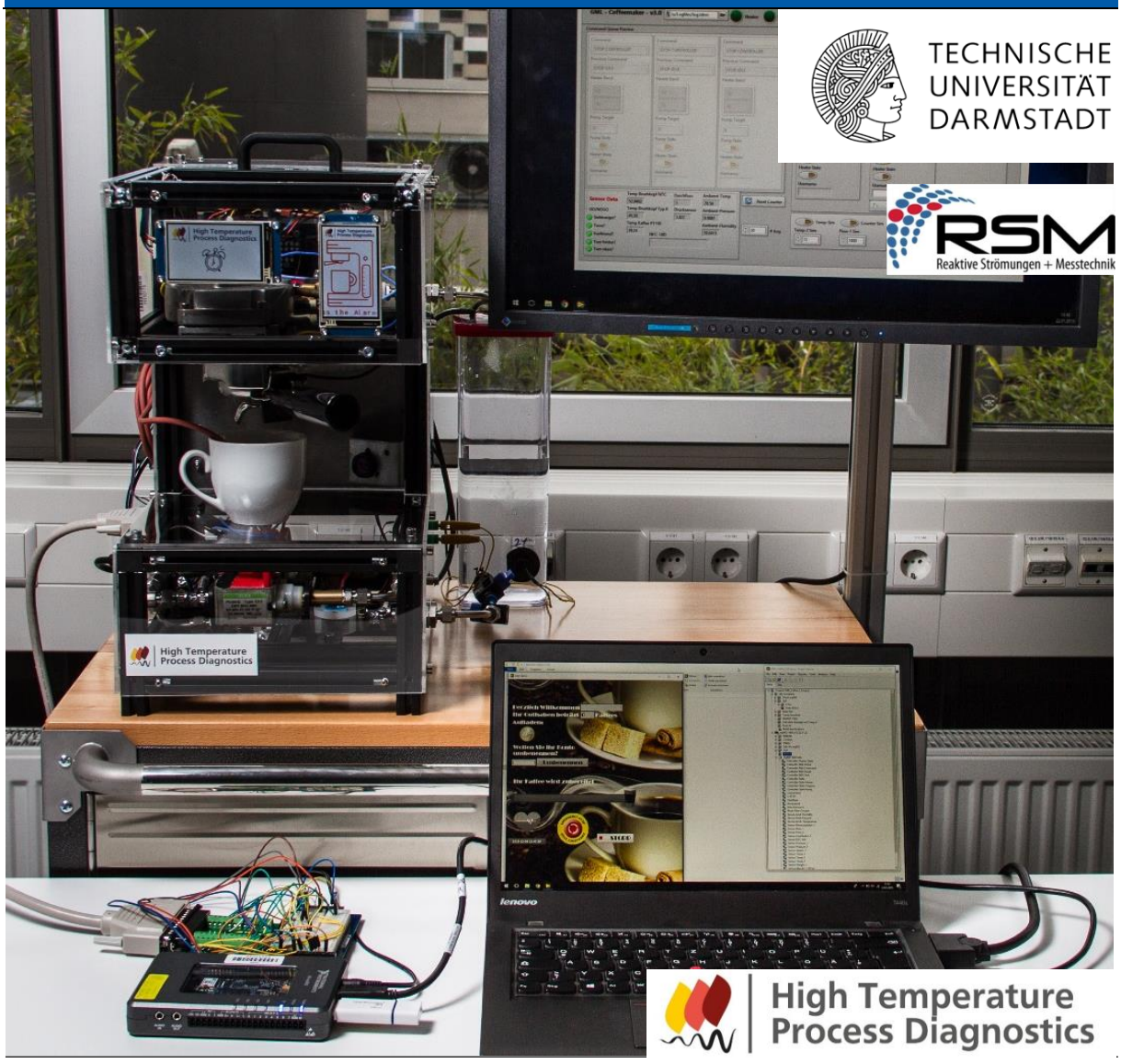
# GML – Espresso Maker 1.0 Abschlussbericht

**Projektarbeit**
Jan Dreiskemper | 2896937
Jan Radzey | 2648525
Annemike Unterschütz | 2987383
Florian Vierneisel | 2811491

Grundlagen der Messtechnik mit LabVIEW – WS 2018/2019




TECHNISCHE UNIVERSITÄT DARMSTADT

RSM
Reaktive Strömungen + Messtechnik

High Temperature
Process Diagnostics

**Jan Dreiskemper**
Matrikelnummer: 2896937
Studiengang: Informatik

**Jan Radzey**
Matrikelnummer: 2648525
Studiengang: Mechanical and Process Engineering

**Annemike Unterschütz**
Matrikelnummer: 2987383
Studiengang: Informatik

**Florian Vierneisel**
Matrikelnummer: 2811491
Studiengang: Mechatronik

**Projektarbeit - Abschlussbericht**
**Thema:** "GML Espressomaker 1.0"

**Eingereicht:** 6. Februar 2019

**Betreuer:**

Dr. rer.-nat. Steven Wagner
High Temperature Process Diagnostics
Technische Universität Darmstadt
Otto-Berndt-Str. 3
64287 Darmstadt

Luigi Biondo, M.Sc

Anna Schmidt, M.Sc

Henrik Schneider, M.Sc

# 1 Software-Struktur

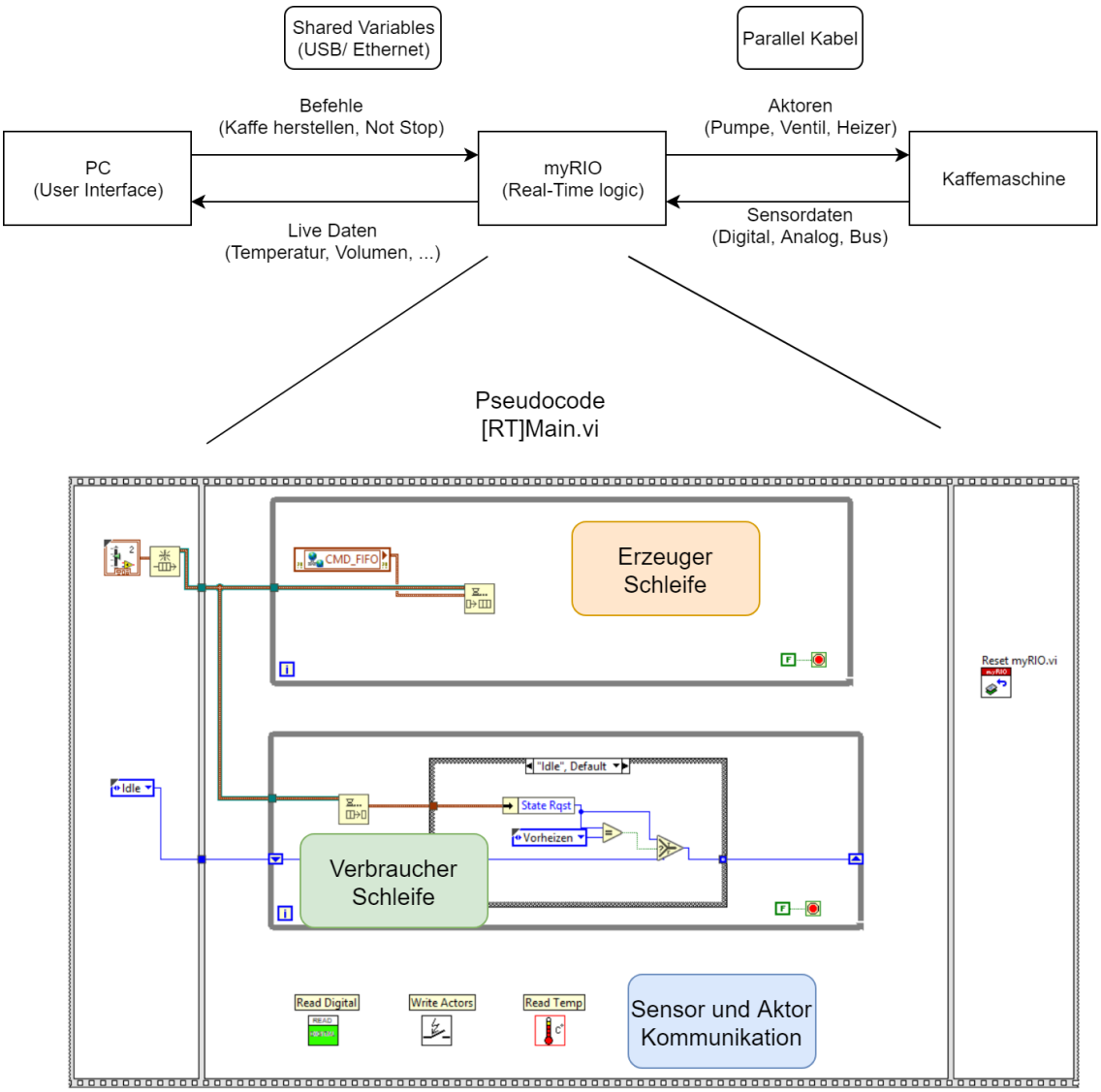



Abbildung 1: Überblick der Gesamtstruktur

Der Code teilt sich in zwei Teile auf. Zum einen in den Code, der auf einem PC ausgeführt wird und den der auf dem myRIO-Realtime-System ausgeführt wird. Der Code auf dem PC erfüllt alle Aufgaben des Nutzer- und Engineering-Interfaces, wie zum Beispiel das Bestellen eines Kaffees, den Not-Stopp und das Anzeigen wichtiger Prozessvariablen. Die Kommunikation zwischen PC und Realtime-System erfolgt ausschließlich über die NI Shared-Variables-Engine. Während Live-Informationen wie Temperaturen vom myRIO nicht gebuffert in den Shared-Variablen veröffentlicht werden, werden Befehle vom PC zum myRIO als gebufferte Netzwerkvariable (CMD_FIFO) übergeben. Dadurch wird eine Befehls-FIFO realisiert die sicherstellt, dass keine Befehle verloren gehen.

Auf Seite des myRIO wird die Befehls-FIFO nach dem Erzeuger-Verbraucher-Prinzip ausgelesen. Dadurch kann sichergestellt werden, dass die Befehls-FIFO nicht überläuft. Des Weiteren wurde hier auch die Anforderung umgesetzt, dass „Emergency Stops" priorisiert verarbeitet werden sollen. Wenn der Befehl „Emergency Stop" aus der Netzwerkvariable ausgelesen wird, wird dieser in die myRIO interne Erzeuger-Verbraucher-Warteschlange an erster Stelle eingereiht und damit in der nächsten Iteration der Controller-Zustandsmaschine umgesetzt. Mehr dazu findet sich im Abschnitt "Not-Stopp Strategie".

## 1.1 Main-VI

Wie in der Abbildung 1 des Abschnittes zu sehen ist, besteht die [RT]Main.vi aus einer Sequenzstruktur mit drei Sequenzen (Initialisierung, Loop, Terminierung). In der zweiten Sequenz findet sich das Kernstück des Codes. Eine Erzeugerschleife nimmt Befehle von der Befehls-FIFO an, eine Verbraucherschleife setzt die Zustandslogik zum Herstellen des Kaffees um und es finden sich mehrere Module zum Lesen der angeschlossenen Sensoren und dem Steuern der Aktoren. Die Kommunikation zwischen der Zustandsmaschine und den Modulen geschieht über globale Variablen. Das Nutzen von globalen Variablen als primärer Weg der Datenübertragung zwischen Modulen war eine Designentscheidung. Durch ihre Nutzung wird der Code sehr modular und damit austauschbar/ersetzbar. Ebenfalls ist durch das gezielte ausklammern bestimmter Blöcke eine einfache Offline-Simulation des Verhaltens ohne angeschlossene Kaffeemaschine möglich.

## 1.2 Controller für den Heizer und die Pumpe

Heizer, Ventil und Pumpe werden in „Actuation.vi“ angesteuert, welche in der „Main.vi“ aufgerufen wird. Die globalen Variablen für den Zustand der Aktoren werden in die entsprechenden DIO-Ports geschrieben, um so ein high- bzw. low-Signal zu erzeugen. Dieser Schritt wird alle 10 ms ausgeführt.

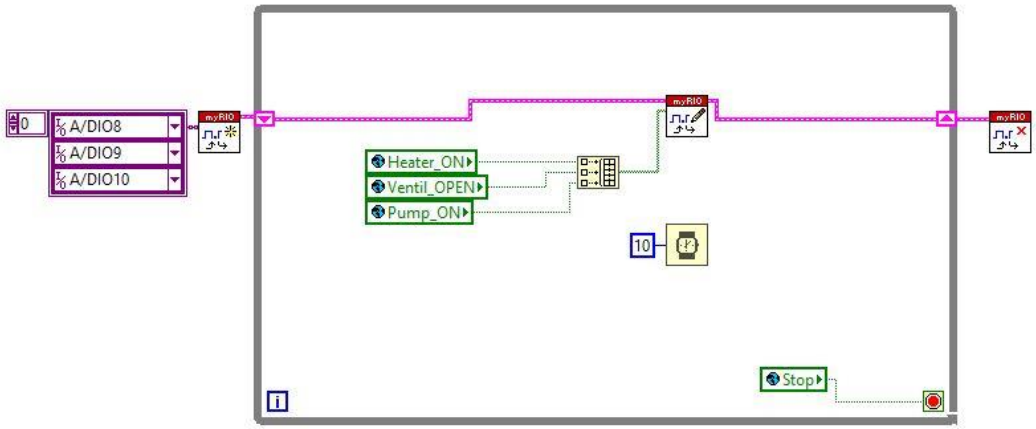



Abbildung 2: Code der Ansteuerung der Aktoren

Die Variablen werden in der „Loop_Controller.vi“ gesetzt, welche ebenfalls direkt in der „Main.vi“ aufgerufen wird. In den Zuständen „Idle“ und „Emergency-Stop“ sind die globalen Variablen auf „false“ gesetzt, wodurch alle Aktoren ausgeschaltet werden. In den Zuständen „Vorheizen“ und „Brühen“ wird die Variable „Heater_ON“ in „CTRL_Heater.vi“ gesetzt. Dabei ist „Heater_ON“ nur „true“, wenn die gemessene Temperatur niedriger als die Solltemperatur ist.

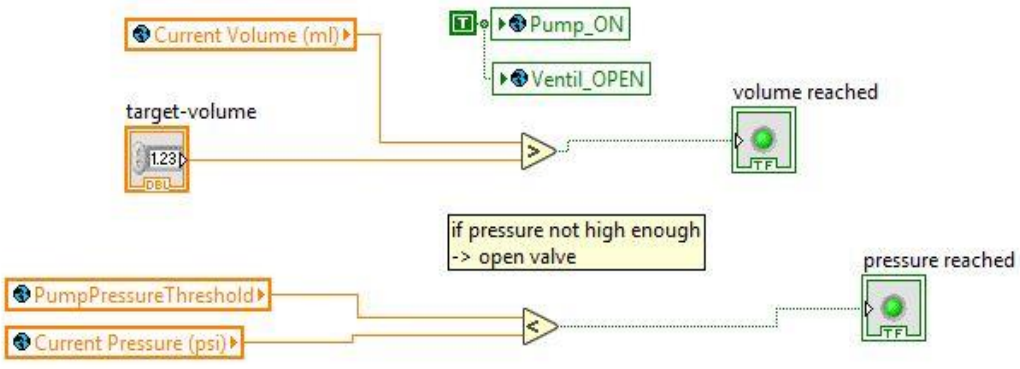



Abbildung 3: Darstellung der CTRL_PumpValve.vi

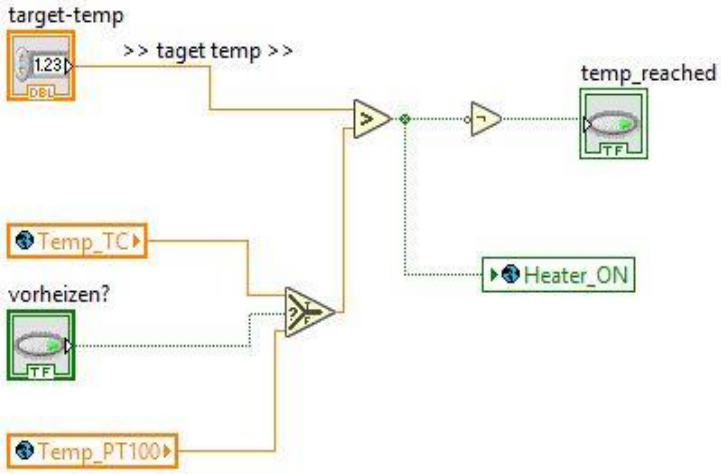



Abbildung 4: Darstellung CTRL_Heater.vi

Die Variablen „Pump_ON“ und „Ventil_OPEN“ werden nur im Zustand „Brühen“ in „CTRL_PumpValve.vi“ auf „true“ gesetzt. Sobald das geforderte Volumen von der Pumpe gepumpt wurde, geht der Loop-Controller in den Zustand „Idle“ zurück und alle Variablen werden auf „false“ gesetzt, was die Aktoren ausschaltet.

## 1.3 Controller für das Touch-Display

Wir haben uns entschieden, zwei verschiedene GUIs, jeweils eins pro Display, an der Kaffeemaschine zu verwenden. Für das Quer-Display wählten wir ein Standbild, welches thematisch zu dem Thema Kaffee passt.


Abbildung 5: Anzeige „Display 1“ (Quer)

Das Hoch-Display nutzen wir zur Darstellung der eigentlichen Information. Zum einen wird angezeigt, wie viele Tassen Kaffee der Nutzer angefordert hat. Außerdem wird angezeigt, in welchen der Hauptzustände sich die Maschine momentan befindet („Idle“, „Vorheizen“, „Brühen“, „Emergency-Stop“). Als drittes Element entschieden wir uns für einen Fortschrittsbalken, welcher den aktuellen prozentualen Füllstand des Kaffeebechers relativ zum Zielfüllstand illustriert.

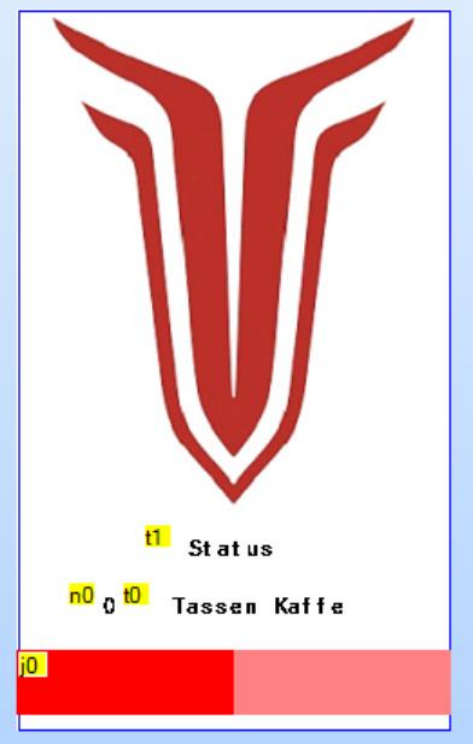


Abbildung 6: Anzeige „Display 2“ (Portrait)

Die beiden Bildschirm-Bilder entwickelten wir wie vorgeschlagen in dem Nextion Editor und exportierten sie als .hmi und .tft. Leider war unsere Transferplatine defekt, weswegen wir die Daten mit einer Micro-SD Karte auf den Monitor überspielen mussten. Dies funktionierte dann reibungslos, wenn auch mit erheblichem Mehraufwand.

In der folgenden Abbildung 7 sieht man den Aufbau dieser VI.

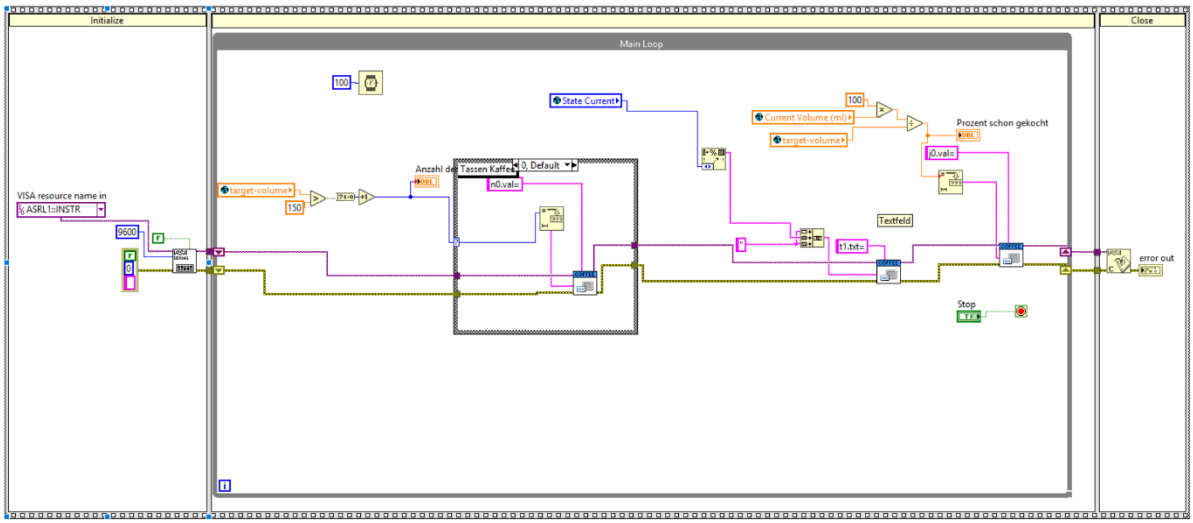



Abbildung 7: Darstellung der DisplayMain.vi

Innerhalb der LabVIEW Implementierung verwenden wir die gegebene Display-Message VI. Diese bekommt als Input die Bezeichnung des Elements (t1 = Status, n0 = Anzahl der Tassen, j0 = Fortschrittsbaken) welche wir der Bildschirm-Datei über Nextion hinzugefügt haben. In dieses schreiben wir dann die globalen Variablen, welche den Fortschritt, den aktuellen Zustand und die Anzahl der angeforderten Kaffees wiedergibt.

## 1.4 Not-Stopp

Der Not-Stopp besitzt besondere Priorität. Da ein zeitnahes Abbrechen aller Vorgänge wichtig ist, wird auch der Befehl „Not-Stopp" in allen Instanzen sofort umgesetzt. Wie bereits in Abschnitt „Codestruktur" beschrieben, wird jeder Befehl in der Erzeugerschleife aus der Shared-Variable CMD_FIFO entnommen. Wird der Befehl „Emergency-Stop" erkannt, so wird er in der Befehlswarteschlange auf dem myRIO priorisiert und am Anfang der Warteschlange eingefügt (siehe Abbildung 8).

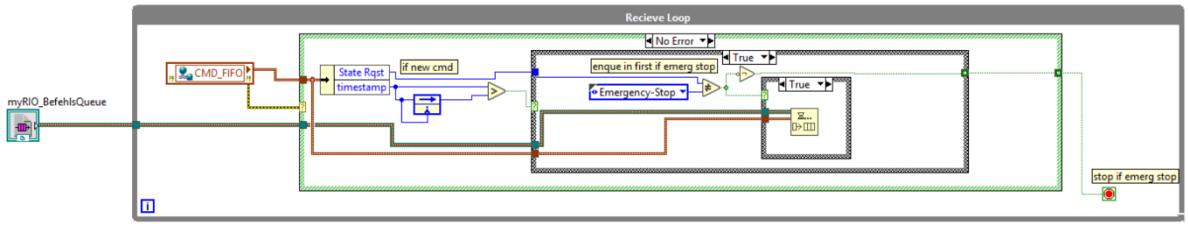



Abbildung 8: Error-Handling in der Command-Queue

Nachdem nun der „Emergency-Stop" Befehl priorisiert weitergegeben wird, muss auch die Zustandsmaschine, die den Brühprozess steuert, auf den Befehl reagieren. Um sofortiges Abbrechen zu gewährleisten ist der Zustandsmaschine eine Logik vorgeschaltet. In folgender Abbildung ist zu sehen, dass der aktuelle Zustand, der über das Schieberegister in die nächste Schleifeniteration getragen wird, durch den „Emergency-Stop"-Zustand ersetzt wird. Im „Emergency Stop"-Zustand werden nun Pumpe, Ventil und Heizer abgeschaltet bevor die Controller Schleife komplett beendet wird. Durch das Setzen der globalen Variable „Stop" werden auch alle Sensor- und Aktor-Module beendet. Danach geht die übergeordnete Sequenzstruktur in die letzte Sequenz über und beendet/resettet dan myRIO (siehe Abbildung 1 unter Kapitel 1).

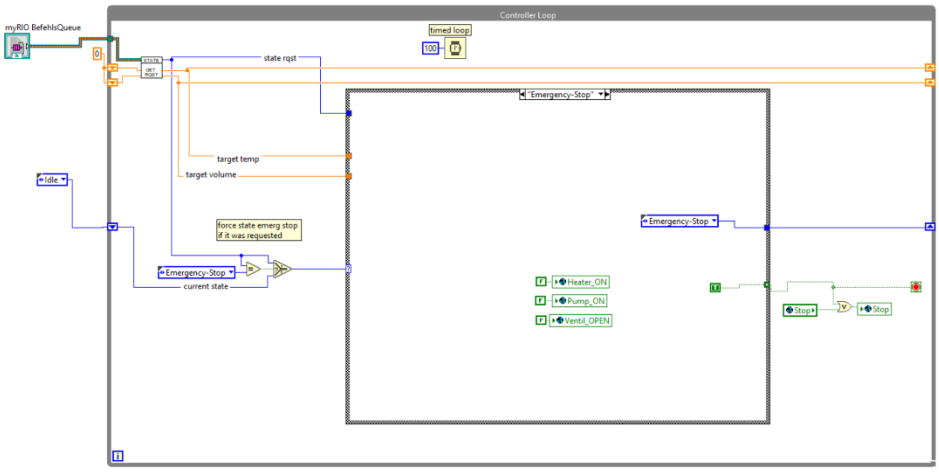



Abbildung 9: Emergency-Stop im Controller-Loop

## 1.5 User-Interface

Die User GUI ist übersichtlich aufgebaut. Über die Einstellungen „Temperatur" und „Volumen" lassen sie die gewünschte Temperatur und das Volumen einer Tasse einstellen. Über „Kaffee Anfordern" kann eine Kaffee Tasse Kaffee angefordert werden. Der Button „großer Kaffee anfordern" liefert die doppelte Menge.

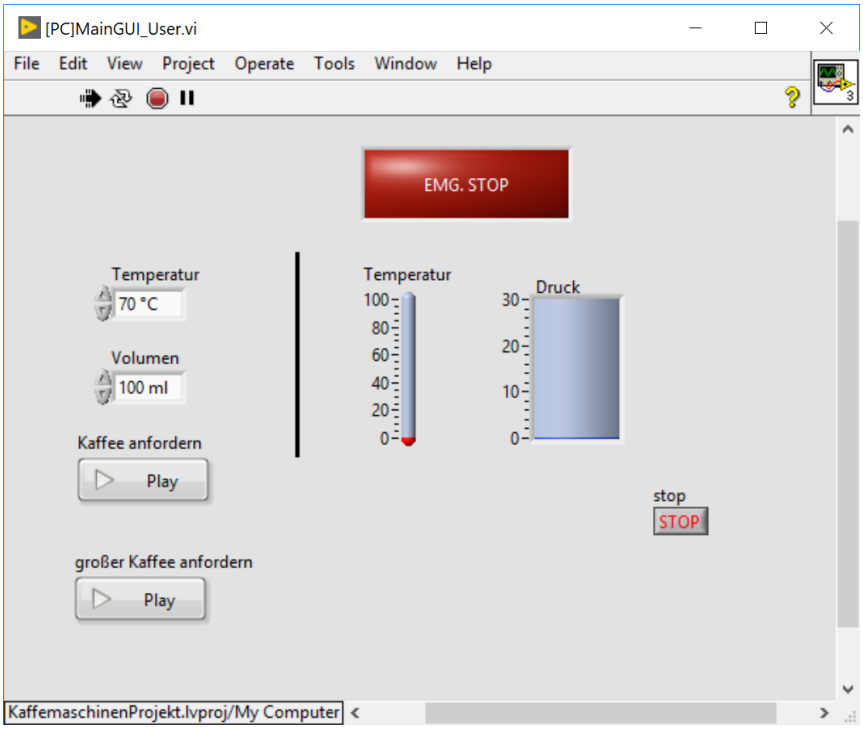



Abbildung 10: User GUI

Im Blockdiagramm wird beim Drücken die VI „[PC]SendCMD.vi" aufgerufen. Sie sendet daraufhin einen neuen Befehl in die Befehls-FIFO des myRIO. Das senden des Zustandes „Vorheizen" entspricht der Anfrage für einen neuen Kaffee.

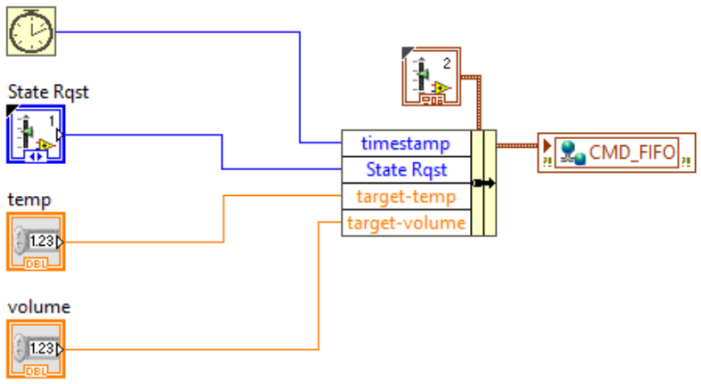



Abbildung 11: Erstellen eines neuen Clusters, inklusive Timestamp

In der „[PC]SendCMD.vi" wird ein neues Cluster erstellt, das als Befehl an das myRIO geschickt wird. Durch einen Timestamp kann überprüft werden, ob ein Befehl neu ist.

Die Engineering GUI oder auch Entwickler GUI erfüllt die Funktion einen visuell gut verständlichen Einblick in den Brühprozess zu geben. Dafür haben wir die wie im Abschnitt 4 beschriebene Synchronisation der globalen Variablen genutzt, um das Frontpanel mit Informationen zu füllen. In der folgenden Abbildung findet sich die Oberfläche der Engineering GUI.

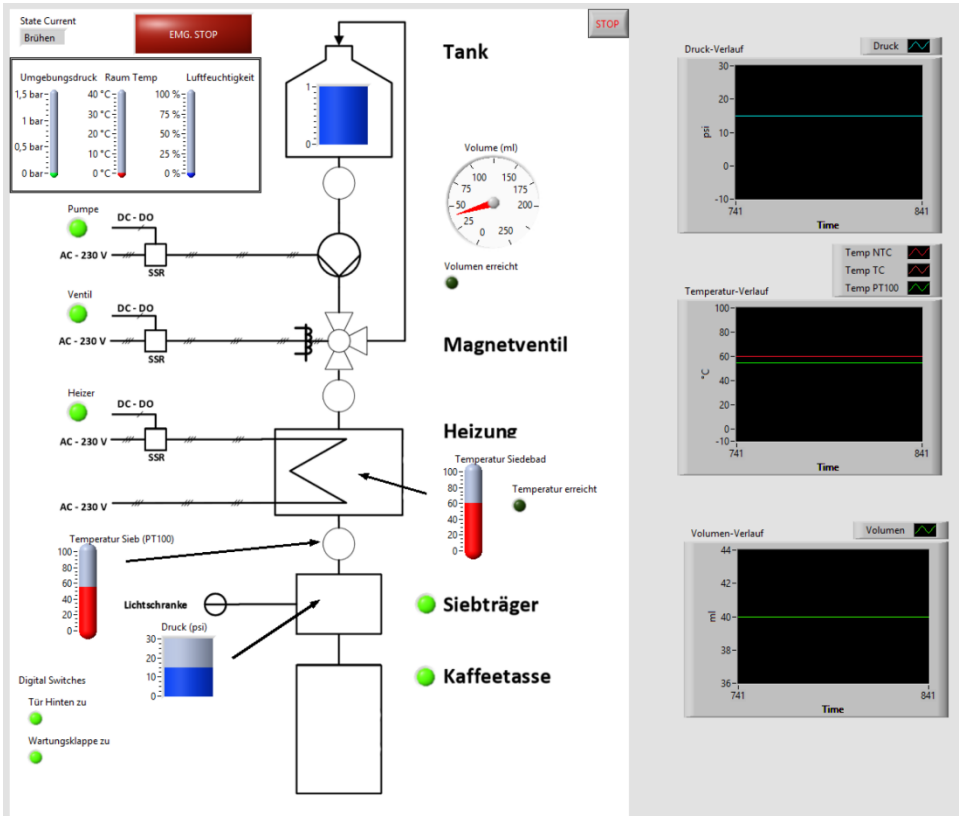


Abbildung 12: Engineering GUI

Über den Button Oben Links lässt sich der Not-Stopp auslösen. Weitere Steuer-Möglichkeiten gibt es in der Entwickler GUI nicht.

## 1.6 Sensor-Daten Erfassung

Für die Erfassung der Sensoren wird ein modularer Aufbau gewählt. Der Grund dafür ist die schnelle Austauschbarkeit und Isolierung des Gesamtsystems der Sensorauswertung. Dadurch lassen sich bei der Inbetriebnahme der Kaffeemaschine fehlerhafte Sensordaten einfach vom System abkapseln, um andere Komponenten in der zur Verfügung stehenden Zeit zu testen.

Dazu wurde für die Sensoren eine eigene Sub-VI erstellt, welche dann in das System integriert wird. Um bei weiteren Anpassungen an der Sensordaten-Erfassung eine Kompatibilität zum

Gesamtsystem zu ermöglichen, wurde sich auf feste globale Variablen geeinigt, welche bei Veränderungen gleichbleiben müssen. Somit ist auch eine gleichzeitige Veränderung des Gesamtsystems wie auch der Sub-VIs möglich.

### 1.6.1 Binäre Sensoren

Für die Auswertung der Schalter zur Absicherung der Türen hinten wie oben, sowie dem Sensor zur Überwachung, ob eine Tasse unter der Kaffeemaschine steht, dem Wasserstandsensor und dem Sensor zur Überwachung des korrekten Verschlusses des Siebträgers wird eine Sub-VI verwendet, welche die Erfassung dieser Sensorwerte bündelt. Anschließend werden die als Array gesammelten Daten aufgeteilt und in einzelnen globalen Variablen geschrieben. Dies erfolgt dabei alle 10 ms in einer While-Schleife. Der genauen Aufbau ist in der Abbildung 13 zu sehen.

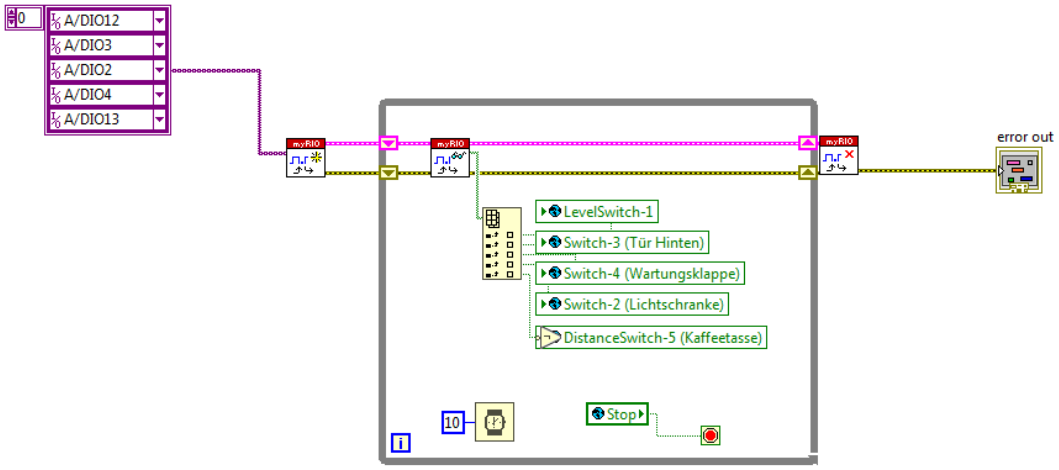



Abbildung 13: Auswertung der sicherheitsrelevanten Sensoren

### 1.6.2 Thermoelement

Ein Sensor zur Temperaturmessung ist ein Thermoelement Typ K was mit einem MAX31855 Breakout-Board verbunden ist. Dieses Board beinhaltet einen Messverstärker sowie eine Kaltstellenkompensation und gibt die Daten über einen SPI-Bus aus. Durch die Kaltstellekompensation kann dabei mittels einer Messung der Referenztemperatur an der Referenzstelle auf die Messtemperatur des Thermoelements geschlossen werden. Der Messverstärker verstärkt dabei die geringe Spannung des Thermoelements, sodass dieses einfacher ausgewertet werden kann. Außerdem ermöglicht das Breakoutboard eine Linearisierung der Messwerte, welche dann über den SPI-Bus ausgegeben werden. Die Auswertung der Messwerte auf dem Bus erfolgt dabei über den in Abbildung 14 dargestellten Code.

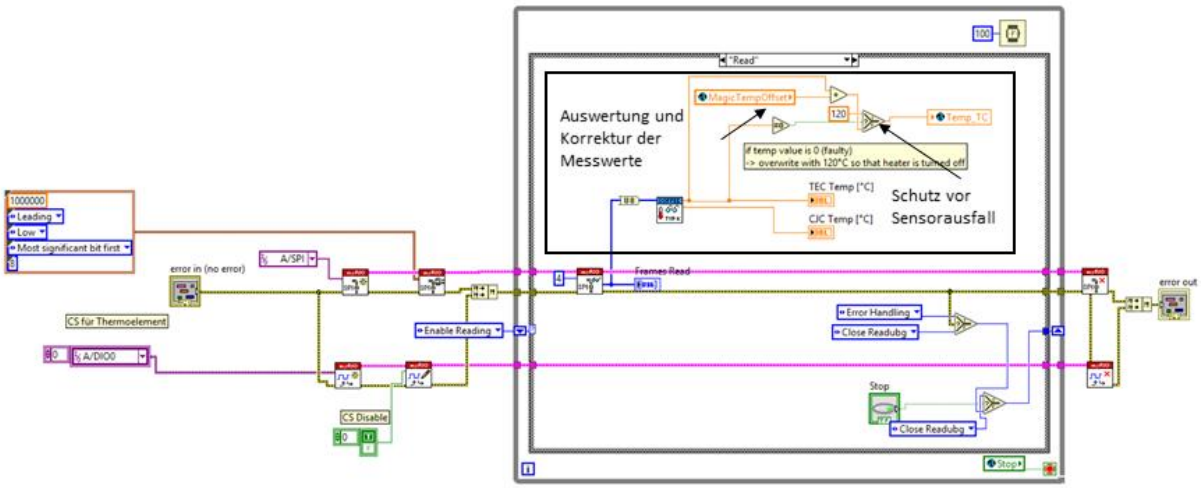



Abbildung 14: Auswertung des Thermoelements

Dabei werden die Daten des SPI-Bus mittels einer Sub-VI in Temperaturwerte umgewandelt, und zum Ausgleich von Messungenauigkeiten mit einem Offset von +10° C versehen. Außerdem ist eine Sicherung vorhanden, falls der Sensor ausfällt und 0° C anzeigt wir der Wert mit 120° C sodass, der Heizer zur Sicherheit abschaltet. Anschließend wird der Temperaturwert in eine globale Variable geschrieben.

### 1.6.3 PT100

Neben der Temperaturmessung mittels des Thermoelements wird diese vor dem Siebträger im System über ein PT100–Messwiderstand erfasst. Dieser ist an einem Messumformer angeschlossen welcher proportional zur Messtemperatur einen eine Spannung ausgibt. Dabei repräsentiert 0 V eine Temperatur von 0° C und 10 V 200° C. Die Spannung kann über einen analogen Input des myRIO gemessen werden und ergibt multipliziert mit 20 die tatsächliche Temperatur. Diese wird dann in eine globale Variable geschrieben. Da sich der Messwiderstand größtenteils linear über diesen Temperaturbereich verhält und auch das Übertragungsverhalten des Messumformers linear ist, kann über den Spannungsbereich linear interpoliert werden, um auf die Messtemperatur zu schließen. Dabei muss beachtet werden, dass die maximale Spannung des Messumformers oberhalb der zulässigen Eingangsspannung des myRIO von 5 V liegt, da aber die zu erwartende Temperatur unterhalb 100° C, was 5 V entspricht, liegt und damit die resultierende Spannung am myRIO kleiner als die maximal erlaubten 5 V ist, muss kein zusätzlicher Spannungsteiler verwendet werden. Der Code für die Auswertung ist in Abbildung 15 zu sehen.

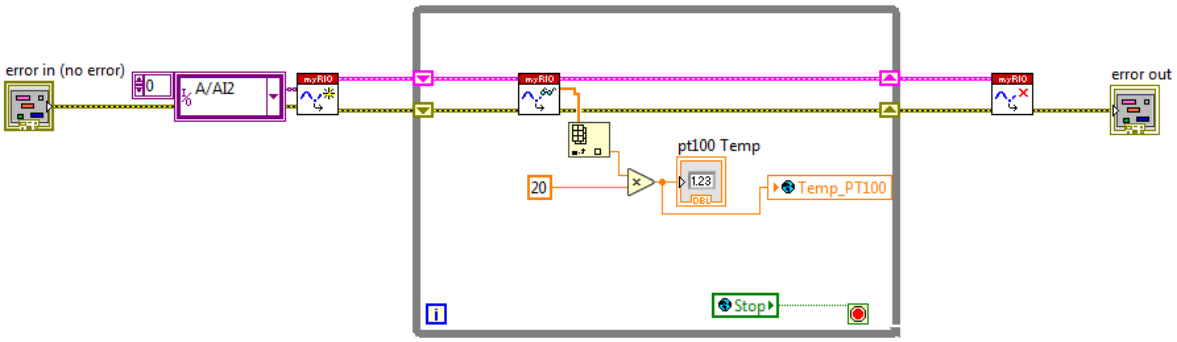



Abbildung 15: Auswertung der Temperaturmessung des PT100-Messwiderstands

### 1.6.4 NTC

Die Temperaturerfassung durch den NTC ist durch einen einfachen Spannungsteiler realisiert. Die Messungenauigkeit ist trotz dieser einfachen Schaltung gering, da die Leitungswiderstände in Vergleich zum eingesetzten Messwiderstand (10 kΩ) und dem Widerstand des NTC-Elements als sehr klein angenommen werden können. Zudem ist eine Änderung der Leiterwiderstände durch Erhitzung in Folge von Stromfluss als sehr klein anzunehmen, da die Schaltung aus großen Widerständen bei geringer Spannung besteht. Die Berechnung des NTC-Widerstandes geschieht durch einen Formula-Node mit:

$$R_{NTC} = \left(\frac{5V}{U_{in}} - 1\right) * 10000\Omega$$

Aus $R_{NTC}$ kann die zu messende Temperatur T mit Hilfe der Formel aus der Vorlesung in einem weiteren Formula-Node berechnet werden:

$$T = \left(A_1 + B_1 * ln\left(\frac{R_{NTC}}{R_{ref}}\right) + C_1 * ln^2\left(\frac{R_{NTC}}{R_{ref}}\right) + D_1 * ln^3\left(\frac{R_{NTC}}{R_{ref}}\right)\right)^{-1}$$

Mit gegebenen Werten $R_{\text{ref}} = 10\ k\Omega$, $A_1 = 3{,}354016 * 10^{-3}$, $B_1 = 2{,}56985 * 10^{-4}$, $C_1 = 2{,}62019 * 10^{-12}$, $D_1 = 6{,}38309 * 10^{-15}$.

Die Temperatur liegt nun in Kelvin vor und wird zur einfacheren Handhabung in Celsius umgerechnet:

$$T_{\circ\text{C}} = T_{\circ\text{K}} - 273{,}15$$

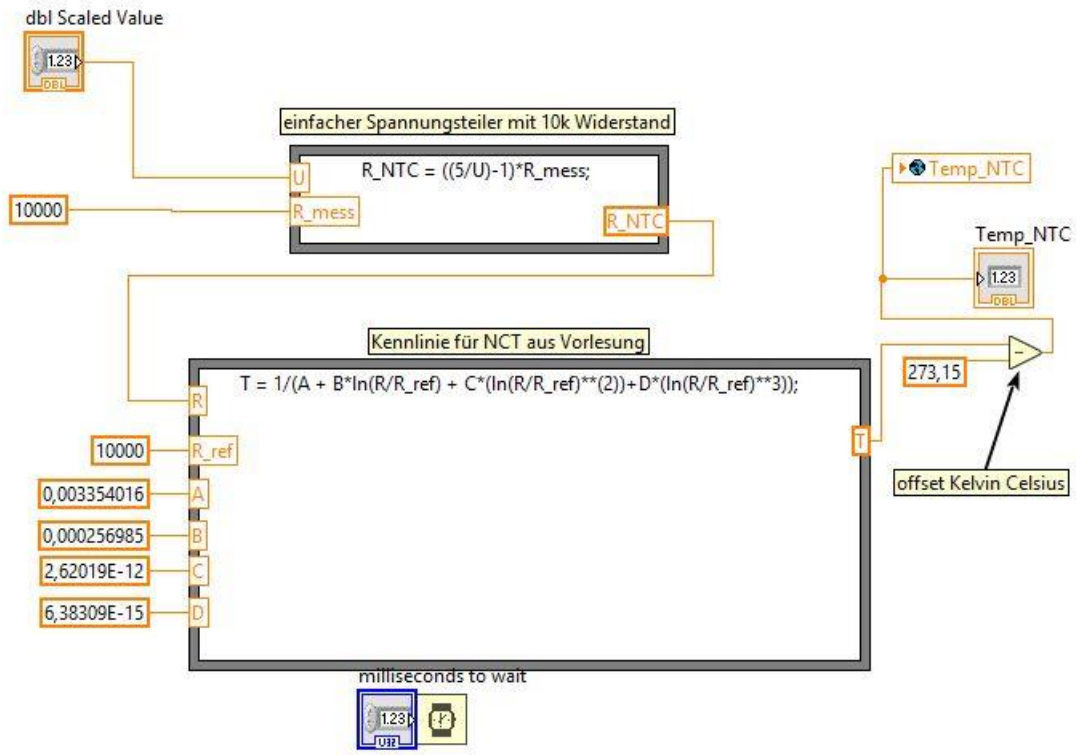



Abbildung 16: Bereechnung der Temperatur aus Eingangsspannung

### 1.6.5 BME/ NFC (I²C)

Zusätzlich zu der Erfassung der Temperatur mittels des Thermoelement Typ K mit dem MAX31855 Breakout-Board und dem PT100 existiert noch eine weitere Temperaturmessung über einen weiteren Temperatursensor. Dabei sitzt dieser direkt auf einem weiteren Breakout-Board, dem BME280. Neben der Temperatur können damit ebenfalls der Druck und die Feuchtigkeit der Luft gemessen werden. Dadurch können die Umgebungsbedingungen der Kaffeemaschine überwacht werden.

Der Sensor selbst gibt seine Messdaten über SPI-Bus oder I²C-Bus aus, wobei bei diesem Projekt der Sensor über I²C-Bus angeschlossen ist. Dieser wird ausgewertet für die unterschiedlichen Sensoren und die Werte in globale Variablen geschrieben. Der Aufbau des Codes ist in Abbildung 17 zu sehen. Dabei ist der Baustein mit der Adresse x77 adressiert und es werden alle 200 ms die Werte erfasst, da diese für das Projekt keine besonders hohe Dringlichkeit besitzt.

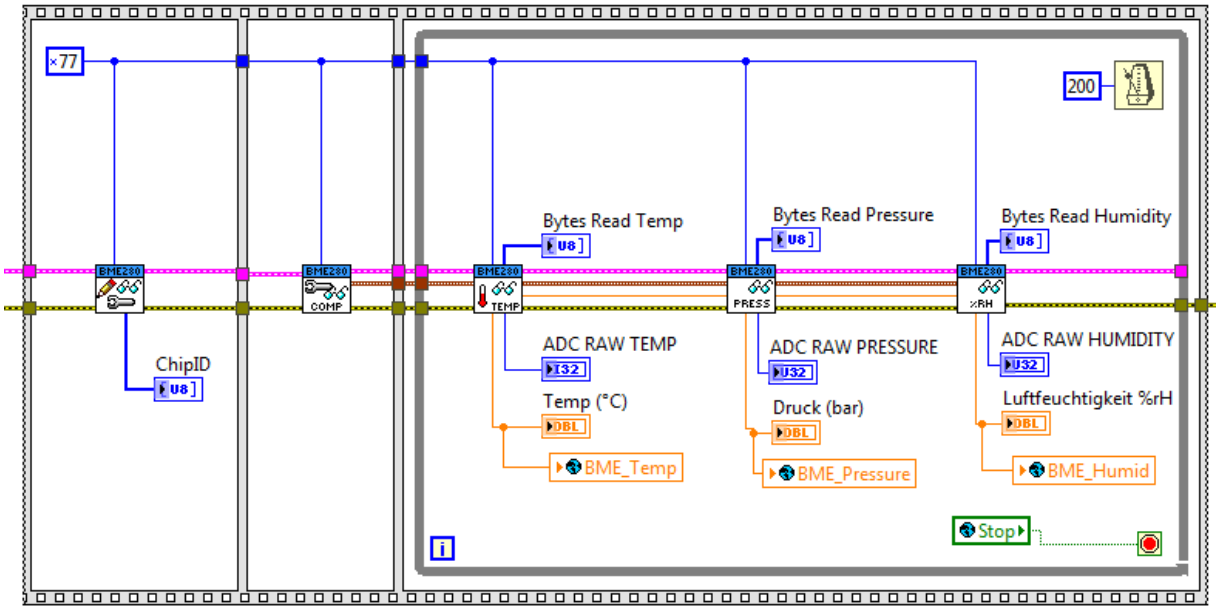



Abbildung 17: Auslesen des BME280 Sensors

Neben dem BME280 sendet auch der NFC-Sensor seine Daten per I²C-Bus an den myRIO. Leider konnten wir die NFC VI nicht aktiv in unser Projekt einbinden, da unser NFC Leser keine zuverlässigen Daten lieferte. Die VI ist zwar in das Projekt eingebunden, allerdings dauerhaft deaktiviert (siehe Abbildung 18). Das Problem mit unserer NFC VI war, dass diese zwar eine aufgelegte NFC Karte erkennt, aber nur etwa in 1/30 der Sekunden und diese 1/30 auch nicht regelmäßig verteilt sind. Das bedeutet mal wurde die Karte mehrere Minuten nacheinander nicht erkannt. Wir fanden das war ein nicht nachvollziehbarer Fehler, den wir selbst mit der Verwendung von Sonden nicht verstanden haben. Wir sind allerdings der Meinung, dass unsere VI an sich richtig implementiert ist und auch insbesondere innerhalb unsere I²C Struktur korrekt eingebunden.

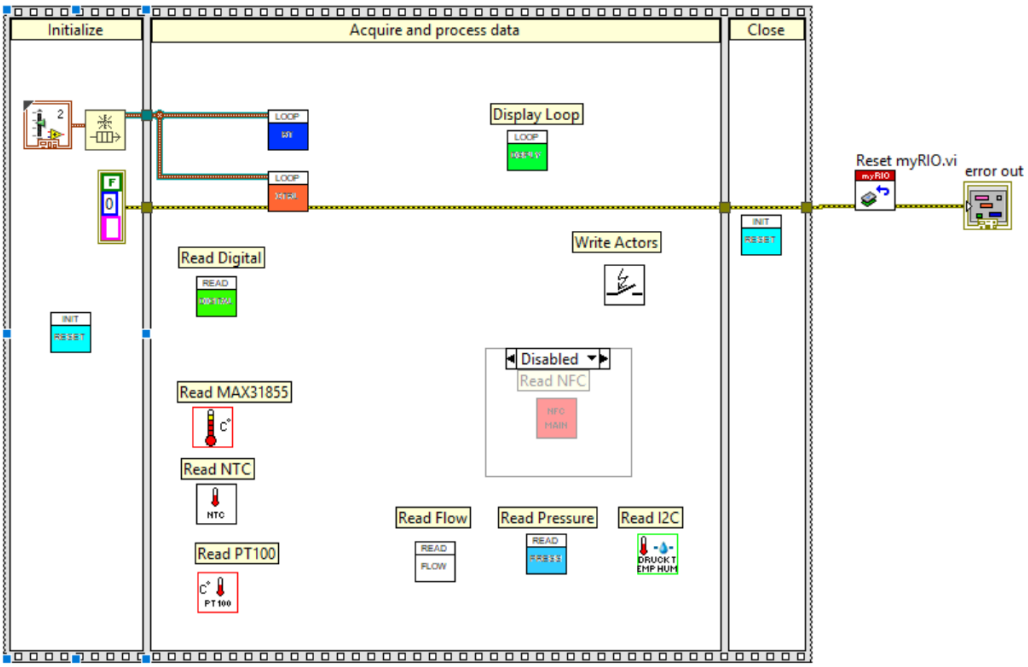



Abbildung 18: Darstellung Main.vi

### 1.6.6 Drucksensor

Auch ist in der Kaffeemaschine ein Drucksensor eingebaut, welcher den Druck des Wassers nach dem Magnetventil von der Heizung erfasst. Über diesen kann dann geschlossen werden, wie stark das Kaffeepulver im Siebträger gepresst ist. Auch kann dieser als Kontrollsensor verwendet werden, dabeispielsweise einen Druckabfall bei Leck im Rohrsystem detektieren kann. Der Sensor gibt dabei proportional zum Druck eine Spannung aus, welcheüber einen analogen Input des myRIOs gemessen werden kann. Der Spannungsbereich liegt dabei zwischen 0,5 V für 0 psi und 4 V 100 psi Druck. Umauf den tatsächlichen Druck zu schließen, wird die gemessene Spannung über die nachstehende Formel in einen Wert für den Druck umgerechnet.

$$p = 28{,}571 \cdot U - 14{,}285$$

Dies ist auch im Code so berücksichtig, was in Abbildung 19 zu sehen ist. Anschließend wird der Wert in eine globale Variable geschrieben.

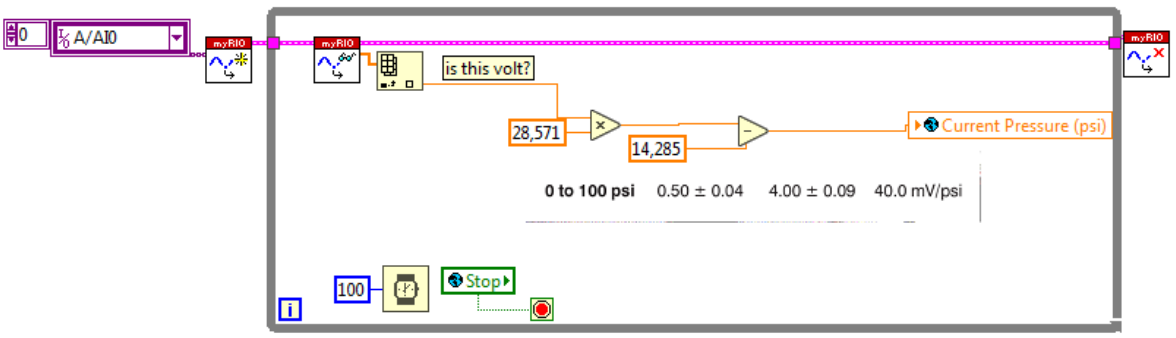



Abbildung 19: Auswertung Drucksensor

### 1.6.7 Durchflussmesser

Neben den genannten Sensoren ist ein Durchflussmesser in der Kaffeemaschine eingebaut, um den Durchfluss und damit die Wassermenge bei einem Brühvorgang zu erfassen. Dabei wird mittels einer Messturbine und einem Hall-Sensor zur Erfassung der Drehzahl der Turbine der Durchfluss gemessen. Es wird ein Encoder-Signal generiert, welches Impulse proportional zum Durchfluss ausgibt. Etwa 20000 Impulsen entsprechen einem Durchfluss eines Liters Wasser. In der Software wird zunächst die Erkennung einer positiven Flanke durchgeführt, um einen Impuls zu detektieren. Anschließend wird mittels eines Shift-Registers die Anzahl der detektierten Impulse gezählt und in der globalen Variablen Flow_Counter gespeichert. Von diesem Wert wird dann ein Offset abgezogen, welches ein Kalibrieren auf Umgebungsdruck ermöglicht. Anschließen wird der Wert durch 20 geteilt, um auf die durchgeflossene Menge in Millilitern zu schließen. Dann wird dieser Wert über einen Faktor an die reale Durchflussmenge angepasst. Dabei wurde mittels mehrerer Versuche und Messungen mit Messbechern probiert, den realen Durchfluss möglichst gut zu erfassen. Außerdem wird auch berücksichtigt, dass bei einer Tasse von ca. 100 ml bei einer Brühzeit von ca. 20 s alle 10ms ein Impuls erfasst werden muss, um eine realitätsnahe Durchflusserfassung zu ermöglichen. Der Wert wird dann in der globalen Variablen Current_Volume (ml) gespeichert. Nach einem Brühvorgang wird die Variable Current_Volume (ml) über die Variable Flow_Counter_Offset zu Null zurückgesetzt. Der Code ist in Abbildung 20 zu sehen.

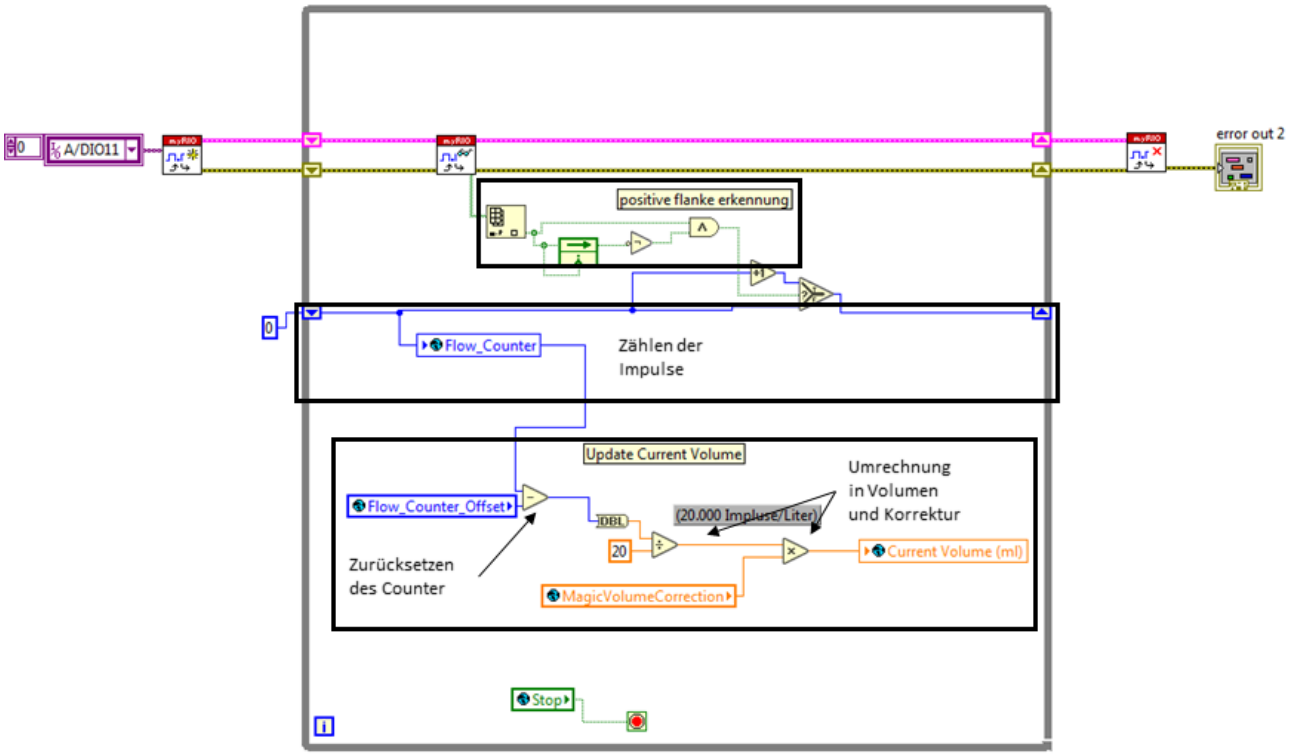



Abbildung 20: Auswertung des Durchflussmessers

# 2 Anschluss der Sensorik

## 2.1 Schaltplan

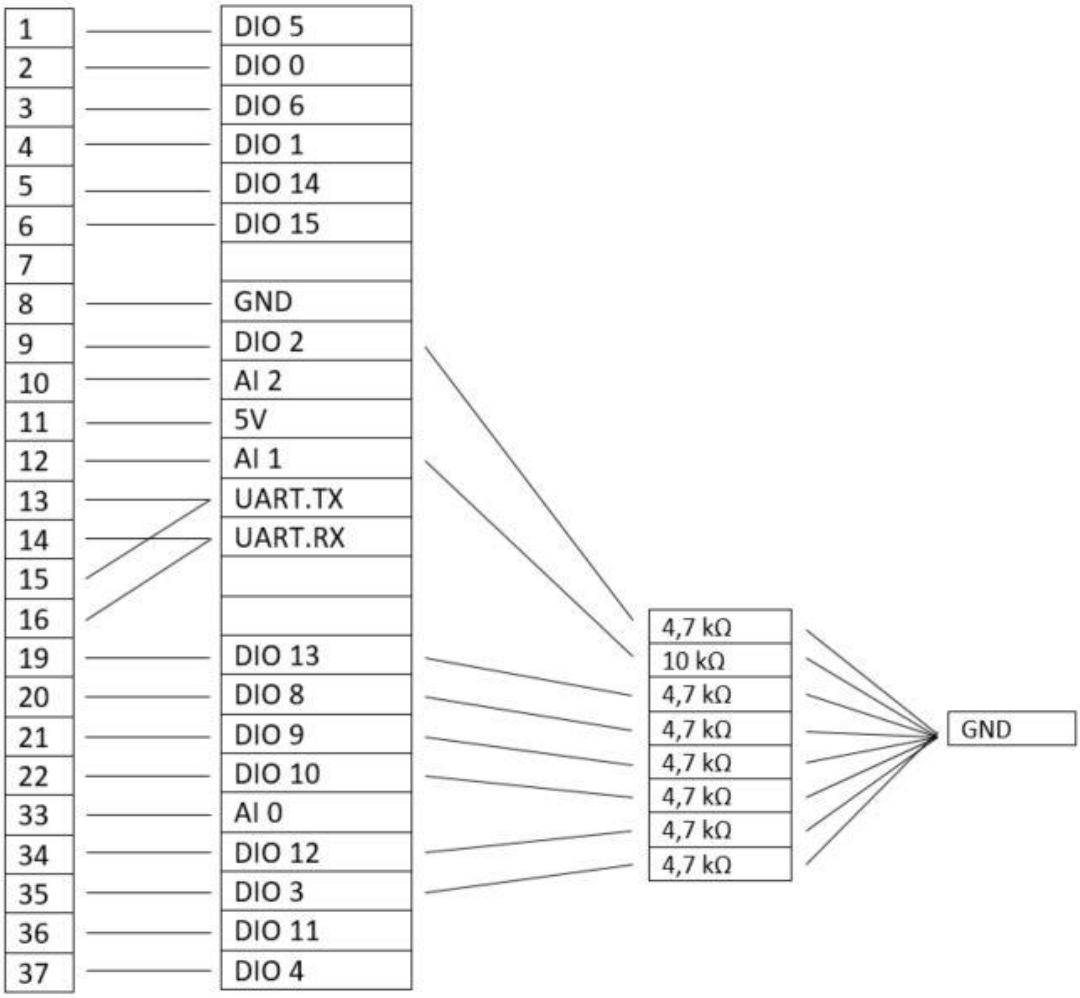



Abbildung 21

Sämtliche Pins des SUB-D-Steckers wurden direkt mit den ihnen zugewiesenen Ports des MXP-Boards verbunden. DIO 2, 3, 8, 9, 10, 12 und 13 wurden parallel über einen 4,7 kΩ Pulldown-Widerstand mit GND verbunden, um undefinierte Zustände zu vermeiden.

AI 1 wurde über einen 10 kΩ Widerstand mit GND verbunden, um einen Spannungsteiler zu realisieren, der es ermöglicht das NTC-Element auszuwerten (siehe 1.6.4).

## 2.2 Steckbrett

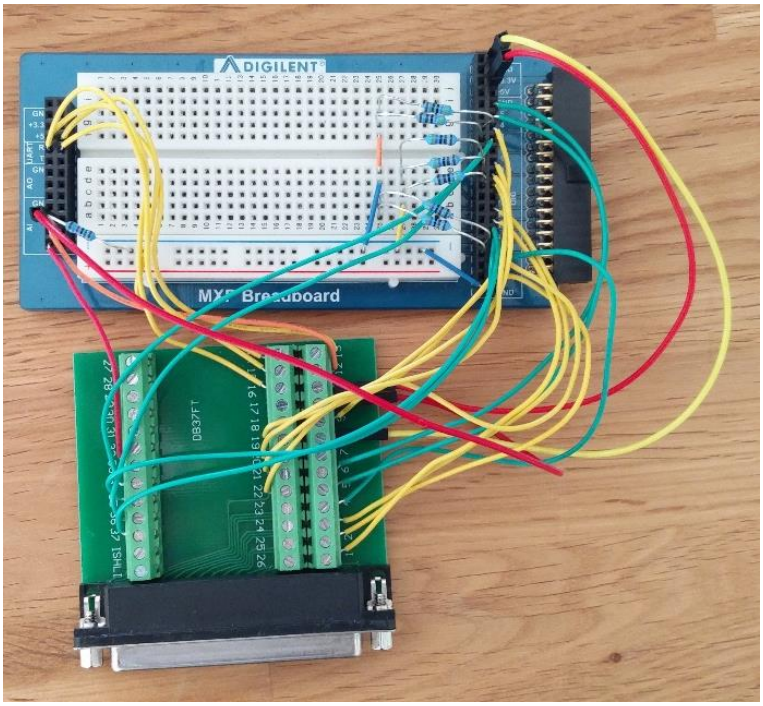



Abbildung 22: MXP-Board mit SUB-D Stecker

Sämtliche SUB-D Pins wurden mittels MXP-Board an Port A des myRIO angeschlossen. Es wurde darauf geachtet, dass die maximale Spannung an den MXP-Ports 5 V beträgt, um den myRIO nicht zu beschädigen. Nur auf Pin 10 könnten theoretisch 10 V anliegen. Da Wasser jedoch keine Temperaturen über 100° C annehmen kann, ist die Spannung an diesem Pin auch auf 5 V beschränkt.

| Pin | Pin-Funktionalität | Komponentenzugehörigkeit | Anschluss MXP |
|---|---|---|---|
| 1 | CLK | Thermoelement (MAX31855) | DIO 5 |
| 2 | CS | | DIO 0 |
| 3 | DO | | DIO 6 |
| 4 | IRQ | NFC / BME (I2C) | DIO 1 |
| 5 | SCK | | DIO 14 |
| 6 | SDA | | DIO 15 |
| 7 | Vcc 5+ | 5 V out | Nicht belegt |
| 8 | GND | GND | GND |
| 9 | Tür oben | Mikroschalter / Switch-4 | DIO 2 |
| 10 | Pt100 | Pt100 | AI 2 |
| 11 | NTC in | NTC | 5V |
| 12 | NTC out | | AI 1 |
| 13 | Rx | Display 1 Quer | UART.TX |
| 14 | Tx | | UART.RX |
| 15 | Rx | Display 2 Portrait | UART.TX |
| 16 | Tx | | UART.RX |
| 19 | Data | DistanceSwitch-5 | DIO 13 |
| 20 | 3,3 V in | Heizer | DIO 8 |
| 21 | 3,3 V in | Ventil | DIO 9 |
| 22 | 3,3 V in | Pumpe | DIO 10 |
| 33 | Data | Druck | AI 0 |
| 34 | Data | LevelSwitch-1 | DIO 12 |
| 35 | Tür hinten | Mikroschalter / Switch-3 | DIO 3 |
| 36 | Data | Flow | DIO 11 |
| 37 | Data | Siebträger / Switch-2 | DIO 4 |

Abbildung 23: Pinbelegung des SUB-D Steckers

# 3 Software-seitige Simulation

Um uns zu ermöglichen unseren Programmcode zu großen Teilen auch ohne den Espressomaschinen Prototyp zu testen nutzen wir das Frontpanel unsere globalen Variablen. Durch Deaktiveren mittels einer Bedingten Deaktivierungsstruktur kann die VI zu Sensordatenerfassung ausgeschaltet werden. Wir können dann über das Frontpanel der globalen Variablen, diese manuel einstellen und verfolgen, zu welchen Reaktionen die Veränderungen führen.

# 4 Datenerfassung und Auswertung eines Brühzyklus

Für das Logging des Brühprozesses haben wir uns für das Loggen auf dem PC entschieden. Durch die Shared Variablen Engine ist eine Übertragung aller wichtigen Zustandsvariablen

möglich. Nachdem die einzelnen Variablen aus den Shared Variablen gelesen wurden, werden sie auf dem PC in globalen Variablen abgelegt. Durch diesen Aufbau, kann man dieselbe globale Variablen VI sowohl auf dem RT als auch auf dem PC zu nutzen. Somit konnten wir den Code minimal halten. Dadurch nutzen wir quasi die Shared Variablen als Tunnel um den Zustand der globalen Variablen zu synchronisieren.

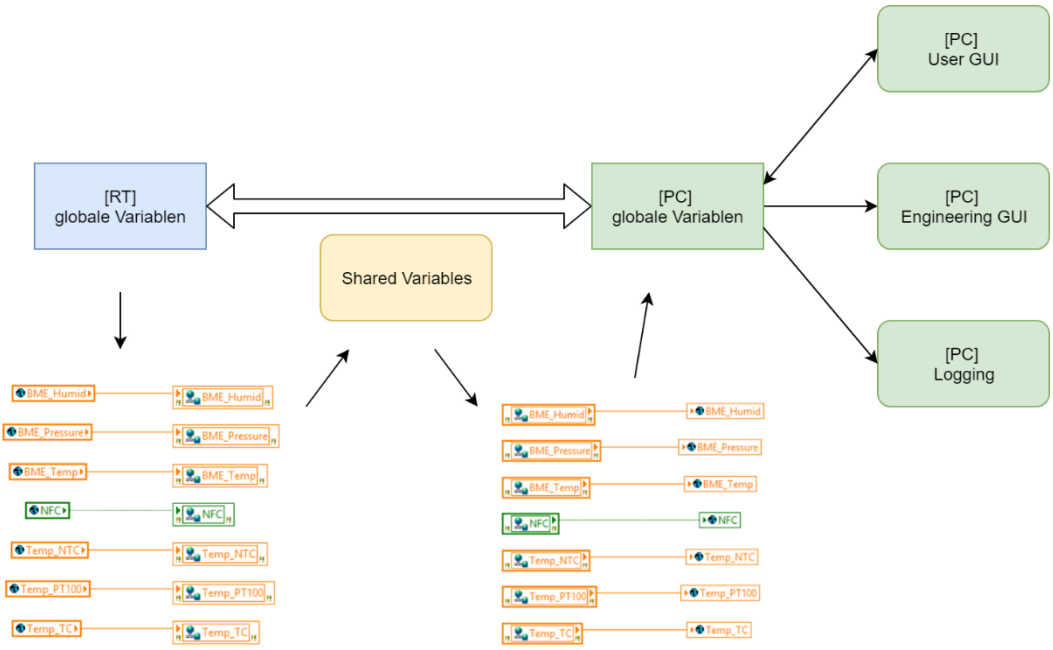



Abbildung 24: Logging-Struktur

Darauf aufbauend ist das Loggen nur noch eine Aufgabe des regelmäßigen Ablegens der globalen Variablen in eine Datei. NI bietet hierfür das TDMS Logdatenformat an. In LabVIEW bietet sich von Haus aus eine einfache API zum Erstellen und suspensiven Auffüllen von Log-Daten an.

Ablauf einer Aufzeichnung:

1. Beim Start der Engineering GUI wird automatisch das Logging-Sub-Modul mitgestartet.
2. Es wird eine neue Logdatei mit dem aktuellen Datum und Uhrzeit im Projektverzeichnis im Ordner "Log" erstellt.
3. In regelmäßigen Abständen (hier 50ms) werden alle relevanten Sensordaten aus den globalen Variablen als Datenpunkt in die TDMS Datei geschrieben.

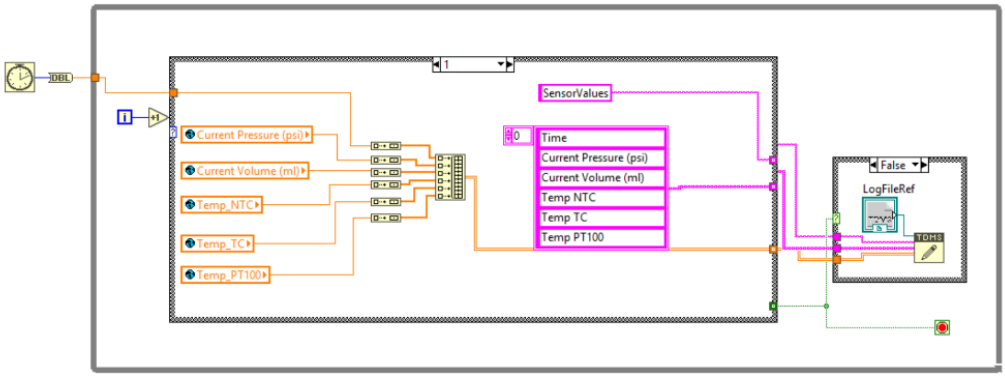



Abbildung 25: Aufbau des Realtime-Loggings

4. Beim Beenden der VI wird die TDMS Datei geschlossen und kann über das Messdatenauswertungswerkzeug NI Diadem oder auch Excel geöffnet werden.

## 4.1 Verwendeter Brühzyklus

- Gewünschte Temperatur: 60° C (NTC)
- Gewünschte Menge: 100 ml
- Eine Tasse
- Kaffee: Fein gemahlener Kaffee, in Siebträger gepresst

## 4.2 Ablauf des Tests

Über die Benutzer-GUI wurde eine Tasse Kaffee angefordert, bei der das Zielvolumen bei 100 ml lag. Dabei wurde zunächst das Wasser auf eine Temperatur von 60° C, am NTC gemessen, erhitzt. Nach dem Vorheizen wurde dann das Magnetventil und die Pumpen gleichzeitig geschaltet, sodass dann das Wasser durch den Siebträger in die Tasse fließt. Dabei wurde sichergestellt, dass die Türen geschlossen waren, ein ausreichender Füllstand vorhanden war, der Siebträger geschlossen und eine Tasse unter der Maschine steht. Kommt es bei einem der Sensoren zu Unregelmäßigkeiten, stoppt die Maschine, wie Tests beweisen. Somit entsteht ein köstlicher Kaffee.

## 4.3 Datenauswertung

Es ist gut zu erkennen, wie zum Startpunkt bei Sekunde 7 die Temperatur im Siedebad beginnt zu steigen. Da der PT100 erst nach dem Siedbad befestigt ist steigt hier die Temperatur zuerst

nicht. Bei Sekunde 13 ist die Ziel-Temperatur erreicht und die Pumpe schaltet und das Ventil öffnet. Hier ist zu sehen, dass der Druck auf ca. 30 psi ansteigt und auf einem Plateau bleibt . Das Volumen beginnt linear zu steigen bis es 100ml erreicht. Ab dort schalten Pumpe und Heizer ab und das Ventil schließt sich. Der Druck sinkt kurz danach wieder ab.

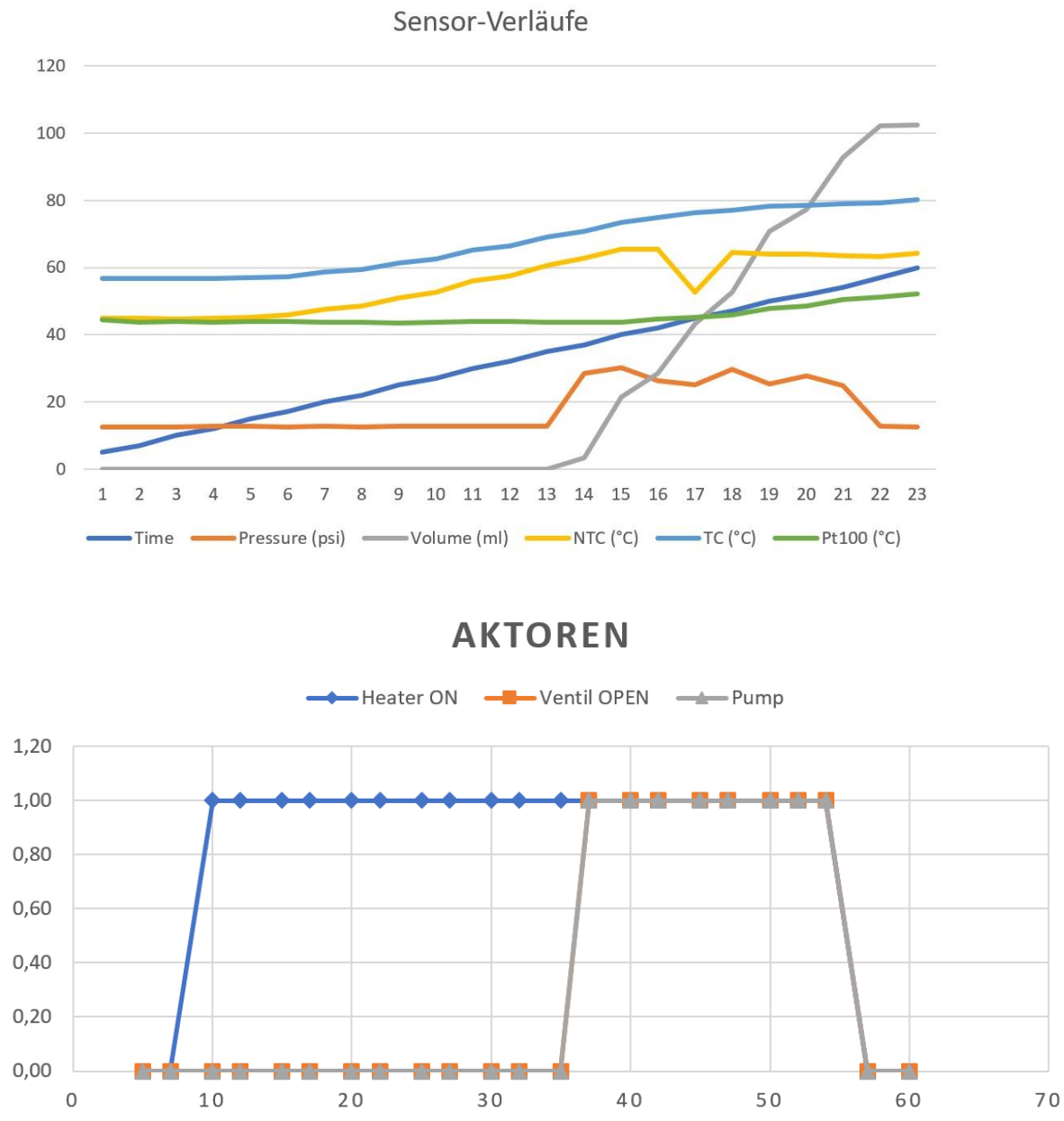



Abbildung 26: Verläufe der Sensoren und Aktoren

## 4.4 Fazit und Empfehlungen

Abschließend ist die Programmierung der Kaffeemaschine erfolgreich und es konnte Kaffee gebrüht werden. Dabei wurden alle sicherheitsrelevanten Sensoren und Schalter überwacht, sodass ein reibungsloser und gefahrloser Betrieb ermöglicht wurde. Allerdings zeigt sich, dass die Temperatur weiter erhöht werden muss, um den Kaffeegenuss weiter zu steigern. Außerdem kann der Funktionsumfang der Kaffeemaschine noch erweitert werden, beispielsweise über ein Touchdisplay, über das die Eingabe erfolgen kann. Daneben können weitere Brühzyklen für andere Kaffeesorten implementiert werden. Somit ergibt sich damit dann eine Kaffeekultur auf einem neuen Niveau.

Darmstadt, den 05. Februar 2019

| | |
|---|---|
| Jan Dreiskemper | Jan Radzey |
| Annemike Unterschütz | Florian Vierneisel |