Hochschule München University of Applied Sciences

Weiterentwicklung der Systemsoftware für eine Siebträger Espressomaschine

Projektbericht

Further development of the system software of a portafilter espresso machine

Project Report

Sommersemester 2023

Hochschule für angewandte Wissenschaften München

Dachauerstr. 98b, 80335 München

~~Noureddine Ait Ouhamou~~	~~7EM~~	~~40217818~~	~~ait-ouha@hm.edu~~
Simon Lorenz Thrainer	MBB4	23687721	thrainer@hm.edu
Melina Scherf	MBB6	49200620	melina.scherf@hm.edu
Madita vom Stein	MBB4	33087021	madita_vom.stein@hm.edu
Hendrik Wegjan	MBB4	12156621	wegjan@hm.edu

1. Abstract

Ziel dieses Berichtes ist es, einen abschließenden Überblick über die Arbeiten der Projektgruppe Systemsoftware SoSe 2023 zu geben. Die bereits vorliegende MATLAB®-GUI [85] wurde neu aufgesetzt und benutzerfreundlicher gestaltet. Im Zuge des Umbaus auf die Multi-MCU-Elektronik während der Projektarbeit wurde der Programmcode entsprechend angepasst und der Leitungsdrucksensor integriert. Das Teilprogramm zu den Spülvorgängen wurde separat geschrieben und erprobt. Ebenso wurden die verbauten Schrittmotoren hinsichtlich der Pausenzeit und Schrittanzahl untersucht. Für die Auslegung der Regler wurde ein erster Vorschlag für das weitere Vorgehen erarbeitet.

The aim of this report is to provide a final overview of the work carried out by the System Software project group in the summer semester of 2023. The existing MATLAB® GUI [85] has been redeveloped and made more user-friendly. As part of the transition to multi-MCU electronics during the project work, the program code was adjusted accordingly, and the pressure sensor was integrated. The subprogram for the cleaning processes was written and tested separately. Similarly, the installed stepper motors were examined in terms of pause time and step count. A preliminary proposal for the further steps in designing the controllers has been developed.

Inhaltsverzeichnis

2. Verzeichnis der Abkürzungen

GUI	Graphical User Interface
LbA	Lehrkraft für besondere Aufgaben
MCU	Microcontroller Unit
NTC	Negative Temperature Coefficient / „Heißleiter“
PID(-Regler)	Proportional Integral Derivative (Regler)
SSR	Solid State Relais

3. Einleitung

Es wurde eine Aufgabenanalyse durchgeführt, auf dessen Basis die Zielvereinbarung entstanden ist [Wiki: „Entwicklung Systemsoftware SoSe2023“ / „Leistungsvereinbarung Projektgruppe“].

Hauptaugenmerk lag dabei auf dem Umbau der aktuellen Hardware im Sinne der geplanten Multi-MCU (Schema s. Abb. 1). Für diesen Umbau mussten sowohl Hard- als auch Software bereitgestellt werden (Zielvereinbarung Nr.13).
Während des Umbaus wurde zudem der schon in die Maschine eingebaute Leitungsdrucksensor angeschlossen (Zielvereinbarung Nr.3)

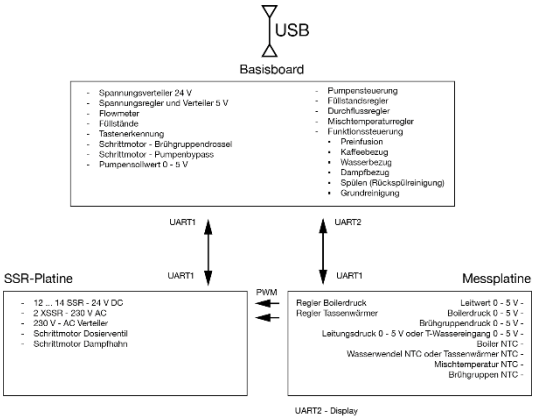

Abbildung 1: Schema der Multi-MCU-Hardware

Ferner wurde die Thematik der Regleranpassung Misch-/Durchflussregler aufgenommen (Zielvereinbarung Nr.4/5).
Die theoretische Vorarbeit wurde hier geleistet, die Erprobung erfolgt in den Folgesemestern, startend mit dem Wintersemester 2023/24.

Weiterhin fehlten der Labormaschine noch Programme, die Standardabläufe automatisieren (Zielvereinbarung Nr.6-9). Dementsprechend wurden Spülvorgänge programmiert, ein Kaffee- bzw. Wasserbezug wurde aus Zeitgründen nicht mehr umgesetzt.

Außerdem war eine Abarbeitung der Schrittmotor-Thematik für alle drei Drosseln (Bypass, Dosierventil, Brühgruppendrossel) vereinbart (Zielvereinbarung Nr.10-12). Es wurde die für alle Motoren notwendige Initialisierung erprobt, sowie eine eindeutige Stell-Pausenzeit festgelegt (s. Kapitel 8: Schrittmotoren). Für das Dosierventil an der Mischstelle Heißwasser/Kaltwasser wurde eine Kennlinie generiert.

~ *Hendrik Wegjan*

4. Erfassung des Leitwerts

Bei diesem ToDo soll statt dem Leitwert des Wassers der Leitungsdruck durch einen Drucksensor ausgegeben werden. Dafür muss in der MATLAB-GUI der Umrechnungsfaktor angepasst werden.

Zu Beginn wurde der Leitwert durch den Leitwertsensor des Typs ICS PPSU erfasst. Mit diesem Sensor lässt sich ein Leitwert von 0 bis 20 mS/cm messen. Dabei entsprechen 0 V gleich 0 mS/cm und 5 V gleich 20 mS/cm [http://www.institut-fuer-

kaffeetechnologie.de/Intern/images/1/1a/01_AVS_R%C3%B6mer_Produktkatalog_2021.pdf]. Der Drucksensor war ebenfalls schon verbaut.

Der Drucksensor sollte geradliniger in der Maschine verbaut werden und an die Elektronik angeschlossen werden. Die Verrechnung des Spannungswerts musste angepasst werden. [Wiki: „Systemsoftware“ / „Messwerte erfassen (Labor)“, Eintrag vom 07.04.2023]

Es wurden ein paar Verbindungen in der Maschine entfernt, der Drucksensor sitzt jetzt direkt nach dem Flowmeter. Der Sensor wurde ebenfalls an die neue Systemelektronik angeschlossen und zeigt im drucklosen Zustand die richtigen Spannungswerte von 0,5 V bis 0,6 V an. Beim Erhöhen des Drucks stieg der Spannungswert ebenfalls an. Der Umrechnungsfaktor kann aus dem Datenblatt entnommen werden, indem man die Geradengleichung vom abgebildeten Graphen erfasst. Diese muss in der MATLAB-GUI noch angepasst werden. Zudem wird beim Betrieb der Maschine der Sensor genauer getestet.

~ Noureddine Ait Ouhamou

5. Mischwassertemperaturregler

Aufgabe des Mischwassertemperaturreglers (kurz: Mischregler) ist es, die Mischwassertemperatur dem Soll-Wert entsprechend einzustellen. Die Mischwassertemperatur wird durch das dosierte Zugeben von Kaltwasser zu dem in der Heizwendel des Boilers erhitzten Wassers reguliert. Dieser Regler muss ausgelegt werden.

Wie in [85] ausführlich beschrieben, ist der Regler aktuell als PID-Regler im Programmcode angelegt, wobei der D-Anteil momentan gleich null ist, also ein PI-Regler vorliegt. Auch Zielvorschläge für die Einschwingdauer und die stationäre Genauigkeit liegen vor [85].

Ziel bei der Reglerauslegung ist, die Soll-Temperatur innerhalb einer möglichst kurzen Einschwingdauer, mit möglichst geringen Überschwingern, möglichst stationär genau zu erreichen. Die quantitativen Ziele für Einschwingdauer und stationäre Genauigkeit sind zu prüfen und gegebenenfalls anzupassen. Die zulässige prozentuale Abweichung durch Überschwinger ist festzulegen.

Zum Erreichen dieser Ziele sollen k_P-, k_I- und k_D-Faktoren, beziehungsweise sinnvolle Kombinationen eben dieser, optimiert werden. Ein P-Anteil reagiert ohne Verzögerung auf den Regelfehler; hinterlässt aber bleibende Abweichungen. Der integrative Anteil eines PID-Reglers sorgt für stationäre Genauigkeit; verzögert die Regelung jedoch. Durch den D-Anteil wird die Wahl eines größeren k_P-Faktors ermöglicht, da das System weniger schnell zum Schwingen beginnt.
Ein von uns vorgeschlagener Lösungsansatz ist, den Temperaturverlauf über der Zeit mittels der Ziegler und Nichols Methode auszuwerten, und so mögliche Kombinationen der Regelparameter zu bestimmen. Der Temperaturverlauf ergibt sich aufgrund von

Eingangssprüngen, also Sprüngen der Halbschrittanzahl des Schrittmotors. Der grundlegende Gedanke dieses Ansatzes ist in Abbildung 2 visualisiert.

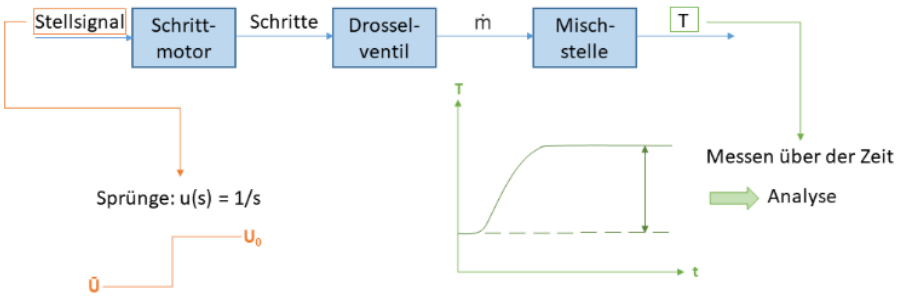

Abbildung 2: Auslegung des Mischreglers

~ Madita vom Stein

6. *Durchflussregler (Flowmeter)*

Aufgabe des Durchflussreglers ist es, den Soll-Durchfluss einzustellen. Dieser liegt für einen einfachen Espresso bei 25 ml in 25 s, also bei einer Durchflussrate von 1 ml / s [74].

Aktuell ist der Durchflussregler als PID-Regler im Programmcode hinterlegt, wobei auch hier der kD-Faktor null ist, also ein PI-Regler vorliegt. Es wurden eine gewünschte stationäre Genauigkeit, sowie vorläufige k-Faktoren festgelegt [85].

Ziel bei der Reglerauslegung ist, die gewünschte Durchflussrate innerhalb einer möglichst kurzen Einschwingdauer, mit möglichst geringen Überschwingern, möglichst stationär genau zu erreichen. Das quantitative Ziel für die stationäre Genauigkeit ist zu prüfen und gegebenenfalls anzupassen. Für die Einschwingzeit, sowie die Überschwinger sind quantitative Ziele festzulegen.

Grundsätzlich sollen auch hier die Regelparameter, beziehungsweise sinnvolle Kombinationen dieser festgelegt werden.
Ein von uns vorgeschlagener Lösungsansatz ist, den Durchflussverlauf (Massen- bzw. Volumenstrom) über der Zeit mittels der Ziegler und Nichols Methode auszuwerten, und so mögliche Kombinationen der Regelparameter zu bestimmen. Der Durchflussverlauf ergibt sich aufgrund von Eingangssprüngen, also Sprüngen der Ansteuerspannung der Pumpe. Diese liegt im Bereich von 0 - 5000 mV. Zu beachten ist dabei der zur Pumpe parallel geschaltete Bypass, durch den die Möglichkeit besteht, den Maximaldruck im System zu limitieren. Es gilt zu prüfen, ob dieser Bypass veränderlich sein soll oder nicht. Der Durchfluss wird mittels einer Durchflussmessung hinter Pumpe und Bypass festgestellt. Der grundlegende Gedanke dieses Ansatzes ist in Abbildung 3 visualisiert.

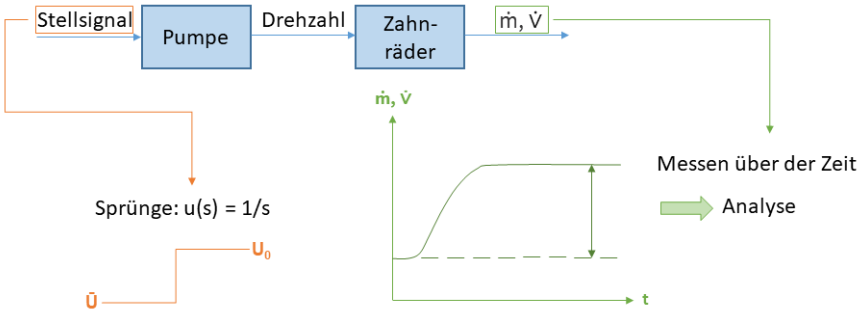

Abbildung 3: Auslegung des Durchflussreglers

~~Für beide Regler gilt, dass die Regelparameter aufgrund eines Ausfalls der Maschine leider nicht mehr von unserer Projektgruppe ermittelt werden konnten.~~ Die vorangegangenen Absätze bezüglich der Reglerauslegung sind also als Vorbereitung und Lösungsansatz für eine Folgegruppe zu verstehen.

~ Madita vom Stein

7. Spülvorgänge

Bei der Reinigung einer Siebträgermaschine wird zwischen dem Flush und der Rückspülung unterschieden. Beide werden durchgeführt, um Rückstände und Verschmutzungen zu entfernen, die den Geschmack negativ beeinträchtigen. Beim Flush, den der Bediener nach jedem Kaffeebezug ausführt, wird die Brühgruppe ohne eingespanntem Siebträger mit Wasser gespült. Bei der Rückspülung wird Reinigungspulver in ein Blindsieb gegeben und durch mehrmaliges Anschalten der Pumpe in heißem Wasser aufgelöst. Die öligen Rückstände des Kaffees werden dabei gelöst und anschließend ausgespült. Die Rückspülung wird bei hohem Kaffeedurchsatz, wie im gastronomischen Bereich, mindestens täglich und bei seltenerem Kaffeebezug, wie im Hausgebrauch, wöchentlich durchgeführt.

Zu Projektbeginn stand noch keine Möglichkeit zur automatisierten Reinigung der Maschine zur Verfügung. Auf Knopfdruck musste ein Spülvorgang ausgelöst werden, der verschiedenen Fälle unterscheidet und jeweils den richtigen Reinigungsablauf startet. Wenn ein Siebträger mit Blindsieb in die Maschine eingespannt war, musste ein Rückspülvorgang ausgeführt werden. Wenn ein Siebträger mit leerem Sieb oder kein Siebträger eingespannt war, musste ein Flush durchgeführt werden. Befand sich ein Sieb mit Kaffeepuk in der Maschine ~~musste~~ eine Fehlbedienung angenommen und die Reinigung abgebrochen werden.

Es wurden Versuche durchgeführt, um Grenzwerte für Brühgruppendruck und Durchflussrate festzustellen, die die Unterscheidung der Fälle eindeutig möglich machen. Dazu wurde mit einer Pumpenleistung von 2000 mV und geschlossenem Bypass Wasser in die Brühgruppe gefördert und die beiden Größen gemessen. Es wurde festgestellt, dass der Brühgruppendruck bei eingespanntem Blindsieb auf einen statischen Wert von > 13 bar ansteigt und die Durchflussrate auf 0 cm^3/s abfällt. Ist kein Siebträger eingespannt, baut sich in der Brühgruppe kein Druck auf und der Durchfluss liegt über $3cm^3/s$. Bei Durchführung des Fehlerfalls wurde festgestellt, dass der Brühgruppendruck stark abhängig vom Mahlgrad ist. Bei Kaffeepulver mit Mahlgrad 3 stellte sich ebenfalls ein Wert > 13 bar ein, wodurch die Unterscheidung von Fehlerfall und Rückspülung problematisch wird. Eine automatische Fallunterscheidung funktioniert also nur bei Einsatz eines bestimmten Mahlgrads. Daher wurde der Versuch mit Mahlgrad 4 wiederholt und ein maximaler Druck von 10,5 bar bei einem Durchfluss < 1 cm^3/s gemessen. Damit ist eine Fehlbedienung eindeutig von den anderen Fällen unterscheidbar. [Wiki: "Systemsoftware" / "Spülvorgänge (Labor)", Eintrag vom 04.04.2023] Um die Spülvorgänge zu programmieren und zu testen, wurde eine Kopie der bisherigen MATLAB®-GUI [85] mit Namen „Reinigungsprogramm_EspressoMenu_20230424“ erstellt. Auf Basis der durch die Versuche gewonnenen Erkenntnisse wurde die Funktion „Reinigungsprogramm“ geschrieben, deren Ablauf im Flussdiagramm in Abbildung 4 zu sehen ist.

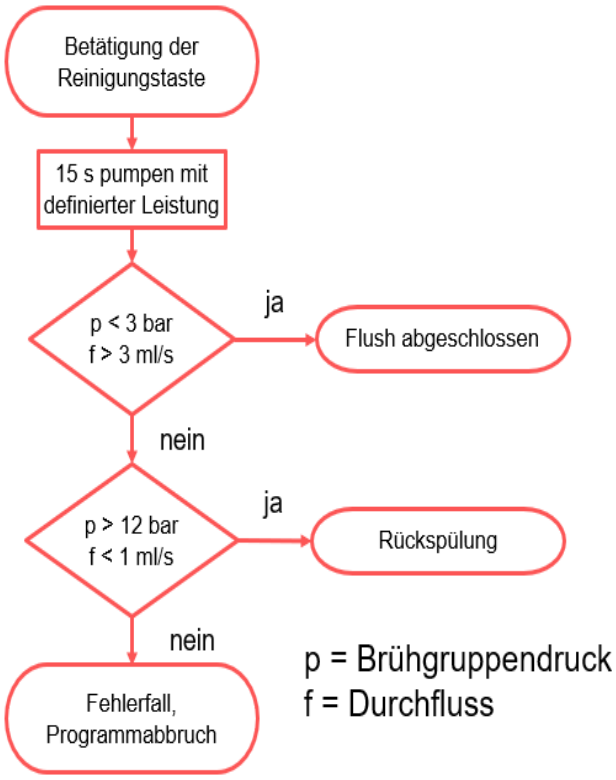

Abbildung 4: Ablauf des Reinigungsprogramms

Bei Betätigung des Reinigungsbuttons in der MATLAB®-GUI wird mit einer Pumpenleistung von 2000 mV und geschlossenem Bypass Wasser in die Brühgruppe fördert. Wenn sich Brühgruppendruck und Durchfluss nach einer Wartezeit von 15 s eingependelt haben und eine Fallunterscheidung eindeutig möglich ist, wird eine if-elseif-Abfrage ausgeführt. Dabei werden immer Brühgruppendruck und Durchflussrate gleichzeitig abgefragt. Wenn der Brühgruppendruck unter 3 bar und der Durchfluss über 3 ml/s liegt, wird ein Flush durchgeführt, der mit der Wasserförderung vor der Fallunterscheidung bereits erledigt ist. Daher wird die Pumpe abgeschaltet und das Programm beendet. Wenn die erste Doppelbedingung nicht erfüllt ist, wird abgeglichen, ob der Brühgruppendruck > 12 bar und der Durchfluss < 1 ml/s ist. In diesem

Fall wird der weitere Ablauf der Rückspülung eingeleitet, der in Abbildung 5 dargestellt ist.
Dabei wird dreimal abwechselnd für 2 s durch Nachfördern von Wasser Druck aufgebaut und dann 10 s lang gewartet, damit sich das im Blindsieb befindliche Reinigungspulver auflöst. Anschließend wird die Reinigungslösung mit den gelösten Kaffeerückständen 20 s lang durch das Rückspülventil aus der Brühgruppe gespült. Sind weder die Bedingungen für Flush noch für Rückspülung erfüllt, wird der Fehlerfall erkannt. Dabei wird das Rückspülventil für 1 s geöffnet, um evtl. vorhandenen Druck abzubauen und dann die Reinigung mit Ausgabe einer Fehlermeldung abgebrochen. Wie der Ablauf des Programms in MATLAB® umgesetzt wurde ist den Kommentaren im Code zu entnehmen.

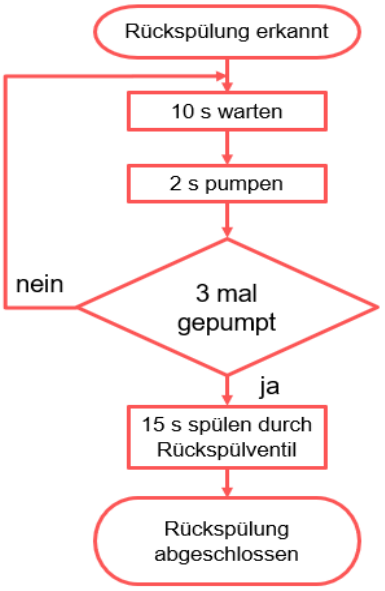

Abbildung 5: Ablauf der Rückspülung

Es muss noch erprobt werden, ob die angesetzten Förder- und Wartezeiten sowie die Anzahl der Nachfördervorgänge bei der Rückspülung zur Reinigung der Maschine ausreichen. Die optimalen Werte, mit denen die Reinigung so schnell und effektiv wie möglich abläuft, müssen in zukünftigen Projektarbeiten untersucht werden. Außerdem ist nicht ausgeschlossen, dass das Rückspülventil alternative Möglichkeiten bietet, um das Reinigungspulver bei der Rückspülung aufzulösen. Es ist denkbar, dass das Pulver durch Erzeugen eines minimalen Durchflusses in der Brühgruppe bei geöffnetem Rückspülventil schneller gelöst wird als beim wiederholten Pumpen und Warten. Diese Möglichkeit muss noch erprobt werden. Die Reinigungsfunktion wurde noch nicht in die, ebenfalls im Rahmen der Projektarbeit im SoSe 23, neu erstellte GUI übernommen. Der Code muss implementiert und an den entsprechenden Stellen mit der Oberfläche im Reinigung-Tab verknüpft werden. (s. Kapitel 9: Neuprogrammierung Systemsoftware)

~ Simon Thrainer

8. Schrittmotoren

In der Labormaschine sind aktuell drei schrittmotorgesteuerte Drosseln verbaut. Hierzu gehören zwei elektronische Dosierventile von AVS Römer (Dosierventil Heiß-/Kaltwasser, Bypass), sowie die über einen Getriebe-Schrittmotor angesteuerte Brühgruppendrossel.

Bei der Übernahme durch die Projektgruppe war bereits eine Initialisierungsroutine der Schrittmotoren vorhanden. Diese beinhaltete das Öffnen und Schließen der drei Drosseln durch eine Ansteuerung der Schrittmotoren mit jeweils 800 Halbschritten.
Ein Stellvorgang erfolgte jedoch mit einer nicht geprüften Pausenzeit von 2500 µs

zwischen den einzelnen Halbschritt-Stellvorgängen. Außerdem wurden die Ventile bei dieser Initialisierung regelmäßig über ihre Endlagen hinaus betrieben, da der maximale Verfahrbereich (zumindest bei Bypass- und Dosierventil Heiß-/Kaltwasser) laut Hersteller nur 560 Halbschritte beträgt. Die Brühgruppendrossel ist von dieser Problematik aufgrund ihres Getriebeschrittmotors und der daraus resultierenden höheren Schrittanzahl nicht betroffen.

Hieraus leiteten sich die Ziele der Projektgruppe Sommersemester 2023 ab.
Es musste ermittelt werden, bei welcher Pausenzeit zwischen den einzelnen Stellvorgängen kein Halbschritt „verloren geht“ bzw. nicht gefahren wird.
Im Anschluss an diese Erkenntnis sollte die Initialisierung optimiert werden.
Für das Dosierventil Heiß-/Kaltwasser musste eine Kennlinie ermittelt werden, die den Verlauf der Mischwassertemperatur über die Dosierventilstellung darstellt.
Für das Bypass-Ventil musste eine Kennlinie ermittelt werden, die den Verlauf des Leitungsdrucks über die Bypass-Ventilstellung darstellt.
Die Brühgruppendrossel sollte separat betrachtet werden. Hier war die Fragestellung, ob eine solche Drossel überhaupt benötigt wird oder ob diese durch Regelungstechnik ersetzbar ist.

Die Pausenzeit-Thematik konnte im Verlaufe des Semesters gelöst werden. Dabei war die Zielsetzung, eine geringe Pausenzeit zu verwenden, um die Initialisierung möglichst kurz zu halten. Gleichzeitig durfte kein aus MATLAB® übergebener Halbschritt „übersprungen“ werden, eine gewisse „Mindest-Pausenzeit“ war also erforderlich.
Das Phänomen des „Überspringens“ wird durch den Aufbau und die Funktionsweise der Schrittmotoren verursacht. Anders als in herkömmlichen Elektromotoren besitzen sie einen Rotor ohne Spulen, dieser ist entweder permanent- oder ferromagnetisch [100].
Eine Bewegung wird nur durch das Umpolen der Statorspulen erzeugt, also durch ein Umschalten der vier Output-Pins.

Diese Umpolung geschieht programmseitig quasi sofort, die Statorspulen benötigen jedoch eine gewisse Zeit, bis das Magnetfeld etabliert ist. Außerdem muss der Rotor, der eine gewisse Massenträgheit besitzt, dem Magnetfeld folgen.
Wird eine kritische Pausenzeit unterschritten, so kommt der Schrittmotor im wahrsten Sinne des Wortes „nicht hinterher“ und bleibt hängen. Dadurch steht er automatisch nicht mehr in der richtigen Position für die nächste Sequenz, ein Springen ist somit unvermeidbar.

Diese Problematik ist in [40], sowie dem Wiki-Eintrag [Wiki: „Systemsoftware“ / „Schrittmotorensteuerung Mischer“, Eintrag vom 05.05.2023] genauer erklärt.

Zur Ermittlung der kritischen Pausenzeit wurden die Bypass-Drossel und der Brühgruppen-Drucksensor verwendet, um einen zeitlichen Verlauf des Brühgruppendrucks zu erfassen. Der gleiche Versuchsablauf wurde mit verschiedenen Pausenzeiten wiederholt:

1. Bypass-Drossel gesichert vollständig öffnen (z.B. mit Thonny-Programm „Schrittmotor_gesicherter_Reset“)
2. Wasserzufuhr auf Brühgruppe schalten (Y1 + Y6 + Y7 ON)
3. Pumpe einschalten (Leistung: 3000mV)
4. In Tab „Tests“ wechseln und „Schrittmotor_Auto“ anklicken (veraltet! Der Tab kommt in der aktuellen GUI „EspressoMenu_20230522“ nicht mehr vor)
5. Warten, bis Maximaldruck (ca. 13 bar) erreicht ist, dann Messwertpuffer speichern und Pumpe ausschalten, evtl. Brühgruppe entlüften (Y08 ON)

Die Funktion „Schrittmotor_Auto“ wurde als zusätzliche MATLAB®-Funktion implementiert und macht kurzgesagt nichts anderes, als die Bypass-Drossel alle 5 Sekunden um 40 Halbschritte zu schließen.
Sollte die kritische Pausenzeit unterschritten sein, so werden bei den einzelnen Ansteuerungen weniger als 40 Halbschritte gefahren. Die Folge daraus ist ein unsteter und langsam steigender Brühgruppendruck.
Für einen Bereich von 5000 – 1000 µs wurden Kennlinien generiert:

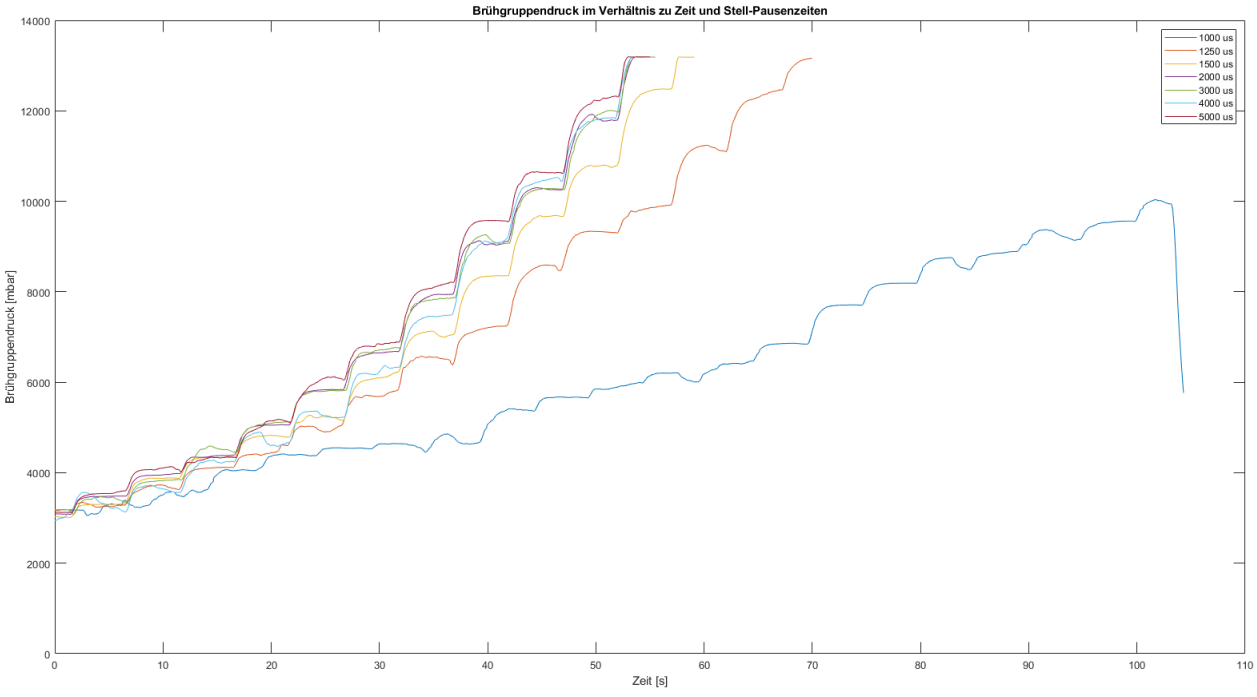

Abbildung 6: Darstellung Brühgruppendruck bei verschiedenen Stell-Pausenzeiten

Es ist klar ersichtlich, dass bei Pausenzeiten unter 1500 µs (orangene und dunkelblaue Kennlinie) Halbschritte verlorengehen, da das oben beschriebene Phänomen des „Überspringens“ auftritt.

Ab einer Pausenzeit von 2000 µs (lila) nähern sich die Grafen deutlich an. Daher ist diese Zeit als sicher annehmbar.

Basierend auf dieser Erkenntnis wurde die Kennlinie Mischwassertemperatur über Dosierventilstellung generiert [Wiki: „Systemsoftware“ / „Schrittmotorensteuerung Mischer“, Eintrag vom 23.05.2023]:

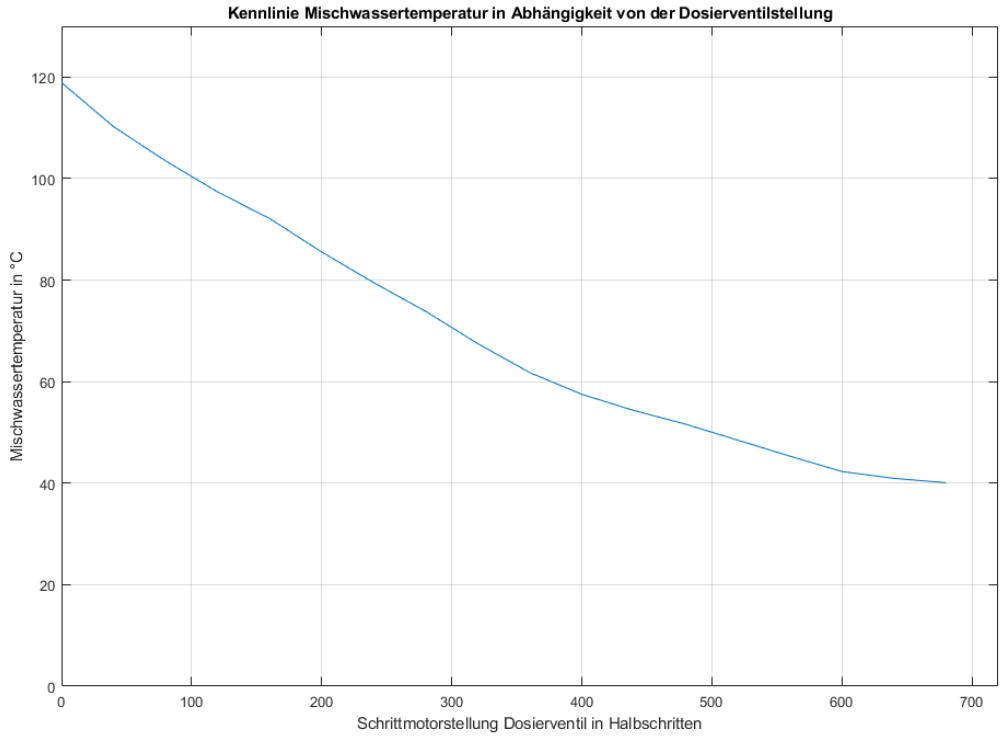

Abbildung 7: Kennlinie der Mischwassertemperatur in Abhängigkeit der Dosierventilstellung

Diese Grafik ermöglicht es einer Folgegruppe, eine gewünschte Bezugstemperatur sofortig durch gezieltes Stellen des Dosierventils zu erreichen, ein „Herumprobieren“ entfällt. Weiterhin soll diese Funktion langfristig in den Programmcode eingebettet werden, um den Prozess der Temperaturregelung zu automatisieren.

Die Kennlinie für die Bypass-Drossel wurde ~~noch~~ nicht angelegt ~~und ist somit eine Aufgabe für kommende Projektgruppen.~~ Die Vorgehensweise ist sehr ähnlich zu dem der hier gezeigten Kennlinien-Erstellung, der Wiki-Eintrag [Wiki: „Systemsoftware“ / „Schrittmotorensteuerung Mischer“, Eintrag vom 23.05.2023] ist als mögliche Anleitung nutzbar.

Im ToDo Brühgruppendrossel wurde kein nennenswerter Fortschritt erzielt.

Abschließend sind beim Dosierventil Heiß-/Kaltwasser während des Semesters mehrere Folgeaufgaben entstanden:
Zunächst ist hier auf die Unstimmigkeiten zwischen den Versuchsergebnisse und den Angaben des Herstellers AVS Römer bezüglich der verfahrbaren Halbschrittanzahl hinzuweisen:

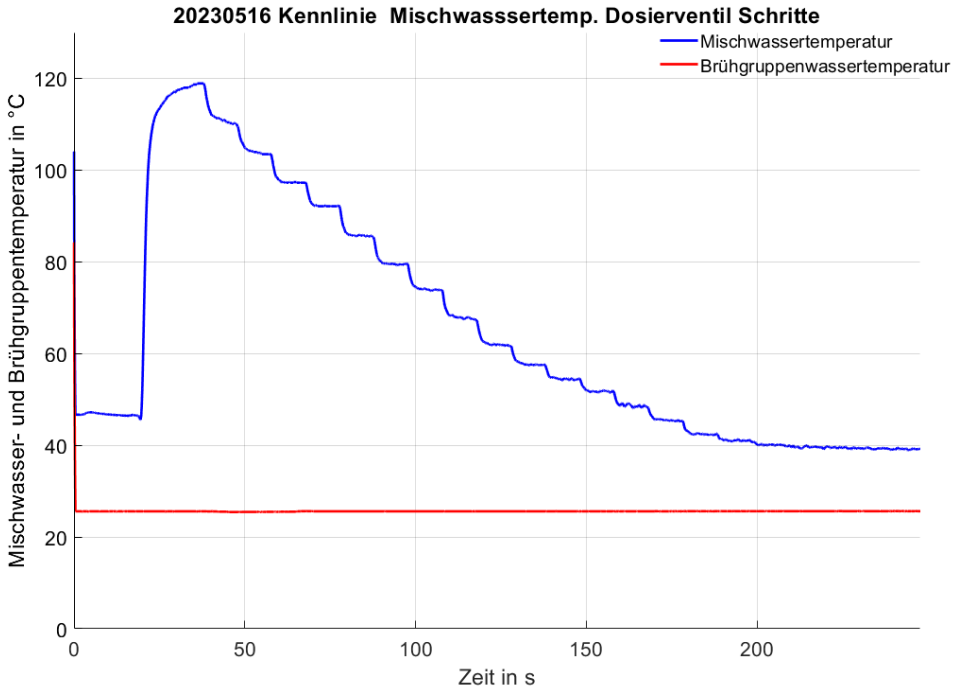

Abbildung 8: Temperaturverlauf des Mischwassers (blau) über der Versuchsdauer

Jeder Temperatursprung in Abb. 8 entspricht einem Verfahren des Dosierventils Heiß-/Kaltwasser um 40 Halbschritte. Bei 560 möglichen Halbschritten (Angabe AVS Römer) wären somit 14 Temperatursprünge zu erwarten.
Tatsächlich entstehen aber ca. 16.5 Sprünge, womit diese Aussage möglicherweise widerlegt wurde.
Dieser Aspekt hängt höchstwahrscheinlich mit dem Problem zusammen, dass die Dosierventile von AVS Römer in Öffnungsrichtung keine direkte mechanische Begrenzung besitzen. Obwohl das Ventil also für 560 Halbschritte ausgelegt ist, ist ein größerer Verfahrweg möglich.
Diese Thematik führt außerdem zu dem Problem, dass das Dosierventil Heiß-/Kaltwasser dazu tendiert, sich in der offenen Endposition „auszuhängen“. Bei einem Zurückfahren gehen dann Schritte verloren, bis das Ventil sich wieder „eingefädelt“ hat.

Es muss also von einer Folgegruppe ermittelt werden, bei welcher Schrittanzahl ein Aushängen stattfindet. Damit wird dann umgekehrt der endgültige „sichere“ Verfahrweg der Ventile bestimmt. Danach ist die Initialisierung auf diese Schrittanzahl zu reduzieren (800 → x Halbschritte).

Zudem wird Stand jetzt in dieser Initialisierung definiert, dass bei einer erstmaligen Schrittmotoransteuerung nach einem Neustarten/Anschalten der Maschine immer die erste Sequenz („seq1“) gestellt wird ~~(Eine genauere Erklärung ist in der Programmcode-Kommentierung enthalten)~~.
Diese Annahme passt aber nicht zum realen Bild, bei dem der Schrittmotor vor dem Ausschalten/Neustarten der Kaffeemaschine in jeder beliebigen Sequenz stehengeblieben sein kann. Dadurch ist ein Springen des Schrittmotors zu Beginn quasi garantiert.
Um dies zu verhindern, wäre z.B. eine Ausschalt-Routine denkbar, die sämtliche Schrittmotoren in eine Mittelposition fährt und dabei darauf achtet, dass als letzte Stellsequenz „seq8“ gefahren wird. Dadurch wird auch das Problem gelöst, dass der Schrittmotor bei längerem Halt in einer Endlage dort „verklebt“.

Abschließend liegt noch die Problematik vor, dass beim Erreichen einer Endlage (vor allem bei einer vollständigen Schließung der Drossel) der Schrittmotor physisch logischerweise nicht mehr weiterfährt, die Stellsequenzen aber weiter durchlaufen werden (da immer die aus MATLAB® übergebene Halbschrittanzahl gefahren wird). Auch hieraus ergibt sich die Problematik, dass der Schrittmotor anfängt zu springen, sobald er aus dieser Endlage hinaus dann erneut angesteuert wird. Dadurch gehen Schritte verloren, was vor allem in einer optimierten Initialisierung nicht erlaubt ist.
Es muss erarbeitet werden, ob dieses Problem z.B. durch eine geschicktere Ansteuerung lösbar ist.
Ansonsten muss eine Alternative definiert werden (z.B. ein Erhöhen der übermittelten Halbschrittanzahl um einen Sequenzdurchlauf (+8), sofern bekannt ist, dass in eine Endlage gefahren wird. Damit wird ein eventuelles Springen des Schrittmotors kompensiert, aber die Stellgenauigkeit leidet natürlich darunter).

Alle Probleme und Folgeaufgaben in ihrer genauen Ausführung sind dem ToDo [Wiki: „Systemsoftware“ / „Schrittmotorensteuerung Mischer“] zu entnehmen.

~ *Hendrik Wegjan*

9. Neuprogrammierung Systemsoftware

Die MATLAB® GUI ist die graphische Oberfläche, die ein Entwickler im Labor sieht und bedient. Die eigentliche Funktionsweise der App wird jedoch in dem zugrundeliegenden Programmcode festgelegt. Dieser Code bildet die Schnittstelle zwischen der Oberfläche und der Systemelektronik der Laborsiebträgermaschine. [41]

Zum Zeitpunkt des Projektbeginns im März 2023 lag bereits eine MATLAB® GUI mit entsprechendem Programmcode für zwei Platinen vor. [85] Wie in Kapitel 10: Neugestaltung MATLAB®-GUI beschrieben, war eine Neugestaltung der Oberfläche zur Optimierung der Bedienbarkeit und eine Anpassung an den Hardwarestand [64] erforderlich. Der bestehende Code musste an diese Oberfläche angepasst werden. Zusätzlich stand der Umbau auf die Multi-MCU (s. Abb. 1) bevor. Diese sollte das Problem des Verlustes der Verbindung von STM32 und MATLAB® verhindern. [Wiki: „Systemsoftware“ / „Software-Bugs“, BUG-001]. Im Code mussten somit die Vorbereitungen für den Umbau angelegt werden. In diesem Zuge wurde der Code darüber hinaus vereinfacht, neu strukturiert und verbessert. Dies erschien aufgrund der hohen Fluktuation der Projektbearbeitenden sinnvoll.

~~Die Bestellung der Platinen erfolgte schneller als zunächst angedacht, somit wurde der neuprogrammierte Code direkt für die Multi-MCU befähigt. Aus demselben Grund wurde statt des vorläufigen Konzeptes mit vier Platinen (SSR-, BAS-, MWP-, RPI-Platine für Schrittmotoren) bereits im Sommersemester 2023 das geplante Konzept mit drei Platinen (s. Abb. 1) eingeführt.~~

Vor dem Hardware-Umbau wurden mittels drei separater Raspberry Pi Picos die programmierten Codeabschnitte nacheinander integriert und bis zum fehlerfreien Ablauf getestet. Mit diesem Vorgehen sind Fehler schnell zu ermitteln und zu korrigieren. Nach dem Umbau erfolgten die Testung und Integration mit den neu erstellten Platinen direkt an der Laborsiebträgermaschine. Die Reinigung der Maschine, Anpassung der Schrittmotoren, Kaffee- und Wasserbezug wurden gesondert als Teilprogramme geschrieben und getestet. Es wurde geplant, diese in den neuprogrammierten Code zu übernehmen und erneut zu testen.

Sobald der Bediener die App startet, werden im Hintergrund alle nötigen Kalibrierkurven der NTCs [Wiki: „Systemsoftware“ / „Messwerte erfassen Multi-MCU“, Eintrag vom 16.06.2023] und die Eingaben im Reiter „Start“ vor der letzten App-Schließung geladen. Zudem werden die Schalter im Reiter „Manueller Modus“ ausgegraut und sind zu diesem Zeitpunkt nicht bedienbar. Dies garantiert, dass keine Funktionen abgerufen werden, die entweder noch nicht initialisiert sind oder überwacht werden müssen. Im Reiter „manueller Modus“ werden für alle Ventile die Start-Icons im Zustand „geschlossen“ eingefügt, sodass die Stellung der Ventile graphisch angezeigt wird. Die mechanische Stellung erfolgt im späteren Programmverlauf. Die zuvor graue Error-Lampe beginnt grün zu leuchten.

Um die Maschine zu bedienen, müssen zunächst die Platinen verbunden und initialisiert werden. Dies wird durch Betätigung des Verbinden-Buttons in der Oberfläche initiiert. Es wird eine Liste aller belegten Ports und eine unidentifizierte MCU inklusive Verarbeitungsgeschwindigkeit, Terminator (Zeilenumbruch) und Verarbeitungsfunktion für den ersten Port angelegt. Durch Aufruf der ident()-Pythonfunktion mittels eines writeline-Befehls [73] wird von der Platine das hinterlegte Identifikationskürzel als Antwort gesendet [Wiki: „Systemsoftware“ / „Grundfunktionen der MCU und mehrere MCUs“, Eintrag vom 28.02.2022). Die Antwort löst die zuvor festgelegte Verarbeitungsfunktion init_mcus() aus. Der writeline-Befehl und die gesendeten Antworten sind für den Bediener in den allgemeinen Statuszeilen zu beobachten.

Die init_mcus()-Funktion identifiziert das gesendete Antwortkürzel und weist der dadurch identifizierten Platine einen neuen Namen und Verarbeitungsfunktion zu. Ab diesem Moment werden alle gesendeten writeline-Befehle und Antworten dieser Platinen in dem jeweiligen Statusfeld angezeigt. Anschließend wird mit der Initialisierung der Platinen in einer separaten Funktion begonnen. Um gegebenenfalls die Ansprache des nächsten Portes zu ermöglichen, wird nach abgeschlossener Initialisierung die hochzaehlen()-Funktion aufgerufen. Wenn stattdessen bereits alle MCUs initialisiert worden sind, wird dies dem Benutzer durch eine Statusmeldung in der allgemeinen Statuszeile angezeigt.

In der hochzaehlen()-Funktion wird für den nächsten Port der Liste, wenn dieser vorhanden ist und noch nicht bearbeitet wurde, eine unidentifizierte MCU angelegt und erneut durch die ident()-Funktion die init_mcus()- Funktion aufgerufen.

Während der Initialisierung werden durch writeline-Befehle [71] alle benötigten Python-Programme implementiert, Python-Funktionen ausgeführt und mechanische Initialisierungen vorgenommen. Auf der MWP-Platine wird das Programm zur Messwerterfassung implementiert. Auf der BAS-Platine werden die Messwerte, die Pumpensteuerung sowie die beiden Schrittmotorentreiber der Brühgruppendrossel und des Pumpenbypasses initialisiert. Letztere werden initialisiert (s. Kapitel 8: Schrittmotoren). Auf der SSR-Platine wird der Schrittmotortreiber des Dosierventils und die Ventile initialisiert, welches eine Initialisierung der Pins in einem Vektor beinhaltet. Die Ventile werden während der Initialisierung alle geschlossen und so in einen gesicherten Zustand gebracht. Dies geschieht je Ventil über den Aufruf einer separaten Funktion, die den zuvor initialisierten Vektor verwendet.

```
writeline(app.ssr_platine, `ventil_schalten.ventil_off(a[0])´);
```

Nach abgeschlossener Initialisierung werden die Schalter im Reiter „manueller Modus“ in der GUI nacheinander sichtbar und bedienbar geschaltet, sobald es die sichere Bedienung der Maschine zulässt.

Die während der Initialisierung definierten Verarbeitungsfunktionen werden aufgerufen, sobald der Terminator erkannt wird, also eine Antwort gesendet wurde. Sendet die SSR-

Platine eine Antwort, wird darin nach dem Signalwort „endDosier“ gesucht. Dieses signalisiert die Beendigung des Stellvorganges des Dosierventil-Schrittmotors. Wenn dies gefunden wird, wird der Schrittmotor stromlos geschaltet. In ähnlicher Weise wurde diese Verarbeitungsfunktion für die BAS-Platine programmiert. Der einzige Unterschied besteht in den gesuchten Signalwörtern „endDrossel“ (Brühgruppendrossel) und „endBypass“ (Pumpenbypass). Zusätzlich werden auf der BAS-Platine Messwerte verarbeitet. Die Verarbeitungsfunktion unterscheidet dabei einkommende Messwerte (Erkennungszeichen "B!") von Fehlern. Der Benutzer ist in der Lage, anhand der Error-Lampe zu erkennen, ob es zu einem Fehler gekommen ist. Die Lampe würde im Fehlerfall rot leuchten. Werden Messwerte erkannt, wird die messwerte_verarbeiten()-Funktion zur weiteren Verarbeitung der Messwerte aufgerufen und die Error-Lampe leuchtet grün. Die Verarbeitungsfunktion der MWP-Platine ist analog zur Messwertverarbeitung der BAS-Platine aufgebaut mit einzigem Unterschied, dass dort das Signalzeichen „M!“ verwendet wird.

Die Verarbeitungsfunktionen der MWP-Platine und BAS-Platine rufen bei einkommenden Messwerten dieselbe Funktion zur weiteren Verarbeitung der Messwerte auf. In dieser wird der Zeitstempel der jeweiligen Messung in Sekunden in einer neuen Zeile der Matrix abgespeichert. Die Messwerte werden in dieselbe Zeile in den Messwertpuffer gespeichert und umgerechnet. Für die NTC werden die zuvor importierten Kalibrierkurven [Wiki: „Systemsoftware“ / „Messwerte erfassen Multi-MCU“, Eintrag vom 16.06.2023) verwendet.

Wenn der Messwertpuffer die im Programmcode definierte Länge überschreitet, wird sowohl dieser als auch die Zeitstempelmatrix gekürzt. Anschließend wird eine separate Funktion zur Anzeige der Messwerte aufgerufen, welche die Messwerte kontinuierlich auf der Oberfläche angezeigt und etwa jede Sekunde aktualisiert. Letztendlich wird in der Messwertverarbeitungsfunktion gegebenenfalls eine erste Datensicherung (bei erster Messung) oder eine automatische Datensicherung (wenn der Zeitabstand den auf der Oberfläche editierbaren Wert überschreitet) ausgelöst.

Bei der Datensicherung handelt es sich um eine kontinuierliche Sicherung der ermittelten Messdaten. Der Zeitabstand der Datensicherung wird im Eingabefeld eingegeben und ist mit 120 Sekunden vorbelegt. Die Datensicherung umfasst die letzten xx Sekunden der erfassten Messdaten. Bei jedem Schließen der GUI wird ebenfalls eine Datensicherung vorgenommen. Diese Datensicherung ist unter dem Dateinamen „last_data.mat“ zu finden und wird bei jeder Sicherung überschrieben. Zusätzlich ist eine manuelle Datensicherung durchführbar. Diese legt die Daten in einer anderen Datei ab. Dabei wird der Dateiname aus Datum und Uhrzeit gebildet, sodass die Daten wieder einem Ereignis zuordenbar sind, die Dateinamen eindeutig bleiben und es zu keinen Überschreibungen von erfassten Messdaten kommt. Der Dateiname hat das Format „YYYYMMDD_Thhmmss.mat“ (ISO-Datum und Zeitangabe ohne Zwischenzeichen). Als Meta-Daten werden alle editierbaren Eingaben des Start-Reiters und eine Messpunktliste abgespeichert. Die mat-Datei verfügt über die Messdatenmatrix und einen Zeitvektor.

Die Schaltung der Aktoren erfolgt durch writeline-Befehle [73], welche von MATLAB® an Python geschickt werden. Die Befehle müssen im MATLAB®-Code an den Python-Code angepasst werden und werden an die jeweils für den Aktor vorgesehene Platine gesendet.

Während der Einarbeitung machte sich bemerkbar, dass das Verständnis der Funktionen zur Stellung der Aktoren einige Zeit bedarf. Um diese zu verkürzen, wurde eine Vereinfachung vorgenommen. Die Stellung erfolgte Stand Januar 2023 im manuellen sowie automatischen Modus über eine Änderung des Soll-Wertes. Am Ende des gerade ablaufenden Programmes wurde ein Soll-Ist-Vergleich aufgerufen, welcher alle Aktoren bei mindestens einer festgestellten Abweichung stellt. [85]

Um einen unübersichtlichen, großen writeline-Befehl zur Stellung aller Aktoren zu vermeiden, wurde der Soll-Ist-Vergleich im automatischen Modus beibehalten. Jedoch wird bei festgestellter Abweichung eines Aktors dieser sofort gestellt. Der Vergleich wird vorerst beibehalten, da ein Direkt-Aufruf in den Funktionen der Programme und Regler viele Konsequenzen nach sich ziehen würde, die in der Praxis getestet werden müssen. Da der Schwerpunkt dieser Projektarbeit zunächst auf der Implementierung der Multi-MCU lag, bleibt dies ein offenes Thema für zukünftige Projektgruppen.

Im manuellen Modus wurde dagegen direkt in die Callbacks bei Wertänderung die Neustellung einprogrammiert. Somit erfolgt die manuelle Stellung Stand August 2023 nicht mehr über einen Soll-Ist-Vergleich. Für die Ventile wurde der Wechsel des Icons bei Mausklick auf das Bild im Reiter „Manueller Modus“ im manuellen Modus hinzugefügt. Dieser bewirkt die Negierung (Umkehrung) der Stellung des Ventils, welches durch eine Flag detektiert wird.

Da auch diese Stellung, vor allem im automatischen Modus, viele Unterfunktionen und somit Unübersichtlichkeit bereithält, wurde die Aktorenstellung neu aufgesetzt. Es wurde je eine Funktion für Heizung, Pumpe und Dosierventil erstellt, die zukünftig an allen Stellen – automatischer und manueller Modus - aufgerufen werden, die einer Stellung bedürfen. Diese Funktionen sind bereits im Januar 2023 vorhanden gewesen [85]. Anders verhält es sich mit der Ventilschaltung, diese wurde vollständig neu programmiert. Da es jedoch zu einer Zerstörung der Platinen vor Abschluss der Testung dieser Schaltung kam, wurde sich entschieden, die funktionierende und getestete, angepasste Schaltung mit Soll-Ist-Vergleich zu übernehmen. Die ungetesteten Funktionen wurden auskommentiert und ebenfalls übergeben.

Die Schaltung der Ventile erfolgte im Januar 2023 über die Festlegung der Sollwerte und anschließende Verarbeitung im Soll-Ist-Vergleich. [85]

Beispielhafter Aufruf: app.ventileSollIntern=[0 1 0 1 0 0 0 0 0 0];

Die Verwendung eines Vektors erwies sich als umständlich. Einerseits fällt die Zuordnung der Ventile schwer. Andererseits verhindert dieser Vektor eine einfache Anpassung an eine andere Ventilanzahl. Es wird stets eine bestimmte Elementanzahl erwartet und

übergeben, die bei Veränderung an jeder aufgerufenen Stelle im Code ebenso verändert werden muss. Daher wurde von der Verwendung eines Vektors abgesehen und sich für einen namentlichen Aufruf der Ventile entschieden. Die neue Funktion ventile_stellen() ist in der Lage eine unbekannte Anzahl Ventilnamen laut Hydraulikplan [64] zu verarbeiten. Die doppelten Anführungszeichen werden benötigt, da bei der Übertragung an Python eines gelöscht wird (Indikator String).

Beispielhafter Aufruf: app.ventile_stellen("‘Y01‘", "‘Y13‘", "‘Y07‘");

Im automatischen Modus werden alle Ventile, die in dieser Funktion genannt werden, geöffnet, alle nicht genannten geschlossen. Eine Übertragung dieser Funktionsweise auf den manuellen Modus ist jedoch nicht möglich, ansonsten werden während der Schaltung eines Ventils (Drücken auf Ventil-Icon) zeitgleich alle anderen Ventile mitgeschaltet. Im manuellen Modus werden dahingehend die übergebenen geschlossenen Ventile geöffnet und andersherum. Die Funktion ist in der Lage zwischen automatischem und manuellem Modus zu unterscheiden.

In Python wird dazu zunächst eine Matrix mit den Ventilnamen und den zugehörigen Pin-Nummern in einer Zeile definiert, die während der Initialisierung aufgerufen wird. Die Schaltung der Ventile im automatischen Modus erfolgt durch Überprüfung, welche namentlich übergebenen Ventile auftauchen und welche Pin-Belegung diese besitzen. Diese Ventile werden geöffnet. Nach der Ermittlung aller nicht auftauchenden Ventile (inklusive Pin-Belegung) werden diese geschlossen. Die Schaltung im manuellen Modus erfolgt genauso, es werden jedoch lediglich die auftauchenden Ventile detektiert und an eine weitere Funktion übergeben, diese negiert deren Werte. [Wiki: „Systemsoftware“ / „Schalten Magnetventile SSR-Platine Multi-MCU“, Eintrag vom 14.06.2023) Durch dieses Vorgehen muss bei einer Veränderung der Ventilanzahl oder deren Namen lediglich die Initialisierungsmatrix angepasst werden.

Der Icon-Wechsel wurde der Funktion zur Magnetventilschaltung in MATLAB® hinzugefügt. Damit ist vermeidbar, dass mit jedem Funktionsaufruf der Icon-Wechsel angelegt werden muss. Zudem ist es möglich, in zukünftigen automatischen Programmabläufen die Schaltung der Ventile auf der Oberfläche zu beobachten. Im manuellen Modus wird dabei anhand der Abfrage der Statusflag das Bild gewechselt. Im automatischen Modus werden alle Icons der in der Übergabe enthaltenden Ventile zur geöffneten Position gewechselt, alle anderen zur geschlossenen. Somit wird auch der Aufruf vereinfacht und ist im automatischen und manuellen Modus identisch.

Beispielhafter Aufruf: app.ventile_stellen("'Y01'");

Bei Beendigung der App wird die in der Datensicherung ~~erwähnte~~ Datei „last_data“ erstellt, welche die editierbaren Eingaben der Oberfläche enthält, die bei Neustart der App geladen werden.

Der generelle Aufbau des Programmcodes wurde in einzelne Unterüberschriften unterteilt, sodass zukünftigen Bearbeitern die Zugehörigkeit der Funktionen verständlich wird. Diese lauten: Initialisierung und Verarbeitung MCUs, Programm- und Regleraufruf, Regler, Aktorenstellung. Es ist zu beachten, dass Programm- und Regleraufruf sowie Regler nur an den Programmcode angepasst wurden, ihre Funktionsweise jedoch nicht verändert wurde.

Von zukünftigen Bearbeitenden sind noch Verbesserungen vorzunehmen. Eine Statusmeldung am Ende der Initialisierung, dass diese beendet wurde, ist nicht der letzte Eintrag in der Statusleiste. Hier muss eine künstliche Verlangsamung vorgenommen werden. Die Initialisierungsfunktion init_mcus() wird zusätzlich häufiger als 4x aufgerufen. Dies liegt daran, dass durch den Import des ident-Programmes, vor Aufruf der ident()-Funktion, bereits ein Terminator gesendet wird.

Die im Reiter „manueller Modus“ angelegten Schalter sind noch nicht auf Funktionalität überprüft worden. Die Messwerte starteten beim letzten Testdurchlauf des Programms automatisch mit der Initialisierung der MWP-Platine. Dies darf erst durch Betätigung eines Schalters geschehen. Zudem ist das Signalwort der einkommenden Messwerte nicht gleichermaßen definiert, dies muss sowohl in MATLAB® als auch in Python angepasst werden (M! und B!).

Die Abspeicherung der Messwerte in den Messwertpuffer erfolgt nach einer nicht mehr passenden Logik. Zuvor wurden alle Messwerte von der STM32 gesendet, wodurch die Reihenfolge der Speicherung der Messwerte von BAS-Platine und MWP-Platine durcheinander erfolgt. Da eine Änderung Auswirkungen auf die Regler und Programme hat und diese thematisch nicht Teil der Projektgruppe SoSe2023 waren, wurde dies nicht angepasst. Die Anzeige der Messwerte ist noch fehlerhaft beziehungsweise es ist kein entsprechendes Anzeigelabel mehr in der Oberfläche vorhanden (Messwert 11-15). Hier muss die Sinnhaftigkeit einer Anzeige dieser Messwerte überprüft werden.

Es muss zudem überprüft werden, ob es während einer Verarbeitungsroutine der Messwerte durch eine neu gesendete Antwort von MWP- oder BAS-Platine zu Fehlern kommt. In der Praxis ist dies nie aufgetreten, der Code bietet jedoch die Option.

Die Schaltung der Heizung ist aktuell fehlerhaft. Der Aufruf muss als PWM-Signal übermittelt werden. Die Zuordnung der Ventile erwies sich ebenso als inkorrekt, sodass manche Ventile nicht oder falsch angesteuert wurden. Die angepasste Ventilschaltung, insbesondere der Icon-Wechsel mit vielen langen, nahezu identischen Codeabschnitten, bietet Potentiale zur Verbesserung und Vereinfachung. Zusätzlich müssen die geänderten Stellungsaufrufe in den Programm- und Regleraufruf eingepflegt werden.

Der ResetError-Button sowie der Notaus-Knopf haben aktuell noch keine Funktion. Außerdem wurde die Funktionen zum Durchrollieren der Antworten [85] entfernt, da die neue Oberfläche nur eine Statuszeile je Platine und eine allgemeine verfügt.

Aufgrund der Zerstörung der Platinen bedurfte es einer ausführlicheren Übergabe, sodass das erstellte Reinigungsprogramm (s. Kapitel 7: Spülvorgänge) nicht implementiert wurde. Dieses muss eingefügt und nochmals getestet werden. Aus denselben Gründen wurde ein automatischer Kaffee- und Teewasserbezug nicht realisiert. Diese Bezüge werden jedoch ähnlich dem zuvor erwähnten Reinigungsprogramm ablaufen, womit eine Orientierung daran möglich ist.

~ Melina Scherf

10. Neugestaltung MATLAB®-GUI

Die MATLAB®-GUI ist die graphische Bedieneroberfläche der Laborsiebträgermaschine. Die in der GUI durch den Anwender eingegebenen Befehle, beispielsweise das Schalten eines Ventils, werden über den im Hintergrund laufenden MATLAB®-Programmcode an eine Schnittstelle weitergegeben. Diese Schnittstelle steuert die Multi-MCUs an, die direkt auf die Maschine zugreifen.
Als Anwender gelten Studierende, sowie Lehrende, die im Rahmen der „Geschmackssache Kaffee“ an der Labormaschine arbeiten; also Testversuche fahren oder die Maschine weiterentwickeln. Die MATLAB®-GUI ist demnach nicht für den Endanwender der Maschine gedacht. Das verkaufsfähige Endprodukt soll, wie auf dem Markt üblich, mit Schaltern und Knöpfen ausgestattet sein.

Zum Zeitpunkt des Projektstartes lag bereits eine MATLAB®-App inklusive GUI vor [85]. Statt einer Einarbeitung und Weiterentwicklung in den bestehenden Code, erschien eine vollständige Neuaufsetzung der MATLAB®-App und GUI sinnvoller.

Im Hinblick auf die oben genannten Anwender wurde bei der Zielsetzung die Erleichterung der Arbeit im Labor priorisiert. Da die studentischen Arbeitskräfte zum Großteil einer hohen Fluktuation unterliegen, wurde außerdem auf eine intuitive Bedienbarkeit und eine hohe Übersichtlichkeit geachtet. Dadurch soll das Einfinden in die MATLAB®-GUI erleichtert, und folglich die Einarbeitungszeit verkürzt werden.

Sowohl die alte als auch die neue MATLAB®-GUI ist in Tabs unterteilt. Diese gliedern sich in die Startseite, einen manuellen Modus und einen automatischen Modus für Bezugsprogramme. Die neue MATLAB®-GUI beinhaltet zusätzlich einen Tab für das Reinigungsprogramm, welches im Rahmen der Projektarbeit entstanden ist.
Die bisherige MATLAB®-GUI weist zudem Verbesserungspotenzial auf. So war der gesamte Laptop-Bildschirm durch die GUI ausgefüllt. In der neuen GUI wurde bewusst ein Format von 1720 x 880 Pixeln gewählt, sodass wahlweise am oberen oder unteren Rand der GUI ein im Hintergrund liegendes Programm sichtbar ist. Im Falle der Weiterentwicklung der Systemsoftware könnte dies das Command Window von MATLAB® sein, sodass Fehlermeldungen sofort zur Kenntnis genommen werden.
Ebenfalls neu ist, dass die Tabs nicht die gesamte Oberfläche der GUI einnehmen, sondern es einen immer sichtbaren Bereich gibt. Dieser bringt vier Vorteile mit sich. Erstens wird nicht für jeden Tab ein eigener Button für das Speichern des Messwertepuffers benötigt. Stattdessen gibt es diesen Button ein einziges Mal, aber in dem Bereich, auf den ein Zugriff jederzeit möglich ist. Ein zweiter Vorteil ist, dass die für das Geschmackserlebnis besonders relevanten Messwerte Leitungsdruck [mbar] und Durchflussrate [ml / s] ständig sichtbar sind. Auch der Error-Button, sowie der zugehörige Reset-Error-Button, ist in allen Tabs relevant. Die Verschiebung in den immer sichtbaren Bereich brachte auch hier eine erhöhte Reaktionsgeschwindigkeit auf Probleme mit sich, worin der dritte Vorteil liegt. Bei der Reaktion auf Probleme ist auch der Notaus-Button von Bedeutung. Hier können durch

schnelles Handeln Schäden an Menschen und Maschine vermieden werden. Ebenfalls in dem immer sichtbaren Bereich ist der jeweilige Programmstatus der einzelnen Platinen. Das hat den Vorteil, dass immer erkennbar ist, ob diese verbunden sind; nicht nur auf der Startseite.

An der Startseite selbst hat sich kaum etwas verändert. Es gibt hier nach wie vor die Buttons „manueller Modus“, „Verbinden“, sowie „Debug“. Weiterhin gibt es Eingabefelder für Daten wie beispielsweise das Datum, die Kaffeesorte oder den Mahlgrad, die den Messwerten zugeordnet werden. Es ist außerdem einstellbar, wie oft die Daten gesichert werden sollen.

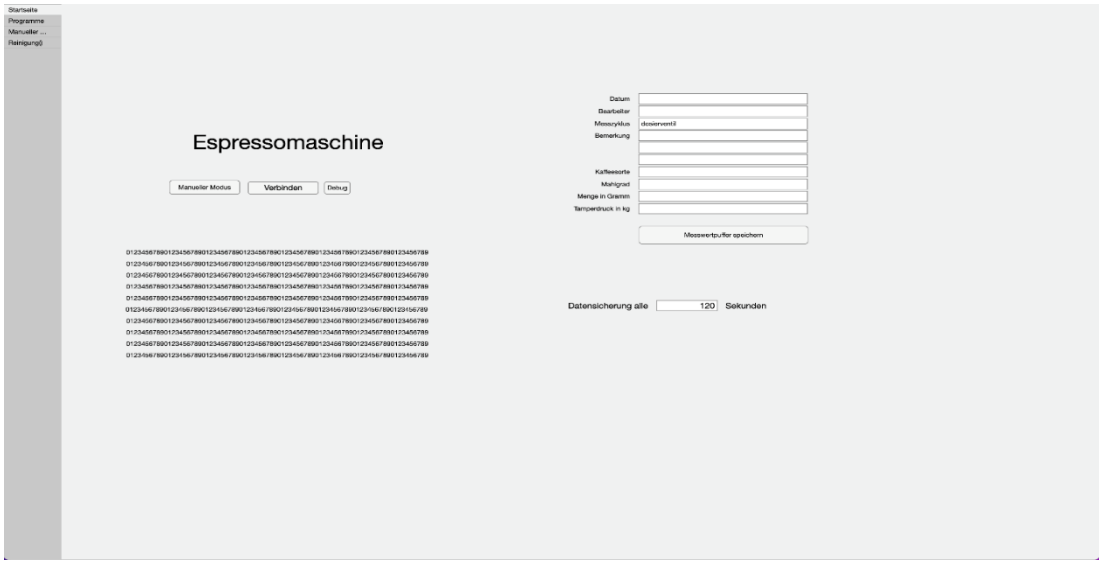

Abbildung 9: Startseite der alten GUI

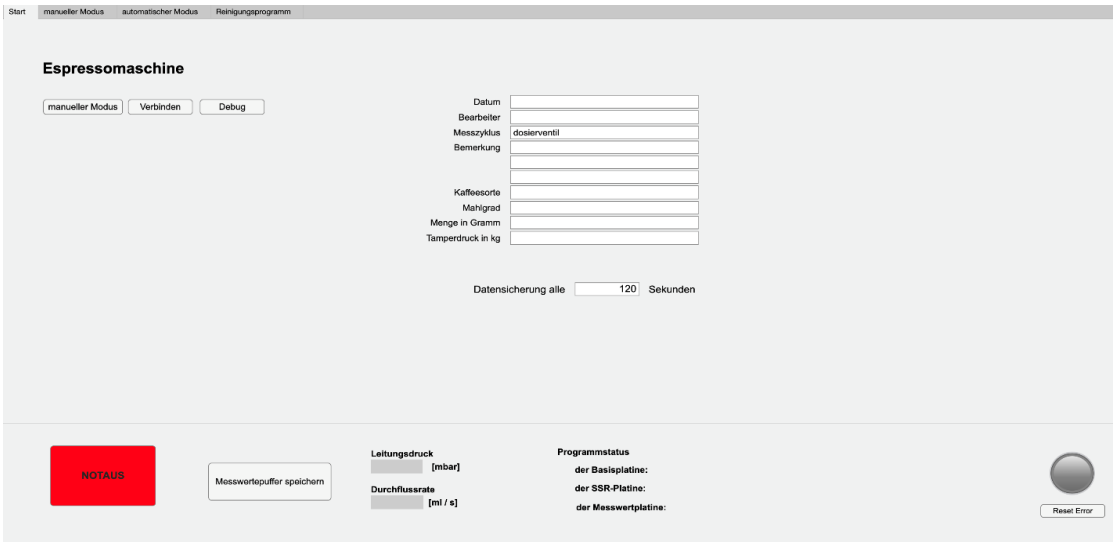

Abbildung 10: Startseite der neuen GUI

Während der Tab „Programme“ in der alten MATLAB®-GUI neben drei Schalter für Kaffeebezug, Teewasserbezug und Dampfbezug auch noch sehr viele andere Inhalte, die nicht direkt den Programmen zuordbar sind, wie beispielsweise die k-Faktoren der Regler, beinhaltet, ist dieser Tab in der neuen GUI minimalistischer gehalten. Da die Programme bisher gar nicht existieren, gibt es hier lediglich vier Buttons als Platzhalter. Ein Programm wird über diese gestartet und gestoppt. Der aktuelle Programmstatus ist an dem jeweils nebenstehenden Lämpchen erkennbar, wobei „rot“ bedeutet, dass das Programm in dem Moment aktiv ist. Während ein Programm durchläuft, sind Ventilschaltungen und Zustandsparameter in dem manuellen Modus nachverfolgbar.

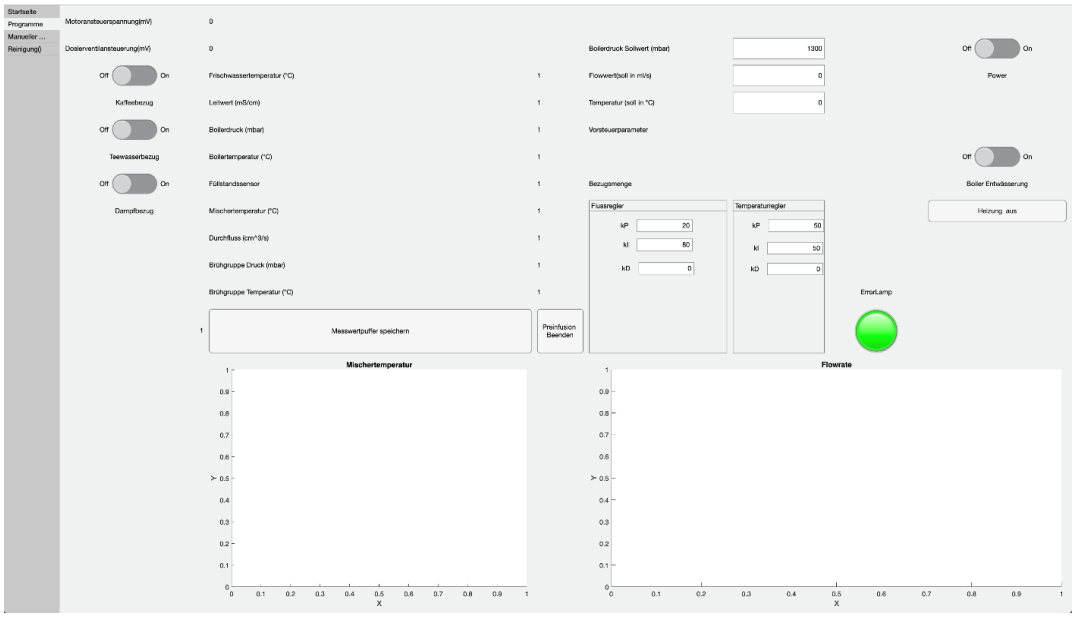

Abbildung 11: Programme der alten GUI

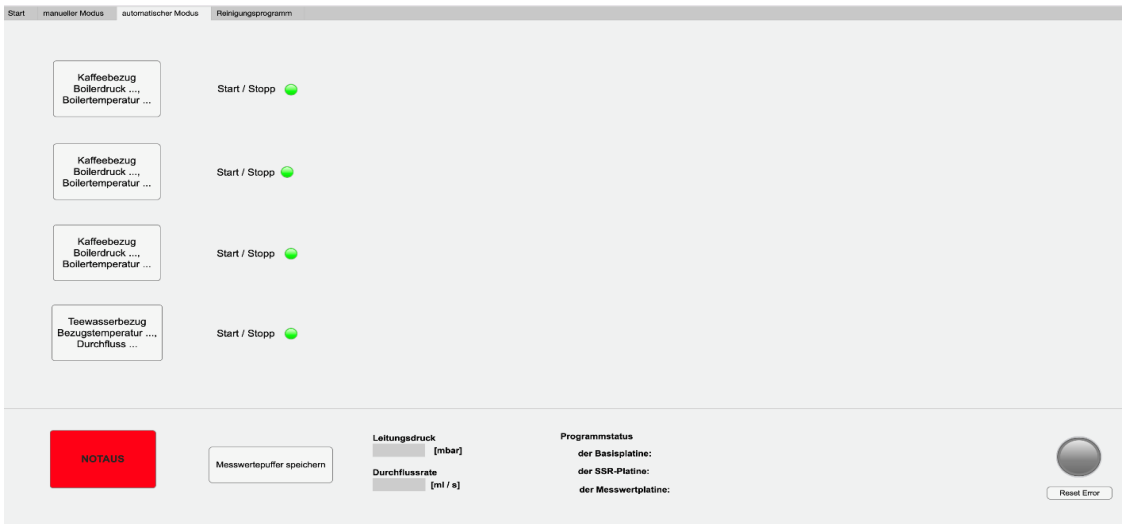

Abbildung 12: Programme der neuen GUI

Ansonsten ist der manuelle Modus primär dazu gedacht, dass Ventile einzeln geschaltet, Parameter einzeln geändert, und dadurch Programme entwickelt und getestet werden können. Das Prozessbild der alten GUI ist sehr detailliert und informativ, dadurch aber leider auch sehr unübersichtlich. Es sind auch die Wege des Brauchwassers hin zur Abtropfschale dargestellt. Dies ist bei Hydraulikplänen unüblich und verbraucht zusätzlichen Platz. Durch die vielen Farben und Formen wird die Aufmerksamkeit des Benutzers immer wieder hin zu anderen Stellen gelenkt. Dadurch, dass die Messwerte recht klein und an wenig intuitiven Stellen dargestellt sind, sind diese schwierig zu finden. Die Schalter für die Grundeinstellungen stehen nicht gesammelt und in keiner sinnvollen Reihenfolge.
In der neuen GUI dagegen sind diese Schalter in schlüssiger Reihenfolge, gesammelt oben links abgebildet. Die Bezeichnungen der Messwerte sind größer und fett gedruckt geschrieben. Die Messwerte selbst sind grau hinterlegt. Die Positionierung der Werte ist im Vergleich zu vorher logisch. Die Mischtemperatur befindet sich zum Beispiel an der Stelle, wo gedrosseltes Kalt- und in der Heizwendel erhitztes Warmwasser zusammentreffen. Die Brühgruppenwerte sind direkt in der angedeuteten Brühgruppe zu finden. Insgesamt ist das neue Prozessbild deutlich ruhiger. Es sind nur die wichtigsten Stränge dargestellt. Die Brauchwasserleitungen sind angedeutet.
Ein Hauptunterschied der neuen zur alten GUI liegt in der Ventildarstellung. Während zuvor durch Lämpchen in den Ventilen signalisiert wurde, ob, beziehungsweise wie, ein Ventil geschaltet ist, wird dies in der neuen GUI durch einen Bildwechsel deutlich. Klickt der Benutzer auf das Ventil im Prozessbild, ändert sich das im MATLAB®-Code hinterlegte Ventilbild und das Ventil wird geschaltet.

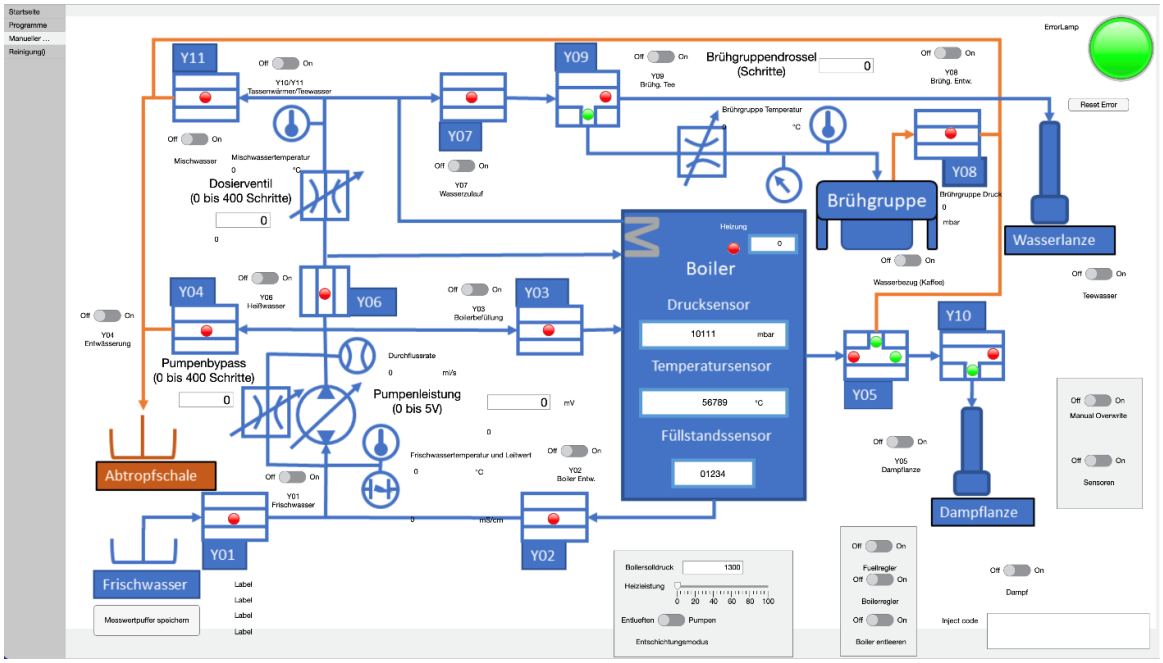

Abbildung 13: manueller Modus der alten GUI

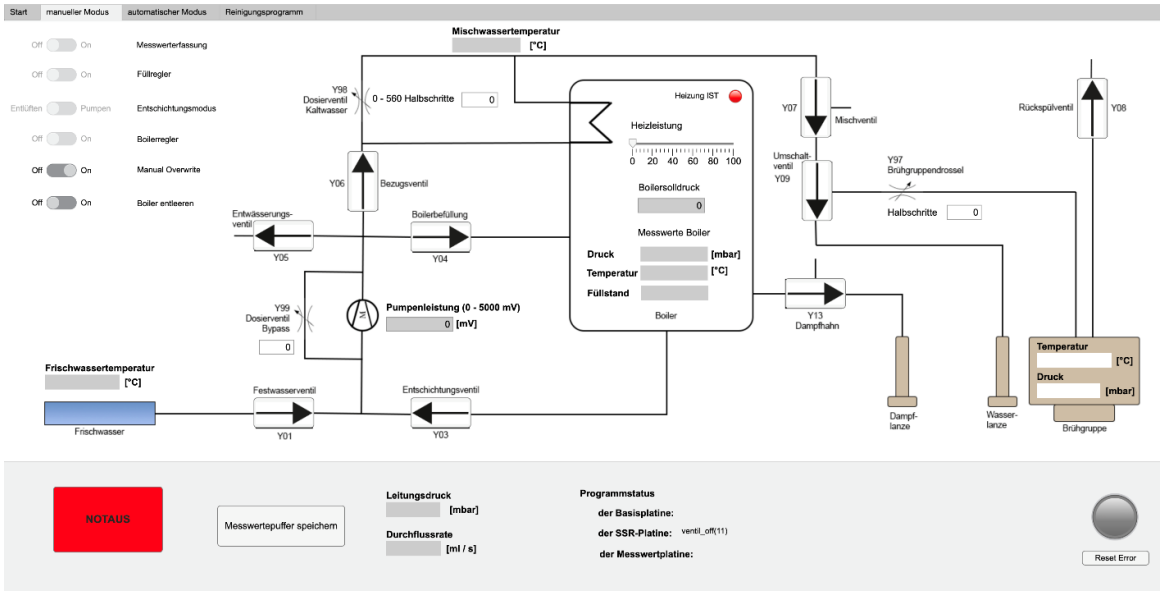

Abbildung 14: manueller Modus der neuen GUI

Da vor Beginn des Projektstartes noch kein Reinigungsprogramm existierte, gibt es den betreffenden Tab lediglich in der neuen GUI. Hier gibt es einen Button zum Start des Programmes, sowie eine Anzeige zum aktuellen Status. Es ist außerdem eine Beschreibung des Ablaufs und Hinweise dazu eingefügt.

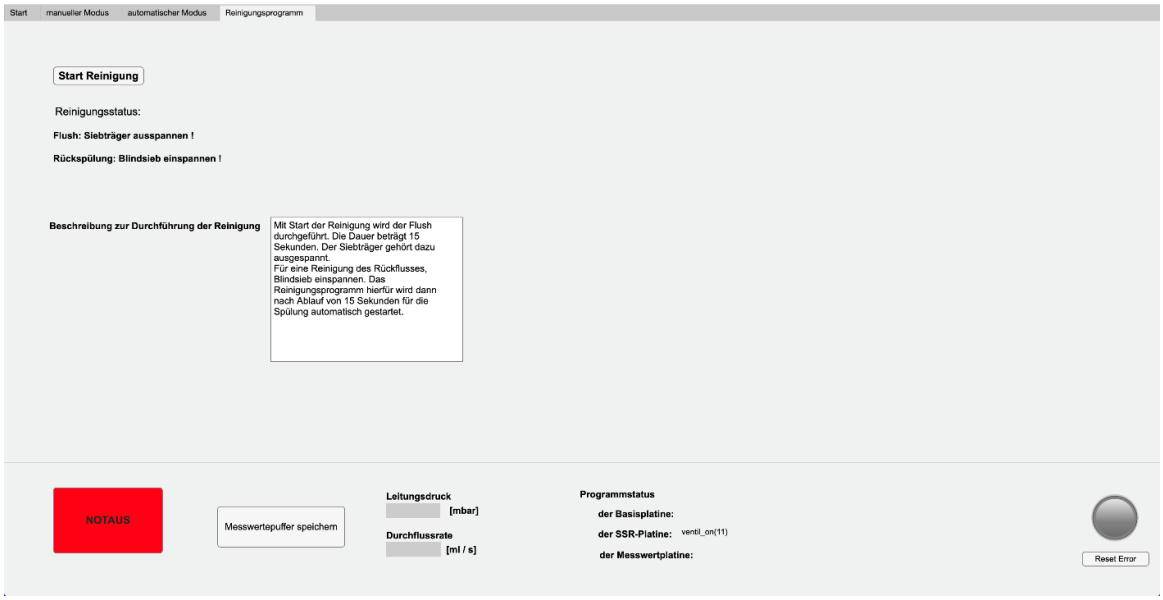

Abbildung 15: Reinigungsprogramm (neue GUI)

Durch die zuvor angesprochenen Maßnahmen ist die neue MATLAB®-GUI übersichtlicher und intuitiver zu bedienen als die alte GUI. Sie ist außerdem auf dem neusten Stand des Hydraulikplans und bildet somit die Labormaschine treffend ab.

~ Madita vom Stein

11. *Zusammenfassung und Ausblick*

Während des Hardware-Umbaus der Elektronik wurde der Leitungsdrucksensor angeschlossen und zeigt die richtigen Spannungswerte an. Aus dem Datenblatt ist der Umrechnungsfaktor zu entnehmen, welcher in die Messwertverarbeitung der MATLAB®-App übernommen werden und getestet werden muss.

Für Misch- und Durchflussregler wurde jeweils eine Möglichkeit zur Auslegung vorgeschlagen, die von einer Folgegruppe durchgeführt werden soll. Hierbei sollen die proportionalen, integrierenden und differenzierenden Anteile des Reglers festgelegt werden.

Zur automatisierten Reinigung der Maschine wurden Programme für Spülvorgänge erstellt. Der richtige Reinigungsablauf wird auf Basis von Druck- und Durchflusswert ausgelöst und von Fehlern unterschieden. Förder- und Wartezeiten müssen zur optimalen Reinigung weiter erprobt werden. Darüber hinaus muss das erstellte Reinigungsprogramm in den Programmcode der MATLAB®-GUI implementiert werden.

Als kritische Pausenzeit der Schrittmotoren wurden 2000 µs ermittelt. Auf dieser Grundlage wurde eine Kennlinie zur Mischwassertemperatur in Abhängigkeit der Dosierventilstellung erstellt. Analog ist zukünftig eine Kennlinie für den Bypass-Drossel-Schrittmotor auszuarbeiten. Es wurde währenddessen eine Abweichung zwischen der Anzahl der vom Hersteller angegebenen und im Labor ermittelten Halbschritte festgestellt. Insbesondere die Endlagen sind daher kritisch zu betrachten, da dort sowohl ein Aushängen und Festkleben als auch ein Überspringen von Halbschritten möglich ist. Um eine problemlose Initialisierung garantieren zu können, wurden daraus zukünftige Maßnahmen abgeleitet.

Der vorhandene Programmcode der MATLAB®-GUI [85] wurde für die Multi-MCU befähigt und optimiert. Die Funktionen wurden entsprechend Abb. 1 auf die Platinen verteilt. In diesem Zuge wurde die Aktoren-Stellung überarbeitet und anwenderfreundlich gestaltet. Da es jedoch zu einer Zerstörung der Platinen vor Abschluss der Tests kam, muss der neu erstellte Code noch erprobt werden. Ein automatischer Kaffee- und Teewasserbezug muss erstellt werden, wobei eine Orientierung am Programm der Spülvorgänge möglich ist.

Simultan mit dem Programmcode wurde auch die GUI neu aufgesetzt. Dabei wurde insbesondere das Prozessbild an den Stand der Maschine angepasst [64] und auf eine benutzer-freundliche Bedienung geachtet. Der Tab für das Reinigungsprogramm wurde ergänzt und der Tab für zukünftige automatische Bezüge angepasst. Sobald die entsprechenden Teilprogramme geschrieben und in den Code implementiert wurden, erfolgt die Integration in die Oberfläche.

~ alle

12. Abbildungsverzeichnis

13. Quellenverzeichnis

Technische Beeinflussbarkeit der Geschmacksache Kaffee
(Link: http://www.institut-fuer-kaffeetechnologie.de/Wiki/index.php?title=Technische_Beeinflussbarkeit_der_Geschmacksache_Kaffee:Literatur)