CARL VON OSSIETZKY universität OLDENBURG

DEPARTMENT FÜR INFORMATIK
SYSTEMSOFTWARE UND VERTEILTE SYSTEME

Entwurf und Implementierung einer Experimentieranlagensteuerung im Rahmen vernetzter Leitstände für chemische Experimente: AniTA

Bachelorarbeit

10. April 2023

Malte Utech
Nadorster Straße 76
26123 Oldenburg

Erstprüfer Prof. Dr.-Ing. Oliver Theel
Zweitprüfer M. Sc. Marvin Banse

Erklärung zur Urheberschaft

Hiermit versichere ich an Eides statt, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.

Oldenburg, den 10. April 2023

Malte Utech

In dieser Bachelorarbeit wurde die Firmware eines Steuergerätes einer Experimentieranlage aus dem Bereich der Technischen Chemie entworfen und implementiert. Hierbei wurde die Anlage darauf vorbereitet, über das Internet ferngesteuert zu werden und die sichere Durchführung von Experimenten an ihr durch Zustandsautomaten gewährleistet.

In this bachelor's thesis, the firmware of a control unit of an experimenting facility in the field of technical chemistry was designed and implemented. The system was prepared for remote control via the Internet and the safe execution of experiments was ensured by the use of state machines.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Glossar

adiabatisch Beschreibt, dass aufgrund einer sehr geringen Wärmeleitung kein Wärmeaustausch mit der Umgebung stattfindet.

AniTA Adiabatisch-instationärer Tankreaktor. Experimentieranlage der Abteilung Technische Chemie 2, die Wasser und Essigsäureanhydrid vermischt und den entstehenden Temperaturverlauf misst.

EFaViS Experiment Facility Virtualization Server. Softwareartefakt des Projektes, welches die durchzuführenden Experimente virtualisiert und mit den Experimentieranlagen verbunden ist.

GECP General Experiment Control Protocol. Eigens im Projekt entwickeltes Protokoll, dass die Kommunikation der Komponenten regelt.

GPIO Pin General Purpose Input/Ouput Pin. Digital steuerbarer Kontaktstift, der zur Eingabe und Ausgabe genutzt werden kann.

IngA Internet-gesteuerte Adsorptionsanlage. Experimentieranlage der Abteilung Technische Chemie 2, mit der mehrere unterschiedliche Versuche zur Adsorption von verschiedenen Stoffen möglich sind.

MiA Mikroreaktionsanlage zur Dehydratisierung von Isopropano. Experimentieranlage der Abteilung Technische Chemie 2.

VerA Verweilzeitanlage. Experimentieranlage der Abteilung Technische Chemie 2, die die Verweilzeit von Stoffen unter verschiedenen Bedingungen misst.

Volumenstrom Physikalische Größe, die den Durchfluss von Volumen pro Zeiteinheit beschreibt.

1. Einleitung

Die Abteilung Technische Chemie 2 der Universität Oldenburg möchte ferngesteuerte Experimentieranlagen kreieren, um Online-Experimente anzubieten, welche Chemiestudenten die Möglichkeit geben, von Zuhause aus Versuche durchzuführen. Bis jetzt gibt es zwei Anlagen mit dieser Funktionalität an der Universität, die Internet-gesteuerte Adsorptionsanlage (IngA) und die Mikroreaktionsanlage zur Dehydratisierung von Isopropanol (MiA), welche Anfang der 2000er Jahre von der Abteilung Technische Chemie 2 entworfen wurden und von einer internationalen Studentenschaft benutzt werden. Dies war vor allem im Kontext der Coronapandemie nützlich, um die Umstellung zur Onlinelehre zu realisieren.

1.1. Aufgabenstellung

Diese existierenden Anlagen, ihre Bedienoberfläche und die Nutzerverwaltung sind jedoch nicht mehr zeitgemäß. Sie sind von einer Reihe an Problemen geplagt: die Nutzer müssen manuell von Administratoren in Datenbanken eingepflegt werden, die Benutzeroberfläche setzt auf veraltete, nicht mehr unterstützte Webtechnologien und die Systeme der Anlagen sind untereinander nicht kompatibel, so dass zur Nutzung jeweils unterschiedliche Prozeduren durchlaufen werden müssen.
Mit der Intention, diese Probleme in Zukunft zu vermeiden und das Angebot der Online-Lehre zu vergrößern, ist die Abteilung Technische Chemie 2 an die Abteilungen Systemsoftware & verteilte Systeme und Softwaretechnik herangetreten, um ein neues System zu entwickeln, welches durch ein Internet of Things die Nutzung von Anlagen international verteilter Universitäten ermöglicht, leichte Erweiterbarkeit von Experimenten sicherstellt und durch eine gemeinsame Nutzerverwaltung und Oberfläche die Kooperation der teilnehmenden Universitäten vereinfacht.
Im Rahmen dessen werden zwei neue Anlagen gebaut, der Adiabatisch-instationäre Tankreaktor (AniTA) und die Verweilzeitanlage (VerA). Aufgabe dieser Bachelorarbeit ist es, die Steuerung der Experimentieranlage AniTA zu entwerfen und zu implementieren.

1.2. Zielsetzung

Um das genannte System zu implementieren, welches die bestehenden Probleme vermeidet, werden die Aufgaben im Projekt folgendermaßen aufgeteilt: Die Abteilung Softwaretechnik entwirft eine Gesamtarchitektur und erstellt die Software, die zur Benutzerverwaltung und als Bedienoberfläche genutzt wird. Die Abteilung Technische Chemie installiert die beiden neuen Anlagen, AniTA und VerA, an denen die Online-Experimente durchgeführt werden sollen und die Abteilung Systemsoftware & verteilte Systeme programmiert die Hardware der Anlagen und liefert Schnittstellen, um diese anzusteuern.
Um die in dieser Arbeit behandelte Experimentieranlage AniTA in dieses Gesamtprojekt zu integrieren, müssen einige Ziele erreicht werden. Zum einen muss die in der Anlage verbaute Steuereinheit die technischen Komponenten der Anlage ansteuern können. Ziel ist hier die Durchführung von Experimenten an der Anlage ohne menschliches Eingreifen. Weiterhin muss die Anlage und ihre Steuereinheit in die anderen Komponenten des Gesamtprojekts eingebunden werden, um die Vision des Projektes zu erfüllen und die Anlage über das Internet zu steuern. Zusätzlich muss erreicht werden, dass die Experimentieranlage weder sich selbst, noch ihrer Umgebung schaden könnte. Ihr Handlungsspielraum muss also insofern eingeschränkt werden, dass es weder für die letztendlich

über das Internet experimentierenden Studenten, noch bei Bedienung vor Ort möglich ist, unsichere Aktionen an der Anlage auszuführen.

1.3. Aufbau

Im folgenden Kapitel 2 wird die Projektumgebung dieser Arbeit genauer erklärt. Ebenfalls wird die Experimentieranlage AniTA im Detail gezeigt und die erhobenen Anforderungen an die zu erbringende Arbeit gelistet.
Kapitel 3 behandelt die Rahmenarchitektur des Projektes und es werden Grundlagen zur Kommunikation zwischen den Bestandteilen des Projektes erklärt. Zusätzlich wird die entworfene Architektur der Firmware des Steuergerätes vorgestellt.
In Kapitel 4 wird die Umsetzung der zuvor erarbeiteten Komponenten der Firmware beschrieben. Ebenfalls wird die Kommunikation der Anlage mit ihrer Umgebung erklärt und die bei der Realisierung benutzten Werkzeuge gezeigt.
Kapitel 5 gibt eine kurze Anleitung zur Inbetriebnahme der Firmware und der Schritte, die davor und danach notwendig sind.
In Kapitel 6 wird die Umsetzung evaluiert und in Kapitel 7 wird die Arbeit anschließend zusammengefasst und ein Fazit gegeben.

2. Projektbeschreibung

Grundessenz des größeren Projektes, in das sich diese Arbeit einbettet, ist es, die Möglichkeit einer Fernsteuerung von chemischen Experimentieranlagen über das Internet zu schaffen. Damit sollen Studenten chemische Versuche lokalitätsunabhängig durchführen und, im Gegensatz zu heutzutage üblichen, simulierten Experimenten, tatsächliche Anlagen steuern und reale Messergebnisse aufzeichnen können.
In der Vision der Abteilung Technische Chemie 2 werden innerhalb des Projektes ebenfalls die Rahmenbedingungen geschaffen, um andere Universitäten in dieses System zu integrieren, so dass jede teilnehmende Universität ihre fernsteuerbaren Experimente beisteuern kann und die Studenten aller Universitäten an einer Fülle von unterschiedlichen Experimenten profitieren.

2.1. Projektziele

Um diese Vision zu realisieren, können folgende Ziele des Projektes definiert werden:

Benutzerfreundlichkeit

Um nicht nur Frust bei den Studenten, sondern auch hohen Zeitaufwand bei den Administratoren zu vermeiden, soll das System möglichst benutzerfreundlich sein. Hierzu gehört, dass eine Benutzerschnittstelle intuitiv zu bedienen ist und keine komplizierten Installationsprozesse erfordert. So kann im Idealfall Zeit gespart werden, welche sonst für aufwändige Supportprozesse oder dem Schreiben von Bedienungsanleitungen in Anspruch genommen werden würde. Dieser Punkt verstärkt sich noch einmal durch die gewünschte Teilnahme anderer teils internationaler Universitäten.

Expertisegerechtigkeit

Dieses Ziel drückt den Wunsch aus, die Durchführung der Experimente dem Kenntnisstand der Studenten anpassen zu können. So sollen erfahrenere Studenten die volle Kontrolle über die Anlagen erhalten können, jedoch sollen die Experimente auch vereinfacht werden können, indem beispielsweise bestimmte Aktionen nicht zugelassen oder mehrere Aktionen an der Benutzerschnittstelle zu einer zusammengefasst werden.

Nutzerverwaltung

Um den Verwaltungsaufwand für die Universität Oldenburg gering zu halten, soll die Menge aller Nutzer des Systems nicht zentral in einer Datenbank der Universität Oldenburg gespeichert und von dieser gepflegt werden. Stattdessen soll jede teilnehmende Universität ihre Nutzer eigenständig verwalten.

Zugriffskontrolle

Studenten teilnehmender Universitäten sollen nicht jederzeit Zugriff auf jede Experimentieranlage erhalten, sondern es muss eine Zugriffskontrolle stattfinden, durch die Universitäten Zeit an bestimmten Anlagen für ihre Studenten buchen können und nur freigeschaltete Nutzer die Experimentieranlagen steuern können.

Leichte Erweiterbarkeit

Die Teilnahme anderer Universitäten soll möglichst einfach ermöglicht werden. Dafür ist es notwendig, Schnittstellen zu schaffen und zu dokumentieren und Anleitungen zum Implementieren der benötigten Komponenten zu erstellen. Wo möglich sollen Komponenten des Systems wiederverwendbar gestaltet sein, damit der Aufwand für interessierte Universitäten möglichst gering ist.

Sicherheit

Die sichere Steuerung der Anlagen ist eines der wichtigsten Ziele. Es muss sichergestellt werden, dass kein Nutzer der Fernsteuerung, bewusst oder unbewusst, einen Zustand an der Experimentieranlage herbeiführen kann, der ihr oder ihrer Umgebung schadet.

2.2. Projektstruktur

Zur Durchführung des Projektes treffen sich die Beteiligten aller Abteilungen alle zwei Wochen zur Besprechung. Dort wird der Fortschritt und das weitere Vorgehen diskutiert. Die Bearbeitung der einzelnen Aufgaben findet größtenteils im Rahmen vergebener Abschlussarbeiten statt. Zur Klärung größerer, interdisziplinärer Fragen werden auch kleine Arbeitsgruppen gegründet, die ihre Ergebnisse während der allgemeinen Treffen vorstellen. Für den Entwurf einer initialen Projektarchitektur ist die Abteilung Software Engineering verantwortlich. Von ihr lassen sich die vergebenen Abschlussarbeiten herleiten. Im Rahmen dieser behandelt diese Abschlussarbeit die Programmierung der Firmware der Experimentieranlage AniTA und ihre Einbettung in das Gesamtsystem. Die Rahmenarchitektur wird genauer in Abschnitt 3.1 vorgestellt.

2.3. AniTA

Die Abteilung Technische Chemie 2 hat im Rahmen des Projektes mehrere Anlagen hergestellt, mit denen chemische Experimente durchführbar sind. Darunter auch der Adiabatisch-instationäre Tankreaktor (AniTA), welcher für diese Arbeit relevant ist. Abbildung 2.1 zeigt den schematischen Versuchsaufbau dieser Anlage, wie er von der Technischen Chemie geplant wurde.

2.3.1. Experiment

Das Experiment, welches mit der Anlage durchführbar ist, ist ein grundlegender Versuch des Chemiestudiums, bei welchem Wasser mit Essigsäureanhydrid vermischt wird, was zu einer exothermen Reaktion führt, deren Produkt Essigsäure ist. Die dabei freigesetzte Energie wird über einen zeitlichen Verlauf gemessen.
Dies wird dadurch ermöglicht, dass der Glasreaktor, in dem die Reaktion stattfindet adiabatisch ist, was bedeutet, dass er kaum Wärme mit seiner Umgebung austauscht. Somit bleibt der bei der Reaktion entstehende Temperaturanstieg erhalten und kann gemessen werden, um Rückschlüsse auf die freigesetzte Energie zu ermöglichen.

2.3.2. Anlage

Die Anlage wurde von der Abteilung Technische Chemie 2 auf einem mobilen Gestell zusammengesetzt, so dass sie bei Bedarf verschoben werden kann.
Im rechten Teil der Anlage ist der eigentliche Versuchsaufbau untergebracht, welcher in Abbildung 2.2 zu sehen ist. Dieser entspricht dem in Abbildung 2.1 gezeigten, schematischen Versuchsaufbau.

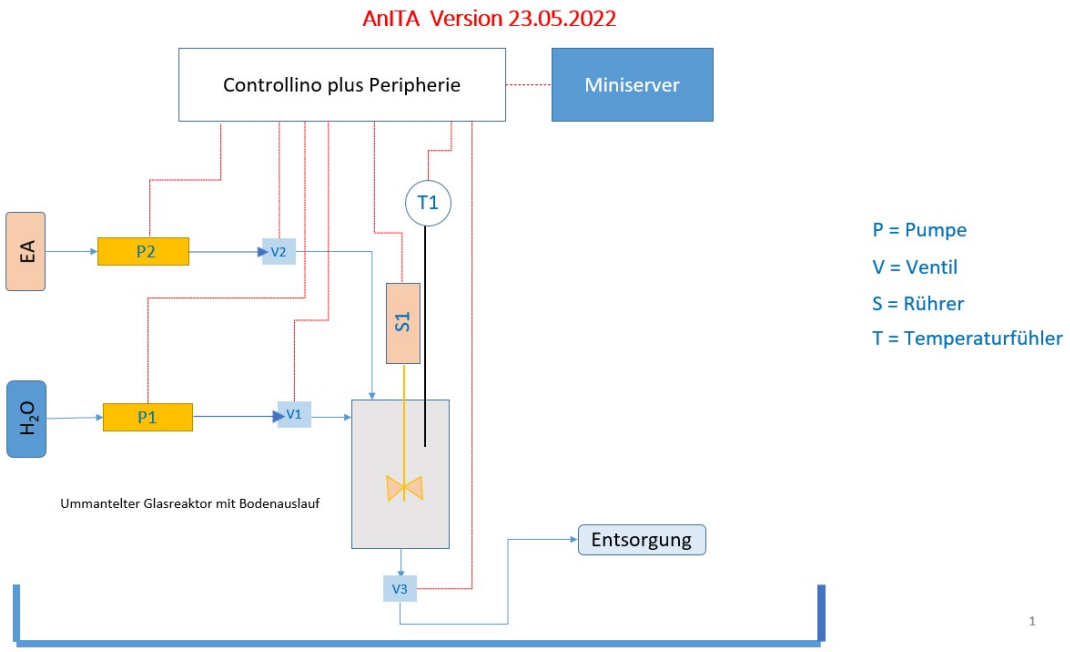

Abbildung 2.1.: Schematischer Versuchsaufbau AniTA

Die für die ferngesteuerte Durchführung eines Experiments relevanten Komponenten des Versuchsaufbaus können unterteilt werden in aktive, steuerbare Komponenten und passive, berechenbare Komponenten.

Aktive Komponenten

Aktive Komponenten sind jene, die von dem Steuergerät angesteuert und kontrolliert werden können, AniTA besteht aus Folgenden:

1. Schlauchpumpen (Verderflex OEM M025 DC)
2. 2/2 Wege Flüssigkeitsventile (Bürkert 2/2 Wege Ventil Typ 6013)
3. Rührer (Heidolph Hei-TORQUE Ultimate 400)
4. Thermoelement (Typ K[1])
5. Licht

Passive Komponenten

Diese Komponenten sind nicht direkt von dem Steuergerät ansteuer- oder messbar, ihr Zustand ist für die Durchführung des Experiments jedoch trotzdem wichtig und muss daher berechnet werden. Wie dies realisiert wird, ist in Abschnitt 4.5 beschrieben.

6. Glasreaktor
7. Schläuche
8. Essigsäureanhydridbehälter

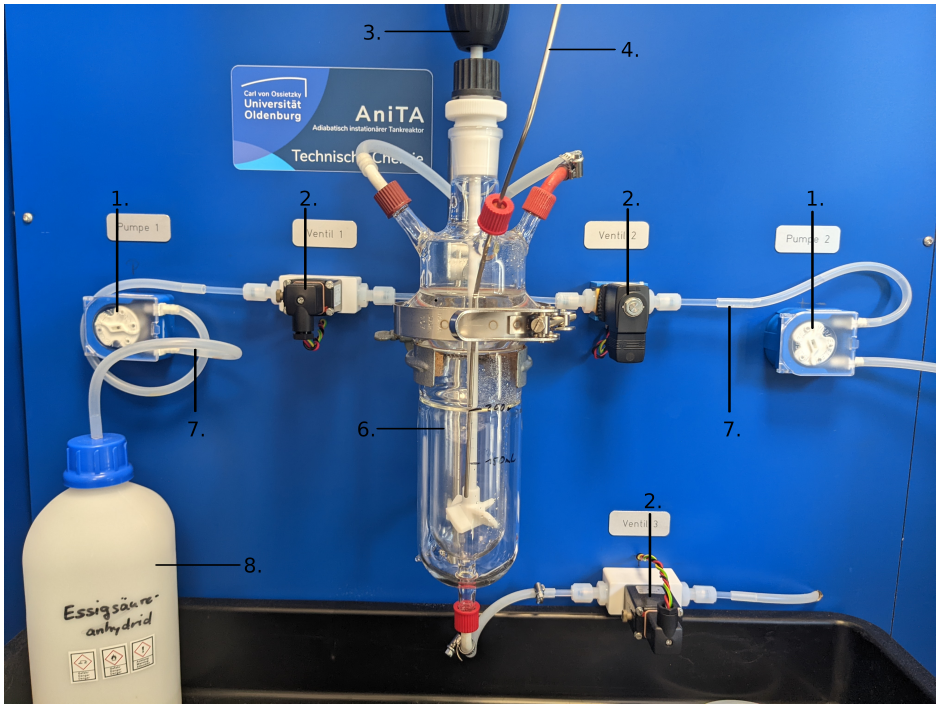

Abbildung 2.2.: Versuchsaufbau AniTA

Da das Ziel ist, Experimente ohne menschliches Eingreifen durchführen zu können, besitzt die Anlage einen großen Wasserbehälter, welcher sich mittels eines Schwimmersystems automatisch am Wasseranschluss auffüllt. Das Essigsäureanhydrid befindet sich in dem erwähnten Essigsäureanhydridbehälter, welcher zwei Liter groß ist und gegebenenfalls händisch aufgefüllt werden muss.
Von den Behältern aus werden die Flüssigkeiten über die elektrisch ansteuerbaren Schlauchpumpen durch die Schläuche in den adiabatischen Glasreaktor gepumpt. Um ungewolltes Nachlaufen nach einem Pumpvorgang zu verhindern, werden die Schläuche in der Nähe des Glasreaktors durch die elektrisch ansteuerbaren Ventile verschlossen.
Im Glasreaktor findet nun die Reaktion statt, welche durch den angebrachten, steuerbaren Rührer begünstigt werden kann. Die Temperatur innerhalb des Glasreaktors kann über das innenliegende Thermoelement gemessen werden.
Nachdem das Experiment durchgeführt wurde, wird die entstandene, unbedenkliche Essigsäure über das Ventil am Boden des Glasreaktors durch einen weiteren Schlauch in den Abfluss geleitet.
Ebenfalls ist die Anlage mit einer Webcam und steuerbarer Beleuchtung ausgestattet, durch welche der Versuch auch per Videostream über das Internet beobachtet werden kann.

Steuernde Komponenten

Im linken Teil der Anlage sind die für die Steuerung der Anlage notwendigen Teile untergebracht. Hinter der Verkleidung befinden sich folgende Komponenten, welche teilweise in Abbildung 2.3 zu sehen sind:

1. Steuergerät (Controllino MEGA)
2. Netzteil (WDR-120-24)
3. Sicherungen
4. RS-232 Adapter

5. Thermoelementleser (Expert DAQ EX-9018BLM)
6. Netzwerkswitch
7. General Purpose Computer
8. Touchscreen

Das Steuergerät ist jenes, welches im Rahmen dieser Arbeit programmiert werden muss. Der Adapter sorgt dafür, dass die RS-232 Schnittstelle des Rührers mit dem Steuergerät verbunden werden kann. Der Thermoelementleser ist über RS-485 mit dem Steuergerät verbunden und misst mit Hilfe des Thermoelements die Temperatur im Reaktor. Auf dem General Purpose Computer können die in Abschnitt 2.4 genannten, anderen Softwareartefakte, welche im Rahmen des Gesamtprojekts entstehen, laufen. Der Touchscreen wurde von der Abteilung Technische Chemie 2 installiert, um direkten Zugriff auf dem General Purpose Computer zu ermöglichen.

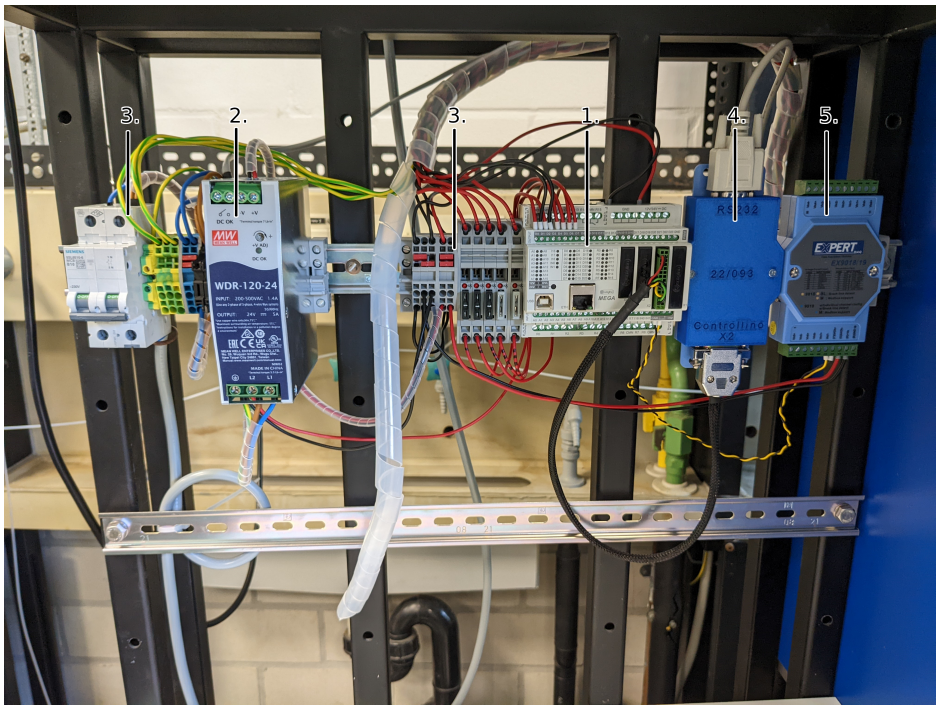

Abbildung 2.3.: Steuernde Komponenten AniTA

2.4. Systemumgebung

Das System, welches im Rahmen des Projektes geschaffen wird, wird aus mehreren Komponenten bestehen, welche jeweils ihre wichtige Rolle beim Erreichen der in Abschnitt 2.1 genannten Ziele spielen.
Grundlegender Bestandteil sind die Experimentieranlagen AniTA und VerA, welche von der Abteilung Technische Chemie 2 hergestellt wurden und an denen die Experimente letztendlich durchgeführt werden. Zum Erreichen des Ziels der Expertisegerechtigkeit wird eine Art “Virtualisierung” der Experimente gebraucht. Diese Software soll die Versuche digitalisieren und es ermöglichen, sie zu vereinfachen. Die Nutzerverwaltung soll ebenfalls ein eigenes Softwaremodul werden, welches jede teilnehmende Universität aufsetzen kann. Letztlich wird für die Benutzerfreundlichkeit ein intuitives GUI benötigt.

Die Experimentieranlage muss in diese Umgebung aus projektrelevanter Software eingebettet werden und ihre jeweiligen Funktionsweisen berücksichtigen, um die Ziele des Gesamtprojektes zu erreichen.

2.5. Einschränkungen

Da bei diesem Projekt ein großer Arbeitsaufwand zu erwarten ist, welcher größtenteils auf Abschlussarbeiter mehrerer Abteilungen der Universität Oldenburg verteilt ist, ist mit operativen Herausforderungen zu rechnen.
So wird es wahrscheinlich der Fall sein, dass die Experimentieranlagen vor den anderen Komponenten des Gesamtsystems, wie den Teilen, die die Virtualisierung übernehmen, oder der Benutzeroberfläche, fertiggestellt werden.
Von daher entfällt die Möglichkeit, im Rahmen dieser Abschlussarbeit die Experimentieranlage und ihre Funktion als Teil des Gesamtsystems zu testen. Somit ist es noch wichtiger, die Schnittstelle zum Rest des Systems genau zu definieren und zu dokumentieren.
Um die Anlage dennoch testen zu können wird eine provisorische Nutzerschnittstelle benötigt, welche mit AniTA kommunizieren und diese testen kann, bis die anderen Teile des Projektes fertiggestellt wurden.

2.6. Anforderungen

Durch Gespräche mit den Verantwortlichen der Abteilungen Technische Chemie 2, Software Engineering und Systemsoftware und verteilte Systeme wurden folgende Anforderungen herausgearbeitet.

2.6.1. Funktionen

F1 Das vorgesehene Experiment muss durchführbar sein.

- F1.1 Die Steuereinheit soll die Kontrolle über alle für die Durchführung des Experiments notwendigen Funktionen der Komponenten haben.
- F1.2 Die Steuereinheit soll die für das Experiment wichtigen Messdaten auslesen können:
  - Die Temperatur im adiabatischen Glasreaktor
  - Das Volumen an Wasser im adiabatischen Glasreaktor
  - Das Volumen an Essigsäureanhydrid im adiabatischen Glasreaktor

F2 Die Anlage soll sich in das größere Rahmenprojekt einfügen können:

- F2.1 Die Kommunikation soll über das WebSocket Netzwerkprotokoll erfolgen.
- F2.2 Die Steuerung soll über ein für das Projekt standardisiertes Protokoll erfolgen.
- F2.3 Die Anlage muss ihre wichtigen Messwerte über das Protokoll mitteilen können.

F3 Die vorgesehene Spülprozedur muss durchführbar sein.

F4 Die Betreiber sollen den Füllstand des Essigsäureanhydrid Vorratsbehälters erfahren können.

F5 Ein Experiment darf nur durchführbar sein, wenn sich dafür genug Essigsäureanhydrid im Vorratsbehälter befindet.

F6 Vor einem Experiment müssen die Schläuche befüllt und der Glasreaktor geleert werden können.

F7 Nach einem Experiment müssen die Schläuche und der Glasreaktor geleert werden können.

F8 Die durch die Begebenheiten der Anlage gegebenen, in der Firmware verwendeten Parameter sollen konfigurierbar sein.

2.6.2. Sicherheit

S1 Die aktiven Komponenten der Anlage dürfen keine Kombination an Stellungen einnehmen, die ihr oder ihrer Umgebung schaden würden.

- S1.1 Bevor eine Pumpe angeschaltet wird, muss das ihr vorgeschaltete Ventil geöffnet werden.
- S1.2 Die Pumpen dürfen nicht gleichzeitig den Glasreaktor befüllen.
- S1.3 Die Reagenzien dürfen während eines Experiments nicht direkt in den Abfluss gepumpt werden.
- S1.4 Der Rührer darf eine zu bestimmende maximale Drehzahl nicht überschreiten.

S2 Ein zu bestimmender maximaler Füllstand des Glasreaktors darf nicht überschritten werden.

S3 Das Verhältnis von Wasser zu Essigsäureanhydrid im Glasreaktor darf 2:1 nicht überschreiten.

S4 Eine zu bestimmende Mindestmenge an Wasser muss im Glasreaktor sein, bevor Essigsäureanhydrid hinzugefügt wird.

S5 Beim Start der Anlage muss diese sicherstellen, dass Schläuche und Glasreaktor geleert sind.

S6 Wenn die Anlage länger unbenutzt ist, müssen die Ventile geschlossen sein.

2.6.3. Projekt

P1 Die zur Steuerung der Komponenten erstellte Software soll für andere Anlagen wiederverwendbar sein.

P2 Die zur Einbindung in das Rahmenprojekt erstellte Software soll für andere Anlagen wiederverwendbar sein.

3. Architektur

In diesem Kapitel wird die Rahmenarchitektur des Gesamtprojektes gezeigt, die Grundlagen für die Kommunikation der Komponenten des Projektes erklärt und die Architektur, auf deren Basis die Realisierung der Firmware der Experimentieranlage AniTA stattgefunden hat, erläutert.

3.1. Rahmenarchitektur

Zum Erreichen der in Abschnitt 2.1 erklärten Ziele des Projektes und unter Berücksichtigung der in Abschnitt 2.4 beschriebenen Systemumgebung wurde von der Abteilung Software Engineering die in Abbildung 3.1 gezeigte Architektur entworfen[1], welche in den folgenden Abschnitten genauer erklärt wird.

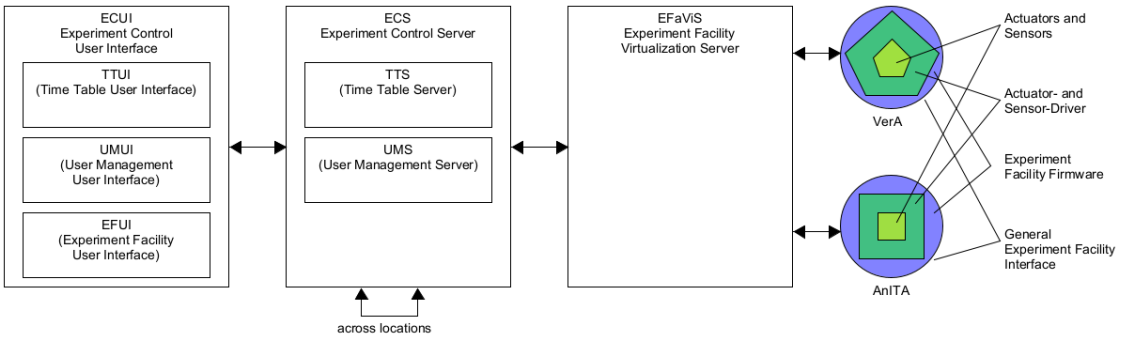

Abbildung 3.1.: Architektur des Gesamtprojektes

3.1.1. Experimentieranlagen

Der Bereich der Experimentieranlagen, in Abbildung 3.1 ganz rechts, beinhaltet die physischen Anlagen, welche von der Abteilung Technische Chemie 2 hergestellt werden, und deren Programmierung, welche von der Abteilung Systemsoftware und verteilte Systeme vollzogen wird. Nach aktuellem Stand des Projektes beinhaltet dies die zwei Anlagen VerA und AniTA.
Ziel der in diesem Abschnitt erforderlichen Arbeit ist es, die Anlagen digital steuerbar zu machen und diese Funktionalität anhand einer dokumentierten Schnittstelle zur Verfügung zu stellen.
Da jede Anlage unterschiedliche Experimente durchführen kann und somit über eine individuelle Konstellation an steuerbaren Komponenten verfügt, müssen jeweils eigene Treiber und eine individuelle Firmware geschrieben werden. Zur Kommunikation mit den anderen Teilen des Projektes braucht es jedoch eine gemeinsame Schnittstelle, die auch von zukünftigen Anlagen verwendet werden kann.

3.1.2. Virtualisierung

Der in Abbildung 3.1 dargestellte dritte Block von links behandelt den Experiment Facility Virtualization Server (EFaViS), welcher für die Virtualisierung der durchführbaren Experimente verantwort-

[1] [2], Seite 9

lich ist. Dies ist notwendig, da mit einer Experimentieranlage mehrere Experimente durchführbar sein können, wie im Falle der VerA, oder um die einzelnen Versuche für die Studenten expertisegerecht zu gestalten[2].
Somit ist es möglich, dass die Anlagen ihren gesamten Handlungsspielraum an ihrer Schnittstelle darlegen und sich nur um den zuvor genannten Schutz von sich selbst und ihrer Umgebung kümmern müssen. Die tatsächlich durchzuführenden Experimente sind stattdessen auf dem EFaViS definiert und werden von diesem kontrolliert.
Ein Beispiel für die Expertisegerechtigkeit wäre etwa, dass der EFaViS, im Falle des Experiments der AniTA, das Hinzufügen von Essigsäureanhydrid zum Wasser erst gestattet, nachdem der Rührer angeschaltet wurde. Sicherheitstechnisch würde es kein Risiko darstellen, den Rührer ausgeschaltet zu lassen und es ist auch problemlos an der Schnittstelle der AniTA möglich, jedoch könnte durch unzureichende Verrührung die Temperaturmessung und somit das Ergebnis des Experiments verfälscht sein. So könnte der EFaViS dies Studenten, die dieses Experiment zum ersten Mal durchführen, verbieten, da es eine häufige Fehlerquelle wäre, während erfahreneren Studenten das Auslassen des Rührers gestattet ist.

3.1.3. Zugriffskontrolle

Um ungehinderten Zugriff auf das System zu verbieten und die Anlagen zu schützen, ist eine Zugriffskontrolle notwendig. Diese wird durch den in der Architektur in Abbildung 3.1 im zweiten Block von links dargestellten Experiment Control Server (ECS) realisiert.
Der ECS besteht aus einem User Management Server (UMS) und einem optionalen Time Table Server (TTS). Der UMS wird verwendet, um die Nutzer und Administratoren der Anlagen zu verwalten. Er muss nach Fertigstellung des Projektes von jeder der teilnehmenden Universitäten implementiert werden, da jede Universität ihre eigenen Nutzer verwaltet. Der TTS wird an jeder Universität implementiert, die eigene Experimentieranlagen zur Verfügung stellt und erfüllt die Aufgabe der Zeitbuchung. Durch den TTS wird geregelt, welche Nutzer zu welcher Zeit Zugriff auf bestimmte Experimente erhalten.

3.1.4. Benutzeroberfläche

Die letzte Komponente der Architektur bildet das Experiment Control User Interface (ECUI). Es besteht aus dem Time Table User Interface (TTUI), dem User Management Interface (UMUI) und dem Experiment Facility User Interface (EFUI). Das TTUI soll die Interaktion mit dem Time Table Server des ECS ermöglichen, damit Nutzer des Systems Zeiten an den Anlagen buchen können. Das UMUI ermöglicht die Verwaltung von Nutzern im UMS und ist somit für die Administratoren einer Universität gedacht. Das EFUI stellt das Interface zur eigentlichen Bedienung, aber auch zur Wartung der Experimentieranlagen bereit. Auch diese Komponente der Architektur muss von jeder teilnehmenden Universität implementiert werden. Im Falle von Universitäten, die keine eigenen Anlagen anbieten, fällt das EFUI weg.
Da Nutzern in der Benutzerverwaltung unterschiedliche Rollen zugewiesen sind (beispielsweise Administrator oder Student), lassen sich die angezeigten UIs auch je nach Nutzergruppe spezialisieren. So kann einem Studenten, der das TTUI öffnet, nur die Option gegeben werden, Zeiten zu buchen, während einem Administrator der eigenen Universität stattdessen verwaltungstechnische Funktionen zur Verfügung gestellt werden. Ebenfalls möglich ist eine Unterscheidung am EFUI, so dass einem Studenten nur sein zugewiesenes Experiment angezeigt wird, dem Administrator aber ein Wartungsinterface der Anlage, welches alle Funktionen exponiert.

[2] [2], Seite 3

3.2. Steuerung der Anlagen über endliche Zustandsautomaten

Um die Sicherheit und korrekte Funktionsweise der Anlagen sicherzustellen und somit Anforderung S1 aus Abschnitt 2.6.2 zu erfüllen, wird sich des Konzepts der endlichen Zustandsautomaten bedient. Hierbei wird versucht, die mit der Experimentieranlage durchführbaren Aktionen so genau wie möglich auf Zustände abzubilden, indem für jede Kombination jeder möglichen Stellung der aktiv steuerbaren Komponenten der Anlage ein Zustand definiert wird. Beim Eintreten in diesen Zustand wird die jeweilige Stellung eingenommen.
Als Beispiel kann eine simple, gedachte Anlage dienen, die mittels einer Pumpe eine Flüssigkeit durch einen Schlauch mit vorgeschaltetem Ventil in ein Gefäß pumpt.
Da die Pumpe entweder an- oder ausgeschaltet und das Ventil entweder offen oder geschlossen sein kann, ergibt sich eine Zustandsmenge, wie sie in Abbildung 3.2 zu sehen ist.

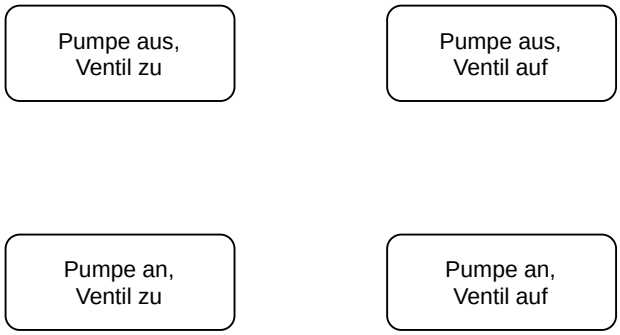

Abbildung 3.2.: Alle möglichen Zustände der aktiven Komponenten der Beispielanlage

Diese Menge aller möglichen Kombinationen der Stellungen der Komponenten enthält jedoch noch einen unsicheren Zustand, in welchem die Pumpe angeschaltet, das Ventil aber geschlossen ist. In einem solchen Zustand könnte der Schlauch, durch den die Flüssigkeit befördert wird, durch Überdruck platzen und somit Schaden an der Anlage anrichten und durch die auslaufende Flüssigkeit könnte Schaden an ihrer Umgebung entstehen.
Um dies zu verhindern, wird der Zustand, in dem die Pumpe an und das Ventil zu ist, entfernt. Außerdem werden Zustandsübergänge eingeführt, welche nur von einem in den nächsten sicheren Zustand wechseln. Das Zwischenergebnis dieser Aktion ist in Abbildung 3.3 zu sehen.

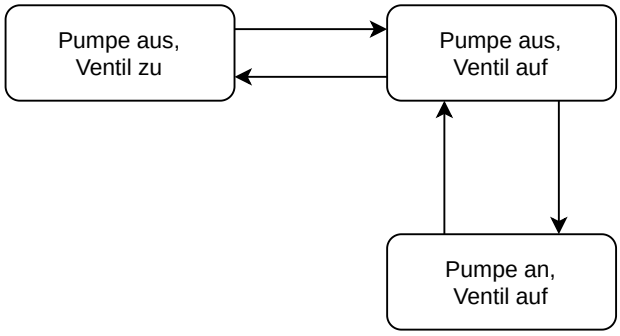

Abbildung 3.3.: Menge sicherer Zustände der Beispielanlage

Zur vollständigen Abbildung der Experimentieranlage als Zustandsautomat muss das Gefäß, in welches die Flüssigkeit gepumpt wird, betrachtet werden. Dieses ist keine aktiv steuerbare Komponente, aber sein Zustand ist trotzdem relevant, da es nicht durch zu langes Pumpen überlaufen darf.

Dies kann im Zustandsautomat dargestellt werden, indem für das Gefäß die Zustände “Gefäß voll” und “Gefäß nicht voll” erstellt werden, welche orthogonal zu den bisherigen Zuständen zur Steuerung der aktiven Komponenten stehen. Voraussetzung hierfür ist, dass der Füllstand des Gefäßes bekannt ist, bespielsweise durch einen Sensor oder die Berechnung der eingepumpten Menge an Flüssigkeit. Zum Wechseln der Zustände werden ebenfalls Events eingeführt, im Falle des Beispiels “ventilAuf”, “ventilZu”, “pumpeAn”, “pumpeAus” und “gefäßVoll”. Die durch die Events verursachten Zustandsübergänge werden an Bedingungen, sogenannte Guards, gekoppelt. So darf die Pumpe nur angeschaltet werden, wenn das Gefäß nicht voll ist. Außerdem lösen sie weitere Events aus. Wenn während des Pumpens das Gefäß in den Zustand “voll” wechselt, wird ein Event “pumpeAus” ausgelöst.
Der fertige Zustandsautomat dieses Beispiels kann in Abbildung 3.4 eingesehen werden. Er bildet die Funktionen der Beispielanlage komplett ab und sorgt gleichzeitig dafür, dass die Sicherheit der Anlage und ihrer Umgebung gewährleistet ist.
Nach diesem Prinzip wird im Umfang dieser Arbeit ein Zustandsautomat für die Steuerung der Experimentieranlage AniTA entworfen, was genauer in Abschnitt 4.6 erläutert wird.

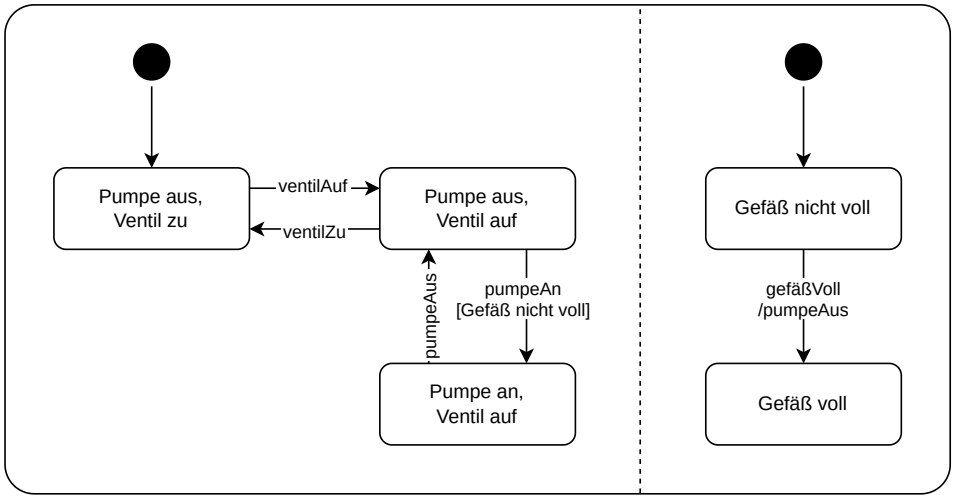

Abbildung 3.4.: Fertiger Zustandsautomat der Beispielanlage

3.3. Kommunikation

Zur Kommunikation der in 3.1 beschriebenen Bestandteile des Projektes untereinander wird ein einheitliches Kommunikationsprotokoll entwickelt.
In diesem werden die Komponenten in Schichten zusammengefasst, welche nur mit den benachbarten Schichten kommunizieren.
Die unterste Schicht bilden hierbei die Experimentieranlagen, welche nur mit der Zwischenschicht kommunizieren. Die Zwischenschicht besteht aus der Virtualisierung und vermittelt zwischen unterer und oberer Schicht. Die obere Schicht enthält die Benutzeroberfläche und die Zugriffskontrolle.

3.3.1. Automaten

Grundlegend für das Kommunikationsprotokoll sind endliche Zustandsautomaten. So wird für jede der genannten Schichten ein Automat entwickelt, der ihre Funktionalität beschreibt.
Angefangen bei den Experimentieranlagen werden, wie in Abschnitt 3.2 dargestellt, Automaten erstellt, die alle möglichen, sicheren Zustände, die die Anlage einnehmen kann, beinhalten.
Für die nachfolgenden Komponenten des Projektes werden ebenfalls Automaten erstellt, die auf denen der Experimentieranlagen basieren. Die zu einer Experimentieranlage zugehörige Virtualisierung

schränkt den durch den Anlagenautomaten abgebildeten kompletten Handlungsspielraum der Anlage insofern ein, dass ein oder mehrere durchführbare Experimente entstehen. Während der Automat der Anlage darauf ausgelegt ist, keine Zustände zu enthalten, die der Anlage oder ihrer Umgebung schaden könnten, ist der der Virtualisierung darauf ausgelegt, das Experiment zu schützen, indem nur Aktionen ausgeführt werden, die dem Ablauf des Experiments entsprechen.
Der Automat der Benutzeroberfläche ermöglicht in Kombination mit der Zugriffskontrolle unterschiedliche Sichten auf die Kontrolle der Anlagen. So können beispielsweise Administratoren andere Funktionen dargestellt werden als Studenten.
Da bei der Durchführung einer Zustandsänderung auf der Anlage Verzögerungen auftreten können, enthalten die Automaten der Schichten oberhalb der Anlagen Zwischenzustände, welche repräsentieren, dass sich die Anlage momentan in einer Zustandsänderung befindet.

3.3.2. Signale

Die Kommunikation zwischen den Schichten findet durch “Signale” statt. Diese haben einen eindeutigen Bezeichner und optionale Parameter.
Die Zustandsübergänge der jeweiligen Automaten werden, wie in Abschnitt 3.2 beschrieben, von Events ausgelöst.
Events können entweder automatenintern, beispielsweise durch das Erreichen eines kritischen Füllstands wichtiger Flüssigkeiten in der Anlage, oder durch externe Benutzereingaben entstehen. Bei externen Benutzereingaben wird ein Signal an die Anlage geschickt, welches ein Event und damit potentiell eine Zustandsänderung auslöst. Finden automateninterne Zustandsänderungen statt, wird ein entsprechendes Signal an die anderen Schichten geschickt, um diese synchronisiert zu halten.
Wird beispielsweise von einem Studenten der Befehl gegeben, eine Pumpe anzuschalten, dann wird vom zuständigen Automaten der Benutzeroberfläche geprüft, ob ein Nutzer der Rolle “Student” diese Pumpe überhaupt anschalten darf. Ist dies der Fall, sendet er ein Signal an die Virtualisierung, welches der Automat der Virtualisierung wiederum an die Anlage weiterleitet, sofern der Ablauf des Experiments dies erlaubt. Die Anlage stellt durch ihren Automaten sicher, dass es momentan problemlos möglich ist, diese Pumpe anzuschalten und dass die zu pumpende Flüssigkeit ausreichend vorhanden ist.
Somit findet durch die Automaten auf jeder Schicht eine Überprüfung statt, sobald ein Signal vom Endnutzer ausgelöst wird. Auf jeder Schicht kann es zur Ablehnung des Signals kommen, was den momentanen Befehl abbrechen würde. Die endgültige Annahme kann erst auf der untersten Schicht geschehen, nachdem die vorherigen Überprüfungen ebenfalls positiv verlaufen sind. In diesem Fall sendet die Anlage ein Akzeptierungssignal zurück und beginnt damit, den neuen Zustand anzunehmen. Das Akzeptierungssignal überführt die Automaten der höheren Schichten in ihre Zwischenzustände. Sobald die Anlage den Wechsel vollzogen hat und sich wieder in einem sicheren, konsistenten und bekannten Zustand befindet, sendet sie eine Bestätigung aus, durch welche die anderen Automaten aus dem Zwischenzustand in den endgültigen Zustand nach Ausführung des Befehls wechseln.
Dieser Vorgang ist in Abbildung 3.5 in einem Sequenzdiagramm beispielhaft dargestellt. Der Aktivitätsbalken der Anlage repräsentiert hierbei die Zeit, die diese für den Zustandswechsel benötigt.
Anders herum, wenn das Event automatenintern auf der Anlage ausgelöst wird, wird direkt ein Akzeptierungssignal an die höheren Schichten geschickt und mit dem Zustandswechsel begonnen.
Da es in manchen Fällen erforderlich ist, den Signalen zusätzliche Informationen hinzuzufügen, können mit einem Signal auch zugehörige Parameter verschickt werden. So ist es etwa möglich, ein Signal zu senden, welches den Rührer einer Anlage anschaltet und als zusätzlichen Parameter eine Drehzahl anzugeben, so dass die Anlage diese anstelle der Standarddrehzahl verwendet. Dieses Konzept kann ebenfalls verwendet werden, um Sensordaten oder Ähnliches zu versenden. Diese Signale würden von den Experimentieranlagen ausgehen und keine Zustandsänderungen in den Automaten der höheren Schichten auslösen, sondern ihre Daten als Parameter enthalten.

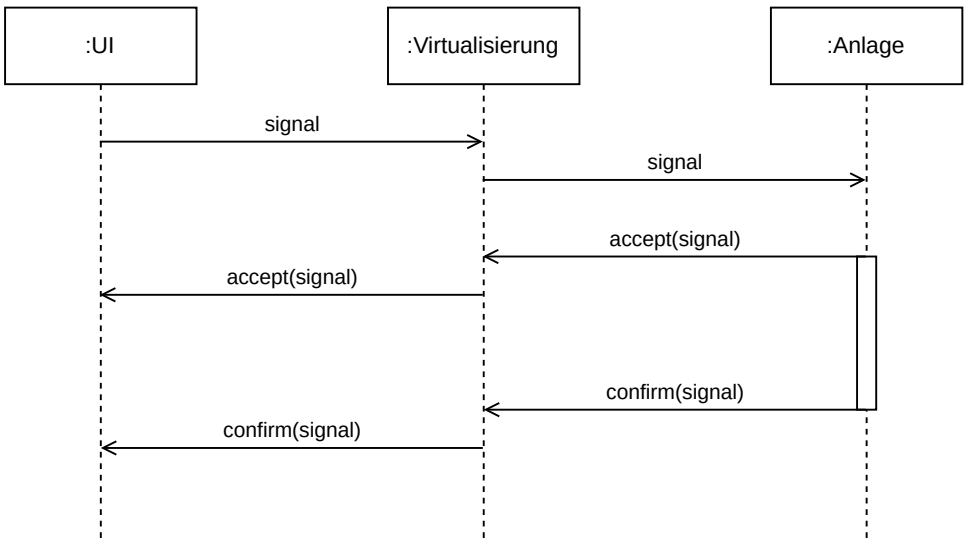

Abbildung 3.5.: Ablauf der Kommunikation über Signale

Somit ergeben sich fünf unterschiedliche Arten von Signalen, welche es zu implementieren gilt:

1. Signale, welche im Automaten der Anlage eine Zustandsänderung bewirken und optional Parameter enthalten.
2. Signale, die von der Anlage als Bestätigung verschickt werden, dass eine Zustandsänderung möglich ist und ausgeführt wird.
3. Signale, die von Automaten jeder Schicht verschickt werden, wenn eine Zustandsänderung nicht möglich ist.
4. Signale, die von der Anlage verschickt werden, sobald ein neuer Zustand eingenommen wurde.
5. Signale, mit denen Daten verschickt werden.

3.3.3. General Experiment Control Protocol

Um die Kommunikation zwischen den Schichten zu gewährleisten und die vorangegangenen Konzepte zu realisieren, wird in einer kleineren Arbeitsgruppe innerhalb des Projektes das General Experiment Control Protocol (GECP) entworfen. Dieses textbasierte Protokoll legt fest, wie Nachrichten, welche zwischen den Schichten versendet werden, aufgebaut sind. Es muss in der Firmware der Experimentieranlage AniTA implementiert werden, um Anforderung F2 aus Abschnitt 2.6.1 zu erfüllen. Eine Nachricht des GECP besteht aus einem Header und einem Body, welche durch eine Leerzeile voneinander getrennt sind. Der Header beinhaltet die Version des Protokolls und einen Operationscode, welcher angibt, um was für eine Art von Nachricht es sich handelt. Es gibt drei für die Anlage relevante Operationscodes: “signal”, mit welchem die in Abschnitt 3.3.2 beschriebenen Signale verschickt werden, “state-req”, mit welchem die Anlage aufgefordert werden kann, ihren aktuellen Zustand mitzuteilen und “state-resp”, womit die Anlage daraufhin antwortet. Konzeptuelle Darstellungen der Nachrichten können in Abbildung 3.6 eingesehen werden.
Der Inhalt des Bodys ist abhängig vom im Header angegebenen Operationscode. Im Falle von “signal” besteht der Body aus einem Feld namens “signal”, welches das zu verschickende Signal enthält und einem optionalen Feld “parameters”, falls es zu dem Signal zugehörige Parameter gibt. Die Parameter haben wiederum eigene Bezeichner und Werte und werden unter dem “parameters” Feld

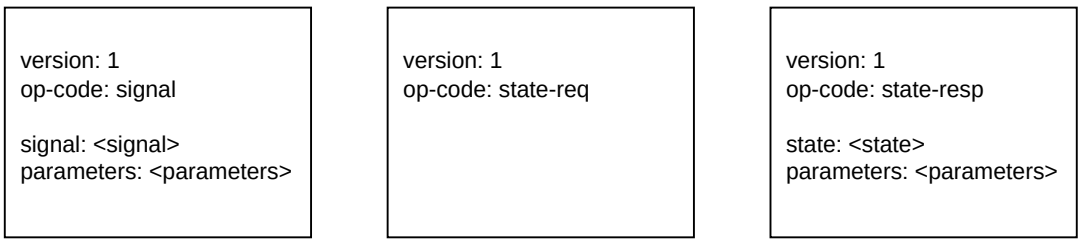

Abbildung 3.6.: Aufbau der unterschiedlichen Nachrichten

eingerückt mit Bezeichner und Wert angehangen. Somit können die in Abschnitt 3.3.2 genannten Signale zur Steuerung der Zustandsautomaten versendet und Anforderung F2.2 aus Abschnitt 2.6.1 erfüllt werden.
Ebenfalls können Messdaten mit diesem Protokoll versendet werden, indem ein Signalname definiert wird, der anzeigt, dass es sich um ein Messdatenupdate handelt und die Daten im Parameterfeld mit jeweils eigenem Bezeichner eingefügt werden. Dies ist notwendig um Anforderung F2.3 aus Abschnitt 2.6.1 zu erfüllen.
Konkrete Beispiele von Implementierungen der genannten Nachrichten werden in Abbildung 3.7 gezeigt.
Beim Operationscode “state-req” ist der Body leer, bei “state-resp” gibt es ein Feld “state”, welches den Zustand der Anlage enthält. Ebenfalls können in der “state-resp” wieder Daten als Parameter angehängt werden. Ein konkretes Beispiel der State Response kann ebenfalls in Abbildung 3.7 eingesehen werden.
Wie die Signale und Parameter im spezifischen Fall der Anlage AniTA bezeichnet werden, wird in Abschnitt 4.7.3 erläutert. Dort wird ebenfalls auf die Kodierung der Zustände und die Bezeichner der Messdaten eingegangen.

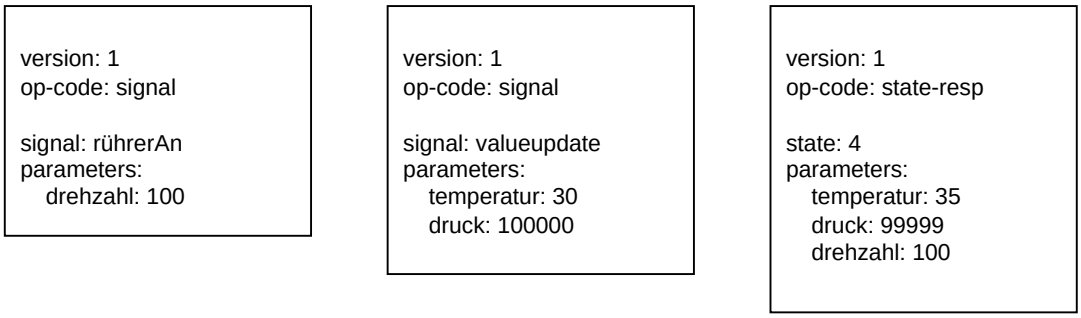

Abbildung 3.7.: Konkrete Beispiele für Nachrichten des GECP

3.4. Anlagenfirmware

Unter Berücksichtigung der in Abschnitt 2.3.2 beschriebenen Komponenten der Anlage, der in Abschnitt 3.1 erklärten Rahmenarchitektur und den in Abschnitt 3.3 eingeführten Konzepten, wird eine Architektur für die Firmware des Steuergerätes der Experimentieranlage AniTA entworfen, welche in Abbildung 3.8 zu sehen ist. In diesem Software Block Diagramm repräsentieren abgerundete Kästen die relevanten, physischen Komponenten, die im System mit der Anlage interagieren. Der große abgerundete Kasten in der Mitte stellt die Steuereinheit der Anlage dar, welche zu programmieren Ziel dieser Arbeit ist. Die peripheren Komponenten ergeben sich, im Falle der Ventile, des Lichts, der Pumpen, des Thermoelements und des Rührers, aus der Beschreibung der Experimentieranlage in

Abschnitt 2.3.2 und für den EFaViS aus der in Abschnitt 3.1 erklärten Rahmenarchitektur. Ebenfalls wurde in der Architektur die Möglichkeit berücksichtigt, zu Debug-Zwecken und zur erleichterten Entwicklung ein externes Terminal anzuschließen.
Die innerhalb der Steuereinheit liegenden eckigen Kästen repräsentieren Software, welche auf dem Steuergerät laufen muss und welche als Gesamtheit die Firmware des Steuergerätes darstellt. Kontakt der Kästen miteinander bedeutet, dass diese Softwarekomponenten interagieren und aufeinander aufbauen.
Jeder Kasten der Architektur muss als Softwaremodul umgesetzt werden, wobei sie in drei Aufgabenbereiche unterteilt werden können, welche im Folgenden genauer beschrieben werden.

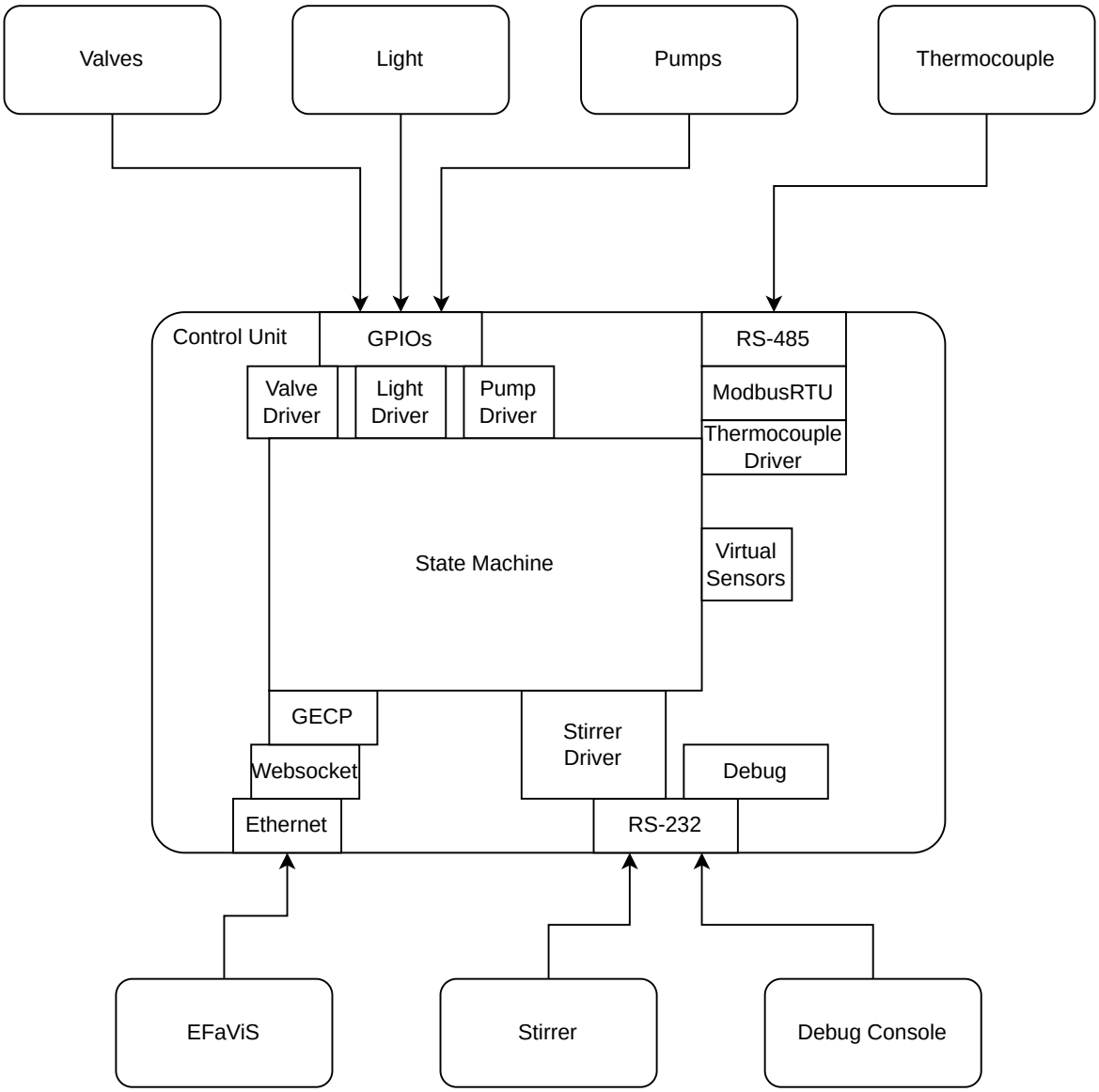

Abbildung 3.8.: Software Block Diagramm der Architektur der AniTA-Firmware

3.4.1. Treiber

Dieser Aufgabenbereich umfasst das Schreiben der Treiber zur einfachen Steuerung der Komponenten der Anlage. Dies ist notwendig, um Anforderung F1.1 aus Abschnitt 2.6.1 zu erfüllen. Die

Treiber sollen eine leichte Bedienbarkeit der Komponenten ermöglichen, so dass andere Module beispielsweise mit einem einfachen Funktionsaufruf die Ventile öffnen und schließen, die Temperatur auslesen oder den Rührer anschalten können.
Da die Komponenten auf unterschiedlichste Weise angeschlossen sind, wird ebenfalls Software benötigt, die diese Schnittstellen ansprechen kann. Im Falle der Ventile, Pumpen und des Lichts ist dies nur Software, die GPIOs an und aus schalten kann, bei dem Thermoelement ist dies aber komplizierter. Angeschlossen ist dieses über eine RS-485 Schnittstelle. Zur Kommunikation mit dem Thermoelement wird darauf aufbauend ModbusRTU, ein mehrteilnehmer Busprotokoll, verwendet[3], so dass neben Code zur Nutzung einer RS-485 Schnittstelle ebenfalls Code zur Implementierung des Modbus Protokolls benötigt wird. Dieser kann dann von dem Thermoelement-Treiber verwendet werden, um das Thermoelement auszulesen.
Der Treiber für den Rührer und das Debugmodul setzen wiederum auf Code für RS-232 Schnittstellen auf. Hierbei muss jedoch beachtet werden, dass das Steuergerät über mindestens zwei dieser seriellen Schnittstellen verfügt, da der Rührer und die Debug Konsole nicht gleichzeitig an derselben Schnittstelle angeschlossen sein können. Selbiges gilt auch für die über GPIOs angesprochenen Komponenten, eine ausreichende Dimensionierung muss vom Steuergerät gegeben sein.

3.4.2. Netzwerk

Im Aufgabenbereich Netzwerk werden die Module behandelt, welche zur Kommunikation mit dem EFaViS und somit zur Erfüllung der Anforderung F2 aus Abschnitt 2.6.1, erforderlich sind. Letztendlich handelt es sich hier um die Implementierung des in Abschnitt 3.3.3 beschriebenen General Experiment Control Protocol. Das GECP-Modul muss empfangene Nachrichten analysieren und die wichtigen gesendeten Informationen aus diesen extrahieren. Ebenfalls müssen Signale von der Anlage in Protokollkonforme Nachrichten umgewandelt werden.
Da das Protokoll auf der Websockettechnologie aufbaut und die Anlage mit dem EFaViS per Ethernet verbunden ist, wird ebenfalls Code benötigt, der eine Ethernetschnittstelle steuert und das Websocketprotokoll implementiert.

3.4.3. Zustandsautomat

Der letzte und wahrscheinlich größte Aufgabenbereich ist der des Zustandsautomaten. Wie man anhand der Architektur in Abbildung 3.8 erkennen kann, wird der Zustandsautomat Kontakt mit sieben weiteren Softwaremodulen haben und ist somit das Herzstück der Firmware. Um die in Abschnitt 3.2 erklärte Funktionalität bereitzustellen und die gesamte Anlage zu steuern, muss er die zuvor erstellten Treiber der Komponenten benutzen. Ebenfalls muss Software für “Virtuelle Sensoren” erstellt werden, um Anforderung F1.2 aus Abschnitt 2.6.1 zu erfüllen. Diese soll auf eine treiberähnliche Art Zugang zu den in Abschnitt 2.3.2 beschriebenen, nicht aktiv ansteuerbaren Komponenten ermöglichen. Damit sollen beispielsweise der Füllstand des Essigsäureanhydridvorrats oder der Anteil an Wasser im Glasreaktor anhand von anderen Größen wie der jeweiligen Pumpdauer errechnet werden können. Dies ist wichtig, um den tatsächlichen, physischen Zustand der Experimentieranlage möglichst genau abbilden zu können und in Kombination mit den Treibern der aktiven Komponenten alle Anforderungen an die Sicherheit aus Abschnitt 2.6.2 zu erfüllen.
Da dieser Automat wie in Abschnitt 3.3.2 beschrieben über eingehende Signale gesteuert werden soll und selber Signale und den aktuellen Zustand im Rahmen des GECP versenden muss, bedient sich dieses Softwaremodul ebenfalls des zuvor erstellten Netzwerk-Codes.

4. Realisierung

In diesem Kapitel wird die Umsetzung der in Abschnitt 3.4 genannten benötigten Softwaremodule der Anlagenfirmware beschrieben. Ebenfalls wird auf die Kommunikation der Anlage mit ihrer Umgebung eingegangen und die zur erleichterten Entwicklung und zum Testen des Systems verwendeten Tools gezeigt.

4.1. Steuereinheit

Als Steuereinheit in der Anlage kommt, wie in Abschnitt 2.3.2 gelistet, ein Controllino MEGA zum Einsatz. Diese Entscheidung wurde von der Abteilung Technische Chemie 2 getroffen und begründet sich daraus, dass der Controllino MEGA über alle benötigten Schnittstellen zur Steuerung der Komponenten der Experimentieranlage verfügt.
So existieren 24 digitale 2 Ampere Output Pins, welche direkt von der Stromversorgung des Controllino gespeist werden und somit die 24 Volt Schlauchpumpen, Flüssigkeitsventile und das Licht steuern. Ebenfalls verfügt der Controllino MEGA über mehrere serielle Schnittstellen, an denen der Rührer, der Thermoelementleser und ein Debug-Computer gleichzeitig angeschlossen werden. Zur Verbindung mit dem Computer, auf dem die in Abschnitt 3.1.2 beschriebene Virtualisierungssoftware läuft, existiert ein Ethernet Interface. Zusätzlich kann der Controllino MEGA, wie in Abbildung 2.3 zu sehen ist, an einer DIN EN 50022 Tragschiene angebracht werden.
Der Controllino MEGA ist eine frei programmierbare SPS, die sich der Arduino Libraries bedient. Der integrierte ATmega2560 kann über die Arduino Toolchain programmiert werden und die vom Hersteller Controllino GmbH bereitgestellten Bibliotheken ermöglichen leichten Zugriff auf die Schnittstellen des Steuergeräts.

4.2. Entwicklungsumgebung

Für die Realisierung des Gesamtprojektes werden von der Abteilung Software Engineering zur Versionsverwaltung mehrere Git-Repositories auf der GitLab Instanz der Universität Oldenburg erstellt[1].
In diesen liegt neben der Software der in Abschnitt 3.1 genannten Komponenten des Projektes auch die Firmware der Experimentieranlage AniTA. Um die Anforderungen P1 und P2 aus Abschnitt 2.6.3 zu erfüllen, existieren ebenfalls Repositories, in denen modulare, wiederverwendbare Teile der im Rahmen dieser Arbeit entstandenen Software gehostet werden. So können zukünftige Experimemtieranlagen diese ebenfalls verwenden und leichter realisiert werden.
Zur Entwicklung der Firmware der in Abschnitt 4.1 beschriebenen Steuereinheit wird die PlatformIO Software[2] benutzt. Diese ermöglicht ein einfaches Programmieren des Flash-Speichers des Controllino MEGA. Ebenfalls besitzt sie eine integrierte Paketverwaltung, mittels derer benötigte Libraries, wie der von der Controllino GmbH entwickelten Controllino Library, automatisch heruntergeladen und in das Projekt eingefügt werden. Über diese Paketverwaltung ist es auch möglich, die Software aus den oben beschriebenen Repositories des Gesamtprojektes zu integrieren.

[1] `https://gitlab.uni-oldenburg.de/se/projects/experiment-system`
[2] `https://platformio.org/`

4.3. Schnittstellen

In diesem Abschnitt werden Informationen zu den benutzten Schnittstellen gegeben und es wird erklärt, wie die in Abbildung 3.8 gezeigten Softwaremodule zur Steuerung dieser Schnittstellen des Steuergerätes umgesetzt sind. Durch Nutzung der Arduino und Controllino Bibliotheken muss keines dieser Module selber geschrieben werden. Die hier beschriebenen Software-Libraries sind notwendig, um mit den in Abschnitt 2.3.2 genannten Komponenten zu kommunizieren und werden somit zum Erfüllen der in Abschnitt 2.6.1 aufgestellten Anforderung F1.1 gebraucht.

4.3.1. GPIOs

Durch die `wiring_digital.h` Bibliothek wird Zugang zu den GPIOs ermöglicht. Sie bietet unter anderem die Funktionen `digitalWrite()` und `digitalRead()` an, mit welchen einzelne GPIO Pins geschaltet und ausgelesen werden können. Durch die im Controllino Framework gegebenen Bezeichner der einzelnen GPIO Pins ist es möglich, die in Abbildung 4.1 gezeigte Verkabelung in der Firmware zu realisieren und die angeschlossenen Komponenten zu steuern. Der Schaltplan ist ebenfalls im Anhang in Abbildung A.2 zu sehen.

4.3.2. RS-232/RS-485

Der Controllino MEGA verfügt über vier serielle Schnittstellen, welche über seine Pinheader, den USB-Anschluss und Schraubverbindungen bei den GPIO Pins erreichbar sind.
Über die Schnittstelle des USB-Anschlusses findet die Programmierung des Controllino MEGA statt. Die weiteren RS-232 Schnittstellen sind über die Pinheader an der Oberseite des Controllino MEGA erreichbar. Dort ist der Rührer angeschlossen. Die RS-485 Schnittstelle befindet sich an Schraubverbindungen in der GPIO Pinleiste, dort ist der Thermoelementleser angeschlossen.
Bei der Steuerung dieser Schnittstellen wird durch die Arduino Bibliotheken wieder ein großes Stück Arbeit abgenommen. Durch die existierenden `HardwareSerial` Libraries wird Zugriff auf die seriellen Schnittstellen gegeben, welcher durch die `Print` Library noch erweitert wird. Die Bibliotheken verwalten den Puffer für ausgehende und einkommende Nachrichten und stellen vereinfachende Funktionen wie `read()`, `write()` und `print()` bereit.

4.3.3. Ethernet

Der Controllino MEGA besitzt eine RJ-45 Buchse, welche von einem WIZnet W5100 Chip gesteuert und als Ethernet Schnittstelle benutzt wird. Der W5100 ist über Serial Peripheral Interface (SPI)

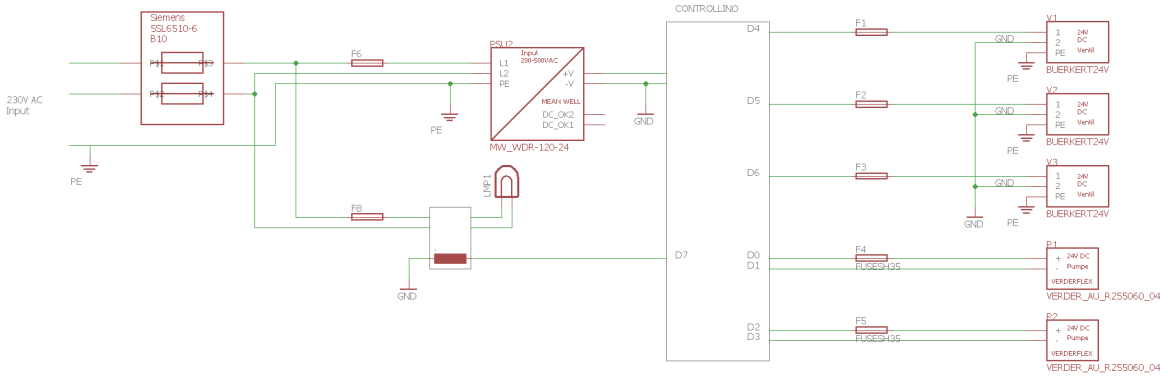

Abbildung 4.1.: Schaltplan des Steuergerätes und der über GPIOs angeschlossenen Komponenten

mit dem ATmega2560 verbunden. Durch die Arduino Library `Ethernet.h` wird die Kommunikation zwischen den Chips geregelt und einfacher Zugriff auf die Ethernetfunktionalitäten ermöglicht. Durch Limitierungen seitens des W5100 können maximal vier unterschiedliche Clients zur selben Zeit mit dem Controllino verbunden sein[3].

4.4. Treiber

In diesem Abschnitt werden die aktiv steuer- und auslesbaren Komponenten der Anlage und die zur Vereinfachung der Steuerung geschriebenen Treiber beschrieben. Die hier gezeigte Software erfüllt in Verbindung mit den in Abschnitt 4.3 beschriebenen Bibliotheken der Schnittstellen die in Abschnitt 2.6.1 gestellte Anforderung F1.1.

4.4.1. Ventile

Die Anlage besitzt drei Ventile, je eines am Ende der beiden Zulaufschläuche, welche diese verschließen und unerwünschtes Nachlaufen der Reagenzien vermeiden sollen, und eines am Boden des Glasreaktors, welches durch bloßes Öffnen den Inhalt des Reaktors in den Abfluss entsorgen kann. Als Komponenten werden Bürkert Typ 6013 A 2/2 Wege Flüssigkeitsventile eingesetzt.
"2/2 Wege" besagt, dass das Ventil über zwei Schlauchanschlüsse verfügt und zwei Schaltstellungen besitzt. Somit können 2/2 Wegeventile in eine Schlauchleitung integriert werden und diese wahlweise verschließen oder öffnen.
Aufgrund der Bauweise als stromlos geschlossene Magnetventile sind die Ventile standardmäßig verschlossen und öffnen sich nur bei Anlegung von 24 Volt Gleichspannung. Dies vereinfacht die Ansteuerung, da nur diese Spannung angelegt werden muss und erhöht die Sicherheit, da die Ventile bei Stromverlust automatisch schließen[4] und unerwünschtes Fließen von Reagenzien verhindern.

Funktion	**Parameters**	**Returns**
`Valve(uint8_t pin)`	`pin`: Pin, an den das Ventil angeschlossen ist	`Valve`
`setValveState(bool valve_state)`	`valve_state`: `true` - Ventil auf; `false` - Ventil zu	—
`bool getValveState()`	—	`true`: Ventil offen; `false`: Ventil geschlossen

Tabelle 4.1.: Funktionen des Ventiltreibers

Zur Umsetzung des laut Abbildung 3.8 benötigten Ventiltreibers wird sich der in Abschnitt 4.3.1 beschriebenen, gestellten GPIO Library bedient. Der Treiber stellt eine Klasse `Valve` bereit, in deren Constructor ein GPIO Pin übergeben werden muss. Hier werden die im Header `Controllino.h` definierten Makros verwendet, um die richtigen Pins auszuwählen und die in Abbildung 4.1 gezeigte Verschaltung des Steuergeräts zu erreichen. Die Klasse verfügt über Methoden zum Öffnen und Schließen des Ventils, welche einen entsprechenden Aufruf zum Schalten des benötigten GPIO Pins weiterleiten und speichert den momentanen Zustand des Ventils, welcher mit einer Methode abgerufen werden kann. Diese Funktionen können in Tabelle 4.1 eingesehen werden.

[3] [4], Seite 5
[4] [5], Seite 5

4.4.2. Dosierpumpen

Die Dosierpumpen vom Typ Verderflex OEM M025 DC sind Schlauchpumpen, von denen zwei eingesetzt werden, um die Reagenzien Wasser und Essigsäureanhydrid über Schläuche in den Glasreaktor zu befördern. Sie werden ebenfalls mit 24 Volt Gleichspannung betrieben, welche an zwei elektrischen Anschlüssen angelegt wird. Da die Pumpen von Gleichstrommotoren betrieben werden[6], bestimmt die Polarität der Anschlüsse die Laufrichtung der Pumpe. Somit kann der Inhalt der Schläuche auch zurück in die Vorratsbehälter gepumpt werden.
Da die Pumpen wie in Abbildung 4.1 zu sehen ebenfalls an den GPIO Pins des Controllinos angeschlossen sind, nutzt der Pumpentreiber analog zu dem Ventiltreiber die GPIO Library, um die beiden Pins der Pumpe, welche im Konstruktor angegeben werden müssen, zu steuern. Es wird jeweils eine Funktion zum Starten der Pumpe in Vorwärts- und Rückwärtsrichtung und eine Funktion zum Stoppen der Pumpe gegeben. Ebenfalls existiert eine ungenutzte Funktion, welche als Parameter ein gewünschtes Pumpvolumen verlangt und die Pumpe eine entsprechende Zeit lang laufen lässt.

Funktion	**Parameters**	**Anmerkung**
`Pump(uint8_t pin1, uint8_t pin2)`	`pin1`: Pin des ersten Terminals `pin2`: Pin des zweiten Terminals	Konstuktor
`on()`	-	Schaltet Pumpe vorwärts
`reverse()`	-	Schaltet Pumpe rückwärts
`stop()`	-	Schaltet Pumpe aus
`pumpMilliliter( uint16_t ml, bool reverse)`	`ml`: Zu pumpendes Volumen in Millilitern `reverse`: True für rückwärts pumpen	Pumpt das gegebene Volumen in gewünschter Richtung. Blockiert für die dauer des Pumpens durch Aufruf von `delay()`

Tabelle 4.2.: Funktionen des Pumpentreibers

4.4.3. Licht

Das Licht ist die letzte Komponente, die per GPIOs angeschlossen ist. Der Treiber stellt ebenfalls eine Klasse `Light` bereit, mit welcher ein Objekt mit zugehörigem Pin erstellt werden kann, welches zwei Methoden zum An- und Ausschalten des Lichts und eine Methode zur Statusabfrage bietet.

4.4.4. Rührer

Der an der Anlage angebrachte Rührer ist ein Heidolph Hei-TORQUE Ultimate 400 Laborrührer. Dieser dient dem Umrühren der Reagenzien im Reaktorglas. Der Rührer ist ein eigenständiges Gerät, welches auch unabhängig vom Steuergerät bedient werden kann. Die Verbindung zwischen Steuergerät und Rührer dient nur der Fernsteuerung, der Rührer verfügt über eine eigene Stromversorgung. Der Rührer ist über seinen RS-232 Port mit dem in Abschnitt 2.3.2 gelisteten RS-232 Adapter verbunden, welcher wiederum an die der dritten Seriellen Schnittstelle (`Serial2`) entsprechenden Pins im Pinheader X2 auf der Oberseite des Controllino MEGA angeschlossen ist. Die Schnittstellenparameter sind 9600 Baud, 8 Bit, kein Paritätsbit, 1 Stopbit[5].

[5] [7], Seite 38

Die Kommunikation geschieht über simple ASCII Kommandos. Von der Gesamtmenge möglicher Befehle, welche im Handbuch des Rührers beschrieben ist[6], wurde eine Untermenge herausgesucht, die für die Zwecke des Projekts ausreicht und in Tabelle 4.3 gelistet ist. Über den erstellten Rührertreiber ist es aus Sicherheitsgründen nur möglich, die definierten Befehle zu versenden.
Es wird eine Klasse `Stirrer` erstellt, welche im Konstruktor ein `HardwareSerial` Interface-Pointer übergeben bekommt. Im konkreten Anwendungsfall ist dies das angesprochene `Serial2` Interface, an das der Adapter angeschlossen ist.
Die Klasse bietet die Methode `sendCommand(allowedCommands cmd, uint16_t rpm)` an, welche einen der erlaubten Befehle ausführt. Hierfür wurde das Enum `allowedCommands` erstellt, welches die erlaubten Befehle durchnummeriert. Einer der Befehle muss als Parameter an `sendCommand()` übergeben werden. Die Methode sendet über das gegebene Interface den spezifizierten Befehl. Falls es sich um den `Rxxxx\r\n` Befehl handelt, muss ebenfalls der optionale Parameter `rpm` übergeben werden. Um die in Abschnitt 2.6.2 gestellte Anforderung S1.4 zu erfüllen, kann die maximal erlaubte Drehzahl in der Konfigurationsdatei (siehe Abschnitt 4.8) eingestellt werden.
Der Ablauf der `sendCommand()` Methode ist in Abbildung 4.2 abgebildet. Da der Rührer etwas Zeit braucht, um den Befehl zu verarbeiten, müssen zwischen einzelnen Befehlen mindestens 0,1 Sekunden liegen[7]. Um dies zu garantieren, blockiert die Funktion bei zu schnellem Aufruf, bis die benötigte Zeit seit des letzten Aufrufs erreicht ist. Nach dem Senden wartet die Methode auf eine Antwort. Sie wartet so lange, bis das letzte empfangene Zeichen ein `\r` ist oder ein im Konstruktor festgelegter Timeout von standardmäßig 50 Millisekunden erreicht wird. Da es von Seiten des Rührers kein Zeichen gibt, dass Übertragungen terminiert, muss auf den Carriage Return Character `\r` zurückgegriffen werden, welcher in jeder Antwort des Rührers vorkommt. Weil dieser in manchen Antworten jedoch mehrmals vorkommt, wird bei diesen der Kontrollfluss schon zurückgegeben, bevor die Antwort komplett angekommen ist. Da diese Teile der Antwort aber keine relevanten Informationen enthalten, wird dies in Kauf genommen.
Wenn es zu einem Timeout beim Warten auf eine Antwort kommt, muss davon ausgegangen werden, dass ein Fehler vorliegt und der Rührer den Befehl nicht ausgeführt hat. Häufigste Fehlerquelle stellt hier wahrscheinlich die separate Stromversorgung des Rührers dar. Er muss separat angeschaltet werden, bevor die Anlage in Betrieb genommen wird.
Der Rückgabewert der Methode enthält die (potentiell gekürzte) Antwort des Rührers oder einen Hinweis, dass der Rührer nicht erreicht werden kann.
Zusätzlich enthält der Treiber noch einige Funktionen, die die Bedienung des Rührers vereinfachen, indem sie häufig gebrauchte Befehle mittels der `sendCommand()` Methode an den Rührer senden und die Antwort des Rührers zurückgeben. So ist es möglich über einfache Funktionsaufrufe den Rührer an- und auszuschalten und Informationen wie die Drehzahl oder den Fehlercode des Rührers auszulesen.
Eine Übersicht über alle öffentlichen Methoden dieses Treibers ist im Anhang in Tabelle A.5 gegeben.

4.4.5. Temperaturfühler

Der im Glasreaktor der Anlage angebrachte Temperaturfühler ist ein Thermoelement vom Typ K nach DIN EN 60584[1]. Dieses wird von einem Expert Daq EX-9018BL-M Thermoelement Eingangsmodul ausgewertet. Dieses ist an der RS-485 Schittstelle (`Serial3`) des Controllino angeschlossen und verfügt über eine eigene Stromversorgung, welche zusammen mit der Experimentieranlage angeschaltet wird.

[6] [7], Seite 37
[7] [7], Seite 38

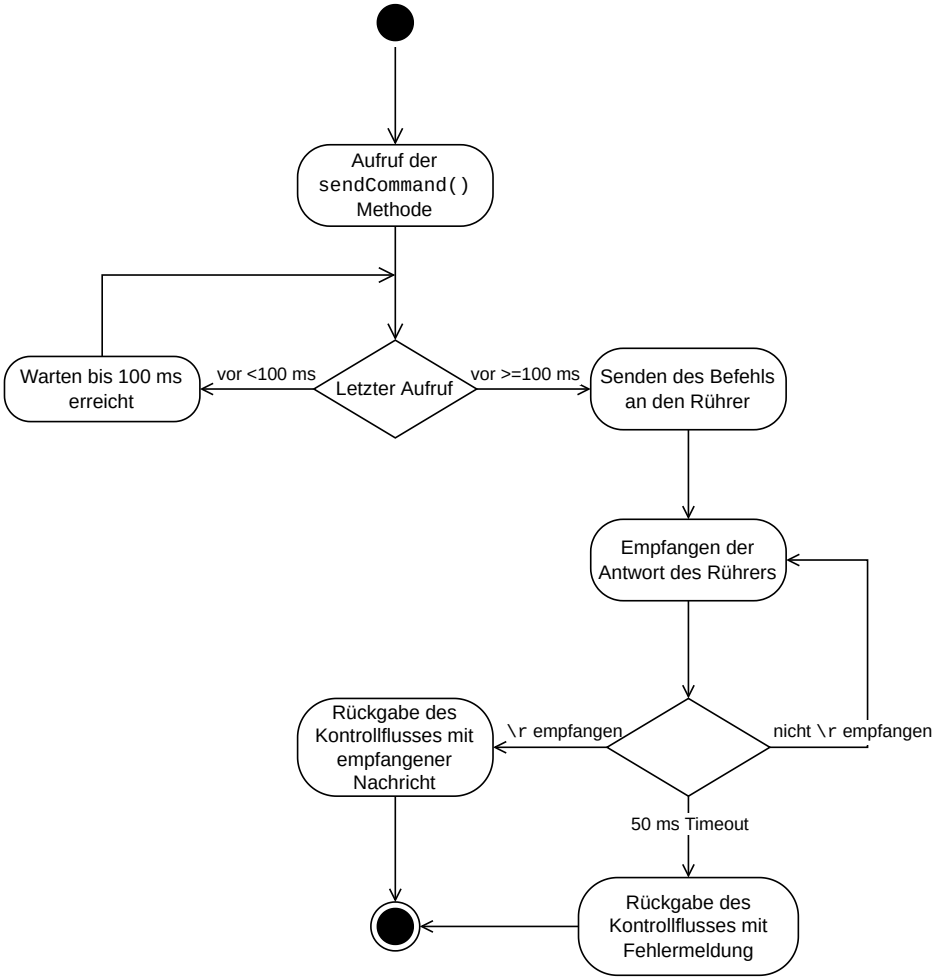

Abbildung 4.2.: Ablauf der `sendCommand()` Methode

Die Kommunikation mit dem Thermoelement-Eingangsmodul geschieht über ModbusRTU, was auf der RS-485 Schnittstelle aufbaut. Somit wird neben der in Abschnitt 4.3.2 erwähnten Library zur Steuerung der RS-485 Schnittstelle des Controllinos ebenfalls Software benötigt, die das Modbus Protokoll implementiert. Es gibt es dafür ebenfalls eine Arduino-Bibliothek, `ArduinoModbus.h`. Diese muss auf die Begebenheiten des Controllino angepasst werden und wird im in Abschnitt 4.2 beschriebenen Repository hinterlegt.
Der erstellte Treiber für den Temperaturfühler erhält über Präprozessormakros die für die RS-485 Schnittstelle und das Modbus Protokoll wichtigen Parameter, welche wie in Abschnitt 4.8 erläutert einstellbar sind. Ihre Standardbelegung ist ebenfalls im Anhang in Tabelle A.4 einsehbar. Besonders betont sei hier, dass die in der Betriebsanleitung des EX-9018BL-M genannten Registeradressen für die Modbusnutzung falsch sind[8]. Die tatsächlichen Adressen sind eine Nummer tiefer, so kann beispielsweise die Cold Junction Temperature an der Adresse 30128 anstatt der angegebenen Adresse 30129 gefunden werden.
Vom Treiber wird wieder eine Klasse bereitgestellt, die im Konstruktor die Slave-Adresse des Thermoelement Eingangsmoduls auf dem Modbus entgegennimmt und die RS-485 Schnittstelle über die Modbus Library initialisiert.
Die Klasse enthält nur die Methode `float getTemperature()`, welche über Modbus das Re-

[8] [3], Seite 49

Befehl zum Rührer	Rückmeldung vom Rührer	Bedeutung
`r\r\n`	`RPM: xxxx\r\n`	Rückmeldung enthält Ist-RPM
`s\r\n`	`SET: xxxx\r\n`	Rückmeldung enthält Soll-RPM
`m\r\n`	`NCM: xxxx\r\n`	Rückmeldung enthält Drehmoment
`f\r\n`	`FLT: No Error!\r\n`	Kein Fehler
	`FLT: Motor Error!\r\n`	Motorfehler
	`FLT: Motor Temperature!\r\n`	Motorüberhitzung
	`FLT: Stopped Manually!\r\n`	Rührer manuell gestoppt
	`FLT: overload!\r\n`	Motor überlastet
`C\r\n`	`Clear Error\r\n`	Fehler “overload” löschen
`R0000\r\n`	`R0000\r\nSET:0\r\n`	Rührer stoppen
`Rxxxx\r\n`	`Rxxxx\r\nSET:xxxx\r\n`	Rührer Soll-RPM setzen und ausführen

Tabelle 4.3.: Erlaubte Befehle an den Rührer und dessen Antworten. `xxxx` steht stellvertretend für eine variable, vierstellige Zahl, `\r` und `\n` für die ASCII-Zeichen carriage return und new line

gister des Thermoelement-Eingangsmoduls ausliest, in dem die aktuelle Temperaturmessung gespeichert ist, diese nach Vorgaben des Handbuchs des EX9018-BL-M konvertiert[9] und als Float zurückgibt. Somit wird ein Unterpunkt der in Abschnitt 2.6.1 gestellten Anforderung F1.2 erfüllt.

4.5. Virtuelle Sensoren

In diesem Abschnitt werden die in der Firmwarearchitektur in Abbildung 3.8 gezeigten “Virtuellen Sensoren” beschrieben.
Die virtuellen Sensoren werden im Folgenden von dem Zustandsautomaten benötigt, um den Zustand der Experimentieranlage korrekt einschätzen zu können.

4.5.1. Glasreaktor

Der Glasreaktor entspricht der in Abschnitt 2.3.2 genannten Komponente, in der die Reaktion zwischen Wasser und Essigsäureanhydrid stattfindet. Zur korrekten Einschätzung des Zustandes der Anlage und zum Erfüllen der in Abschnitt 2.6.1 gestellten Anforderung F1.2 wird hierbei der Füllstand des Glasreaktors insgesamt und die Anteile von je Wasser und Essigsäureanhydrid im Einzelnen benötigt.
Hierfür stellt der virtuelle Sensor eine Klasse bereit, welche in privaten Variablen speichert, wie viel Zeit mit dem Füllen von dem jeweiligen Stoff verbracht wurde. Anhand des tatsächlichen Volumenstroms der Pumpen kann damit das im Glasreaktor befindliche Volumen errechnet werden.
Die bei Anlagenstart, Füllvorgang und Entleerung benötigten Aktionen zum Gebrauch des virtuellen Sensors werden in Abbildung 4.3 in einem Aktivitätsdiagramm gezeigt.
Für den Füllvorgang werden für jeden Stoff zwei Methoden gegeben. Eine, die einmalig beim Start des Füllvorgangs aufgerufen werden muss und eine, die während des Füllvorgangs periodisch aufgerufen wird. Die initiale Methode setzt eine Zeitvariable auf die aktuelle Systemzeit. Die periodische

[9] [3], Seite 51

Methode addiert die Differenz der Zeitvariable zur aktuellen Systemzeit auf die private Variable, die die Gesamtzeit speichert und setzt danach die Zeitvariable auf die aktuelle Systemzeit. Somit muss ein Füllvorgang nicht zusätzlich beendet werden, dies geschieht automatisch durch das Stoppen der Aufrufe der periodischen Methode.
Analog werden zum Leeren des Glasreaktors zwei Methoden bereitgestellt, die auf die selbe Weise funktionieren. Da das Entleeren jedoch über das Ventil am Boden des Glasreaktors geschieht, was im Gegensatz zum Füllen durch die Dosierpumpen nicht präzise ist, kann nach dem Start des Entleervorgangs keine Aussage über die Füllstände mehr getroffen werden.
Somit gibt es eine Indikatorvariable, die zum Anfang eines Entleervorgangs gesetzt wird, welche ein Auslesen der Füllstände unterbindet. Die Indikatorvariable wird erst wieder gelöscht, sobald die periodisch aufzurufende Leerungsmethode solange aufgerufen wurde, bis der Reaktor sicher sein kann, dass er leer ist und die Füllstandsvariablen zurückgesetzt wurden. Die Dauer dessen hängt vom letzten bekannten Füllstand des Reaktors und einer in der Konfiguration (siehe Abschnitt 4.8) einstellbaren Abflussgeschwindigkeit ab. Die Abflussgeschwindigkeit sollte mit genügend Puffer gewählt werden, damit sie auf jeden Fall langsamer als die tatsächliche Abflussgeschwindigkeit ist. Es ist nicht schlimm, wenn der Reaktor meldet, dass er noch nicht leer ist, obwohl er schon leer ist. Hingegen kann es zu Schäden führen, wenn der Reaktor meldet, dass er leer ist, obwohl er noch nicht leer ist.
Zum Abrufen der Füllstände gibt es für Wasser, Essigsäureanhydrid und den Gesamtfüllstand je eine Methode, welche durch Verrechnen der privaten Variablen, die die Füllzeit in Millisekunden speichern, mit dem tatsächlichen Volumenstrom der Pumpen die Füllstände in Milliliter als Float zurückgibt.
Da der Füllstand des Glasreaktors beim Start der Anlage nicht bekannt ist, weiß der Reaktor nicht, wie lange er geleert werden muss. Um dies anzuzeigen, gibt es ebenfalls eine Variable, die während der Ausführung des Konstruktors des Glasreaktors gesetzt wird und das Auslesen der Füllstände verhindert, bis die `setEmpty()` Methode einmalig aufgerufen wurde. Also muss die Anlage zur korrekten Inbetriebnahme des Reaktors sicherstellen, dass dieser komplett geleert wurde, indem beispielsweise das Abflussventil so lange geöffnet wird, dass selbst ein gänzlich gefüllter Glasreaktor leer würde und danach die `setEmpty()` Methode aufrufen. Wie dies bei der AniTA implementiert wurde, kann in Abschnitt 4.6.5 eingesehen werden.

4.5.2. Essigsäureanhydrid Vorratsbehälter

Dies ist der Behälter, aus dem die Anlage das für das Experiment wichtige Essigsäureanhydrid bezieht. Da er im Gegensatz zum Wasserbehälter händisch aufgefüllt werden muss, ist es für die Betreiber der Anlage wichtig zu wissen, wann der Inhalt des Behälters zur Neige geht. Dies spiegelt sich in der in Abschnitt 2.6.1 genannten Anforderung F4 wider, für deren Erfüllung dieser virtueller Sensor benötigt wird. Er wird ebenfalls zur Erfüllung von Anforderung F5 benötigt, welche verbietet Experimente durchzuführen, wenn nicht genug Essigsäureanhydrid vorhanden ist.
Der “virtuelle Sensor” des Essigsäureanhydrid Vorratsbehälters funktioniert auf ähnliche Weise, wie der des Glasreaktors. Jedoch ist die Bedienung, welche in Abbildung 4.4 als Aktivitätsdiagramm dargestellt ist, weniger komplex, da nur eine Füllstandsänderungsrichtung berechnet werden muss. Wie beim Glasreaktor wird die Zeit, die der Behälter geleert wird, gemessen und damit auf den Füllstand rückgeschlossen. Hierfür gibt es wieder eine Methode, die beim Start des Entleervorgangs und eine, die kontinuierlich bis zu dessen Ende aufgerufen wird. Hinzu kommt noch eine Methode zum Auslesen des Füllstandes und eine, die den Füllstand auf das maximale, in der Konfiguration (siehe Abschnitt 4.8) einstellbare Volumen des Behälters setzt.
Zum Auffüllen des Behälters muss dieser mindestens bis zu dem eingestellten Wert gefüllt und danach die Methode aufgerufen werden.
Da beim Start der Anlage der Füllstand des Vorratsbehälters ähnlich wie bei dem Glasreaktor nicht

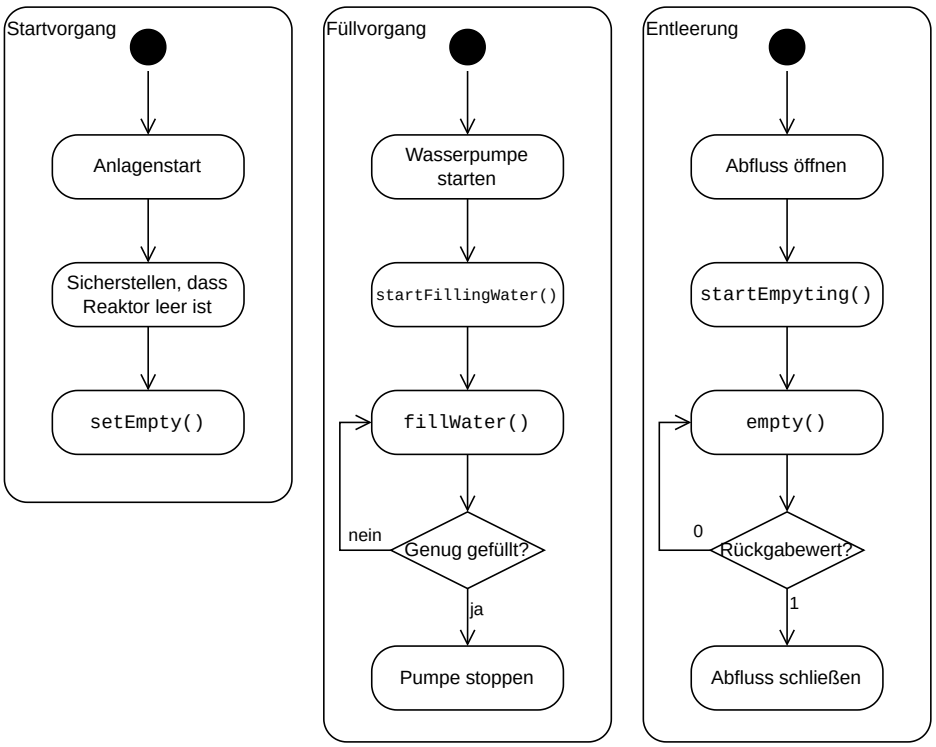

Abbildung 4.3.: Glasreaktor Aktivitätsdiagramm

bekannt ist, muss von einem leeren Behälter ausgegangen werden. Deswegen wird der berechnete Füllstand im Konstruktor auf 0 gesetzt und eine Inbetriebnahme ist erst nach einmaligem Auffüllen nach dem Start möglich.

4.5.3. Schläuche

Für die Schläuche werden ebenfalls virtuelle Sensoren nach demselbem Prinzip geschrieben. Mit diesen soll festgestellt werden, ab wann der Betrieb der Pumpen nicht mehr nur die Schläuche befüllt, sondern auch Flüssigkeiten in den Glasreaktor befördert werden und ab wann bei Rückwärtsbetrieb der Pumpen die Schläuche wieder geleert sind. Dies ist notwendig um die in Abschnitt 2.6.1 gestellten Anforderungen F6 und F7 zu erfüllen.
Während der Testphase hat sich jedoch herausgestellt, dass dies nicht zuverlässig möglich ist, da die Zeit, die die Schläuche zum Befüllen brauchen, variabel ist. Die Annahme, dass die Schläuche einfach Stück für Stück gefüllt werden, bis sie voll sind und die Flüssigkeit in den Reaktor fließt, stellte sich insofern als falsch heraus, dass beim Füllen der Schläuche immer Luft mit eingeschlossen wird. Somit werden bei annähernd vollem Schlauch immer abwechselnd Flüssigkeit und eingeschlossene Luftblasen herausgepumpt. Als Konsequenz hängt die Zeit zum Befüllen des Schlauchs von der Menge der eingeschlossenen Luftblasen ab und es ist nicht möglich, den Schlauch komplett zu befüllen, ohne Flüssigkeiten in den Reaktor zu befördern.
Deswegen wird der virtuelle Sensor der Schläuche verworfen. Wie die genannten Anforderungen trotzdem erfüllt werden wird in Abschnitt 4.6.5 erläutert.

4.6. Zustandsautomat der Experimentieranlage

Wie in Abschnitt 3.2 erklärt, geschieht die Steuerung der Komponenten der Experimentieranlage AniTA über endliche Zustandsautomaten. Es werden nach dem gezeigten Prinzip Unterautomaten

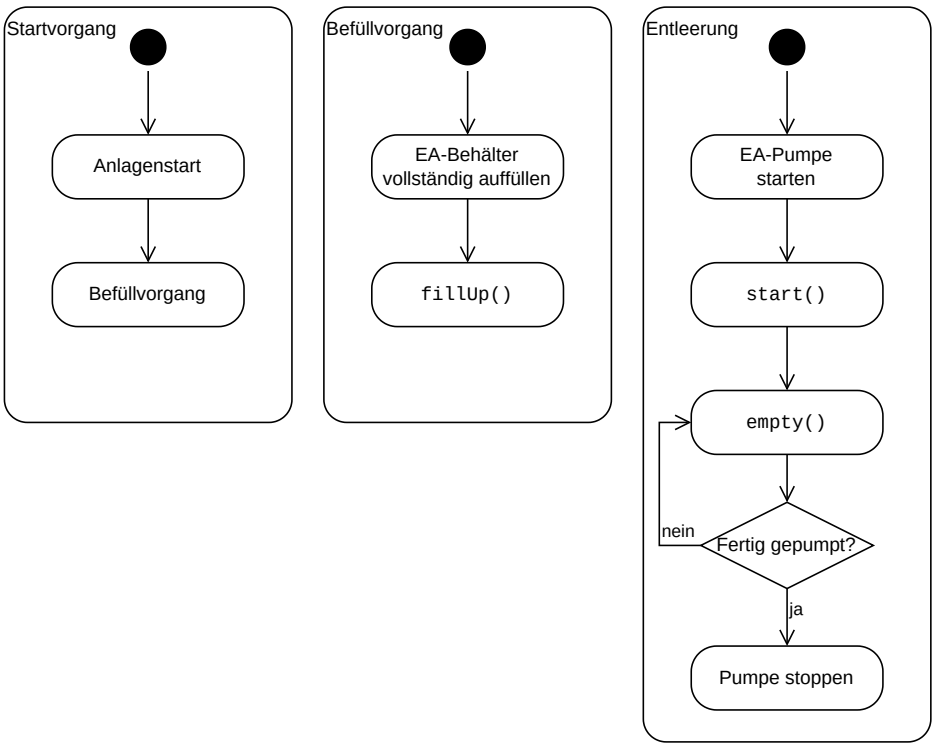

Abbildung 4.4.: Essigsäureanhydridbehälter Aktivitätsdiagramm

für die einzelnen Aktor- und Sensorgruppen erstellt, welche in Abschnitt 4.6.5 zu einem großen Gesamtautomat zusammengeführt werden. Dieser wird dann in der Firmware der Anlage umgesetzt. Bei der Umsetzung auf dem Controllino kommt die Embedded Template Library (ETL) zum Einsatz, welche Teile der Standard Template Library unter Nutzung von statischer Speicherallokation für eingebettete Systeme zur Verfügung stellt. Ebenfalls wird durch sie eine Bibliothek für endliche Automaten bereitgestellt, welcher sich für die Firmware bedient wird. Mittels der Bibliothek können die im Folgenden beschriebenen Unterautomaten erstellt werden. Die Bibliothek stellt für die Zustände `on_enter_state()` Funktionen bereit, mittels derer die je nach Zustand gewünschten Aktorstellungen beim Betreten eingenommen werden. Hierfür werden die in Abschnitt 4.4 beschriebenen Treiber verwendet. Außerdem werden durch die Bibliothek die in Abschnitt 3.2 beschriebenen Events, welche zu Zustandsänderungen führen, implementiert. Mittels der `on_event()` Funktionen der einzelnen Zustände wird festgelegt, in welchen Zustand sie bei welchen Events wechseln. Durch diese Funktionen können auch die beschriebenen Guards erstellt und bei Zustandsübergängen weitere Events ausgelöst werden.
Darüber hinaus wird eine Funktion `serviceStates()` erstellt, welche periodisch aufgerufen wird und je nach Zustand, in dem sich die Anlage momentan befindet, unterschiedliche Aktionen, wie das Berechnen von Füllständen, ausführt.
Events können durch die in Abschnitt 3.3.2 erklärten zustandsändernden Signale oder automatenintern ausgelöst werden. Um darzustellen, dass ein Zustandsübergang nur von der Anlage selbst ausgelöst werden kann, wird in den folgenden Abbildungen der Automaten das Präfix “FF::” vor das Event gesetzt. Lautet das Präfix “VF::”, dann kann die Zustandsänderung durch ein von außen kommendes Signal gestartet werden. Dies schließt eine automateninterne Auslösung jedoch nicht unbedingt aus. Guards werden in eckige Klammern gesetzt und bezeichnen hier, dass die Anlage ebenfalls im genannten Zustand verweilen muss, um den Zustandsübergang zu nehmen. Negationen mittels Ausrufezeichen sind ebenfalls möglich. Auslösung weiterer Events bei einem Zustandsübergang werden

durch einen Schrägstrich angezeigt.
Für AniTA ergibt sich eine Liste folgender Aktoren, die aktiv ansteuerbar und für die konkrete Durchführung der Experimente relevant sind:

- Ventil Wasser
- Ventil Essigsäureanhydrid
- Ventil Abfluss
- Pumpe Wasser
- Pumpe Essigsäureanhydrid
- Rührer
- Licht

Die Pumpen und Ventile werden zu einem Unterautomaten zusammengefasst. Für das Licht und den Rührer werden dazu orthogonale Automaten erstellt, da diese vollständig unabhängig von den Pumpen und Ventilen sind. Ebenfalls wird ein Automat für den Essigsäureanhydrid Vorratsbehälter und einer für den Glasreaktor, das Gefäß in dem die Reaktion stattfindet, kreiert.

4.6.1. Pumpen und Ventile

Die Pumpen und Ventile werden wie erwähnt zu einem Automaten zusammengefasst. Da jede Pumpe und jedes Ventil zwei Zustände besitzt, im Falle der Ventile “offen” und “geschlossen” und für die Pumpen “an” und “aus”, würde es insgesamt $2^5 = 32$ Zustände für die vorliegende Kombination an Aktoren geben. Diese Zahl wird durch die folgenden Beschränkungen reduziert, welche aus den in Abschnitt 2.6.2 gestellten Anforderungen S1.1 bis S1.3 abgeleitet sind:

- Eine Pumpe darf nur angeschaltet sein, wenn das ihr zugehörige Ventil offen ist.
- Die Pumpen für Wasser und Essigsäureanhydrid dürfen nicht gleichzeitig laufen.
- Das Abflussventil darf nur offen sein, wenn alle anderen Pumpen aus und die zugehörigen Ventile geschlossen sind.

Somit fallen alle Zustände weg, die mindestens eine dieser Beschränkungen verletzen und es bleiben nur noch neun Zustände übrig. Diese werden mit Zustandsübergängen versehen, welche nur jeweils eine Komponente schalten. Der resultierende Automat ist in Abbildung 4.5 zu sehen. “P” und “V” stehen dort für Pumpe und Ventil, subskript “w”, “ea” und “d” jeweils für Wasser, Essigsäureanhydrid und Abfluss. Negierte Aktoren sind ausgeschaltet/geschlossen, nicht negierte angeschaltet/geöffnet.
Bei den Übergängen zum Öffnen und Schließen des Abflussventils und dem Einschalten der Pumpen werden Events ausgelöst, auf die der in Abschnitt 4.6.4 erklärte Automat des Glasreaktors reagiert. Ebenfalls ist das Anschalten der Pumpen mittels Guards geschützt, um die in Abschnitt 4.6.4 genannten Beschränkungen an den Inhalt des Glasreaktors nicht zu verletzen.
Die Pumpen können und müssen zusätzlich auch rückwärts laufen, was einem dritten Zustand entspräche. Dies soll jedoch im normalen Experimentierablauf nicht durch die Studenten steuerbar sein und bildet einen Sonderfall, der in Abschnitt 4.6.5 besprochen wird.
Beim Betreten jedes dieser Zustände wird sich der in Abschnitt 4.4.1 und Abschnitt 4.4.2 beschriebenen Treiber zur Steuerung der Ventile und der Pumpen bedient, um die entsprechende Aktorstellung zu erzielen. In Zuständen, in denen die Pumpen aktiv sind, wird mittels der `serviceState()` Funktion wiederholt der in Abschnitt 4.5.1 eingeführte virtuelle Sensor des Glasreaktors aufgerufen, um zu messen, wie viel Flüssigkeit diesem beigeführt wurde.

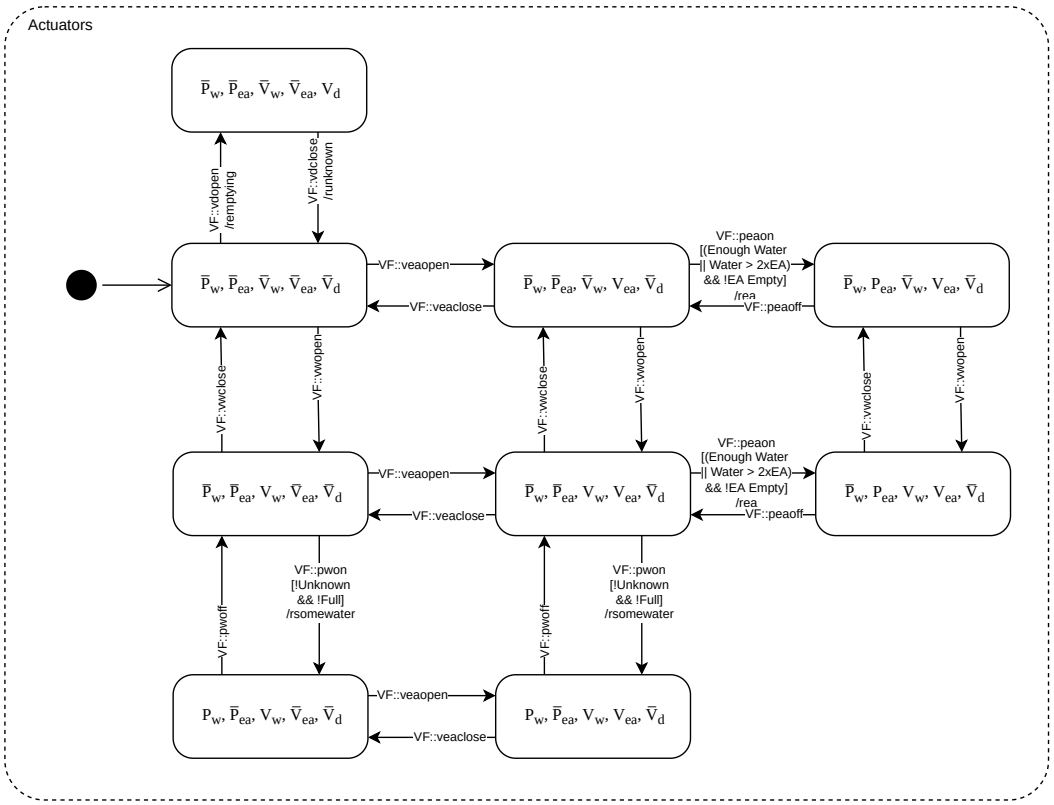

Abbildung 4.5.: Zustände der Pumpen und Ventile

4.6.2. Rührer und Licht

Da der Rührer und das Licht nicht in direkter Abhängigkeit zu den Pumpen und Ventilen stehen, können sie als orthogonale Automaten zu deren Automat dargestellt werden. Abbildung 4.6 zeigt die Zustandsautomaten des Rührers und des Lichts. Sie können jeweils nur angeschaltet oder ausgeschaltet sein und zwischen diesen Zuständen wechseln, wobei der Rührer einen zusätzlichen Zustandsübergang hat, welcher im angeschalteten Zustand beginnt und endet, da die Drehzahl des Rührers beim Betreten dieses Zustandes eingestellt wird. Somit ist es möglich, die Drehzahl zu ändern, während der Rührer schon angeschaltet ist.

4.6.3. Essigsäureanhydridbehälter

Da der Vorrat an Essigsäureanhydrid im Gegensatz zum Wasser nicht automatisch aufgefüllt wird und das Verhalten der anderen Komponenten von diesem Vorrat abhängt, muss ein Zustandsautomat für den Behälter des Essigsäureanhydrid erstellt werden. Dieser ist ebenfalls in Abbildung 4.6 enthalten. Der Automat bedient sich des in Abschnitt 4.5.2 beschriebenen virtuellen Sensors, um zu zeigen, ob noch genug Essigsäureanhydrid für die Durchführung eines Experiments zur Verfügung steht.
Beim Start der Anlage startet der Automat im Zustand “EA Empty”, da die Anlage den tatsächlichen Füllstand nicht wissen kann und vom worst-case ausgehen muss. Sobald ein Zustandsänderungssignal “eafill” empfangen wird, wechselt der Automat in den Zustand “EA Full”. Wenn mittels des virtuellen Sensors ein Füllstand erkannt wird, der nicht mehr für die Durchführung eines ganzen Experiments reicht, wird in den Zustand “EA Not Enough” gewechselt. Wenn dieser letzte Vorrat auch aufgebraucht ist, erfolgt ein Wechsel zu “EA Empty”. Mittels dieses Automaten können wie

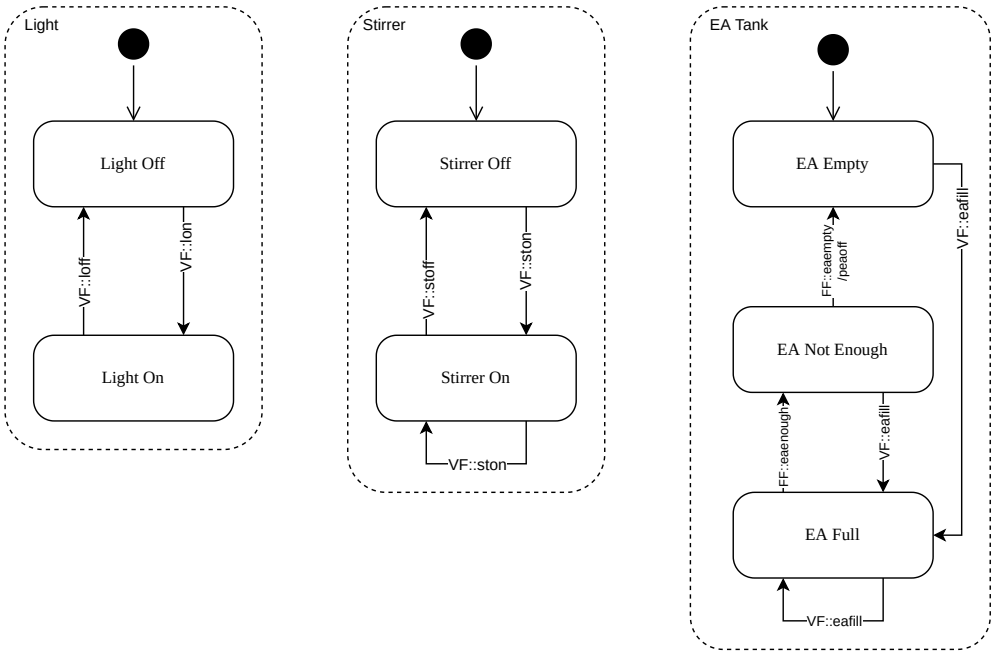

Abbildung 4.6.: Zustände des Lichts, des Rührers und des EA Vorratsbehälters

in Abschnitt 4.6.5 erläutert Entscheidungen getroffen werden, ob Experimente gestartet werden oder nicht. Somit wird Anforderung F5 erfüllt.

4.6.4. Glasreaktor

Der Zustand des Glasreaktors ist für die Sicherheit der Anlage von elementarer Bedeutung, da abhängig von ihm entschieden werden muss, ob das Pumpen von Flüssigkeiten erlaubt ist. Folgende aus den in Abschnitt 2.6.2 genannten Anforderungen S2 bis S4 abgeleiteten Beschränkungen für den Inhalt des Glasreaktors gibt es:

- Ein maximaler Füllstand von 200 Millilitern darf nicht überschritten werden.
- Es muss immer mindestens doppelt so viel Wasser wie Essigsäureanhydrid im Reaktor sein.
- Es muss Mindestmenge von 80 Millilitern an Wasser im Reaktor sein, bevor Essigsäureanhydrid hinzugefügt wird.
- Wenn der Füllstand des Reaktors oder das Verhältnis der Flüssigkeiten in diesem unbekannt sind, darf keine weitere Flüssigkeit hinzugefügt werden.

Die letzte Beschränkung leitet sich aus den ersten beiden ab, da die Anlage diese nicht garantiert einhalten kann, wenn der Inhalt des Glasreaktors unbekannt ist. Die erwähnten Volumina wurden in Vorversuchen der Abteilung Technische Chemie 2 erarbeitet und sind bei Bedarf in der in Abschnitt 4.8 erwähnten Konfigurationsdatei einstellbar.
Um diese Anforderungen zu erfüllen, wird ein weiterer Zustandsautomat erstellt, welcher den Zustand des Glasreaktors abbildet und in Abbildung 4.7 zu sehen ist. Vor der Aktivierung des Automaten muss sichergestellt sein, dass der Glasreaktor leer ist, da der Automat im Zustand “Empty” startet. Von diesem Zustand aus ist es nur erlaubt, durch Einfüllen von Wasser in den Zustand “Some Water” zu wechseln.
Der “Some Water” Zustand beschreibt einen nur mit Wasser gefüllten Glasreaktor, dessen Füllstand noch nicht hoch genug ist, um das Experiment durch Hinzufügen von Essigsäureanhydrid sicher

durchzuführen. Es gibt zwei Zustandsübergänge, die aus diesem Zustand herausführen. Durch das Öffnen des Abflussventils wird der Übergang in den Zustand “Emptying” ausgelöst. Dieser Übergang gilt auch für alle folgenden Zustände und wird im Folgenden nicht jedes Mal explizit erwähnt. Der andere Zustandsübergang führt in den Zustand “Enough Water”, sobald genug Wasser zur Durchführung des Experimentes eingefüllt wurde. Da erst in diesem Zustand das Einfüllen von Essigsäureanhydrid ermöglicht wird, ist hierdurch Anforderung S4 erfüllt.
Vom “Enough Water” Zustand wird in den “Full” Zustand gewechselt, falls weiterhin Wasser eingefüllt werden sollte, bis der Glasreaktor einen kritischen Füllstand erreicht. Bei diesem Übergang wird die Wasserpumpe durch das erzeugte Event automatisch ausgeschaltet. Im “Enough Water” Zustand ist jedoch auch das sichere Einfüllen von Essigsäureanhydrid möglich, woraufhin in den Zustand “Water > 2xEA” gewechselt wird.
“Water > 2xEA” sagt aus, dass sich mindestens das doppelte Volumen an Wasser im Vergleich zu Essigsäureanhydrid im Glasreaktor befindet. Von daher ist es erlaubt, beide Substanzen nach Belieben weiter einzufüllen. Sollte der Reaktor seinen kritischen Füllstand erreichen, wird wieder in “Full” gewechselt. Wenn so lange Essigsäureanhydrid eingefüllt wird, bis nicht mehr doppelt so viel Wasser vorhanden ist, wird in “Water = 2xEA” übergegangen und die EA Pumpe durch das erzeugte Event ausgeschaltet. Hierdurch wird Anforderung S3 erfüllt.
In “Water = 2xEA” ist es nicht mehr erlaubt Essigsäureanhydrid zu pumpen. Sollte der Reaktor voll sein, wird wieder zu “Full” gewechselt, durch das erneute Hinzufügen von Wasser zurück zu “Water > 2xEA”.
Beim Betreten des Zustandes “Full” werden die aktiven Pumpen ausgeschaltet. Durch die in Abschnitt 4.6.1 genannten Guards des Automaten für Pumpen und Ventile können diese dann auch nicht mehr eingeschaltet werden. Somit wird Anforderung S2 erfüllt.
Der Zustand “Emptying” wird durch das Öffnen des Abflussventils von jedem Zustand außer “Empty” erreicht. Im “Emptying” Zustand ist es nicht erlaubt, Wasser oder Essigsäureanhydrid zu pumpen. Sollte so lange in diesem Zustand verweilt werden, bis der Glasreaktor leer ist, wird in den Zustand “Empty” gewechselt. Wird das Abflussventil geschlossen, bevor der Glasreaktor leer ist, wird in den “Unknown” Zustand übergegangen.
Im Zustand “Unknown” ist es ebenfalls nicht erlaubt zu pumpen, da nach einmaligem Öffnen und Schließen des Abflussventils der Füllstand des Glasreaktors und das Verhältnis der Flüssigkeiten in ihm nicht mehr sicher bestimmt werden können. Somit ist der einzige erlaubte Zustandsübergang die Rückkehr zu “Emptying” mittels einer erneuten Öffnung des Abflussventils.
Die Übergänge zu “Enough Water”, “Water = 2xEA”, “Full”, “Empty” und von “Water = 2xEA” zu “Water > 2xEA” werden mittels des in Abschnitt 4.5.1 beschriebenen virtuellen Sensors in Kombination mit der in Abschnitt 4.6 erwähnten `serviceStates()` Funktion ausgelöst. Die übrigen Zustandsübergänge werden vom Automaten der Pumpen und Ventile ausgelöst, wie in Abschnitt 4.6.5 beschrieben wird.

4.6.5. Gesamtautomat

Um die Automaten dieser einzelnen Komponenten zusammenzuführen, wird ein großer Gesamtautomat erstellt, welcher im Anhang in Abbildung A.1 zu sehen ist. In diesem synchronisieren sich die orthogonalen Zustandsautomaten untereinander mittels der an den Zustandsübergängen ausgelösten Events und Guards, welche Übergänge verhindern, wenn sich ein anderer Automat in bestimmten Zuständen befindet. Ebenfalls werden einige der beschriebenen Unterautomaten an- und ausgeschaltet, da sie sich hierarchisch in einem anderen Zustand befinden.
Da die Anlage vor der letzendlichen Durchführung eines Experiments erst vorbereitet werden muss, gibt es noch zusätzliche Zustände. Konkret müssen die von den Behältern von Wasser und Essigsäureanhydrid zum Glasreaktor führenden Schläuche befüllt werden, damit während der Durchführung eines Experiments bei Aktivierung der Pumpen sofort die entsprechende Flüssigkeit in den Reaktor

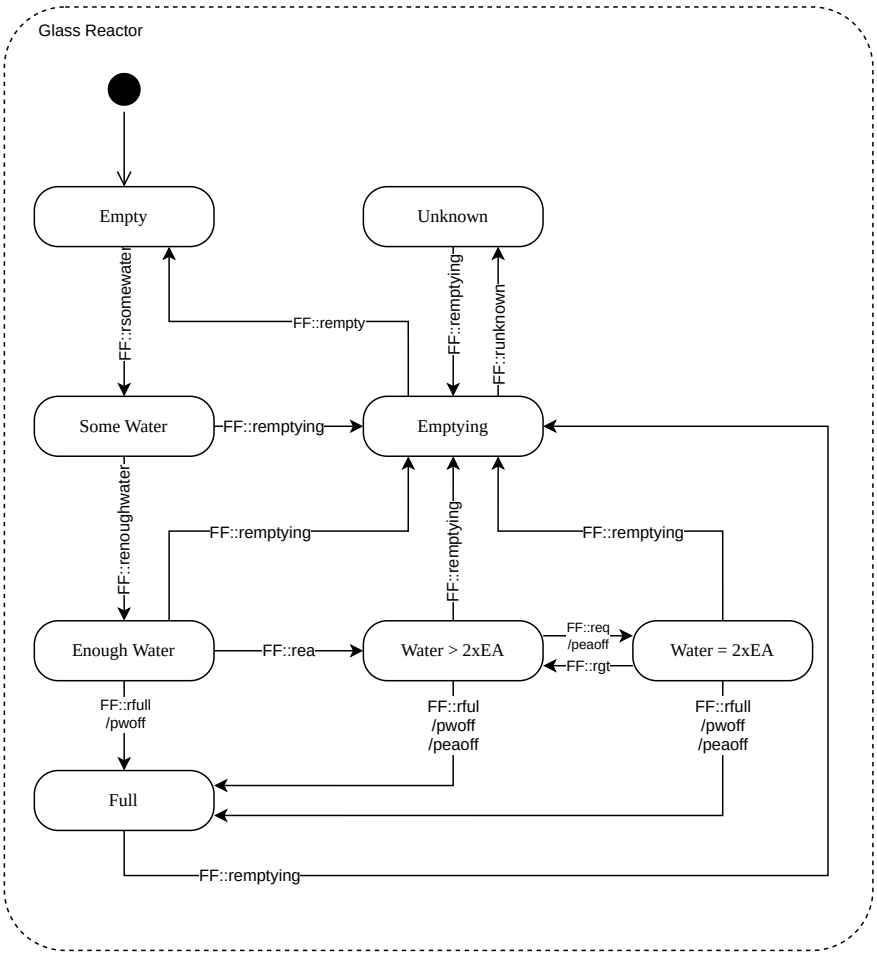

Abbildung 4.7.: Zustände des Reaktors

gepumpt und Anforderung F6 erfüllt wird. Nach Beendung eines Experiments müssen die Schläuche wieder entleert werden und sichergestellt sein, dass der Glasreaktor leer ist, da die Chemikalien nicht über einen längeren Zeitraum in der Anlage verbleiben dürfen. Dies entspricht Anforderung F7.
Somit ergibt sich ein “Experimentiermodus”, der eingenommen werden muss, bevor Experimente durchgeführt werden können und ein “Wartemodus”, in welchem die Anlage auch längere Zeit unbenutzt verweilen kann.
Da für die Steuerung des Lichts die Vorbereitungen nicht notwendig sind, ist dieser Unterautomat orthogonal zum “Mode” Automat angelegt, so dass das Licht im Experimentiermodus und im Wartemodus geschaltet werden kann. Ebenso verhält es sich mit dem Essigsäureanhydridbehälter, welcher in beiden Modi Verwendung findet: im Experimentiermodus wird er geleert, im Wartemodus kann er gefüllt werden. Die anderen beschriebenen Unterautomaten sind hierarchisch im “Experiment” Zustand des “Mode” Automaten angesiedelt. Sie werden erst nach dem Durchlaufen des im Folgenden beschriebenen Vorbereitungsvorgangs aktiviert.

Startvorgang

Der Startvorgang sorgt dafür, dass die Anlage in den “Wartemodus” überführt wird. Dafür durchläuft sie direkt nach dem erstmaligen Start, aber auch nach dem Beenden eines Experimentiervorgangs eine Reihe von Zuständen. Die Aktionen dabei werden in Abbildung 4.8 in einem Aktivitätsdiagramm dargestellt. Die Zustände können im Gesamtautomat in Abbildung A.1 nachvollzogen werden. Zu-

erst wird der “Initialising” Zustand eingenommen, in welchem die Pumpen rückwärts laufen, um die Schläuche zu leeren und das Abflussventil offen ist. Die Einnahme dieses Zustandes ist notwendig, da die Anlage nach dem Einschalten nicht wissen kann, ob die Schläuche gefüllt sind oder nicht und wie hoch der Füllstand des Glasreaktors ist.
Nach einer, in der in Abschnitt 4.8 erwähnten Konfigurationsdatei einstellbaren, Pumpdauer, nach der die Schläuche sicher geleert sind, wird in den “Draining” Zustand übergegangen. In diesem sind die Pumpen wieder ausgeschaltet, das Abflussventil ist aber noch offen, um den Glasreaktor zu leeren. Die Zeit, die die Anlage in diesem Zustand verbringt, ist variabel. Beim erstmaligen Betreten im Startvorgang muss davon ausgegangen werden, dass der Glasreaktor komplett gefüllt sein könnte, ohne dass die Anlage dies wüsste, beispielsweise nach einem Stromausfall während eines Experiments. Deswegen wird beim ersten Betreten des Zustandes eine Verweilzeit gewählt, welche ausreicht um einen komplett gefüllten Glasreaktor zu leeren. Bei nachfolgendem Betreten wird sich des in Abschnitt 4.5.1 beschriebenen virtuellen Sensors des Glasreaktors bedient, um eine Verweildauer zu erreichen, welche vom Füllstand des Reaktors abhängt. Eine konstante Verwendung der Maximalzeit wäre auch möglich gewesen. Da eine komplette Leerung des vollen Glasreaktors aber mehrere Minuten dauert, wurde dieser variable Ansatz gewählt, um die Wartezeit zwischen den an der Anlage durchgeführten Experimenten nicht unnötig in die Länge zu ziehen.

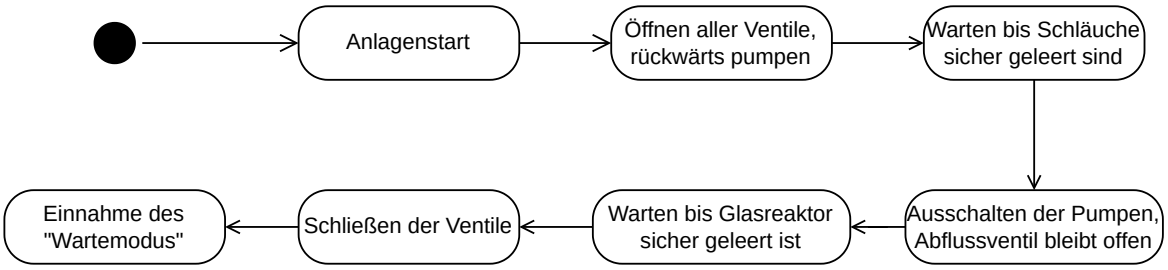

Abbildung 4.8.: Startvorgang der Anlage

Sobald die Anlage sicher sein kann, dass auch der Glasreaktor komplett geleert ist, geht sie in den “Idling” Zustand über. Dies ist der “Wartezustand”, in welchem sichergestellt ist, dass sich Flüssigkeiten weder in den Schläuchen, noch im Glasreaktor befinden. Ebenfalls sind alle Pumpen ausgeschaltet und Ventile geschlossen. Dies ist sehr wichtig, da die Ventile durch Aktivierung eines Elektromagneten geöffnet werden, welcher dauerhaft Strom verbraucht und Wärme erzeugt, welche bei Dauerbetrieb zum Versagen der Komponente führen könnte. Durch diesen Wartezustand werden die Anforderungen F7 und S6 erfüllt.

Vorbereitungsvorgang

Die Experimentieranlage AniTA verbleibt so lange im “Idling” Zustand, bis das Signal “getrdy” empfangen wird, welches die Vorbereitung für den “Experimentiermodus” anstößt. Nach einer Überprüfung, ob noch genug Essigsäureanhydrid zur Durchführung eines Experiments vorhanden ist, wird in den “Filling Tubes” Zustand gewechselt. In diesem pumpen beide Dosierpumpen gleichzeitig vorwärts, um die Schläuche zu füllen. Ebenfalls wird das Abflussventil geöffnet, um am Schlauchende austretende Flüssigkeiten direkt in den Abfluss zu leiten.
Da der Essigsäureanhydridschlauch kürzer als der Wasserschlauch ist, wird nach einer in der Konfiguration einstellbaren Zeit in den “WT Filling” Zustand gewechselt, sobald dieser voll ist. In diesem Zustand wird die Pumpe für Essigsäureanhydrid wieder ausgeschaltet, die Wasserpumpe läuft jedoch noch weiter. Nach einer weiteren einstellbaren Zeit wird noch kurz im “After Drain Emptying” Zustand verweilt, um sicherzugehen, dass der Glasreaktor nach dem Füllen der Schläuche wieder komplett leer ist.

Sobald dies geschehen ist, wird in den “Experiment” Zustand übergegangen. Während des Befüllens des EA-Schlauchs wird sich nicht des virtuellen Sensors für den Essigsäureanhydridbehälter bedient, da die zum Füllen des Schlauchs aus dem Behälter entnommene Flüssigkeit später beim Leeren des Schlauchs wieder in den Behälter geführt wird. Jedoch kann nicht verhindert werden, dass beim Befüllen etwas Essigsäureanhydrid in den Abfluss läuft, diese Menge ist aber vernachlässigbar klein. Beim Betreten des “Experiment” Zustandes sind durch die Vorbereitung die Schläuche gefüllt, der Glasreaktor entleert und die Pumpen, die Ventile und der Rührer ausgeschaltet. Hier werden jetzt durch den hierarchischen Aufbau des Gesamtautomaten die Unterautomaten für die Pumpen und Ventile, den Rührer und den Glasreaktor aktiviert, wodurch die Kontrolle über diese freigegeben und Experimente durchgeführt werden können.
Im Experimentiermodus kann auch die in den Vorversuchen der Abteilung Technische Chemie 2 erarbeitete Spülprozedur durchgeführt werden, womit Anforderung F3 aus Abschnitt 2.6.1 erfüllt ist.
Wenn das Experiment abgeschlossen ist kann durch das Event “backtoinit” wieder in den Wartemodus gewechselt werden, wobei der Startvorgang wieder durchlaufen wird, um Schläuche und Glasreaktor zu leeren.

4.7. Netzwerk

In diesem Abschnitt wird auf die Realisierung der für die Einbettung in das Rahmenprojekt notwendigen Kommunikation mit dem Virtualisierungsserver und die dafür benötigten Softwaremodule eingegangen.

4.7.1. WebSocket

Da die einzelnen Komponenten des Gesamtprojektes mittels des WebSocket-Protokolls miteinander kommunizieren, muss dieses, wie in Anforderung F2.1 gefordert, auch in der Firmware der Experimentieranlage AniTA implementiert werden.
Es gibt eine open-source Library, die das WebSocket-Protokoll für den Atmega2560 bereitstellt[10], jedoch wurde sie zuletzt 2018 aktualisiert und funktioniert nicht in Kombination mit dem Controllino MEGA.
Es wird ein Fork der Library erstellt und die Probleme behoben, um die Kompatibilität mit dem Controllino wiederherzustellen.
Da diese modifizierte WebSocket Library von der nachfolgend in Abschnitt 4.7.2 beschriebenen Network Library benutzt wird und auch von zukünftigen Experimentieranlagen verwendet werden kann, wird sie in den in Abschnitt 4.2 beschriebenen Repositories des Projektes gehostet, um die in Abschnitt 2.6.3 gestellte Anforderung P2 zu erfüllen.

4.7.2. Network Library

Zur Implementierung des in Abschnitt 3.3.3 beschriebenen, von der Projektgruppe entworfenen Netzwerkprotokolls wird eine Library namens “Network” geschrieben, welche auch auf zukünftigen Experimentieranlagen, welche einen Controllino zur Steuerung verwenden, eingesetzt werden kann. Hierfür befindet sie sich im in Abschnitt 4.2 beschriebenen Repository des Projektes und erfüllt somit in Kombination mit der WebSocket Library die in Abschnitt 2.6.3 gestellte Anforderung P2.
Die Funktionalitäten der Library motivieren sich aus der in Abschnitt 2.6.1 genannten Anforderung F2 und umfassen das Empfangen und Versenden von Nachrichten, die dem GECP entsprechen.

[10] `https://github.com/Links2004/arduinoWebSockets/tree/ATmega`

Hierfür muss eine Anlage ebenfalls die in Abschnitt 4.7.1 genannte WebSocket Library benutzen. Zusätzlich müssen vom Nutzer zwei Funktionen bereitgestellt werden, die von der Library als Callback aufgerufen werden.
Zur korrekten Nutzung der Library sollte die gestellte `networkInit` Funktion beim Start der Anlage aufgerufen werden, beispielsweise in der vom Arduino Framework bereitgestellten `setup()` Routine. Als Parameter müssen der `networkInit` Funktion eine IP-Adresse, eine MAC-Adresse, ein Port und zwei Funktionspointer auf die Callbackfunktionen übergeben werden. Daraufhin initialisiert die Funktion die Ethernet-Schnittstelle mit der gegebenen MAC- und IP-Adresse und startet einen WebSocket Server auf dem gegebenen Port. Im Falle der AniTA sind die genannten Parameter über die in Abschnitt 4.8 beschriebene Konfigurationsdatei einstellbar.
Die `networkService()` Funktion muss vom Nutzer periodisch aufgerufen werden, beispielsweise mittels der vom Arduino Framework bereitgestellten `loop()` Funktion.

Callbackfunktionen

Die Funktionen der Funktionspointer, welche in der `networkInit` Funktion übergeben werden, müssen die folgenden Signaturen haben, wobei die Namen der Funktionen und Parameter frei gewählt werden können:

- `void signalCB(uint8_t client, char *sig, char *params)`
- `struct State_response_s* createStateResponse(void)`

Diese Funktionen müssen vom Nutzer der Library implementiert werden, wobei die `signalCB` Funktion für das Weiterverarbeiten der entgegengenommenen Signale und eventueller Parameter, und die `createStateResponse()` für die Erstellung einer State Response, welche der im Header der Library definierten Form eines `State_response_s` Structs entspricht, verantwortlich ist.

Senden

Es werden drei Funktionen zum Versenden von Nachrichten zur Verfügung gestellt:

- `sendSignal(uint8_t client, const char *sig, const char *params)`
- `broadcastSignal(const char *sig, const char *params)`
- `sendStateResp(uint8_t client, struct State_response_s *response)`

Die Funktionen zum Verschicken von Signalen erwarten einen konstanten Pointer auf das zu versendende Signal als terminierten C-String. Dieser wird von der Funktion in eine Nachricht, welche dem GECP entspricht, eingesetzt und entweder an den spezifizierten Client oder, im Falle des Broadcasts, an alle verbundenen Clients gesendet. In Abbildung 4.9 wird der zum Versenden eines gewünschten Signals an alle Clients zugehörige Funktionsaufruf gezeigt. Ebenfalls ist der tatsächliche, über WebSocket verschickte Text zu sehen. Optional können auch Parameter mit einem Signal verschickt werden. Hierbei ist jedoch besondere Vorsicht geboten, da der im Funktionsparameter “params” übergebene String in der Nachricht direkt hinter das Feld “Parameters:” angehängt wird. Somit muss vom Benutzer der Network Library auf die korrekte Formatierung des Strings geachtet werden, um GECP-Konformität zu erreichen. Ein Beispiel hierfür wird in Abbildung 4.10 gezeigt: Um die gewünschte Nachricht zu erhalten, werden die benötigten Newline- und Leerzeichen direkt im String des GECP-Parameters der Funktion übergeben.

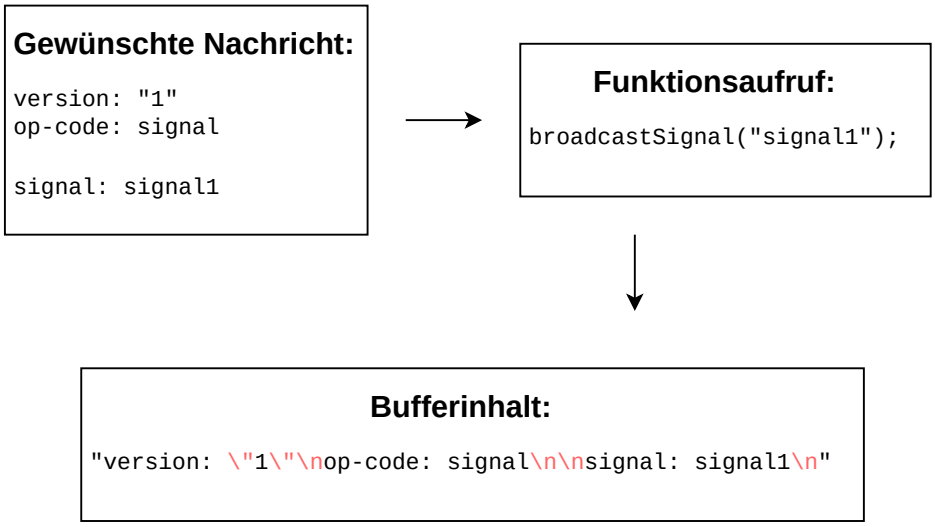

Abbildung 4.9.: Broadcast eines Signals ohne Parameter

Empfangen

Bei jedem Aufruf von `networkService()` wird von der WebSocket Library auf neue, empfangene Nachrichten geprüft. Falls eine Nachricht empfangen wurde, wird die Payload der WebSocket-Nachricht an die Network Library-interne Funktion `extractSignal` weitergeleitet. Diese analysiert die Nachricht und überprüft den eventuell vorhandenen OP-Code. Ist dieser "state-req", wird über die `createStateResponse()` Callbackfunktion eine State Response erstellt und zurückgesendet. Ist der OP-Code "signal", wird das Signal extrahiert und zusammen mit eventuellen Parametern an die Signalverarbeitende Callbackfunktion geschickt.

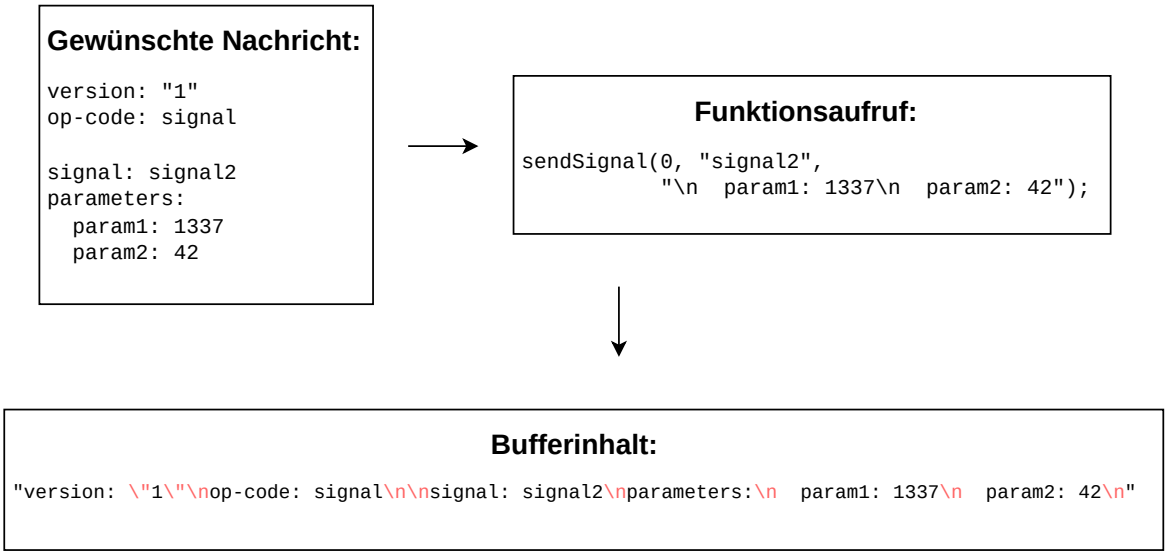

Abbildung 4.10.: Senden eines Signals mit Parameter

4.7.3. Kommunikation mit dem EFaViS

In diesem Unterabschnitt wird beschrieben, welche Nachrichten von der Experimentieranlage AniTA zum EFaViS gesendet werden und welche Nachrichten AniTA vom EFaViS erwartet.

Events

Es kann jedes der im Anhang in Tabelle A.2 als "VF" markierten Signale in der im GECP definierten Form an die AniTA geschickt werden. Die anderen Signale werden nur anlagenintern ausgelöst. Wenn Signale zu einer Änderung im Zustandsautomaten der Anlage führen, antwortet die Anlage mit einem Broadcast eines entsprechenden akzeptierenden Signals aus Tabelle A.2 an alle verbundenen Clients. Zusammen mit diesem aktzeptierenden Signal werden auch potentielle, gültige Parameter verschickt.
Welche Signale zu welchem Zustandsübergang führen, kann anhand des Gesamtautomaten in Abbildung A.1 und den Ausführungen in Abschnitt 4.6 nachvollzogen werden.
Sollte ein Signal geschickt werden, welches existiert, aber momentan zu keiner Änderung im Zustandsautomat führt oder ungültige Parameter enthält, wird ein entsprechendes, ablehnendes Signal aus Tabelle A.2 mit eventuellen, ungültigen Parametern an den Client gesendet.
Sollte ein Signal geschickt werden, welches nicht existiert oder die Nachricht eine ungültige Form haben, ignoriert die Anlage diese.
Sobald eine Zustandsänderung vollzogen wurde, wird ein Bestätigungssignal, welches dem Signal das die Zustandsänderung ausgelöst hat angehört, an alle verbundenen Clients gesendet.

Daten

Messdaten und andere für die Anlage wichtige Werte werden mittels des `valueupdate` Signals verschickt. Hierfür wird von der Experimentieranlage AniTA bei Bedarf ein entsprechendes Signal an alle verbundenen Clients gesendet, welches die Daten als Parameter enthält. Eine Übersicht über die Daten, die gesendet werden und die zugehörigen Parameternamen bietet Tabelle 4.4.
Die Daten werden in drei Gruppen unterteilt: Rührer, Temperatur und Reaktor, welche jeweils in einem der Länge nach konfigurierbaren Intervall ausgelesen und gemeinsam verschickt werden. Es werden nur Daten versendet, wenn sie sich seit der letzten Nachricht verändert haben.

State Response

Die State Response wird nur nach einem, wie in Abschnitt 3.3.3 beschriebenen, eingegangenen State Request verschickt und enthält neben einer Kodierung des Zustandes der Anlage auch alle Parameter aus Tabelle 4.4, die dem Signal `valueupdate` zugehörig sind.
Die Zustände des Gesamtautomaten werden kodiert, indem die Zustände einzelner Unterautomaten durchnummeriert werden. Zustände von Automaten, die orthogonal zueinander sind, werden, durch Schrägstriche voneinander getrennt, eingeklammert. So beschreibt die beispielhafte Kodierung `(2/4)`, dass sich der erste Automat im Zustand Nummer zwei befindet, der zweite gleichzeitig in Zustand vier. Zustände, in denen sich hierarchisch weitere Zustandsautomaten befinden, öffnen weitere Klammern, um diese abzubilden. So sagt `(3(0/0)/4)` aus, dass sich im Zustand drei des ersten Automaten weitere hierarchische Automaten befinden, die beide in Zustand null sind.
Nach diesem Prinzip kann jede eindeutige Kombination der orthogonalen und hierarchischen Zustände des Gesamtautomaten der Anlage, welcher im Anhang in Abbildung A.1 zu sehen ist, kodiert werden. Die Kodierung der atomaren Zustände kann im Anhang in Tabelle A.1 eingesehen werden.

Parameter	Einheit	Zugehöriges Signal	Anmerkung
`stirError`	String	`valueupdate`	Enthält den ausgelesenen Fehlerspeicher des Rührers
`stirActualRPM`	min^{-1}	`valueupdate`	Die tatsächlichen, momentanen RPM des Rührers
`stirSetRPM`	min^{-1}	`valueupdate`	Soll-RPM des Rührers
`stirTorque`	Nmm	`valueupdate`	Drehmoment des Rührers
`temperature`	°C	`valueupdate`	Gemessene Temperatur im Glasreaktor
`reactorLevel`	ml	`valueupdate`	Füllstand des Glasreaktors
`reactorWaterLevel`	ml	`valueupdate`	Volumen von Wasser im Glasreaktor
`reactorEaLevel`	ml	`valueupdate`	Volumen von Essigsäureanhydrid im Glasreaktor
`eaTankLevel`	ml	`valueupdate`	Verbleibendes Volumen im Essigsäureanhydridbehälter
`stirRPM`	min^{-1}	`ston`	Gewünschte RPM des Rührers

Tabelle 4.4.: Parameter und zugehörige Signale

4.8. Konfiguration

Um die Firmware auf die tatsächlichen Begebenheiten der Anlage abzustimmen, müssen Anlagenparameter eingestellt werden. Um diese zentral zu verwalten und die in Abschnitt 2.6.1 gestellte Anforderung F8 zu erfüllen, wird die `config.h` Datei erstellt. Hier werden die Anlagenparameter in Form von Präprozessor Definitionen eingestellt und dann in den restlichen Teilen der Firmware verwendet. Eine ausführliche Übersicht über alle einstellbaren Parameter mit ihrer Beschreibung ist im Anhang in Tabelle A.4 gegeben.

4.9. Test und Optimierung

Um die Entwicklung der Firmware der Experimentieranlage zu vereinfachen und die korrekte Funktionsweise dieser zu verifizieren, wird die über den USB Port des Controllino erreichbare Serielle Schnittstelle verwendet, um Diagnosenachrichten an einen angeschlossenen Computer zu senden. Ebenfalls wird eine Webapplikation erstellt, mit welcher die Kommunikation über Ethernet getestet werden kann.

4.9.1. Debug-Modus

Die Serielle Schnittstelle wird nur aktiviert, wenn der Debug-Modus in der Konfiguration angeschaltet wurde (siehe Tabelle A.4). Über sie werden beim Start der Anlage die Version, der Build-Zeitpunkt und ihre IP Adresse ausgegeben. Ebenfalls wird im Debug-Modus die Funktion der aktiven Komponenten getestet, indem zuerst alle Ventile geöffnet werden, dann die Pumpen eine Sekunde vorwärts und eine Sekunde rückwärts laufen und schließlich die Ventile wieder geschlossen werden. Die korrekte Funktionsweise muss hierbei vom Betreiber beobachtet werden, da die Anlage keine Möglichkeit hat, diese zu verifizieren. Die Funktion des Rührers wird überprüft, indem sein Feh-

lerspeicher ausgelesen und über die serielle Schnittstelle an den Debug-Computer gesendet wird. Bei korrekter Funktionsweise wird `FLT: No Error!` ausgegeben. Sollte keine Verbindung vom Controllino zum Rührer hergestellt werden können, wird `can't connect to stirrer!` ausgegeben. Das Temperaturmodul wird überprüft, indem die Temperatur einmal ausgelesen und ausgegeben wird. Hierbei sollte ein realistischer Wert ausgegeben werden. Wenn keine Verbindung zum Temperaturmodul hergestellt werden kann, wird immer `0.00` ausgegeben.

4.9.2. Webapplikation

Die Webapplikation zum Testen der Experimentieranlage AniTA basiert auf der im Zuge von Kersten Blümels Arbeit zur Experimentieranlage VerA erstellten Webapplikation. Sie stellt eine WebSocket Verbindung zu der Anlage her und benutzt dann die in Tabelle A.2 gelisteten Signale, um GECP-konforme Nachrichten an die Anlage zu senden.
In der im Anhang in Abbildung A.3 gezeigten Benutzeroberfläche werden die Schaltflächen der oberen Leiste benutzt, um die Verbindung zur Anlage herzustellen, zu beenden oder einen State-Request zu senden. Mit den Schaltflächen an der linken Seite können alle Signale, die von der Virtualisierung an die Anlage geschickt werden dürfen, gesendet werden.
Die Ausgabefenster sind zusätzlich in Abbildung 4.11 zu sehen. Das linke der drei Ausgabefenster gibt dabei empfangene Antwortsignale der Anlage aus, ebenfalls werden dort die bei anlageninternen Zustandsübergängen versendeten Signale angezeigt. So sind die Zustandsübergänge der Anlage nachvollziehbar. Im mittleren Fenster werden Signale des Typs `valueupdate` ausgegeben, mit welchen die Anlage, wie in Abschnitt 4.7.3 beschrieben, Messdaten und andere wichtige Werte verschickt. Das rechte Fenster gibt die von der Anlage versendeten State-Responses aus.

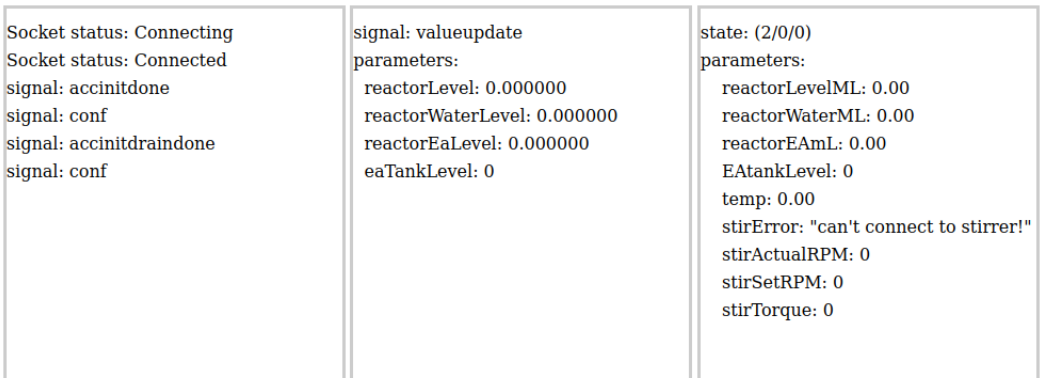

Abbildung 4.11.: Ausgabefenster der Webapplikation

5. Inbetriebnahme

Für den Fall, dass die Firmware der Anlage modifiziert werden soll oder das Projekt auf einen neuen Controllino programmiert werden muss, ist es notwendig, das Projekt neu aufzusetzen. Dieser Vorgang und die erstmalige Inbetriebnahme der Anlage werden in diesem Kapitel beschrieben.

5.0.1. Projekt aufsetzen

Dank der in Abschnitt 4.2 beschriebenen Strukturierung des Projekts ist es leicht möglich, dieses erneut aufzusetzen.
Nach einer Installation des Command Line Tools von PlatformIO (PlatformIO Core) [1] muss das im GitLab der Abteilung Software Engineering der Universität Oldenburg liegende Repository "anita"[2] rekursiv geklont werden.
In dem Repository ist eine Makefile gegeben, welche die durch PlatformIO gegebenen Kommandos vereinfacht. Bei Eingabe von `make build` kompiliert das Projekt und die Meldung `SUCCESS` wird zurückgegeben. Hierbei werden benötigte Libraries automatisch heruntergeladen, wobei es dazu kommen kann, dass die eigenen Anmeldedaten für das GitLab der Universität Oldenburg eingegeben werden müssen.

5.0.2. Konfigurieren

Zum Anpassen der Einstellungen der Anlage wird wie in Abschnitt 4.8 beschrieben die im `src/` Ordner zu findende Datei `config.h` verwendet. Eine Beschreibung der einzelnen einstellbaren Optionen und ihrer zum Zeitpunkt der Abgabe dieser Arbeit verwendete Standardbelegung ist in Tabelle A.4 einzusehen.
Vor der Inbetriebnahme im Produktivbetrieb sollten die Konfigurationsoptionen für Debug-Modi deaktiviert werden, da diese sehr viel Arbeitsspeicher in Anspruch nehmen (siehe Abschnitt 6.4).

5.0.3. Programmierung

Zum Flashen der Firmware auf den Controllino muss dieser mittels der USB-B Buchse, welche im linken Paneel an der Vorderseite der Experimentieranlage AniTA liegt, mit dem Computer, auf welchem das Projekt liegt, verbunden werden.
Die Programmierung erfolgt durch Eingabe des Befehls `make upload`. Sollte das Projekt seit dem letzten Kompiliervorgang verändert worden sein, wird es automatisch neu kompiliert.
Bei erfolgreicher Programmierung öffnet die Anlage sofort alle Ventile, inklusive des Ablassventils und fängt an, rückwärts zu pumpen. Dies ist der in Abschnitt 4.6.5 beschriebene Startvorgang. Entsprechende Sicherheitsvorkehrungen sollten vorher getroffen worden sein.

[1] `https://platformio.org/install/cli`
[2] `https://gitlab.uni-oldenburg.de/se/projects/experiment-system/anita`

6. Evaluation

Um die Funktionsweise der Anlage zu evaluieren, werden die in Abschnitt 4.9 beschriebenen Tools verwendet.

6.1. Antwortzeiten

Um die Antwortzeiten der Anlage zu evaluieren, wird das im Webbrowser Firefox eingebaute Netzwerkanalysetool genutzt, welches in Abbildung 6.1 zu sehen ist. Damit ist es möglich, die einzelnen WebSocket-Nachrichten, ihren Inhalt und den genauen Zeitpunkt ihres Eintreffens auf dem zum Evaluieren benutzten Computer zu ermitteln.

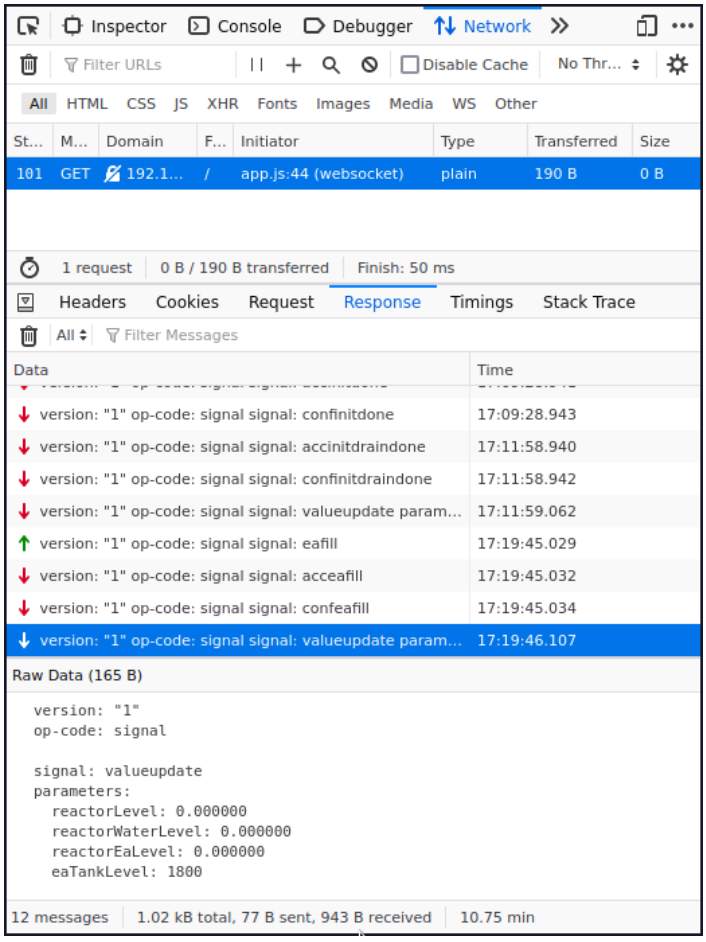

Abbildung 6.1.: Netzwerkanalysetool des Firefox Webbrowsers

Damit konnte in Verbindung mit der in Abschnitt 4.9.2 beschriebenen Webapplikation gemessen werden, wie lange die Experimentieranlage braucht, um auf bestimmte Anfragen zu reagieren und Antworten zurückzusenden. Das Ergebnis dieser Messungen ist in Tabelle 6.1 zu sehen. In dieser Tabelle beschreibt “Zeit bis Accept” die Zeit in Millisekunden, die verstrichen ist, bis nach einem Signal an die Experimentieranlage ein akzeptierendes Signal, welches bestätigt, dass der gewünschte Zustandsübergang genommen werden kann, zurückgeschickt wurde. “Zeit bis Confirm” bezieht sich

auf die Differenz des ursprünglichen Signals und der Bestätigung, dass der Zustandswechsel vollzogen wurde. Somit ist dies die Gesamtzeit, die es tatsächlich brauchte, um die gewünschte Aktion durchzuführen.
Bis auf einen einmaligen Ausreißer beim Signal `lon`, welcher nicht reproduzierbar war, schickte die Anlage die Bestätigung, dass ein Zustandswechsel durchgeführt werden kann, immer innerhalb von 4 Millisekunden ab. Die Zeit bis zu einem vollzogenen Zustandswechsel beträgt in den Fällen, in denen nur GPIO Pins des Controllino geschaltet oder interne Berechnungen durchgeführt werden müssen meist 5 Millisekunden. Ausnahme war in diesen Messungen das erstmalige Einschalten der Wasserpumpe. Bevor in diesem Fall die Bestätigung des Zustandswechsels versendet wird, werden noch Signale, die über den geänderten Zustand des Glasreaktors informieren, verschickt, was die Gesamtzeit auf 9 Millisekunden angehoben hat. Bei Signalen, die eine Kommunikation mit dem Rührer zur Folge haben, erhöht sich die Gesamtdauer des Zustandswechsels auf 24 bis 59 Millisekunden. Dies liegt an dem erforderlichen Nachrichtenaustausch zwischen Controllino und Rührer, welcher mittels serieller Schnittstelle und einer Symbolrate von 9600 Baud stattfindet.
Darüber hinaus wurde die Zeit gemessen, die die Anlage braucht, um auf einen State-Request mit einer State-Response zu antworten. Hierbei wurden Zeiten von 368 bis 399 Millisekunden gemessen, was darauf zurückzuführen ist, dass mit einer State-Response auch alle der Anlage bekannten Messwerte verschickt werden. Im Zuge dessen wird viermal mit dem Rührer kommuniziert, was die oben beschriebenen Verzögerungen mit sich bringt. Ebenfalls wird das Temperaturmodul ausgelesen, welches wie in Abschnitt 4.4.5 beschrieben über ModbusRTU ausgelesen wird, was einige Zeit in Anspruch nimmt.
Da State-Responses jedoch laut der Dokumentation des Gesamtprojektes nur verwendet werden, um beim initialen Verbinden eines Clients mit einer Anlage den Zustand derer abzufragen[1], sollte diese erhöhte Antwortzeit im normalen Experimentierablauf der Studenten keine Behinderung darstellen.

Signal	**⌀Zeit bis Accept (ms)**	**⌀Zeit bis Confirm (ms)**
`eafill`	3	5
`lon`	10	11
`loff`	3	4,5
`getrdy`	3	5
`ston`	3	28
`stoff`	3,67	45,67
`vwopen`	3	5
`vwclose`	3,33	5
`pwon`	3,5	7
`pwoff`	4	5
`backtoinit`	4	59

Tabelle 6.1.: Messungen der Antwortzeiten auf bestimmte Signale

[1] [2], Seite 24

6.2. Sicherheitskritische Füllstände des Glasreaktors

Beim Evaluieren der Anlage ist es einmalig dazu gekommen, dass nach einem kompletten Befüllvorgang des Glasreaktors und anschließendem, automatischen Abschalten der Wasserpumpe der berechnete Füllstand des Glasreaktors über den festgelegten Maximalwert von 200 Millilitern gestiegen ist. Es wurde die Vermutung aufgestellt, dass dies mit der Kommunikation mit dem Rührer oder dem Temperaturmodul zusammenhängt. Wie in Abschnitt 6.1 gezeigt, führt dies zu größeren Verzögerungen, in denen der normale Programmfluss der Firmware blockiert wird.
Um diese Vermutung zu überprüfen, wurden mehrere Befüllvorgänge durchgeführt, gegen deren Ende eine große Anzahl State-Requests an die Anlage gesendet wurden. So konnte die Anlage tatsächlich durchweg dazu gebracht werden den Maximalfüllstand zu überschreiten. Die Ergebnisse dieser Versuche sind in Tabelle 6.2 zu sehen.

Versuch	Höchstfüllstand
1	200,74
2	200,29
3	200,34
4	200,31
5	200,37

Tabelle 6.2.: Messungen der berechneten Höchstfüllstände in Millilitern bei State-Request Spam

Jedoch wird dies im normalen Betrieb der Anlage nur sehr selten auftreten, da hierfür genau im Moment des Erreichens des Maximalfüllstandes eine Kommunikation über eine serielle Schnittstelle stattfinden muss, welche mit Standardeinstellungen im Falle des Rührers alle fünf und im Falle der Temperatur alle zehn Sekunden geschehen.
Ebenfalls gibt es mehrere Gründe, weshalb das Überschreiten des Maximalfüllstandes nicht sicherheitskritisch ist. Zum einen wird zwischen mehreren Netzwerkkommunikationen jedesmal der Zustandsautomat behandelt, was verhindert, dass die Anlage durch wiederholtes Senden von State-Requests nur noch mit diesen beschäftigt ist. Damit können die Pumpen zeitnah ausgeschaltet werden. Dies ist auch in den oben gezeigten Daten zu sehen, wo der maximale Füllstand der Glasreaktors 200,74 Milliliter betrug, obwohl so viele State-Requests gesendet wurden, dass die Anlage mit dem Bearbeiten dieser mehrere Sekunden verbrachte.
Somit gibt es einen tatsächlichen maximalen Füllstand von unter 201 Millilitern, was noch weit unter einem tatsächlich kritischen Füllstand liegt, da der maximal erlaubte Füllstand von 200 Millilitern mit sehr viel Sicherheitsspielraum gewählt wurde.
Zusätzlich liegt dieser Bruchteil eines Milliliters auch weit unter den Schwankungen, die durch das Befüllen der Schläuche oder die Reste im Ablassschlauch beim Entleeren des Glasreaktors entstehen.

6.3. Befüll- und Entleervorgang der Schläuche

Bevor Experimente an der Anlage durchgeführt werden können, müssen die Schläuche wie in Abschnitt 4.6.5 beschrieben, befüllt werden. Da jedoch immer Luftblasen beim Befüllen im Schlauch eingeschlossen werden, ist die Zeit, die es dauert bis ein Schlauch komplett gefüllt ist, sehr variabel. Es ist erforderlich, dass so lange gepumpt wird, bis die großen Luftblasen im Schlauch komplett rausgepumpt wurden, da sie sonst im Experimentierbetrieb die Ergebnisse eines Versuchs unerwartet verändern könnten, wenn ein Student beispielsweise einen kurzen Stoß Essigsäureanhydrid in den

Glasreaktor pumpen möchte, tatsächlich aber nur die Luftblase aus dem Schlauch austritt. Daher war es notwendig, die Befüllzeit der Schläuche ausreichend groß anzusetzen.
Die Konsequenz dessen ist, dass der Wasserschlauch noch durchschnittlich fünf Sekunden weiter befüllt wird, nachdem erste Tropfen am Schlauchende austreten. Für den Essigsäureanhydridschlauch beträgt diese Zeit zehn Sekunden. Da das Befüllen und Entleeren des Essigsäureanhydridschlauchs sich nicht auf den berechneten Füllstand des Essigsäureanhydridbehälters auswirkt, gehen bei jedem Befüll- und Entleervorgang unweigerlich einige Milliliter Essigsäureanhydrid verloren, da sie in den Abfluss fließen und nicht später wieder zurück in den Behälter gepumpt werden.
Ein gegenteiliges Problem konnte ebenfalls festgestellt werden: Beim Entleeren der Schläuche muss nicht darauf geachtet werden, die Pumpdauer zu beschränken, da die Flüssigkeiten zurück in ihre Behälter gepumpt werden. Jedoch ist es trotz dieser wegfallenden Beschränkung nicht möglich, die Schläuche komplett zu leeren. Insbesondere in der in Abbildung 2.2 zu sehenden Schlinge des Essigsäureanhydridschlauchs, direkt hinter der Pumpe, sammelt sich Flüssigkeit, die nicht vollständig rausgepumpt werden kann.

6.4. Netzwerknachrichtenlänge

Da der Arbeitsspeicher des Controllino MEGA sehr begrenzt ist, wurde eine Evaluation durchgeführt, um das Verhalten der Anlage bei unterschiedlichen Längen der über WebSocket ankommenden Nachrichten zu bestimmen.
Softwareseitig ist die Länge der Nachrichten durch die WebSocket-Bibliothek auf 1024 Byte begrenzt. Wenn größere Nachrichten das Steuergerät erreichen, werden diese nicht verarbeitet und die WebSocket-Verbindung getrennt.
Eine weitere Herausforderung stellt die dynamische Speicherallokation dar: Eingehende Nachrichten sowie Informationen zu den verbundenen Clients werden von der WebSocket-Bibliothek auf den Heap gelegt. Die Nachrichten belegen den dortigen Speicher, bis sie verarbeitet wurden. Die Client-Informationen liegen auf dem Heap bis die Verbindung zum Client unterbrochen wird. Sollte zu wenig freier Speicher vorhanden sein, um eine komplette Nachricht zu speichern, wird diese wie beim Überschreiten der Maximallänge nicht verarbeitet und die Verbindung abgebrochen.
Um unerwartete Verbindungsabbrüche im Produktivbetrieb zu vermeiden, sollte beim Kompilieren darauf geachtet werden, dass mindestens 2 KiB RAM für den dynamischen Gebrauch von Heap und Stack frei bleiben. In der zum Zeitpunkt der Abgabe dieser Arbeit verwendeten Firmware ist dies berücksichtigt (siehe Tabelle 6.3). Ebenfalls sollten keine Nachrichten an die Anlage geschickt werden, die die oben genannte Länge von 1024 Byte überschreiten.

Speicherart	Benutzter Speicher	Gesamter Speicher	Benutzung in Prozent
RAM	5671 Byte	8192 Byte	69,2%
Flash	73912 Byte	253952 Byte	29,1%

Tabelle 6.3.: Speicherbedarf der installierten Firmware

6.5. Langzeittest

Die Experimentieranlage befand sich vom 16.02.2023 bis zum 09.03.2023 unangetastet und angeschaltet drei Wochen im Wartemodus, woraufhin vor dem erneuten Programmieren mit einer neuen Firmware mit kleinen Veränderungen noch ein Funktionstest durchgeführt wurde, bei dem kein abweichendes Verhalten festgestellt werden konnte.

Größte potenzielle Fehlerquelle bei einer Langzeitnutzung ist die `millis()` Funktion des Arduino-Frameworks. Diese gibt die Zeit seit dem Start des Steuergerätes in Millisekunden als 32 Bit unsigned Integer zurück, welcher nach ungefähr 50 Tagen überläuft. Um dieses Risiko direkt zu mitigieren, wurde bei jeder Verwendung von `millis()` in der Firmware darauf geachtet, dass die durchgeführten Berechnungen Overflow-sicher sind.

7. Zusammenfassung und Ausblick

In dieser Bachelorarbeit wurde die Firmware eines Steuergerätes einer Experimentieranlage aus dem Bereich der technischen Chemie entworfen und implementiert. Ziel des größeren Projektes, in das diese Arbeit eingebettet war, ist es, chemische Experimente für Studenten aus dem Internet durchführbar zu machen.
Im Rahmen dessen wurden für diese Arbeit drei Aufgabenblöcke identifiziert:

- Die Ansteuerung der technischen Komponenten der Experimentieranlage
- Die Gewährleistung der sicheren und korrekten Schaltstellungen dieser Komponenten
- Die Eingliederung in das Rahmenprojekt

Die Experimentieranlage verfügt über eine Fülle unterschiedlicher technischer Komponenten, die zur Durchführung der Experimente notwendig sind. Um diese vom Steuergerät kontrollieren zu können, mussten für jede Gruppe an Komponenten eigene Treiber geschrieben werden, die jeweils über deren Schnittstellen mit den Komponenten kommunizieren und ihren Handlungsspielraum dem Rest der Firmware zur Verfügung stellen. Ebenfalls wurde Software geschrieben, die wichtige, nicht auslesbare Größen wie die Füllstände in der Anlage berechnet.
Die sichere und korrekte Durchführung von Experimenten an der Anlage wurde durch die Benutzung von endlichen Zustandsautomaten gewährleistet. Hierbei wurde ein Zustandsautomat erstellt, der nur Zustände sicherer Kombinationen an Schaltstellungen der technischen Komponenten enthält. Der Automat bedient sich der zuvor genannten Treiber, um eine reale, seinem Zustand entsprechende Schaltstellung zu erreichen. Ebenfalls werden die berechneten Größen berücksichtigt, um die Anlage in einem sicheren Zustand zu halten und Komponenten gegebenenfalls aus- oder umzuschalten.
Für die Einbettung in das Rahmenprojekt wurde Netzwerkcode geschrieben und ein eigens im Projekt entwickeltes Protokoll implementiert. Mit diesem ist es möglich, den Zustandsautomaten der Experimentieranlage über das Internet zu steuern.
Besondere Herausforderung in dieser Arbeit war die Entwicklung und Umsetzung des Zustandsautomaten, der viele Iterationen durchlief, bis eine Version erstellt werden konnte, die alle Besonderheiten der anzusteuernden Komponenten, der im chemischen Experiment zu vermischenden Reagenzien und der Vor- und Nachbereitung von Experimenten berücksichtigt. Eine weitere Hürde war es, alle Benötigten Funktionen im sehr begrenzten Arbeitsspeicher der Steuereinheit unterzubringen und Probleme zu diagnostizieren, ohne diesen durch Debug-Nachrichten zum Überlaufen zu bringen.
Schlussendlich konnte ein sicheres, robustes und gut dokumentiertes System geschaffen werden, das alle Anforderungen, die an es gestellt wurden, erfüllt.
Durch die erstellte Netzwerkschnittstelle und die schnelle Umsetzung eingehender Befehle kann die Experimentieranlage in Zukunft leicht in das Rahmenprojekt integriert werden, sobald die anderen Softwaremodule dessen fertiggestellt wurden.
Das erstellte System wurde anhand einer praktischen, internetgesteuerten Durchführung eines Experimentes vor der Projektgruppe vorgestellt und seine Funktionsweise erklärt, wobei es durchweg positive Rückmeldungen gab.

A. Anhang

A.1. Zustandskodierung

Kodierung	Zustand
(0/X/X)	Initialising
(1/X/X)	Draining
(2/X/X)	Idling
(3/X/X)	Filling Tubes
(4/X/X)	WT Filling
(5/X/X)	After Fill Drain
(6(0/X/X)/X/X)	$\overline{P}_w, \overline{P}_{ea}, \overline{V}_w, \overline{V}_{ea}, \overline{V}_d$
(6(1/X/X)/X/X)	$\overline{P}_w, \overline{P}_{ea}, \overline{V}_w, \overline{V}_{ea}, V_d$
(6(2/X/X)/X/X)	$\overline{P}_w, \overline{P}_{ea}, \overline{V}_w, V_{ea}, \overline{V}_d$
(6(3/X/X)/X/X)	$\overline{P}_w, P_{ea}, \overline{V}_w, V_{ea}, \overline{V}_d$
(6(4/X/X)/X/X)	$\overline{P}_w, \overline{P}_{ea}, V_w, \overline{V}_{ea}, \overline{V}_d$
(6(5/X/X)/X/X)	$P_w, \overline{P}_{ea}, V_w, \overline{V}_{ea}, \overline{V}_d$
(6(6/X/X)/X/X)	$\overline{P}_w, \overline{P}_{ea}, V_w, V_{ea}, \overline{V}_d$
(6(7/X/X)/X/X)	$P_w, \overline{P}_{ea}, V_w, V_{ea}, \overline{V}_d$
(6(8/X/X)/X/X)	$\overline{P}_w, P_{ea}, V_w, V_{ea}, \overline{V}_d$
(6(X/0/X)/X/X)	Empty
(6(X/1/X)/X/X)	Some Water
(6(X/2/X)/X/X)	Enough Water
(6(X/3/X)/X/X)	Water > 2xEA
(6(X/4/X)/X/X)	Water = 2xEA
(6(X/5/X)/X/X)	Full
(6(X/6/X)/X/X)	Unknown
(6(X/7/X)/X/X)	Emptying
(6(X/X/0)/X/X)	Stirrer Off
(6(X/X/1)/X/X)	Stirrer On
(X/0/X)	EA Empty
(X/1/X)	EA Enough
(X/2/X)	EA Full
(X/X/0)	Light Off
(X/X/1)	Light On

Tabelle A.1.: Kodierung der Zustände

A.2. Gesamtautomat

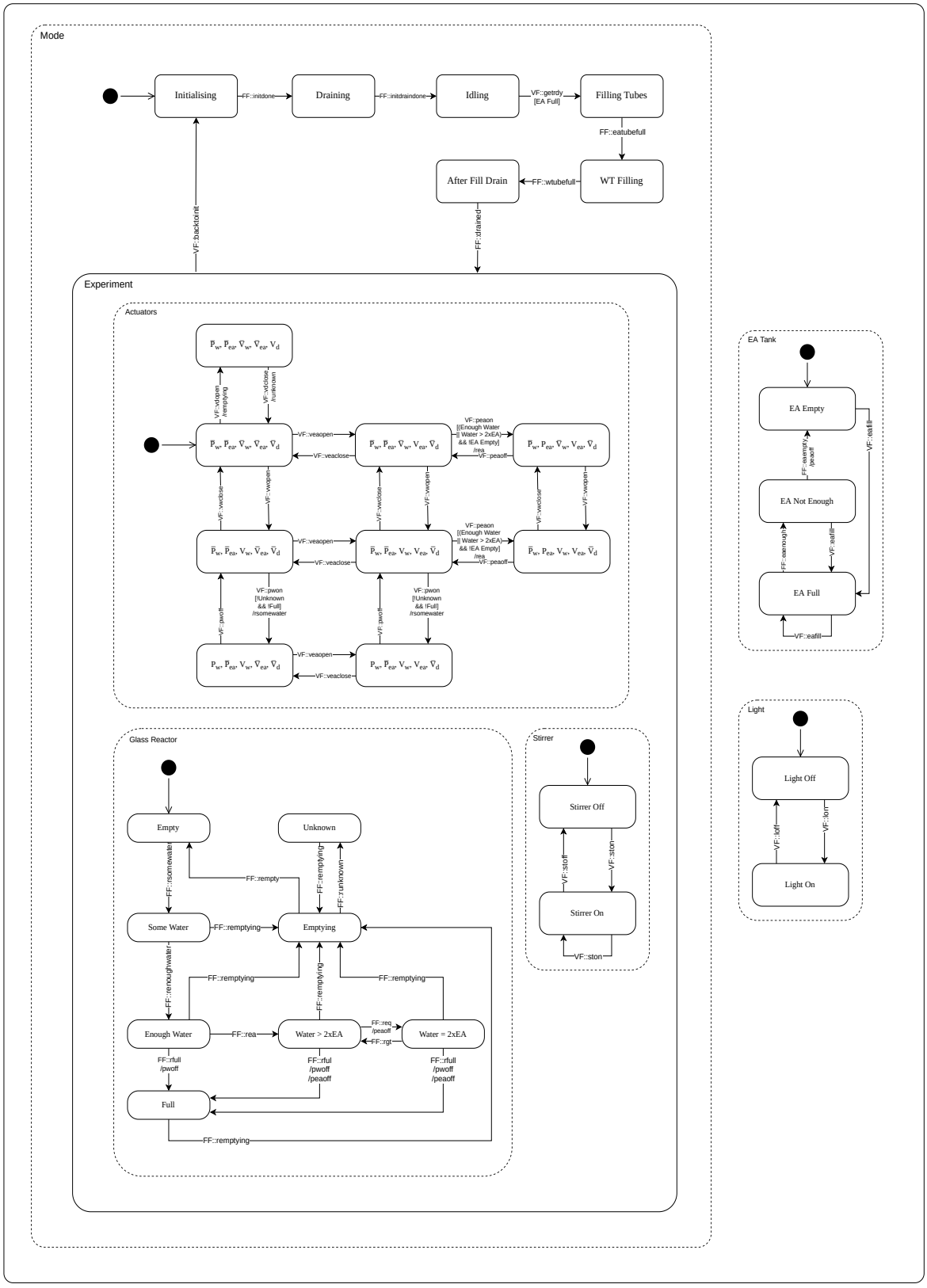

Abbildung A.1.: Gesamtautomat

A.3. Signale

Signal	Accept	Decline	Confirm	VF
initdone	accinitdone	decinitdone	confinitdone	—
initdraindone	accinitdraindone	decinitdraindone	confinitdraindone	—
getready	accgetready	decgetready	confgetready	✓
eatubefull	acceatubefull	deceatubefull	confeatubefull	—
wtubefull	accwtubefull	decwtubefull	confwtubefull	—
drained	accdrained	decdrained	confdrained	—
backtoinit	accbacktoinit	decbacktoinit	confbacktoinit	✓
vdopen	accvdopen	decvdopen	confvdopen	✓
vdclose	accvdclose	decvdclose	confvdclose	✓
veaopen	accveaopen	decveaopen	confveaopen	✓
veaclose	accveaclose	decveaclose	confveaclose	✓
vwopen	accvwopen	decvwopen	confvwopen	✓
vwclose	accvwclose	decvwclose	confvwclose	✓
peaon	accpeaon	decpeaon	confpeaon	✓
peaoff	accpeaoff	decpeaoff	confpeaoff	✓
pwon	accpwon	decpwon	confpwon	✓
pwoff	accpwoff	decpwoff	confpwoff	✓
runknown	accrunknown	decrunknown	confrunknown	—
remptying	accremptying	decremptying	confremptying	—
rempty	accrempty	decrempty	confrempty	—
rsomewater	accrsomewater	decrsomewater	confrsomewater	—
renoughwater	accrenoughwater	decrenoughwater	confrenoughwater	—
rea	accrea	decrea	confrea	—
req	accreq	decreq	confreq	—
rgt	accrgt	decrgt	confrgt	—
rfull	accrfull	decrfull	confrfull	—
ston	accston	decston	confston	✓
stoff	accstoff	decstoff	confstoff	✓
eafill	acceafill	deceafill	confeafill	✓
eaempty	acceaempty	deceaempty	confeaempty	—
eaenough	acceaenough	deceaenough	confeaenough	—
lon	acclon	declon	conflon	✓
loff	accloff	decloff	confloff	✓

Tabelle A.2.: Übersicht über alle zustandsändernde Signale

Parameter	Einheit	Zugehöriges Signal	Anmerkung
stirError	String	valueupdate	Enthält den ausgelesenen Fehlerspeicher des Rührers
stirActualRPM	min^{-1}	valueupdate	Die tatsächlichen, momentanen RPM des Rührers
stirSetRPM	min^{-1}	valueupdate	Soll-RPM des Rührers
stirTorque	Nmm	valueupdate	Drehmoment des Rührers
temperature	°C	valueupdate	Gemessene Temperatur im Glasreaktor
reactorLevel	ml	valueupdate	Füllstand des Glasreaktors
reactorWaterLevel	ml	valueupdate	Volumen von Wasser im Glasreaktor
reactorEaLevel	ml	valueupdate	Volumen von Essigsäureanhydrid im Glasreaktor
eaTankLevel	ml	valueupdate	Verbleibendes Volumen im Essigsäureanhydridbehälter
stirRPM	min^{-1}	ston	Gewünschte RPM des Rührers

Tabelle A.3.: Parameter und zugehörige Signale

A.4. Konfigurationsoptionen

Option	Standardbelegung	Beschreibung
DEBUG	0	Schaltet den Debug Modus an oder aus.
WS_DEBUG	0	Schaltet den Debug Modus des WebSocket Softwaremoduls an oder aus.
STIR_DEBUG	0	Schaltet den Debug Modus des Rührers an oder aus.
BUFF_SIZE	64	Größe des Debug Buffers
SERIAL_BD	115200	Baudrate der seriellen Verbindung zum Debug-Computer
MAX_CLIENTS	4	Maximale Anzahl Clients, die über Ethernet verbunden sein dürfen. Hardwareseitig beschränkt auf 4.
WS_SEND_BUFSIZE	400	Größe des WebSocket Buffers. Maximale Länge über WebSocket gesendeter Nachrichten.
WS_IP	"192.168.1.42"	IP Adresse der Anlage
WS_PORT	80	Port des WebSocket Servers
STBUFSIZE	50	Größe des Buffers, in dem die Zustände für eine State Response kodiert werden.
UPDATE_SIZE	400	Größe des Buffers, in dem die Parameter für eine State Response gesammelt werden.
V_EA_PIN	CONTROLLINO_D4	Pin des Essigsäureanhydridventils
V_WATER_PIN	CONTROLLINO_D5	Pin des Wasserventils
V_DRAIN_PIN	CONTROLLINO_D6	Pin des Abflussventils
P_EA_PIN1	CONTROLLINO_D0	Pin 1 der Essigsäureanhydridpumpe
P_EA_PIN2	CONTROLLINO_D1	Pin 2 der Essigsäureanhydridpumpe
P_WATER_PIN1	CONTROLLINO_D2	Pin 1 der Wasserpumpe
P_WATER_PIN2	CONTROLLINO_D3	Pin 2 der Wasserpumpe
LIGHT_PIN	CONTROLLINO_D7	Pin des Lichts
STIR_SERIAL	Serial2	Arduinobezeichnung der seriellen Schnittstelle, an die der Rührer angeschlossen ist.

Option	Standardbelegung	Beschreibung
STIR_DEFAULT_RPM	200	Standardmäßige Umdrehungen pro Minute des Rührers, wenn diese nicht spezifiziert werden.
STIR_MAX_RPM	300	Maximale Umdrehungen pro Minute, die der Treiber des Rührers erlaubt.
STIR_MIN_RPM	50	Minimale Umdrehungen pro Minute, die der Treiber des Rührers erlaubt.
STIR_BAUD_RATE	9600	Baudrate der seriellen Verbindung zum Rührer
STIR_RESPONSE_WAIT_TIME_MS	50	Zeit, die vom Rührertreiber maximal auf eine Antwort gewartet wird, nachdem ein Befehl verschickt wurde. (Falls keine Antwort kommt, blockiert der Treiber für diese Zeit)
STIR_CMD_PAUSE_MS	100	Zeit, die zwischen zwei gesendeten Befehlen zum Rührer verstrichen sein muss. Zu niedrige Werte führen zu undefiniertem Verhalten des Rührers. (Wenn versucht wird, einen Befehl zu versenden, bevor die Zeit verstrichen ist, blockiert der Treiber vor dem Versenden, bis die Zeit erreicht wurde)
STIR_MAX_RESP_LEN	65	Größe des Buffers für die Antworten des Rührers
THERM_BAUD_RATE	9600	Baudrate der seriellen Verbindung zum Thermoelementleser
THERM_MODBUS_MASTER_ADDR	0	Modbusadresse des Controllino
THERM_MODBUS_SLAVE_ADDR	1	Modbusadresse des Thermoelementlesers
RT_EMPTYING_TIME_SEC	150L	Sekunden die es braucht, einen komplett gefüllten Glasreaktor vollständig zu leeren. Wird bei Start der Anlage benutzt, um sicherzugehen, dass dieser leer ist.
RT_DRAIN_FACTOR	750L	Ungefähre Anzahl Millisekunden die es braucht, einen Milliliter aus dem Glasreaktor zu leeren. Sollte höher als die tatsächliche Zeit gesetzt werden.

Option	Standardbelegung	Beschreibung
RT_ENOUGH_WATER_THRESH	80.0	Wasserstand des Glasreaktors, bei dem vom Zustand “Some Water” in den Zustand “Enough Water” gewechselt wird.
RT_TANK_FULL_THRESH	200.0	Füllstand des Glasreaktors, bei dem in den Zustand “Full” gewechselt wird.
MS_PER_ML	420.0	Anzahl Millisekunden die es braucht, einen Milliliter Flüssigkeit zu pumpen. Sollte so genau wie möglich gesetzt werden.
MS_PER_ML_INT	420L	Selbiges als Ganzzahl.
WTUBE_FULL_MS	28000L	Zeit in Millisekunden, die die Anlage pumpen muss, um den Wasserschlauch zu füllen. Sollte zur Sicherheit etwas höher als tatsächlich gesetzt werden.
EATUBE_FULL_MS	20000L	Zeit in Millisekunden, die die Anlage pumpen muss, um den Essigsäureanhydridschlauch zu füllen. Sollte zur Sicherheit etwas höher als tatsächlich gesetzt werden.
EA_TANK_MAX_LEVEL_ML	1800L	Füllstand in Millilitern, auf den der EA-Vorratsbehälter aufgefüllt wird. Sollte zur Sicherheit niedriger als tatsächlich gesetzt werden.
EA_TANK_MIN_LEVEL_ML	100L	Mindestfüllstand in Millilitern des EA-Vorratsbehälters, die vorhanden sein müssen, um in den Experimentiermodus zu wechseln. Sollte zur Sicherheit höher als tatsächlich gesetzt werden.
STIRRER_INTERVAL_MS	5000	Abstand in Millisekunden, in denen die Parameter des Rührers per `valueupdate` verschickt werden, sollten sie sich geändert haben.
TEMP_INTERVAL_MS	10000	Abstand in Millisekunden, in denen der Parameter der Temperatur per `valueupdate` verschickt wird, sollte er sich geändert haben.

Option	Standardbelegung	Beschreibung
`LEVELS_INTERVAL_MS`	`1000`	Abstand in Millisekunden, in denen die Parameter des Glasreaktors per `valueupdate` verschickt werden, sollten sie sich geändert haben.
`EMPTYING_GR_TIME_MS`	`10000L`	Millisekunden, die nach Befüllen der Schläuche gewartet werden, damit der Glasreaktor wirklich leer ist. Sollte höher als tatsächlich gesetzt werden.

Tabelle A.4.: Konfigurationsoptionen

A.5. Rührertreiber

Methode	Aufrufparameter	Rückgabewert	Beschreibung
`char *sendCommand( allowedCommands cmd, uint16_t rpm=0)`	`cmd`: Auszuführender Befehl aus Enum allowedCommands `rpm`: Einzustellende RPM	Pointer auf Puffer der Klasse, welcher die letzte Antwort des Rührers enthält. Puffer wird von nachfolgender Antwort überschrieben.	Sendet den als Parameter spezifizierten Befehl an den Rührer.
`void asyncSendCommand( allowedCommands cmd, uint16_t rpm=0)`	Siehe `sendCommand`	—	Unbenutzte Implementierung einer asynchronen Version der `sendCommand` Methode
`char *asyncReadResponse()`	—	Siehe `sendCommand`	Unbenutzte Implementierung einer asynchronen Version der `sendCommand` Methode
`char *getErrorCode()`	—	Siehe `sendCommand`	Wrapper der `sendCommand` Methode, welcher den Fehlercode Befehl ausführt.
`void setRPM(uint16_t rpm)`	`rpm`: Einzustellende RPM	—	Legt die beim Anschalten des Rührers benutzten RPM fest.
`uint16_t getRPM()`	—	Die momentan festgelegten RPM	—
`uint16_t getActualRPM()`	—	Die momentane, tatsächliche Drehzahl des Rührers	Wrapper der `sendCommand` Methode, welcher den `r\r\n` Befehl ausführt.
`uint16_t getSetRPM()`	—	Die momentane Soll-RPM	Wrapper der `sendCommand` Methode, welcher den `s\r\n` Befehl ausführt.

Methode	Aufrufparameter	Rückgabewert	Beschreibung
`int16_t getTorque()`	—	Das momentane Drehmoment	Wrapper der `sendCommand` Methode, welcher den `m\r\n` Befehl ausführt.
`void on()`	—	—	Wrapper der `sendCommand` Methode, welcher den Rührer einschaltet.
`void off()`	—	—	Wrapper der `sendCommand` Methode, welcher den Rührer ausschaltet.
`char *RPMtoCmd( uint16_t rpm)`	Die zu konvertierende Drehzahl	Ein der Drehzahl entsprechender `Rxxxx\r\n` Befehl	Helferfunktion, die eine gegebene Drehzahl in einen entsprechenden Befehl umwandelt.

Tabelle A.5.: Funktionen des Rührertreibers

A.6. Schaltplan

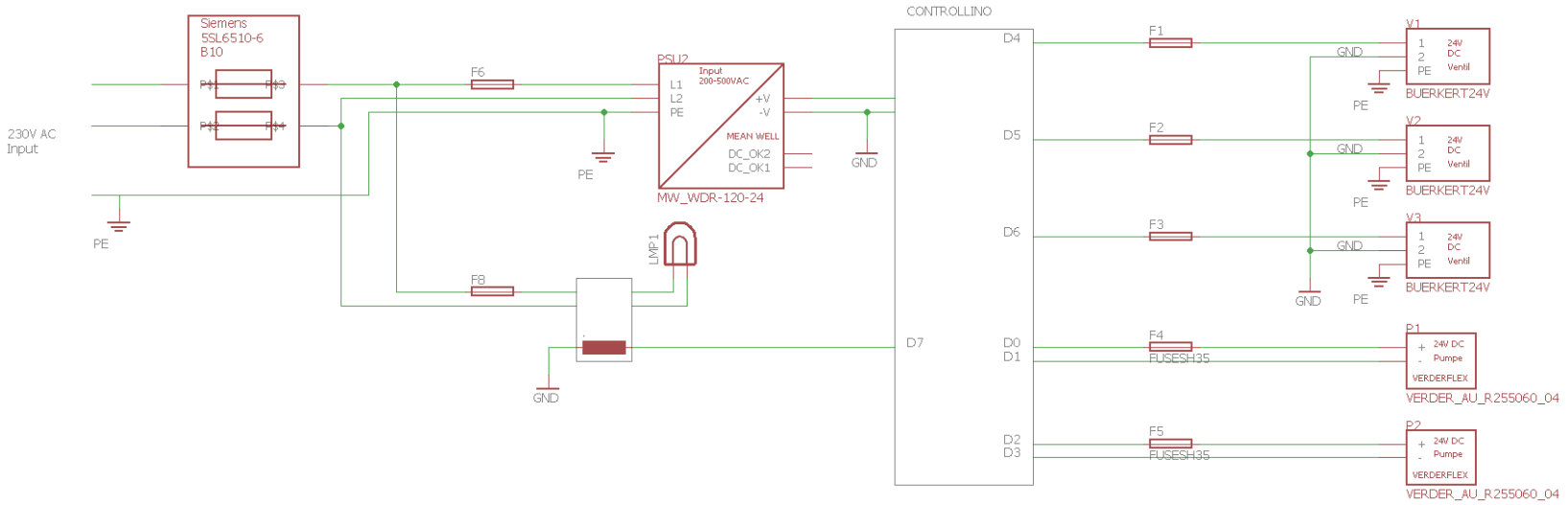

Abbildung A.2.: Schaltplan des Steuergerätes und der über GPIOs angeschlossenen Komponenten

A.7. Webapplikation

Abbildung A.3.: Oberfläche der Webapplikation

Literatur

[1] *Thermoelemente - Teil 1: Thermospannungen und Grenzabweichungen (IEC 60584-1:2013)*. Norm. Juli 2014.

[2] Lukas Focken und Florian Schmalriede. *Entwicklung einer internet-basierten Plattform für die Durchführung von fernsteuerbaren Experimenten*. 2023.

[3] Topsccc Technology Co., LTD. *Operation Manual of EX-9018/18-M/18BL/18BL-M/19/19-M*.

[4] WIZnet Co., Inc. *W5100 Datasheet*. Version 1.2.8.

[5] Christian Bürkert GmbH & Co. KG. *Typ 6013 Datenblatt | direktwirkendes Hubankerventil*.

[6] Verder Deutschland GmbH & Co. KG. *Verderflex OEM M025 DC*.

[7] Heidolph Instruments GmbH & Co. KG. *Betriebsanleitung Hei-TORQUE Expert / Ultimate*.