Fachhochschule Nordwestschweiz
Hochschule für Technik

Bachelorthesis

Entwicklung einer Wärmepumpen-Steuerung

Basierend auf quelloffener Hard- & Software

[redacted]

Zusammenfassung

Die mangelnde Verfügbarkeit von Ersatzteilen älterer Wärmepumpenheizungen führt, insbesondere bei Defekten der Steuerung, nicht selten zum Austausch des gesamten Heizungssystems. Die Instandsetzungskosten bei Kleinanlagen unterschreiten den Neupreis moderner Systeme oft nur gering, was eine Reparatur unattraktiv macht.

Auf Basis offener Hard- und Software, soll deshalb eine herstellerunabhängige und modulare Steuerung für Wärmepumpen entworfen werden. Primärer Auftrag ist die Neuentwicklung einer passenden Steuerung für die Luft-Wasser Wärmepumpe des Typs Hemair SWAP-15 deren Nutzung sich auf weitere Wärmepumpentypen ausweiten lässt. Anhand der Erkenntnisse der bereits durchgeführten Systemanalyse aktuellen Heizungssystems, soll der Prototyp einer gleichwertigen Steuerungslösung entwickelt und aufgebaut werden. Der vorliegende Bericht dokumentiert die Umsetzung des elektrischen und mechanischen Aufbaus, des Programms sowie die Integration der Steuerung in bestehende Anlage.

Das Resultat ist eine Steuerung, welche sich je nach Anlagenanforderung konfigurieren und erweitern lässt. Sie lässt sich mittels breit verfügbaren Erweiterungen unterschiedlich zusammenstellen und modifizieren. Durch die Veröffentlichung der Konstruktionsunterlagen und Quelltexte ist die Steuerung für Dritte transparent und kann unentgeltlich nachgebaut werden. Die Steuerung sieht eine strikte Trennung zwischen Benutzerteil, dem sogenannten Frontend, und Systemteil, auch Grundsteuerung genannt, vor. Beide Teile enthalten einen eigenen Mikrocontroller wodurch sie eigenständig lauffähig sind und über eine Schnittstelle miteinander kommunizieren. Mit dieser Trennung wurde eine noch grössere Modularität erreicht, da die Kommunikationsschnittstelle der Grundsteuerung auch zur Integration in eine Gebäudeautomation oder ein Leitsystem genutzt werden kann.

Im Vergleich zur Originalsteuerung der Hemair SWAP-15 wurde das Betriebsverhalten optimiert. Der Abtauvorgang erfolgt oberhalb von 8°C Aussenlufttemperatur (einstellbar) nun mittels Ventilator und nicht mehr mittels energieintensiver Heissgasabtauung [1]. Daneben wurde die Zyklenzahl durch intelligentes Steuern der Lade- und Enteisungsvorgängen reduziert sowie die Differenzierung von Betriebszuständen verbessert und deren Behandlung optimiert. Die entwickelte Steuerung ist kostengünstig herstellbar und dank integriertem WLAN modern und vernetzbar.

Abstract

Project Title	Open source heat pump control system
Date	March 2018
Internal Number	[redacted]
Project Members	
Advisor	
Client	

The primary task for project 4795-S was to design a heat pump controller as replacement for an existing Hemair SWAP-15 air/water heat pump. Furthermore, it should be designed as versatile controller developed on open source base. First, this report explains the invented modular design and the benefits of separating a control system into an user and system related part. Second, the report describes the mechanical and electrical design of the realized device built on top of open hardware. As a third part it outlines the developed software which runs on the controller. The report concludes with a summary of the benefits and further development possibilities of open heat pump controllers. In the appendix the reader will find technical documents, schematics and source code with detailed documentation.

Inhaltsverzeichnis

1 Einleitung

Zum Beheizen von Wohnräumen werden immer häufiger Wärmepumpen eingesetzt. In der Schweiz sind heute (2017) mehr als 200'000 Elektro-Wärmepumpen in Betrieb wobei diese Zahl, laut Branchenvertretern, auf 400'000 Elektro-Wärmepumpen im Jahre 2020 ansteigen wird [2]. Dies entspricht im Jahre 2017 einem Anteil von 17.9% aller Heizsysteme in Wohngebäuden, während dieser Wert zur Jahrtausendwende gerade mal bei 4.4% lag [3].

Aufgrund der steigenden Bedeutung von Wärmepumpen im Heizungsbereich, lohnt es sich, ein genauerer Blick auf die hiesige Marktsituation zu werfen. Viele Hersteller buhlen um Kundschaft und möchten ihre Systeme, oft auch mit Zusatzfunktionen zum Kaufanreiz, absetzen. Diese Entwicklung hat aus Kundensicht jedoch einen gravierenden Nachteil. Die meisten Systeme sind inkompatibel mit denjenigen anderen Herstellern was im Unterhalt, Wartung und Teiletausch in eine Markenbindung mündet. Daneben ist die Lebensdauer eines Produkts vom Hersteller abhängig, was zum wirtschaftlichen Totalschaden bei einer Reparatur älterer Wärmepumpen führen kann.

Da der Steuerungsaufbau einer Anlage in der Regel vom Hersteller unter Verschluss gehalten wird, ist der Austausch besonders schwierig, da technische Spezifikationen und Auslegungen völlig unbekannt sind. Kann man diese bei Komponenten ausserhalb der Steuerung noch durch Anlagenschemas und Besichtigung vor Ort verifizieren, so steht man bei der zentralen Steuerung vor einer Blackbox, dessen Inhalt Unternehmensgeheimnis ist. In Zeiten in der eine Internetanbindung mit Remote Interface zum guten Ton gehört, hinterlässt dies einen faden Beigeschmack. Zu oft haben solche proprietären IoT-Geräte mit massiven Sicherheitslücken schon von sich reden gemacht.

Dieser Prototyp einer quelloffenen Steuerung soll zeigen, dass es auch anders geht. Der Kunde kann sich selber ein Bild der Schaltung und des implementierten Quellcodes machen. Damit sind nicht nur Ersatzsteuerung selber herstellbar, sondern auch Anpassungen, Sicherheitsupdates und spezifische Erweiterungen möglich. Auf der anderen Seite bleibt für Kunden, welche einfach eine fertige Lösung wollen alles beim Alten. Sie können weiterhin den Hersteller für Wartung, Reparatur oder Austausch beauftragen. Neu könnten sie jedoch auch freie Serviceunternehmen oder fremde Hersteller beauftragen.

Eine Open-Source Steuerung bot sich für den gegebenen Auftrag förmlich an, da die Wärmepumpe deren teildefekte Steuerung ausgetauscht werden soll, von nicht mehr existierendem Hersteller HEMAIR AG stammt. Auch hier war die Anlagendokumentation äusserst dürftig und die Steuerung eine Blackbox. Im vorhergehenden Projekt (vgl. [4]) wurde die Analyse zu genannter Wärmepumpe und deren Steuerung mittels Studium vor Ort und Fachliteratur durchgeführt. Die gewonnenen Erkenntnisse werden in dieser Arbeit genutzt, um eine gleichwertige oder bessere Steuerung zu entwickeln, welche vorhandene Steuerung ablösen soll und dabei möglichst modular und einfach herstellbar ist. Das Ziel dahinter ist nicht nur den Ersatz der Steuerung oben genannter Wärmepumpenheizung, sondern auch eine quelloffene, frei verfügbare Grundlage für Entwickler und Interessierte anzubieten, welche nach den eigenen Wünschen und Bedürfnissen angepasst und weiterentwickelt werden kann.

Der Aufbau dieses Berichts soll den Leser von einer groben Übersicht über das Konzept langsam bis zur detaillierten Beschreibung des Aufbaus und Programms in die Materie eintauchen lassen.

2 Gesamte Steuerung im Überblick

Die entworfene Steuerung setzt sich aus einer Hard- und Softwarekomponente zusammen. Gemeinsam bilden sie eine erschwingliche und modulare Basis zur Kontrolle verschiedener Heizungswärmepumpen. Durch Einsatz von quelloffenen Plattformen sind jegliche Unterlagen verfügbar und der Aufbau transparent, ganz im Gegensatz zu den proprietären Lösungen am Markt. Bei entwickelter Lösung handelt es sich um einen Funktionsprototypen welcher absichtlich ohne selbst entwickelte PCBs aufgebaut ist, sondern sich breit verfügbarer Schaltungsmodulen bedient. Damit ist der Nachbau einfach möglich. Die Vielfalt an Schnittstellen zur Signalverarbeitung, Kommunikation und Ansteuerung ermöglichen den Einsatz in unterschiedlichsten Szenarien. Eine schematische Darstellung der, in aktueller Konfiguration implementierten, Schnittstellen entwickelter Wärmepumpenheizungs-Steuerung ist auf Abb. 2-1 *Schnittstellen und deren Belegung in entwickelter Steuerung* zu sehen.

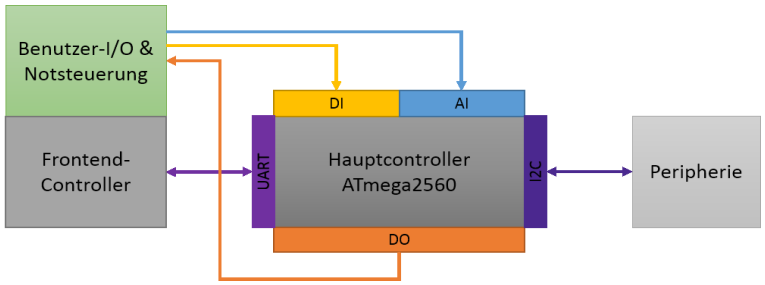

Abb. 2-1 Schnittstellen und deren Belegung in entwickelter Steuerung

Die Benutzerinteraktion mit der Wärmepumpen-Steuerung kann auf mehrere Arten geschehen. Einerseits mit einer direkt verbundenen Notbedienung und anderseits auch über eine lokale, in Zukunft ebenfalls webbasierte, Bedienung die auf einer eigenständigen Plattform läuft, dem sogenannten Frontend-System. Durch die Trennung von Wärmepumpen-Betriebssteuerung und Benutzerinteraktions-Steuerung steigt die Systemstabilität, Modularität, Erweiterbarkeit sowie die Leistung.

2.1 Aufbau

Schaltungstechnisch besteht die entwickelte Steuerung aus einem ATmega2560 8-bit Mikrocontroller auf Basis des Arduino Mega2560, welcher das Herzstück darstellt. Dieser kann auf eine periphere Echtzeituhr sowie einen EEPROM-Baustein zur Sicherung der Einstellungen oder Logs zurückgreifen. Zum Schalten der Lasten stehen 12 galvanisch getrennte Relaisausgänge zur Verfügung welche jeweils einen Öffner- und Schliesser-Kontakt aufweisen. Für Eingangssignale stehen acht digitale 5V Eingänge mit Tiefpassfilter und sechs analoge Eingänge zur Verfügung. Zum Anschluss eines Frontend ist eine Kommunikationsschnittstelle entwickelt worden, über welche Daten und Speisung laufen. Abb. 2-2 *Grundsteuerungsübersicht* zeigt die Grundsteuerung mit ihren einzelnen Komponenten nach Einbau in den vorhandenen Steuerkasten. Die linke Seite mit Klemmen, Schützen und Sicherungen wird wiederverwendet.

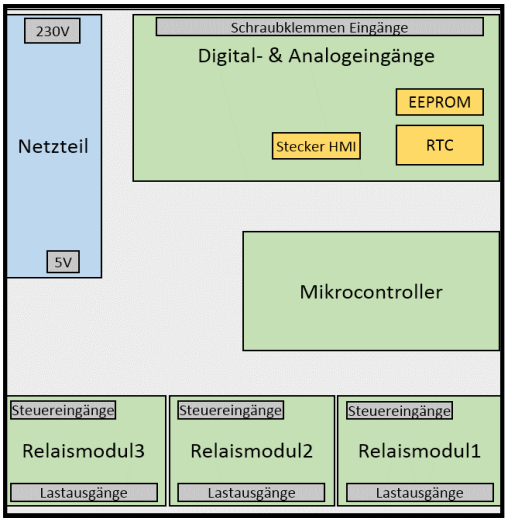

Abb. 2-2 Grundsteuerungsübersicht

Das Frontend besteht aus einem ESP8266 SoC mit 32-bit Mikrocontroller und integriertem WLAN Modul. An dieses ist ein LCD und Drehencoder zur lokalen Bedienung angeschlossen. Auf dem Frontend läuft nebenbei ein Webserver zur ortsunabhängigen Bedienung der Wärmepumpe. Es kann in vorhandene Netzwerke eingebunden werden oder ein eigenes Netz aufspannen. Die Notbedienung ist aus zweckmässigen Gründen auf derselben Montageplatte wie das Frontend fixiert, schaltungstechnisch jedoch absolut unabhängig. Abb. 2-3 *Frontendübersicht (Ansicht von hinten)* zeigt das Frontendsystem am schlussendlichen Einbauort.

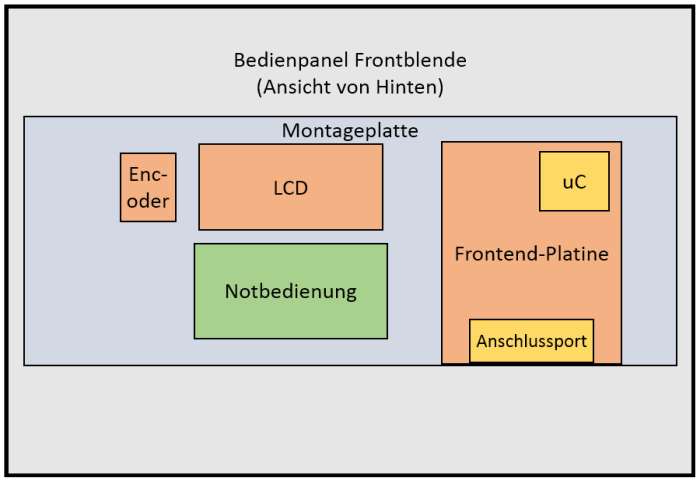

Abb. 2-3 Frontendübersicht (Ansicht von hinten)

2.2 Softwareseitig

Der geschriebene Quellcode ist für die Entwicklungsumgebung Arduino geschrieben, weist jedoch eine hohe Komptabilität mit der Sprache C/C++ durch Einsatz von Standarddatentypen und Vermeidung des Typs Boolean auf. Durch eine Modularisierung ist die Fehlersuche einfacher und eine Codeanpassung mit weniger Aufwand verbunden. Die Programmierung ist auf reduzierten Speicherbedarf und schnelle Verarbeitung ausgelegt. Zustandsautomaten sind mit Enumerator-Variablentypen ausgestattet, was die Identifizierung der Zustände vereinfacht. Die Dokumentation des Programms geschieht gleich im Quelltext mittels Doxygen.

3 Steuerungskonzept

Dieses Kapitel behandelt Kriterien, welche beim Entwickeln und Fertigen neben der eigentlichen Funktion im Vordergrund standen, also das Konzept welches für dieses Projekt, respektive die Steuerung angewendet wurde.

3.1 Open-Source

Ein zentrales Kriterium bei der Entwicklung, war die Umsetzung mit weitgehender Unabhängigkeit von einzelnen Unternehmen und deren proprietären Lösungen. Die Steuerung soll also mittels quelloffener Hard- und Software selber auch quelloffen konzipiert werden. Damit ist der Aufbau einerseits einsehbar, anderseits aber auch selbstständig änderbar und für eigene Zwecke nutzbar ohne Lizenzen und Copyrights zu verletzen. Vor allem im Bereich Software gibt es hierzu eine Anzahl verschiedener Open-Source Lizenzen. Erwähnt seien etwa die MIT-Lizenz oder GPL. Hardware fällt, insbesondere in den USA unter das Patentrecht, weshalb für Open-Hardware oftmals die CCPL oder MIT-Lizenz verwendet werden. Diese Lizenzen ermöglichen den Nachbau und die Weitergabe durch frei verfügbare Baupläne der Hardware.

Der Open-Source Ansatz bietet indessen neben der Herstellerunabhängigkeit auch weitere Vorteile. So ist beispielsweise das System transparent im Aufbau was insbesondere bei sicherheitskritischen Anwendungen von grossem Vorteil ist. Weiter sind häufig keine speziellen Programme und Werkzeuge für die Bedienung und Wartung notwendig, was vor Allem für Anwender mit geringem Budget einen weiteren wesentlichen Pluspunkt darstellt. Zuletzt sei noch der Vorteil der Produktlebensdauer erwähnt, welche nicht mehr von der Produkteunterstützung des Herstellers abhängt, da dank den Quelldateien Funktionsweise und Aufbau bekannt sind.

In vorliegendem Fall unterliegen benötigte Toolchains und Codebibliotheken der GPLv2, die Hardware der CCPL weshalb es sich für dieses Projekt anbietet, für die Veröffentlichung dieselben Lizenzen zu verwenden. Die Veröffentlichung geschieht dabei mittels GIT Repository welches auf Github gehostet ist.

3.2 Modularität

Mit dem Gedanken der universalen Nutzung für verschiedene Wärmepumpensysteme macht es Sinn, das Programm sowie auch den elektrischen und mechanischen Aufbau in einzelne Bausteine aufzuspalten. Die Anpassung an neue Szenarien kann damit ohne Änderung der gesamten Steuerung geschehen, genauso wie die Erweiterung.

3.2.1 Programm

Das erste Kriterium für das Programm ist die Programmiersprache in welcher zu erstellende Quelltext geschrieben wird. Beispielsweise hat ein modulares Programm in einer kaum genutzten Sprache etwa denselben Nutzen wie ein unstrukturierter Code in einer weit verbreiteten Sprache.

Für aktuelles Projekt soll also eine häufig verwendete Programmiersprache Anwendung finden. Dass die Wahl auf Arduino fiel ist einerseits begründet durch die Nähe zu den Sprachen C respektive C++ welche im Embedded-Bereich dem informellen Standard entsprechen. Der wesentliche Punkt weshalb statt C/C++ die Sprache Arduino bevorzugt wurde, ist die Verbreitung in der Maker-Community sowie die Anfängerfreundlichkeit welche es auch Quereinsteigern ermöglicht, relativ schnell ein kompilierbares Programm zu schreiben.

Die Umsetzung des eigentlichen Codes erfolgte gemäss den im Pflichtenheft definierten Module welche das Programm in einzelne Teilfunktionen aufspalten. In folgender Grafik sind die Softwaremodule hierarchisch abgebildet. Nähere Einzelheiten sind im Kapitel 6 *Programmaufbau* beschrieben.unten

3.2.2 Steuerungsaufbau

Die Modularität in hardwareseitigem Aufbau ergibt sich durch den Einsatz von fertigen Schaltungen welche als Einzelteile eingekauft werden können. Meist handelt es sich dabei um Erweiterungsboards für die Arduino Plattform. Für eine Anpassung können somit einzelne Boards entfernt, ersetzt oder hinzugefügt werden. Sie müssen nur noch an den Mikrocontroller angeschlossen werden, welcher ja ebenfalls auf einem Arduino-Board, genauer dem Arduino Mega2560, sitzt.

Erweiterungsboards sind meist Open-Hardware und können über eine Vielzahl Händler bezogen werden, wobei die Liste verfügbarer Boards täglich länger wird. Sie benötigen normalerweise keine externe Beschaltung mehr. Erweiterungsmodule sind jedoch nicht zwingend notwendig, da die Signalpegel TTL-kompatibel sind und Busschnittstellen wie UART, I2C und ISP unterstützt werden, können unzählige verschiedene elektronische Komponenten, Sensoren und ICs verwendet werden. Der verwendete Mikrocontroller besitzt viele Ein- und Ausgänge welche mit aktueller Konfiguration noch unbelegt sind. Damit ist die Flexibilität im schaltungstechnischen Bereich sehr hoch.

3.3 Trennung Benutzer- und Systemumgebung

Oftmals wird bei Embedded-Systems ein grosser Teil der Ressourcen von der Benutzer-Ein- und Ausgabe verbraucht weshalb für die eigentlichen Steueraufgaben nur noch begrenzt Leistung zur Verfügung steht. Da die Rechenleistung, welche für die Benutzeraufgaben benötigt wird, sehr stark schwankt, beeinflusst sie auch die Abarbeitungszeit der Steuerfunktionen. Dies ist insbesondere dann problematisch, wenn zeitkritische Steueraufgaben nicht mehr in einem angemessenen Zeitraum bearbeitet werden oder es durch Bedienfunktionen zum Deadlock kommt.

Ein Weg dies zu verhindern, ist der Einsatz zweier Mikrocontroller, welche mittels Schnittstelle miteinander kommunizieren können. Dieses Design verhindert, dass die Nutzerverarbeitung auf die Ressourcen der Steuerfunktionen und umgekehrt zugreifen können. Damit die Systemsteuerung auch bei Ausfall der Nutzerverarbeitung weiterläuft, werden die vom Nutzer gemachten Parameter auf der Systemsteuerungs-Recheneinheit gespeichert. Für längere Ausfallzeiten oder Abkopplung des Frontend, bietet die Systemsteuereinheit eine zusätzliche Notsteuerung, welche aber nur minimale Optionen bietet.

3.4 Bedienungskonzept

Das finale Bedienkonzept umfasst mehrere Nutzerschnittstellen. Diese laufen bis auf die Notsteuerung auf dem Frontendcontroller. Die Schnittstellen auf dem Frontend sind erstens eine lokale Bedienung, deren Eingaben mittels Drehencoder erfolgen. Für die Menünavigation und im Betrieb zur Anzeige des Zustandes ist lokal ein zweizeiliges LC-Display mit Hintergrundbeleuchtung vorhanden. Als zweite Benutzerschnittstelle ist die Bedienung per Browser vorgesehen, welche eine bessere Übersicht der Systemzustände und Optionen erlaubt. Als zukünftige dritte Schnittstelle soll die Bedienung über einen Bot des Messenger Telegram ebenfalls möglich sein.

Damit die Systemsteuerung nicht komplett von einem Frontendcontroller abhängig ist, kann die Bedienung auch über eine Notsteuerung welche sich unterhalb des LC-Displays befindet, erfolgen. Dabei können jedoch nur die Betriebsmodi angewählt, die Heizkurve eingestellt und die Parallelverschiebung geändert werden. Eine Rückmeldung erfolgt allein durch LEDs.

4 Systemaufbau Grundsteuerung

Das Kapitel Systemaufbau umfasst den elektrischen und mechanischen Aufbau der entwickelten Grundsteuerung und gliedert sich in mehrere Unterkapitel welche die verschiedenen Aspekte und Überlegungen des Aufbaus dokumentieren. Der Aufbau des Frondend mit der Benutzerbedienung ist im Kapitel 5 *Systemaufbau Frontend* beschrieben.

4.1 Teilsysteme

Als Teilsysteme werden in dieser Dokumentation einzelne elektronischen Schaltungen bezeichnet, welche zur Steuerung hinzugefügt werden können, jedoch nicht zwingend erforderlich sind. Es handelt sich dabei, wie bereits im Kapitel 3.2.2 *Steuerungsaufbau* erwähnt, grösstenteils um sogenannte Erweiterungsmodule für die Arduino-Plattform. Die Teilsysteme und deren Funktion als Glied der Steuerung werden nachfolgend beschrieben.

4.1.1 Speisung

Die Steuerung kann dank eigenem Netzteil direkt an 230V~ angeschlossen werden. Das Teilsystem Speisung übernimmt die Transformation und Gleichrichtung. Dazu wird ein Schaltnetzteil mit einer Ausgangsspannung von 5V DC bei maximal 3A Dauerstrom verwendet, welche die gesamte Steuerung versorgt. Die zentrale Speisung hat den Nachteil, dass alle 5V-Stromkreise miteinander verbunden sind und es somit keine galvanische Trennung zwischen den 5V Lasten gibt. Eine gegenseitige Beeinflussung muss deshalb mittels geeigneter Massnahmen verhindert werden, welche im Kapitel 4.4 *Störfestigkeit* behandelt werden. Der grosse Vorteil einer einzelnen Speisung liegt in der Energieeffizienz, welche durch mehrere Netzteile verschlechtert würde.

Das verbaute Netzteil befindet sich auf der Trägerplatte der Grundsteuerung in maximalem Abstand zu den weiteren Elementen. Eine zusätzliche Siebung sowie ein LC-Filter sind auf der Sekundärseite des Netzteils verbaut. Erst danach erfolgt die sternförmige Verteilung von 5V und Masse auf die einzelnen Stromkreise. Die komplette Trägerplatte, welche aus Aluminium besteht, ist schutzgeerdet und kapazitiv mit der Sekundärmasse verbunden. Netzteil und Siebung sind auf Abb. 4-1 *Schaltnetzteil mit Siebung und zentraler Verteilung rechts* zu sehen.

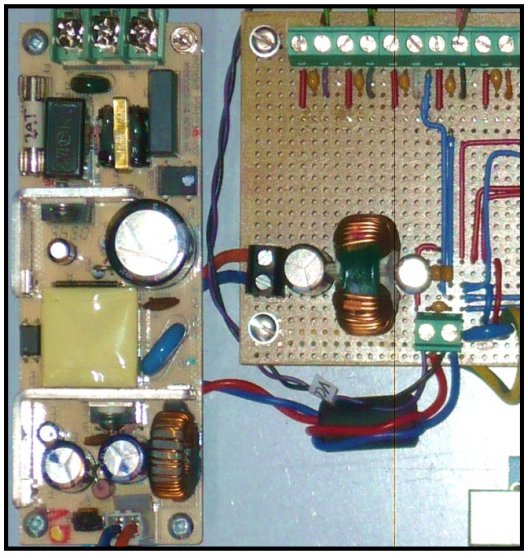

Abb. 4-1 Schaltnetzteil mit Siebung und zentraler Verteilung rechts

4.1.2 Mikrocontroller

Der verwendete Mikrocontroller ist ein ATmega 2560 der Firma Atmel (heute Microchip). Dieser bietet genügend Ein- und Ausgänge sowie diverse im Embedded-Bereich verwendete Schnittstellen [5]. Die nötige Peripherie und Stromversorgung, sowie den zur Programmierung nötigen USB-Serial Wandler, werden dank Einsatz des Mikrocontroller-Boards Arduino Mega2560, auf welchem genannter Mikrocontroller verbaut ist, nicht mehr separat benötigt. Die Anschlüsse für Peripherie sind als Pinheader ausgeführt, der Anschluss am PC erfolgt über USB, was die Nutzung des Mikrocontrollers extrem vereinfacht. Da alle Teilsysteme der Grundsteuerung mit 5V TTL-Pegel arbeiten, müssen keine Pegeländerungen an den Ein- und Ausgängen des ATmega mehr vorgenommen werden.

In folgender Tabelle sind die Spezifikationen des verwendeten Mikrocontrollers zur besseren Übersicht zusammengefasst. Zu beachten ist, dass nicht alle der verfügbaren Pins auf dem Arduino Mega2560 nach aussen geführt sind. Beispielsweise sind beim Arduino nur 53 von 86 möglichen GPIOs nach aussen geführt.

Eigenschaft	**Wert**
Plattform	AVR 8-bit RISC Mikrocontroller
Typ	ATmega2560
ISP Flash Memory	256KB
SRAM	8KB
EEPROM	4KB
CPU-Takt	16 MHz
Kommunikations-Schnittstellen	4x USART, 5x SPI, I2C, JTAG
Digitale GPIO	86x davon 16x PWM
Analog Inputs	16x 10-bit
Integrierte Timer	2x 8-bit, 4x 16-bit
Temperaturbereich	-40 bis 85°C
Versorgungsspannung	1.8-5.5V

Tabelle 4-A Spezifikationen ATmega2560

4.1.3 Relaismodule

Zu Schalten externer Lasten sind insgesamt 12 Relaisausgänge verfügbar, wobei jeweils ein Relaismodul vier einzelne Kanäle enthält. Die Schaltleistung der Kontakte beträgt bei 230V~ pro Kanal 10A was 2300W entspricht. Ein Relaisausgang enthält jeweils ein Wechselkontakt, dessen Anschlüsse mittels Schraubklemmen nach Aussen geführt sind. Die Versorgung der Relaismodule selber erfolgt mit 5V DC und wird mittels LED angezeigt.

Die Ansteuerung der Relaisspulen kann nicht direkt durch Mikrocontroller-Ausgänge geschehen, da deren maximaler Ausgangsstrom zu gering ist und sie dadurch zerstört würden. Aus diesem Grund werden die Spulen auf den Relaismodulen mittels Stromtreiber geschalten, welche über Optokoppler die Schaltsignale vom Mikrocontroller erhalten. Diese Entkopplung schützt weiter den Mikrocontroller vor Spannungsspitzen, welche beim Ausschalten einer induktiven Last, sprich der Relaisspulen, entstehen. Steuersignale können als Stromquelle oder Stromsenke erfolgen wobei die Umschaltung mittels Jumper sehr simpel ist. Der Mikrocontrollerausgang kann also zum Schalten des Relais entweder Active-low oder –high sein. Der Schaltzustand pro Kanal wird mittels LED angezeigt. Da alle Anschlüsse der Module als Schraubklemmen ausgeführt sind und die Signalrichtung änderbar ist, weist dieses Teilsystem eine maximale Flexibilität für unterschiedlichste Anforderungen auf.

4.1.4 Echtzeituhr

Damit zeitgesteuerte Programme, Ausnutzung des Niedertarifs und Anderes möglich sind, wurde ein RTC-Modul des Typs Maxim DS3231 verbaut, welches eine genauere Uhr als die im Mikrocontroller integrierte besitzt. Das Modul wird im Normalbetrieb mit 5V DC betrieben, kann jedoch auch eine begrenzte Zeit lang ohne primäre Stromversorgung weiterlaufen. Dies ist möglich durch Einsatz eines Superkondensators mit 17 Farad, welche normalerweise im professionellen IT-Bereich zur Sicherung des RAM-Inhalts eingesetzt werden [6].

Die RTC besitzt einen separaten Eingang für eine Backup-Stromversorgung auf welche bei Ausfall der Primärversorgung automatisch umgeschaltet wird. Bei aktiver Primärversorgung wird der Kondensator ständig geladen wofür, im Gegensatz zu einem Akkumulator, keine spezielle Ladeschaltung erforderlich ist. Dass statt des Kondensators keine Batterie, wie man sie häufig unter anderem in Computern findet, eingesetzt wurde, liegt vor allem daran, dass eine Batterie von Zeit zu Zeit ausgetauscht werden muss weshalb man deren Ladestand periodisch kontrollieren muss. Dies erfordert ein ständiges Warten und anschliessendes Neueinstellen der Uhr. Neben der Gefahr des Auslaufens einer Batterie ist dies keineswegs wartungsfreundlich weshalb auf den Einsatz einer Batterie verzichtet wurde.

Verbunden ist die Echtzeituhr über den I2C-Bus mit dem Mikrocontroller und kann von diesem abgefragt werden. Zusätzlich besteht die Möglichkeit, auf dem RTC-Modul einen Alarm zu stellen, welcher ein Interrupt auslöst. Dadurch entfällt das Polling durch den Mikrocontroller wodurch dieser entlastet werden kann. Die RTC ist in nebenstehender Abb. 4-2 *Echtzeituhr* zu sehen.

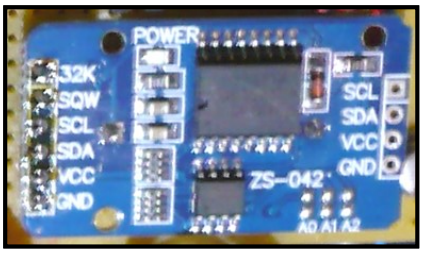

Abb. 4-2 Echtzeituhr

4.1.5 EEPROM

Da der nichtflüchtige Speicherplatz im Microcontroller begrenzt ist, wurde ein externer Speicherbaustein verbaut welcher über den I2C-Bus mit dem Microcontroller verbunden ist. Dieser EEPROM ist gedacht, um Benutzereinstellungen abzuspeichern welche nach einem Stromausfall wiederherstellbar sind oder Messreihen festzuhalten, aus welchen Verlaufsdiagramme erstellt werden können. Grundsätzlich handelt es sich diesem Teilsystem um eine Option, welche primär für Erweiterungen des erstellten Sourcecodes gedacht ist. Somit ist dieser zusätzliche Speicher für den aktuellen Ausbaustand des Programms nicht nötig, ermöglicht jedoch die flexible Erweiterung der Software ohne die Steuerung umzubauen.

Bei eingesetztem EEPROM handelt es sich um den Typ AT24C256 des Herstellers Atmel mit 256 Kilobyte nichtflüchtigem Speicher. Dieser kann als Erweiterungsboard erworben werden auf welchem sich zusätzliche Pullup-Widerstände für die Datenleitungen, Jumper zur Adresskonfiguration und eine LED zur Kontrolle der Spannungsversorgung befinden [7].

4.2 Verdrahtung

Um eine Verbindung zwischen den Teilsystemen auf der Steuerungs-Grundplatte herzustellen, ist eine Verdrahtung nötig. Aufgrund des modularen Konzepts des Aufbaus welches ohne speziell angefertigtes PCB auskommt, müssen die elektrischen Verbindungen selber angefertigt werden, wozu aufgrund der räumlichen Trennung der Module Kabel geeignet sind. Da die Verschaltung mittels Kabel eine relativ aufwendige Arbeit ist, wurden die Anschlüsse an den Komponenten steck- oder schraubbar gestaltet und nicht unlösbar, etwa durch löten, verbunden. Damit bleibt die Steuerung weiterhin modular und die Teilsysteme einfach austauschbar, ähnlich der Komponenten eines Schaltschranks.

Mit Augenmerk auf die im Kapitel 4.4 *Störfestigkeit* behandelten Massnahmen zur Verringerung von Störungen, sind Leitungen zur Übertragung analoger Signale und Busverbindungen geschirmt ausgeführt. Die Anschlüsse sind wo immer möglich als Schraubklemmen ausgeführt, welche eine solide und vibrationsresistente Befestigung bieten. Einzig am Arduino Mega2560 musste darauf verzichtet und auf Steckverbinder zurückgegriffen werden, da die Anschlüsse am Arduino als Pinheader ausgeführt sind. Durch die meist grosse Zahl an Signalleitungen pro Kabelstrang, welcher jeweils einen Stecker umfasst, halten sie trotzdem gut und lassen sich nur mit erhöhter Kraftaufwendung herausziehen. In Abb. 4-3 *Pinheader-Anschlüsse am Arduino* lassen sich die Kabelstränge gut erkennen.

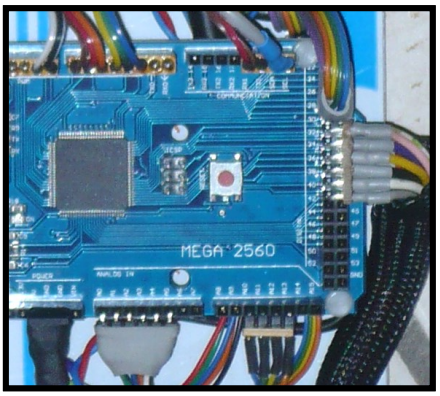

Abb. 4-3 Pinheader-Anschlüsse am Arduino

4.3 Ein- und Ausgänge

Die entwickelte Wärmepumpen-Steuerung weist diverse Ein- und Ausgänge zum Anschluss von Sensorik und Lasten sowie eine Schnittstelle zum Anschluss des Frontend auf. Im Folgenden werden die einzelnen Anschlüsse und deren Funktion detailliert beschrieben. Alle extern erreichbaren Abgänge der Hauptsteuerung sind als Schraubklemmen wie in Abb. 4-4 *Eingänge der Steuerung als Schraubklemmen* zu sehen, ausgeführt. Damit kann die Steuerung sehr einfach in die Verdrahtung eines Schaltschranks integriert werden. Jedem Eingangskanal steht eine eigene Schraubklemme mit Masse zur Verfügung.

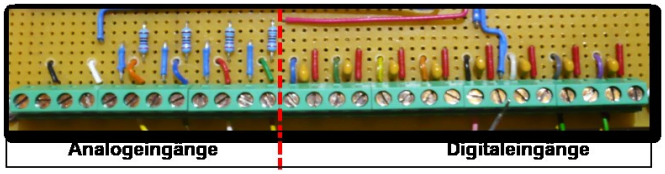

Abb. 4-4 Eingänge der Steuerung als Schraubklemmen

4.3.1 Digitaleingänge

Die digitalen Eingänge der Steuerung ermöglichen Rückmeldungen über den Systemzustand in Sinne einer binären Logik, beispielsweise «Motorschutzschalter in Ordnung» und «Motorschutzschalter ausgelöst». Für die Verarbeitung muss erst eine Umgebung geschaffen werden, die es ermöglicht die beiden Zustände sicher zu erkennen und voneinander zu unterscheiden. Dazu wurden als erstes die zu erwartenden Signalpegel und der Default-Zustand des Signals definiert. Aufgrund der störbehafteten Umgebung und dessen möglicher Einfluss auf die Signalzustände wurde entschieden, dass bei offenem Eingang, sprich der logische Zustand «Null» respektive «Aus», der Signalpegel bei 5VDC liegt. Dadurch muss der Signalpegel für die Verarbeitung durch den Mikrocontroller nicht angepasst werden da dessen GPIOs 5V tolerant sind. Zusätzlich können den Digitaleingängen des Mikrocontrollers intern Pullup-Widerstände zugeschalten werden wodurch die Eingangspegel ohne

externe Beschaltung im Grundzustand auf 5V gehoben werden. Im Falle eines geschlossenen Eingangsstromkreises wird der Mikrocontrollereingang mit Masse verbunden wodurch das Signal auf 0V abfällt was als Zustandsänderung erkannt wird.

Während in der Theorie ein Schalter entweder geöffnet oder geschlossen ist, zeigt sich bei mechanischen Schaltkontakten in der Praxis ein etwas anderes Bild. Dort tritt im Umschaltmoment ein Kontaktprellen auf, das durch den mechanischen Stoss auf die Kontaktzungen, welche anschliessend in Schwingung geraten, erzeugt wird. Dadurch wechselt im Augenblick des Schaltens der Kontakt mehrmals zwischen seinen Schaltzuständen hin und her, bis er schlussendlich seinen endgültigen Zustand einnimmt. Dieses Verhalten ist bei der Auswertung durch den Mikrocontroller störend da jeder Kontaktschluss beim Prellen als einzelne Zustandsänderungen interpretiert wird, was zu unerwünschtem Verhalten der Steuerung führt. Massnahmen zur Unterdrückung, in der Fachsprache Entprellung oder Debouncing genannt, können schaltungs- oder programmtechnisch aber auch als Kombination ergriffen werden [8].

Das schaltungstechnische Entprellen hat den Vorteil, dass es einfach zu realisieren ist und darüber hinaus gleichzeitig eingekoppelte Störsignale höherer Frequenzen dämpft. Es handelt sich also um einen Tiefpassfilter welcher meist als simples RC-Glied realisiert ist [9]. Programmtechnisch kann Debouncing durch einstweiliges Zurückhalten einer erkannten Signaländerung implementiert werden. Bei erkannter Änderung wird dazu der alte Zustand zwischengespeichert und der entsprechende Mikrocontroller-Eingang nach einer definierten Stabilisierungszeit erneut ausgelesen. Falls nun der Messwert nicht dem zwischengespeicherten Wert entspricht, kann der neue Zustand im Programm bekannt gegeben werden.

Aufgrund der langsamen Signalwechseln und der störungsbehafteten Umgebung in welcher die Steuerung eingesetzt wird, bot sich das Debouncing mittels passivem Tiefpassfilter an. Dazu wurde nach jeder Schraubklemme der Digitaleingänge ein RC-Glied wie in Abb. 4-5 ersichtlich verbaut und das gefilterte Signal anschliessend dem Mikrocontroller zugeführt. Durch den relativ grossen Kondensator ist das erfassen schneller Signale nicht möglich und in dieser Anwendung auch nicht erwünscht, handelt es sich doch bei ankommenden Signalen ausschliesslich um Zustandsrückmeldungen von Aktoren und Sicherheitsorganen. Die Anschlussbelegung der Digitaleingängen bei entwickelter Steuerung sind in untenstehender Tabelle 4-B aufgelistet. Wie ersichtlich, gibt es bei bestehender Wärmepumpe nur eine kleine Anzahl Komponenten, welche eine Rückmeldung über ihren Betriebszustand liefern. Im Gegensatz zur originalen Steuerung, welche nur die Zustände der Sicherheitsorgane auswertete, werden in der entworfenen Steuerung alle verfügbaren Rückmeldesignale verwendet.

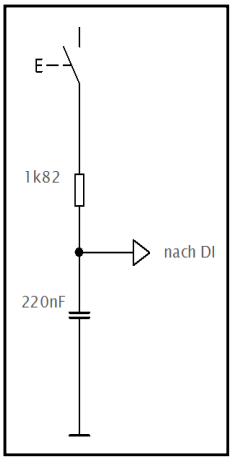

Abb. 4-5 Tiefpassfilter der Digitaleingänge

DI	µC Pin	Belegung der DI-Klemmen	Schaltertyp
1	D22	Niederdruck-Sensor Kältemittelkreis	Öffner
2	D23	Hochdruck-Sensor Kältemittelkreis	Öffner
3	D24	Motorschutzschalter Kompressor	Öffner
4	D25	Signalkontakt Anlassschütz Kompressor	Schliesser
5	D26	Signalkontakt Betriebsschütz Kompressor	Schliesser
6	D27	EW Tarifsperre	Schliesser
7	D28	Reserve-Eingang	-
8	D29	Reserve-Eingang	-

Tabelle 4-B Belegung der Digitaleingänge

4.3.2 Analogeingänge

Um Signale mit mehr als zwei Zustandswerten zu erfassen, bedarf es neben den Digitalen auch analoge Eingänge, welche in der Lage sind, das Signal als Wert einer Messbereichsmenge auszugeben. Dazu werden Analog-Digital-Wandler genutzt, die meist in geringer Zahl bereits in Mikrocontroller integriert sind. Beim genutzten Atmega2560 sind es 16 Kanäle was für einen relativ kleinen Mikrocontroller eine hohe Zahl ist. Dies war nicht zuletzt einer der Gründe, weshalb die Entscheidung auf diesen Chip viel. Die Auflösung je A/D-Wandler beträgt 10-bit, der Wertebereich also 2^{10} was 1'024 Abstufungen entspricht.

Die A/D-Wandler werden zum Auswerten der Temperatursensorsignale genutzt. Bei den Temperatursensoren handelt es sich um die Typen KTY81/210 von NXP welche in der Heiz- und Klimatechnik häufig Anwendung finden. Diese Sensoren sind temperaturabhängige Widerstände deren Widerstand sich mit steigender Temperatur erhöht. Man nennt sie deshalb auch Kaltleiter oder PTC was die Kurzform von *Positive Temperature Coefficient Thermistor* ist. Der KTY81 weist eine annähernd lineare Charakteristik im Widerstands-Temperatur Diagramm auf, was in Abb. 4-6 *KTY81/210 Widerstands-Kennlinie* ersichtlich ist. Der maximale Strom durch den PTC soll 1mA nicht überschreiten da ansonsten das Messresultat durch die Eigenerwärmung des Sensors verfälscht würde [10, 11].

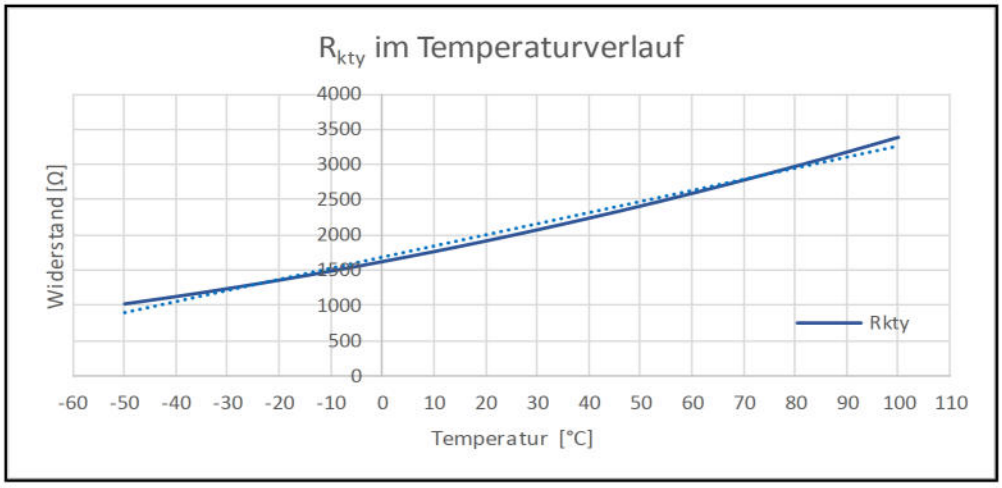

Abb. 4-6 KTY81/210 Widerstands-Kennlinie

Unter Berücksichtigung der Herstellervorgaben wurde eine Eingangsbeschaltung entworfen, in welcher der KTY81 und ein Festwiderstand zusammen einen Spannungsteiler bilden in deren Mitte das Signal abgegriffen und dem A/D-Wandler zugeführt wird. Der Spannungsteiler selber ist wie in Abb. 4-7 *Temperatur-Messschaltung* ersichtlich aufgebaut. Der Spannungsabfall am Temperatursensor ist also die gemessene Grösse, welche in einen numerischen Wert digitalisiert welcher einem Temperaturwert zugewiesen werden kann. Durch den Festwiderstand von 2,7kΩ kann der temperaturabhängige Spannungsabfall am PTC fast vollständig linearisiert werden. Grund dafür ist der Strom durch den Sensor, welcher bei steigender Temperatur abnimmt. Als Folge davon steigt dieser jedoch unterhalb von etwa 22°C auf über 1mA, genauer bis zu 1,13mA bei -50°C an, was jedoch aufgrund der Umgebungstemperatur-Abhängigkeit der Eigenerwärmung, in diesem Temperaturbereich keinen Einfluss auf die Messgenauigkeit hat [11].

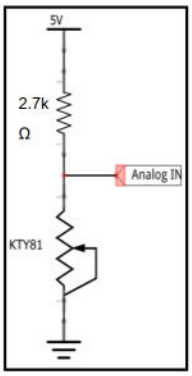

Abb. 4-7 Temperatur-Messschaltung

Die Berechnung des temperaturspezifischen Spannungsabfall am KTY81 U_{kty} im Spannungsteiler bei bekannten Festwiderstand R_{fest} mit 2,7kΩ und am Teiler anliegende Spannung U_0 von 5V erfolgt durch:

$$U_{kty} = \frac{U_0}{(R_{fest} + R_{kty})} * R_{kty}$$

Aufgezeichnet führt dies zur roten, in Abb. 4-8 *temperaturabhängiger Spannungsabfall am KTY81/210* ersichtlichen Kennlinie. In einem weiteren Schritt kann aus der Menge der Kennlinienwerte die Steigung *m* und der y-Achsenabschnitt *n* der lineare Regression wie folgt berechnet werden:

$$Steigung: m = \frac{\sum_{i=1}^{k}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{k}(x_i - \bar{x})^2} \approx 0{,}009445$$

$$y - Achsenabschnitt: n = \bar{x} - m * \bar{y} = 1{,}88$$

Dabei sind die Tupel x_i und y_i die Koordinaten der Stichprobenwerte mit der Anzahl k, der entsprechende Mittelwert jeweils $\bar{x}$ und $\bar{y}$. Die berechnete Steigung enspricht der mittleren Kennliniensteigung, wodurch die am PTC abfallende Spannung $U_{kty\ linear}$ als Funktion der Temperatur f(ϑ) in Grad Celsius angegeben werden kann. Diese lautet wie folgt:

$$U_{kty\ linear} = f(\vartheta) = m * \vartheta + n = 0{,}009445 * \vartheta + 1{,}88$$

In Abb. 4-8 *temperaturabhängiger Spannungsabfall am KTY81/210* ist die resultierende Gerade schwarz gestrichelt eingezeichnet.

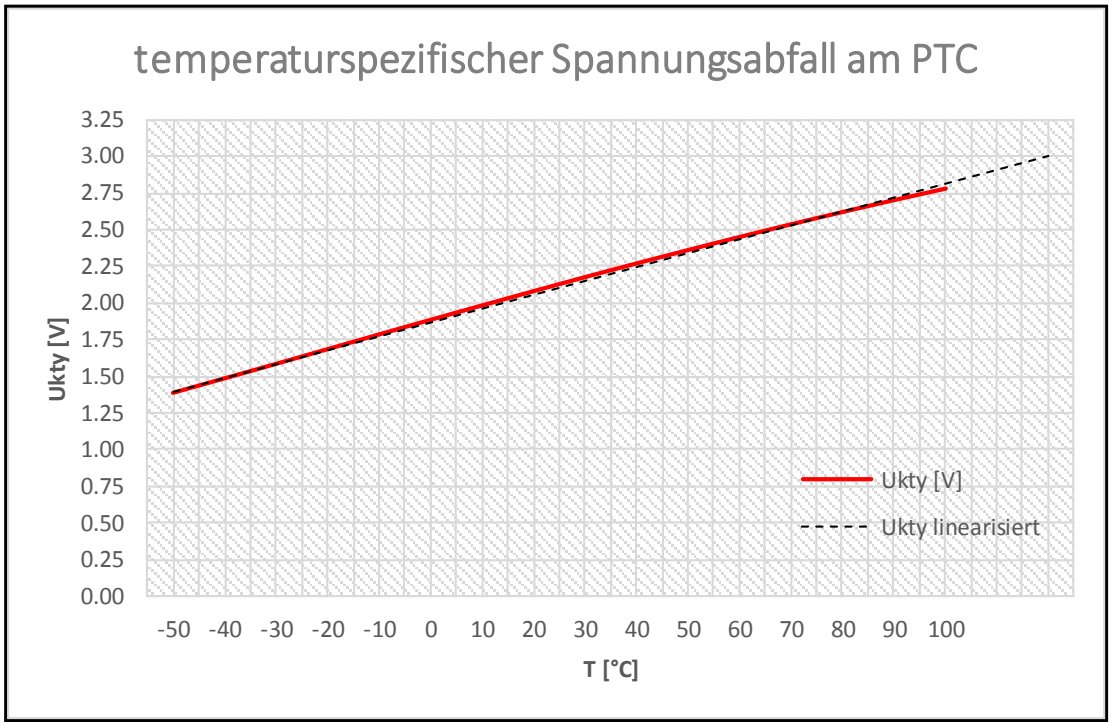

Abb. 4-8 temperaturabhängiger Spannungsabfall am KTY81/210

Da der A/D-Wandler im Mikrocontroller den Messwert nicht als Spannung, sondern als numerischer Wert ausgibt, soll dieser zuletzt wieder in eine Temperaturgrösse übersetzt werden. Dies ist möglich, da die Schrittweite einer Quantisierungsstufe U_{quant} aus der Referenzspannung U_{aref} und Auflösung 2^n des A/D-Wandlers wie folgt berechnet werden kann:

$$U_{quant} = \frac{U_{aref}}{2^n} = \frac{5V}{2^{10}} \approx 0{,}0049V$$

Der numerische Wandler-Ausgangswert AD_{val} erhält man mittels Division des linearisierten Eingangssignals $U_{kty\ linear}$ durch die Schrittweite U_{quant}:

$$AD_{val} = \frac{U_{kty\ linear}}{U_{quant}}$$

Fasst man diese Ergebnisse zusammen kann daraus die am Sensor anliegende Temperatur ϑ aus dem konvertierten Ausgangswert AD_{val} des A/D-Wandler mit folgender Formel berechnet werden:

$$Gemessene\ Temperatur\ [°C]: \vartheta = \frac{U_{kty\ linear} - n}{m} = \frac{AD_{val} * U_{quant} - n}{m} = \frac{AD_{val} * 0{,}0049 - 1{,}88}{0{,}009445}$$

Gesamthaft stehen sechs Analogeingänge bereit. Vier davon werden für die Temperatursensoren im Heizungsvorlauf, Pufferspeicher, Kondensator-Eingang und an der Aussenwand benötigt, die restlichen sind für zukünftige Anwendungen frei.

4.3.3 Lastausgänge

Um Aktoren wie Pumpen und Ventile ansteuern zu können, werden Relais eingesetzt, welche bereits im Kapitel 4.1.3 *Relaismodule* ausführlich behandelt wurden. Es stehen insgesamt 12 Lastausgangskanäle zur Verfügung, welche jeweils einen Öffner- und Schliesser-Kontakt aufweisen. Die Relaisspulen sind durch Optokoppler galvanisch vom Schaltsignaleingang getrennt wodurch Störimpulse, erzeugt durch Schaltvorgänge, nicht auf den Mikrocontroller rückwirken können. In Tabelle 4-C *Belegung der Lastausgänge* sind die angeschlossenen Aktoren, die korrespondierenden Pins am Mikrocontroller sowie die Schaltrichtung ersichtlich.

Kanal	Pin Nr.	Anschluss	Schaltrichtung
1	D30	Kurbelwannenheizung Kompressor	Schliesser
2	D31	Anlassschütz Kompressor	Schliesser
3	D32	Betriebsschütz Kompressor	Schliesser
4	D33	Heissgas-Bypassventil	Schliesser
5	D34	Ventilator	Schliesser
6	D35	Speicherladungs-Pumpe	Schliesser
7	D36	Umwältspumpe Heizkreislauf	Schliesser
8	D37	3-Wege Mischer Drehrichtung auf	Schliesser
9	D38	3-Wege Mischer Drehrichtung zu	Schliesser
10	D39	Sammelalarm	wählbar
11	D40	Reserve	wählbar
12	D41	Reserve	wählbar

Tabelle 4-C Belegung der Lastausgänge

4.3.4 weitere Schnittstellen

Die Grundsteuerung besitzt neben bereits beschriebenen Schnittstellen noch zwei weitere. Erstens diejenige für den Anschluss des Frontend zur Benutzerinteraktion und zweitens den USB Anschluss zur Programmübertragung und Debugging.

Die Frontend-Schnittstelle ist als 16-poliger Flachbandkabel-Verbinder ausgeführt. Der Anschluss überträgt analoge, digitale und Bus-Signale sowie eine Speisespannung von 5V. Somit muss nur ein einziges Kabel zwischen Grundsteuerung und Frontend verlegt werden, was den Installationsaufwand minimiert. Der Anschluss ist vepolungssicher ausgelegt. Die Steckerbelegung ist in Tabelle 4-D *Steckerbelegung Frontendanschluss Grundsteuerungsseite* ersichtlich.

Pin	Belegung	Pin	Belegung
1	Speisung GND (Pfeilmarkierung bei Pin1)	9	Notbedienung aktiv LED Ausgang (D9)
2	Speisung 5V DC	10	Betriebsstörung LED Ausgang (D8)
3	Parallelverschiebung Normalbetr. (A10)	11	Kältemitteldruck i.O. LED Ausgang (D7)
4	Parallelverschiebung reduziert (A11)	12	Manueller Betrieb LED Ausgang (D6)
5	Heizkurvenverstellung (A12)	13	Autobetrieb reduziert LED Ausgang (D5)
6	Notbedienung Nutzereingaben (A13)	14	Autobetrieb normal LED Ausgang (D4)
7	UART RX zu TX Frondend (D19)	15	Standby LED Ausgang (D3)
8	UART TX zu RX Frontend (D18)	16	Eingangssignal Notbedienung aktiv (D2)

Tabelle 4-D Steckerbelegung Frontendanschluss Grundsteuerungsseite

Der USB Anschluss am Arduino dient zum Überspielen eines Programms sowie auch zur Beobachtung des laufenden Betriebs, falls die Debugging-Flags im Programm gesetzt wurden. Dazu wurde der Anschluss so modifiziert, dass keine Speisung des Systems über USB möglich ist. Ansonsten würden systeminterne und USB Stromversorgung miteinander konkurrieren wodurch es zur Schädigung von Mikrocontroller oder Computer kommen kann.

Abb. 4-9 Frontendschnittstelle

Ferner können alle noch unbelegten Anschlüsse am Arduino Mega2560 als weitere Schnittstellen betrachtet werden. Diese müssen vor einer allfälligen Nutzung jedoch noch entsprechend verschaltet und konfiguriert werden.

4.4 Störfestigkeit

Da die Steuerung für einen Schaltkasteneinbau vorgesehen ist, muss mit starken elektromagnetischen Störungen gerechnet werden. Diese werden verursacht durch Schaltvorgänge von Leistungsschützen, wechselspannungsführenden Leitungen und auch durch externe Störsignale welche über abgehende Kabel in den Steuerkasten gelangen. Aus diesem Grunde ist es notwendig, geeignete Massnahmen zur Vermeidung potentieller Störquellen und Behandlung bereits vorhandener Störgrössen zu ergreifen. In diesem Projekt wurde deshalb grossen Wert auf die Optimierung der Störfestigkeit gelegt. Die detaillierte Beschreibung erfolgt nachfolgend unterteilt in die Bereiche Störungsvermeidung und Störgrössenbehandlung wobei zu beachten ist, dass der Übergang beider Themen fliessend ist.

4.4.1 Störungsvermeidung

Zur Vermeidung der gegenseitigen Störbeeinflussungen der Teilsysteme innerhalb der Steuerung, ist die sinnvolle Anordnung der Komponenten essentiell. Sinnvoll heisst in diesem Fall Komponenten, welche starke elektromagnetische Emissionen erzeugen von Komponenten, welche darauf empfindlich reagieren räumlich zu trennen und in grösstmöglichen Abstand zueinander anzuordnen. [12]

In der entwickelten Steuerung sind das elektronische Schaltnetzteil (vgl. Kapitel 4.1.1), die Relaismodule (vgl. Kapitel 4.1.3) und die im Schaltkasten befindlichen Klemmen rechts der Grundsteuerung als starke Emissionserzeuger anzusehen. Die Anordnung auf der Grundsteuerungsplatte wurde deshalb so gestaltet, dass sich das Schaltnetzteil am rechten und die Relaismodule am unteren Rand der Platte montiert sind. Die Einbauposition der Steuerung im Schaltkasten und deren Komponentenanordnung sind in Abb. 4-10 *Anordnung Steuerungskomponenten im Schaltkasten* visualisiert. Es führen keine netzspannungsführenden Leitungen durch die Steuerung, sprich unter oder über der Montageplatte durch.

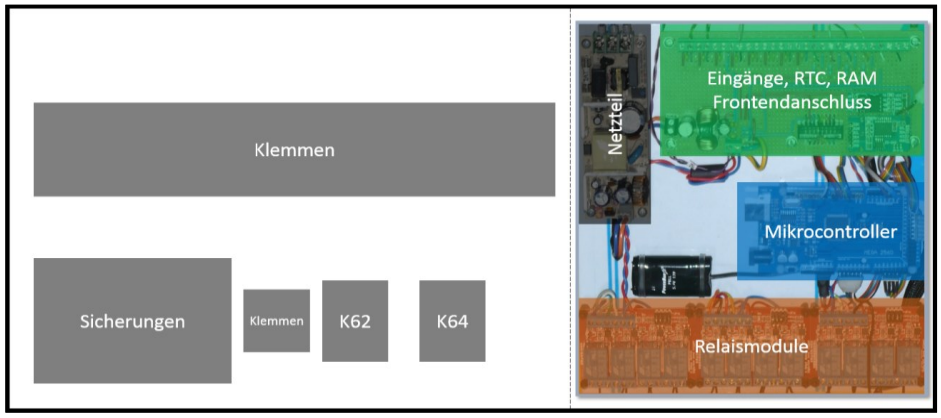

Abb. 4-10 Anordnung Steuerungskomponenten im Schaltkasten

Die steuerungsinterne Verdrahtung ist ebenfalls nicht ausser Acht gelassen worden. Die Verbindung vom Speisespannungs-Verteiler zu den Relaismodulen weist einen Mantelwellenfilter zur Unterdrückung von hochfrequenten Störsignalen auf. Dies aufgrund der Leitungsführung unter dem Schaltnetzteil durch, welches diese Art Signale erzeugt. Zusätzlich sind die Adern verdrillt, was Gleichtaktstörungen verhindert. Einen Mantelwellenfilter weist ebenfalls das Flachbandkabel zwischen den Digitaleingangsklemmen und Mikrocontrollerpins auf. Für die Verbindung zwischen Analogeingangsklemmen und Arduino reicht dies nicht. Hier kommt eine geschirmte Leitung zum Einsatz da schon geringe Störungen ausreichen um analoge Signale zu verfälschen. Die Verdrahtung zwischen Mikrocontroller und Relaismodule kommt dagegen ohne spezielle Massnahmen aus, lediglich eine Verdrillung der Adern liegt vor.

4.4.2 Störgrössenbehandlung

Da sich elektromagnetische Störgrössen nie vollständig eliminieren lassen, müssen zusätzliche Schutzmassnahmen zur Dämpfung, Filterung und Ableitung vorgesehen werden. Die Schutzbeschaltung sollte sich möglichst nah am Verursacher befinden um eine Ausbreitung von Störfrequenzen zu verhindern. Für entwickelte Steuerung heisst das, die Beschaltung möglichst nah an den Abgängen und bei internen Verursachern an der Quelle zu platzieren. Folgend sind die Schutzmassnahmen im Detail erklärt.

Schaltnetzteile erzeugen konstruktionsbedingt Sekundärspannungen welche nur mit erheblichem Aufwand zu glätten sind, weshalb sie meist noch eine hohe Restwelligkeit aufweisen. Für den stabilen Betrieb von Mikrocontrollern ist eine möglichst lineare Versorgungsspannung sehr wichtig, ebenso für passive Messfühler wie PTC-Temperatursensoren, deren Signal ansonsten inkonsistent und damit unbrauchbar wird. Aus diesen Gründen wurde eine zusätzliche Glättung und Filterung der 5V Versorgungsspannung integriert welche in Abb. 4-11 *Siebschaltung 5V- Speisung* schematisch dargestellt ist. Die Schaltung besteht aus einem Glättungskondensator gefolgt von einer stromkompensierten Drossel zur Unterdrückung von Gleichtaktstörungen. Dieser ist ein weiterer Glättungskondensator mit halber Kapazität nachgeschaltet. Zur Filterung allfälliger Gegentaktstörungen befinden sich zuletzt noch zwei keramischer Kondensatoren unterschiedlicher Kapazität um eine möglichst breite Filterwirkung zu erreichen. Die Elektrolytkondensatoren sind Typen mit niedrigem Serienwiderstandsverhalten. Nach dieser Filterschaltung erfolgt die Verteilung der Speisung auf die einzelnen Module. Die Versorgung erfolgt sternförmig. Die Speisung wird also nicht durch mehrere Verbraucher geschleift, sondern jeder Verbraucher erhält eine eigene Zuleitung welche im Sternpunkt zusammenlaufen. Somit können sich die einzelnen Module nicht gegenseitig beeinflussen [13].

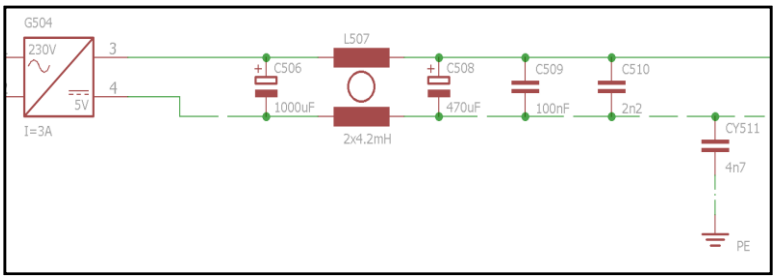

Abb. 4-11 Siebschaltung 5V- Speisung

Die Kopplung der sekundären Masse, also diejenige der 5 Volt Stromversorgung, mit der Schutzerde erfolgt kapazitiv über einen Y-Kondensator. Aufgrund der Ableitung aller Störungen auf die Masse bildet sich dort sogenanntes «ground noise» was zu einem undefinierten Bezugspunkt führt [14]. Verbindet man nun Masse und Erde an einem Punkt, wird dieses Rauschen nach Erde abgeleitet. Eine galvanische Verbindung sollte unterlassen werden, da dies zu Erdschleifen führt. Man nutzt deshalb einen Entstörkondensator mit speziellen Sicherheitsanforderungen, den Y-Kondensator. Die Störsignale höherer Frequenz passieren dabei den Kondensator, ohne dass eine stetig leitende Verbindung zwischen Masse und Schutzleiter besteht [12].

Die Steuerungseingänge sind mit RC-Tiefpassfilter ausgerüstet (vgl. Kapitel 4.3). Die Masseleitung der Abblockkondensatoren ist als separate Verbindung zum Bezugspunkt gehalten. Es wurde generell darauf geachtet, dass keine Leiter über längere Strecken parallel verlaufen und sich jeweils im rechten Winkel überkreuzen. Die Montageplatte aus Aluminium ist ganzflächig geerdet.

5 Systemaufbau Frontend

Das Frontend ist für die Benutzerinteraktion zuständig und kommuniziert mit der Grundsteuerung. Dieses System ist grundsätzlich eine eigene Steuerung mit integrierter Recheneinheit, die auch ohne die Grundsteuerung funktioniert. Die Gründe dazu sind im Kapitel 3.3 *Trennung Benutzer- und Systemumgebung* bereits erläutert worden. Die Bedieneinheit findet sich in einer Aussparung der Wärmepumpen-Verschalung auf Augenhöhe und ist damit von der Grundsteuerung, welche sich auf Bodenhöhe befindet, räumlich getrennt. Folgend wird der Aufbau im Detail erklärt.

5.1 Teilsysteme

Die Frontend-Steuerung besteht hauptsächlich aus der Stromversorgung, einem Mikrocontroller, LCD und Eingabeelementen sowie optischen Zustandsindikatoren. Diese sind auf einem Kunststoffträger befestigt, welcher bei der Montage hinten an die Frontblende angeschraubt wird. Die Blende ihrerseits wird zum Schluss in der Aussparung der Dämmverkleidung fixiert.

5.1.1 Speisung

Die Speisung des Frontend wird von der Grundsteuerung geliefert und beträgt 5 Volt Gleichspannung. Da der verbaute Mikrocontroller eine Betriebsspannung von 3,3V benötigt, wird diese mit einem AMS1117 Festspannungsregler erzeugt. Dieser zeichnet sich durch eine hohe Effizienz und einer Regelabweichung von unter 0,5 Prozent aus. Der maximale Ausgangsstrom beträgt 1A [15].

Vor dem Spannungsregler befindet sich wiederum eine Filterschaltung bestehend aus Glättungskondensator, stromkompensierender Drossel und Abblockkondensator. Am Reglerausgang findet sich wiederum ein Elektrolytkondensator zur Stabilisierung sowie ein Abblockkondensator. Die Verteilung erfolgt wiederum sternförmig zur Reduzierung von Störungen. Auf Abb. 5-1 *Frontend Speisung und ESP8266* ist die Speisung rechts unten ersichtlich.

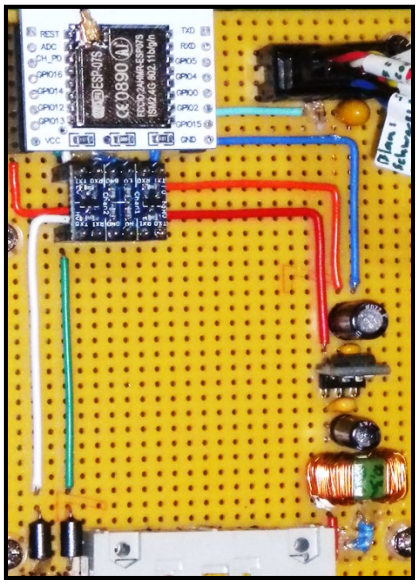

Abb. 5-1 Frontend Speisung und ESP8266

5.1.2 Mikrocontroller

Der Mikrocontroller ist nicht ein Atmega2560 wie auf der Grundsteuerung, sondern ein ESP8266 von Espressif mit integriertem WLAN. Dieser MCU ist in der Maker-Community aufgrund der ansehnlichen Leistung bei geringen Preis sehr beliebt. Entsprechend hoch fällt die Unterstützung für verschiedener Programmierumgebungen aus. Der ESP8266 lässt sich unter anderem in Lua, MicroPython, JavaScript und auch in C++ programmieren [16]. Besonders praktisch ist die Unterstützung der Arduino-Umgebung womit die mittlerweile sehr umfangreiche Code-Library von Arduino ebenfalls genutzt werden kann. Ebenfalls kann damit dem Open-Source Konzept gerecht werden. Die Tabelle 5-A *Technische Daten ESP8266* fasst die wichtigsten Parameter des Chips übersichtlich zusammen. Ein Vergleich mit den Daten des Atmega2560 (vgl. Tabelle 4-A) ist sicherlich lohnenswert.

Eigenschaft	**Wert**
Plattform	Espressif ESP8266
Prozessor	Tensicila Xtensa 32-bit RISC MCU
Typ	ESP-07S
WiFi Protokolle	IEEE 802.11 b/g/n
WiFi Modi & Verschlüsselung	Station/Access-Point/Station+AP, WPA/WPA2
CPU-Takt	80 bis 160MHz
ISP Flash Memory	4096KB
SRAM	128KB
Kommunikations-Schnittstellen	UART, HSPI, I2C, I2S, IR-Remote-Control
Digitale GPIO	13x davon 13x PWM
Analog Inputs	1x 10-bit
Temperaturbereich	-40 bis 125°C
Versorgungsspannung	3.0-3.6V

Tabelle 5-A Technische Daten ESP8266

Das verwendete Modell ist der ESP-07, welcher einen Anschluss für externe WLAN Antennen besitzt, was aufgrund der stark dämpfenden Umgebung sicherlich von Vorteil ist.

Der ESP-07 wird alleine für die Verarbeitung der Benutzer-Ein- und Ausgaben eingesetzt. Dabei kann der User wählen, ob er mit der Wärmepumpen-Heizung vor Ort, also mittels Bedienpanel, interagieren will, oder per Weboberfläche auf einem beliebigen Gerät. Wahlweise kann der Mikrocontroller dazu in ein bestehendes Intranet eingebunden werden, oder als Access-Point ein eigenes WLAN-Netz aufspannen. Die Bedienung per Browser ist derzeit noch nicht umgesetzt worden.

Die Kommunikation zwischen der Grundsteuerung und dem Frontend erfolgt über eine serielle, bidirektionale Schnittstelle mit 115200 bit/s. Über diese tauschen die Controller Systeminformationen und Benutzereinstellungen aus. Da der Atmega die Daten mit einem Spannungspegel von 5V sendet, der ESP jedoch nicht 5V tolerant ist, muss der Signalpegel angepasst werden. Dies geschieht mittels einem sogenannten Level-Converter Baustein welcher die Pegel in beide Richtungen anpasst. Der Level-Converter ist auf dem Frontendsystem eingebaut, damit die Daten mit höherem Spannungspegel durch das Verbindungskabel zur Grundsteuerung gelangen, was die Störungsresistenz erhöht. Die Pinbelegung des Frontendcontrollers ist unter 5.2 *Verdrahtung* aufgeführt.

5.1.3 Display

Beim Display der Lokalbedienung handelt es sich um ein monochromes LCD mit zwei Zeilen a 16 Zeichen und zuschaltbarer Hintergrundbeleuchtung. Die Verbindung zum Mikrocontroller erfolgt über den I2C-Bus was eine erheblich geringere Anzahl an GPIOs benötigt als die parallele Ansteuerung. Das Display kann mit einer Spannung von 3,3V oder 5V gespeist werden. In vorliegendem Fall sind es 3,3 Volt. LCD und Encoder sind auf Abb. 5-2 *LCD und Drehencoder* abgebildet

5.1.4 Drehencoder

Die lokale Eingabe erfolgt über einen Drehencoder was eine komfortable und intuitive Bedienung zulässt. Durch simples Drehen des Encoders nach links oder rechts, kann auf der LCD Ausgabe hoch- und runtergescrollt werden und mittels Drücken des Drehknopfs die Auswahl erfolgen. Der genutzte Encoder ist ein anschlagloser Inkrementalgeber. Die Reihenfolge der Rechtecksignale zweier Ausgänge bestimmt die Drehrichtung, das Signal des eingebauten Drucktasters verfügt über einen eigenen Ausgang. Der Encoder hat Pullup-Widerstände integriert. Drehgeber und LCD sind mit einem gemeinsamen Stecker auf der Frontend-Platine angeschlossen. Eine Entprellung erfolgt hardwareseitig vor dem Mikrocontrollereingang.

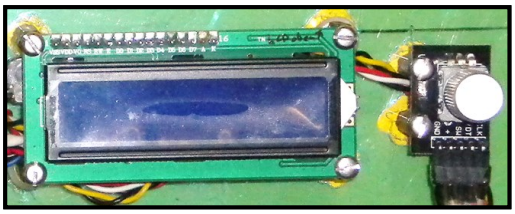

Abb. 5-2 LCD und Drehencoder

5.1.5 Notbedienung

Die Notbedienung soll den Heizbetrieb auch bei Ausfall der Benutzersteuerung ermöglichen. Dazu sind einige Elemente des Frontend-Ausbaus direkt mit der Grundsteuerung verbunden und werden nicht vom Frontendcontroller ausgewertet. Wie für eine Notbedienung üblich, ist die Benutzerinteraktion spartanisch gehalten. Konkret gibt es einen mechanischen Schalter zur Aktivierung, insgesamt sechs Taster zur Funktionswahl, drei Potentiometer zur Heizkurvenanpassung und mehrere LEDs als optische Zustandsindikatoren. Die detaillierten Angaben sind in der Tabelle 5-B *Elemente der Notsteuerung und deren Pinbelegung* unten aufgelistet. Es ist zu beachten, dass die Notbedienung mit den GPIOs des Atmega2560 verbunden ist.

Element	Funktion	GPIO
Schiebeschalter «Lokal»	aktiviert die Notbedienung, überschreibt andere Einstellungen	D2
Taster 1 «Aus»	versetzt die Wärmepumpenheizung in Bereitschaft	A13
Taster 2 «Manuell»	aktiviert den manuellen Betriebsmodus	A13
Taster 3 «Auto reduziert»	aktiviert den reduzierten Automatikbetrieb	A13
Taster 4 «Auto normal»	aktiviert den normalen Automatikbetrieb	A13
Taster 5	Unbelegt und für zukünftige Anwendungen nutzbar	A13
Taster 6	Unbelegt und für zukünftige Anwendungen nutzbar	A13
LED 1 «Aus»	optischer Indikator des Bereitschaftsmodus	D3
LED 2 «Manuell»	optischer Indikator des manuellen Betriebs	D6
LED 3 «Auto reduziert»	optischer Indikator des reduzierten Automatikbetriebs	D5
LED 4 «Auto normal»	optischer Indikator des normalen Automatikbetriebs	D4
LED 5 «Lokal aktiv»	optischer Indikator der Notbedienungsaktivierung	D9
LED 6 «HD/ND ok»	optischer Indikator zum Kältemitteldruck	D7
LED 7 «Alarm»	optischer Indikator zur Störungsanzeige (blinkend)	D8

Tabelle 5-B Elemente der Notsteuerung und deren Pinbelegung

5.2 Verdrahtung

Die Verdrahtung des Frontend ist wiederum reversibel gestaltet in dem die einzelnen Teilsysteme mit Steckverbindern ausgestattet sind. Die Stecker sind codiert und können deshalb nur in den vorgesehenen Gegenstücken eingesteckt werden. Die Steckerbelegung der Verbindung zur Grundsteuerung ist in Tabelle 5-C *Steckerbelegung Frontendverbindung frontendseitig* ersichtlich, die des Encoder und LCD Steckers in Tabelle 5-D *Steckerbelegung Encoder & LCD* und diejenige der Notbedienung in Tabelle 5-E *Steckerbelegung Notbedienung*. Die GPIO-Belegung des ESP-07 kann daraus ebenfalls entnommen werden. Wo nötig wurden Massnahmen zur Störsicherheit ergriffen, welche im Folgeabschnitt 5.3 *Störfestigkeit* dokumentiert sind.

Pin	Belegung	Pin	Belegung
1	GND (Pfeilmarkierung Pin1)	9	LED Notbedienung aktiv
2	5V DC	10	LED Betriebsstörung
3	Notbedienung Parallelvs normal.	11	LED Kältemitteldruck i.O.
4	Notbedienung Parallelvs reduziert	12	LED Manueller Betrieb
5	Notbedienung Heizkurve	13	LED Autobetrieb reduziert
6	Notbedienung Taster	14	LED Autobetrieb normal
7	UART TX zu RX Grundsteuerung	15	LED Aus/Standby
8	UART RX zu TX Grundsteuerung	16	Schalter Notbedienung Ein

Tabelle 5-C Steckerbelegung Frontendverbindung frontendseitig

Pin	ESP-07	Belegung	Pin	ESP-07	Belegung
1	-	Encoder GND (Pfeil bei Pin1)	6	-	LCD GND
2	-	Encoder 3,3V	7	-	LCD 3,3V
3	D05	Encoder CLK	8	D14	LCD SCL
4	D04	Encoder DT	9	D02	LCD SDA
5	D12	Encoder Drucktaster	10	-	*mechanische Codierung*

Tabelle 5-D Steckerbelegung Encoder & LCD

Pin	Belegung	Pin	Belegung
1	Speisung GND (Pfeilmarkierung bei Pin1)	9	LED Notbedienung aktiv
2	Speisung 5V DC	10	LED Betriebsstörung
3	Notbedienung Parallelvs Normalbetr.	11	LED Kältemitteldruck i.O.
4	Notbedienung Parallelvs reduziert	12	LED Manueller Betrieb
5	Notbedienung Heizkurvenverstellung	13	LED Autobetrieb reduziert
6	Notbedienung Nutzereingabetaster	14	LED Autobetrieb normal
7	Schirm	15	LED Aus/Standby
8	*mechanische Codierung*	16	Schalter Notbedienung Aktivierung

Tabelle 5-E Steckerbelegung Notbedienung

5.3 Störfestigkeit

Zur Störfestigkeit wurden an den Versorgungsanschlüssen der Teilsysteme Abblockkondensatoren verbaut, sowie auch an den Signaleingängen. Die Adern zu Encoder und LCD sind verdrillt um Störungen zu eliminieren. Die Verbindung zwischen Notsteuerung und der Frontendplatine erfolgt mittels geschirmtem Kabel aufgrund der störanfälligen Analogsignale. Die serielle Verbindung zwischen Frontend und Grundsteuerung wird durch Ferritperlen geführt um Gleichtaktstörungen zu minimieren. Ein Beispiel von Störsignalen ist in Abb. 5-3 *Störgrössen auf der Speisespannung* zu sehen, welches vom Prüfen des Schaltnetzteilausgang stammt.

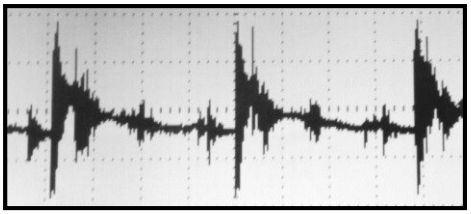

Abb. 5-3 Störgrössen auf der Speisespannung

Durch das integrierte WLAN-Modul des ESP8266 entsteht ebenfalls eine starke elektromagnetische Strahlung. Um die Beeinflussung der Komponenten durch das ausgesendete Signal zu vermeiden, ist der ESP mit einer metallischen Schutzkappe umgeben. Verwendetes Modell ESP-07S besitzt ausserdem keine PCB-Antenne, sondern einen U.FL Stecker für den Anschluss einer externen WiFi-Antenne [17]. Um die Antenne ausserhalb der Wärmepumpe anbringen zu können, wurde dort ein Adapterkabel von U.FL auf SMA angeschlossen. Die SMA-Buchse ermöglicht dabei eine Wanddurchführung. Als Resultat findet die Abstrahlung erst in beträchtlichen Abstand zur Frontendsteuerung statt, was Störungen vermindert und den Betriebsradius stark vergrössert.

6 Programmaufbau

Nachdem die elektrische und mechanische Umsetzung in den vorhergehenden Kapiteln ausführlich beschrieben wurde, soll im weiteren noch die Programmierung der Steuerung behandelt werden. Das modulare Konzept, welches dabei verfolgt und unter 3.2 *Modularität* beschrieben wird, sollte für den nun folgenden Teil stets im Hinterkopf behalten werden. Nur so lassen sich die auf den ersten Blick kompliziert anmutenden Strukturen nachvollziehen. Der Quelltext ist in Arduino geschrieben. Eine Gesamtübersicht der Modulabhängigkeiten ist in Abb. 6-1 *Modulabhängigkeiten* zu finden.

Die Beschreibung des Quelltextes wird in diesem Bericht kurz abgehandelt. Dies weil der Beschrieb und Erklärung jeweils gleich im Quellcode erstellt wurde mit dem Ziel, eine saubere und einfach durchsuchbare Dokumentation zu erhalten. Dazu wurde das freie Programm Doxygen verwendet und die Kommentare im Code mit der nötigen Beschreibungssyntax erstellt. Als Resultat ist jede Datei, Funktion und weitere Elemente ausführlich beschrieben. Diese sehr detaillierte Beschreibung des Codes sowie der visuellen Darstellung der Abhängigkeiten, ist im Anhang als HTML-Datei enthalten, in welcher das komplette Programm interaktiv und mit Erklärungen einsehbar ist.

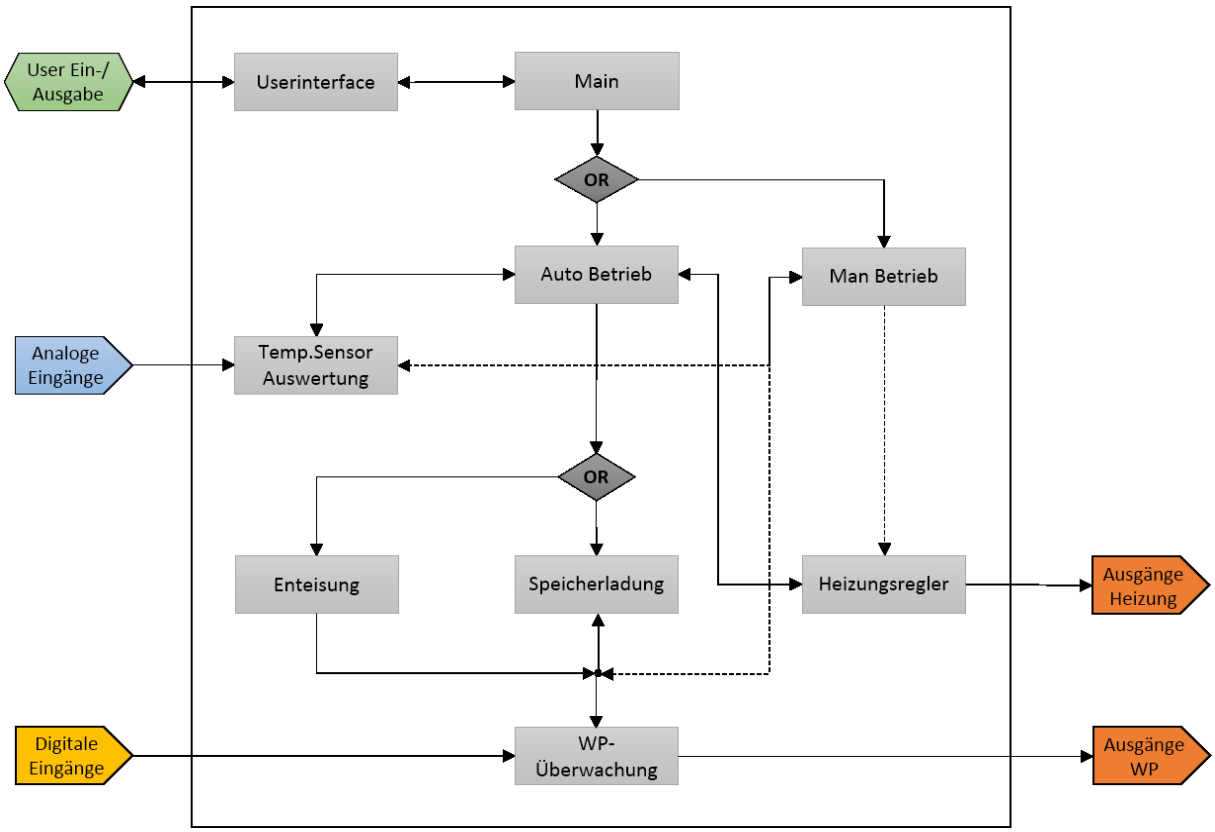

Abb. 6-1 Modulabhängigkeiten

6.1 Formales

Unter Formales sollen dem Leser gewisse Konformitäten welche geschriebener Code aufweist, nähergebracht werden. Dazu zählen beispielsweise Namenssysteme, Strukturierung und Programmierstil. Mit einer kurzen Dokumentation solcher Dinge ermöglicht es einem Aussenstehenden, den schnellen Einstieg ins Programm zu finden.

Grundsätzlich gilt zu beachten, dass Arduino keine eigentliche Programmiersprache ist, sondern vielmehr ein für Anfänger optimierter Überbau der Sprache C++ sowie der GNU Compiler Collection. Beim Kompilieren eines Arduino-Programms wird der Buildvorgang mit GCC einfach durch die Arduino Entwicklungsumgebung gesteuert, wovon der Benutzer freilich nichts bemerkt. Dadurch ist es möglich, ein Programm auch in C++ und, dank der Komptabilität des C++ Compilers zu C, auch in C zu schreiben [18]. Dies wiederum heisst, dass durch sparsames verwenden des arduinospezifischen Syntax der geschriebene Code auch einfach für native C++ Programme portiert werden kann. Für den erstellten Quellcode wurden folgende Richtlinien definiert und angewandt:

1. Nomenklatur:
    a. Variablennamen sind durchwegs klein geschrieben.
    b. Funktionsnamen sind in lowerCamelCase geschrieben.
    c. Klassen und Datenstrukturen beginnen mit Grossbuchstaben.
    d. Makros sind durchwegs gross zu schreiben.
    e. jede ino-Datei besitzt ein gleichnamiges Headerfile.
    f. Kommentare sind mit Doxygen-Syntax zu erstellen.
2. Makros und Headerfiles:
    a. Makros werden nur in Headerfiles definiert.
    b. Makros mit Pinzuweisungen beginnen mit PIN_, Zeitkonstanten mit T_ usw.
    c. Headerfiles sind nur bedingt zu includen.
3. Datenstrukturen:
    a. Strukturtyp-Definitionen erfolgen in Headerfiles.
    b. Enum-Typen resp. deren Typedefinition mit dem Anhängsel *_t zu kennzeichnen.
4. Funktionen:
    a. Funktionsdeklarationen erfolgen in Headerfiles.
    b. Funktionsnamen sollen auf deren Inhalt schliessen lassen.
5. Variablen:
    a. alle ganzzahligen Variablen werden als Standard Integer Types (stdint.h) erstellt.
    b. der Variablentyp ist dem Wertbereich der Variable anzupassen.
    c. der Datentyp Boolean ist zu vermeiden und stattdessen uint8_t zu verwenden.
    d. Gleitkommaarithmetik nur bei absoluter Notwendigkeit da ohne ALU sehr langsam.
    e. die Sichtbarkeit von Variablen ist einzuschränken.

6.2 Module

Die Programmfunktionen, welche jeweils eine gewisse Aufgabe des Wärmepumpenbetriebs wie kontrollieren, steuern und überwachen ausführen, sind in Modulen zusammengefasst. Module haben jeweils eine übergeordnete, spezifische Aufgabe aus welchen sich der Betriebsablauf zusammensetzen lässt. Die Module sind also voneinander abhängig und besitzen dazu geeignete Schnittstellen. Auf Dateiebene gibt es jeweils eine Quelltextdatei und ein Headerfile pro Modul.

Die Verknüpfung der Module lässt sich am besten visuell darstellen. In Abb. 6-1 *Modulabhängigkeiten* wurden deshalb die Verbindung der Module aufgezeichnet, wobei zu beachten ist, dass Enteisung und Speicherladung aufgrund abwechselnder Ausführung in einem Modul zusammengefasst sind. Die Hierarchie der Module sagt aus, welches Modul welche anderen Module aufrufen kann. Die soll dies Abb. 6-2 *Modulhierarchie* visualisieren.

Die Aufgaben der verschiedenen Module sollen im Folgenden beschrieben werden. Weiterführende Beschreibungen sind im Anhang zu finden. Die Module sind dort ebenfalls sehr ausführlich dokumentiert.

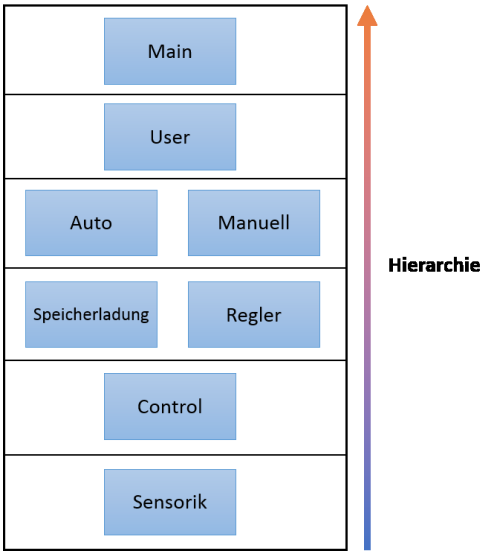

Abb. 6-2 Modulhierarchie

6.2.1 Modul Sensorik

Das Modul Sensorik liest die analogen Eingänge mit den Temperatursensoren aus. Es sorgt dafür, dass die gemessenen Werte in eine Temperaturgrösse übersetzt werden. Ausserdem grenzt es den möglichen Wertebereich des Rückgabewertes ein und verwirft unrealistische Temperaturen. Für eine bessere Konsistenz wird das Signal dreifach gemessen und die Ergebnisse gemittelt. Erst danach erfolgt die Umrechnung und Ausgabe.

6.2.2 Modul Control

Im Modul Control findet die Ansteuerung der Aktoren statt. Es kontrolliert als einziges Modul die Lastausgänge der Aktoren vom Kältemittelkreis. Jeder Anforderung zum Zustandswechsel der Aktoren muss zwingend über das Modul Control gemacht werden, welches prüft ob die Anforderung einen sicheren Betrieb nicht verletzt. Es dient in diesem Fall als Sicherheitsorgan um unzulässige Betriebszustände zu verhindern. Darüber hinaus führt das Modul bei einer gültigen Anforderung die notwendigen Start- und Stopp-Sequenzen selbständig aus.

6.2.3 Modul Speicherladung und Enteisung

Die Speicherladung und Enteisung sind als Zustandsautomat programmiert, welche den Ablauf des Aufheizens steuern. Aufgrund der Tatsache, dass nur aufgrund von Vereisung der Verdampferlamellen während der Speicherladung ein Abtauen erfolgen muss, ist es naheliegend das der als eigenes Modul geplante Enteisungsvorgang in das der Speicherladung migriert wird. Zusammen bilden sie nun einen Automaten, welcher die Speichertemperatur zyklisch mit der Solltemperatur des Vorlaufs vergleicht und bei Unterschreiten einer Einschaltgrenze den Speicher wieder auflädt bis dieser die Ausschaltgrenze erreicht (vlg. [19, 20]). Ein Abtauen wird dabei in Abhängigkeit der Aussentemperatur eingeleitet.

6.2.4 Modul Regler

Beim Regler handelt es sich um einen quasi-stetigen Dreipunktregler, welcher die Vorlauftemperatur der Bodenheizung über einen Dreiwege-Mischer regelt. Die Solltemperatur des Vorlaufs wird dabei aus der eingestellten Heizkurvenfunktion in Abhängigkeit der Parallelverschiebung und Aussentemperatur berechnet, was ebenfalls im Modul Regler geschieht. Der entworfene Regler wird bevorzugt für derartige Temperaturmischer eingesetzt, da diese schneller reagieren was bei einem sehr trägen System durchaus von Vorteil ist. Um dies zu erreichen wartet der Regler nicht die Ausgangsänderung ab, sondern verrechnet die Abweichung mit einem Wert, der dem Rückführwert bei aktuellem Reglerausgang entsprechen würde. Die zeitabhängige Grösse des Pseudorückführwertes, also der Integral Anteil wird über ein Speicherglied erreicht. Durch Ergänzung einer Totzone und Hysterese wird ein Schwingen des Reglers unterbunden [21, 22]. Der Regler steuert die Lastausgänge des Mischventils und der Umwälzpumpe im Heizkreis eigenständig.

6.2.5 Modul Auto

Das Modul Auto ist für den Automatikbetrieb zuständig und ruft die jeweils benötigten Module auf tieferer Stufe auf. Ebenso aktiviert es den Regler welcher für die korrekte Vorlauftemperatur zuständig ist. Die Wärmeenergie-Erzeugung und Heiztemperaturanpassung erfolgt in diesem Modus ohne weiteres Zutun des Benutzers

6.2.6 Modul Manuell

Im Modul Manuell kann die Ansteuerung der Aktoren und der Heizbetrieb in gewissem Grad von Hand durch den Benutzer erfolgen. Da die Anforderungen auch hier in jedem Fall über das Sicherheitsmodul «Control» laufen, können keine unzulässigen Betriebszustände ausgelöst werden. Der manuelle Modus ist für Wartungs- und Prüfzwecke vorgesehen und wird dementsprechend selten gebraucht.

6.2.7 Modul User

Das Usermodul enthält Funktionen zur Kommunikation mit dem Frontend, der Speicherung von Nutzereinstellungen und zur Verarbeitung der Eingaben der Notbedienungseinheit. Es sorgt dafür, dass auf der seriellen Schnittstelle eingehende Nachrichten verarbeitet und gegebenfalls als Einstellungen gespeichert werden. Ebenso sorgt es dafür, dass der Systemzustand und Messwerte periodisch an das Frontend gesendet werden. Im Falle einer Aktivierung der Notbedienung, werden darüber eingehende Nutzerwünsche priorisiert und die Ausgabe des Systemzustands an die dortigen Signalleuchten aktiviert.

6.2.8 Modul Main

Das Modul Main beinhaltet die Systeminitialisierung, die globalen Objekte und die oberste Programmebene, welche die anderen Module direkt oder indirekt anspricht und für den zyklischen Programmdurchlauf sorgt.

6.3 Frontend-Programm

Das Programm des Frontend wird in diesem Bericht nicht behandelt. Der primäre Fokus dieser Arbeit lag dabei auf der Grundsteuerung, da eine Implementierung des Frontend-Programms mit Webinterface und Lokalbedienung den Rahmen des Projekts gesprengt hätte. Die Umsetzung dürfte mit der hier geleisteten Vorarbeit jedoch nicht schwerfallen.

7 Zusammenfassung und Fazit

Nach der Umsetzung der Arbeit stellt sich die Frage, in wie weit die Anforderungen nun tatsächlich erfüllt wurden und wie gut deren Umsetzung gelang. Zusammenfassend kann festgehalten werden, dass die gegebenen Anforderungen an die Arbeit erfüllt wurden und ein solider und durchdachter Prototyp entstanden ist, welcher sich nicht vor kommerziellen Lösungen zu verstecken braucht. Die Steuerung bietet gegenüber der Originalsteuerung der SWAP-15 deutliche Mehrwerte wie geringerer Stromverbrauch, höherer Funktionsumfang, Erweiterbarkeit und ortsunabhängige Bedienung. Nicht alle Wünsche und Ideen konnten im Zeitraum, der zur Verfügung stand, umgesetzt werden und eine weitere Optimierung ist sicher nötig. Die Basis jedoch ist geschaffen und diese erfüllt ihren Zweck umfassend. Die Erstellung als modulares System und mittels quelloffenen Tools wurde ebenfalls erreicht. Die Investitionskosten in verwendete Komponenten beliefen sich insgesamt auf weniger als hundert Franken.

Als abschliessendes Fazit kann gesagt werden, dass die Umsetzung einer solchen Steuerung mittels Komponenten, welche zu oft noch als untaugliche Basteleien und Spielzeuge angesehen, kostengünstig möglich ist. Da auf Plattformen wie Arduino und Raspberry-Pi immer mehr Projekte entstehen, welche für den industriellen Einsatz konzipiert und auch eingesetzt werden, ist hier ein Umdenken angebracht. Als technisch versierter Konsument ist der zunehmende Einfluss von quelloffenen Produkten und deren Weiterentwicklungen als positiv zu werten. Mit geringem finanziellen Aufwand können neue Ideen umgesetzt oder kostspielige Austauschkomponenten mit etwas Arbeit selber gebaut werden, was neben dem Geldbeutel auch die Umwelt schont. Die Arbeit an der Steuerung ist mit Abschluss der Thesis nicht beendet, sondern soll in Zukunft weitergehen, was im nächsten Kapitel noch ausgeführt wird.

8 Ausblick und Dank

Es liegt an der Natur der Sache, dass die Entwicklung eines Produkts nicht mit der Planung abgeschlossen ist. Während man mit der Umsetzung beschäftigt ist, kommen stetig neue Ideen und Verbesserungen hinzu, welche man im Nachhinein in das Produkt einfliessen lassen würde. Auch Wunschanforderungen, welche aufgrund des begrenzten Zeitrahmens nicht umgesetzt werden können, verschwinden oft in der Versenkung obschon sie vielleicht genau das Kaufargument, die Innovation oder das nützlichste Feature gewesen wären. Es wäre deshalb wohl ein Gedanken wert, ein fertiges Produkt nicht fertig zu sein lassen. Es als erste Version, als noch ungeschliffenen Diamanten zu betrachten und die Eingebungen in der zweiten Version einzubeziehen. Die entwickelte Steuerung in dieser Arbeit ist so ein Produkt, wo noch viele Ideen vorhanden sind. Deshalb soll sie in Zukunft weiterentwickelt werden. Konkrete Pläne sind einerseits die Fertigstellung des Frontend und Implementierung der Weboberfläche sowie einen Messenger-Bot an welchen Bedienbefehle gesendet werden können. Eine Aufzeichnung verschiedener Daten und Darstellung mittels Chart ist ein weiterer Plan. Neben der Optimierung des Programms und Erweiterung des Funktionsumfangs soll der Quellcode und darin enthaltene Beschreibung auf Englisch übersetzt und auf einschlägigen Maker-Webseiten veröffentlicht werden.

Trotz aller weiteren Ideen und Plänen, bin ich nun froh, den fertigen Bericht in den Händen zu Halten und einen weiteren Meilenstein der Thesis geschafft zu haben. An dieser Stelle ist es mir ein Bedürfnis, den zum Gelingen beitragenden Personen meinen Dank auszusprechen.

Besonderen Dank gilt [redacted], der als Fachbetreuer mit Rat und Tat zur Seite stand sowie mit wertvollen Beiträgen und fachspezifischen Fragen massgeblich zum Gelingen beitrug. Weiter möchte ich [redacted] für das Ermöglichen dieses Projektes herzlich danken. Im Folgenden möchte ich noch [redacted] für sein Engagement und zur Verfügung stellen als externen Experten sowie [redacted] für Zurverfügungstellung der Anlage einen grossen Dank aussprechen.

[redacted] 23.03.2018

[redacted]

9 Literatur

[1] M. Ehrbar and P. Hubacher, “Verbesserung des Abtauens bei luftbeaufschlagten Verdampfern: Phase 3: technische Umsetzung, Labor- und Feldversuche,” 2005.
[2] Josephy, Nipkow, Bush, and Berger-Wey, *Ratgeber Wärmepumpen.* [Online] Available: http://www.topten.ch/private/adviser/ratgeber-warmepumpen. Accessed on: Dec. 22 2017.
[3] Bundesamt für Statistik, *Energiebereich: Heizsystem und Energieträger.* [Online] Available: https://www.bfs.admin.ch/bfs/de/home/statistiken/bau-wohnungswesen/gebaeude/energiebereich.html. Accessed on: Dec. 28 2017.
[4] D. Schmissrauter, “Entwicklung einer Waermepumpen-Steuerung,” Dokumentation, FHNW, 2017.
[5] Atmel, “Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V Datasheet,” 2014.
[6] Tecate Group, “Passively balanced ultracapacitor module Data Sheet,” 2017. [Online] Available: https://www.tecategroup.com/products/data_sheet.php?i=PBLL&t=SERIES. Accessed on: Dec. 05 2017.
[7] Atmel, “2-Wire Serial EEPROMs: AT24C256,” 1998. [Online] Available: http://pdf1.alldatasheet.com/datasheet-pdf/view/56138/ATMEL/24C256.html.
[8] K. Padberg, *Störbeeinflussung durch nicht beschaltete Lasten / Kontakte.* [Online] Available: http://www.comat.ch/doc/fa/Fa_StoerbeeinfICEC1_d.pdf. Accessed on: Jan. 10 2018.
[9] P. Schnabel, *Glättung und Siebung.* [Online] Available: http://www.elektronik-kompendium.de/sites/slt/0210251.htm. Accessed on: Mar. 20 2018.
[10] sprut.de, *Temperaturmessung: Grundlagen der Temperatursensoren.* [Online] Available: http://www.sprut.de/electronic/temeratur/temp.htm. Accessed on: Nov. 30 2017.
[11] NXP, “KTY81 series data sheet: Silicon temperature sensors,” 2008. [Online] Available: https://www.nxp.com/docs/en/data-sheet/KTY81_SER.pdf. Accessed on: Nov. 11 2017.
[12] U. Lorenzen, “Elektromagnetische Verträglichkeit und induktive Bauelemente,” Schulungsunterlagen, Wolpertshausen, 2014.
[13] M. Fortunato, *Successful PCB Grounding with Mixed-Signal Chips.* [Online] Available: https://www.maximintegrated.com/en/app-notes/index.mvp/id/5450. Accessed on: Mar. 13 2018.
[14] O. Pirlich, “Was ist Masse? Oder warum Cinch verboten gehört: Eine Zusammenfassung über analoge Audio-Schnittstellen aus der Sichtweise der Selbstbau Community.,” 2017. Accessed on: Dec. 05 2017.
[15] Advanced Monolithic Systems, “AMS1117 Datasheet: 1A low dropout voltage regulator,” 2016. [Online] Available: http://www.advanced-monolithic.com/pdf/ds1117.pdf. Accessed on: Dec. 17 2017.
[16] Wikipedia, *ESP8266.* Accessed on: Mar. 21 2018.
[17] Ai-Thinker Technology, *ESP-07S Datasheet.* [Online] Available: http://wiki.ai-thinker.com/_media/esp8266/docs/aithinker_esp_07s_datasheet_en_v1.0.pdf. Accessed on: Dec. 30 2017.
[18] Wikipedia, *GNU Compiler Collection.* [Online] Available: https://de.wikipedia.org/wiki/GNU_Compiler_Collection. Accessed on: Nov. 02 2017.
[19] Bundesamt für Konjunkturfragen, “Wärmepumpen: Planung, Bau und Betrieb von Elektrowärmepumpenanlagen,” *RAVEL*, no. 3, 1993.
[20] Bundesamt für Konjunkturfragen, “Standardschaltungen: Praxiserprobte Schaltungen für Wärmepumpen, Wärmekraftkopplung, Wärmerückgewinnung und Abwärmenutzung,” *RAVEL*, vol. 1995, no. 5, 1995.

[21] Siemens Building Technologies, “Electrical Actuators SSY319…,” 2003. [Online] Available: http://novreczky.eu/siemens/beavatkozok/pdf/ssy319%20q4899en%5B1%5D.pdf. Accessed on: Jan. 13 2018.
[22] Bundesamt für Konjunkturfragen, “Gebäudeautomation: Inbetriebsetzung und Abnahme,” *RAVEL*, vol. 1992, 1992.

10 Abbildungsverzeichnis

11 Anhang

Der Anhang ist aufgrund der Grösse und den vielen unterschiedlichen Dateiformaten als Ordnerstruktur gemäss «Handbuch Projektschiene SG ST_1718» wie folgt organisiert:

01 Bericht	11 Dokumentation
	12 Fachartikel
	13 Plakat
	14 Bilder
02 Technische Unterlagen	21 Quelltext Grundsteuerung
	22 Doxygen Codedokumentation
	23 Quelltext Frontend
	24 Elektroschema
	25 Hydraulikschema
	26 Heizkurve und PTC Berechnungen
	27 WP Systemanalyse Dokumentation
	28 Temperaturmessreihen Auswertung
03 Projektdokumente	31 Projektauftrag
	32 Projektvereinbarung
	33 Zeitplanung
	34 Pflichtenheft
	35 Statusberichte
	36 Präsentationen
04 Anhang	41 Datenblätter
	42 Quellenliteratur

12 Glossar

CCPL	Creative Commons Public License: Open-Source Lizenz für Hard- und Software
Deadlock	Stelle in einem Programm, die nicht mehr verlassen werden kann
Debouncing	Verhinderung multipler Impulse beim Umschalten eines mech. Kontaktes.
EEPROM	Electrically Erasable Programmable Read-Only Memory
GIT	Versionsverwaltungssystem für Dateien
GUI	Grafical User-Interface Interaktive Bedienoberfläche an einer Maschine.
GPIO	General-Purpose-Input/Output
GPL	General Public License: Open-Source Lizenz für Software
HMI	Human-Machine-Interface Benutzerschnittstelle an einer Maschine.
I2C	serieller Datenbus für geräteinterne Kommunikation
IC	Integrated-Circuit: generelle Bezeichnung für integrierte Schaltungen
IoT	Internet of Things
ISP	In-System-Programmierung
LCD	Liquid-Crystal-Display
LED	Licht emittierende Diode
MIT-License	Massachusetts Institute of Technology License: Open-Source Lizenz
PCB	Printed-Circuit-Board: Leiterplatte / Platine
Polling	Periodisches Abfragen eines Teilnehmers
PTC	Positive Temperature Coefficient (Thermistor): Kaltleiter
RAM	Random-Access-Memory: Flüchtiger Arbeitsspeicher
RTC	Real-Time-Clock: Echtzeituhr meist mit Pufferbatterie
SoC	System on a Chip
TTL	Transistor-Transistor-Logik: Standardschaltungen mit 5V Versorgungsspannung
UART	Universal Asynchronous Receiver Transmitter: serielle Schnittstelle

13 Ehrlichkeitserklärung

Hiermit erkläre ich, die vorliegende Bachelor Thesis selbständig, ohne Hilfe Dritter und nur unter Benutzung der angegebenen Quellen verfasst zu haben.

Ort und Datum: [redacted]

[redacted]