetzt. Der Kunde kann seiner Kreativität freien Lauf lassen und sich selbst sein Lieblingsgetränk zubereiten. Das verfügbare Angebot kann auch nachträglich noch grundlegend modifiziert werden. Man kann andere Zutaten in die Flaschenhalterungen einspannen, muss dies der Software aber mitteilen. Dazu sind einige kleine Änderungen am Quellcode nötig, der dabei entstehende Aufwand ist aber sehr gering.

6.4. Testphase

Um Fehlerquellen und Unklarheiten in der Software zu finden, wurde der Automat nach der erfolgreichen Konstruktion ausgiebig getestet. Die erste Testphase bestand darin, den Automaten ohne Zutaten zu bedienen und jede mögliche Abfolge von Aktionen zu testen. Diese Phase deckte einige Fehler in der Software auf, welche direkt behoben werden konnten. Diese erste Testphase habe ich alleine absolviert, da ein Kunde den Automat noch nicht perfekt hätte bedienen können.

Die erste Testphase hat einen relativ schwerwiegenden Fehler aufgedeckt: die Bechererkennung funktioniert nur bei Bechern, welche für den Infrarot-Sensor nicht völlig durchsichtig sind. Wird also beispielsweise ein transparenter Becher mit einigermassen senkrechten Seitenwänden verwendet, gibt der Automat fälschlicherweise eine Fehlermeldung aus. Das Problem lässt sich lösen, indem man entweder nicht transparente Becher verwendet oder den Sensor mit einem undurchsichtigen Gegenstand austrickst, wie es in der nebenstehenden Abbildung zu sehen ist.

Nachdem alle weniger gravierenden Fehler ausgebessert waren, entschloss ich mich, eine zweite Testphase mit realen Testkunden durchzuführen. Dieser Test sollte vor allem überprüfen, ob das GUI einfach bedienbar und selbsterklärend ist. Dazu habe ich einige Klassenkameraden zu einem kostenlosen Drink aus dem Automaten eingeladen. Das Feedback war weitgehend sehr positiv, viele der Testpersonen waren vom Gerät sogar begeistert. Dies zeigt, dass ich das zweite Ziel der Arbeit erreicht habe: der Automat fasziniert und beweist, dass mit Informatik und Elektrotechnik komplexe und begeisternde Aufgabenstellungen gelöst werden können.

6.5. Konstruktionsbericht

In diesen Abschnitt beschreibe ich den Bauprozess der Hardware. Der Physikassistent der Kantonschule, Herr Günter Schmidtner, war mir dabei eine grosse Hilfe.

Der Automat wurde während der Herbstferien 2013 in der Physikwerkstatt der Kantonsschule Schaffhausen konstruiert. In der Werkstatt konnte ich professionelles Werkzeug benutzen, zum Beispiel eine Standbohrmaschine und eine Bandsäge. Ich habe sehr viel Dinge selbst herstellen können und dabei viel gelernt.

Die Konstruktion des Automaten dauerte etwa sieben Arbeitstage. Am ersten Tag wurden die U-förmigen Halterungen und die Standflächen für die Flaschen gefertigt. Dazu wurde fünf Millimeter dickes Aluminium verwendet, welches gut formbar ist.

Bild 43: Standbohrmaschine

Bild 44: Standfläche der Flasche

Am zweiten Tag haben wir die Befestigungslöcher in die U-Halterungen gebohrt, die Flaschenhalterungen montiert und damit begonnen, die Mechanik für die Linearschienen-Trichterpositionierung zu planen.

Einen Tag später haben wir die Dosiermechanik fertig konstruiert, die Ventile angeschraubt und die Steuerungselektronik auf der Plexiglasplatte befestigt. Auch die Trichter-Positionierung wurde am dritten Tag gebaut und erfolgreich getestet.

Am vierten Tag wurden die benötigten Löcher in den Kühlschrank gebohrt und der Ausschank konstruiert. Ausserdem platzierten wir die ersten Leuchtstoffröhren im Kühlschrank.

Am fünften Tag konnten wir dann die Infrarot-Sensoren und den Schlauch zum Ausschank montieren. Dieser musste zusätzlich mit Kabelbinder befestigt werden, um zu verhindern, dass er zu stark gekrümmt wird und sich die Flüssigkeit in der Krümmung sammelt. Ausserdem haben wir die Ventile mit den Relais verkabelt.

Am sechsten Tag haben wir den Bildschirm am Kühlschrank befestigt und die Silikondichtungen der Löcher und der Plexiglasscheibe gespritzt.

Am letzten Tag wurden die Stromkabel für die Elektronik im Innern des Automaten verlegt und den Automaten gründlich gereinigt.

Bild 45: Montierte Flaschenhalterungen

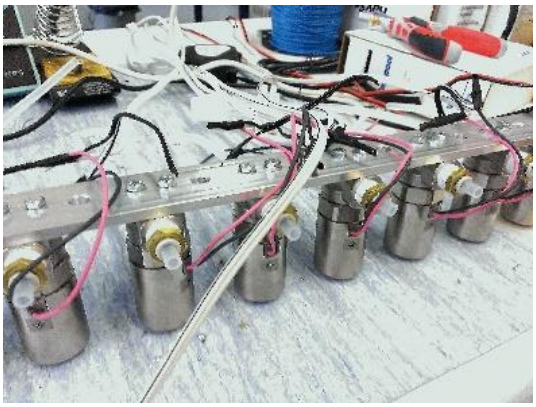


Bild 46: Löten der Ventilverkabelung

7. Fazit

Im diesem Kapitel meiner Maturaarbeit möchte ich zurückschauen und ein Fazit aus der Arbeit ziehen. Ausserdem möchte ich das fertige Projekt mit den anfangs vorgestellten Cocktail-Robotern vergleichen.

7.1. Rückblick

Nach über einem Jahr intensiver Arbeit möchte ich in diesem Abschnitt beschreiben, was während der Durchführung meiner Maturaarbeit gut verlaufen ist und was besser hätte laufen können.

Mit dem Endergebnis meiner Arbeit, dem Getränkeautomaten „*pySpenser*“ mitsamt der Software, bin ich überaus zufrieden. Das ansprechende Design, die umfangreiche Software und besondere Features wie etwa die Benutzer- und Bechererkennung sind sehr gut gelungen. Während der Maturaarbeit habe ich ausserdem unglaublich viel gelernt, vom Programmieren mit Python über den Umgang mit professionellem Werkzeug bis hin zum Verfassen eines wissenschaftlichen Berichts.

Die beiden entscheidenden Prozesse, die Konstruktion und das Programmieren der Software, sind sehr gut verlaufen und haben weniger Zeit benötigt, als ich erwartet hatte. Diese Arbeiten haben mir ausserdem am meisten Spass gemacht. Die Planungsphase hat hingegen deutlich länger gedauert, als ich mir das ursprünglich vorgestellt hatte. Schlussendlich ist es vielleicht aber auch der genauen Planung zu verdanken, dass meine Ideen genau in der Art umgesetzt werden konnten, wie ich es mir vorgestellt hatte.

Die beiden Hauptziele meiner Maturaarbeit habe ich klar erfüllen können. Ich habe einen Roboter entwickelt, welcher in akzeptabler Zeit vollautomatisch Cocktails zubereiten kann.

Der Roboter ist ausserdem optisch sehr ansprechend und macht dem Kunden klar, dass es sehr viel Elektronik und Software benötigt, um einen Roboter einwandfrei betreiben zu können. Des Weiteren hat der Kunde in der Software viele Möglichkeiten, sich über die technischen Details des Automaten zu informieren. Der Automat kann den Kunden beziehungsweise den Schülerinnen und Schülern der Kantonsschule auch klar machen, dass man als Kantonsschüler mit Hilfe von Informatik und Elektrotechnik sehr komplizierte Problemstellungen lösen und faszinierende Produkte entwickeln kann.

7.2. Vergleich mit eigenem Getränkeautomaten

In diesem Abschnitt soll der fertige Getränkeautomat mit ähnlichen Projekten verglichen werden. Diese Projekte wurden im zweiten Kapitel der Arbeit bereits vorgestellt.

Im Hinblick auf die Funktion mögen die beiden Projekte meinem Automaten sehr ähnlich sein, sie bereiten beide auch vollautomatisch Cocktails zu. Die Roboter haben dennoch einen völlig anderen Einsatzbereich. Währenddem die beiden vorgestellten Projekte nur über kurze Zeiträume verwendbar sind, also zum Beispiel für ein Fest, war eines der Ziele meines Projekts das Entwickeln eines beinahe wartungsfreien Automaten, welcher langfristig und dauerhaft betrieben werden kann.

Ein direkter Vergleich wird somit ziemlich schwierig, auch wenn sich mit Sicherheit sagen lässt, dass beide Konkurrenzprodukte über eine deutlich grössere Drinkauswahl verfügen. Beim *Bartendro* ist zudem die Dosiergenauigkeit enorm hoch und die schlichte Bauweise bietet einen Mobilitätsvorteil. Der effektvollste Roboter ist sicherlich *melmacc*, vor allem der Roboterarm zum Zufügen der Strohhalme ist sehr beeindruckend.

8. Ausblick

Im letzten Kapitel der Arbeit möchte ich aufzeigen, was mit dem Projekt in Zukunft noch alles möglich wäre. Nachfolgend sind einige Verbesserungen vorgestellt. Ausserdem wird die Reproduzierbarkeit des Geräts beschrieben und es werden Erweiterungen präsentiert, welche dem Projekt den Zugang zum Massenmarkt ermöglichen könnten.

8.1. Kurzfristige Verbesserungen

Hier möchte ich einige kleine Verbesserung beschreiben, welche ich im Laufe des nächsten Jahres noch einbauen könnte. So möchte ich dem Nutzer die Wahl der Bechergrösse ermöglichen und ihm mehr Informationen zu den verwendeten Zutaten und angebotenen Cocktails bereitstellen. Ein stärkeres optisches Feedback des GUI, also ein kurzes Aufblinken eines gedrückten Knopfes, soll dem Nutzer ausserdem deutlicher zeigen, dass seine Eingabe erkannt wurde.

Falls noch weitere Geldgeber gefunden werden, würden die Messing-Dichtungen vor und nach den Magnetventilen durch Modelle aus Edelstahl ersetzt werden, um den Geschmack der Getränke nicht zu beeinflussen und das mögliche Ablösen von Schwermetallen ganz zu verhindern.

8.2. Reproduzierbarkeit

In diesem Kapitel der Arbeit soll untersucht werden, wie gut sich der entwickelte Getränkeautomat reproduzieren, also erneut herstellen lässt und wie diese Reproduzierbarkeit noch verbessert werden könnte.

Während des gesamten Konstruktionsprozesses war sehr viel Handarbeit nötig, gerade die Fertigung der Metallkonstruktionen im Inneren des Geräts war sehr zeitintensiv. Bezüglich dieser Komponente ist die Reproduzierbarkeit also eher gering. Der Fertigungsprozess könnte durch eine Kooperation mit einem professionellen Metallbauer zwar stark beschleunigt werden, dadurch würden aber die Kosten steigen.

Da die Kühlung nur aus einem Kühlschrank besteht, ist diese Komponente sehr leicht reproduzierbar. Die Dosiermechanik ist ebenfalls einfach reproduzierbar, dafür müssten lediglich neue Ventile und Schläuche beschafft werden. Auch hier sind die Metallverbindungen die aufwendigste Komponente.

Bezüglich der Steuerungselektronik wäre der Aufwand bei einer Reproduktion sehr gering. Alle Tinkerforge-Komponenten können im Internet bestellt werden, als Zentralrechner könnte eine günstige Entwickler-Platine wie das *Raspberry Pi* verbaut werden.

Die Software könnte praktisch ohne Aufwand reproduziert werden. Dafür müssten lediglich einige Codezeilen abgeändert werden, welche die Kommunikation mit den Tinkerforge-Bausteinen betreffen. Konkret müssten die Platinen-spezifischen UID-Nummern aktualisiert werden, danach würde die Software einwandfrei funktionieren.

Zusammenfassend lässt sich also festhalten, dass der Grossteil des Automaten sehr einfach erneut hergestellt beziehungsweise vervielfacht werden kann. Die metallischen Verbindungsteile sind zwar reproduzierbar, dazu muss aber grosser zeitlicher oder finanzieller Aufwand betrieben werden.

8.3. Markttauglichkeit des aktuellen Prototypen

In diesem Abschnitt wird die Markttauglichkeit des aktuellen Prototyps untersucht.

Die Herstellung des aktuellen Prototyps war sehr kostspielig: dies liegt in erster Linie an den hohen Materialkosten und der zeitintensiven Fertigung des Geräts. Sollen diese Kosten gedeckt werden, muss ein sehr hoher Verkaufspreis verlangt werden. Die hohen Anschaffungskosten sind vor allem für kleinere Betriebe ein finanzielles Risiko.

Auch die beschränkte Auswahl an Zutaten mindert die Markttauglichkeit, da das Gerät nur ein kleines Sortiment anbieten kann und daher für den produktiven Einsatz in einer Bar (also den tatsächlichen Ersatz eines Barkeepers) nur bedingt geeignet ist.

Eine weitere Schwierigkeit stellt die benötigte Bewilligung zum Betrieb des Automaten dar. Systeme, welche Lebensmittel verarbeiten und anschliessend anbieten, benötigen eine spezielle Bewilligung des Kantons. Um diese zu erhalten, müssen einige Tests bestanden werden, wobei man die Ventile eventuell als kritisch betrachten würde.

Aus diesen Gründen kann der aktuelle Prototyp meiner Meinung nach kaum als kommerzielles Produkt vertrieben werden. Das Potenzial des Konzepts beziehungsweise die Nachfrage nach einem Getränkeautomaten ist jedoch durchaus vorhanden. Ein vollautomatischer Cocktail-Roboter wäre für viele Gastronomie-Betriebe eine Kundenattraktion und könnte Lohnausgaben einsparen, wenn er einen Barkeeper zumindest teilweise ersetzt, also beispielsweise die drei beliebtesten Getränke zubereitet.

Für den jetzigen Prototypen besteht aber die folgende Anwendungsmöglichkeit: der Getränkeautomat könnte als Demonstrationsgerät in Restaurants oder Bars ausgestellt werden. Im Vordergrund würde dabei das Anlocken von Kunden stehen und nicht das effiziente Bereitstellen von Cocktails. Um die hohen Anschaffungskosten zu reduzieren, könnte der Automat monatlich vermietet werden.

8.4. Erweiterungen für den Massenmarkt

Um das Projekt zu einem Massenmarkt-tauglichen Produkt weiterzuentwickeln, müssten einige grundlegende Verbesserungen umgesetzt werden, welche ich hier erläutern möchte.

8.4.1. Sortiment erweitern

Die Auswahl ist zum jetzigen Zeitpunkt mit nur sieben Zutaten eher niedrig. Durch zusätzliche Zutaten würde sich die Auswahl stark erhöhen und das Gerät für den kommerziellen Einsatz interessanter machen. Mit Sirups könnte man die Auswahl zusätzlich erweitern. Ausserdem müsste das Verhältnis von gekühltem Raum zu gekühlten Zutaten gesenkt werden, also konkret mehr Zutaten auf engerem Raum untergebracht werden.

Die Elektronik und die Dosiermechanik sollten aus dem gekühlten Bereich ausgelagert werden, wodurch mehr Raum zur Lagerung von Zutaten vorhanden wäre. Dadurch würden die Ausgaben für die Kühlung sinken, wenn man die Kosten pro gekühlte Flüssigkeit betrachtet.

8.4.2. Wartung vereinfachen

Ein geringerer Wartungsaufwand würde die laufenden Kosten, die das Gerät im kommerziellen Betrieb generiert, deutlich senken. Dafür müsste insbesondere das Auswechseln beziehungsweise Nachfüllen von Zutaten vereinfacht werden. So könnten die Zutaten in Behältern gelagert werden, welche durch einen Schlauch befüllt werden könnten. Dadurch würde das aufwendige Demontieren der Behälter entfallen, unter Umständen könnte sich sogar das Öffnen des Kühlschranks erübrigen.

Mittels eines Spülsystems für die Magnetventile könnte die Sauberkeit der Ventile erhöht und damit der Wartungsaufwand gesenkt werden, da allfällige Reinigungen der Ventile wegfielen. Eine Schnittstelle zur Fernwartung könnte der Wartungsaufwand weiter verringert werden, da bei vielen Problemen kein Service vor Ort nötig wäre.

8.4.3. Zutaten vermischen

In der jetzigen Ausführung des Automaten fehlt der eigentliche Mischprozess. Diese Erweiterung würde den Demonstrationswert des Roboters weiter steigern und dem Automaten marketing-technisch gesehen den folgenden Vorteil bieten: der Kunde könnte beim Bestellen in Anlehnung an den bekannten James Bond-Spruch *„geschüttelt, nicht gerührt“* nach der Mischmethode gefragt werden.

Dafür wären zwei Mischvorrichtungen nötig, wobei eine davon das Getränk umrührt und die andere die Zutaten durch Schütteln vermischt. Die Konstruktion eines solchen Rühr- oder Schüttelmechanismus wäre allerdings relativ aufwendig und kostspielig.

Das Durchmischen mit Hilfe eines Rohres mit starker Rückmischung wäre diesbezüglich viel effizienter und günstiger. Dieser Ansatz ist allerdings nicht so spektakulär und würde daher nicht ganz zum Konzept des Projektes passen.

8.4.4. Effizienz steigern

Würde man die Trichterpositionierung durch Schlauchverbindungen ersetzten, würde der Mischvorgang deutlich verkürzt werden, da alle benötigten Zutaten gleichzeitig dosiert werden könnten. Dies würde die Produktivität des Automaten deutlich steigern und ihn dadurch wirtschaftlicher machen. Andererseits würde der Demonstrationswert des Geräts sinken, bezüglich dieser Anpassung muss also zwischen Effizienz und Ästhetik abgewogen werden.

Der Einsatz von grösseren Zutatenspeichern hätte gleich zwei Vorteile: erstens müssten die Behälter seltener aufgefüllt werden, da mehr Flüssigkeit in ihnen gelagert werden könnte. Ausserdem würde die Flüssigkeit mit einer höheren Austrittsgeschwindigkeit ausfliessen. Dadurch würden die Dosiervorgänge kürzer werden, da die Ventile weniger lange geöffnet werden müssen.

8.4.5. Kundenstamm vergrössern

Um den Automaten für noch mehr Leute zugänglich zu machen, wäre eine Smartphone-App zur Fernbedienung denkbar. Damit könnte man zum Beispiel im Voraus Bestellungen aufgeben, welche dann zu einem gewählten Zeitpunkt bereitgestellt würden.

Eine Sprachsteuerung würde zwar die berührungslose Bedienung des GUI ermöglichen, wäre in Bars und Clubs aufgrund des lauten Umfelds aber nicht einsetzbar.

8.4.6. Ästhetik verbessern

Auch bei einem kommerziellen Produkt sollte die Optik stimmig sein und das Gerät den Nutzer faszinieren, was zu höherer Bekanntheit und mehr Kunden führt. Darum sollte auch im finalen Produkt sehr viel Glas und Aluminium verbaut werden.

Die Lichteffekte sollten deutlich komplexer umgesetzt werden: es sollten sicherlich LEDs in verschiedenen Farben verbaut werden. Diese könnten in einer Bar dann im Takt der Musik aufblicken oder sogar die Musik visualisieren. In ruhigeren Umgebungen wäre es spannend, die Beleuchtung dem Verhalten des Kunden anzupassen und im Optimalfall sogar auf seine Stimmung reagieren zu können.

9. Danksagung

9.1. Danksagung an Sponsoren

Ich konnte mich bei meiner Maturaarbeit auf ein sehr hilfsbereites Umfeld verlassen, welches mir bei Fragen und Problemen immer weitergeholfen hat. Doch schlussendlich muss jedes Hardware-Projekt auch finanziert werden. Die Suche nach Sponsoren ist äusserst erfolgreich verlaufen. Vor allem in der Region Schaffhausen ist mir der hohe Stellenwert einer Maturaarbeit sehr positiv aufgefallen.

Ich möchte mich in diesem Abschnitt der Arbeit bei meinen grosszügigen Unterstützern bedanken, ohne die ich mein Projekt niemals hätte umsetzten können. Ich habe mich bei jeder Zusage erneut darüber gefreut, dass es Unternehmen gibt, die selbst einem Kantonsschüler ohne jegliche Garantien grosszügige Beträge zusichern. Neben finanzieller Hilfe habe ich auch oft viel technischen Support und Beratung erhalten.

Danken möchte ich vor allem meinem grössten Sponsor, Iseli + Albrecht AG, die mir einen verglasten Getränkekühlschrank zur Verfügung gestellt und mir gleich zu Beginn der Arbeit eine grosse Hürde aus dem Weg geräumt haben.

Die Kantonsschule Schaffhausen hat mich mit grosszügigen finanziellen Mitteln unterstützt und mir wichtige Räumlichkeiten zur Verfügung gestellt. Für diese Hilfestellungen möchte ich mich herzlich bedanken.

PCP.ch möchte ich für das Sponsoring eines hervorragenden Touchscreens danken. Dank der hohen Qualität und der Robustheit des Bildschirms ist er für meinen Einsatz perfekt geeignet.

Ein grosszügiger Bargeldbeitrag von MTF Schaffhausen AG hat es mir ermöglicht, bei den Mikrocontrollern und Sensoren nicht kürzen zu müssen und meinem Gerät eine maximale Funktionalität zu geben. Vielen Dank.

Auch Distrelec AG hat mit einem sehr hilfreichen Bargeldbeitrag einen grossen Dank verdient. Mit diesen Mitteln konnte ich das effektvolle Linearschienen-Trichtersystem finanzieren.

Die SMC Pneumatik AG hat mir mit starkem Rabatt auf Ventile und gratis Steckverbindungen einen grossen Dienst erwiesen, ich möchte mich auch hierfür bedanken.

Tinkerforge hat mir einen Rabatt auf die Mikrocontroller und Sensoren gewährt und so die Kosten für die Elektronik gesenkt. Ich bedanke mich für dieses Entgegenkommen.

9.2. Danksagung an Mitwirkende

Während des Verfassens meiner Maturarbeit bin ich vielen Problemen verschiedenster Art begegnet. Dank zahlreichen Unterstützern konnten die meisten aus dem Weg geräumt werden. In diesem Abschnitt möchte ich mich bei diesen Helfern herzlich bedanken. Dieses Projekt wäre ohne ihre Hilfe über meinen technischen und menschlichen Möglichkeiten gestanden.

Als erstes möchte ich meinem Betreuer Herrn Raphael Riederer danken, welcher mich während der gesamten Zeit begleitet und mir von Beginn an voll und ganz vertraut hat. Durch dieses Vertrauen war ich sehr frei und konnte sehr kreativ sein. Bei den regelmässigen Besprechungen zeigte er seine Begeisterung immer sehr offen, was mich unglaublich motiviert hat. Vielleicht noch wichtiger waren jedoch das ständige Hinterfragen und das offene Äussern von Bedenken. Ich danke ihm für seine begeisterte, kritische und flexible Betreuung.

Als nächstes möchte ich mich bei Herrn Günter Schmidtner bedanken, welcher mir bei der Planung und Umsetzung der Hardware eine riesige Hilfe war. Als Physikassistent der Kantonsschule Schaffhausen besitzt er ein grosses Know-How bezüglich der Realisierung von mechanischen Systemen. Dieses enorme Wissen stellte er mir voll und ganz zur Verfügung. Er hat bei vielen meiner teils ungenauen Ideen zu den Konstruktionen entscheidende Unterstützung gegeben. Des Weiteren stellte er mir die Kanti-Werkstatt mit professionellen Werkzeugen zur Verfügung und zeigte mir, wie diese zu benutzen sind. Ich möchte mich für seine Unterstützung bei den praktischen Arbeiten vielmals danken!

Ein grosser Dank geht an meinen Betreuer bei „Schweizer Jugend forscht“, Herrn Andreas Reinhard, welcher mich seit Januar 2014 betreut hat. Er hat mir mit seinen kreativen Ideen und spannenden Ansätzen nochmals viel Verbesserungs- und Erweiterungspotenzial aufgezeigt und mich dazu motiviert, diese Möglichkeiten auszunutzen. Ich danke ihm für die riesige Begeisterung für mein Projekt, das grosse Engagement beim Organisatorischen und den festen Glauben, das Projekt stets weiter verbessern zu können.

Nun möchte ich mich bei Herrn Michael Vonarburg von der SMC Pneumatik AG bedanken. Als Product Manager hat er mir bei allgemeinen Fragen zu Ventilen und Fluidsystemen immer weitergeholfen. Vor allem die schnellen Rückmeldungen und ausführlichen Hilfestellungen sind mir sehr positiv aufgefallen. Ich war begeistert, dass ein grosses Unternehmen wie SMC Pneumatik AG sich so viel Zeit für einen Maturanden nimmt. Vielen Dank für die grosse Unterstützung!

Da viele verwendete Teile meines Projekts nur für eine kommerzielle Nutzung vorgesehen sind, habe ich auch eine Partnerfirma für Lieferungen und Rechnungsabwicklungen benötigt. Ich möchte mich hiermit allen voran bei Frau Verena Prager bedanken. Aber auch den Mitarbeitern des Güterhofs Schaffhausen und prager.gastronomie ag bin ich sehr dankbar!

Die ersten Schritte waren bei meiner Arbeit zweifellos die schwierigsten. Einerseits musste ich kreativ sein, andererseits musste ich zielgerichtet arbeiten, um das Vorhaben im Rahmen des Machbaren zu halten. Ich möchte mich hier bei meinem Bruder Moritz bedanken, der mir als Maschinenbaustudent Erfahrungen und sehr wertvolle Tipps weitergeben konnte. Ich möchte mich herzlich für seine Ratschläge, seine Begeisterung und seine mahnenden Worte bedanken, die mich das Ziel nie aus den Augen verlieren liessen.

Damit aus meinem Projekt eine wissenschaftliche Arbeit werden konnte, brauchte sie Struktur und eine saubere Dokumentation. Ich möchte mich darum auch bei meinen Eltern bedanken. Meinem Vater Thomas möchte ich für die vielen Tipps zum wissenschaftlichen Dokumentieren danken. Meiner Mutter Karin danke ich für die vielen Lieferfahrten und das grosse Interesse an meiner Arbeit.

Doch meine Maturaarbeit verlangte mir nicht nur technisches Wissen ab, sondern forderte mich auch menschlich. Vor allem durch meine grosse Begeisterung dachte ich auch oft während meiner Freizeit an mein Projekt. Ich möchte mich hier bei allen bedanken, die mich die Arbeit vergessen liessen und mich zwischendurch motiviert haben, ich danke meinen Freunden für die aufbauenden Worte.

Die Durchführung meiner Maturaarbeit hat viel Zeit und Energie benötigt. Ich möchte mich hier bei meiner Freundin bedanken, die oft auf mich verzichten musste und mich trotzdem voll und ganz unterstützt hat. Mit ihrer Begeisterung und Neugier hat sie mir unglaublich viel Energie und Motivation gegeben. Ich bedanke mich herzlich für ihr Verständnis, ihre Geduld und ihr grosses Interesse an meinem Projekt.

10. Redlichkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit eigenständig verfasst habe. Dort wo Hilfestellungen oder Quellen Eingang in die Arbeit gefunden haben, habe ich dies korrekt und vollständig bezeichnet.

Datum: 29.03.2014 Unterschrift: _____

11. Quellenverzeichnis

11.1. Literaturquellen

Fredrik Lundh, (1999), An Introduction to Tkinter
Gerhard Vetter (2001), Handbuch Dosieren, 2. Auflage
Kapitel 13 von: Paul A. Tipler und Gene Mosca (2009), Physik, 6. Auflage
Ralv Wohlgethan (2007), High Level Programmiersprachen als Brückensprachen zwischen Mensch und Maschine, 1. Auflage
Seite 53 von: Gottfried W. Ehrenstein und Sonja Pongratz (2007), Beständigkeit von Kunststoffen, Band 1
Seite 65 von: Duden Informatik (2001)
Sexl, Raab und Streeruwitz (2002), Einführung in die Physik, Band 1: Mechanik und Wärme

11.2. Internetquellen

Alle nachfolgenden Internet-Quellen wurden am 27. November 2013 auf ihre Aktualität geprüft.

http://content2.smcetech.com/pdf/VDW_1_DE.pdf
http://de.wikipedia.org/wiki/Hardware
http://de.wikipedia.org/wiki/Inverter_(Energietechnik)
http://de.wikipedia.org/wiki/Leuchtr%C3%B6hre
http://de.wikipedia.org/wiki/Open_Source
http://de.wikipedia.org/wiki/Polyethylen
http://de.wikipedia.org/wiki/Programmiersprache
http://de.wikipedia.org/wiki/Software
http://de.wikipedia.org/wiki/Station%C3%A4re_Str%C3%B6mung
http://de.wikipedia.org/wiki/Tinkerforge
http://definitions.uslegal.com/r/robotics/
http://docs.python.org/3.3/library/tkinter.html
http://docs.python.org/3.3/library/tkinter.tix.html
http://docs.python.org/3.3/library/tkinter.ttk.html#tkinter.ttk.Widget
http://en.wikipedia.org/wiki/High-level_programming_language
http://en.wikipedia.org/wiki/Low-level_programming_language
http://en.wikipedia.org/wiki/Python_(programming_language)
http://en.wikipedia.org/wiki/Unified_Modeling_Language
http://en.wikipedia.org/wiki/User_interface

http://partyrobotics.com/
http://pcsupport.about.com/od/termshm/g/hardware.htm
http://vsa.labeaux.ch/docs_public/03%20MT%20pH-Wert%20DP.pdf
http://wirtschaftslexikon.gabler.de/Archiv/1598/pin-v9.html
http://wirtschaftslexikon.gabler.de/Definition/programmiersprache.html
http://www.chip.de/news/CHIP-AWARDS-2012-Das-sind-die-Preistraeger_54928401.html
http://www.deltron.ch/pdf/produkte/peltier/peltier-element_kurz_erklaert_d.pdf
http://www.freudenberg-process-seals.de/ecomaXL/get_blob.php?name=FKM_de.pdf
http://www.golem.de/news/tinkerforge-im-test-elektronik-zum-stapeln-1205-91624-4.html
http://www.golem.de/specials/tinkerforge/
http://www.highscore.de/uml/
http://www.informatik.uni-leipzig.de/lehre/Heyer9900/kap18/sld001.htm
http://www.itwissen.info/definition/lexikon/Benutzeroberflaeche-UI-user-interface.html
http://www.itwissen.info/definition/lexikon/Mikrocontroller-MCU-micro-controller-unit-microC.html
http://www.kickstarter.com/projects/partyrobotics/bartendro-a-cocktail-dispensing-robot
http://www.melmacc.at/
http://www.raspberrypi.org/faqs
http://www.systemdesign.ch/index.php/Ausflussgesetz_von_Torricelli
http://www.systemdesign.ch/index.php?title=Gesetz_von_Bernoulli
http://www.wasserinbayern.de/uv-anlagen/wissenswertes/
https://www.lucidchart.com/pages/uml/what-is-uml

11.3. Bildquellen

Bild 1: http://www.wired.com/design/wp-content/uploads/2013/03/bartendro-8-unit-wired-design.jpg
Bild 2: http://lh5.ggpht.com/-oSpyRxSjQVc/SvmTaHC7SsI/AAAAAAAAA1c/G9qtze5Ptk/DSC02778.jpg?imgmax=800
Bild 3: http://www.highscore.de/uml/img/usecase_assoziationen.gif
Bild 4: http://www.highscore.de/uml/img/aktivitaet_startendknoten.gif
Bild 5: http://www.highscore.de/uml/img/aktivitaet_aktionobjektknoten.gif
Bild 6: http://www.highscore.de/uml/img/aktivitaet_verzweigunggabelung.gif
Bild 7: http://www.python-kurs.eu/klassen.php
Bild 8: http://www.highscore.de/uml/img/klassen_assoziationen.gif
Bild 10: http://download.tinkerforge.com/press/media/brick_stack_back.jpg
Bild 11: http://www.tinkerforge.com/en/doc/_images/Bricks/
Bild 12: http://www.tinkerforge.com/de/doc/_images/Bricklets/
Bild 13: http://arduino.cc/de/uploads/Main/ArduinoUno_R3_Front_450px.jpg
Bild 14: http://www.adafruit.com/adablog/wp-content/uploads/2012/03/window-2-39.jpg
Skizze 1: http://web.physik.rwth-aachen.de/~fluegge/Vorlesung/PhysIpub/Exscript/9Kapitel/Image119.gif
Skizze 2: http://aplusphysics.com/courses/honors/fluids/images/water_jug_diagram.png
Skizze 14: http://content2.smcetech.com/pdf/VDW_1_DE.pdf

12. Anhang

12.1. Kostenübersicht

12.1.1. Gesamtwert

Anbieter	Artikel/Leistung	Preis	Datum
Brack AG	Arduino Uno, FingerPrint-Sensor	Fr. 101,00	26.02.2014
Conrad	Servo	Fr. 9,95	19.11.2012
Conrad	Servo	Fr. 47,85	27.03.2013
Conrad	Servo, Netzteil	Fr. 55,80	10.01.2013
Conrad	Dichtungen	Fr. 37,60	27.07.2013
Conrad	Dichtungen, Schlauch	Fr. 48,35	29.06.2013
Conrad	Netzteil, Luftentfeuchter	Fr. 87,20	27.11.2013
Conrad	Stepper Motor, Zahnriemen, LEDs	Fr. 114,20	09.08.2013
Intergastro	3 Dosierer, 2 Wand-halterungen	Fr. 106,95	07.01.2013
Intergastro	Wandhalterungen, Aufsätze (Preis geschätzt)	Fr. 155,45	25.08.2013
Iseli und Albrecht	Kühlschrank gesponsert	Fr. 1 650,00	01.02.2013
PCP.ch	Bildschirm-Halterung	Fr. 117,00	12.10.2013
PCP.ch	Monitor gesponsert	Fr. 332,00	01.04.2013
RS-Online	VDW21-5G-2-01F-Q	Fr. 26,87	25.06.2013
RS-Online	Befestigungsplatte für VDW21	Fr. 2,50	26.07.2013
SMC	Acht Magnetventile	Fr. 330,00	21.08.2013
SMC	Rabatt auf Sortiment, gratis Steckverbindungen	Fr. 360,00	21.08.2013
Tinkerforge	Servo Brick inkl. Mounting Kit	Fr. 85,40	13.11.2012
Tinkerforge	Master Brick, Step-Down-Brick, DC-Adapter	Fr. 59,25	03.01.2013
Tinkerforge	Sensoren	Fr. 58,80	13.02.2013
Tinkerforge	Dual Relay Bricklets, Kabel, Befestigungen, Temperatur-Sensor	Fr. 87,35	01.08.2013
Tinkerforge	Stepper Brick, Kabel, Befestigung	Fr. 71,20	09.08.2013
Tinkerforge	Sensor, Kabel, Gehäuse	Fr. 80,80	23.09.2013
Tinkerforge	Rabatt auf Sortiment	Fr. 45,00	02.01.2013
	Gesamtwert:	**Fr. 4 070,52**	
	Anteil an gesponserten Kosten	**96,23%**	

12.1.2. Sponsoring

Firma	Leistung	Wert	Datum
Iseli und Albrecht	Kühlschrank gesponort	Fr. 1 650,00	01.02.2013
SMC	Rabatt auf Sortiment, gratis Steckverbindungen	Fr. 360,00	21.08.2013
PCP.ch	Monitor gesponsert	Fr. 332,00	01.04.2013
Kantonsschule SH	Ventilkosten übernommen; Geldbetrag	Fr. 930,00	27.03.2014
Distrelec	Geldbetrag	Fr. 300,00	19.06.2013
MTF	Geldbetrag	Fr. 300,00	06.08.2013
Tinkerforge	Rabatt auf Sortiment	Fr. 45,00	02.01.2013
	Total gesponsert:	**Fr. 3 917,00**	

12.1.3. Ausgaben

Anbieter	Artikel	Preis	Datum
Brack AG	Arduino Uno, FingerPrint-Sensor	Fr. 101,00	26.02.2014
Conrad	Servo	Fr. 9,95	19.11.2012
Conrad	Servo	Fr. 47,85	27.03.2013
Conrad	Servo, Netzteil	Fr. 55,80	10.01.2013
Conrad	Dichtungen	Fr. 37,60	27.07.2013
Conrad	Dichtungen, Schlauch	Fr. 48,35	29.06.2013
Conrad	Netzteil, Luftentfeuchter	Fr. 87,20	27.11.2013
Conrad	Stepper Motor, Zahnriemen, LED's	Fr. 114,20	09.08.2013
Intergastro	3 Dosierer, 2 Wand-halterungen	Fr. 106,95	07.01.2013
Intergastro	Wandhalterungen, Aufsätze	Fr. 155,45	25.08.2013
PCP.ch	Bildschirm-Halterung	Fr. 117,00	12.10.2013
RS-Online	VDW21-5G-2-01F-Q	Fr. 26,87	25.06.2013
RS-Online	Befestigungsplatte für VDW21	Fr. 2,50	26.07.2013
SMC	Acht Magnetventile	Fr. 330,00	21.08.2013
Tinkerforge	Servo Brick inkl. Mounting Kit	Fr. 85,40	13.11.2012
Tinkerforge	Master Brick, Step-Down-Brick, DC-Adapter	Fr. 59,25	03.01.2013
Tinkerforge	Sensoren	Fr. 58,80	13.02.2013
Tinkerforge	Dual Relay Bricklets, Kabel, Befestigungen, Temperatur-Sensor	Fr. 87,35	01.08.2013
Tinkerforge	Stepper Brick, Kabel, Befestigung	Fr. 71,20	09.08.2013
Tinkerforge	Sensor, Kabel, Gehäuse	Fr. 80,80	23.09.2013
	Totale Ausgaben:	**Fr. 1 683,52**	
	Davon ungedeckt:	**Fr. 153,52**	

12.2. Quellcode und Video auf CD

Beigelegt zu dieser Arbeit befindet sich eine CD, auf welcher der Quellcode der Software und ein Video des Automaten gespeichert sind.

Der Code kann leider nicht auf anderen Rechnern ausgeführt werden, da die benötigten Mikrocontroller nicht angeschlossen sind und es darum zu einem Timeout-Error kommt.