Fakultät für Maschinenbau, Fahrzeugtechnik, Flugzeugtechnik

Abschlussbericht zur Projektarbeit Inbetriebnahme Pumpenprüfstand

Abgabedatum: 11.02.2022

Betreuer: Dipl.-Ing. Armin Rohnen LbA

Autor	**Mat.Nr.**	**E-mail**
Eric Hübner	03417819	ehuebner@hm.edu
Semih Kum	62058916	kum@hm.edu

Eigenständigkeitserklärung

Ich versichere durch meine Unterschrift, dass ich diese Ausarbeitung mit den praxisbezogenen Unterlagen selbstständig in der vorgegebenen Zeit erarbeitet habe. Alle Stellen, die ich aus Veröffentlichungen entnommen habe, wurden von mir als solche kenntlich gemacht.

Ebenso bestätige ich, bei der Erstellung dieser Dokumentation weder teilweise noch vollständige Passagen aus Ausarbeitungen übernommen zu haben, die bei dem prüfenden oder einem anderen Professor eingereicht wurden.

Eric

München, 11.Februar 2022

Abstract

In dem vorliegenden Bericht geht es um die Inbetriebnahme eines Pumpenprüfstands für die Untersuchung verschiedener Pumpen und Sensoren zur Entwicklung einer Espressomaschine. Hierbei wird insbesondere auf die Themen Hardware, Software, Elektronik und Funktion eingegangen.

The following report is based around the installation of a testing aperture for pumps and Sensors in the development of espresso machines. The topics of hardware, software, electronics and function will be highlighted.

Verzeichnis der verwendeten Formelzeichen

Symbol	Einheit	Beschreibung
U	mV	El. Spannung
p	bar	Druck
q	cm^3/s	Volumenstrom
t	s	Zeit

Abkürzungsverzeichnis

Abkürzung	Bedeutung
SSR	Solid State Relais
GUI	Graphical User Interface
PWM	Pulsweitenmodulation
MCU	Microcontroller Unit

Inhaltsverzeichnis

1. Einleitung und Leistungsvereinbarung

In Zusammenarbeit mit der Firma Turnus GmbH wird von Studenten der Hochschule München im Rahmen der Projektarbeit eine Espressomaschine entwickelt. ~~Geleitet wird das Projekt durch den Hochschuldozenten Armin Rohnen und den Geschäftsführer der Turnus GmbH, Andreas Goclik. Ziel ist es, in einem Zeitraum von März 2021 bis April 2023 eine marktfertige Espressomaschine zu entwickeln.~~

Zur Entwicklung der Espressomaschine zählt unter anderem die Umsetzung eines funktionierenden Pumpenkonzepts, einschließlich Sensorik, Kalibrierung, Regelung und Steuerung von Aktoren, Hydraulik als auch weiteren Bestandteilen. Hierzu ist es wichtig, die genannten Komponenten zu erproben und aufeinander abzustimmen. Für diese Erprobung wurde durch vorherige Gruppen (siehe Dokumentation Stephan Hase [35] und Korbinian Ass [25]) ein Pumpenprüfstand entwickelt, der fertiggestellt und in Betrieb genommen werden soll.

~~Nach Einarbeitung in das Projekt durch die Studenten wurde gemeinsam mit Hr. Rohnen eine Leistungsvereinbarung getroffen.~~ Es wurde vereinbart, dass bis Anfang Dezember 2021 für die Getriebepumpe eine Kennlinie Durchflussrate über Druck ausgelesen wird, hierfür waren die entsprechenden Vorbereitungen zu treffen. Bis zum Ende des Projekts am 11.Februar 2022 soll ein Pumpenprüfstand mit funktionsfähiger Elektronik und Mechanik fertiggestellt werden. Hierzu ist zudem die vorherige Steuerung über ein Raspberry Pico durch eine Basisplatine zu ersetzen. Die Steuerung soll durch eine selbsterklärende MATLAB GUI umgesetzt werden. Die genauen Schnittstellen zur Übergabe des Projekts sind im Kapitel „Übergabeschnittstellen“ definiert.

~~Ursprünglich sollte das Projekt von den 3 Studierenden Semih Kum, Oksana Prusova und Eric Hübner durchgeführt werden, aus verschiedenen Gründen musste Frau Prusova jedoch leider vor Fertigstellung aus dem Projekt ausscheiden, sodass es durch Hr. Kum und Hr. Hübner fertiggestellt wird.~~

Im Folgenden wird zunächst auf die Hardware am Pumpenprüfstand eingegangen. Anschließend werden die Themen Elektronik, Software und Funktion behandelt.

2. Hardware

Eine Espressomaschine besteht aus verschiedensten Hardwarekomponenten. Zur Simulation des Betriebs einer Espressomaschine wurde durch eine vorherige Gruppe ein Gestell konstruiert [25], das die Erprobung verschiedener Pumpen und Sensoren im Zusammenspiel mit anderen Hardwarekomponenten wie Boiler, Schläuchen und Ventilen erlaubt. Einige Baugruppen waren zum Beginn der Projektarbeit schon montiert oder für die Montage vorbereitet, es mussten jedoch noch viele Anpassungen getroffen und Fehler behoben werden.

Das Arbeitspaket Hardware enthält alle mechanischen Aufgaben am Prüfstand. Hierzu gehören die vorbereitenden Tätigkeiten zur Befestigung der DC-Antriebe und der Pumpen am Gerüst, sowie die Optimierung des Gehäuses für den Einbau der Basisplatine. Der Vorbereitung folgt die Montage aller Antriebskomponenten und der Anpassung der Verrohrung, soweit erforderlich.

2.1 Montage der Antriebskomponenten – Getriebepumpe

Um das erste Ziel der Leistungsvereinbarung (Auslesen einer Pumpenkennlinie für die Getriebepumpe) umzusetzen, war es notwendig, die Getriebepumpe auf dem Pumpenprüfstand zu montieren. Hierzu wurde die ursprünglich für den DC-Antrieb konstruierte Halterung aus Aluminium Montageprofilen umfunktioniert, um die Getriebepumpe zu befestigen. Die Getriebepumpe wurde mit mehreren Kabelbindern befestigt und zwischen Boiler und Flowmeter an den Wasserkreislauf angeschlossen (siehe Abb. 1 und 2). Anschließend wurde die Verbindung zur Basisplatine hergestellt. Genauer wird die Stromversorgung im Kapitel Elektronik beschrieben.

Abbildung 1 Montageposition Getriebepumpe (vorne)

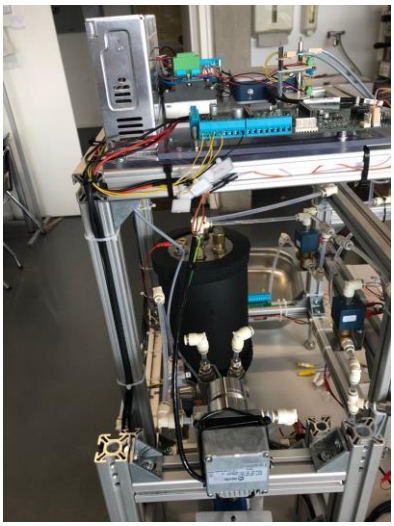


Abbildung 2 Montageposition Getriebepumpe (hinten)

2.2 Montage der Antriebskomponenten – DC-Antriebe

Neben der Getriebepumpe sollen ebenso zwei verschiedene DC-Antriebe (66W und 100W) erprobt werden, die über eine Welle eine am Prüfstand montierte Pumpe antrieben. Hierzu wurde von Stephan Hase eine Halterung konstruiert [35]. Durch fehlende Teile ließ sich der Anbau der DC-Antriebe am Prüfstand nur schrittweise umsetzten. Verschiedene Schrauben, Passstifte und Halterungen mussten durch andere Lösungen ersetzt werden. Drehzahl- und Drehmomentsensoren wurden ebenso angeschlossen.

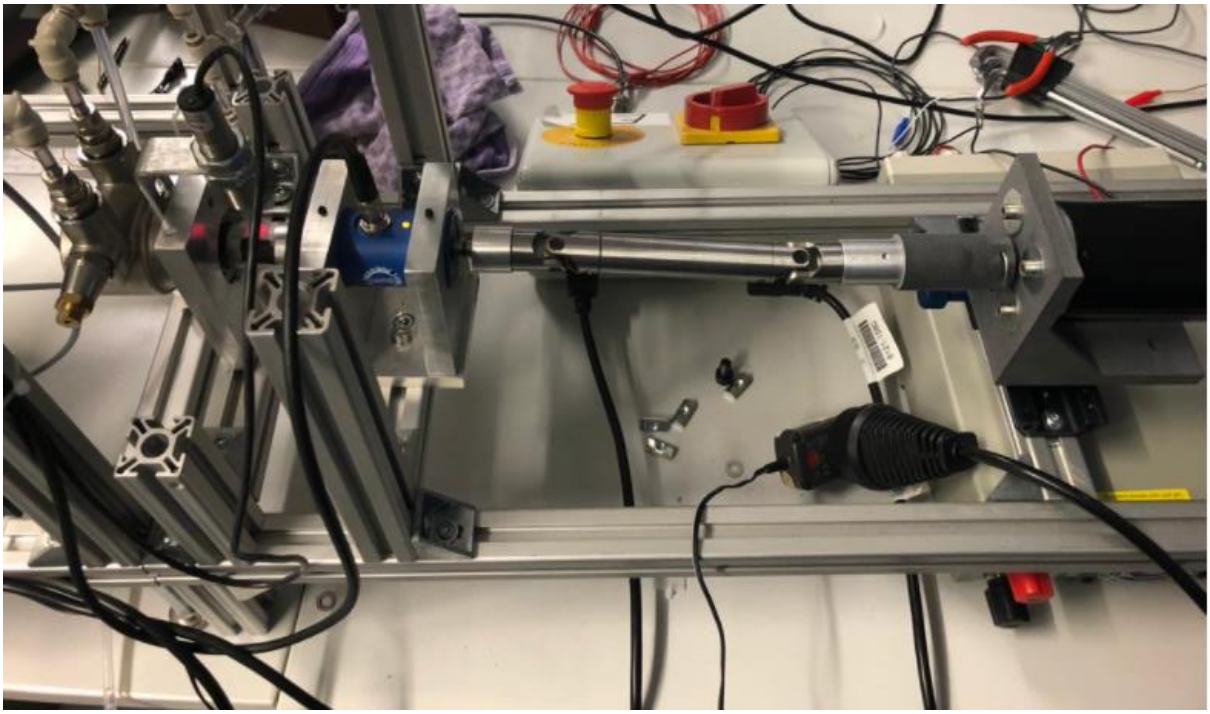


Abbildung 3 Aufbau DC-Antrieb inkl. Pumpe

Die Halterung der DC-Antriebe ist aktuell mit zwei Schrauben oberhalb eines Aluminiumprofils befestigt. Hierdurch wird die Gelenkwelle leicht ausgeknickt. Zur Fertigstellung der Montage ist es notwendig, die Verstellvorrichtung (siehe Abb.3 unterhalb des Antriebs) anzubringen. Die Schrauben in der Welle sind durch Madenschrauben auszutauschen. Die provisorische Befestigung der Welle am Motor durch ein Nagelstück, ist durch einen Passstift auszutauschen.

3. Elektronik

Dieses Kapitel befasst sich mit dem Verbau, der Verkabelung, der Anpassung und Funktion elektrischer und elektronischer Komponenten am Pumpenprüfstand.

Die Energieversorgung und Signalübertragung am Pumpenprüfstand findet elektronisch statt. Zur Steuerung und Auswertung der Aktoren und Sensoren wird unter anderem eine Basisplatine und NI-Messkarte verwendet.

Da die Basisplatine nicht sofort verfügbar war, hat eine vorherige Gruppe eine provisorische Steuerung über ein Schaltbrett mit einem Rapsberry-PICO aufgebaut [25] (siehe Abb. 4). Diese Steuerung sollte jetzt durch die nun vorrätige Basisplatine ersetzt und vervollständigt werden.

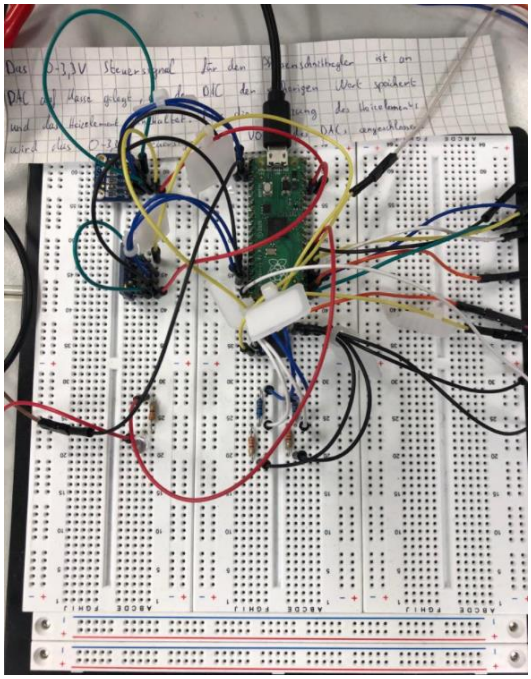


Abbildung 4 Provisorische Steuerung über Raspberry-PICO

3.1 Basisplatine - Einleitung und Aufbau

Zur Befestigung der Basisplatine wurde anfangs eine Plexiglashalterung vorbereitet. Die Halterung wurde an der Oberseite des Prüfstands angebracht. Auf dieser Halterung wurde die Basisplatine an vier Stellen verschraubt.

Die Basisplatine verfügt über eine 24V Anschlussleiste, 2 SSR Leisten, einen möglichen Anschluss für einen DC-Motor sowie ein Display. Über weitere Anschlüsse können Flowmeter, Füllstandssensor und 10V Komponenten angeschlossen werden.

Zusätzlich gibt Anschlüsse für eine Brückenschaltungsplatine, eine NTC-Messplatine und einer Spannungsteilerplatine. Oberhalb der Basisplatine ist eine ST-Nucleo Platine angeschlossen. Über einen USB-Anschluss lässt sich hiermit die Programmierung, Steuerung und Messung an der Basisplatine vornehmen.

Der Gesamtschaltplan der Basisplatine ist der im Quellenverzeichnis aufgeführten Datei „Basisplatine Kaffeemaschine v02“ zu entnehmen.

3.2 Basisplatine – Konfiguration Getriebepumpe

Schrittweise wurden die verschiedenen elektronischen Komponenten von der Raspberry-PICO Schaltung abgesteckt und an die Basisplatine angeschlossen. Zuerst wurde die Schaltung so aufgebaut, dass die Kennlinie der Getriebepumpe ausgelesen werden konnte. Später wurde der Aufbau für die Verwendung des DC-Antriebs angepasst. Dabei wurde wie folgt vorgegangen:

An der 24V Anschlussleiste wurde zuerst die 24V Versorgungsspannung angeschlossen. Diese Spannung wird aus dem 24 V Verteiler entnommen. Für die Spannungsversorgung der Getriebepumpe wurde diese mit der 24V Anschlussleiste der Basisplatine verbunden.

Zur Ansteuerung der Magnetventile Y01 bis Y04 dient die SSR1-Leiste. Hierbei wird das Ansteuersignal über die Kanäle D03 bis D06 ausgegeben (Belegung entsprechend Tab. 1)

Ventil	**Ansteuerungssignal**
Y01	D03
Y02	D04
Y03	D05
Y04	D06

Tabelle 1 Ventilansteuerung

Das Auslesen des Füllstands im Boiler geschieht durch den Füllstandssensor. Dieser Sensor ist über einen vierpoligen Stecker mit der Basisplatine verbunden. Belegt sind hierbei die Anschlüsse von FUELL1.

Am 10V Anschluss wurde die Ansteuerung für das Dosierventil und die Ansteuerung der Pumpe über das APPOLDT Signal angeschlossen. Die Stromversorgung und Ansteuerung des Boilers ist in dieser Konfiguration abgeklemmt. Die Funktionen PWM TASSENWÄRMER, DC-MOTOR, DISPLAY, FUELL2 und FLOWMETER werden hierbei ebenso nicht benutzt.

3.3 Basisplatine – Konfiguration DC-Antrieb und Boiler

Zur Inbetriebnahme der DC-Antriebe in Verbindung mit dem Boiler muss die Verkabelung an der Basisplatine angepasst werden, der grundsätzliche Aufbau kann jedoch bestehen bleiben.

Die Heizleistung des Boilers wird über das APPOLDT Signal gesteuert (steuert PWM). Hierzu muss die Ansteuerung der Getriebepumpe abgeklemmt werden. Die DC-Antriebe werden über ein externes DC-Labornetzgerät angesteuert und versorgt.

Beide Konfigurationen werden in dem Schaltplan in Abb.5 dargestellt.

3.4 NI-Messkarte – Einleitung und Funktion

Neben den Messplatinen an der Basisplatine ist die NI-Messkarte für die Aufnahme der Messsignale zuständig. Ausgelesen werden hierbei Kontrolldruck und Kontrolltemperatur sowie Referenzdruck und Referenztemperatur als auch das Flowmeter (Durchflusssensor). Für die Untersuchung der DC-Antriebe gibt es einen Drehzahl- und einen Drehmomentsensor, deren Signale ebenfalls über die NI-Messkarte aufgenommen werden.

Die Eingangssignale liegen hierbei auf den Kanälen 1 bis 7 und die zugehörigen Masseanschlüsse auf den Kanälen 19 bis 25. Die genaue Belegung ist dem Schaltplan zur NI-Messkarte in Abb. 6 zu entnehmen. Der Gesamtschaltplan der NI-Messkarte inkl. angeschlossener Komponenten ist im Anhang A1 hinterlegt.

Die Auswertung der Messsignale erfolgt über MATLAB (siehe Kap. Software).

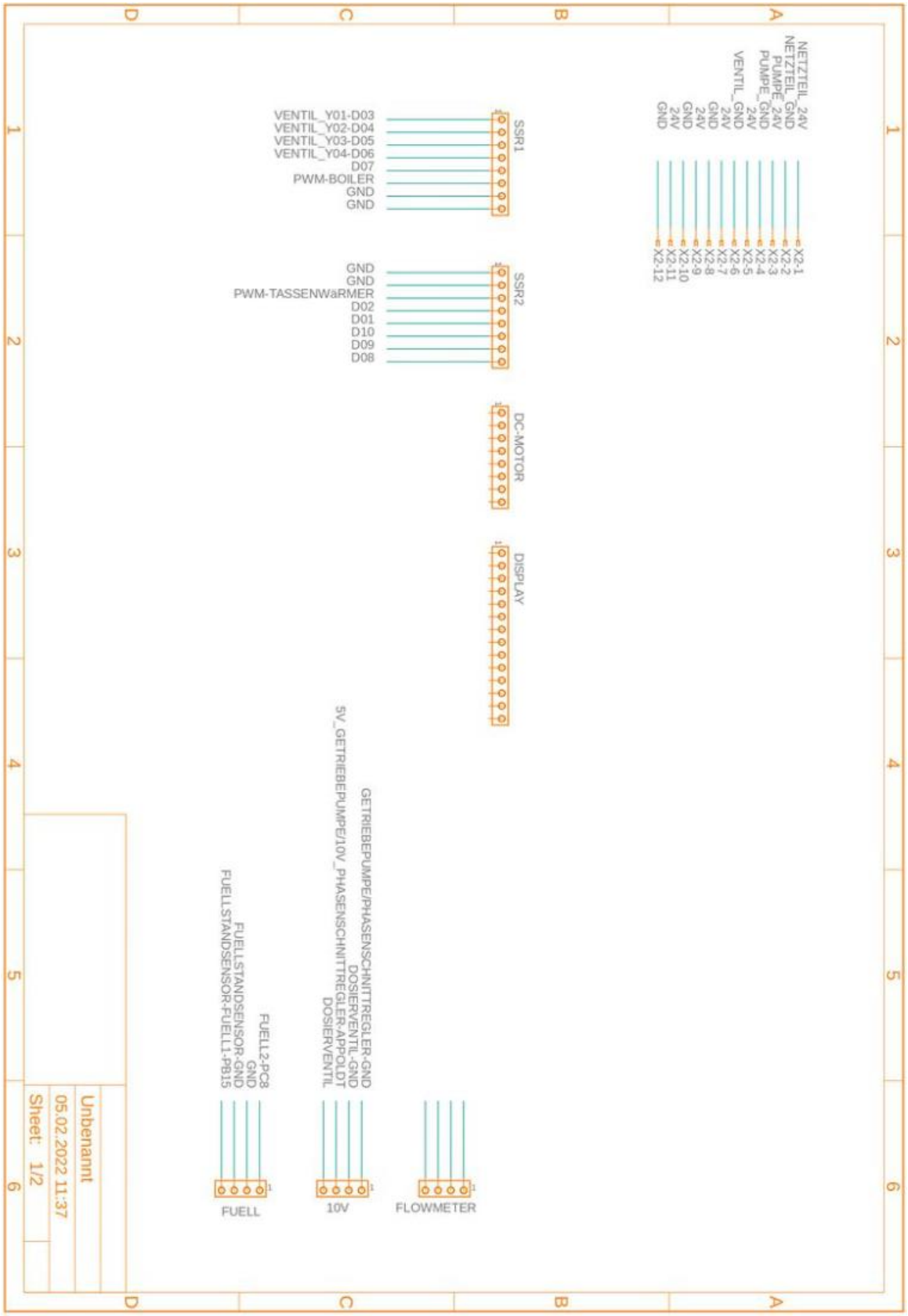


Abbildung 5 Schaltplan Anschlüsse an die Basisplatine

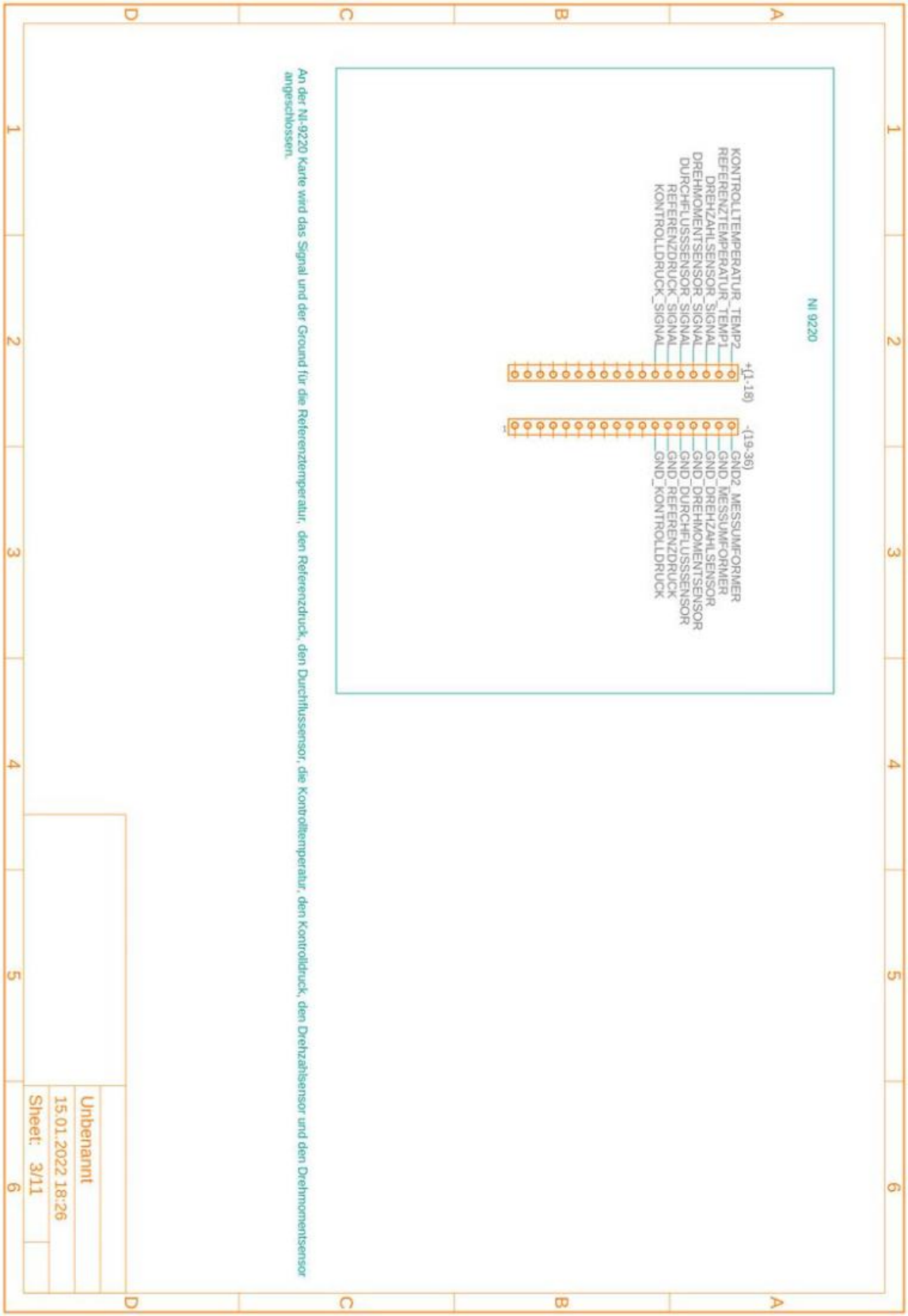


Abbildung 6 Schaltplan Anschlüsse der NI-Messkarte

3.5 Basisplatine - Messplatinen

Oberhalb der Basisplatine befinden sich 3 Ansteckplätze für Messplatinen. Hier können Platinen für zukünftige Sensor Kalibrierungen angeschlossen werden, sobald die Software hierfür ausgereift ist. Aktuell sind eine NTC-Messplatine und eine Spannungsteiler Platine angeschlossen. Die Funktion der NTC-Messplatine konnte durch einen Versuch mit einem NTC-Sensor bereits bestätigt werden.

Nach der Inbetriebnahme des Prüfstands mit angeschlossener Brückenmessschaltungsplatine gab es auf der Platine einen Kurzschluss, der durch einen Konstruktionsfehler an der Platine hervorgerufen wurde. ~~Der Entwickler der Platine, Marius Ghica, arbeitet hierzu an einer Lösung. Herr Rohnen hat eine Ersatzlösung angekündigt, die bis dahin – soweit nötig - eingesetzt werden könnte. Der Schaltplan zu den Messplatinen ist im Anhang A2 hinterlegt.~~

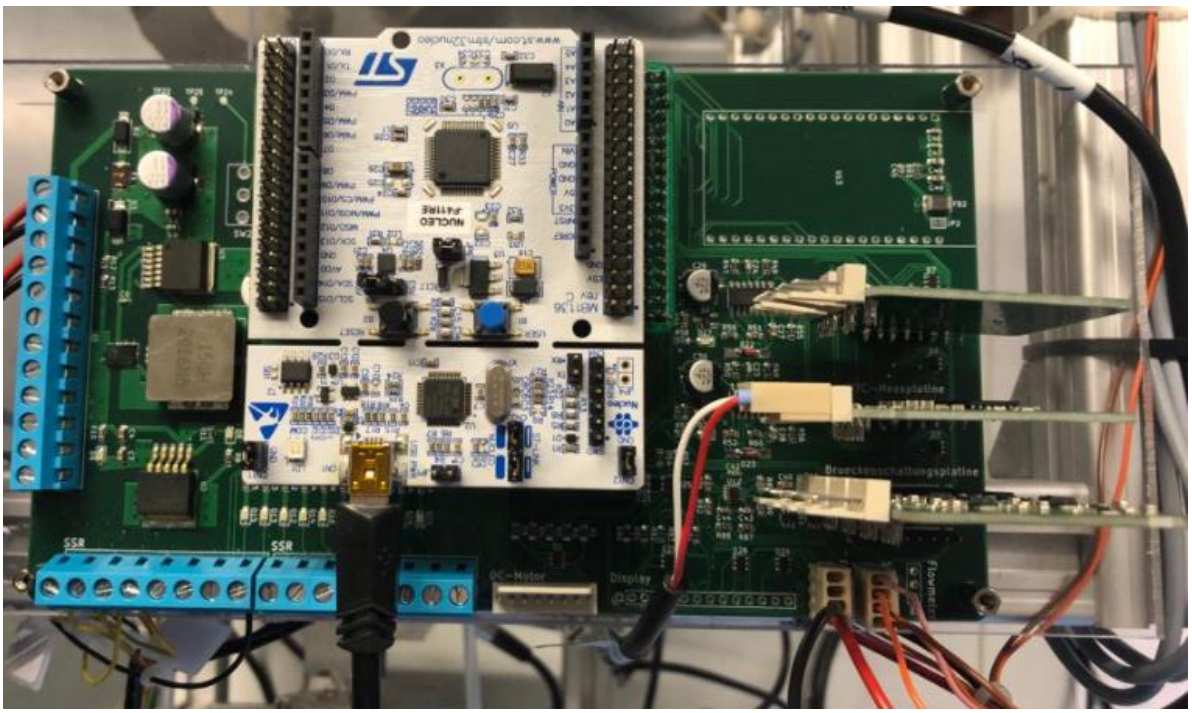


Abbildung 7 Basisplatine mit 3 angeschlossenen Messplatinen (rechts im Bild)

3.6 Sensoren

Der Pumpenprüfstand verfügt über eine Vielzahl von Sensoren, deren Signalaufbereitung in MATLAB umgesetzt wird. Verwendet werden derzeit Druck-, Temperatur-, Drehzahl-, Drehmoment-, Durchfluss-, Füllstandsensoren.

Am Prüfstand sind B+B Drucksensoren mit einem Messbereich von 0 bis 10 bar verbaut. Im Rahmen der Inbetriebnahme Getriebepumpe konnte festgestellt werden, dass ein Messbereich von 0 bis 10 bar für den Prüfstand nicht ausreichend ist (siehe Kap. 5.3 Funktionsprüfung Getriebepumpe). Daher wurde zum Auslesen der Getriebepumpenkennlinie Druck über Volumenstrom provisorisch ein Drucksensor mit einem 0 bis 40 bar Messbereich verbaut (siehe Abb.8). Das Messsignal wurde hierbei über Kanal 8 der NI-Messkarte ausgelesen. ~~B+B Drucksensoren mit einem größeren Messbereich sind bereits bestellt. Sobald diese eintreffen, sind die 10 bar Sensoren auszutauschen und die Software ggf. an den neuen Messbereich anzupassen.~~

Abbildung 8 Provisorischer Messaufbau mit 40 bar Drucksensor

4. Software

4.1 Grundfunktion der MCU

Der Teilbereich Grundfunktion der MCU befasst sich mit der Programmierung und somit der Erstellung der Python-Skripte auf der Basisplatine. Da die Bedeutung und Funktion der verwendeten Codes bereits in der Dokumentation [41] von LbA Rohnen genauestens erklärt werden, wird in diesem Bericht nicht näher auf die Details eingegangen.

Trotz allem muss unter diesem Punkt erwähnt werden, dass bei Initialisierung des Microcontrollers (initboard.py) d.h. beim Einschalten des Stromes, die Getriebepumpe sofort angelaufen ist, was unter normalen Umständen nicht der Fall sein sollte. Die Vermutung liegt hierbei, dass die anliegende Spannung am appoldt-PIN in einer bestimmten Weise gespeichert und im System hinterlegt wird. Um dieses Problem zu überbrücken, wurde deshalb ein Code auf dem Python-Skript main.py hinterlegt, der nach dem Anschluss des USB-Steckers die anliegende Spannung auf 0 setzen und somit die Getriebepumpe zum Stillstand bringen soll. In einem nachfolgenden Test wurde die Funktion dieser Notlösung bestätigt.

4.2 Programmierung der Schnittstelle zwischen MCU und MATLAB

Im aktuellen Stand ist eine Bedienoberfläche [25] vorhanden, die zwar eine Struktur hat, jedoch noch keine Funktionalität besitzt. Deshalb dient dieser Teilabschnitt der Softwareprogrammierung zur Erarbeitung der benötigten Funktionen und somit zur Vorbereitung der Codes für die Implementierung in die bereits vorhandene Benutzeroberfläche. Hierfür wurde folgende Vorgehensweise festgelegt:

1) Verbindung des MCU mit MATLAB
2) Definition und Ansteuerung der Magnetventile (Y01, Y02, Y03, Y04)
3) Auslesen der Spannungswerte an *appoldt* und *dosierventil*
4) Auslesen der Messgrößen auf der NI-Messkarte
5) Auslesen der Sensoren auf der Basisplatine

4.2.1 Verbindung des MCU mit MATLAB

Beim Anschluss der Basisplatine an den Rechner wird diesem im System automatisch eine COM-Portnummer zugeordnet, die manuell in den Einstellungen je nach Belieben verändert werden kann. Zur Vereinfachung ist der COM6 für die Schnittstelle zwischen dem Microcontroller und MATLAB ausgewählt worden. Um jetzt eine Verbindung mit genau diesem COM-Port aufzubauen, wurde folgender Ansteuerungscode verwendet:

```
stm32 = serialport("COM6", 115200);
configureTerminator(stm32, "CR/LF");
configureCallback(stm32, "terminator", @STM32SerialRead);
      .
      .
Python-Codes mit dem Befehl writeline über MATLAB an MCU übergeben.
Dabei müssen diese verzögert übergeben werden (siehe Anhang A3).
```

Für eine genaue Erklärung der Schnittstellenverbindung siehe Dokumentation [40] „MATLAB® meets MicroPython“.

4.2.2 Definition und Ansteuerung der Magnetventile (Y01, Y02, Y03, Y04)

Um die korrekte Ansteuerung der Magnetventile gewährleisten zu können, war es zuallererst nötig, die richtige PIN-Zuordnung auf den SSR-Inseln herauszufinden. Hierfür mussten alle Ventile nach der Reihe durchgeschaltet werden. Dies wurde mit folgenden Codes realisiert:

```
D01: writeline(stm32, „gpio_exp.pin(1, value = 1)“)
D02: writeline(stm32, „gpio_exp.pin(0, value = 1)“)
D03: writeline(stm32, „gpio_exp.pin(8, value = 1)“)
D04: writeline(stm32, „gpio_exp.pin(9, value = 1)“)
D05: writeline(stm32, „gpio_exp.pin(10, value = 1)“)
D06: writeline(stm32, „gpio_exp.pin(11, value = 1)“)
D07: writeline(stm32, „gpio_exp.pin(12, value = 1)“)
D08: writeline(stm32, „gpio_exp.pin(13, value = 1)“)
D09: writeline(stm32, „gpio_exp.pin(14, value = 1)“)
D10: writeline(stm32, „gpio_exp.pin(15, value = 1)“)
```

Nach Betätigung der einzelnen Ventile, hat sich herausgestellt, dass die ursprüngliche PIN-Belegung in der Dokumentation [35] Fehler enthielt. Deshalb wurde der Elektronikschaltplan überarbeitet und die neue Anschlussbelegung dokumentiert (siehe Abb. 5).

4.2.3 Auslesen der Spannungswerte an *appoldt* und *dosierventil*

Zur Überprüfung der korrekten Spannungsübertragung an die PINs *appoldt* und *dosierventil*, wurden anhand unten gezeigter Beispiele Werte übergeben und im Anschluss mit einem Multimeter die ankommende Spannung gemessen.

Formel für die Übergabe des Spannungswertes: $Wert = \frac{4095}{10000} * U \quad \rightarrow U\ in\ mV$

Programmbefehle für die Übergabe:

```
writeline(stm32, „appoldt.write(2047)“)          ☾ 5V-Signal
writeline(stm32, „dosierventil.write(4095)“)     ☾ 10V-Signal
```

Nach eingehender Auswertung der Messungen wurde festgestellt, dass die eingegebenen Größen zwar an den PINs ankommen, jedoch der Spannungswert vom *appoldt* am *dosierventil* und andersherum anliegt. Daher wurde die Verkabelung getauscht und diese richtig im Elektronikschaltplan (siehe Abb. 5) hinterlegt.

4.2.4 Auslesen der Messgrößen auf der NI-Messkarte

Die NI-Messkarte spielt in dem Prüfstandsaufbau eine sehr große Rolle, da an dieser nicht nur die Referenzsensoren für die Kalibrierung, sondern auch wichtige Sensoren wie Durchfluss-, Drehzahl-, Drehmoment-, Kontrolldruck-, Kontrolltemperatursensoren angeschlossen werden. Zum Auslesen dieser Messgrößen wurde ein einfaches MATLAB-Skript „NI-Messkarte.m“ geschrieben, das später in die Bedienoberfläche integriert werden und in Verbindung mit einem definierten Puffer konstant die Daten im Hintergrund verarbeiten soll:

```
messgeraet = daq("ni");
addinput(messgeraet, "cDAQ4Mod1", "ai0", "Voltage");
addinput(messgeraet, "cDAQ4Mod1", "ai1", "Voltage");
addinput(messgeraet, "cDAQ4Mod1", "ai2", "Voltage");
addinput(messgeraet, "cDAQ4Mod1", "ai3", "Voltage");
addinput(messgeraet, "cDAQ4Mod1", "ai4", "Voltage");
addinput(messgeraet, "cDAQ4Mod1", "ai5", "Voltage");
addinput(messgeraet, "cDAQ4Mod1", "ai6", "Voltage");
messgeraet.Rate = 60000;
fs = messgeraet.Rate;
```

```
messgeraet.ScansAvailableFcnCount = ceil(fs/refreshRate);
messgeraet.ScansAvailableFcn = @(src,evt) datenverarbeitung(app, event);
start(messgeraet, 'continuous');

function datenverarbeitung(app, ~, ~)
    [data, time, ~] = read(messgeraet,
messgeraet.ScansAvailableFcnCount,'OutputFormat','Matrix');

end

stop(messgeraet);
```

Anhand der Messergebnisse beim Testen des MATLAB-Skripts ist allerdings aufgefallen, dass die Belegung der NI-Messkarte in der Dokumentation von Stephan Hase [35] falsch hinterlegt war. Deshalb wurden anhand der Verkabelung die richtigen Anschlüsse definiert und im NI-Schaltplan (siehe Abb. 6) protokolliert.

4.2.5 Auslesen der Sensoren auf der Basisplatine

Zum Auslesen der verschiedenen Sensoren, die an die Basisplatine angeschlossen werden, sind folgende Befehle nötig:

```
writeline(app.stm32, 'T_Boiler.read()*3300/4095');
writeline(app.stm32, 'T_Befuellung.read()*3300/4095');
writeline(app.stm32, 'T_Eingang.read()*3300/4095');
writeline(app.stm32, 'P_Gruppe.read()*3300/4095');
writeline(app.stm32, 'P_Boiler.read()*3300/4095');
writeline(app.stm32, 'T_Zwischenraum.read()*3300/4095');
writeline(app.stm32, 'T_Mischer.read()*3300/4095');
writeline(app.stm32, 'P_Boiler_Alt.read()*3300/4095');
writeline(app.stm32, 'Leitwert.read()*3300/4095');
writeline(app.stm32, 'Taste.read()*3300/4095');
writeline(app.stm32, 'Gewicht1.read()*3300/4095');
writeline(app.stm32, 'Gewicht2.read()*3300/4095');
writeline(app.stm32, 'Fuell_1.value()');
writeline(app.stm32, 'Fuell_2.value()');
```

Bei den Sensoren, die sowohl an der NI-Messkarte, als auch an der Basisplatine angeschlossen werden, muss beachtet werden, dass die Ausgabe immer in mV erfolgt. Für eine korrekte Anzeige in den richtigen Einheiten sind für die Umrechnung noch aus den Datenblättern oder durch eine Kalibrierung über die Bedienoberfläche Formeln herauszuarbeiten.

4.3 Weiterentwicklung der Bedienoberfläche / MATLAB GUI

4.3.1 Verbindungs- und Entkopplungsfunktionen

Zum Aufbau der Verbindung mit der Basisplatine wurde die Funktion *stm32_connect* definiert, die anhand der Schnittstellencodes (siehe Anhang A3) über die Betätigung des Buttons „Mit MCU verbinden“ im jeweiligen Panel ausgeführt werden soll.
Parallel dazu wurde zur Trennung der Verbindung die Funktion *stm32_disconnect* erstellt, die beim Wechsel zur Startseite, bei Betätigung des Button „MCU ausschalten“ oder durch Beendigung des Programms durchlaufen werden soll. In einem Verbindungstest anhand der Bedienoberfläche wurde die Funktion dieser Befehle bestätigt.

4.3.2 Wasserwechsel

Im Wasserwechsel-Modus soll die Möglichkeit bestehen, den Boiler mit Wasser zu füllen, ihn zu leeren oder eine Zirkulation des Wassers durch eine Systemspülung zu ermöglichen. Zum Schalten der vollen Pumpenleistung und der Ventile wurden die jeweiligen Buttons mit den Funktionscodes belegt.

Außerdem wurde zur Unterbrechung dieser Vorgänge, der STOP-Button mit einem Setup verknüpft, um den Ursprungszustand wieder herzustellen. Bei der Inbetriebnahme des Wasserwechsel-Modus konnten diese Vorgänge reibungslos durchgeführt werden.

Eine weitere Eigenschaft des Wasserwechsel-Modus besteht darin, die aktuelle Temperatur, den Druck und den Füllstand anzuzeigen. Der gewählte Ansatz war über einen Scheduler alle 0,5 s die Daten aus der Basisplatine auszulesen und die dazugehörigen Labels zu aktualisieren. Allerdings wurden dabei Timer-Fehler ausgegeben, wodurch dieses Konzept nicht funktionsfähig war. Deshalb ist hierzu ein neuer Lösungsansatz auszuarbeiten (siehe Kapitel 4.3.4)

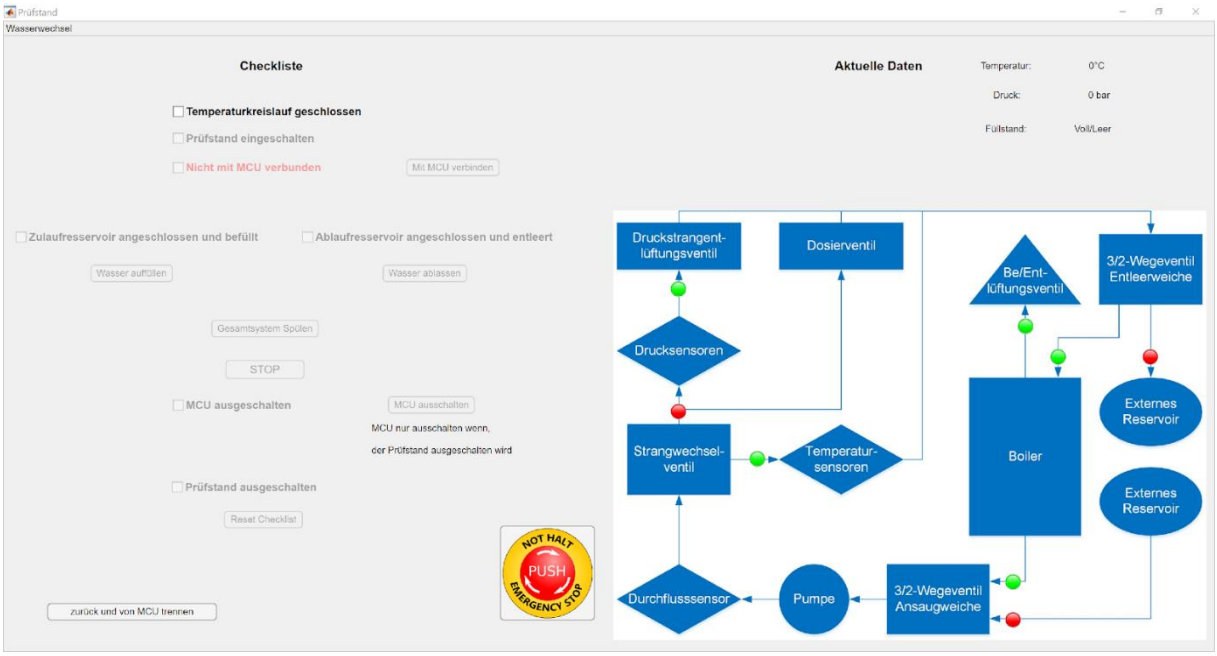


Abbildung 9 MATLAB GUI – Wasserwechsel

4.3.3 Manueller Modus

Im manuellen Modus wurde aufbauend auf den ursprünglichen Zustand von Korbinian Ass [25] die Ventilsteuerung neu definiert und in den jeweiligen Callbacks die richtigen Codes für die Umstellung der Ventile hinterlegt. Beim Durchschalten dieser ist sofort aufgefallen, dass die Funktion zwar gegeben war, aber die Lampen, die den Verlauf des Wasserstroms aufzeigen sollen, Fehler enthielten. Zur Lösung dieses Problems wurde die Lampenfunktion *ventil_lamps_check* überarbeitet und in einem erneuten Versuch bestätigt.

Im Rahmen dieses Panels sollten neben der Ventilsteuerung auch die Einstellung der Heizleistung (bzw. Ansteuerung Getriebepumpe), Motorleistung und des Dosierventils ermöglicht werden. Zusätzlich wird die Anzeige verschiedenster Messgrößen in Real-time umgesetzt. Dies wurde über einen Scheduler, der bereits in die Bedienoberfläche integriert war, versucht zu realisieren. Jedoch kam es hier ebenfalls zu Verzögerungen bei der Aktualisierung und zum Aufhängen des Programms. Die Fehlfunktion lässt sich durch die immer wiederkehrenden Timer-Fehler erklären. Analog zum Wasserwechsel ist hier auch ein neues Konzept zu finden (siehe Kapitel 4.3.4).

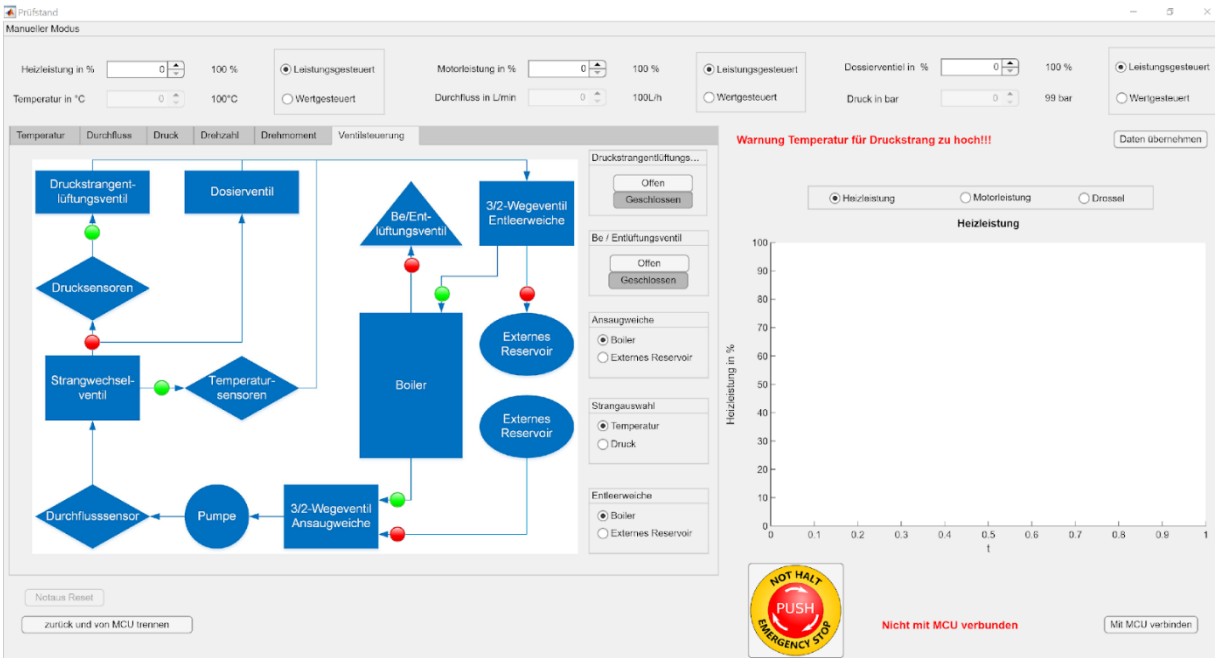


Abbildung 10 MATLAB GUI - Manueller Modus

4.3.4 Timerfehler – Neuer Lösungsansatz

Zum Ende des Projekts wurde festgestellt, dass sowohl die NI-Messkarte, als auch die Basisplatine, nicht mit den definierten Scheduler-Funktionen kompatibel sind. Begründet werden kann dies durch die unterschiedliche Systemstruktur von Raspberry PICO und der NI-Messkarte bzw. Basisplatine.

Deshalb wurden zur Übersichtlichkeit alle Funktionen, die einen Scheduler verwenden, aus der Bedienoberfläche entfernt.

Für die zukünftige Weiterentwicklung muss zur konstanten Verarbeitung der Daten in beiden Fällen ein Puffer angelegt werden, der die Informationen zwischenspeichert und von dort aus zur Verwendung abrufbar macht. Dabei muss auch beachtet werden, dass in diesem Zyklus Sicherheitsmechanismen nötig sind. Bei kritischen Betriebsbedingungen, wie z.B. Überschreiten der Betriebstemperatur oder bei Trockenlauf, soll sich das System automatisch abschalten. Zuerst ist hierfür ein Funktionsplan zu erstellen.

5. Funktion

Das Kapitel Funktion umfasst die Dichtheits- und Funktionsprüfung der Magnetventile und des Regelungsstrangs, aber auch die Überprüfung der Kommunikation zwischen MATLAB und der Basisplatine. Eine hohe Priorität hat hier auch der Betriebstest der Getriebepumpe, die die Eignung dieses Bauteils für den weiteren Verlauf feststellen soll. Im Angesicht der Aufgabenstellung sollen ebenso zwei vorliegende DC-Motoren in Betrieb genommen werden.

5.1 Funktionsprüfung - Aufbau, Ansteuerung und Dichtheitsprüfung

Im Rahmen der ersten Inbetriebnahme der Getriebepumpe wurde sowohl eine Funktionsprüfung des Aufbaus als auch ein Verbindungstest mit MATLAB durchgeführt. Die erfolgreiche Ansteuerung der Getriebepumpe und Steuerung der Ventile über MATLAB konnte bestätigt werden.

Die Anlage wurde bei laufender Pumpe auf ihre Dichtigkeit überprüft. Es konnten in beiden Strängen keine undichten Stellen gefunden werden.

5.2 Funktionsprüfung - Getriebepumpe

5.2.1 Belastungstest

Schon bei der ersten Inbetriebnahme ist aufgefallen, dass sich die Getriebepumpe bei bestimmten Bedingungen stärker erwärmt und abschaltet. Um dieses Verhalten nachzustellen, wurden 3 Tests unter verschiedenen Rahmenbedingungen durchgeführt und nebenbei die Temperatur außen an der Pumpe mit einem NTC-Sensor gemessen.

Hierbei ist zu erwähnen, dass die Temperaturmessung nicht dafür galt einen spezifischen Wert auszumessen, sondern nur um das Vorhandensein eines Grenzwertes zu bestätigen.

<u>Test 1) Getriebepumpe bei Volllast im Temperaturstrang</u>

Nach 5 Minuten wurde der Test abgebrochen, da die Getriebepumpe nicht abgeschaltet hat und keine wesentliche Erwärmung festgestellt wurde. Dieses Ergebnis ist erwartet und plausibel, da im Temperaturstrang kein Gegendruck aufgebaut wird und somit keine wirkliche Belastung entsteht.

Test 2) Getriebepumpe bei Volllast im Druckstrang mit Dosierventil auf 50%

Nachdem die Getriebepumpe wieder auf den Anfangswert abgekühlt wurde, wurde der Test erneut gestartet. Auffällig war sofort, dass der Temperaturanstieg im Gegensatz zu vorher größer war. Dies ist damit zu erklären, dass durch die Begrenzung des Durchflusses mit dem Dosierventil ein Gegendruck entsteht, der von der Pumpe bewältigt werden muss. Die Abschaltdauer lag bei 5 min.

Test 3) Getriebepumpe bei Volllast im Druckstrang mit Dosierventil auf 5%

Nach erneuter Herstellung der Anfangsbedingungen wurde die Getriebepumpe wieder gestartet. Logischerweise war der Anstieg der Temperatur in diesem Fall größer, sodass die Getriebepumpe bereits bei 4min. 27s abgeschaltet hat.

Fazit:

Beim Vergleich von Test 2 und 3 ist aufgefallen, dass die Pumpe ungefähr immer beim selben Wert abschaltet, was das Vorhandensein eines Sicherheitsmechanismus vermuten lässt.

Außerdem kann resultierend gesagt werden, dass die Abschaltdauer in Bezug auf die Betriebszeit der Getriebepumpe in einer Kaffeemaschine (ca. 30-50 s) kein Problem darstellt.

5.2.2 Betriebstest

Zur Beurteilung der Getriebepumpe hinsichtlich der Eignung für eine Espressomaschine, wurde im Rahmen des Betriebstests eine Kennlinie Druck über Volumenstrom ausgelesen. Die Getriebepumpe und das Dosierventil wurden dabei mit dem MATLAB Skript „mcu_connection_test.m“ angesteuert. Die Messung der Kennwerte erfolgte dann über ein separates Skript (siehe Kapitel 4.2.4).

Testparameter:

- Pumpenleistung: Volllast
- Strangauswahl: Druckstrang
- Dosierventilstellung: Zuerst komplett geschlossen 0 mV (0%), danach von 0-1000 mV (0-10%) in 100mV-Schritten und ab 1000-10000 mV (10-100%) in 1000mV-Schritten geöffnet
- Messdauer für jede Dosierventilstellung: 30 s

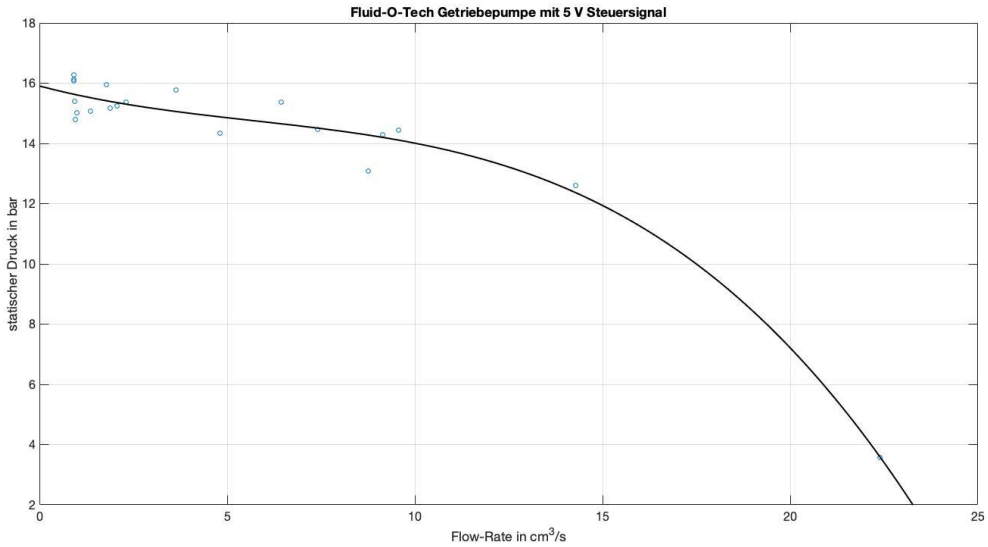


Abbildung 11 Volllastkennline der Getriebepumpe

Die oben dargestellte Auswertung zeigt, dass die Getriebepumpe für die Nutzung in einer Espressomaschine ausreichend dimensioniert ist. Ein Indikator für die Eignung ist der hohe Druck von ca. 15 bar, der sich bereits bei einer Flow-Rate von 2-3 cm³/s einstellt. Dieser dürfte ausreichen, um Druckverluste entlang der Verrohrung innerhalb der Maschine zu kompensieren. Der ersten Einschätzung nach würde im Betrieb nur sehr selten die Volllast benötigt werden.

5.4 Funktionsprüfung - DC-Antriebe

Nach Auswertung der Getriebepumpe wurden zwei DC-Antriebe der Firma Groschopp mit 66 W und 100 W Nennleistung in Betrieb genommen. Die Vorgehensweise war hierbei analog zur Auswertung der Getriebepumpe (gleiche Prüfparameter). Es wurden die Daten für eine Kennlinie Druck über Volumenstrom ausgelesen.

Die Messungen wurden bei Nennleistung und nicht bei Maximalleistung durchgeführt, da es zu starker Aufheizung an den Motoren kam und sich die Spannungen und Ströme über das Labornetzteil nur bedingt einstellen ließen.

Die Messdaten der Funktionsprüfung wurden auf dem Laborlaptop hinterlegt und müssen noch ausgewertet und dargestellt werden.

Eine erste Abschätzung zeigt, dass die DC-Antriebe einen Betriebsdruck von nur etwa 4 bis 5 bar erzeugen.

5.5 Funktionsprüfung - Fazit Getriebepumpe und DC-Antriebe

Die folgende Tabelle dient der Vergleichbarkeit der Getriebepumpe und den DC-Antrieben in verschiedenen Kriterien.

Kriterium	**DC-Antriebe**	**Getriebepumpe**
Größe	sehr groß	klein
Verbaubarkeit	Nur in besonderen Lagen verbaubar	beliebig verbaubar
Geräusch	Relativ laut, viel Vibration und schlagen, ungleichmäßiges Klangbild	Mittelmäßig laut, etwas Vibration, kein schlagen, gleichmäßiges Klangbild (siehe Anhang A3)
Leistung	Vergleichbar sobald beide Auswertungen vorliegen	
Preis	135 € / Stück	170,70€ / Stück mit Verbindungskomponenten, ohne Microcontroller
Wärmeentwicklung	Starke Wärmeentwicklung, keine Abschaltfunktion, langsames Abkühlen	Mittlere Wärmeentwicklung, mit Abschaltfunktion, mittelschnelles Abkühlen

Tabelle 2 Vergleich DC-Antriebe und Getriebepumpe

Es lässt sich erkennen, dass die Getriebepumpe in allen Vergleichskriterien besser dasteht als beide DC-Antriebe, daher ist die Getriebepumpe für die zukünftige Verwendung in der Espressomaschine zu empfehlen.

6. Übergabeschnittstellen

~~Zum Abschluss des Semesters wurden zwischen den Studenten und Hr. Rohnen Schnittstellen für das Projekt definiert.~~ Diese Schnittstellen sollen den aktuellen Stand des Projekts verdeutlichen und eine nahtlose Übergabe zur weiteren Bearbeitung ermöglichen. Hierbei wurde sich auf die folgenden Punkte geeinigt:

- Inbetriebnahme der DC-Antriebe wird durchgeführt inkl. Erfassung Druck über Volumenstrom, Daten werden übergeben
- Vorbereiten der Messplatinen und Anschluss der Sensoren an die Basisplatine
- In der Bedienoberfläche wird die Funktion Wasserwechsel und manueller Modus betriebsbereit fertiggestellt, Programmierung der Kalibrierungsfunktionen in der Bedienoberfläche wird nicht übernommen
- Heizelement wird nach Abbau der Getriebepumpe an den Phasenschnittregler angeschlossen und der Phasenschnittregler an den zugehörigen 0 bis 10 V Sollwert an der Basisplatine

Kurz vor Projektübergabe wurde vereinbart, dass zum Übergabezeitpunkt die Getriebepumpe an den Prüfstand angeschlossen bleibt, da so die Benutzeroberfläche erprobt werden kann. Ebenso bleiben die Messplatinen wie in Kap. 3.5 beschrieben angeschlossen, sodass die nächste Gruppe weitere Modifikationen im Rahmen der Kalibrierung vornehmen kann.

7. Ausblick

Zum aktuellen Zeitpunkt lassen sich am Prüfstand bereits beliebige Pumpen einbauen und einige Messgrößen für Kennlinien auslesen. Zukünftig muss aufgrund der Belegung des APPOLDT-Signals durch den Phasenschnittregler ein Konzept erarbeitet werden, bei dem die Ansteuerung der Pumpen durch die Basisplatine erfolgen kann.

Für die Bedienoberfläche ist geplant, eine Software zu entwickeln, die Daten aus den verbundenen Sensoren automatisch auswertet und ausgibt. Die Programmierung muss so ausgelegt werden, dass kritische Betriebszustände automatisch verhindert werden.

Eine weitere große Teilfunktion des Prüfstands ist die Kalibrierung von Sensoren. Geplant ist verschiedene Sensoren an den Prüfstand anzuschließen und diese mittels der zugehörigen Messplatinen zu kalibrieren. Softwareseitig ist es hierzu nötig, ein Programm für die Druck- und Temperaturregelung zu entwickeln und diese in den Kalibriermodus der Bedienoberfläche zu integrieren. Seitens der Hardware ist aufgrund der fehlenden Brückenmessplatine ein Workaround nötig. Im Rahmen dieses Workarounds soll zwischen Sensor und MCU eine Platine verbaut werden, die das Signal aufbereitet.

Ferner ist es denkbar, dass am Prüfstand ~~Schallmessungen und~~ Schwingungsanalysen durchgeführt werden. Das Konzept hierzu müsste noch ausgearbeitet werden.

8. Verzeichnis der verwendeten Abbildungen und Tabellen

9. Literatur und Quellen

Folgende Quellen sind dem Quellenverzeichnis im Turnus-Wiki entnommen:

[25] Korbinian Ass, Valentin Sachmann, Simon Schmetz, Entwicklung eines Kalibriersystems für Druck- und Temperatursensoren, Projektarbeit 2021

[35] Stephan Hase, Inbetriebnahme des Pumpenprüfstands, Praxissemester Sommersemester 2021

[40] Armin Rohnen, MATLAB® meets MicroPython, Vorabversion, 2021

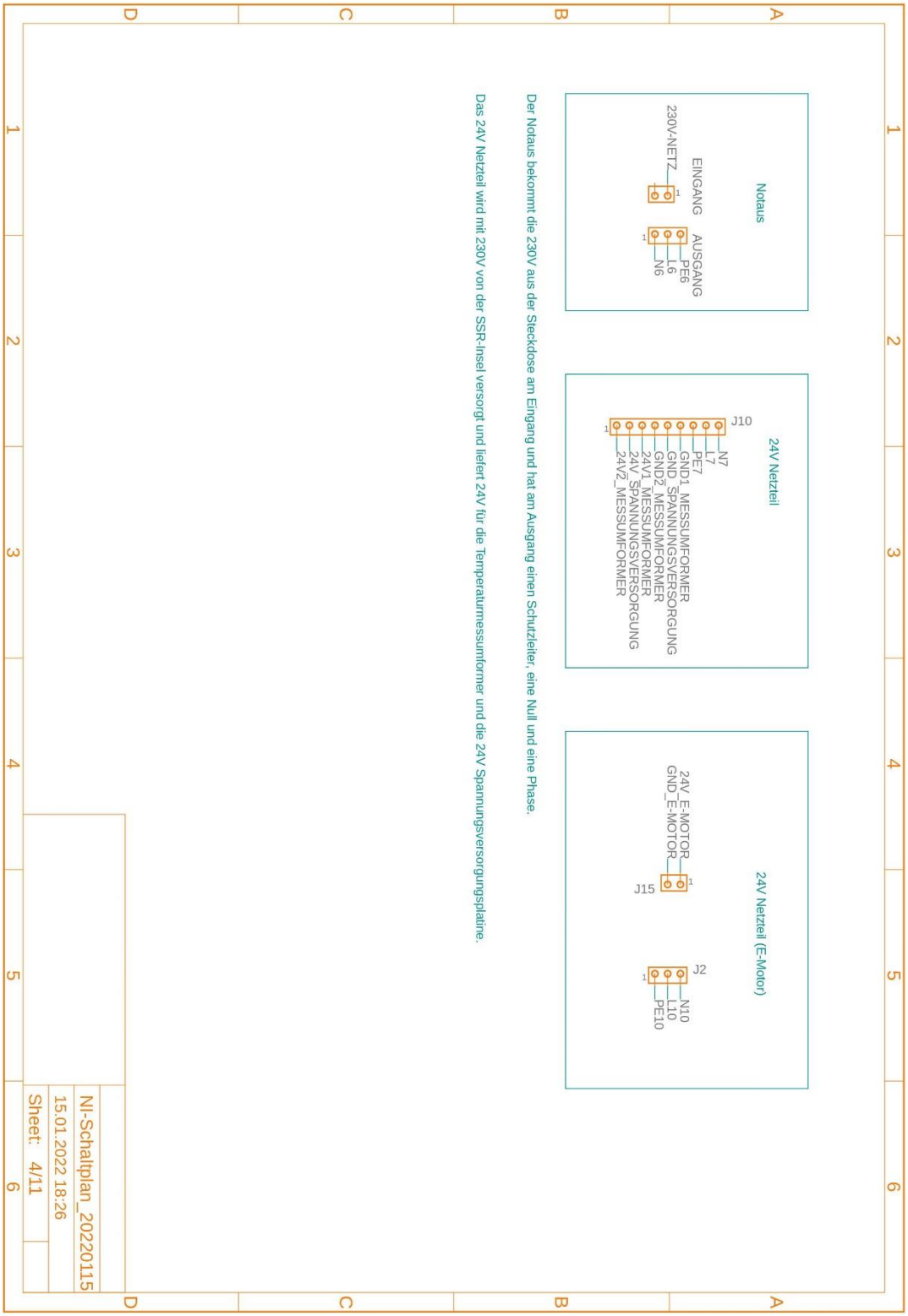
[41] Armin Rohnen, STM32F411 nucleo - MATLAB® Schnittstelle, Stand November 2021

Weitere Quellen:

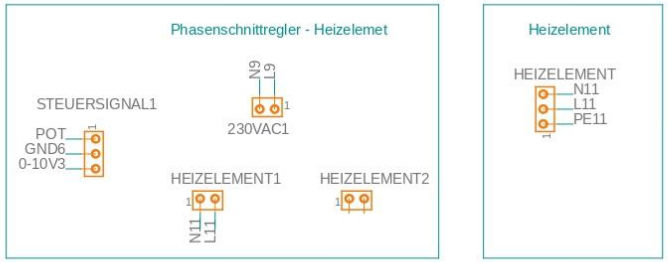
Marius Ghica, Armin Rohnen, Basisplatine Kaffeemaschine v02, November 2021 https://wiki.turnus-espresso.de/Datei:20210511_Basisplatine_Kaffeemaschine_v02.pdf

10. Anhang

Anhang A1 – Schaltplan Basisplatine



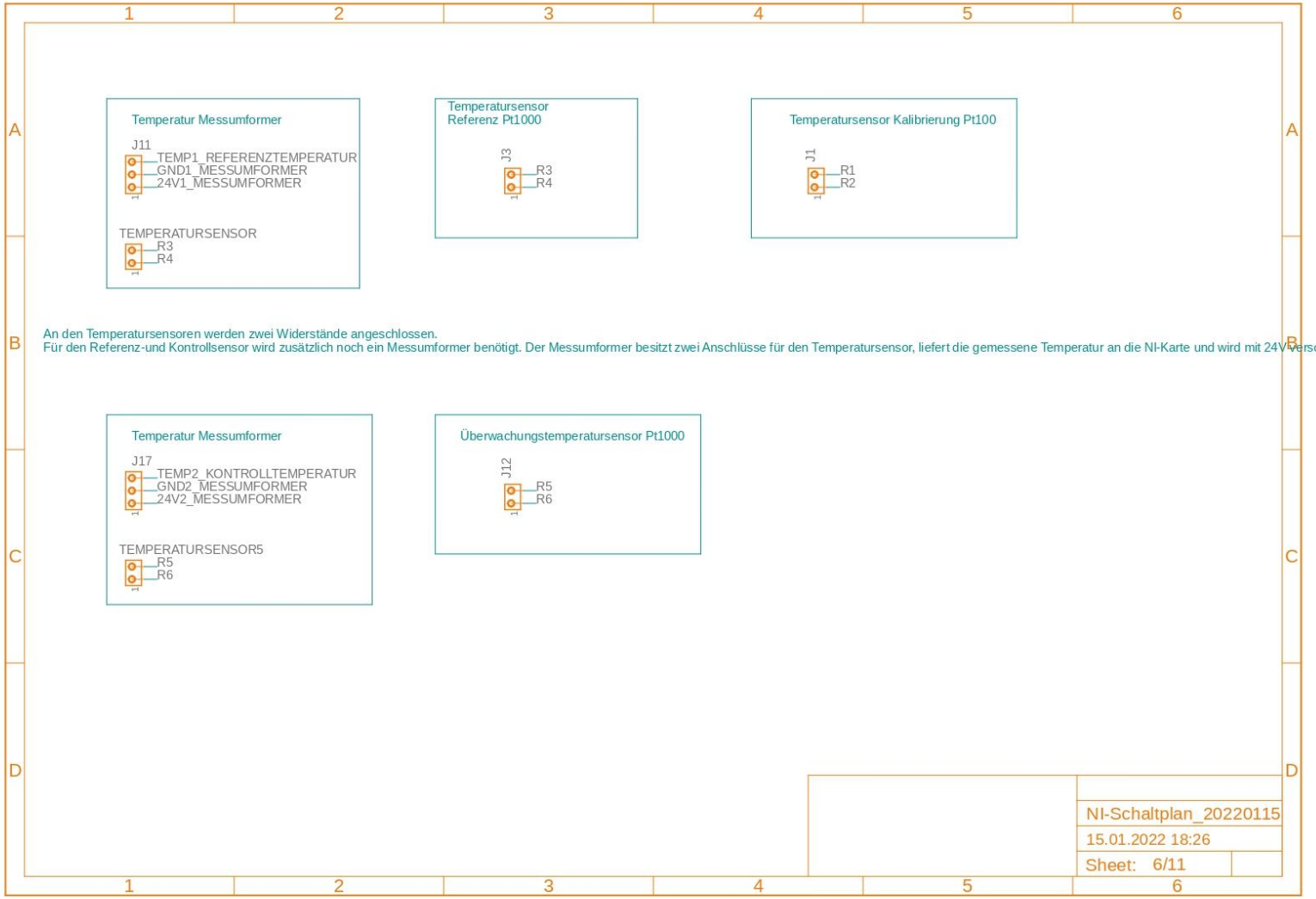
07.02.2022 16:19 C:\Users\Semih\OneDrive\Desktop\Programmierung Prüfstand\NI-Messkarte\NI-Schaltplan_20220115.sch (Sheet: 4/11)

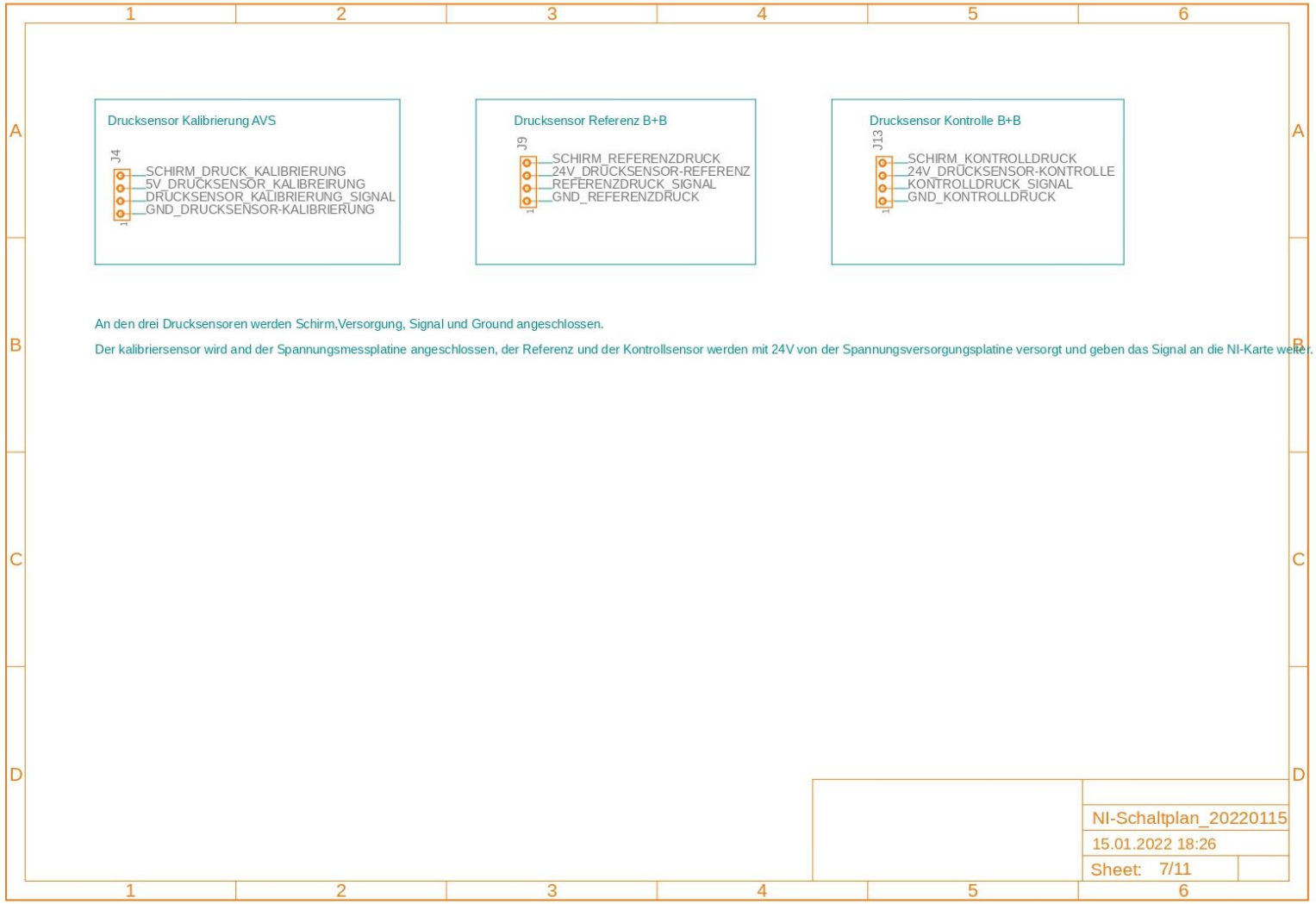


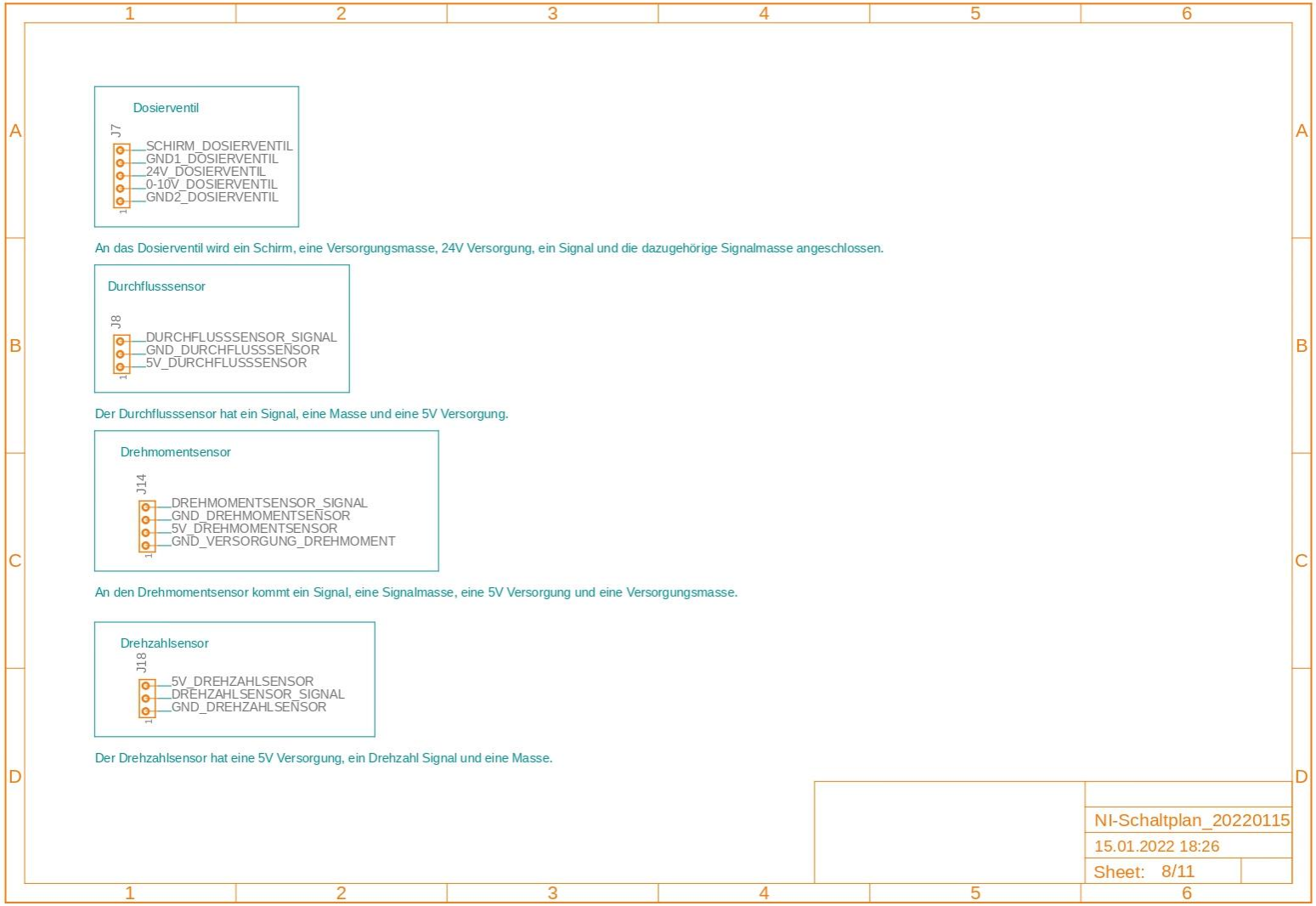
Der Phasenschnittregler wird von zweiten SSR-Insel mit 230V versorgt und liefert dem Heizelement 230V.
Zudem erhält der Regler von der Basisplatine ein Steuersignal und eine Masse. Zusätzlich kann noch ein Potentiometer angeschlossen werden.

Falls eine AC-Pumpe verwendet wird, wird diese auch durch den Phasenschnittregler versorgt und geregelt.

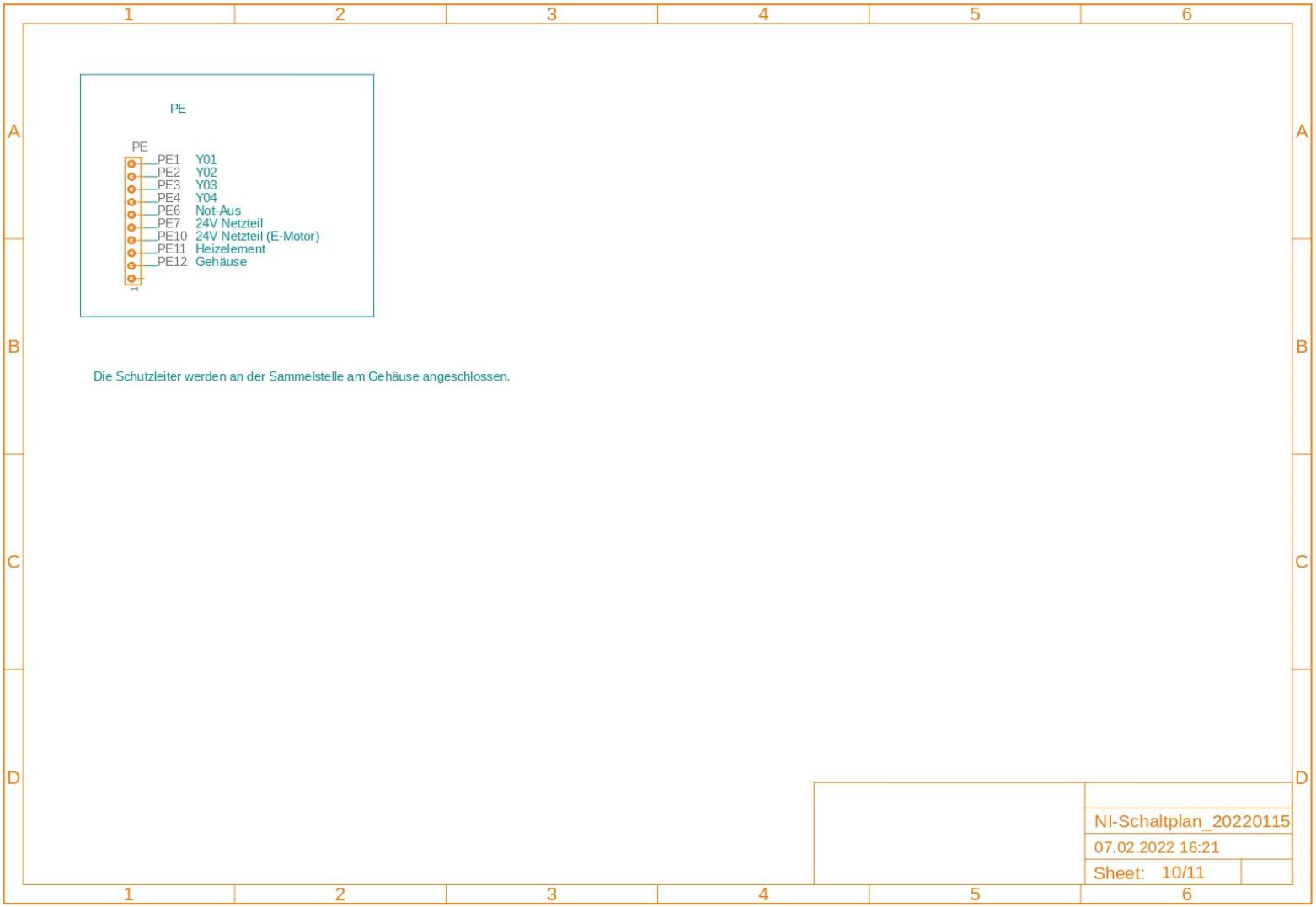
NI-Schaltplan_20220115
15.01.2022 18:26
Sheet: 5/11

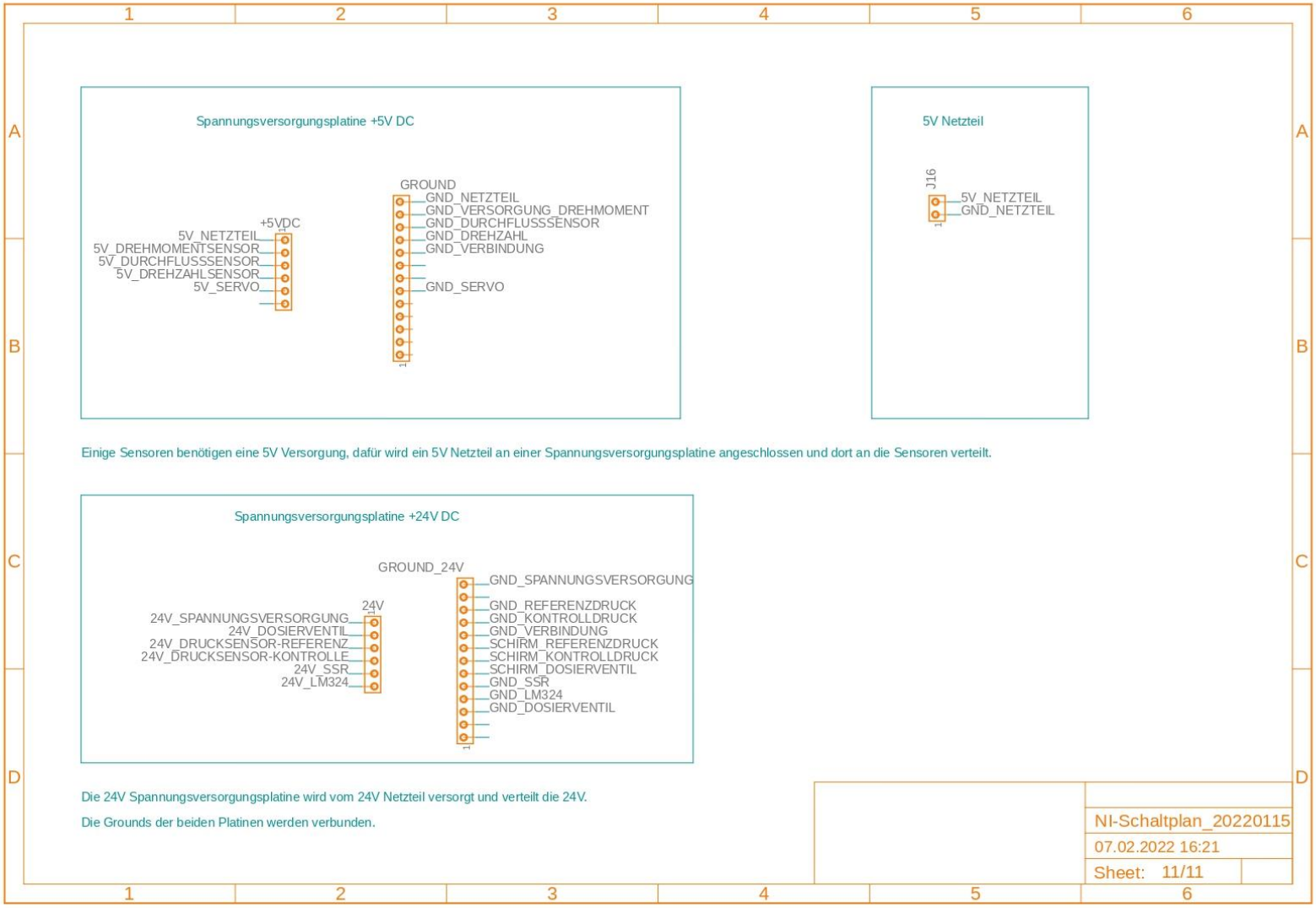

Temperatur Messumformer
J11
TEMP1_REFERENZTEMPERATUR
GND1_MESSUMFORMER
24V1_MESSUMFORMER
TEMPERATURSENSOR
R3
R4
Temperatursensor
Referenz Pt1000
J3
R3
R4
Temperatursensor Kalibrierung Pt100
J1
R1
R2
An den Temperatursensoren werden zwei Widerstände angeschlossen.
Für den Referenz-und Kontrollsensor wird zusätzlich noch ein Messumformer benötigt. Der Messumformer besitzt zwei Anschlüsse für den Temperatursensor, liefert die gemessene Temperatur an die NI-Karte und wird mit 24V versorgt.
Temperatur Messumformer
J17
TEMP2_KONTROLLTEMPERATUR
GND2_MESSUMFORMER
24V2_MESSUMFORMER
TEMPERATURSENSOR5
R5
R6
Überwachungstemperatursensor Pt1000
J12
R5
R6
NI-Schaltplan_20220115
15.01.2022 18:26
Sheet: 6/11


Drucksensor Kalibrierung AVS
J4
SCHIRM_DRUCK_KALIBRIERUNG
5V_DRUCKSENSOR_KALIBREIRUNG
DRUCKSENSOR_KALIBRIERUNG_SIGNAL
GND_DRUCKSENSOR-KALIBRIERUNG
Drucksensor Referenz B+B
J9
SCHIRM_REFERENZDRUCK
24V_DRUCKSENSOR-REFERENZ
REFERENZDRUCK_SIGNAL
GND_REFERENZDRUCK
Drucksensor Kontrolle B+B
J13
SCHIRM_KONTROLLDRUCK
24V_DRUCKSENSOR-KONTROLLE
KONTROLLDRUCK_SIGNAL
GND_KONTROLLDRUCK
An den drei Drucksensoren werden Schirm,Versorgung, Signal und Ground angeschlossen.
Der kalibriersensor wird and der Spannungsmessplatine angeschlossen, der Referenz und der Kontrollsensor werden mit 24V von der Spannungsversorgungsplatine versorgt und geben das Signal an die NI-Karte weiter.
NI-Schaltplan_20220115
15.01.2022 18:26
Sheet: 7/11

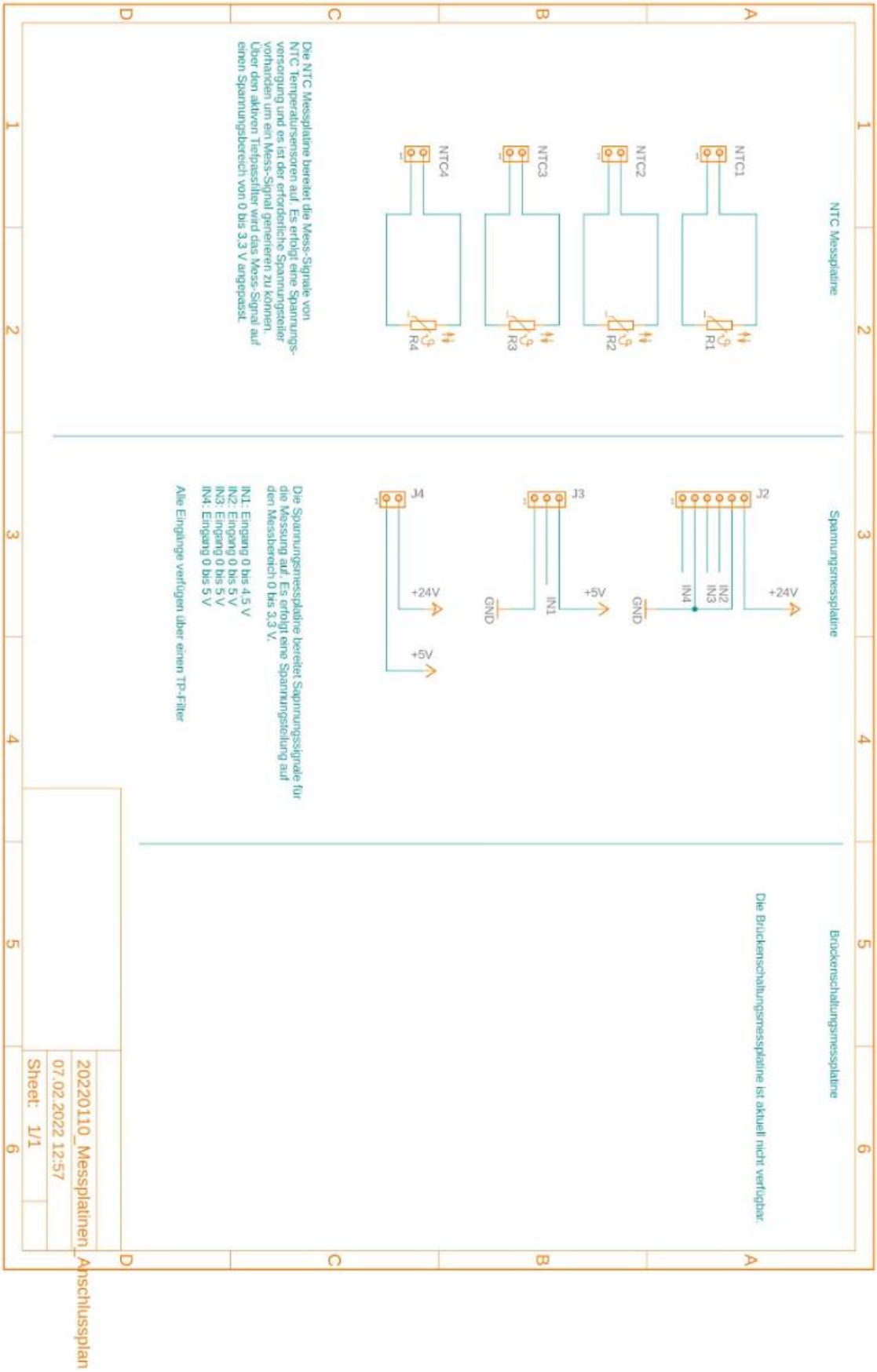

Dosierventil
J7
SCHIRM_DOSIERVENTIL
GND1_DOSIERVENTIL
24V_DOSIERVENTIL
0-10V_DOSIERVENTIL
GND2_DOSIERVENTIL
An das Dosierventil wird ein Schirm, eine Versorgungsmasse, 24V Versorgung, ein Signal und die dazugehörige Signalmasse angeschlossen.
Durchflusssensor
J8
DURCHFLUSSSENSOR_SIGNAL
GND_DURCHFLUSSSENSOR
5V_DURCHFLUSSSENSOR
Der Durchflusssensor hat ein Signal, eine Masse und eine 5V Versorgung.
Drehmomentsensor
J14
DREHMOMENTSENSOR_SIGNAL
GND_DREHMOMENTSENSOR
5V_DREHMOMENTSENSOR
GND_VERSORGUNG_DREHMOMENT
An den Drehmomentsensor kommt ein Signal, eine Signalmasse, eine 5V Versorgung und eine Versorgungsmasse.
Drehzahlsensor
J18
5V_DREHZAHLSENSOR
DREHZAHLSENSOR_SIGNAL
GND_DREHZAHLSENSOR
Der Drehzahlsensor hat eine 5V Versorgung, ein Drehzahl Signal und eine Masse.
NI-Schaltplan_20220115
15.01.2022 18:26
Sheet: 8/11


E-Motor DC
J5
1
24V_E-MOTOR
GND_E-MOTOR
Der DC Elektromotor wird mit 24V versorgt.
Falls ein AC Elektromotor verwendet wird, wird dieser durch den Phasenschnittregler gesteuert und mit 230V versorgt.
NI-Schaltplan_20220115
15.01.2022 18:26
Sheet: 9/11


PE
PE
PE1 Y01
PE2 Y02
PE3 Y03
PE4 Y04
PE6 Not-Aus
PE7 24V Netzteil
PE10 24V Netzteil (E-Motor)
PE11 Heizelement
PE12 Gehäuse
Die Schutzleiter werden an der Sammelstelle am Gehäuse angeschlossen.
NI-Schaltplan_20220115
07.02.2022 16:21
Sheet: 10/11


Spannungsversorgungsplatine +5V DC
+5VDC
5V_NETZTEIL
5V_DREHMOMENTSENSOR
5V_DURCHFLUSSSENSOR
5V_DREHZAHLSENSOR
5V_SERVO
GROUND
GND_NETZTEIL
GND_VERSORGUNG_DREHMOMENT
GND_DURCHFLUSSSENSOR
GND_DREHZAHL
GND_VERBINDUNG
GND_SERVO
5V Netzteil
J16
5V_NETZTEIL
GND_NETZTEIL
Einige Sensoren benötigen eine 5V Versorgung, dafür wird ein 5V Netzteil an einer Spannungsversorgungsplatine angeschlossen und dort an die Sensoren verteilt.
Spannungsversorgungsplatine +24V DC
24V
24V_SPANNUNGSVERSORGUNG
24V_DOSIERVENTIL
24V_DRUCKSENSOR-REFERENZ
24V_DRUCKSENSOR-KONTROLLE
24V_SSR
24V_LM324
GROUND_24V
GND_SPANNUNGSVERSORGUNG
GND_REFERENZDRUCK
GND_KONTROLLDRUCK
GND_VERBINDUNG
SCHIRM_REFERENZDRUCK
SCHIRM_KONTROLLDRUCK
SCHIRM_DOSIERVENTIL
GND_SSR
GND_LM324
GND_DOSIERVENTIL
Die 24V Spannungsversorgungsplatine wird vom 24V Netzteil versorgt und verteilt die 24V.
Die Grounds der beiden Platinen werden verbunden.
NI-Schaltplan_20220115
07.02.2022 16:21
Sheet: 11/11

Anhang A2 – Schaltplan Messplatinen



07.02.2022 12:57 C:\Users\SemihOneDrive\Desktop\Programmierung Prüfstand\Anschlussplan\20220207_Messplatinen_Anschlussplan.sch (Sheet: 1/1)

Anhang A3 – Vollständiger Schnittstellencode

```
app.stm32 = serialport('COM6', 115200);          % STM32F411 nucleo
configureTerminator(app.stm32, 'CR/LF');
configureCallback(app.stm32, 'terminator', @app.STM32SerialRead);

zeit = 0.25;            % Pause zwischen den MicroPython Anweisungen
writeline(app.stm32, 'from machine import SoftI2C');
pause(zeit);
writeline(app.stm32, 'from machine import Pin');
pause(zeit);
writeline(app.stm32, 'import mcp23017');
pause(zeit);
writeline(app.stm32, 'from pyb import ADC');
pause(zeit);
writeline(app.stm32, 'from pyb import Timer');
pause(zeit);
writeline(app.stm32, 'import mcp4725');
pause(zeit);

writeline(app.stm32, 'i2c_exp = SoftI2C(scl = Pin("PB6"), sda = Pin("PB7"), freq=400000)');
pause(zeit);
writeline(app.stm32, 'gpio_exp = mcp23017.MCP23017(i2c_exp, 0x20)');
pause(zeit);
writeline(app.stm32, 'gpio_exp.porta.mode = 0b00000000');
pause(zeit);
writeline(app.stm32, 'gpio_exp.portb.mode = 0b00000000');
pause(zeit);
writeline(app.stm32, 'gpio_exp.porta.gpio = 0b00000000');
pause(zeit);
writeline(app.stm32, 'gpio_exp.portb.gpio = 0b00000000');
pause(zeit);
writeline(app.stm32, 'gpio_exp.pin(1, value = 0)');
pause(zeit);
writeline(app.stm32, 'gpio_exp.pin(0, value = 0)');
pause(zeit);
writeline(app.stm32, 'gpio_exp.pin(8, value = 0)');
pause(zeit);
writeline(app.stm32, 'gpio_exp.pin(9, value = 0)');
pause(zeit);
writeline(app.stm32, 'gpio_exp.pin(10, value = 0)');
pause(zeit);
writeline(app.stm32, 'gpio_exp.pin(11, value = 0)');
pause(zeit);
writeline(app.stm32, 'gpio_exp.pin(12, value = 0)');
pause(zeit);
writeline(app.stm32, 'gpio_exp.pin(13, value = 0)');
pause(zeit);
writeline(app.stm32, 'gpio_exp.pin(14, value = 0)');
pause(zeit);
writeline(app.stm32, 'gpio_exp.pin(15, value = 0)');
pause(zeit);
writeline(app.stm32, 'T_Boiler = ADC("PA0")');
pause(zeit);
writeline(app.stm32, 'T_Befuellung = ADC("PA1")');
pause(zeit);
writeline(app.stm32, 'T_Eingang = ADC("PA4")');
```

```
pause(zeit);
writeline(app.stm32, 'Leitwert = ADC("PC2")');
pause(zeit);
writeline(app.stm32, 'P_Gruppe = ADC("PC1")');
pause(zeit);
writeline(app.stm32, 'Taste = ADC("PC3")');
pause(zeit);
writeline(app.stm32, 'P_Boiler = ADC("PC0")');
pause(zeit);
writeline(app.stm32, 'T_Zwischenraum = ADC("PA5")');
pause(zeit);
writeline(app.stm32, 'Gewicht1 = ADC("PA6")');
pause(zeit);
writeline(app.stm32, 'T_Mischer = ADC("PA7")');
pause(zeit);
writeline(app.stm32, 'P_Boiler_Alt = ADC("PB1")');
pause(zeit);
writeline(app.stm32, 'Gewicht2 = ADC("PC2")');
pause(zeit);
writeline(app.stm32, 'Fuell_1 = Pin("PB15", Pin.IN, Pin.PULL_UP)');
pause(zeit);
writeline(app.stm32, 'Fuell_2 = Pin("PC8", Pin.IN, Pin.PULL_UP)');
pause(zeit);
writeline(app.stm32, 'i2c_dac = SoftI2C(scl = Pin("PB10"), sda = Pin("PB9"), freq=400000)');
pause(zeit);
writeline(app.stm32, 'dosierventil = mcp4725.MCP4725(i2c_dac, mcp4725.BUS_ADDRESS[1])');
pause(zeit);
writeline(app.stm32, 'dosierventil.write(0)');
pause(zeit);
writeline(app.stm32, 'appoldt = mcp4725.MCP4725(i2c_dac, mcp4725.BUS_ADDRESS[0])');
pause(zeit);
writeline(app.stm32, 'appoldt.write(0)');
pause(zeit);
writeline(app.stm32, 'timer1 = Timer(1, freq = 8)');
pause(zeit);
writeline(app.stm32, 'timer3 = Timer(3, freq = 50))');
pause(zeit);
writeline(app.stm32, 'timer4 = Timer(4, freq = 50)');
pause(zeit);
writeline(app.stm32, 'PWM_Dampf = timer4.channel(4, mode = Timer.PWM, pin = Pin("PB8"))');
pause(zeit);
writeline(app.stm32, 'PWM_Dampf.pulse_width_percent(0)');
pause(zeit);
writeline(app.stm32, 'PWM_Entnahme = timer3.channel(1, mode = Timer.PWM, pin = Pin("PC6"))');
pause(zeit);
writeline(app.stm32, 'PWM_Entnahme.pulse_width_percent(0)');
pause(zeit);
writeline(app.stm32, 'PWM_Licht = timer3.channel(2, mode = Timer.PWM, pin = Pin("PC7"))');
pause(zeit);
writeline(app.stm32, 'PWM_Licht.pulse_width_percent(0)');
pause(zeit);
writeline(app.stm32, 'PWM_Dosierventil = timer3.channel(4, mode = Timer.PWM, pin = Pin("PC9"))');
pause(zeit);
writeline(app.stm32, 'PWM_Dosierventil.pulse_width_percent(0)');
pause(zeit);
writeline(app.stm32, 'gpio_exp.porta.gpio = 0b00000000');
pause(zeit);
writeline(app.stm32, 'gpio_exp.portb.gpio = 0b00000000');
```

Anhang A4 – Schwingungsmessung der Getriebepumpe durchgeführt von A. Rohnen