Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Alice Heeb

Smart Pro HeaT: Entwurf und Umsetzung einer bedarfsgeführten Regelung zur Optimierung der Vorlauftemperatur

Fakultät Technik und Informatik
Department Maschinenbau und Produktion

Faculty of Engineering and Computer Science
Department of Mechanical Engineering and Production Management

Alice Heeb

Smart Pro HeaT: Entwurf und Umsetzung einer bedarfsgeführten Regelung zur Optimierung der Vorlauftemperatur

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Energie- und Anlagensysteme
am Department Maschinenbau und Produktion
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

in Zusammenarbeit mit:
Competence Center für erneuerbare Energien
und Energieeffizienz (CC4E)
„Smart Pro HeaT“
Am Schleusengraben 24
21029 Hamburg

Erstprüfer: Prof. Dr. Thomas Veeser
Zweitprüfer: M. Sc. Philipp Eike Janßen

Abgabedatum: 20. Juli 2021

Alice Heeb

Thema der Bachelorthesis
Smart Pro HeaT: Entwurf und Umsetzung einer bedarfsgeführten Regelung zur Optimierung der Vorlauftemperatur

Stichworte
Regelung, Reglerentwurf, Ventilöffnungsgrad, Vorlauftemperatur

Kurzzusammenfassung
Diese Arbeit umfasst den Reglerentwurf für eine bedarfsgeführte Vorlauftemperaturregelung. Es wird untersucht, ob die in einem Testgebäude eingestellte Heizkreisvorlauftemperatur durch die bedarfsgeführte Regelung verringert werden kann. Der Bedarf wird durch den Ventilöffnungsgrad der Heizkörperthermostate bestimmt. Das Programm des Reglers wird in der Progammiersprache Python umgesetzt. Die Regelung wird über mehrere Testphasen optimiert. Die Ergebnisse zur benötigten Vorlauftemperatur während der finalen Testphase werden abschließend mit den Werten der ursprünglichen, außentemperaturgeführten Regelung verglichen. Es konnte eine Absenkung von 10-15 °C erreicht werden.

Alice Heeb

Title of the paper
Design and implementation of a demand-driven control system to optimize the flow temperature

Keywords
control, controller design, valve opening degree, flow temperature

Abstract
This work includes the controller design for a demand-led flow temperature control. It is investigated whether the heating circuit flow temperature set in a test building can be reduced by the demand-led control. The demand is determined by the valve opening degree of the radiator thermostats. The program of the controller is implemented in the Python programming language. The control is optimized over several test phases. The results for the required flow temperature during the final test phase are then compared with the values of the original, outside temperature guided control. A reduction of 10-15 °C could be obtained.

Aufgabenstellung

Im Rahmen dieser Arbeit soll vorübergehend eine bedarfsgeführte Regelung in einem Testgebäude implementiert werden. Ziel der Arbeit soll eine Erkenntnis darüber sein, ob und inwieweit sich die Vorlauftemperatur eines Heizkreises im Vergleich zur vorherigen außentemperaturgeführten Regelung optimieren lässt.

Ansatz:

- Es soll eine bedarfsgeführte Regelung als Alternative zur Heizkurve entworfen und an einem Testgebäude umgesetzt werden.
- Der Bedarf soll «Schlechtraum» basiert geregelt werden: die Vorlauftemperatur wird so weit abgesenkt, dass der am schlechtesten mit Wärme versorgte Raum gerade noch ausreichend beheizt werden kann.

Ausgangslage / Randbedingung:

- Im Rahmen einer vorangegangenen Studienarbeit wurde eine Auswahl an Schlechträumen bereits identifiziert und daraus ein Referenzraum bestimmt.
- Geregelt wird anhand des Ventilöffnungsgrads der Thermostate im Referenzraum.
- Die Umsetzung des Reglers erfolgt in der Programmiersprache Python.

Zwischenziele:

- Es soll ein gängiges Verfahren zum Reglerentwurf, wie in der Vorlesung zur Regelungstechnik behandelt, angewandt werden.
- Zunächst soll überprüft werden, ob sich die Regelung durch einen Standardregler realisieren lässt.
- Kann kein Standardregler verwendet werden, so soll eine heuristische Methode gewählt und ein eigenes Programm geschrieben werden.

Inhaltsverzeichnis

Symbol- und Abkürzungsverzeichnis

$x(t)$	Regelgröße
$w(t)$	Führungsgröße
$e(t)$	Regeldifferenz
$y(t)$	Stellgröße
$z(t)$	Störgröße
K_{PR}	Proportionalitätsbeiwert Regler
K_{PS}	Proportionalitätsbeiwert Strecke
K_I	Integrierbeiwert
T_I	Integrationszeitkonstante oder Nachstellzeit
T_g	Ausgleichszeit
T_u	Verzugszeit
T_t	Totzeit
t	Zeit
T	Temperatur
Q_H	Wärmeleistung
Q_V	Heizleistung
T_A	Temperatur außen
T_R	Temperatur im Raum

T_V	Temperatur im Vorlauf
SPHT	Smart Pro Heat
CC4E	Competence Center für erneuerbare Energien und Energieeffizienz
VLT	Vorlauftemperatur
P-Beiwert	Proportionalitätsbeiwert
WÜST	Wärmeübergabestation
VÖG	Ventilöffnungsgrade der Heizkörperthermostate
SPS	Speicherprogrammierbare Steuerung
AuT	Außentemperatur
EE	Erneuerbare Energien
HKT	Heizkörperthermostate

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

In diesem Kapitel wird zunächst die Motivation für das Verfassen dieser Arbeit erläutert und das Projekt beschrieben, in dessen Kontext das Thema erarbeitet wurde. Es folgt eine Kurzfassung der Vorarbeit, die für die Umsetzung dieser Arbeit von Bedeutung ist. Im letzten Abschnitt wird auf den Aufbau der Arbeit eingegangen.

1.1 Motivation und Kontext

In Anbetracht des immer weiter fortschreitenden Klimawandels ist es unabdingbar, Emissionen zu reduzieren. Der Anteil der erneuerbaren Energien (EE) muss erhöht werden. In Abbildung 1.1 ist der Primärenergieverbrauch in Deutschland im Jahr 2018 dargestellt. Der Anteil an erneuerbaren Energien lag bei 13,8 %.

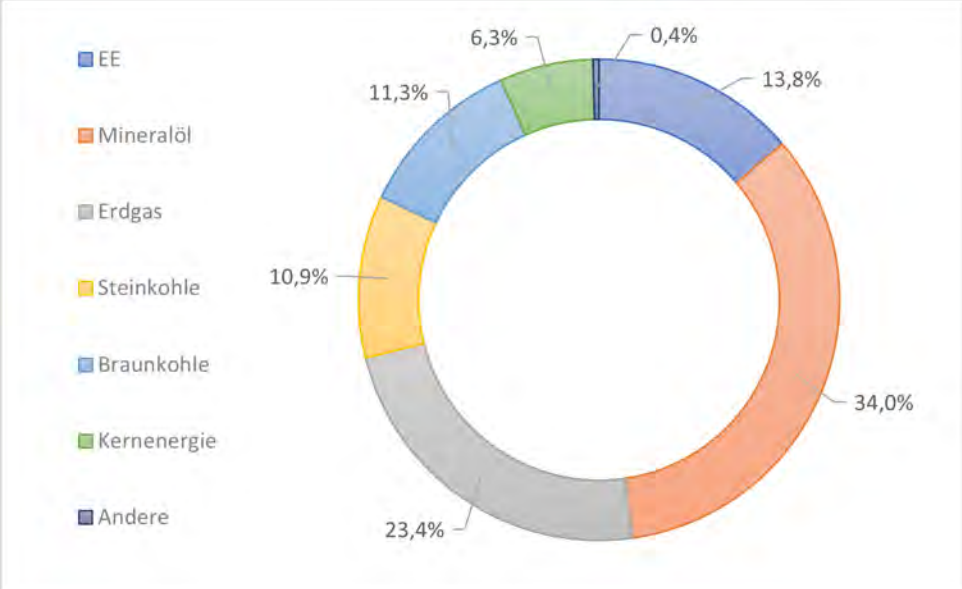


Abbildung 1.1: Primärenergieverbrauch in Deutschland 2018 [Bun19, S. 11]

Da ein komplettes Umstellen von fossilen auf erneuerbare Energien Zeit braucht, ist es mindestens ebenso wichtig, den momentanen Verbrauch von nicht erneuerbaren Energien effizienter und damit emissionsärmer zu gestalten.

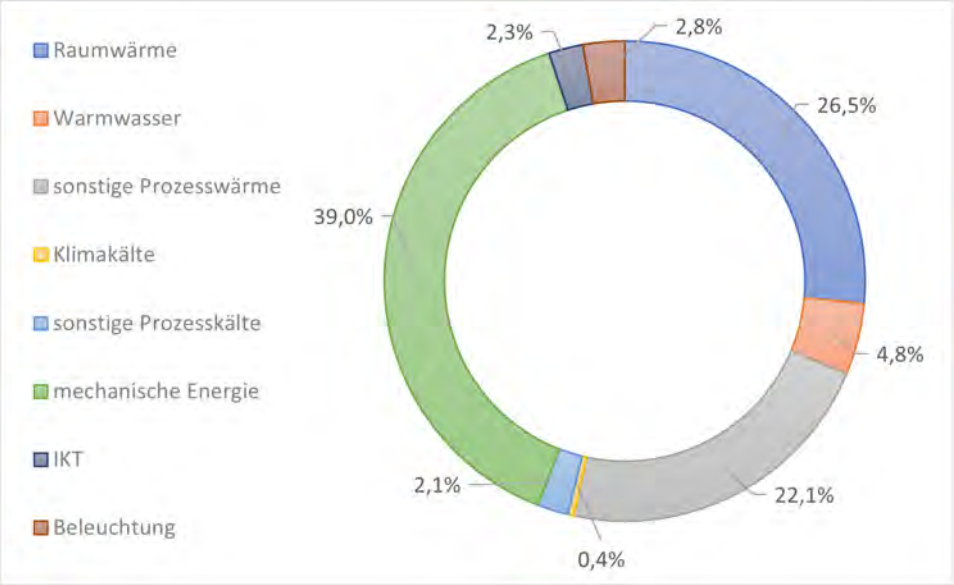


Abbildung 1.2: Energieverbrauch nach Anwendungsbereichen in Deutschland 2017 [Bun19, S. 17]

Wie in Abbildung 1.2 zu sehen ist, fallen 26,5 % des gesamten Endenergieverbrauchs in Deutschland im Jahr 2017 auf die Erzeugung von Raumwärme [Bun19, S. 17]. In diesem Anwendungsbereich liegt demzufolge großes Einsparpotential. Die Klimaschutzziele der Bundesregierung sehen einen klimaneutralen Gebäudebestand bis 2050 vor. Um dieses Ziel zu erreichen, hat das Umweltbundesamt im Jahr 2014 ein Referenzszenario für bestehende Gebäude entwickelt. Es müsste demnach eine Verminderung des Heizenergiebedarfs von circa 150 $\frac{kWh}{m^2a}$ aus dem Jahr 2008 auf 74 $\frac{kWh}{m^2a}$ bis 2050 erfolgen. Es wäre dann möglich, den verbleibenden Energiebedarf nahezu vollständig mit erneuerbaren Energien zu decken. Die Wärmeversorgung soll mehrheitlich mit Wärmepumpen und Kraft-Wärme-Kopplung realisiert und durch Solarthermie unterstützt werden [Umw14, S. 5–9].

In dieser Arbeit soll eine Möglichkeit zur Optimierung der Vorlauftemperatur in Gebäuden untersucht werden. Niedrige Vor- und Rücklauftemperaturen führen zu einer Effizienzsteigerung bei der Wärmeerzeugung. Die im Wärmenetz entstehenden Transportverluste können außerdem verringert werden [ØS16, S. 375]. Das Thema der vorliegenden Bache-

lorarbeit wurde im Rahmen des Projektes «Smart Pro Heat» (SPHT) am Energie-Campus in Bergedorf erarbeitet. Das Technologiezentrum ist Teil der wissenschaftlichen Einrichtung «Competence Center für Erneuerbare Energien und Energieeffizienz» (CC4E) der HAW Hamburg. Das CC4E forscht und entwickelt Lösungen zu Themen rund um die Energiewende. Zusammen mit dem Projekt «Smart Heat Grid Hamburg» (SHGH) wird im Rahmen von SPHT untersucht, wie eine Erweiterung intelligenter Wärmenetze mit intelligenter Gebäudeautomation die Gesamteffizienz und den Anteil erneuerbarer Strom- und Wärmeerzeugung erhöhen kann. Die Forschungsinhalte von SHGH fokussieren sich dabei auf die Wärmeerzeugung und -verteilung bis zur Liefergrenze des Netzbetreibers und somit auf die Primärseite der Wärmeübergabestationen. SPHT entwickelt Konzepte für die Sekundärseite [CC421].

1.2 Vorarbeit

Für die Entwicklung einer bedarfsgeführten Regelung, wie sie in der Aufgabenstellung beschrieben ist, wurden in einer vorangegangenen Studienarbeit die Räume des Testgebäudes auf ihren Wärmebedarf untersucht [Hee20]. Ziel hierbei war es, Räume zu finden, die, im Vergleich zu den anderen Räumen, einen erhöhten Wärmebedarf aufweisen. Konkret wurden verschiedene Möglichkeiten herausgearbeitet, wie solche Räume auch ohne Messtechnik gefunden werden könnten. Anhand dieser Möglichkeiten wurde eine erste Auswahl an «Schlechträumen» getroffen. Diese wurde anschließend mit der vorhandenen Messtechnik überprüft. Dabei erfolgte ein Vergleich der Ventilöffnungen der Heizkörperthermostate bei unterschiedlichen Vorgaben für die Solltemperatur im Raum. Müssen die Thermostate bereits bei niedriger Solltemperatur weit öffnen, so kann dies auf einen erhöhten Wärmebedarf, beziehungsweise auf eine unzureichende Versorgung mit Wärme zurückzuführen sein. Es wurden abschließend fünf Räume definiert, die, im Vergleich zu allen anderen Räumen im betrachteten Heizungsstrang, nur schlecht mit Wärme versorgt werden. Aus diesen fünf Räumen wurde ein Referenzraum bestimmt, der für die folgende Regelung zum Einsatz kommen soll. Mit einem solchen Referenzraum soll sichergestellt werden, dass allen anderen Räumen genügend Heizleistung zur Verfügung steht. Es wird zunächst davon ausgegangen, dass dies in der Regel der Raum mit dem größten Wärmebedarf und somit mit der höchsten Ventilstellung ist. Da im Verlauf der Arbeit nicht alle Räume die gewünschte Temperatur erreichten, musste die Wahl des Referenzraumes noch einmal angepasst werden.

1.3 Aufbau der Arbeit

Das Ziel der Arbeit ist die Antwort auf die Frage, ob und wenn ja, um wieviel die Vorlauftemperatur mit der neu umgesetzten, bedarfsgeführten Regelung abgesenkt werden kann. Hierfür soll im Anschluss an diese Einleitung der notwendige Stand der Technik im Bereich der Vorlauftemperaturregelung geklärt werden. Die theoretischen Grundlagen zur Regelungstechnik, insbesondere zum Reglerentwurf, werden in Kapitel 3 erklärt. Anschließend werden in der Methode zunächst die technischen Rahmenbedingungen für das Testgebäude geklärt. Weiter werden in diesem Kapitel der eigene Ansatz für die Umsetzung der Regelung vorgestellt und das Vorgehen für den Reglerentwurf, wie es in den Grundlagen beschrieben ist, angewandt. In Kapitel 5 wird die Umsetzung des realisierten Programms für den Regler beschrieben, sowie die Testphasen, welche für die erfolgreiche Implementierung erforderlich waren. Zum Schluss folgt die Darstellung und Diskussion der Ergebnisse während der finalen Testphase mit abschließendem Fazit und eine Zusammenfassung der gesamten Arbeit.

2 Stand der Technik

Dieses Kapitel befasst sich mit dem Stand der Technik zu den verschiedenen Möglichkeiten der Vorlauftemperaturregelung. Zunächst wird die klassische außentemperaturgeführte Regelung beschrieben, wie sie momentan im Testgebäude angewendet wird. Der zweite Abschnitt behandelt zwei Ansätze zur Umsetzung der bedarfsgeführten Regelung.

2.1 Außentemperaturgeführte Regelung

Für die Bestimmung der Vorlauftemperatur von Heizungsanlagen ist heute die außentemperaturgeführte Regelung am häufigsten anzutreffen [Heß20, S. 41]. Diese Regelung erfolgt anhand einer bei der Anlagenauslegung erstellten Heizkurve, welche den Zusammenhang zwischen Vorlauftemperatur und Außentemperatur vorgibt. In Abbildung 2.1 sind beispielhaft Heizkurven für unterschiedliche Heizungstypen dargestellt.

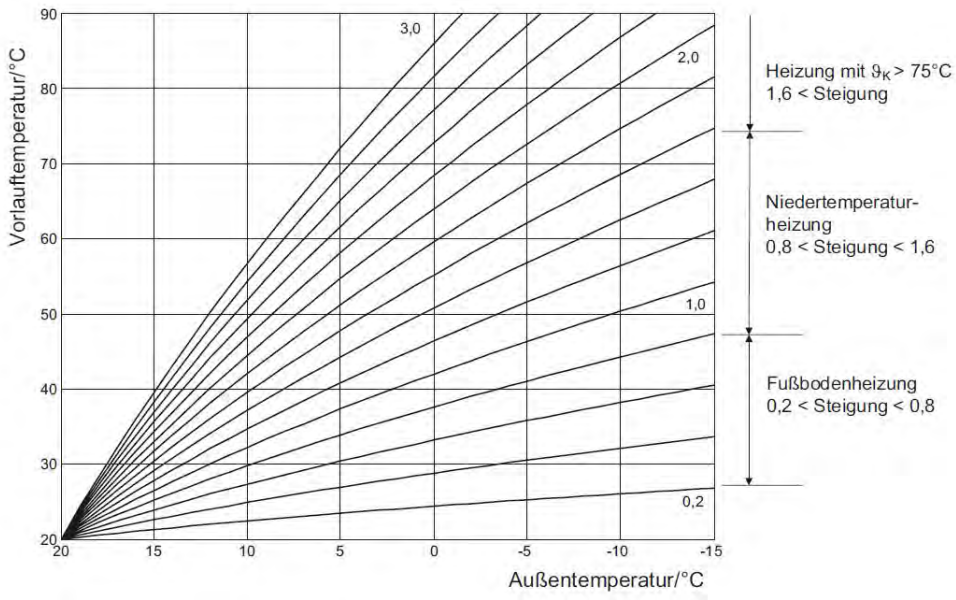


Abbildung 2.1: Heizkurven [LK20, S. 206]

Wie steil die Kurve verläuft, hängt vom Wärmebedarf der Räume und vom Heizungssystem ab. Bei Flächenheizungen, wie beispielsweise eine Fußbodenheizung, verläuft die Kurve flacher, siehe Abbildung 2.1. Durch die größere Heizfläche wird die gleiche Wärmemenge bei geringeren Vorlauftemperaturen als bei klassischen Heizkörpern (z.B. Rippen- oder Plattenheizkörper) übertragen. Berechnet werden Heizkurven anhand der von dem Gebäude nach außen abgegebenen Wärmeleistung Q_V, sowie der an die Innenräume abgegebenen Heizleistung Q_H. Im stationären Zustand sind die beiden Leistungen als gleich groß anzunehmen.

$$Q_V = f_1(T_R - T_A) \tag{2.1}$$

$$Q_H = f_2(T_V - T_R) \tag{2.2}$$

T_R ist dabei die Raumtemperatur, T_A die Außentemperatur und T_V die Temperatur im Vorlauf. Je genauer die Funktionen f_1 und f_2 definiert sind, desto komplizierter wird die Heizkurve. Im unkompliziertesten Fall hängen Q_H und Q_V linear von den Temperaturdifferenzen ab, und die Heizkurve kann mit einer Geraden beschrieben werden. In der Realität werden aber Nichtlinearitäten wie beispielsweise Wärmestrahlung mitberücksichtigt, wodurch sich eine Krümmung ergibt [Koc94, S. 7]. Auf die genaue Berechnung einer Heizkurve wird nicht näher eingegangen, da dies für den weiteren Verlauf der Arbeit nicht relevant ist.

Ein Nachteil der außentemperaturgeführten Regelung sind die vernachlässigten äußeren und inneren Fremdwärmeleistungen. Als solche gelten beispielsweise Wärmegewinne durch Sonneneinstrahlung, elektrische Geräte oder Personen, die Wärme an die Umgebung emittieren. Der Einfluss der inneren Fremdwärmeleistungen gewinnt infolge der stets verbesserten Wärmedämmung von Gebäuden zunehmend an Einfluss [Koc94, S. 7]. Die solaren Gewinne als äußere Fremdwärmeleistung werden durch eine mehrfache Verglasung der Fenster jedoch verringert [SW13, S. 205]. Als weiterer Nachteil bei dieser Art von Regelung sind die bei der Heizkurve üblicherweise eingeplanten Sicherheitsaufschläge zu nennen. So sind die Temperaturen für den Vorlauf meist zu hoch eingestellt, um jederzeit eine genügende Heizleistung sicherstellen zu können. Eine Rückmeldung darüber, welche Temperatur für die geforderte Heizleistung ausreichen würde, erfolgt bei dieser Regelung nicht.

Eine solche erhöhte Vorlauftemperatur führt zu Effizienz- und Wärmeverlusten und in der Folge zu einem erhöhten Energieverbrauch. Es wäre demnach sinnvoll, die Heizkurve an den tatsächlichen Wärmebedarf des Gebäudes durch Monitoring der Raumtemperaturen und der Ventilöffnungen anzupassen. Das Ergebnis einer Studie aus Großbritannien lässt die Vermutung zu, dass dies in der Realität mehrheitlich nicht geschieht [LSD05, S. 346]. Eine weitere Studie aus Dänemark kam zu dem Ergebnis, dass, nebst energieeffizienten Neubauten auch Gebäude, deren Baujahr im 20. Jahrhundert liegt, den größten Teil des Jahres über mit einer Vorlauftemperatur von unter 55 °C auskommen. Dabei kann eine eine Raumtemperatur von 21 °C gewährleistet werden. Die Studie basiert allerdings mehrheitlich auf theoretischen Annahmen und nicht auf der Auswertung von durchgeführten Messungen [ØS16, S. 382].

2.2 Bedarfsgeführte Regelung

Bei bedarfsgeführten Regelungen soll die Vorlauftemperatur nach dem aktuellen Wärmebedarf bestimmt werden. Um dies zu realisieren, müssen andere Parameter als nur die Außentemperatur herangezogen werden. Auf den aktuellen Wärmebedarf wirken weitere Größen, wie die zuvor genannten inneren und äußeren Fremdwärmeleistungen, ein. Im Folgenden werden zwei Ansätze vorgestellt, nach denen sich die Regelung, wie sie in dieser Arbeit umgesetzt wird, orientiert.

Ansatz 1: Raumtemperaturgeführte Regelung der Vorlauftemperatur
Die gebräuchlichste Umsetzung einer bedarfsgeführten Regelung ist die nach der Temperatur in einem Referenzraum. Je nachdem, welches Vorzeichen und wie groß die Differenz zwischen gemessener und gewünschter Raumtemperatur ist, wird die Vorlauftemperatur schrittweise erhöht oder gesenkt. Als Referenzraum sollte derjenige gewählt werden, dessen Wärmeverhalten typisch für die Räume im Gebäude oder im Heizungsstrang ist. In Wohngebäuden wird in der Regel das Wohnzimmer gewählt. In diesem Referenzraum sollen die Thermostate vollständig geöffnet sein. Ein Nachteil bei dieser Art von Regelung ist die Gefahr, dass einer oder mehrere Räume im Gebäude unterversorgt sein könnten, da der Wärmebedarf zeitweise oder kontinuierlich größer als der des gewählten Referenzraum ist. In einer Wohnung könnte das beispielsweise im Badezimmer auftreten, in welchem mit höheren Temperaturen für die Behaglichkeit gerechnet wird als im Wohnzimmer [LK20, S. 205]. Diese Art der bedarfsorientierten Regelung nach der Temperatur

in nur einem Referenzraum wird deshalb besonders in kleinen Wohneinheiten oder einem entsprechend gut gedämmten Gebäude empfohlen [Bos21]. In Abbildung 2.2 ist eine solche raumtemperaturgeführte Regelung dargestellt. Die Wärmequelle ist hier beispielhaft eine Kesselheizung, das Prinzip der Regelung gilt jedoch auch für andere Wärmequellen.

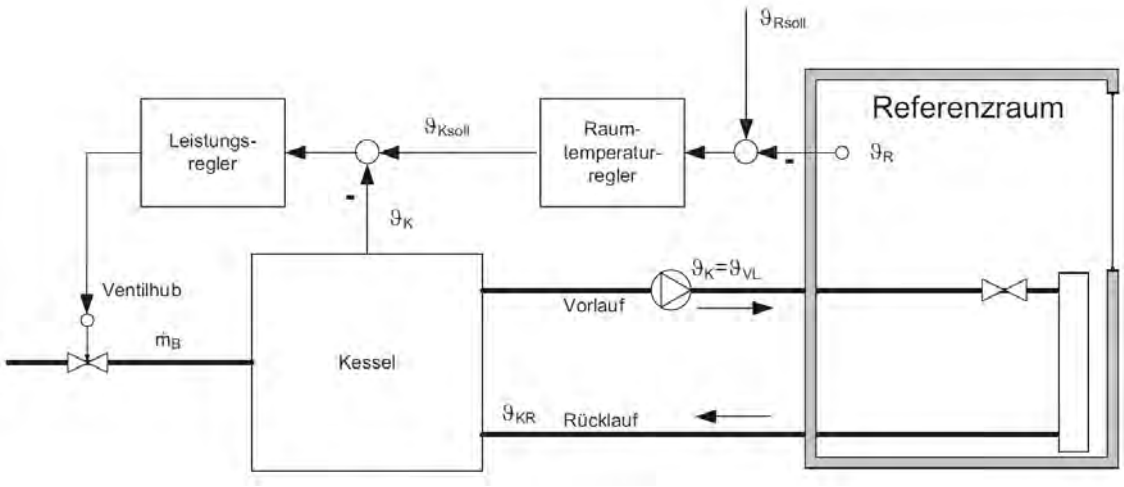


Abbildung 2.2: Regelung nach der Temperatur eines Referenzraumes[LK20, S. 205]

Ansatz 2: Anpassung der Heizkurve durch eine ventilöffnungsgradabhängige Regelung

Eine andere Möglichkeit, um den realen Wärmebedarf zu berücksichtigen, bietet eine Regelung nach den Ventilöffnungsgraden der Heizkörperthermostate (VÖG). Diese können ebenfalls als Maß für den Wärmebedarf in einzelnen Räumen herangezogen werden. Dabei wird unterschieden, ob mit sogenanntem vollständigen oder mit teilweisem Informationsverbund geregelt wird. Bei einem vollständigen Informationsverbund werden die Thermostate aller Räume im Gebäude oder Heizungsstrang miteinbezogen, bei einem teilweisen Informationsverbund entsprechend nur wenige oder gar nur einer. Die Regelung über die Außentemperatur, wie in Abschnitt 2.1 beschrieben, findet ohne Informationsverbund statt [LK20, S. 113]. Anders als bei der raumtemperaturgeführten Regelung wird bei diesem zweiten Ansatz die Vorlauftemperatur nicht gänzlich ohne die Berücksichtigung der Außentemperatur geregelt. Durch die Regelung wird die bestehende Heizkurve fortlaufend angepasst. Dafür wird stetig der Raum mit dem größten Ventilöffnungsgrad bestimmt. Die Heizkurve wird so lange korrigiert, bis das Ventil in diesem Raum zu 90 % geöffnet ist [LK20, S. 209].

3 Grundlagen

Dieses Kapitel beginnt mit einigen notwendigen Grundlagen der Regelungstechnik. Dabei werden die gängigsten Regler, wie sie häufig in der Gebäudeautomation verwendet werden und für diese Arbeit von Bedeutung sind, vorgestellt. Der zweite Abschnitt beschreibt ein Verfahren zum Reglerentwurf, welches in Kapitel 4 angewandt wird.

3.1 Aufbau und Komponenten einer Regelung

Unter einer Regelung wird die gezielte Beeinflussung einer Größe (Istwert) verstanden, um Abweichungen von einem gewünschten Wert (Sollwert) zu minimieren oder zu beseitigen. Kennzeichnend für eine Regelung ist der geschlossene Wirkungsablauf. Es erfolgt ein ständiger Abgleich mit dem Zustand des zu regelnden Systems [Man+18, S. 29]. Durch den geschlossenen Wirkungsablauf entsteht ein Regelkreis. In Abbildung 3.1 ist ein Standardregelkreis dargestellt. Er besteht im Wesentlichen aus einem Regler und einer Regelstrecke.

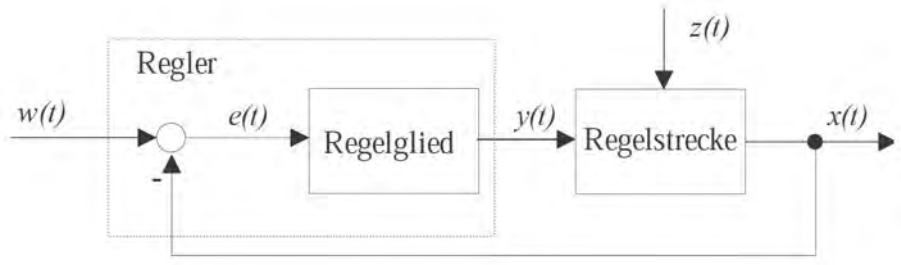


Abbildung 3.1: Grundregelkreis [Phi19, S. 3]

Die Regelstrecke beschreibt das zu steuernde oder zu beeinflussende System. Die Ausgangsgröße der Regelstrecke ist die Regelgröße $x(t)$, welche gemessen und entsprechend konstant gehalten oder verändert werden soll. Sie wird als Istwert rückgekoppelt und mit der Führungsgröße $w(t)$ verglichen, welche als gewünschter Sollwert dem Regelkreis

zugeführt wird. Auf die Regelstrecke können außerdem Störgrößen $z(t)$ einwirken, wie beispielsweise das Öffnen eines Fensters bei einer Raumtemperaturregelung.
Die Abweichung zwischen der Regel- und der Führungsgröße wird als Regeldifferenz $e(t)$ bezeichnet und ist die Eingangsgröße des Reglers. Der Regler errechnet aus der Regeldifferenz die passende Stellgröße $y(t)$. Die genaue Funktionsweise des Reglers unterscheidet sich je nach Reglertyp. Die Wahl des Reglers orientiert sich am Verhalten der Regelstrecke [Heß20, S. 89], siehe Abschnitt 3.2, Tabelle 3.1.

Eine spezielle Form des Standardregelkreises ist die sogenannte Kaskadenregelung. Diese Regelungsstruktur kommt bei der Automatisierung von gebäudetechnischen Anlagen häufig vor. Sie sorgt für ein verbessertes Verhalten gegenüber Störgrößen und beschleunigt die gesamte Regelung [LK20, S. 32]. Sie besteht aus einem Hauptregelkreis und einem oder mehrerer Folgeregelkreise. Der Ausgang des Führungsreglers wird zum Eingang des Folgereglers. Die Regeldifferenz des Folgeregelkreises wird mit einer Folgeregelgröße gebildet. In der unteren Abbildung 3.2 ist die Struktur einer Kaskadenregelung exemplarisch zu sehen. Die Folgeregelgröße und der Folgeregelkreis werden in der Abbildung 3.2 als Hilfsregelgröße und Hilfsregelkreis bezeichnet. Diese Begriffe werden im Folgenden nicht mehr verwendet.

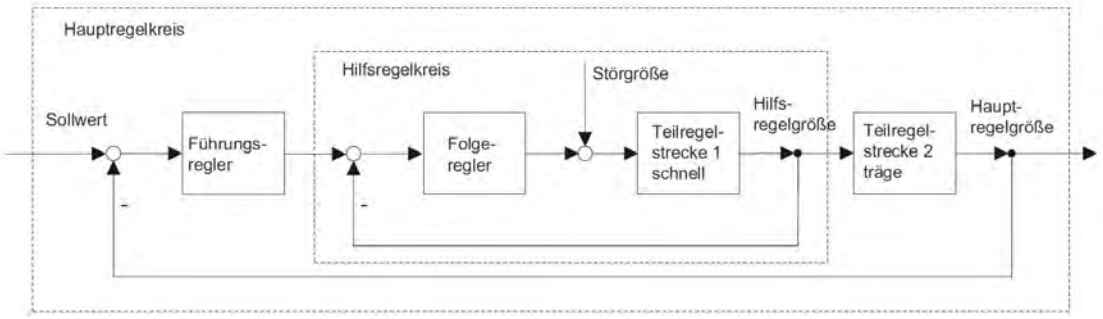


Abbildung 3.2: Exemplarischer Kaskadenregelkreis [LK20, S. 32]

Es werden nun zunächst die Grundformen der stetigen Regler aufgeführt. Stetig bedeutet, dass für die Stellgröße beliebige Zwischengrößen im Stellbereich vorgegeben werden können. Aus diesen Grundformen können weitere Regler zusammengesetzt werden, wobei die positiven Eigenschaften der einzelnen Grundformen jeweils miteinander kombiniert werden. Als wichtigste Standardregler sind hier vor allem P- und PI-Regler zu nennen. Diese kommen in der Gebäudeautomation überwiegend zum Einsatz. PID-Regler werden

nur in Ausnahmefällen eingesetzt, da diese eine genaue Kenntnis über das dynamische Speicherverhalten der Regelstrecke voraussetzen und der Inbetriebnahmeaufwand hoch ist [LK20, S. 55–64]. Aufgrunddessen wird im Folgenden auf die Erklärung des D-Anteils verzichtet.

Proportional-Anteil (P-Regler)
Beim P-Regler erfolgt eine Verstärkung der Regeldifferenz, welche durch den Proportionalitätsbeiwert (P-Beiwert) K_{PR} beschrieben wird. Der Nachteil des P-Reglers ist die bleibende Regeldifferenz. Wäre die Regeldifferenz Null, so würde auch die Stellgröße zu Null werden, wie an Gleichung 3.1 zu erkennen ist. Der Vorteil des P-Reglers ist seine schnelle Arbeitsweise.

$$y(t) = K_{PR} * e(t) \tag{3.1}$$

Integral-Anteil (I-Regler)
Bei einem integrierenden Regler wird die Regelabweichung zu einer Ausgangsgröße aufsummiert. Die Stellgröße ist proportional zum Integral der Regeldifferenz. Die Verstärkung erfolgt durch den Integrierbeiwert $K_I = \frac{K_{PR}}{T_I}$. Der I-Regler arbeitet langsam, aber exakt. Es bleibt keine Regelabweichung zurück.

$$y(t) = K_I * \int e(t)d(t) \tag{3.2}$$

3-Punkt-Regler
Ein weiterer wichtiger Regler in der Gebäudetechnik ist der 3-Punkt-Regler, welcher den schaltenden, nichtstetigen Reglern zugeordnet wird. Der 3-Punkt Regler wird an motorischen Stellantrieben, wie zum Beispiel für Klappen und Ventile, eingesetzt. Wird der Motor angesteuert, so ändert sich der Drehwinkel der Klappe oder der Hub des Ventils. Der Regler kann über den gesamten Wertebereich drei Ausgangszustände für die Stellgröße annehmen [LK20, S. 52–54]:

- 1 = Motor Rechtslauf = Ventil auf
- 0 = Motor Halt
- -1 = Motor Linkslauf = Ventil zu

Der 3-Punkt-Regler kann für die Regelung der Vorlauftemperatur zum Einsatz kommen. In Abbildung 3.3 ist ein solcher Aufbau zu sehen. Auch hier wird als Wärmequelle beispielhaft ein Heizungskessel genannt. Das Prinzip dieses Aufbaus gilt jedoch auch für andere Wärmequellen an. Um die gewünschte Vorlauftemperatur zu erhalten, wird mithilfe eines 3-Wege-Ventils die benötigte Menge an Wasser aus dem kälteren Rücklauf dem Vorlauf beigemischt.

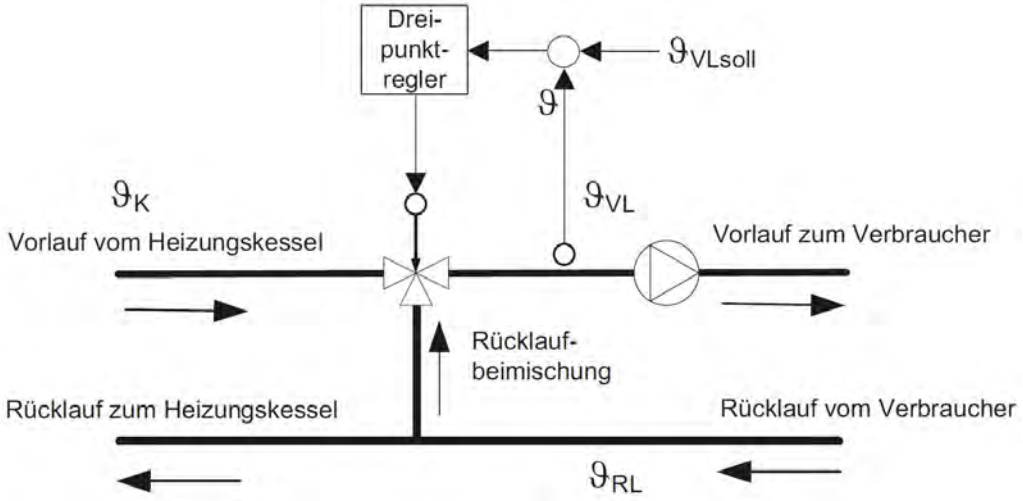


Abbildung 3.3: Regelung der Vorlauftemperatur mit einem 3-Punkt-Regler [LK20, S. 54]

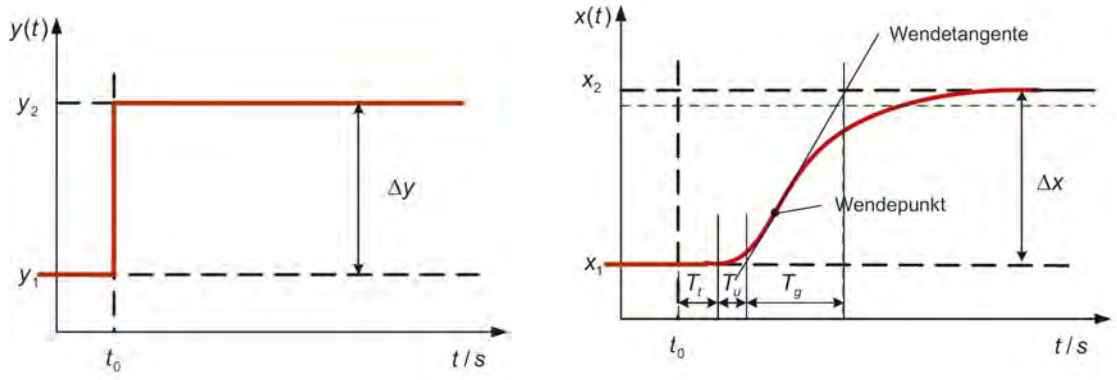
3.2 Vorgehen Reglerentwurf

Ist das Verhalten der Regelstrecke bekannt, so kann aus der Regelstrecke der passende Regler gefunden werden. Es gibt Empfehlungen, welcher Regler zur vorhandenen Regelstrecke passt. Diese Empfehlungen und einige Beispiele für die Regelstrecken sind in Tabelle 3.1 aufgeführt.

Tabelle 3.1: Zuordnung zwischen Regelstrecke und Regler [LK20, S. 69]

Regelstrecke	Beispiel	Regler
P	Ventil, Massestrom, Hebel	I
I	Füllhöhe, Drehwinkel	P
PT1	Temperatur in Klimaanlagen, Druck	PI
PT2	Temperatur an Wärmeübertragern, Raumtemperatur	PI (PID)

Eine Möglichkeit für die Vorgehensweise bei einem Reglerentwurf ist die graphische Auswertung einer Sprungantwort. Dafür wird ein Sprung auf das System gebracht und der Verlauf im Anschluss analysiert. Die Reglerparameter können aus dem Verlauf graphisch bestimmt werden. In Abbildung 3.4 ist ein Beispiel für ein solches Verfahren dargestellt. Es ist das Wendetangenten-Verfahren und kann für Strecken, die ein PTn-Verhalten aufweisen, angewendet werden.



(a) (b)

Abbildung 3.4: Sprungantwort nach [Heß20, S. 92]

Zunächst werden die Verzugszeit T_u und die Ausgleichszeit T_g durch das Anlegen einer Wendetangente bestimmt. Anhand des Verhältnisses von T_g zu T_u (und der Totzeit T_t, falls vorhanden) lässt sich eine Aussage über die Regelbarkeit eines Systems treffen. Je größer die Verzugszeit im Vergleich zur Ausgleichszeit ist, desto schwieriger gestaltet sich die Regelung des Systems [Bög11, R18-R19]. Eine große Ausgleichszeit sorgt dafür, dass der negative Effekt einer vorhandenen Totzeit oder einer großen Verzugszeit auf die Regelbarkeit nicht mehr ins Gewicht fällt.

$$R = \frac{Ausgleichszeit}{Verzugszeit} = \frac{T_g}{T_u + T_t} \tag{3.3}$$

$R > 5$	gut regelbar
$R = 2,5...5$	mäßig regelbar
$R = 1,2...2,5$	schlecht regelbar
$R < 1,2$	sehr schlecht regelbar

Lässt das Verhältnis die Weiterarbeit zu, so können die für die Reglerauswahl benötigten Parameter aus der Sprungantwort graphisch ausgearbeitet werden. Die Verzugs- und Anregelzeit können direkt abgelesen werden. Zur Berechnung des Verstärkungsfaktors gilt die folgende Gleichung [Heß20, S. 91]:

$$K_{PS} = \frac{\Delta x}{\Delta y} \tag{3.4}$$

Sind alle Parameter bekannt, kann die Reglereinstellung nach Erfahrungswerten erfolgen. Für die Umsetzung kann zum Beispiel das Verfahren nach Ziegler/Nichols angewandt werden. Allerdings sind dies nur grobe Näherungswerte. Deshalb sollten mehrere Messungen durchgeführt und die Reglerparameter weiter an den vorhandenen Betriebspunkt angepasst werden [LK20, S. 69].

Tabelle 3.2: Einstellregeln nach Ziegler/Nicols [LK20, S. 69]

Reglertypen	Reglereinstellwerte	
	K_{PR}	T_I
P-Regler	$\frac{1}{K_{PS}} * \frac{T_g}{T_u}$	-
PI-Regler	$\frac{0,9}{K_{PS}} * \frac{T_g}{T_u}$	$3,33 * T_u$

4 Methodik

Im folgenden Kapitel wird zunächst das Testgebäude beschrieben. Es sollen die technischen Rahmenbedingungen geklärt werden. Danach wird der eigene Ansatz für die Regelung vorgestellt. Zuletzt wird das Vorgehen für den Reglerentwurf, wie es in Kapitel 3, Abschnitt 3.2 beschrieben wurde, im Testgebäude angewandt.

4.1 Beschreibung Testgebäude

Das Testgebäude ist ein Kultur- und Veranstaltungszentrum. Gebaut wurde es Mitte der 1980er Jahre, 2018 wurde es im Zuge von Sanierungsarbeiten an das örtliche Wärmenetz angeschlossen. Die folgenden Abschnitte widmen sich dem technischen Aufbau beziehungsweise den Rahmenbedingungen im Testgebäude.

4.1.1 Aufbau der Heizkreise

Die Wärmeversorgung besteht im Wesentlichen aus den Heizkreisen: Süd (1), Fußbodenheizung (2), Verwaltung (3), Nord (4) und Keller/Restaurant (5). Im Anhang A.1 ist ein aktuelles R&I-Fließschema der fünf Heizkreise, sowie den Lüftungsanlagen und der Wärmeübergabestation (WÜST) zu finden. Die Vorlauftemperaturen der Heizkreise werden momentan mit einer gleitenden Betriebsweise mit der Außentemperatur als Führungsgröße geregelt. Jeder Heizkreis, so wie die WÜST, besitzt je eine individuelle Heizkurve. In dieser Arbeit wird ausschließlich Heizkreis Nord betrachtet. Dieser wurde in der vorangegangenen Studienarbeit für die Wahl des Referenzraumes bestimmt, da sich in diesem Heizstrang die meisten Seminar- und Büroräume befinden. Die Räume weisen daher ein ähnliches Nutzerverhalten auf. Dies kann bei einer bedarfsgeführten Regelung von Vorteil sein. Von der Wahl des Referenzraumes wurden alle Räume ausgeschlossen, die als Lager oder Treppenhäuser genutzt werden. Es wurde davon ausgegangen, dass diese Räume nicht den größten Wärmebedarf aufweisen, da die Aufenthaltsdauer von Personen in

jenen Räumen begrenzt und somit kein behagliches Raumklima vonnöten ist. Als Behaglichkeitskriterium für die Raumtemperatur gilt die Toleranzgrenze zwischen 20 °C und 26 °C [Fit13, S. 3]. Der zweite Parameter für ein behagliches Raumklima ist die Luftfeuchtigkeit. Diese wird im Rahmen dieser Arbeit nicht berücksichtigt, da kein Einfluss auf die Regelung erfolgen kann. Aufgrund fehlender Messdatensätze wurden die Räume 0.11a und 1.17 ebenfalls ausgeschlossen.

Tabelle 4.1: Raumliste Heizkreis Nord

Geschoss	Nummer	Raumnutzung	Fläche [m^2]	Anzahl HKT
EG	0.09a	Eingang	13,49	1
EG	0.09b	Eingang	17,41	2
EG	0.11a	Schulung	9,57	1
EG	0.11	Schulung	37,33	2
EG	0.12	Schulung	42,61	3
EG	0.13	Schulung	56,72	4
EG	0.14	Teeküche	9,9	1
EG	0.15	Büro	25,17	3
EG	0.18	Büro	19,69	2
EG	0.21	Büro	22,98	2
EG	0.29	Flur	11,08	1
EG	0.32	Büro	14,01	1
EG	0.38	Treppenhaus	13,33	1
EG	0.40	Treppenhaus	16,56	1
OG	1.10	Schulung	52,46	4
OG	1.11	Schulung	46,97	3
OG	1.12	Schulung	65,6	4
OG	1.13	Schulung	44,81	2
OG	1.14	Lager	9,12	1
OG	1.17	Büro	19,68	2
OG	1.18	Tanzen	85,07	4
OG	1.21	Schulung	46,23	4
OG	1.22	Lager	9,73	1
OG	1.23	Putzraum	3,76	1
OG	1.24	Treppenhaus	21,39	2

In Tabelle 4.1 sind die Räume des Heizkreises Nord aufgeführt. Die fünf «Schlechträume», die in Abschnitt 1.2 erwähnt wurden, sind die Folgenden: 0.11, 0.12, 0.32, 1.18 und 1.21. Die Räume 0.11, 0.12 und 1.21 sind Schulungsräume, Raum 0.32 wird als Büro genutzt und Raum 1.18 als Tanzraum. Als Referenzraum wurde Raum 0.11 bestimmt. Dieser wies im Schnitt die höchsten Ventilöffnungsgrade bei verschiedenen Solltemperaturen auf, wie in Abschnitt 1.2 beschrieben. Weiter sind die Anzahl der Heizkörperthermostate (HKT),

die sich in den jeweiligen Räumen befinden, in der Tabelle aufgeführt. Die Grundrisse vom Erd- und Obergeschoss sind in Anhang A.2 zu finden, die Räume des Heizkreises Nord sind blau markiert und beschriftet.

4.1.2 Datenübertragung, Messtechnik und Visualisierung

In diesem Abschnitt sollen die für die zu entwickelnde Regelung wichtigen Komponenten zum Abrufen der Daten beschrieben werden.

Thermostate

Die relevanten Räume des Testobjektes wurden im Rahmen des Projektes «Smart Pro Heat» mit elektronischen Heizkörperthermostaten der Firma Fourdeg ausgestattet. Die Heizkörperthermostate geben den Ventilöffnungsgrad für die geforderte Raumtemperatur vor. Sie verbinden sich alle 15-60 Minuten per WiFi mit einem separaten WLAN-Netzwerk und schicken die Messwerte für die Ventilöffnung und die Raumtemperatur an einen externen Server des Herstellers. Zeitgleich werden mögliche Sollwertänderungen abgefragt. Die Thermostate können entweder manuell am Gerät, über eine webbasierte Oberfläche oder über eine Programmierschnittstelle (API) angepasst werden. Da ein Abstand von bis zu 60 Minuten zwischen den Messwerten für das Aufzeichnen einer Sprungantwort zu ungenau ist, wurden für den Zeitraum dieser Arbeit Thermostate der Marke HomeMaticIP im Referenzraum verbaut. Diese können dank eines eigenen Funkprotokolls ihre Messwerte alle 3-4 Minuten senden. Auch diese Thermostate können entweder manuell am Gerät, über eine webbasierte Oberfläche oder eine API verstellt werden. Letzteres ermöglicht, über einen in Python umgesetzten Service, die zyklische Archivierung der Messdaten auf einer Zeitreihen-Datenbank.

WÜST

Für den zu entwerfenden Regler ist vor allem der Wert für die Vorlauftemperatur im Sekundärkreislauf an der WÜST entscheidend. Sie begrenzt die maximal möglichen Temperaturen der fünf Heizkreise und somit auch die des Reglers. Für die Auswertung ist des Weiteren die Höhe der Rücklauftemperaturen und die Werte des Außentemperaturfühlers von Bedeutung, welche ebenfalls hierfür aufgezeichnet werden. Alle Daten für die Auswertung können neben den Weboberflächen des Herstellers, beziehungsweise des Wärmenetzbetreibers, auch über die lokalen Webapplikationen «Grafana» oder «Chronograf» auf dem für das Projekt installiertem Datenbankserver aufgerufen und visualisiert werden.

Zur besseren Übersicht für die im Testgebäude verbaute Messtechnik und die entsprechende Datenübertragung dient die untenstehende Abbildung 4.1. Der zentrale Knotenpunkt ist die gelb eingekreiste Zeitreihen-Datenbank «InfluxDB» der HAW Hamburg. Hier werden die Messdaten systematisch nach Messort und -größe gespeichert und können nach demselben Schema wieder abgerufen werden. Diejenigen Komponenten, welche für die Umsetzung der bedarfsgeführten Regelung entscheidend sind, sind in der Grafik grau gefärbt. Unten rechts befindet sich der «Regler», der in der Programmiersprache Python umgesetzt werden soll. Das Programm des Reglers läuft auf einem externen (nicht im Testgebäude stationierten) Computer und fragt über eine VPN-Verbindung aus der InfluxDB die Messdaten der oben genannten Thermostate (Fourdeg und HomeMatic) und diejenigen der WÜST ab.

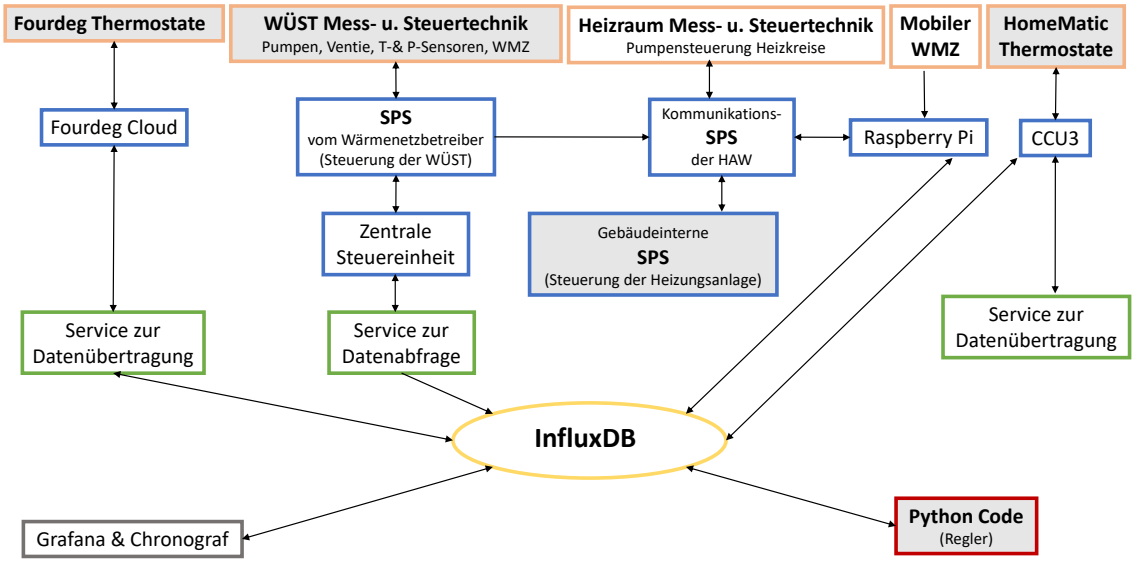


Abbildung 4.1: Kommunikationstechnik und Datenübertragung im Testgebäude

Der Programmcode des Reglers liefert als Ergebnis einen spezifischen Sollwert für die Vorlauftemperatur, der wiederum in der InfluxDB gespeichert wird. Über das Raspberry Pi und einer zwischengeschalteten speicherprogrammierbaren Steuerung (SPS) wird der Sollwert an die in der Grafik grau gefärbte SPS weitergegeben. Hier erfolgt die Regelung der Vorlauftemperatur über einen 3-Punkt-Regler, wie in Abbildung 3.3 dargestellt.

4.2 Eigener Ansatz zur bedarfsgeführten Regelung

Für die Umsetzung der Regelung soll eine Kombination aus den beiden Verfahren aus Abschnitt 2.2 gewählt werden. Die Vorlauftemperatur soll bedarfsgeführt anhand eines Referenzraumes, dessen Wahl in [Hee20] und im Abschnitt 1.2 beschrieben wurde, geregelt werden. Dies entspricht dem ersten Ansatz. Die Wahl fällt allerdings nicht auf den Raum mit dem typischen Wärmebedarf, sondern auf einen «Schlechtraum». Dadurch soll vermieden werden, dass es Räume gibt, die unterversorgt werden. Des Weiteren soll nicht nach der Raumtemperatur, sondern nach den Ventilöffnungsgraden geregelt werden, wie es der zweite Ansatz vorsieht. Grund dafür ist die Unsicherheit über die internen Regelungsfunktionen der verwendeten elektronischen Thermostate. Eine Anpassung der Heizkurve innerhalb des Regelvorgangs, wie es im zweiten Ansatz umgesetzt wurde, ist nicht vorgesehen. Mit den neu gewonnenen Messwerten für die Vorlauftemperatur soll lediglich für die Auswertung eine neue Heizkurve erstellt werden (die Außentemperatur wird während den entsprechenden Messzeiträumen ebenfalls aufgezeichnet), um diese mit der bestehenden Heizkurve vergleichen zu können. Die Heizkurve wird mit den Messdaten der Vorlauftemperatur und den zugehörigen Werten der Außentemperatur als X-Y Plot erstellt.

4.3 Reglerentwurf für das Testgebäude

Für den Reglerentwurf im Testgebäude wurde zunächst der Regelkreis definiert, welcher im folgenden Abschnitt erläutert wird. Es folgt die Auswertung der Sprungantworten der Regelstrecke nach dem Verfahren aus Abschnitt 3.2.

4.3.1 Regelkreis

Die Abbildung 4.2 zeigt den Regelkreis für die zu entwickelnde Regelung im Testgebäude. Es ist ein einfach vermaschter Kaskadenregelkreis mit einem Führungs- und einem Folgeregelkreis. Die Legende für die Komponenten des Regelkreises ist unter dem Bild aufgeführt. Die Komponenten für den Folgeregelkreis sind zur Übersichtlichkeit in Bild und Text unterstrichen, die des Führungsregelkreises breit markiert.

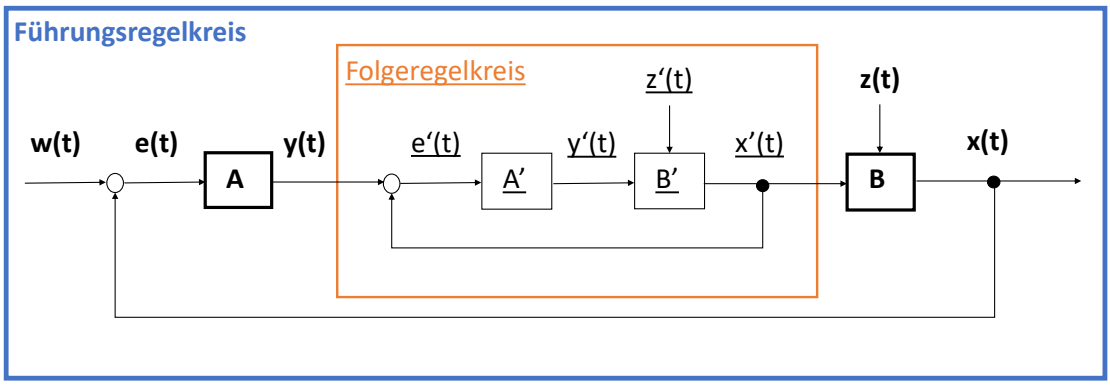


Abbildung 4.2: Regelkreis zur Umsetzung der bedarfsgeführten Regelung

Führungsregelkreis

- **w(t):** VÖG in % (Soll)
- **x(t):** VÖG in % (Ist)
- **e(t):** VÖG (Soll) – VÖG (Ist)
- **y(t):** VLT in °C (Soll)
- **z(t):** Wärmequellen/-senken
- **A:** Führungsregler VÖG
- **B:** Teilregelstrecke VÖG

Folgeregelkreis

- x'(t): VLT in °C (Ist)
- e'(t): VLT (Soll) – VLT (Ist)
- y'(t): Stellung des 3-Wege-Ventils
- z'(t): T am Heizkreisverteiler
- A': Folgeregler VLT
- B': Teilregelstrecke VLT

Der **Führungsregelkreis** beschreibt die Regelung der Vorlauftemperatur nach den Thermostatöffnungen. Dementsprechend beschreibt die Regelstrecke **B** das Verhalten der Thermostate. Der Sollwert der Ventilöffnungen geht als Führungsgröße **w(t)** in den Regelkreis ein. Durch Subtraktion der aktuellen Ventilöffnungen **x(t)** im Referenzraum ergibt sich für den Führungsregelkreis die Regelabweichung **e(t)**. Diese Regelabweichung wird dem Regler (**A**), der in dieser Arbeit umgesetzt werden soll, zugeführt. Der Regler soll eine passende Vorlauftemperatur errechnen, um die geforderte Ventilöffnung zu erreichen. Die berechnete Vorlauftemperatur wird als Sollwert, beziehungsweise als Stellgröße **y(t)**, an den Folgeregelkreis weitergegeben.
Der Folgeregelkreis beschreibt die eigentliche Regelung der Vorlauftemperatur durch das 3-Wege-Ventil, dessen Verhalten durch die Regelstrecke B' beschrieben wird. Die Ventil-

stellung ist dabei die Stellgröße y'(t). Die aktuell gemessene Vorlauftemperatur beschreibt die Regelgröße x'(t). Diese wird von **y(t)** abgezogen, um die Regledifferenz e'(t) für den Folgeregelkreis zu erhalten, welche dem 3-Punkt-Regler (A') zugeführt wird. Die Regelung des Folgeregelkreises existiert bereits und soll im Rahmen dieser Arbeit nicht verändert werden. Sie wird lediglich beeinflusst durch die Vorgabe eines spezifischen Sollwerts aus dem Führungsregelkreis, so dass der Sollwert nun nicht mehr aus der Heizkurve entnommen wird.

Nachdem der Regelkreis für die neue Regelung definiert wurde, gilt es nun das Verfahren zum Reglerentwurf anzuwenden. Dafür werden im folgenden Abschnitt Sprungantworten aufgenommen und mit dem Wendetangenten-Verfahren die Reglerparameter bestimmt.

4.3.2 Aufnahme und Auswertung der Sprungantworten

Um das Verhalten der Regelstrecke analysieren zu können soll nun manuell der Sollwert für die Vorlauftemperatur sprunghaft erhöht werden. Dabei soll die Reaktion der Ventile im Referenzraum 0.11 beobachtet werden. In diesem Raum befinden sich zwei Heizkörperthermostate. In diesem Abschnitt erfolgt die Auswertung beispielhaft für den linken Thermostat, da beide ähnliche Reaktionen zeigen. Die Diagramme für den rechten Thermostat sind im Anhang A.3 zu finden. In der folgenden Grafik 4.3 sind die Ergebnisse des Sprungs zu erkennen. Die Vorlauftemperatur (orange bzw. violett) wurde von 70 °C auf 55 °C abgesenkt.

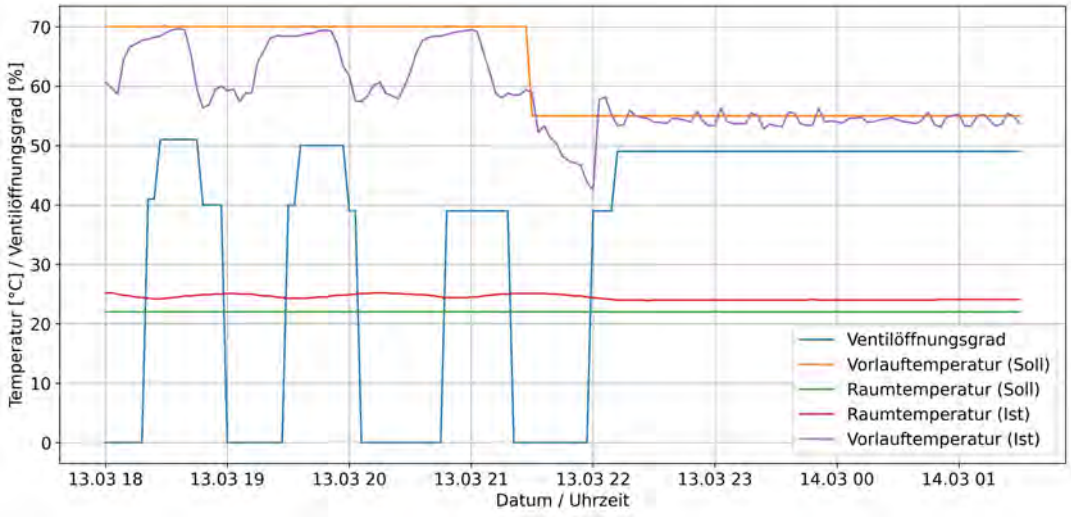


Abbildung 4.3: Sprungantwort Thermostat links

Zunächst wird deutlich, dass die Thermostate von HomeMaticIP etwa zwei Grad höher als die geforderte Raumtemperatur (grün bzw. rot) regeln. Es ist außerdem zu erkennen, dass sich das Ventil (blau) vor dem Sprung abwechselnd öffnet und wieder schließt, um die gewünschte Raumtemperatur zu erreichen. Durch das Absenken der Vorlauftemperatur bleibt das Ventil kontinuierlich offen und die Raumtemperatur konstanter.

Zur besseren Übersicht bei der Auswertung sind im folgenden Diagramm in der Abbildung 4.4 nur der Graph des Ventils und derjenige des Sollwerts für die Vorlauftemperatur zu sehen.

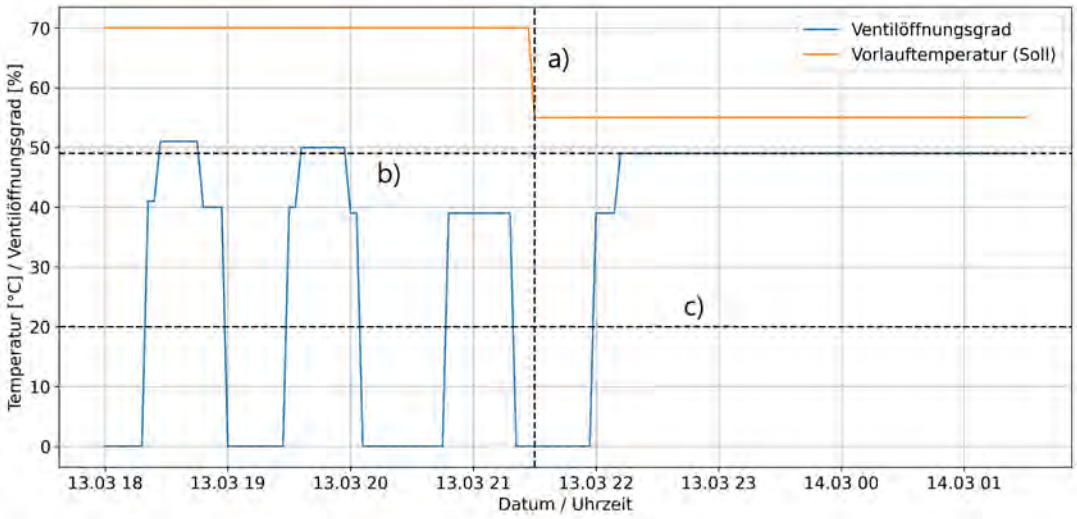


Abbildung 4.4: Teilauswertung Sprungantwort Thermostat links

Die gestrichelte Line a) beschreibt das Ende des Sprungs und somit den Anfang der Totzeit Tt. Die gestrichelte Linie b) ist eine Verlängerung des Endwertes, in welchem sich die Thermostate eingeschwungen haben. Bei Linie c) handelt es sich um einen Mittelwert für den Ausgangswert der Thermostatöffnungen, da dieser durch das wiederholte Öffnen und Schließen nicht konstant ist. Die Sprungantwort der Thermostate lässt sich mit einem PT1-Verhalten mit Totzeit beschreiben. Anders als beim PTn-Verhalten kommt es hier zu keiner eigentlichen Verzugszeit, da der Graph sprunghaft ansteigt und nicht langsam wie in Abbildung 3.4. Möglich ist auch, dass keine Verzugszeit erkennbar ist durch die Grobheit der Messdaten. Die Verzugszeit wäre dann jedoch vernachlässigbar klein. Bei einer PT1-Strecke ist es ebenfalls möglich, das Wendetangentenverfahren anzuwenden. In diesem speziellen Fall ist die Verzugszeit gleich der Totzeit [Bög11, R19].

Als nächster Schritt soll die Wendetangente eingezeichnet werden. Dafür wurde im folgenden Diagramm in Abbildung 4.5 das Zeitfenster auf den für die Auswertung wesentlichen Bereich verkleinert.

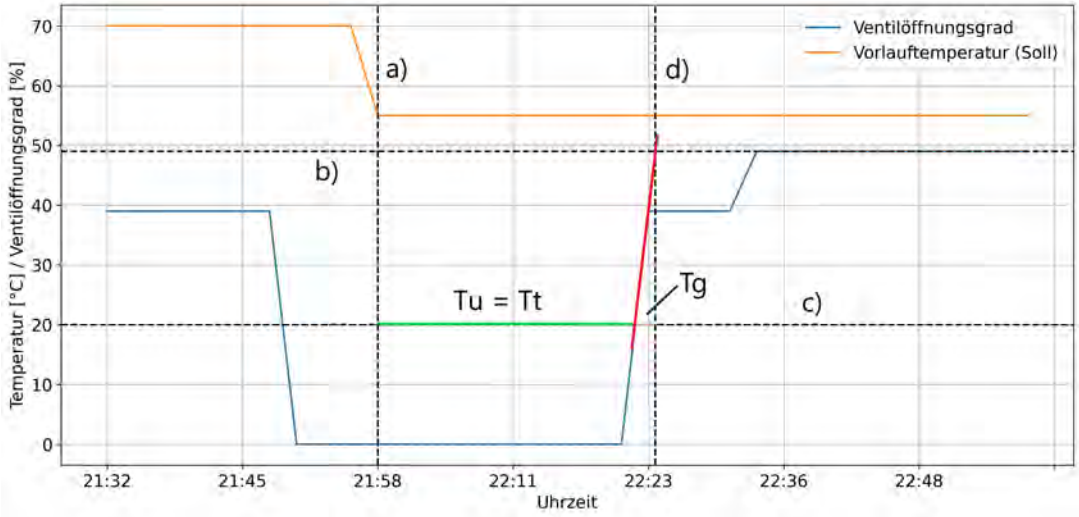


Abbildung 4.5: Auswertung Sprungantwort Thermostat links

Da die Sprungantwort der Thermostate ohne Wendepunkt ansteigt, gibt es keine eigentliche Wendetangente. Die Wendetangente ist in diesem Fall gleich der Steigung des Graphen. Im Fall der aufgenommenen Sprungantwort ist dies die erste Steigung des Graphen, die nach dem Sprung erfolgt. Die zweite Steigung, die nach dem ersten Abflachen des Graphen auftritt, wird nicht berücksichtigt. Wie bereits beschrieben, ist im Fall eines PT1-Verhaltens T_u gleich T_t (grün markiert und beschriftet). Durch den Schnittpunkt der Tangente (rot) mit der Linie c) wird eine neue Linie d) gezogen, welche zusammen mit der Tangente die Ausgleichszeit T_g begrenzt (rosa markiert und beschriftet). Wird die Formel 3.3 aus Kapitel 3 für die Auswertung hinzugezogen, so lässt sich bereits ohne genaue Zahlen erkennen, dass die Regelung des Systems schwierig wird. Für eine gute Regelbarkeit müsste T_g in jedem Fall größer als T_u sein. In dieser Sprungantwort ist T_g aber deutlich kleiner als T_u.
Da die Sprungantwort durch ein PT1- oder ein schwer erkennbares PTn-Verhalten beschrieben werden kann, wäre nach Tabelle 3.1 der dazu passende Regler ein PI-Regler. Der Einsatz eines PID-Reglers wird, wie in Kapitel 3 beschrieben, nicht in Erwägung gezogen. Ein PI-Regler wird in diesem Fall jedoch nicht das gewünschte Ergebnis bringen, da dieser die vorhandene Totzeit nicht berücksichtigen kann. Er würde somit bereits während der Totzeit versuchen, die Regeldifferenz zu minimieren. In der Literatur wird in diesem Fall

empfohlen zu überprüfen, ob sich die Totzeiten durch technische Optimierung verringern lassen [Bög11, R19]. Im Rahmen dieser Arbeit ist eine solche Optimierung nicht möglich. Die langen Totzeiten entstehen unter anderem durch die thermische Trägheit des Heizkreises. Es soll im Zuge dessen ein eigenes Programm geschrieben und umgesetzt werden, welches die langen Totzeiten der Thermostate berücksichtigt und dennoch die Vorlauftemperatur nach dem Bedarf regeln lässt. Für die Umsetzung dieses Programms kann es dennoch von Vorteil sein, das K_{PS} der Sprungantwort zu berechnen. Durch diesen Wert kann eine Abschätzung darüber erfolgen, um wieviel die Vorlauftemperatur gesenkt werden muss, um eine gewünschte Thermostatöffnung zu erreichen. Für die Sprungantwort wurde die Vorlauftemperatur, wie bereits erwähnt, um 15 °C gesenkt. Durch die Linien b) und c) aus Diagramm 4.5 lässt sich eine Erhöhung der Thermostatöffnung von knapp 30 % ablesen. Mit der Formel 3.4 erhalten wir:

$$K_{PS} = \frac{\Delta x}{\Delta y} = \frac{30}{15} = 2 \tag{4.1}$$

Das bedeutet, dass wir pro 1 °C Verlust in der Vorlauftemperatur 2 % Ventilöffnung erhalten. Die Dauer vom Zeitpunkt des Sprungs in der Vorlauftemperatur bis zum Ausregeln der Ventilöffnungsgrade beträgt in etwa 30 Minuten. Dies kann in Abbildung 4.4 an der Zeitachse abgelesen werden. Die Auswertung der Messdaten des rechten Thermostaten liefert ein identisches Ergebnis. Die Diagramme der Auswertung sind in Anhang A.3 zu finden. Im nächsten Kapitel erfolgt die Umsetzung des Programms und dessen Implementierung.

5 Umsetzung der Regelung

In diesem Kapitel wird das entwickelte Programm für den Regler vorgestellt. Anhand eines Flussdiagramms soll der Aufbau des Reglers und der Ablauf des Regelalgorithmus erläutert werden. Im zweiten Abschnitt erfolgt die Implementierung. Hier werden die ersten Testphasen der neuen Regelung, sowie die dabei auftretenden Komplikationen beschrieben. Am Ende dieses Kapitels soll die Regelung soweit optimiert sein, dass diese ohne Komplikationen läuft. Wichtigstes Kriterium ist hierbei, dass alle Räume im Heizkreis ausreichend mit Wärme versorgt werden.

5.1 Programm

In diesem ersten Abschnitt wird der Code für den Regler anhand des Flussdiagrammes in Abbildung 5.1 erläutert. Der eigentliche Python-Code ist in Anhang A.4.1 aufgeführt. Zu Beginn des Programms werden zunächst die aktuellen Datenpunkte aufgerufen. Die grün, gelb, blau und rot markierten Blöcke symbolisieren «while-Schleifen». Diese werden so lange durchlaufen, bis die Bedingung beim Eintritt in die Schleife nicht mehr zutrifft. Die erste while-Schleife (grün markiert) wird aktiv, falls die beiden Thermostate im Referenzraum zu 80 % oder mehr geöffnet sind und die Raumtemperatur zeitgleich unter dem Sollwert liegt. In diesem Fall wird angenommen, dass die Vorlauftemperatur zu niedrig ist, um die gewünschte Raumtemperatur halten zu können. Die Vorlauftemperatur soll dann um 2 °C erhöht werden und das Programm bereits nach 30 Minuten erneut die aktuellen Datenpunkte abrufen. Nach den Ergebnissen der Sprungantwort kann damit stündlich eine Verringerung des Ventilöffnungsgrads um 8 % erreicht werden. Die Schleife wird solange durchlaufen, bis die Raumtemperatur wieder den Sollwert erreicht. Beträgt der Unterschied zwischen Soll- und Isttemperatur mehr als 3 °C, so wird davon ausgegangen, dass in diesem Raum ein Fenster geöffnet wurde. In diesem Fall soll das Programm eine Stunde pausieren, um die Vorlauftemperatur nicht unnötig in die Höhe zu treiben.

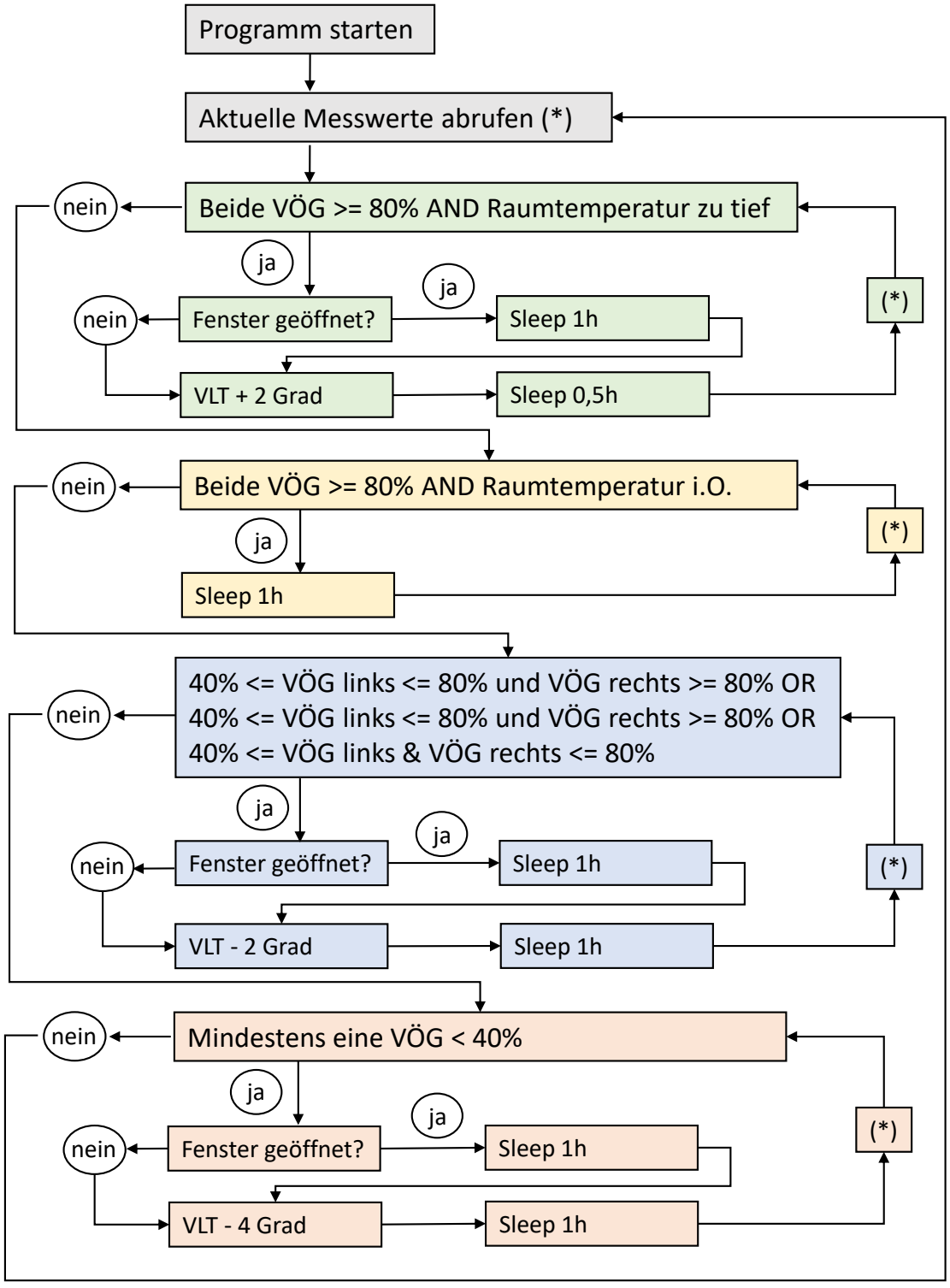


Abbildung 5.1: Flussdiagramm zur Programmerläuterung

Nachfolgend ist beispielhaft ein Auszug aus dem Code aufgeführt, der die erste while-Schleife beinhaltet:

```
    # Schleife 1
    i = 2
    while (x_r >= 80) and (x_l >= 80) and (dt > 0):
        if dt > 3:
            print("Schlafe eine Stunde wegen geoeffnetem Fenster")
            time.sleep(1800)
        vlt_soll_neu = vlt_soll_haw + i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print("Schlafe in Schleife 1 fuer eine Stunde.", time.strftime("
                                                %d.%m.%Y %H:%M:%S"))
        time.sleep(3600)
        print("Neue Werte:")
        x_r, x_l, temp_soll, temp_ist, vlt_ist, vlt_soll_haw, vlt_sek =
                                                get_data()
        dt = temp_soll - temp_ist
        print("dT:", dt)
```

Die zweite while-Schleife (gelb markiert) beschreibt den Fall für den gewünschten Zustand. Die Thermostate sind mindestens zu 80 % geöffnet und die Raumtemperatur entspricht dem Sollwert oder höher. In diesem Fall bleibt die Vorlauftemperatur unverändert. Die Schleife wird solange durchlaufen, bis entweder die Raumtemperatur absinkt (die Vorlauftemperatur muss angehoben werden, Schleife 1) oder die Raumtemperatur so weit ansteigt, dass sich die Thermostate infolgedessen schließen und nicht mehr den Sollwert von 80 % erreichen. Die Vorlauftemperatur muss abgesenkt und im Flussdiagramm die dritte Schleife durchlaufen werden. Die dritte, vierte und fünfte while-Schleife (alle zusammengefasst im blau markierten Feld) werden durchlaufen, falls mindestens einer der beiden Ventilöffnungsgraden unter dem gewünschten Sollwert, aber über 40 % liegt. Solange dieser Zustand vorherrscht, wird die Vorlauftemperatur, im Gegensatz zu Schleife 1, stündlich um 2 °C gesenkt. Nach den Ergebnissen der Sprungantwort werden die Thermostate den Ventilöffnungsgrad damit pro Stunde um circa 4 % erhöhen. Das Aufheizen der Vorlauftemperatur soll schneller erfolgen als das Abkühlen, um unter keinen Umständen zu niedrige Temperaturen in den Räumen zu erhalten. Die sechste while-Schleife

wird durchlaufen, wenn mindestens einer der Thermostate weniger als 40 % geöffnet ist. In diesem Fall kann davon ausgegangen werden, dass die Vorlauftemperatur deutlich zu hoch eingestellt ist. Sie soll dementsprechend um 4 °C pro Stunde abgesenkt werden.

Analogie zum PI-Regler

Der entworfene Regler orientiert sich näherungsweise an einem PI-Regler. Wie in Abschnitt 3.1 beschrieben, wird beim PI-Regler durch den I-Anteil die bleibende Regeldifferenz aufaddiert. Dies geschieht im Groben auch im entworfenen Programm. Für die Addition der Regeldifferenz wird dabei nicht jedes veränderte Prozent der Ventilöffnung miteinbezogen, sondern sie wird in vier Bereiche eingeteilt, siehe auch Flussdiagramm in Abbildung 5.1. Zwei Bereiche (grün und gelb) beschreiben den Fall einer nicht vorhandenen Regelabweichung, die jedoch an eine zweite Bedingung (Raumtemperatur) geknüpft ist. Der blaue Bereich beschreibt den Fall einer geringen Regelabweichung und der rote Bereich den einer groben Regelabweichung. Es gibt demnach vier verschiedene Zustände, nach welchen die Höhe der Addition für den I-Anteil bestimmt wird. Die Auswertung der Regeldifferenz und die daraus folgende Umrechnung in die Änderung der Vorlauftemperatur, sowie die Einteilung in die vier Zustände folgt der Analogie eines P-Reglers. Das immer wiederkehrende Durchlaufen der Schleife und die dabei wiederkehrende Addition, beziehungsweise Subtraktion aufgrund der Regeldifferenz stellt den I-Anteil dar.
Anders als beim PI-Regler werden hier die Werte nicht stetig abgefragt und verändert. Zusätzlich wird die Vorlauftemperatur im Falle einer zu tiefen Raumtemperatur in kürzeren Abständen angehoben (alle 30 Minuten) als sie im Falle einer zu hohen Vorlauftemperatur abgesenkt wird (alle 60 Minuten). Da ein zu kalter Raum einen negativen Einfluss auf die Behaglichkeit ausübt, sollte ein solcher Zustand schneller ausgeregelt werden.

5.2 Implementierung

Bei der Implementierung soll zunächst die Funktionalität der Regelung überprüft werden. Es soll sichergestellt werden, dass der Sollwert für die Vorlauftemperatur vom Regler übernommen wird und die Vorlauftemperatur den vorgegebenen Wert annimmt. Des Weiteren soll kontrolliert werden, ob alle Räume die geforderten Solltemperatur erreichen. Hier gilt es insbesondere, die Temperaturen der vier anderen «Schlechträume», die in Abschnitt 4.1.1 genannt wurden, auszuwerten. Die Messwerte in Räumen mit mehreren vorhandenen Thermostaten sind nahezu identisch. Zur besseren Übersicht werden im Folgenden

nur noch die Messwerte eines Thermostaten pro Raum dargestellt, auch wenn mehrere vorhanden sind. Die Implementierung erfolgte in vier Testphasen, dessen Zeiträume in Tabelle 5.1 aufgelistet sind.

Tabelle 5.1: Testphasen für die Implementierung

Testphase 1	31.03.21 09:00 Uhr bis 06.04.21 06:00 Uhr
Testphase 2	08.04.21 12:30 Uhr bis 09.04.21 18:00 Uhr
Testphase 3	12.04.21 15.00 Uhr bis 13.04.21 14.00 Uhr
Testphase 4	13.04.21 14:00 Uhr bis 19.04.21 00:00 Uhr

Testphase 1 mit Raum 0.11 als Referenzraum

Im folgenden Abschnitt sind die Ergebnisse der ersten Testphase aufgeführt. Die Solltemperaturen in den Räumen wurde in dieser Zeit nicht manuell verändert und behielten den von den Nutzerinnen und Nutzern zuvor eingestellten Wert.

1.1: Überprüfen der Temperatur im Referenzraum

In Abbildung 5.2 ist der Verlauf der Soll- und Isttemperatur, sowie des Ventilöffnungsgrads von Referenzraum 0.11 dargestellt. Der sprunghafte Anstieg der Solltemperatur und die anfänglichen zu kalten Temperaturen sind darauf zurückzuführen, dass der Regelalgorithmus in den ersten zwei Tagen noch agil angepasst wurde. Da diese Anpassungszeit bereits einen Einfluss auf das Temperaturverhalten der Räume ausübte, sind die Messwerte im Diagramm mit dargestellt. Die Isttemperatur im Raum fällt gegen Ende der ersten Testphase am 06.04 um circa 05:30 Uhr unter die Solltemperatur. Dies lässt sich jedoch auf das Öffnen von Fenstern zurückführen. Der Ventilöffnungsgrad befindet sich im Schnitt deutlich über 80 %.

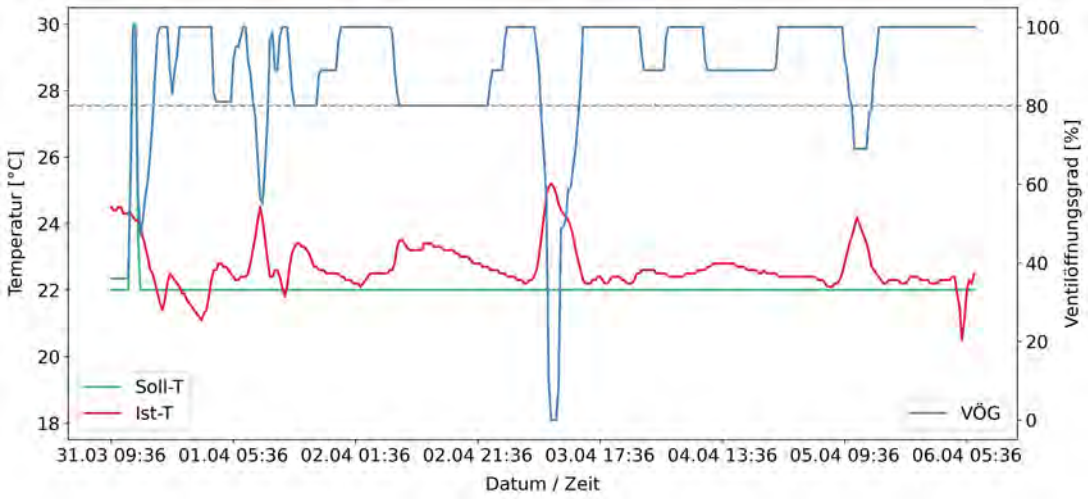


Abbildung 5.2: Testphase 1: Referenzraum 0.11

1.2: Überprüfen der Vorlauftemperaturregelung

In Abbildung 5.3 ist der Verlauf der Vorlauftemperatur und des dazugehörigen Sollwertes zu sehen. Es ist zu erkennen, dass die Vorlauftemperatur nach der Anpassungszeit ohne merkliche Abweichung dem Sollwert folgt. Die letzten abweichenden Werte im Diagramm zeigen den Abbruch der ersten Testphase. Der Sollwert der Vorlauftemperatur wird zu Null und die Vorlauftemperatur wieder außentemperaturgeführt geregelt. Dies führt zu einem Anstieg der Vorlauftemperatur, zu sehen am 06.04 um 04:48 Uhr. Dieser Teil des Reglers funktioniert und muss in den nachfolgenden Testphasen nicht mehr geprüft werden.

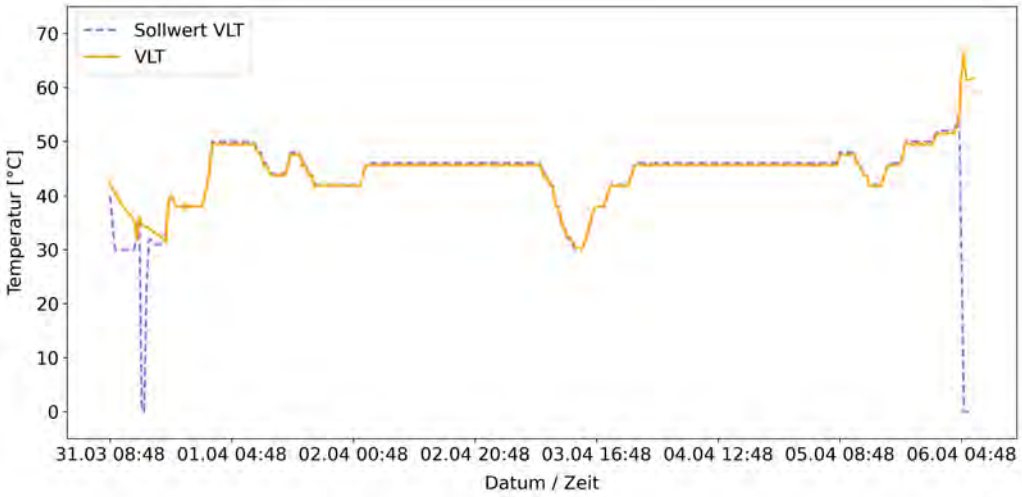


Abbildung 5.3: Testphase 1: Vorlauftemperatur

1.3: Überprüfen der Temperaturen in den «Schlechträumen»

Im Gegensatz zu Raum 0.11 sind die «Schlechträume», wie bereits beschrieben, mit den Thermostaten der Firma Fourdeg ausgestattet. Die Ergebnisse der ersten Testphase zeigen, dass zwei davon ihre Solltemperaturen nicht erreichen konnten. Die Temperaturverläufe für Raum 0.12 und Raum 0.32 sind in den beiden Diagrammen 5.4 und 5.5 dargestellt. Die zwei weiteren «Schlechträume», sowie alle anderen Räume im Heizkreis, wurden ausreichend mit Wärme versorgt.

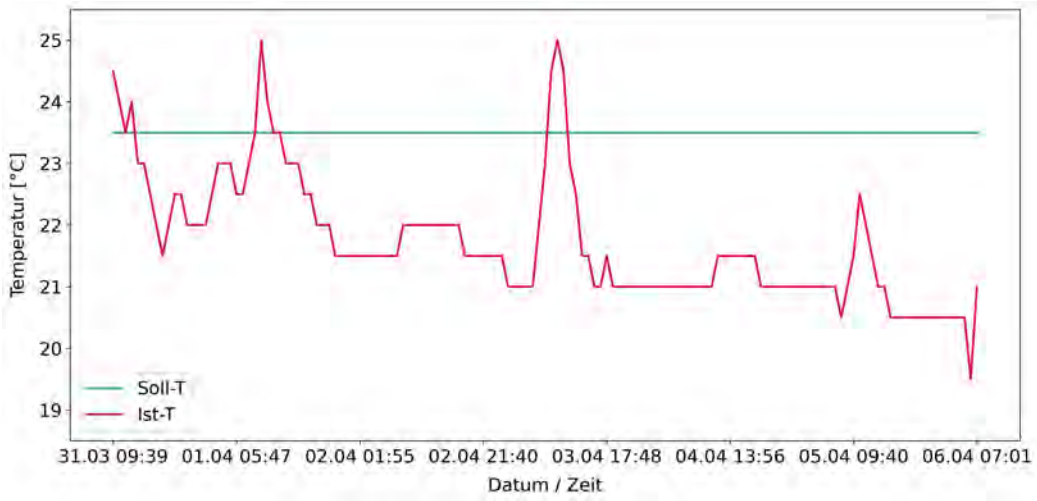


Abbildung 5.4: Testphase 1: Raum 0.12

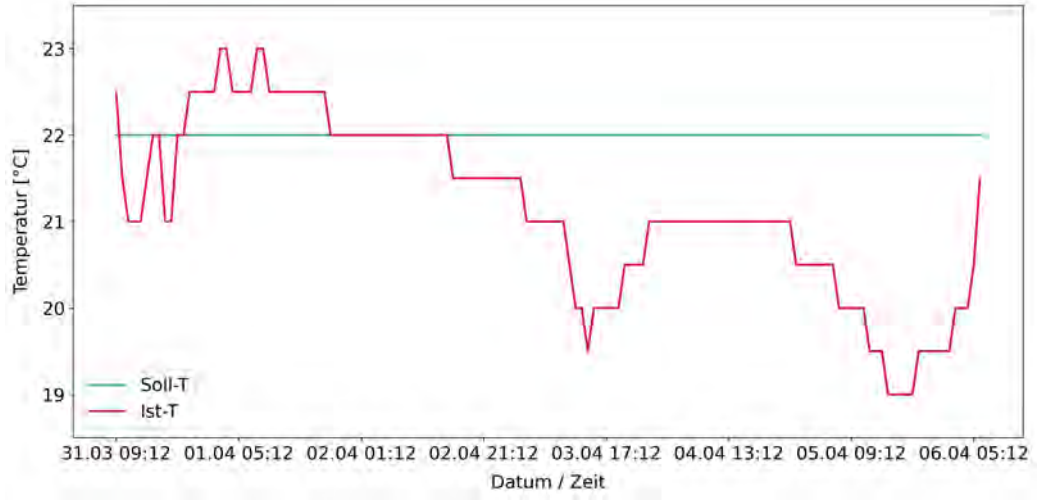


Abbildung 5.5: Testphase 1: Raum 0.32

1.4: Weiteres Vorgehen

Die Ergebnisse der ersten Testphase legen die Vermutung nahe, dass Raum 0.11 nicht der korrekte Referenzraum ist, sondern möglicherweise Raum 0.12 oder Raum 0.32. Diese konnten die geforderte Solltemperatur nicht erreichen und scheinen schlechter mit Wärme versorgt zu werden als Raum 0.11. Aufgrund der unterschiedlichen Sollwerte für die Raumtemperatur in den beiden Räumen war es anhand der Daten der ersten Testphase nicht möglich, den «kälteren» der beiden eindeutig zu bestimmen.
Basierend auf diesen Ergebnissen werden in einer zweiten Testphase die Solltemperaturen in allen drei Räumen (0.11, 0.12 und 0.32) auf 23 °C gestellt. Das Ziel ist, den «kältesten» der drei Räume und damit den Referenzraum bestimmen zu können, da das Temperaturverhalten bei gleichen Solltemperaturen besser zu vergleichen ist. Raum 0.11 bleibt für die Durchführung der zweiten Testphase weiterhin der Referenzraum.

Testphase 2 mit Raum 0.11 als Referenzraum

Im Folgenden werden nur noch die Ergebnisse von Raum 0.12 und Raum 0.32 dargestellt. Raum 0.11 konnte auch während der weiteren Testphasen die geforderten Solltemperaturen erreichen. Die Auswertung dieser und der nachfolgenden Testphasen von Raum 0.11 ist in Anhang A.5 zu finden.

2.1: Vergleich der Temperaturen in den Räumen 0.12 und 0.32

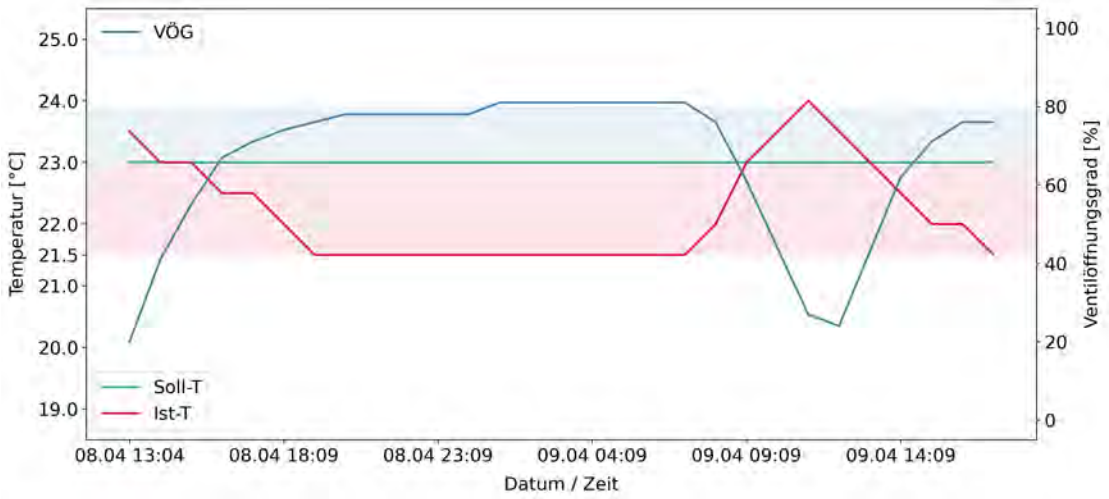


Abbildung 5.6: Testphase 2: Raum 0.12

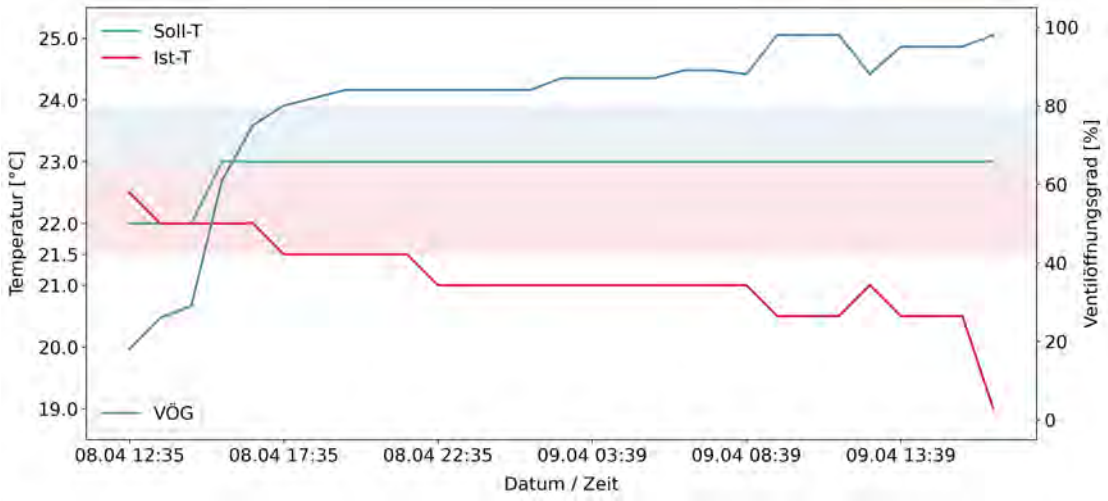


Abbildung 5.7: Testphase 2: Raum 0.32

Die Auswertung der Daten für die Räume 0.12 und 0.32 ist in den Abbildungen 5.6 und 5.7 zu sehen. Zusätzlich zu den Soll- und Isttemperaturen ist hier der Ventilöffnungsgrad mit dargestellt. Beim Vergleich der Diagramme fällt auf, dass Raum 0.32 im Schnitt niedrigere Isttemperaturen aufweist als Raum 0.12. Dennoch können beide die Solltemperatur von 23 °C größtenteils nicht erreichen.

2.2: Weitere Auffälligkeiten

Zusätzlich zu den bereits genannten Ergebnissen zu der Raumtemperatur zeigen die Ventilöffnungsgrade der Räume weitere Abweichungen auf. Der Ventilöffnungsgrad von Raum 0.12 bleibt konstant bei 80 %, obwohl die Raumtemperatur 1.5 °C unter dem Sollwert liegt. Der Bereich bis 1.5 °C unter Solltemperatur ist in den Diagrammen rot markiert. Der Bereich bis 80 % Ventilöffnung ist blau markiert. Bei Raum 0.32 ist ein ähnliches Verhalten der Ventile zu beobachten. Erst bei einer Regelabweichung in der Raumtemperatur von mindestens 1.5 °C öffnet es sich über 80 %. Es bleibt im Schnitt eine Regeldifferenz von circa 2 °C bestehen.

2.3: Weiteres Vorgehen

Aufgrund der soeben beschriebenen Beobachtung liegt die Vermutung nahe, dass sich die Ventile der Fourdeg-Thermostate, im Gegensatz zu den Ventilen der HomeMaticIP-Thermostate in Raum 0.11, erst bis zu 80 % oder mehr öffnen, wenn die Regelabweichung in der Raumtemperatur mindestens 1.5 °C beträgt. Weiter wies Raum 0.32 zwar während

der zweiten Testphase niedrigere Temperaturen auf als Raum 0.12. Die Temperaturunterschiede sind mit 0,5 - 1 °C jedoch gering. Es ließ sich noch kein eindeutiger Referenzraum bestimmen.
Im Zuge dieser Ergebnisse sollen in einer dritten und vierten Testphase beide Räume nacheinander einmal die Rolle des Referenzraumes übernehmen. So kann eindeutig bestimmt werden, welcher sich besser als Referenzraum eignet. Zusätzlich gilt es, die aufgestellte Vermutung bezüglich des Verhaltens der Thermostatventile von der Firma Fourdeg zu bestätigen oder zu widerlegen.

Testphase 3 mit Raum 0.12 als Referenzraum

In der dritten Testphase wird zunächst Raum 0.12 als Referenzraum getestet. Der dazugehörige Code ist in Anhang A.4.2 aufgeführt. Die Struktur des Codes ist, bis auf wenige Ausnahmen, gleich aufgebaut wie bei Raum 0.11. Die Thermostate von Fourdeg, welche in den Räumen 0.32 und 0.12 verbaut sind, zeichnen ihre Messdaten nur alle 60 Minuten auf. Es können in Schleife 1 dementsprechend nicht alle 30 Minuten neue Messdaten abgefragt werden. Die Messdaten werden neu stündlich abgefragt, und die Vorlauftemperatur infolgedessen um 3 °C anstelle der 2 °C erhöht. Weiter sind in Raum 0.12 drei Thermostate vorhanden und werden in die Regelung miteinbezogen. Die Solltemperaturen in den Räumen sind dieselben wie in der ersten Testphase.

3.1: Überprüfen der Temperaturen im Referenzraum

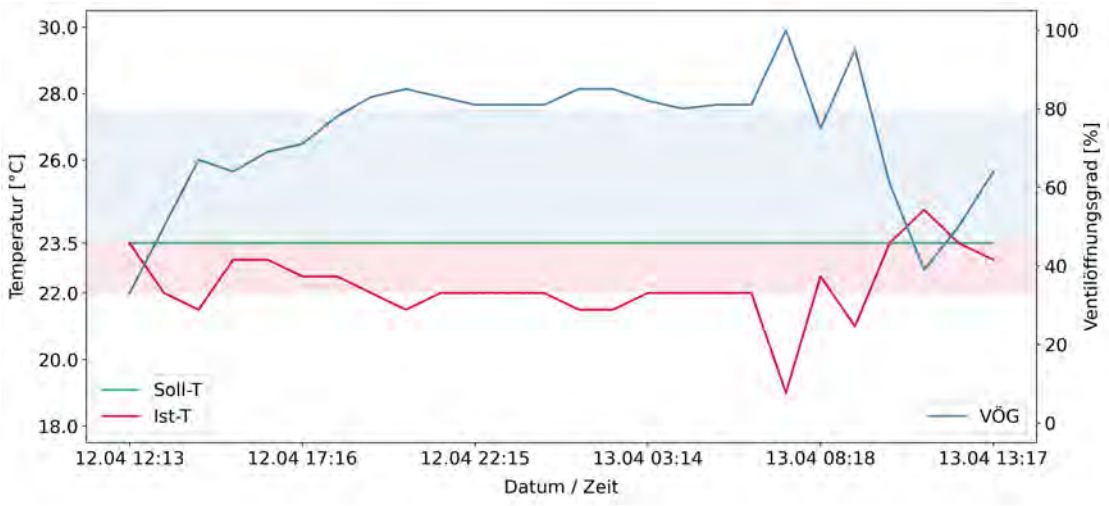


Abbildung 5.8: Testphase 3: Referenzraum 0.12

Obwohl Raum 0.12 die Rolle des Referenzraumes einnimmt, erreicht dieser die gewünschte Solltemperatur nicht. Weiter bestätigen die Verläufe in Diagramm 5.8 die Annahme, dass die Raumtemperatur bei 1.5 °C unter dem Sollwert konstant bleibt und die Ventilöffnungsgrade zu dieser Zeit unter oder bei 80 % liegen. Es ist zu erkennen, dass sich die Ventile erst bei einer Regelabweichung von über 1.5 °C weiter öffnen.

3.2: Überprüfen der Temperaturen in Raum 0.32

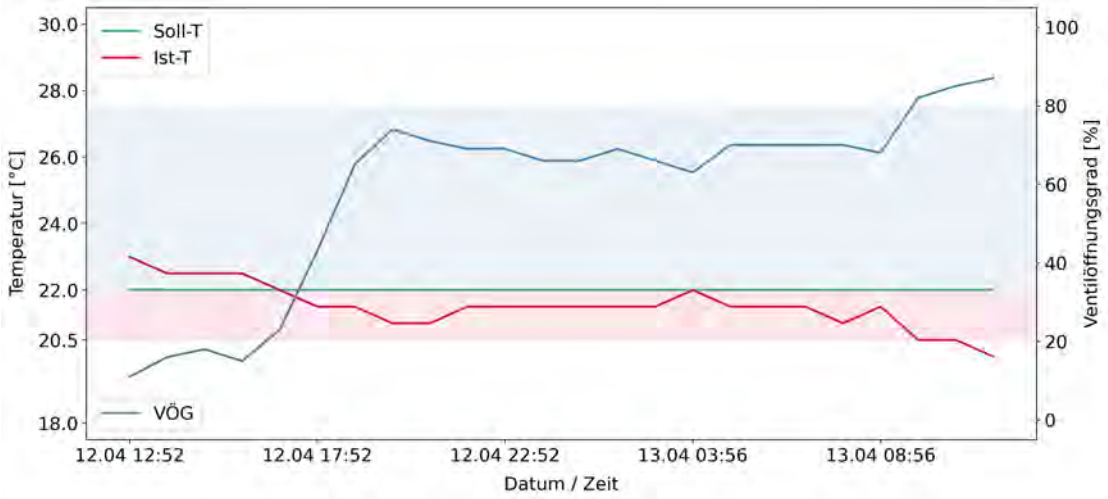


Abbildung 5.9: Testphase 3: Raum 0.32

Auch Raum 0.32 erreicht die vorgegebene Solltemperatur nicht, wie in Abbildung 5.9 zu sehen ist. Der Unterschied zwischen Soll- und Isttemperatur ist jedoch geringer als in den Ergebnissen von Testphase 1 und 2. Grund dafür kann sein, dass die Solltemperatur im Referenzraum 0.12 in der dritten Testphase mit 23.5 °C höher ist als in Raum 0.32 mit 22 °C. In Testphase 1 und 2 lag die Solltemperatur in beiden Räumen bei 22 °C.

3.3: Weiteres Vorgehen

In einer vierten und letzten Testphase der Implementierung soll geklärt werden, ob Raum 0.32 der gesuchte Referenzraum ist. Da die Annahme bezüglich des Verhaltens der Thermostatventile der Firma Fourdeg in dieser dritten Testphase bestätigt wurde, muss die Regelung für die nächste Testphase entsprechend angepasst werden. Um einen Ventilöffnungsgrad von mindestens 80 % zu erhalten, wird die Solltemperatur im Raum 0.32 auf 24

°C gestellt, 2 °C höher als die gewünschte Raumtemperatur. Es sollte überprüft werden, ob damit eine Raumtemperatur von 22 °C erreicht wird.

Testphase 4 mit Raum 0.32 als Referenzraum

In der folgenden Testphase wird Raum 0.32 als Referenzraum getestet. Der dazugehörige Code ist in Anhang A.4.3 zu finden. Wie bereits erwähnt, wird hier eine Abweichung von 2 °C anstelle der 0 °C von der geforderten Raumtemperatur toleriert. In Raum 0.32 befindet sich nur ein Thermostat.

4.1: Überprüfen der Temperaturen im Referenzraum

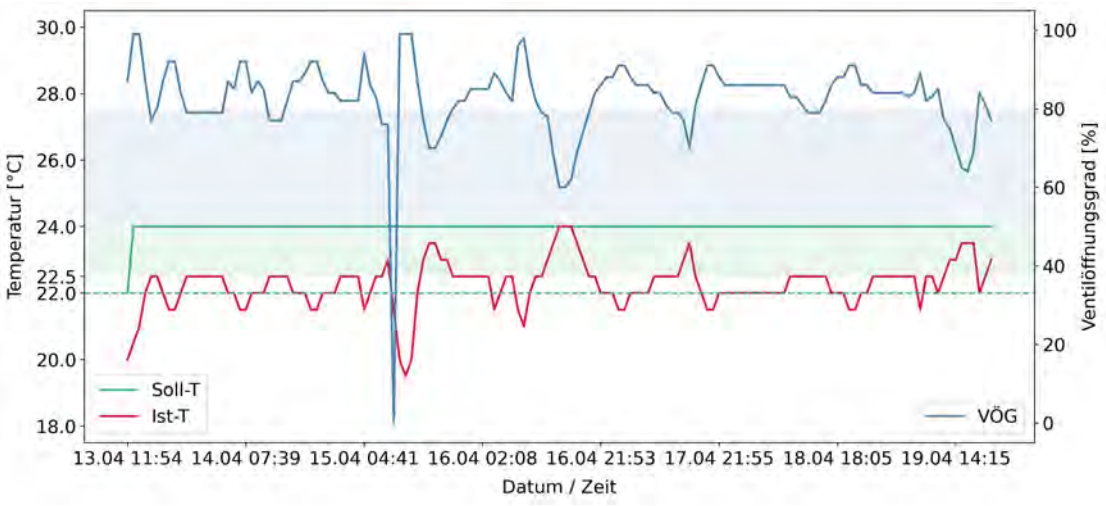


Abbildung 5.10: Testphase 4: Referenzaum 0.32

In Abbildung 5.10 lässt sich ein weiteres Mal erkennen, dass sich die Thermostatventile der Firma Fourdeg erst bei einer Regelabweichung von 1.5 °C in der Raumtemperatur über die geforderten 80 % öffnen. Der Bereich ist grün markiert. Die gestrichelte Linie kennzeichnet die fehlenden 0.5 °C bis zur gewünschten Solltemperatur. Es lässt sich die Annahme bestätigen, dass die geforderte Raumtemperatur von 22 °C bis auf wenige Ausnahmen erreicht wird. Größere Temperaturschwankungen, wie am 15.04 um circa 06.00 Uhr, sind auf geöffnete Fenster zurückzuführen.

4.2: Überprüfen der Temperaturen in Raum 0.12

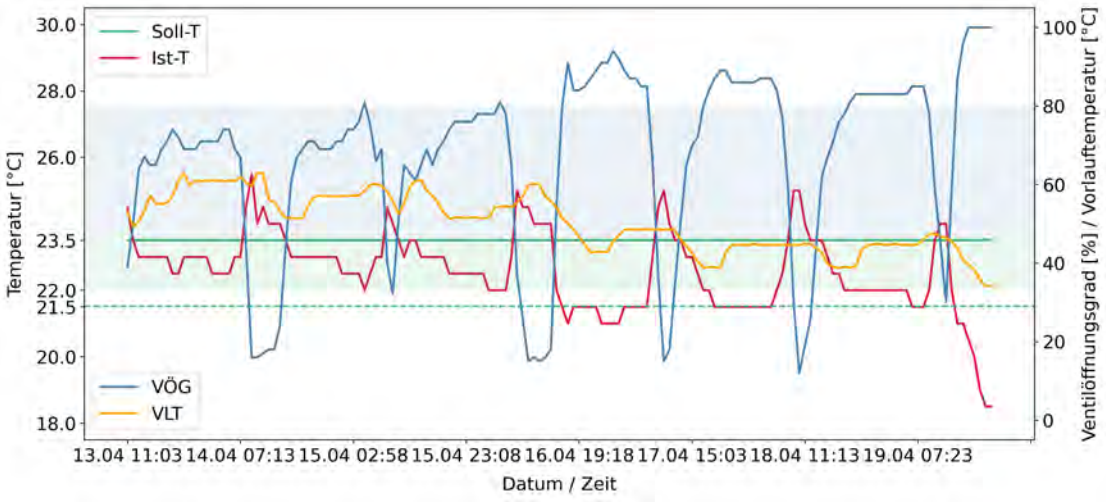


Abbildung 5.11: Testphase 4: Raum 0.12

Dieselben Annahmen lassen sich für Raum 0.12 in Abbildung 5.11 bestätigen. Hier ist zusätzlich die Vorlauftemperatur mit dargestellt. Raum 0.12 wird am letzten Tag der Testphase deutlich zu kalt. Es ist unwahrscheinlich, dass um diese Zeit (gegen 20 Uhr abends) ein Fenster im Seminarraum geöffnet wurde. Es könnte vielmehr auf die niedrige Vorlauftemperatur zurückzuführen sein, welche zu diesem Zeitpunkt circa 35 °C betrug. Die Außenwandfläche von Raum 0.12 besteht zu über 60 % aus Fensterglas. Bei Raum 0.32 beträgt der Anteil nur knapp 25 %. Dadurch entstehen in Raum 0.12 hohe solare Gewinne, vor allem in der Mittagszeit. Gleichzeitig geht die Wärme im Raum bei einer niedrigen Vorlauftemperatur und ohne Sonneneinstrahlung schneller verloren als in einem Raum ohne einen solchen Anteil an Fensterfläche. Es ist anzunehmen, dass Raum 0.32 aufgrund seiner genannten Eigenschaften an diesem Tag viel Wärme speichern konnte, weshalb die Vorlauftemperatur bis auf 35 °C abgesunken ist. Für Raum 0.12 reichte diese Vorlauftemperatur nicht aus, um die geforderte Raumtemperatur zu erreichen.

Fazit

Während der vier Testphasen der Implementierung konnte festgestellt werden, dass es keinen eindeutigen Referenzraum gibt. In der vorangegangenen Studienarbeit wurde Raum 0.11 als Referenzraum bestimmt, da dieser im Schnitt der am schlechtesten versorgte Raum ist. Aufgrund der vorliegenden Ergebnisse lässt sich jedoch schlussfolgern, dass

dies im Umkehrschluss nicht bedeutet, dass Raum 0.11 auch zu jedem Zeitpunkt der am schlechtesten versorgte Raum und damit der Referenzraum ist. Es hat sich stattdessen gezeigt, dass Räume mit unterschiedlichen baulichen Eigenschaften zu unterschiedlichen Zeiten die Rolle des am schlechtesten versorgten Raumes und somit des Referenzraumes übernehmen.
Auch in der Studienarbeit wurde zunächst eine Auswahl an «Schlechträumen» (0.11, 0.12, 0.32, 1.18, 1.21) getroffen. Dies wäre für die bedarfsgeführte Regelung des Testgebäudes die geeignetere Methode gewesen, als aufgrund von Durchschnittswerten daraus einen einzelnen Referenzraum zu bestimmen. Auf diese Weise können mehrere Räume mit unterschiedlichem Temperaturverhalten und verschiedenen Solltemperaturen in die Regelung miteinbezogen werden. Die Rolle des Referenzraumes übernimmt dabei jeweils der Raum, der zum aktuellen Zeitpunkt das schlechteste Wärmeverhalten aufweist.
Im Fall des Testgebäudes konnten sich in den vier Testphasen aus den fünf «Schlechträumen» der Studienarbeit die Räume 0.12 und 0.32 als zusätzliche Referenzräume bestätigen lassen, da sie die geforderten Raumtemperaturen im Gegensatz zu allen weiteren Räumen im Heizkreis nicht erreichen konnten. Raum 0.11 konnte in den vier Testphasen zu jeder Zeit die Solltemperaturen erreichen. Die Diagramme sind in Anhang A.5 zu finden. Aufgrund seiner Wahl zum Referenzraum in der Studienarbeit, und um sicherzustellen, dass bei der finalen Auswertung kein Raum zu kalt wird, bleibt 0.11 dennoch als Referenzraum in der Regelung bestehen. In Testphase 4 wurde deutlich, dass das Ergebnis eines falschen Referenzraumes und die damit verbundene, zu geringe Vorlauftemperatur, unter Umständen erst nach einer Woche festgestellt werden kann. Die finale Testphase soll demnach mit den drei «Schlechträumen» 0.11, 0.12 und 0.32 als Referenzräume erfolgen.
Des Weiteren konnte eine spezielle Eigenheit in Bezug auf das Öffnungsverhalten der Fourdeg Thermostatventile festgestellt werden. Bei den Referenzräumen 0.12 und 0.32 wird deswegen eine Raumtemperatur toleriert, die um 2 °C kälter ist als die geforderte Solltemperatur. In Referenzraum 0.11 sind nicht die Thermostate von Fourdeg, sondern die HomeMaticIP Thermostate verbaut. Da sich diese auch ohne Regeldifferenz in der Raumtemperatur über 80 % öffnen, wird hier erwartet, dass die vorgegebene Solltemperatur erreicht wird. Dies gilt ebenfalls für die Temperaturen aller weiteren Räume im Heizkreis. Da diese nicht als «Schlechträume» identifiziert wurden, kann davon ausgegangen werden, dass durch die Thermostatventile mit einer Öffnung von unter 80 % genügend Wärme in den Räumen zur Verfügung gestellt wird.

Der finale Code für den Regelung mit den drei Referenzräumen ist in Anhang A.4.4 zu finden. Der Code von Raum 0.11 wurde dahingehend angeglichen, dass die Vorlauftemperatur, wie es bei den anderen der Fall ist, stündlich um 3 °C angehoben wird. Bei den Codes von Raum 0.12 und 0.32 wurden keine Anpassungen mehr vorgenommen. Im Groben lässt sich das finale Programm als ein Zusammenschluss der bereits vorhandenen Codes der einzelnen Referenzräume beschreiben. Neu muss zusätzlich eine Entscheidung getroffen werden, welcher Raum die Rolle des Referenzraumes übernimmt. Zum besseren Verständnis wurde der Entscheidungsalgorithmus in einem weiteren Flussdiagramm in Abbildung 5.12 graphisch aufgebaut. Die Codes der einzelnen Räume sind vereinfacht als Kästchen mit der Raumnummer dargestellt. Nach dem Abfragen der aktuellen Messwerte (∗) wird geprüft, ob Raum 0.11 der aktuelle Referenzraum ist. Dafür werden die Unterschiede zwischen Soll- und Isttemperatur (dT) der drei möglichen Referenzräume miteinander verglichen. Der Raum mit der größten Regelabweichung wird ausgewählt und die Schleifen des Programms durchlaufen. Dabei wurde ein Offset zwischen Raum 0.11 und den Räumen 0.12 und 0.32 eingebaut, da bei Letzteren eine Abweichung von 2 °C toleriert wird. Bei jeder Schleife wird am Ende ein weiteres Mal abgeklärt, ob sich die Regelabweichung in einem der anderen Räume vergrößert hat und der Referenzraum gewechselt werden sollte. Falls dies der Fall ist, wird die Schleife verlassen und eine andere, passende Schleife betreten. Im Folgenden ist eine solches Exitszenario beispielhaft aufgeführt:

```
if (dt_011 > dt_032-2) and (dt_011 > 0):
    break
elif (dt > dt_032) and (dt > 2):
    break
else:
    continue
```

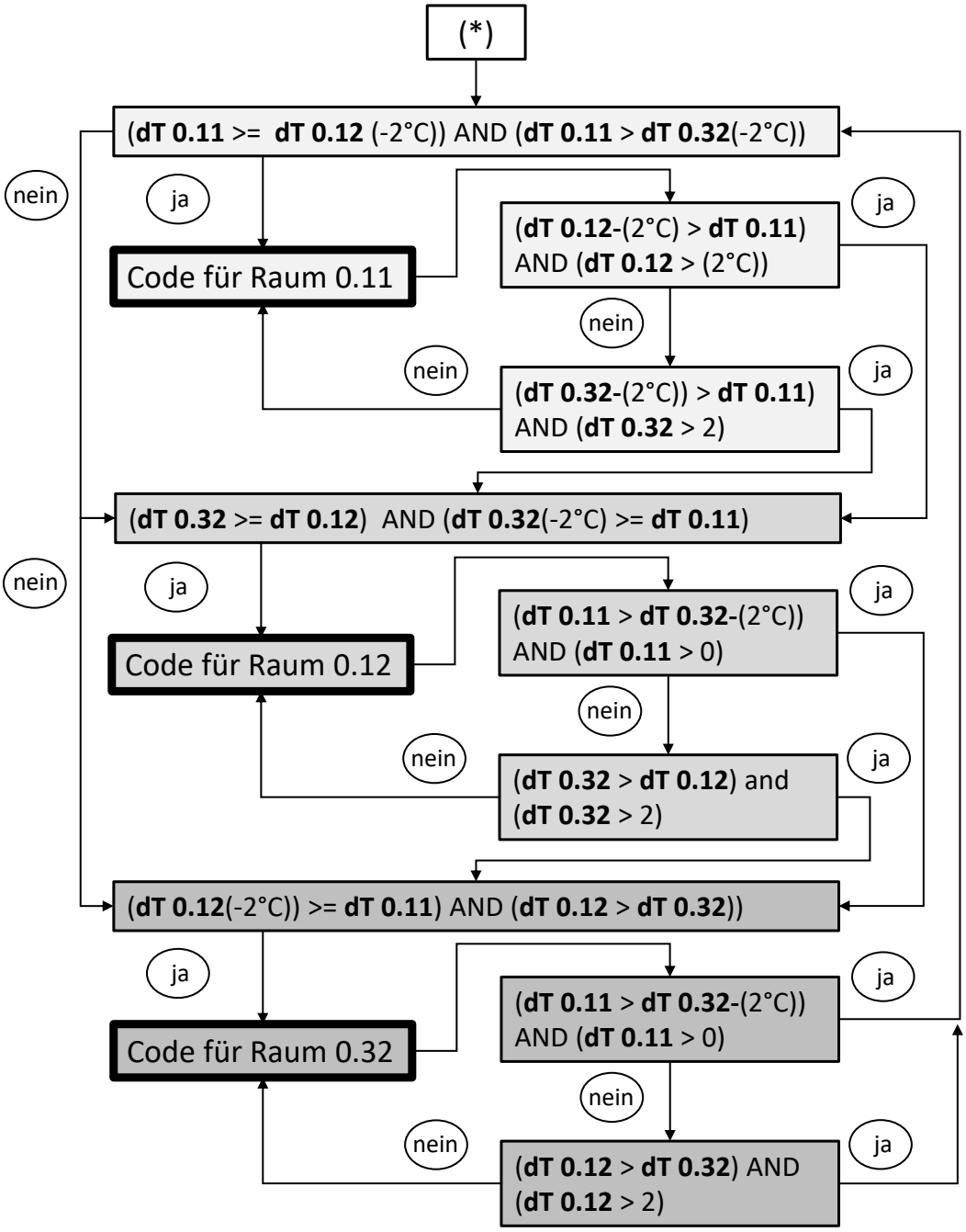


Abbildung 5.12: Flussdiagramm zur Wahl des Referenzraumes

6 Darstellung der Ergebnisse, Diskussion und Fazit

In diesem Kapitel werden die Ergebnisse der finalen fünften Testphase dargestellt und diskutiert. Als Erstes soll gezeigt werden, ob und inwieweit die Vorlauftemperatur der bedarfsgeführten Regelung im Vergleich zur außentemperaturgeführten Regelung abgesenkt werden konnte. Im zweiten Teil soll überprüft werden, ob die geforderten Solltemperaturen in den Räumen im Heizkreis Nord während der finalen Testphase erreicht werden konnten. Abschließend folgt ein Fazit zu den beiden vorherigen Abschnitten und zur Regelung im Allgemeinen.

6.1 Ergebnisse zur Vorlauftemperatur und Diskussion

Um die Ergebnisse der Vorlauftemperatur mit den Werten der ursprünglichen Regelung vergleichen zu können, soll aus den Messdaten eine neue Heizkurve erstellt werden. Dafür müssen Werte für die Außentemperatur in den Messdatensatz der Vorlauftemperatur integriert werden. Für die grafische Darstellung der Heizkurve kann anschließend die Vorlauftemperatur über der Außentemperatur als X-Y Plot abgebildet werden. Wie bereits in Kapitel 4 Abschnitt 4.1.2 erwähnt, ist am Testgebäude ein Sensor zur Messung der Außentemperatur angebracht. Da der genaue Standort des Sensors nicht bekannt ist und durch eine Fehlplatzierung die Messwerte verfälscht werden können, sollen die Messdaten des Sensors mit den Referenzwerten zweier Wetterstationen verglichen werden.

Zunächst sollen ein Vergleich mit den Messdaten des Hamburger Luftmessnetzes stattfinden. Das Hamburger Luftmessnetz zeichnet Messdaten an drei verschiedenen Standorten auf [ham21]. Der nächste Standort befindet sich sechs Kilometer vom Testgebäude entfernt. Anhand des Diagramms in Abbildung 6.1 ist zu erkennen, dass die Messdaten des Sensors am Testgebäude durchweg höhere Werte aufweisen als die des Hamburger Luft-

messnetzes. Der Temperaturunterschied zwischen den Messdaten ist nicht konstant und kann bis zu 9 °C betragen.

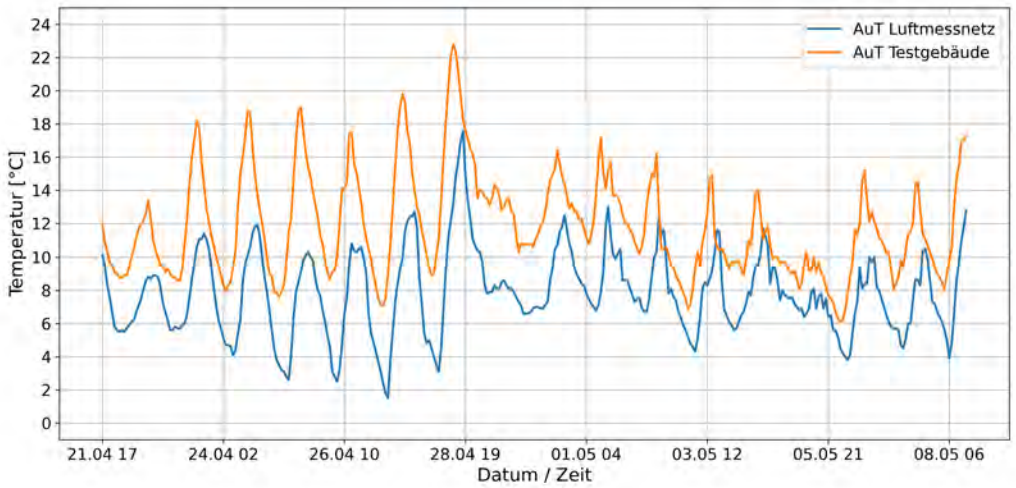


Abbildung 6.1: Vergleich der Messdaten für die Außentemperatur des Sensors am Testgebäude und des Hamburger Luftmessnetzes [ham21]

Es werden nun zusätzlich die Werte der dritten Wetterstation an der Grotestraße hinzugezogen, welche sich einen Kilometer vom Testgebäude entfernt befindet [Dar21]. Hier werden die Werte für die Außentemperatur stündlich aufgezeichnet. Es sollen im Folgenden die Höchst- und Tiefstwerte der zwei Wetterstationen und des Sensors am Testgebäude an einem beliebigen Tag miteinander verglichen werden. Die Messdaten werden gerundet. In Tabelle 6.1 sind die Messdaten für den 28. April aufgeführt. Der Sensor vom Testgebäude zeichnete eine Tiefsttemperatur von 9 °C und eine Höchsttemperatur von über 22 °C auf. Die Werte vom Luftmessnetz weisen einen Temperaturbereich von 3 °C bis 18 °C auf und die Werte der Wetterstation Grotestraße von 4 °C bis 18 °C.

Tabelle 6.1: Vergleich der Messdaten für die AuT des Sensors am Testgebäude mit der Wetterstation Grotestraße [Dar21] und des Luftmessnetzes HH [ham21]

AuT [°C] / Zeit	00	02	04	06	08	10	12	14	16	18	20	22
Grotestraße	4	4	2	2	5	10	13	15	18	18	15	11
Luftmessnetz HH	5	5	4	3	5	9	12	15	16	18	13	11
Sensor Testgebäude	11	9	9	11	13	18	22	22	20	18	17	16

Es wird deutlich, dass die Messdaten der beiden Wetterstationen gleichwertig sind und nicht mit den Messdaten des Sensors übereinstimmen. Für das Erstellen der Heizkurven erweist sich dies als problematisch. Für einen korrekten Vergleich der beiden Heizkurven ist es vonnöten, jeweils die gleiche Außentemperatur für die Auswertung zu nutzen. Die Vorlauftemperatur des Heizkreises Nord wird in der ursprünglichen, außentemperaturgeführten Regelung nach den Werten des Sensors am Testgebäude geregelt. Sie sollte demnach mit ebendiesen Werten erstellt werden, da ein Austausch der Messdaten für die Außentemperatur zu einer Verzerrung der Heizkurve führen könnte.
Für die Erstellung der neuen Heizkurve sind jedoch die Daten der Wetterstationen besser geeignet als die Messdaten des Sensors am Testgebäude, da die Messdaten des Sensors nicht die realen Bedingungen während der Testphase wiedergeben. Aus diesem Grund sollen im Folgenden beide Heizkurven jeweils mit den Messdaten der Außentemperatur des Sensors am Testgebäude und mit den Messdaten des Hamburger Luftmessnetzes erstellt und diskutiert werden. Die Messdaten der Wetterstation Grotestraße werden für die Auswertung nicht verwendet, da davon ausgegangen wird, dass diese mit den Werten des Hamburger Luftmessnetz übereinstimmen.

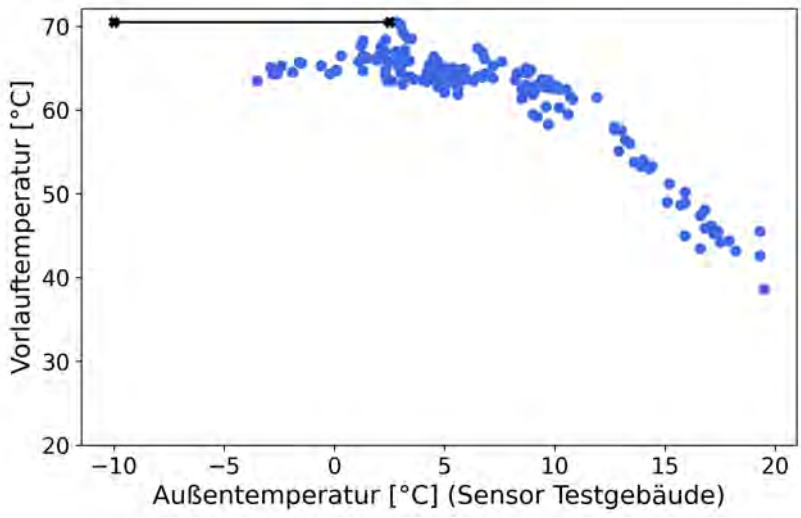


Abbildung 6.2: Heizkurve Heizkreis Nord: Messdaten der VLT aufgetragen über AuT Sensor Testgebäude

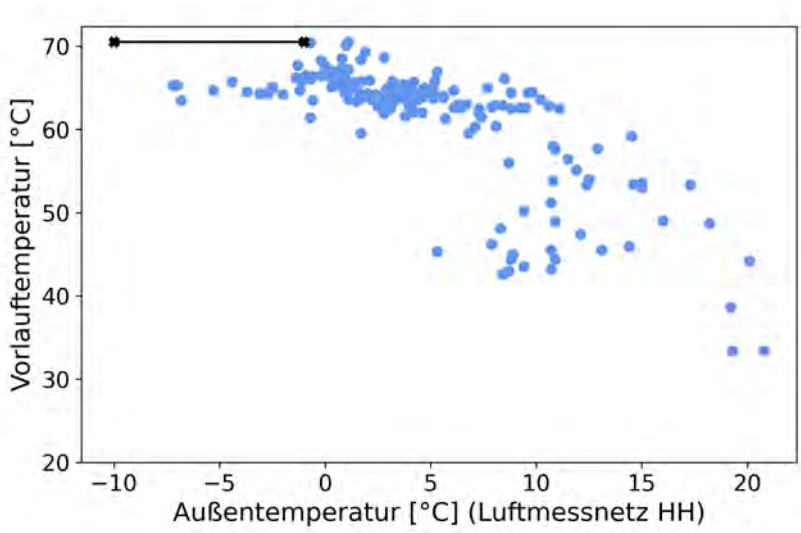


Abbildung 6.3: Heizkurve Heizkreis Nord: Messdaten der VLT aufgetragen über AuT Luftmessnetz HH

Zunächst wurden die ursprünglichen Heizkurven für den Heizkreis Nord erstellt. Diese sind in den Abbildungen 6.2 mit den Messdaten vom Sensor am Testgebäude und 6.3 mit den Messdaten vom Hamburger Luftmessnetz zu sehen. Es wurden die Daten im Zeitraum vom 17.12.2020 bis zum 31.05.2021 ausgewertet (ohne den Zeitraum vom 04.03.2021 bis 08.05.2021, da in dieser Zeit der Sollwert für die Vorlauftemperatur für die Sprungantworten mehrmals verändert wurde und die Testphasen der bedarfsgeführten Regelung stattfanden). Beim Vergleich der beiden Abbildungen bestätigt sich die Vermutung, dass die Heizkurve mit den Messdaten des Hamburger Luftmessnetzes mehr streut als die Heizkurve mit den Messdaten des Sensors am Testgebäude. Insbesondere im Bereich mit einer Außentemperatur ab 10 °C folgt die Kurve keinem eindeutigen Verlauf. Eine weitere Auffälligkeit ist darin zu erkennen, dass der maximale Wert für die Vorlauftemperatur (im Diagramm eingezeichnet bei 70 °C) nicht bei der geringsten Außentemperatur, sondern bei circa 3 °C, beziehungsweise 0 °C auftritt.

In den Abbildung 6.4 und 6.5 sind zusätzlich die neuen Heizkurven mit dargestellt. Für eine bessere Gegenüberstellung sind die beiden Heizkurven zusammen in einem Diagramm aufgeführt. Die finale Testphase für die neue Heizkurve lief vom 21.04.21 um 17:00 Uhr bis zum 08.05.21 um 13:45 Uhr. Bis 23.04.21 um 23 Uhr musste die Regelung noch angepasst werden, weswegen diese Daten nicht in die Auswertung miteinbezogen werden.

Weiter fiel vom 03.05.21 um 12 Uhr bis 07.05.21 um 16 Uhr die Regelung aufgrund von Verbindungsproblemen aus. Auch diese Daten wurden nicht ausgewertet.

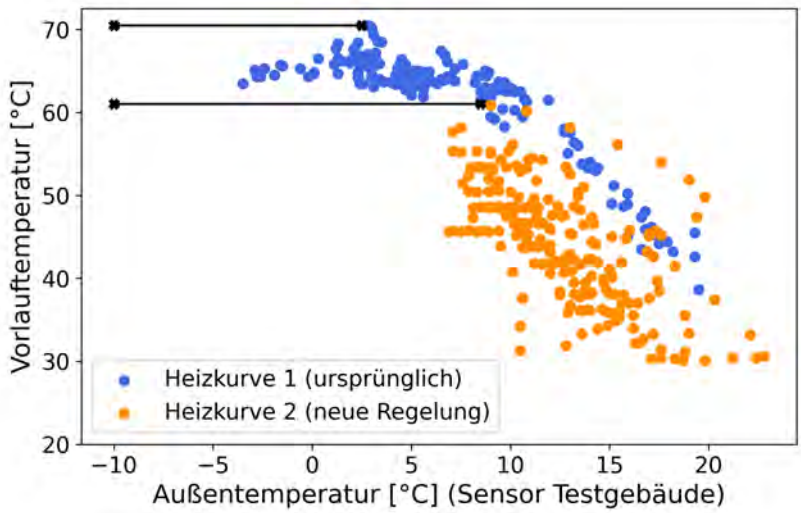


Abbildung 6.4: Vergleich der Messdaten der ursprünglichen und der neuen Regelung (AuT Sensor Testgebäude)

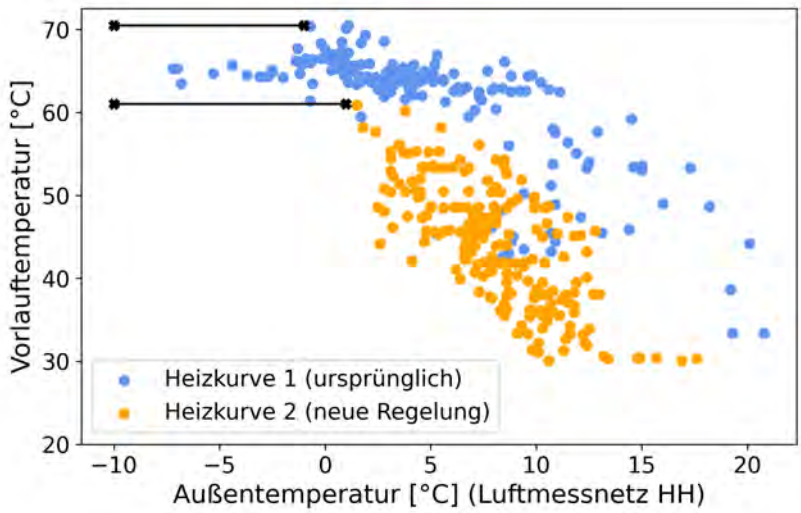


Abbildung 6.5: Vergleich der Messdaten der ursprünglichen und der neuen Regelung (AuT Luftmessnetz HH)

Zunächst fällt auf, dass bei Heizkurve 2 nur Werte bis zu einer Außentemperatur von 2 °C vorhanden sind. Das liegt daran, dass während der finalen Testphase keine tieferen Außentemperaturen erreicht wurden. In beiden Abbildungen ist zu erkennen, dass nahezu die gesamten Datenpunkte der Heizkurve der neuen Regelung (2) unter denjenigen der ursprünglichen Heizkurve (1) liegt. Dass bedeutet, dass durch die bedarfsgerechte Vorlauftemperaturreglung größtenteils eine geringere Vorlauftemperatur benötigt wird. Die beiden maximalen Vorlauftemperaturen sind eingezeichnet. Bei Heizkurve 2 liegt diese bei 60 °C und ist somit 10 °C niedriger als die maximale Vorlauftemperatur von Heizkurve 1. Heizkurve 2 verschiebt sich durch die Messdaten des Sensors am Testgebäude in Abbildung 6.4 nach rechts im Vergleich zu Abbildung 6.5. Die tiefste Außentemperatur, die mit dem Sensors am Testgebäude aufgezeichnet wurde, liegt bei 8 °C. Sie ist damit deutlich wärmer, als die tiefste gemessene Temperatur vom Hamburger Luftmessnetz, welche bei 2 °C liegt. Des Weiteren ist eine deutliche Streuung der Datenpunkte von Heizkurve 2 festzustellen. In Abbildung 6.5 kann die Vorlauftemperatur bei einer Außentemperatur von 5 °C beispielsweise in einem Bereich zwischen 40 °C und 60 °C liegen.
Um eine bessere Auswertung der stark variierenden Daten zu ermöglichen, wurden in den folgenden Diagrammen 6.6 und 6.7 die beiden Heizkurven erneut dargestellt, diesmal mit den Durchschnittswerten. Hierfür wurden die Werte der Außentemperatur ohne Nachkommastelle gerundet. Anschließend wurde von allen zugehörigen Daten der Vorlauftemperatur der Mittelwert gebildet.

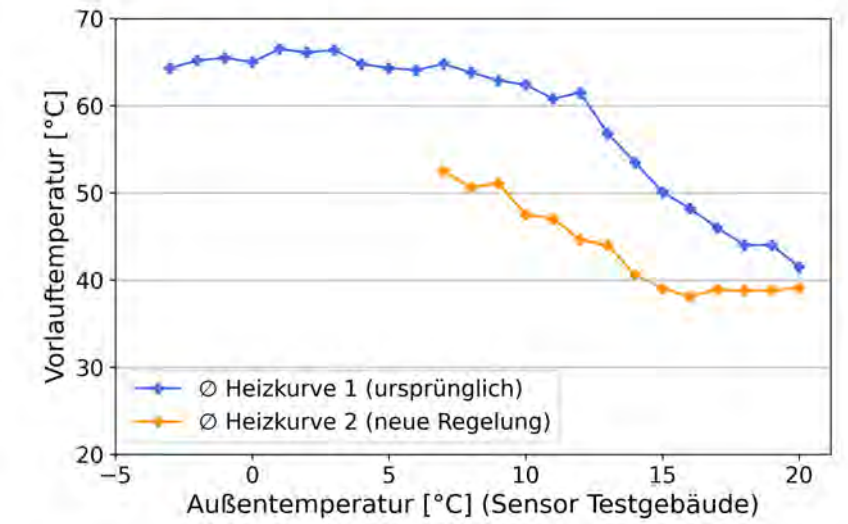


Abbildung 6.6: Vergleich der Durchschnittswerte (AuT Sensor Testgebäude)

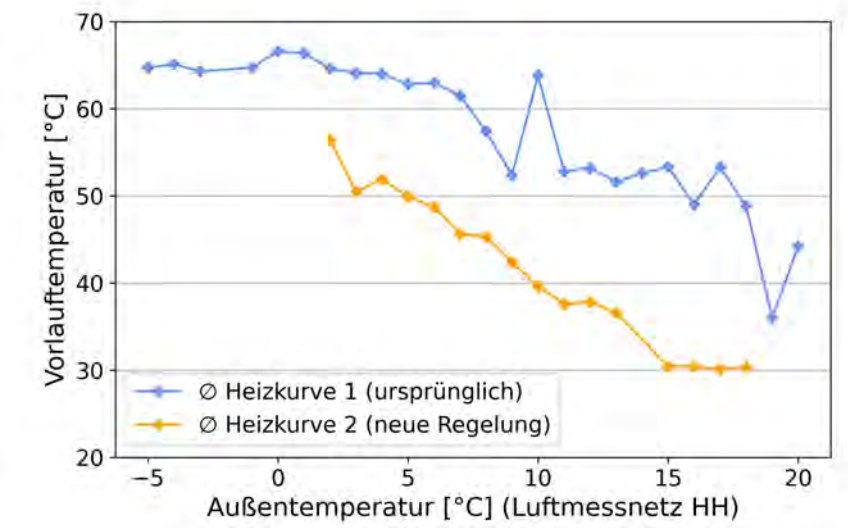


Abbildung 6.7: Vergleich der Durchschnittswerte (AuT Luftmessnetz HH)

Es lässt sich feststellen, dass in beiden Abbildungen die durchschnittliche Heizkurve der bedarfsgeführten Regelung unter der durchschnittlichen Heizkurve der außentemperaturgeführten Regelung liegt. In Abbildung 6.6 weist Heizkurve 2 eine geringere Steigung auf als Heizkurve 1. In Abbildung 6.6 verhält es sich umgekehrt, wobei hier die Daten von Heizkurve 1 durch die Streuung (siehe Abbildung 6.3) keinen eindeutigen Verlauf aufweisen. Im Schnitt liegt der Temperaturunterschied zwischen den Heizkurven mit den Messdaten des Sensors bei 10,7 °C, bei den Messdaten des Luftmessnetzes bei 15,2 °C (der stark abweichende Wert der Vorlauftemperatur bei 10 °C Außentemperatur wurde nicht mitgerechnet).

Diskussion der Ergebnisse

Durch einen Vergleich der ursprünglichen Heizkurve der außentemperaturgeführten Regelung (1) mit der Heizkurve der neuen bedarfsgeführten Regelung (2) soll die Absenkung der Vorlauftemperatur dargestellt werden. Zunächst wurde eine Diskrepanz zwischen der Außentemperatur des Sensors am Testgebäude und den Messdaten zweier Wetterstationen festgestellt. Der Sensor am Testgebäude weist fast durchweg zu hohe Temperaturen auf. Die außentemperaturgeführte Regelung von Heizkurve 1 basiert auf den Messdaten des Sensors. Heizkurve 2 wird durch diese zu hoch gemessenen Temperaturen im Diagramm deutlich nach rechts verschoben. Um einen direkten Vergleich zu ermöglichen wurden aufgrund dessen die beiden Heizkurven jeweils einmal mit den Temperaturdaten des Sensors

und denjenigen des Hamburger Luftmessnetzes ausgewertet.
Bei beiden Auswertungen konnte eine deutliche Streuung der Datenpunkte von Heizkurve 2 festgestellt werden. Diese Streuung kann einerseits auf die begrenzte Anzahl der Daten zurückzuführen sein, da die finale Testphase nur über rund zwei Wochen lief. Weiter können hier die großen Temperaturschwankungen zwischen Tag und Nacht während der finalen Testphase als Gründe genannt werden. Wie in Abschnitt 6.1 bereits beschrieben und in Abbildung 6.1 zu erkennen ist, konnten diese Temperaturunterschiede bis zu 15 °C betragen. Als Beispiel wäre die Vorlauftemperatur morgens nach einer kalten Nacht bei 10 °C Außentemperatur noch bei einem maximalen Wert von 50 °C. Tagsüber sinkt die Vorlauftemperatur und die Räume heizen sich auf. Erreicht die Außentemperatur abends wieder die 10 °C, so liegt die Vorlauftemperatur unter Umständen noch bei 30 °C, sofern und solange die Referenzräume die Wärme speichern können. Als Ergebnis werden im Diagramm bei der gleichen Außentemperatur für die Vorlauftemperatur zwei Werte angezeigt, die sich deutlich voneinander unterscheiden. Ein Beispiel für ein solches Szenario ist in Tabelle 6.2 aufgeführt, die auffälligen Werte sind gekennzeichnet.

Tabelle 6.2: Vergleich des Sollwertes der Vorlauftemperatur während unterschiedlicher Außentemperaturen am 28.04.21

Temperatur [°C] / Zeit	00	02	04	06	08	10	12	14	16	18	20	22
VLT Soll	41	47	50	56	56	**49**	41	33	30	30	30	**30**
AuT Grotestraße	4	4	2	2	5	**10**	13	15	18	18	15	**11**

Es wäre zu prüfen, ob sich die Streuung von Heizkurve 2 verringert, wenn die Testphase in den Wintermonaten durchgeführt wird. Dies würde einerseits die Vollständigkeit der Werte verbessern, da aktuell nur Werte ab einer Außentemperatur von 2 °C aufgezeichnet werden konnten. Die finale Testphase fand in den Monaten April und Mai stattfand, in welchen die Außentemperatur keine Minusgrade mehr aufwies. Außerdem wären die Temperaturschwankungen zwischen Tag und Nacht im Winter geringer als in den wärmeren Monaten [DWD21]. Weiter könnte es von Vorteil sein, die Testphase über einen längeren Zeitraum durchzuführen. Beides kann im Rahmen dieser Arbeit nicht mehr durchgeführt werden.
Heizkurve 1 folgt mit den Temperaturdaten des Sensors am Testgebäude einem klaren Verlauf. Bei der Auswertung mit den Daten des Hamburger Luftmessnetzes streut die Heizkurve ab einer Außentemperatur von 10 °C merklich. Dies könnte ein Hinweis darauf sein, dass der Sensor am Testgebäude nicht im Schatten platziert ist, wie es eigentlich

der Fall sein sollte. Durch die Sonneneinstrahlung erfasst der Sensor hohe Temperaturen und die Vorlauftemperatur wird abgesenkt. Im Diagramm mit den Werten des Luftmessnetzes führt dies zu einer unpassenden Position dieses Datenpunktes. Misst der Sensor beispielsweise eine Außentemperatur von 15 °C, so wird die Vorlauftemperatur auf 45 °C abgesenkt. Die eigentliche Außentemperatur liegt jedoch bei 6 °C. Die Kombination aus diesen beiden Werten (45 °C Vorlauftemperatur bei 6 °C statt 15 °C Außentemperatur) führt zu einer merklichen Abweichung des Heizkurvenverlaufs. Dies erklärt auch, warum die Datenpunkte erst ab einer Außentemperatur von 10 °C streuen. Die Sonneneinstrahlung ist im Winter geringer und übt somit auch einen geringeren Effekt auf den Sensor aus. Auch der Sonnenstand verändert sich zu dieser Jahreszeit und gegebenenfalls erreichen die Sonnenstrahlen im Winter den Sensor nicht. Eine weitere Verfälschung könnte durch die Wärmeabstrahlung des Gebäudes entstehen.
Um eine eindeutige Aussage zur Höhe der Absenkung der Vorlauftemperatur treffen zu können, wurden die Werte der Vorlauftemperatur beider Heizkurven für jedes Grad Celsius an Außentemperatur zu einem Durchschnittswert zusammengefasst. Anhand dieser durchschnittlichen Heizkurven lässt sich erkennen, dass Heizkurve 2 in beiden Auswertungen jeweils komplett unter Heizkurve 1 liegt. Insgesamt ergibt sich aus den Diagrammen eine durchschnittliche Absenkung der Vorlauftemperatur von rund 10 °C mit den Messdaten des Sensors am Testgebäude und 15 °C mit den Messdaten des Hamburger Luftmessnetzes.

6.2 Ergebnisse zu den Raumtemperaturen und Diskussion

In den nachfolgenden Abbildungen 6.8, 6.9 und 6.10 sind die Soll- und Isttemperaturen der drei Referenzräume während dem Zeitraum vom 21.04.21 um 17:00 Uhr bis zum 08.05.21 um 13:45 Uhr dargestellt. Grau hinterlegt sind die bereits genannten Zeiträume, die z.B. aufgrund von Datenübertragungsproblemen nicht in die Auswertung miteinbezogen werden, siehe auch Abschnitt 6.1.

In der ersten Abbildung 6.8 ist zu erkennen, dass in Raum 0.11 die geforderte Solltemperatur größtenteils erreicht werden konnte. Diese lag in der ersten Hälfte der finalen Testphase bei 22 °C und in der zweiten Hälfte bei 21 °C. Die zeitweise deutlich zu tiefen Raumtemperaturen sind auf das Öffnen von Fenstern zurückzuführen. Die Annahme, dass

ein deutliches Absenken der Temperatur über einen kurzen Zeitraum durch das Öffnen von Fenstern verursacht wird, wurde während der Testphase durch ein Telefonat mit dem Hausmeister bestätigt. Auch die kurzzeitigen Tiefstwerte bei der Solltemperatur sind eine Reaktion der Thermostate auf das Öffnen der Fenster. Durch ein Absenken der Solltemperatur soll vermieden werden, dass sich Thermostate durch die kälteren Temperaturen im Raum, die durch das Lüften im Raum erzielt werden, weiter öffnen und den Raum unnötig aufheizen. In Abbildung 6.8 wird jedoch deutlich, dass die Thermostate nicht jeden Lüftungsvorgang erkennen. So geschehen am 28.04 gegen 21 Uhr und am 03.05.21 um circa 11 Uhr.

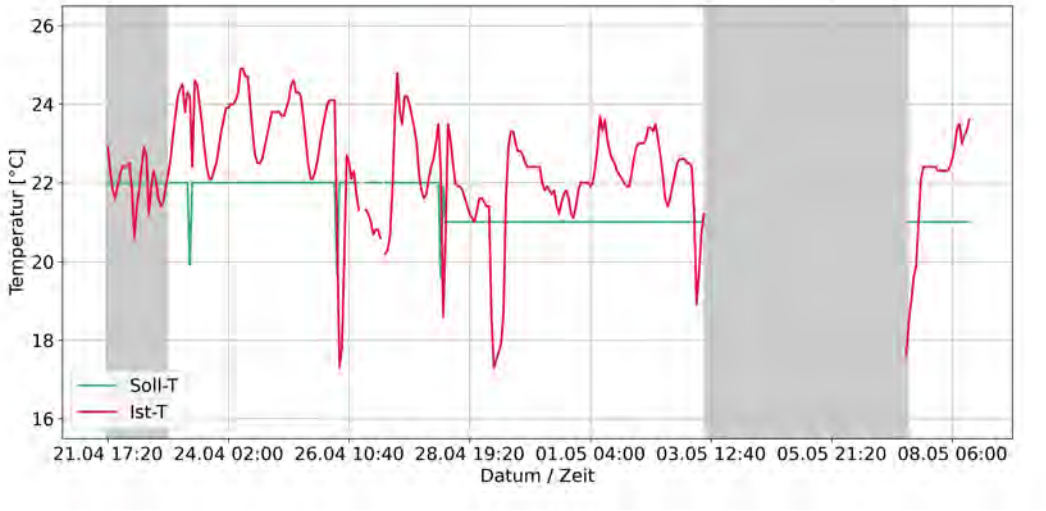


Abbildung 6.8: Finale Testphase: Temperaturen in Raum 0.11

Die Ergebnisse von Raum 0.32 sind in Abbildung 6.9 dargestellt. Auch hier konnten die verschiedenen Solltemperaturen erreicht werden. Im Gegensatz zu Raum 0.11 gilt es hier wieder, die Temperatur innerhalb des Toleranzbereichs zu erreichen. Dieser liegt bis 2 °C unter der eingestellten Solltemperatur und ist in der Abbildung grün gefärbt. Die eingestellte Solltemperatur beträgt während der finalen Testphase anfänglich 24 °C, danach über einen längeren Zeitraum 20 °C und gegen Ende 22 °C. Während der niedrigen Solltemperatur von 20 °C liegt die Raumtemperatur über dem geforderten Sollwert, während der höheren Solltemperaturen liegt sie im Toleranzbereich.

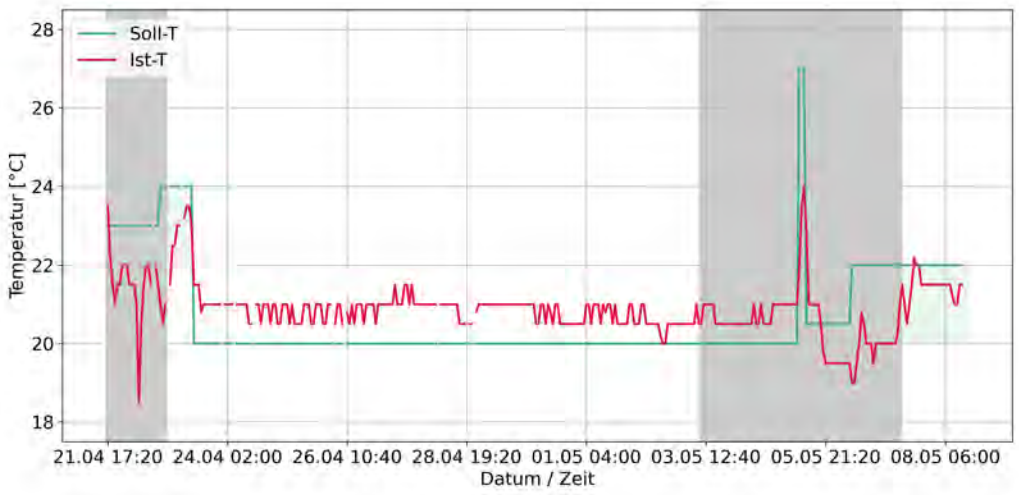


Abbildung 6.9: Finale Testphase: Temperaturen in Raum 0.32

Die Messdaten von Raum 0.12, in Abbildung 6.10 dargestellt, zeigen deutliche Temperaturschwankungen im Raum. Die eingestellte Solltemperatur liegt anfänglich bei 24 °C, danach bei 23 °C. Die Isttemperaturen liegen größtenteils im oder über dem Toleranzbereich, zeitweise aber auch 0,5 °C bis 1 °C unter dem Toleranzbereich.

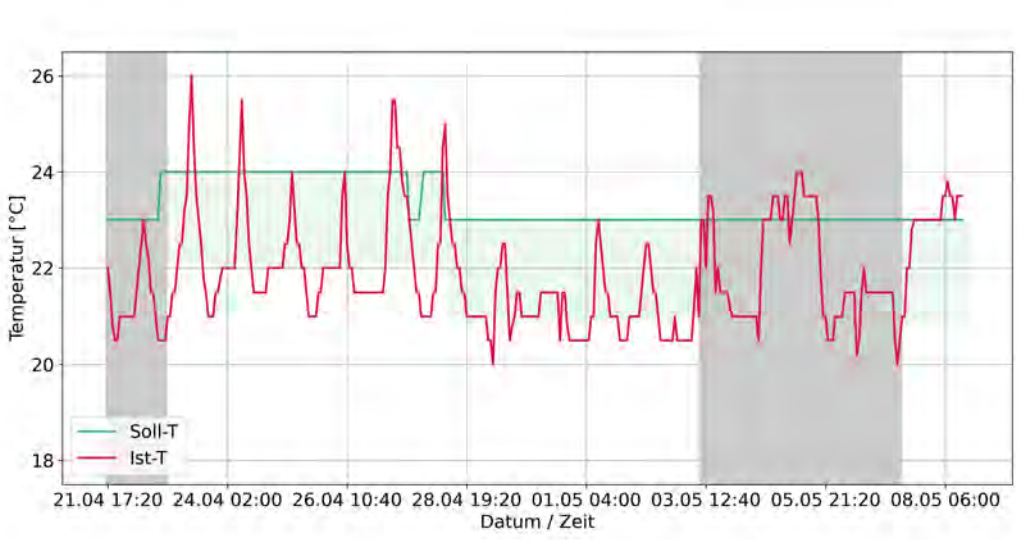


Abbildung 6.10: Finale Testphase: Temperaturen in Raum 0.12

In allen weiteren Räumen im Heizkreis konnten die geforderten Raumtemperaturen erreicht werden. Die Abbildungen sind in Anhang A.6 zu finden.

In Tabelle 6.3 sind zusätzlich die durchschnittlichen Ventilöffnungsgrade, sowie die durchschnittliche Solltemperatur während der finalen Testphase aufgeführt. Die Thermostatventile sind bei den beiden Referenzräumen 0.11 und 0.12 im Schnitt über 70 % geöffnet. Bei Referenzraum 0.32 sind es nur 18 %. Auch die Ventilöffnungsgrade der weiteren Räume im Heizkreis Nord liegen im Schnitt unter 25 %.

Tabelle 6.3: Durchschnittliche Solltemperaturen und Ventilöffnungsgrade der Räume im Heizkreis Nord während der finalen Testphase

0.11	0.12	0.13	0.15	0.18	0.21	0.32	1.10	1.11	1.12	1.13	1.18	1.21
21,4 °C	23,4 °C	15,7 °C	20,3 °C	22,0 °C	18,1 °C	20,2 °C	22,0 °C	22,0 °C	21,9 °C	17,8 °C	20,5 °C	21,7 °C
74,0 %	78,6 %	1,4 %	11,6 %	13,1 %	4,6 %	18,0 %	16,1 %	15,7 %	25,4 %	7,7 %	11,9 %	24,7 %

Diskussion der Ergebnisse

Die geforderte Raumtemperatur konnte, trotz Toleranzbereich, in Raum 0.12 zeitweise nicht erreicht werden. Der Raum unterliegt deutlichen Temperaturschwankungen. Dies ist zum einen auf die hohen solaren Gewinne zurückzuführen, die in Raum 0.12 durch den erhöhten Anteil an Fensterflächen erreicht werden. Die Temperaturschwankungen im Raum werden dadurch verstärkt, dass die Außentemperatur im Monat April, in welchen die finale Testphase hauptsächlich stattfand, größere Unterschiede zwischen Tag und Nacht aufweist, als dies in den Wintermonaten der Fall ist. Die zeitweise zu kalten Temperaturen im Referenzraum könnten darauf zurückzuführen sein, dass die Regelung zu langsam auf die Temperaturschwankungen reagiert. Durch das Aufheizen des Raumes in der Mitte des Tages wird die Vorlauftemperatur weit abgesenkt. Durch den hohen Anteil an Fensterfläche verliert der Raum die Wärme der Mittagssonne abends wieder. Ist die Vorlauftemperatur zu diesem Zeitpunkt noch zu niedrig, kann die geforderte Temperatur im Raum unter Umständen nicht erreicht werden. Eine Optimierung der Regelung kann dadurch erfolgen, dass, im Falle eines zu kalten Raumes, die Vorlauftemperatur statt um 3 °C um 4 °C stündlich angehoben wird.

Anhand der Daten aus Tabelle 6.3 lässt sich erkennen, dass sich nur die Ventilöffnungsgrade der zwei Referenzräume 0.11 und 0.12 mit über 70 % deutlich öffnen. Die geforderte Solltemperatur von Raum 0.32 wurde während der finale Testphase auf niedrige 20 °C eingestellt. Dadurch nahm er nicht mehr die Rolle des Referenzraumes ein, wie an der durchschnittlichen Ventilöffnung von 18 % zu erkennen ist. Der geringe Ventilöffnungs-

grad von unter 25 % lässt die Vermutung zu, dass für diese Räume die Vorlauftemperatur noch immer zu hoch eingestellt ist. Es gilt zu prüfen, ob die Wärmeversorgung dieser «Schlechträume» durch geeignete Maßnahmen, wie beispielsweise ein erneuter hydraulischer Abgleich oder die Erweiterung durch einen zusätzlichen Heizkörper, verbessert werden kann. Kann eine Verbesserung erzielt werden, so kann unter Umständen mit einer weiteren Absenkung der Vorlauftemperatur gerechnet werden. Weiter wurde durch die gesamten Testphasen deutlich, dass die neue Regelung fehleranfällig ist. Es konnten mehrmals aufgrund von Verbindungsproblemen nicht die aktuellen Daten der Thermostate übermittelt werden. Für eine dauerhafte Installation müsste dies sichergestellt werden.

6.3 Fazit

Für die Auswertung der Ergebnisse zur Vorlauftemperatur wurden die Heizkurven der ursprünglichen und der neuen Regelung miteinander verglichen. Die Auswertung erfolgte je einmal mit den Messdaten des Sensors am Testgebäude und mit den Messdaten des Hamburger Luftmessnetzes. Bei der Heizkurve der neuen Regelung konnte bei beiden Auswertungen eine deutliche Streuung der Datenpunkte festgestellt werden. Grund dafür sind die großen Temperaturunterschiede zwischen Tag und Nacht während der finalen Testphase. Die ursprüngliche Heizkurve weist nur eine Streuung bei der Auswertung mit den Messdaten des Hamburger Luftmessnetze auf, und dies erst ab einer Außentemperatur von 10 °C. Dies deutet daraufhin, dass der Sensor am Testgebäude nicht vorschriftsgemäß im Schatten angebracht ist. Durch einen Vergleich der Durchschnittswerte der beiden Heizkurven konnte eine Absenkung der Vorlauftemperatur von rund 10 °C bzw. 15 °C festgestellt werden. Es lässt sich demnach eindeutig eine Tendenz erkennen, dass die Wärmeversorgung mit einer verringerten Vorlauftemperatur möglich ist. Für eine exakteres Ergebnis müsste die Regelung über einen längeren Zeitraum während der kälteren Monate der Heizperiode getestet werden. Dies war im Rahmen dieser Arbeit aufgrund der zeitlichen Planung und des Starttermins nicht möglich.

Weiter sollte überprüft werden, ob alle Räume im betrachteten Heizkreis, insbesondere die Referenzräume, während der Testphase ihre geforderten Raumtemperaturen erreichen konnten. Dies ist bei Referenzraum 0.12 zeitweise nicht der Fall. Die Regelung reagiert zu langsam auf die starken Temperaturunterschiede im Raum. Die Isttemperatur liegt im Schnitt zwar über 23 °C und die Abweichungen sind mit 0,5 °C bis 1 °C unter Toleranzbereich relativ gering. Dennoch kann das erwähnte Ergebnis zur Vorlauftemperatur

auch aufgrund dieses Ergebnisses nur als eine erste Tendenz gewertet werden, dass eine Absenkung der Vorlauftemperatur möglich ist. Die Regelung sollte im Fall weiterer Untersuchungen für ein exakteres Ergebnis dahingehend optimiert werden, dass eine raschere Anhebung der Vorlauftemperatur, im Falle eines zu kalten Referenzraumes, erfolgt. Bei allen weiteren Räumen im Heizkreis Nord wurden keine zu tiefen Temperaturen festgestellt. Es konnte außerdem gezeigt werden, dass die durchschnittlichen Ventilöffnungsgrade in jenen Räumen während der finalen Testphase im Schnitt unter 25 % lagen. Es gilt zu prüfen, ob sich Maßnahmen zur Verbesserung der Wärmeversorgung in den Referenzräumen lohnt, da unter Umständen eine weitere Reduktion der Vorlauftemperatur möglich ist.
Die Regelung, wie sie für das Testgebäude entworfen wurde, ist aufgrund von Verbindungsproblemen fehleranfälliger als die ursprüngliche Regelung. Sie ist daher für eine dauerhafte Installation eher ungeeignet. Eine bessere Verwendung kann darin liegen, die Regelung zu nutzen, um die Parameter einer bestehenden Heizkurve anzupassen. Mittels einer länger andauernden Testphase in den kälteren Wintermonaten und der bereits genannten Optimierung der Regelung kann die tatsächlich benötigte Vorlauftemperatur im Heizkreis bestimmt werden. Ist diese niedriger als in der ursprünglichen Heizkurve, kann diese nach unten korrigiert werden. Das entworfene Programm zur Regelung lässt sich im Groben auch bei anderen Gebäuden anwenden, sofern die Referenzräume und das Verhalten der Thermostate bekannt ist.

7 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde eine bedarfsgeführte Regelung der Vorlauftemperatur anhand eines Testgebäudes entworfen und getestet. Dafür wurde ein «Schlechtraum» basiertes Heizkonzept angewandt. Die Vorlauftemperatur wird so weit abgesenkt, dass der «am schlechtesten mit Wärme versorgte» Raum im jeweiligen Heizkreis noch ausreichend beheizt wird. Im Rahmen einer vorgegangenen Studienarbeit wurde für einen Heizkreis im Testgebäude bereits ein Referenzraum bestimmt.
Zunächst war es notwendig, das Verhalten der Thermostate im Referenzraum anhand von Sprungantworten auszuwerten. Dafür wurde die Vorlauftemperatur sprunghaft angehoben und gesenkt und der Verlauf der Ventilöffnungen der Thermostate als Antwort auf diesen Sprung analysiert. Die Reaktion der Thermostate wies kein geeignetes Verhalten auf, um aus der Sprungantwort direkt den passenden Standardregler ablesen zu können. Es wurde infolgedessen ein eigenes Programm in der Programmiersprache Python umgesetzt. Der Bedarf, und damit auch die Höhe der Vorlauftemperatur, orientiert sich am Ventilöffnungsgrad. Dieser soll im Referenzraum bei mindestens 80 % liegen. Liegt der tatsächliche Ventilöffnungsgrad unter 80 %, so wird die Vorlauftemperatur abgesenkt. Dadurch verringert sich zunächst die Raumtemperatur. Dies führt dazu, dass sich die Thermostatventile öffnen. Liegt der Ventilöffnungsgrad über 80 %, so kann dies auf eine Unterversorgung des Raumes hindeuten. In diesem Fall wird Vorlauftemperatur angehoben, sofern die Raumtemperaturen unter dem geforderten Sollwert liegen. Andernfalls bleibt die Vorlauftemperatur konstant.
Während der ersten Testphasen wurde deutlich, dass es keinen eindeutigen «kältesten» Raum gibt, der zu jedem Zeitpunkt als Referenzraum geeignet ist. Daraufhin wurden zwei weitere Räume bestimmt, die als Referenzräume geeignet sind. Die Regelung wurde dahingehend angepasst. Je nach aktuellem Wärmeverhalten entscheidet die Regelung stündlich anhand des Ventilöffnungsgrads und der Raumtemperatur, welcher Raum die Rolle des Referenzraums übernimmt. Dabei konnte eine Eigenheit der Thermostate der Firma Fourdeg festgestellt werden. Diese Öffnen sich erst ab einer Regeldifferenz von 1.5 °C in der

Raumtemperatur bis zu den geforderten 80 %. Um diesen Fehler auszugleichen, wurde die Temperatur in den beiden Referenzräumen mit den Fourdeg Thermostaten 2 °C höher als die gewünschte Solltemperatur gestellt. Der Regelalgorithmus wurde dahingehend angepasst, dass diese Abweichung toleriert wird.
Aus den Messdaten der Vorlauftemperatur der finalen Testphase wurde eine neue Heizkurve erstellt und diese mit der ursprünglichen Heizkurve der außentemperaturgeführten Regelung verglichen. Die Auswertung erfolgte einmal mit den Messdaten für die Außentemperatur vom Sensor am Testgebäude und einmal mit den Messdaten einer Wetterstation. Beide Auswertungen der neuen Heizkurve wiesen mehrheitlich tiefere Temperaturen auf als die ursprünglichen Heizkurve. Aufgrund einer deutlichen Streuung der Datenpunkte gestaltete es sich als schwierig, eine genaue Aussage darüber zu treffen, wie weit die Vorlauftemperatur abgesenkt werden konnte. Daraufhin wurden die Messdaten der Vorlauftemperatur zu durchschnittlichen Werten zusammengefasst und erneut zwei Heizkurven erstellt. Es konnte eine durchschnittliche Absenkung der Vorlauftemperatur von rund 10 °C bzw. 15 °C festgestellt werden. Dieses Ergebnis kann als eine erste Tendenz gewertet werden, dass eine Wärmeversorgung mit einer geringeren Vorlauftemperatur möglich ist. Anhand der durchschnittlichen Ventilöffnungen der Räume während der finalen Testphase ließ sich außerdem feststellen, dass zwei der Referenzräume eine deutlich schlechtere Wärmeversorgung erhalten, als die restlichen Räume im Heizkreis. Durch eine Optimierung der Wärmeversorgung in jenen Räumen ließe sich unter Umständen eine zusätzliche Absenkung der Vorlauftemperatur realisieren.
Weiter gilt es zu prüfen, ob sich Eindeutigkeit der neuen Heizkurve durch eine längere Testphase in den kälteren Wintermonaten verbessert. Für den Heizkreis Nord kann die Heizkurve dementsprechend optimiert werden. Für eine dauerhafte Installation ist die neue Regelung aufgrund von häufigen Verbindungsproblemen zu fehleranfällig. Sie kann aber in den anderen Heizkreisen im Testgebäude ebenfalls dazu genutzt werden, die Heizkurven zu optimieren. Sind auch diese zu hoch eingestellt, kann langfristig eine Reduktion der Temperatur vom Fernwärmeanbieter in Betracht gezogen werden.

Literatur

[Bög11] Alfred Böge. *Handbuch Maschinenbau: Grundlagen und Anwendungen der Maschinenbau-Technik ; mit 412 Tabellen*. 20., überarb. und aktualisierte Aufl. Wiesbaden: Vieweg + Teubner, 2011. ISBN: 9783834898982. DOI: 10.1007/978-3-8348-9898-2.

[Bos21] Bosch Thermotechnik GmbH. *Einsatzbereiche der raumtemperaturgeführten Regelung*. 12.04.2021. URL: https://www.heizung-steuern.com/de/de/raumtemperaturgefuehrte-regelung/.

[Bun19] Bundesministerium für Wirtschaft und Energie. *Energiedaten: Gesamtausgabe*. Hrsg. von Bundesministerium für Wirtschaft und Energie. 2019. URL: https://www.bmwi.de/Redaktion/DE/Downloads/Energiedaten/energiedaten-gesamt-pdf-grafiken.pdf?__blob=publicationFile&v=40.

[CC421] CC4E. *Interne Dokumente «Smart Pro HeaT»*. 6.07.2021.

[Dar21] Dark Sky. *Weather*. 23.05.2021. URL: https://darksky.net/forecast/53.5075,9.9964/ca12/en.

[DWD21] DWD - Deutscher Wetterdienst. *Erläuterung der Klimagrafiken*. 21.06.2021. URL: https://www.dwd.de/DE/leistungen/kvo/editorial/erlaeuterungen.html.

[Fit13] Klaus Fitzner. *Raumklimatechnik*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. ISBN: 978-3-540-57181-0. DOI: 10.1007/978-3-540-68213-4.

[ham21] hamburg.de. *Hamburger Luftmessnetz – Meteorologie – Temperatur – FHH*. 21.05.2021. URL: http://luft.hamburg.de/clp/temperatur/clp1/.

[Hee20] Alice Heeb. "Smart Pro HeaT: Ermittlung eines Referenzraumes zur wärmebedarfsgerechten Regelung der Vorlauftemperatur". Studienarbeit. Hamburg: Hochschule für angewandte Wissenschaften, 2020.

[Heß20] Wolfgang Heße. *Energieeffiziente Wärmeversorgung von Gebäuden*. Wiesbaden: Springer Fachmedien Wiesbaden, 2020. ISBN: 978-3-658-27570-9. DOI: 10.1007/978-3-658-27571-6.

[Koc94] Peter Kocher. "HLH: Lüftung, Klima, Heizung, Sanitär, Gebäudetechnik: Zeitschrift des Vereins Deutscher Ingenieure: Organ der VDI-Gesellschaft Technische Gebäudeausüstung (VDI-TGA)". In: Vol. 45, No. 1 (1994), S. 7–9.

[LK20] Gunter Lauckner und Jörn Krimmling. *Raum- und Gebäudeautomation für Architekten und Ingenieure*. Wiesbaden: Springer Fachmedien Wiesbaden, 2020. ISBN: 978-3-658-30142-2. DOI: 10.1007/978-3-658-30143-9.

[LSD05] Z. Liao, M. Swainson und A. L. Dexter. "On the control of heating systems in the UK". In: *Building and Environment* 40.3 (2005), S. 343–351. ISSN: 03601323. DOI: 10.1016/j.buildenv.2004.05.014.

[Man+18] Heinz Mann u. a. *Einführung in die Regelungstechnik*. 12., neu bearbeitete Auflage. München: Hanser, 2018. ISBN: 978-3-446-45694-5. DOI: 10.3139/9783446456945. URL: https://www.hanser-elibrary.com/doi/book/10.3139/9783446456945.

[ØS16] Dorte Skaarup Østergaard und Svend Svendsen. "Theoretical overview of heating power and necessary heating supply temperatures in typical Danish single-family houses from the 1900s". In: *Energy and Buildings* 126 (2016), S. 375–383. ISSN: 03787788. DOI: 10.1016/j.enbuild.2016.05.034.

[Phi19] Hans-Werner Philippsen. *Einstieg in die Regelungstechnik mit Python*. München: Carl Hanser, 2019. ISBN: 978-3-446-45157-5.

[SW13] Kai Schild und Wolfgang M. Willems. *Wärmeschutz*. Wiesbaden: Springer Fachmedien Wiesbaden, 2013. ISBN: 978-3-658-02570-0. DOI: 10.1007/978-3-658-02571-7.

[Umw14] Umweltbundesamt. *Der Weg zum klimaneutralen Gebäudebestand*. Hrsg. von Umweltbundesamt. 2014. URL: https://www.umweltbundesamt.de/publikationen/der-weg-klimaneutralen-gebaeudebestand.

A Anhang

A.1 R&I-Fließschema Testgebäude

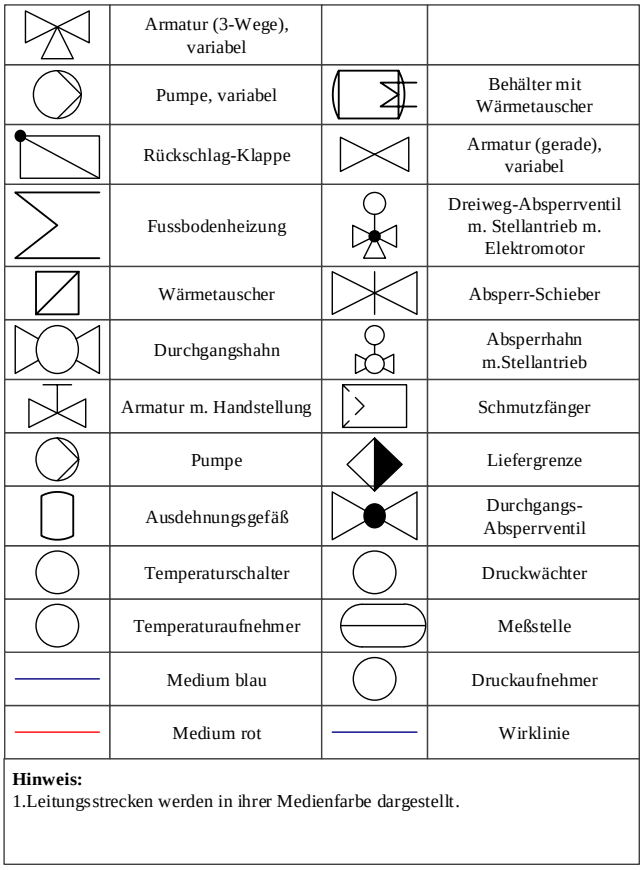


Abbildung A.1: R&I-Fließschema Testgebäude Legende [CC421]

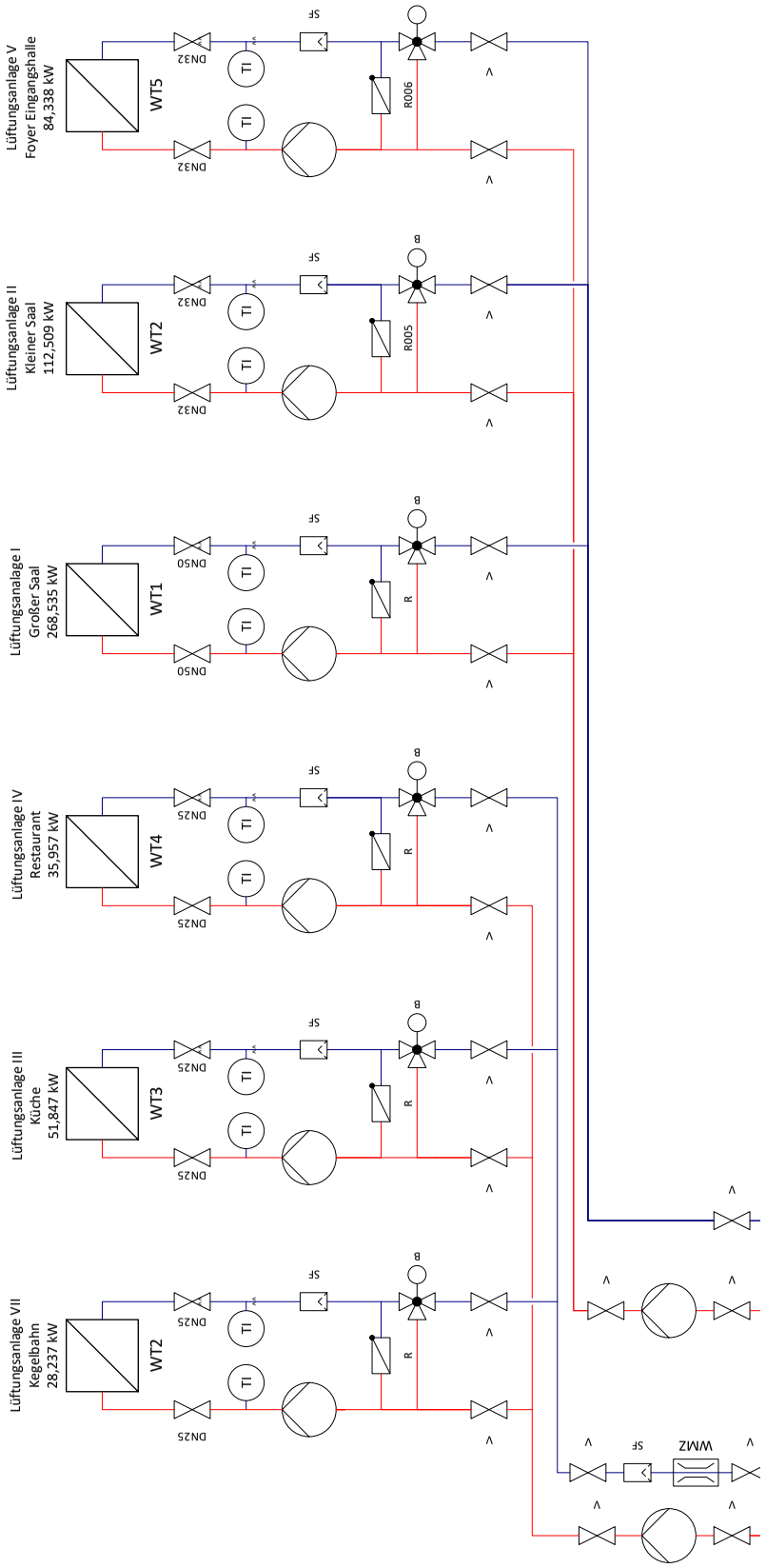


Abbildung A.2: R&I-Fließschema Testgebäude Teil 1 [CC421]

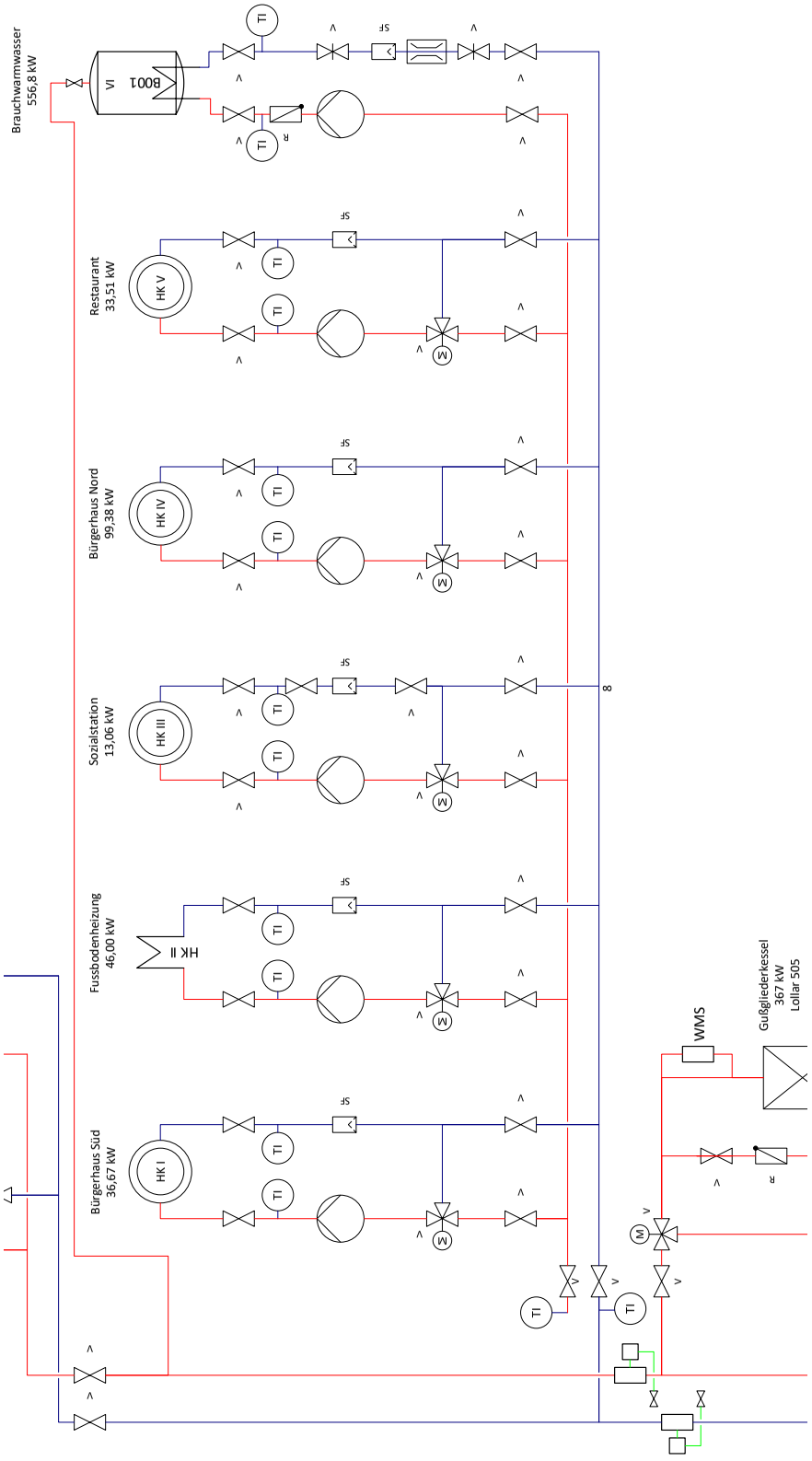


Abbildung A.3: R&I-Fließschema Testgebäude Teil 2 [CC421]

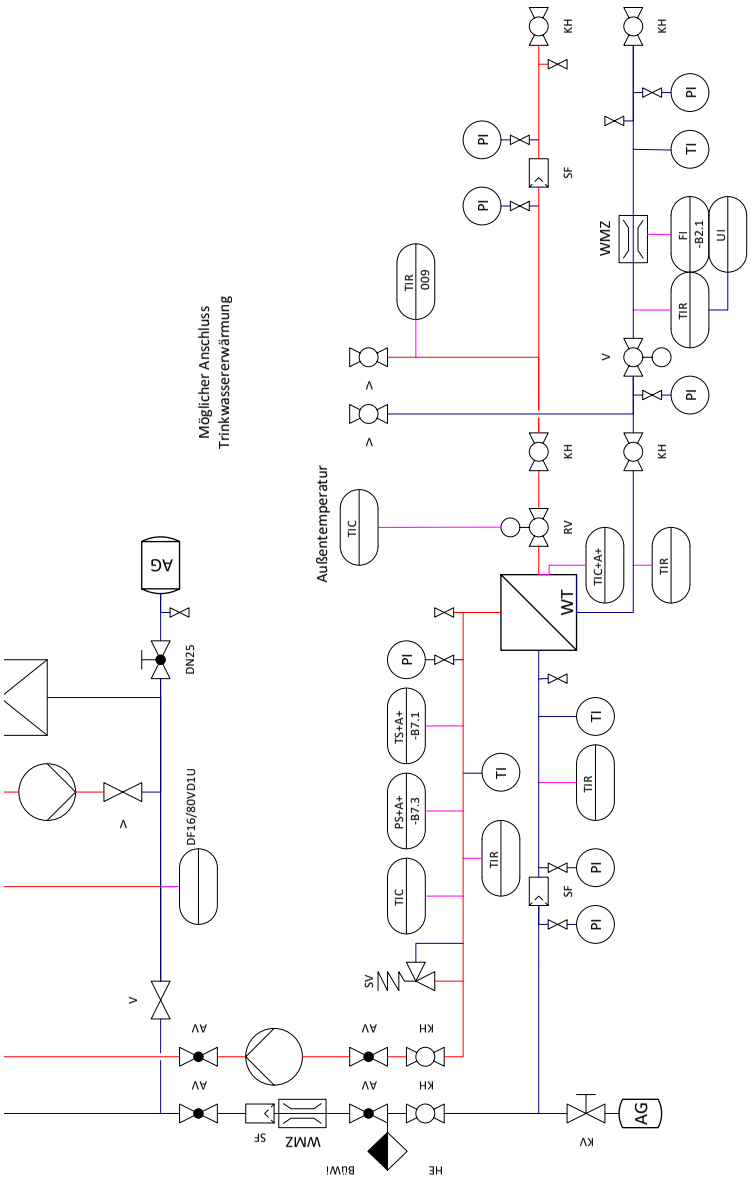


Abbildung A.4: R&I-Fließschema Testgebäude Teil 3 [CC421]

A.2 Grundrisse Testgebäude EG/OG

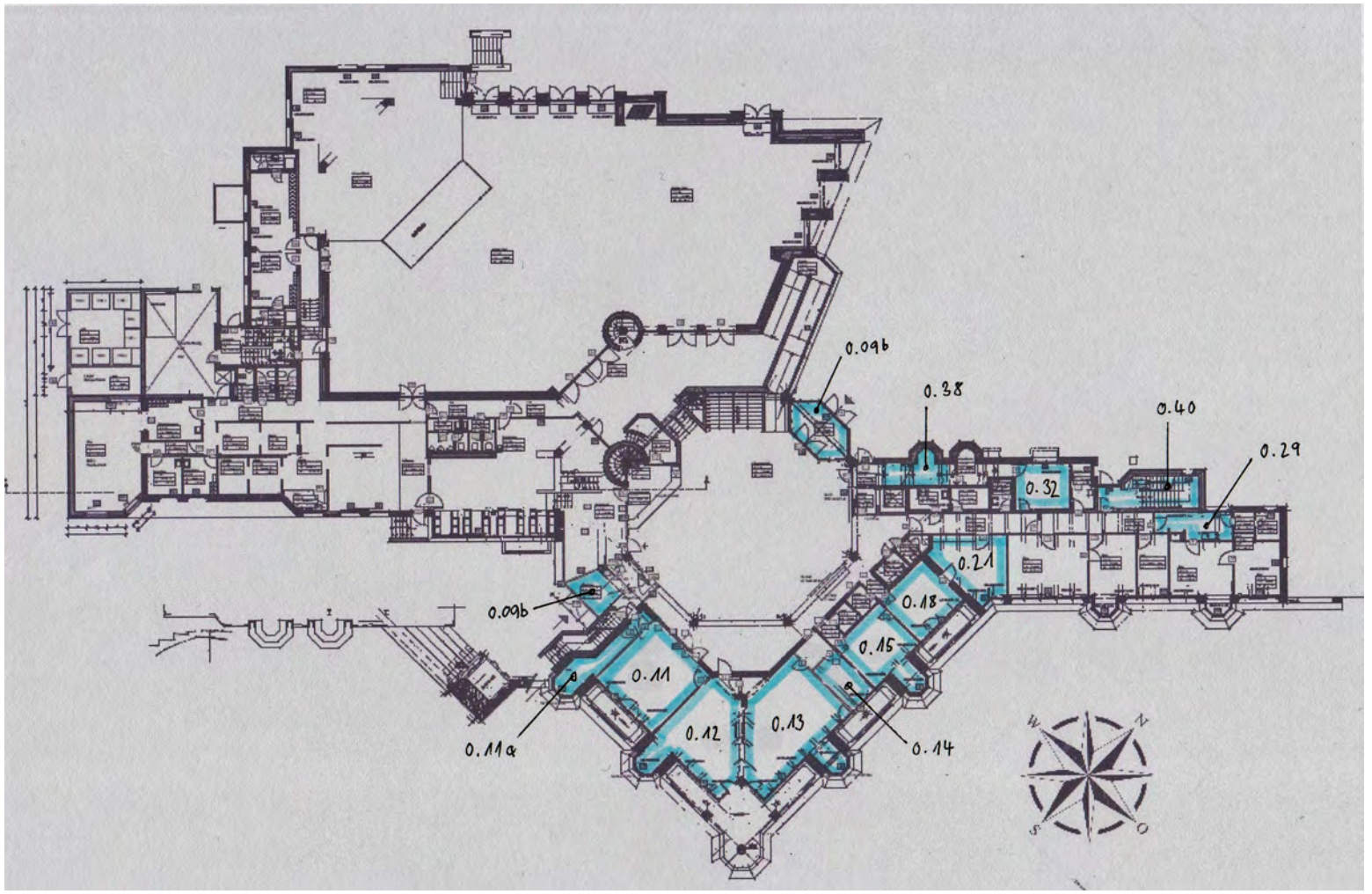


Abbildung A.5: Grundriss Testgebäude EG [CC421]

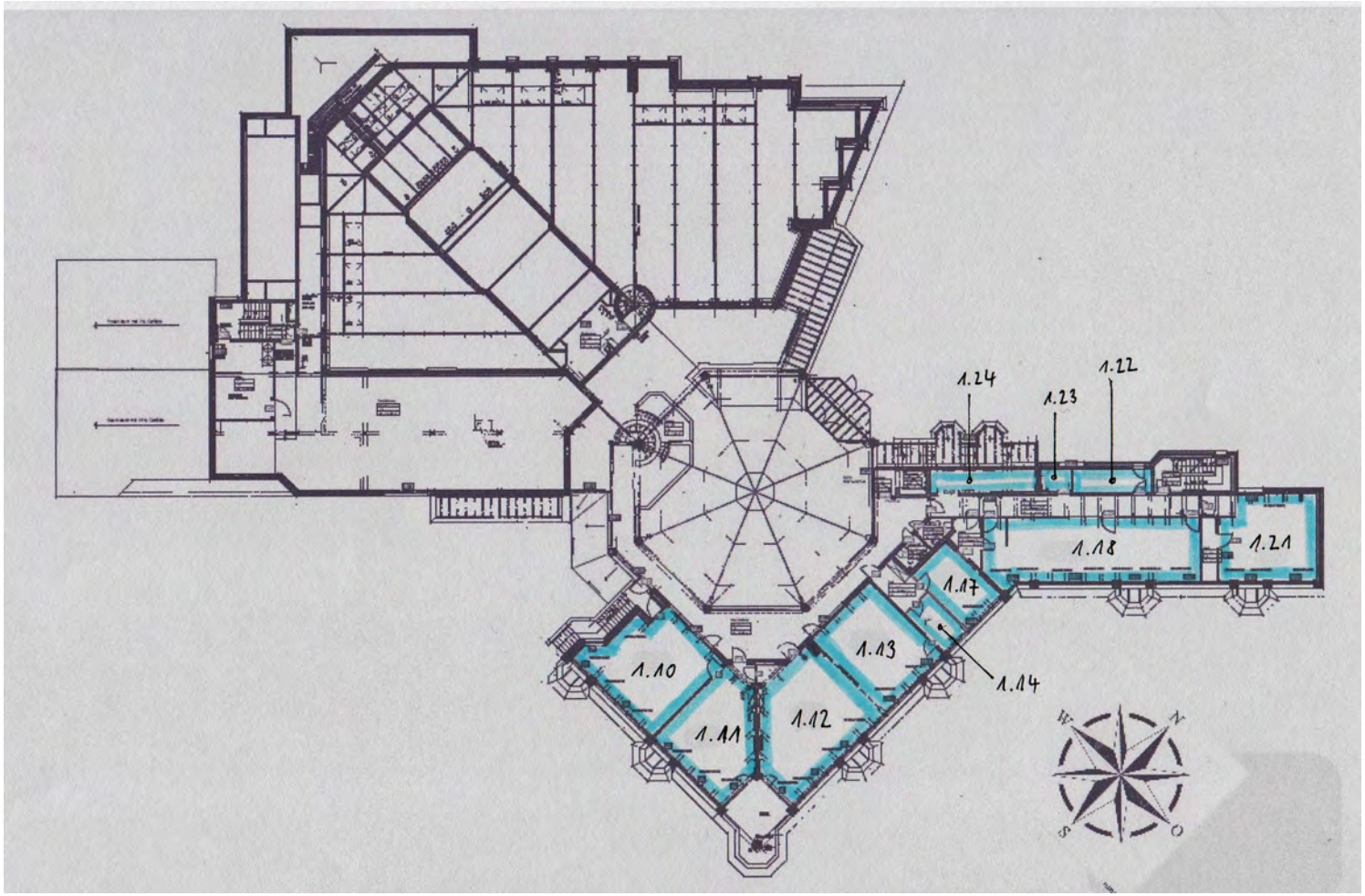


Abbildung A.6: Grundriss Testgebäude OG [CC421]

A.3 Auswertung Sprungantwort Thermostat rechts

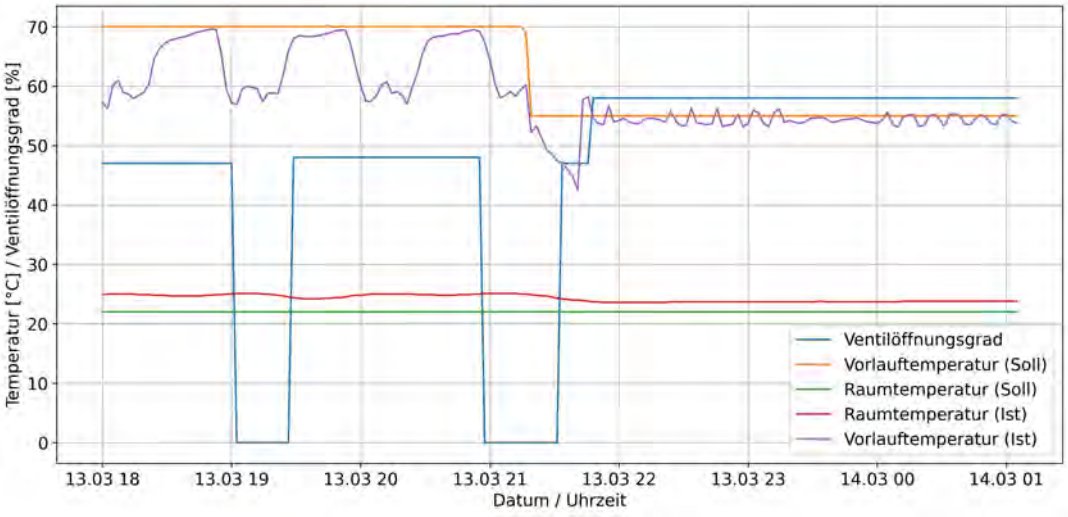


Abbildung A.7: Sprungantwort Thermostat rechts

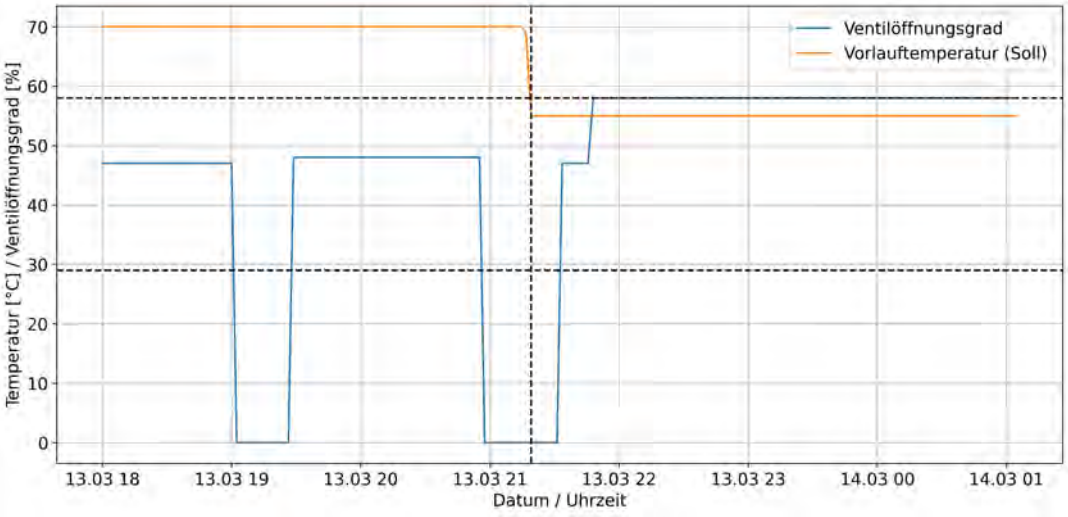


Abbildung A.8: Teilauswertung Sprungantwort Thermostat rechts

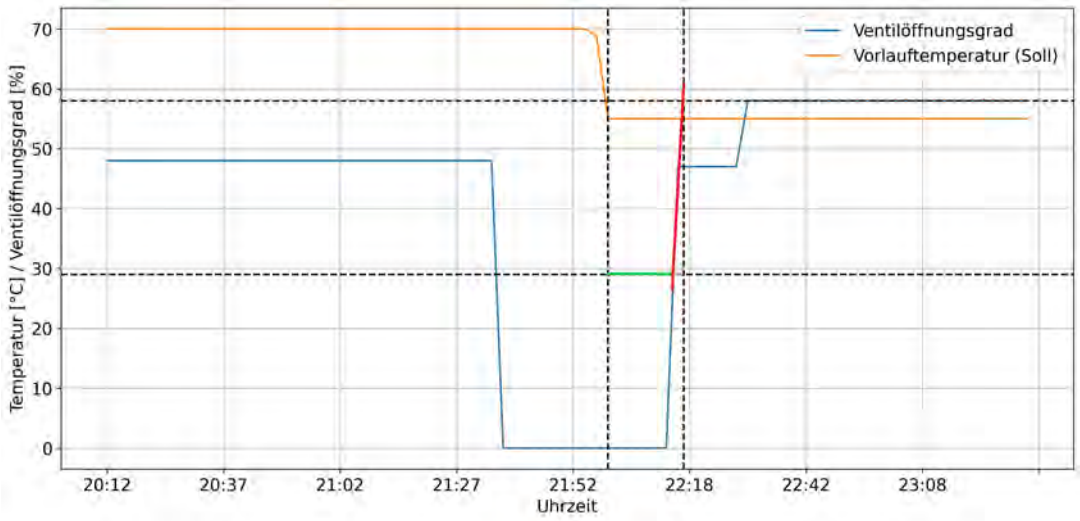


Abbildung A.9: Auswertung Sprungantwort Thermostat rechts

A.4 Programme

A.4.1 Referenzraum 0.11

```
# Imports
import time
from influxdb import InfluxDBClient
import influxdb.resultset
import requests

# Umgebungsvariablen
host = os.environ['INFLUX_HOST']
username = os.environ['INFLUX_USERNAME']
password = os.environ['INFLUX_PASSWORD']
database = os.environ['INFLUX_DATABASE']
INFLUX_WRITE_URL = f'http://{INFLUX_HOST}:8086' \
                   f'/write?db={INFLUX_DATABASE}' \
                   f'&precision=s'
INFLUX_AUTH = (INFLUX_USERNAME, INFLUX_PASSWORD)
influx_client = InfluxDBClient(
    host=INFLUX_HOST,
    username=INFLUX_USERNAME, password=INFLUX_PASSWORD,
    database=INFLUX_DATABASE)

# Hype Query strings
INFLUX_MEASUREMENT_011_RECHTS = "Thermostat_0.11_Seminarraum_rechts"
INFLUX_FIELD_XR = "current_valve_position"
INFLUX_MEASUREMENT_011_LINKS = "Thermostat_0.11_Seminarraum_links"
INFLUX_FIELD_XL = "current_valve_position"
INFLUX_FIELD_TEMP_SOLL = "current_set_point"
INFLUX_FIELD_TEMP_IST = "current_temperature"
INFLUX_MEASUREMENT_WAGO_SPS = "Wago_SPS"
INFLUX_FIELD_VLT_IST = "Hzk_BH_Nord_VLT"
INFLUX_FIELD_VLT_SEK = "VLT_sek"
INFLUX_MEASUREMENT_CCU3 = "ccu3"
INFLUX_FIELD_VLT_SOLL_HAW = "HK_Nord_Sollwert_VLT_HAW"

def main():
    # Gather data from InfluxDB
    x_r, x_l, temp_soll, temp_ist, vlt_ist, vlt_soll_haw, vlt_sek =
                                        get_data()
```

```
# Anfangssollwert setzen
if vlt_soll_haw == 0:
    vlt_soll_haw = vlt_ist
    vlt_soll_neu = vlt_soll_haw
    write_vlt_soll_in_DB(vlt_soll_neu)

# Temperaturdifferenz
dt = temp_soll - temp_ist
# Optimierung der Vorlauftemperatur
# Schleife 1
i = 2
while (x_r >= 80) and (x_l >= 80) and (dt > 0):
    if dt > 3:
        print("Schlafe eine Stunde wegen geoeffnetem Fenster")
        time.sleep(3600)
    vlt_soll_neu = vlt_soll_haw + i
    if vlt_soll_neu > vlt_sek:
        vlt_soll_neu = vlt_sek
    if vlt_soll_neu < 30:
        vlt_soll_neu = 30
    write_vlt_soll_in_DB(vlt_soll_neu)
    print("Schlafe in Schleife 1 fuer eine halbe Stunde.", time.
                                        strftime("%d.%m.%Y %H:%M:%S"
                                        ))
    time.sleep(1800)
    print("Neue Werte:")
    x_r, x_l, temp_soll, temp_ist, vlt_ist, vlt_soll_haw, vlt_sek =
                                        get_data()
    dt = temp_soll - temp_ist
    print("dT:", dt)

# Schleife 2
while (x_r >= 80) and (x_l >= 80) and (dt <= 0):
    print("Schlafe in Schleife 2 fuer eine Stunde.", time.strftime("
                                        %d.%m.%Y %H:%M:%S"))
    time.sleep(3600)
    print('Neue Werte:')
    x_r, x_l, temp_soll, temp_ist, vlt_ist, vlt_soll_haw, vlt_sek =
                                        get_data()
    dt = temp_soll - temp_ist
```

```
        print("dT:", dt)

    # Schleife 3
    i = 2
    while (x_r >= 40) and (x_r < 80) and (x_l >= 80):
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print("Schlafe in Schleife 3 fuer eine Stunde.", time.strftime("
                                                %d.%m.%Y %H:%M:%S"))
        time.sleep(3600)
        print("Neue Werte:")
        x_r, x_l, temp_soll, temp_ist, vlt_ist, vlt_soll_haw, vlt_sek =
                                                get_data()
        dt = temp_soll - temp_ist
        print("dT:", dt)

    # Schleife 4
    i = 2
    while (x_l >= 40) and (x_l < 80) and (x_r >= 80):
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print("Schlafe in Schleife 4 fuer eine Stunde.", time.strftime("
                                                %d.%m.%Y %H:%M:%S"))
        time.sleep(3600)
        print("Neue Werte:")
        x_r, x_l, temp_soll, temp_ist, vlt_ist, vlt_soll_haw, vlt_sek =
                                                get_data()
        dt = temp_soll - temp_ist
        print("dT:", dt)

    # Schleife 5
    i = 2
    while (x_r >= 40) and (x_r < 80) and (x_l >= 40) and (x_l < 80):
```

```
            vlt_soll_neu = vlt_soll_haw - i
            if vlt_soll_neu > vlt_sek:
                vlt_soll_neu = vlt_sek
            if vlt_soll_neu < 30:
                vlt_soll_neu = 30
            write_vlt_soll_in_DB(vlt_soll_neu)
            print("Schlafe in Schleife 5 fuer eine Stunde.", time.strftime("
                                                %d.%m.%Y %H:%M:%S"))
            time.sleep(3600)
            print('Neue Werte:')
            x_r, x_l, temp_soll, temp_ist, vlt_ist, vlt_soll_haw, vlt_sek =
                                                get_data()
            dt = temp_soll - temp_ist
            print("dT:", dt)

        # Schleife 6
        i = 4
        while (x_r < 40) or (x_l < 40):
            vlt_soll_neu = vlt_soll_haw - i
            if vlt_soll_neu > vlt_sek:
                vlt_soll_neu = vlt_sek
            if vlt_soll_neu < 30:
                vlt_soll_neu = 30
            write_vlt_soll_in_DB(vlt_soll_neu)
            print("Schlafe in Schleife 6 fuer eine Stunde.", time.strftime("
                                                %d.%m.%Y %H:%M:%S"))
            time.sleep(3600)
            print('Neue Werte:')
            x_r, x_l, temp_soll, temp_ist, vlt_ist, vlt_soll_haw, vlt_sek =
                                                get_data()
            dt = temp_soll - temp_ist
            print("dT:", dt)

# notwendig fuer 'gather data from influxDB'
def read_last_point_from_influx_field(measurement, field_key):
    """
    Request last point of a field from an InfluxDB measurement
    :param measurement: Name of the measurement
    :param field_key: Field key to query
    :return: Latest value of measurement
    """
```

```
    select_query = f'select last("{field_key}") from "{measurement}"'
    result_query: influxdb.resultset.ResultSet = influx_client.query(
        query=select_query)
    result = list(result_query.get_points(measurement=measurement))[0]['
                                          last']
    return result


# Neuen Sollwert fuer VLT ueberschreiben
def write_vlt_soll_in_DB(vlt_soll_neu):
    name = "ccu3"
    value = vlt_soll_neu
    data_point = "HK_Nord_Sollwert_VLT_HAW"
    data = f'{name} {data_point}={value}'
    response = requests.post(INFLUX_WRITE_URL, data=data, auth=
                                          INFLUX_AUTH)
    try:
        response.raise_for_status()
    except requests.HTTPError:
        print(f"Error: Write request to InfluxDB for measurement '{name}
                                              '"
              f' failed: {response.status_code}: {response.reason}')
    print()
    print("data=", data, INFLUX_WRITE_URL)


def get_data():
    x_r = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_011_RECHTS,
        field_key=INFLUX_FIELD_XR)
    print("Ventil rechts:", x_r)
    x_l = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_011_LINKS,
        field_key=INFLUX_FIELD_XL)
    print("Ventil links:", x_l)
    temp_soll = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_011_RECHTS,
        field_key=INFLUX_FIELD_TEMP_SOLL)
    print("Soll-T:", temp_soll)
    temp_ist = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_011_RECHTS,
```

```
        field_key=INFLUX_FIELD_TEMP_IST)
    print("Ist-T:", temp_ist)
    print("dT:", temp_soll - temp_ist)
    vlt_ist = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_WAGO_SPS,
    field_key=INFLUX_FIELD_VLT_IST)
    print("VLT_ist:", vlt_ist)
    vlt_soll_haw = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_CCU3,
        field_key=INFLUX_FIELD_VLT_SOLL_HAW)
    print("VLT_soll", vlt_soll_haw)
    vlt_sek = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_WAGO_SPS,
    field_key=INFLUX_FIELD_VLT_SEK)
    print("VLT_sek", vlt_sek)
    return x_r, x_l, temp_soll, temp_ist, vlt_ist, vlt_soll_haw, vlt_sek



while True:
    if __name__ == '__main__':
        main()
```

A.4.2 Referenzraum 0.12

```
# Imports
import time
from influxdb import InfluxDBClient
import influxdb.resultset
import requests

# Umgebungsvariablen
host = os.environ['INFLUX_HOST']
username = os.environ['INFLUX_USERNAME']
password = os.environ['INFLUX_PASSWORD']
database = os.environ['INFLUX_DATABASE']
INFLUX_WRITE_URL = f'http://{INFLUX_HOST}:8086' \
                   f'/write?db={INFLUX_DATABASE}' \
                   f'&precision=s'
```

```
INFLUX_AUTH = (INFLUX_USERNAME, INFLUX_PASSWORD)
influx_client = InfluxDBClient(
    host=INFLUX_HOST,
    username=INFLUX_USERNAME, password=INFLUX_PASSWORD,
    database=INFLUX_DATABASE)

# Hype Query strings
INFLUX_MEASUREMENT_012_MITTE = "Thermostat_0.12_Seminarraum_mitte"
INFLUX_FIELD_XM = 'current_valve_position'
INFLUX_MEASUREMENT_012_LINKS = "Thermostat_0.12_Seminarraum_links"
INFLUX_FIELD_XL = 'current_valve_position'
INFLUX_MEASUREMENT_012_RECHTS = "Thermostat_0.
                                    12_Seminarraum_rechts_Erker"
INFLUX_FIELD_XR = 'current_valve_position'
INFLUX_FIELD_TEMP_SOLL = 'current_set_point'
INFLUX_FIELD_TEMP_IST = 'current_temperature'
INFLUX_MEASUREMENT_WAGO_SPS = "Wago_SPS"
INFLUX_FIELD_VLT_IST = 'Hzk_BH_Nord_VLT'
INFLUX_FIELD_VLT_SEK = 'VLT_sek'
INFLUX_MEASUREMENT_CCU3 = 'ccu3'
INFLUX_FIELD_VLT_SOLL_HAW = 'HK_Nord_Sollwert_VLT_HAW'

def main():
    # Gather data from InfluxDB
    x_r, x_l, x_m, temp_soll, temp_ist, vlt_ist, vlt_soll_haw, vlt_sek =
                                        get_data()

    # Anfangssollwert setzen
    if vlt_soll_haw == 0:
        vlt_soll_haw = vlt_ist
        vlt_soll_neu = vlt_soll_haw
        write_vlt_soll_in_DB(vlt_soll_neu)

    # Temperaturdifferenz
    dt = temp_soll - temp_ist
    # Optimierung der Vorlauftemperatur
    # Schleife 1
    i = 2
    while (x_r >= 75) and (x_l >= 80) and (x_m >= 80) and (dt > 0):
        if dt >= 3:
```

```
            print("Schlafe eine halbe Stunde wegen geoeffnetem Fenster."
                                                    )
            time.sleep(3600)
            break
        vlt_soll_neu = vlt_soll_haw + i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('Schlafe in Schleife 1 fuer eine Stunde.', time.strftime("
                                            %d.%m.%Y %H:%M:%S"))
        time.sleep(1800)
        print('Neue Werte:')
        x_r, x_l, x_m, temp_soll, temp_ist, vlt_ist, vlt_soll_haw,
                                            vlt_sek = get_data()
        dt = temp_soll - temp_ist
        print("dT:", dt)

    # Schleife 2
    while (x_r >= 75) and (x_l >= 80) and (x_m >= 80) and (dt <= 0):
        print('Schlafe in Schleife 2 fuer eine Stunde.', time.strftime("
                                            %d.%m.%Y %H:%M:%S"))
        time.sleep(3600)
        print('Neue Werte:')
        x_r, x_l, x_m, temp_soll, temp_ist, vlt_ist, vlt_soll_haw,
                                            vlt_sek = get_data()
        dt = temp_soll - temp_ist
        print("dT:", dt)

    # Schleife 3
    i = 2
    while (x_r >= 40) and (x_r < 75) and (x_m >= 40) and (x_m < 80) and
                                        (x_l >= 80):
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
```

```
        print('Schlafe in Schleife 3 fuer eine Stunde.', time.strftime("
                                                %d.%m.%Y %H:%M:%S"))
        time.sleep(3600)
        print('Neue Werte:')
        x_r, x_l, x_m, temp_soll, temp_ist, vlt_ist, vlt_soll_haw,
                                                vlt_sek = get_data()
        dt = temp_soll - temp_ist
        print("dT:", dt)

    # Schleife 4
    i = 2
    while (x_l >= 40) and (x_l < 80) and (x_m >= 40) and (x_m < 80) and
                                        (x_r >= 75):
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('Schlafe in Schleife 4 fuer eine Stunde.', time.strftime("
                                                %d.%m.%Y %H:%M:%S"))
        time.sleep(3600)
        print('Neue Werte:')
        x_r, x_l, x_m, temp_soll, temp_ist, vlt_ist, vlt_soll_haw,
                                                vlt_sek = get_data()
        dt = temp_soll - temp_ist
        print("dT:", dt)

    # Schleife 5
    i = 2
    while (x_r >= 40) and (x_r < 75) and (x_l >= 40) and (x_l < 80) and
                                        (x_m >= 80):
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('Schlafe in Schleife 5 fuer eine Stunde.', time.strftime("
                                                %d.%m.%Y %H:%M:%S"))
        time.sleep(3600)
```

```
        print('Neue Werte:')
        x_r, x_l, x_m, temp_soll, temp_ist, vlt_ist, vlt_soll_haw,
                                                vlt_sek = get_data()
        dt = temp_soll - temp_ist
        print("dT:", dt)

    # Schleife 6
    i = 2
    while (x_r >= 40) and (x_r < 75) and (x_m >= 80) and (x_l >= 80):
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('Schlafe in Schleife 6 fuer eine Stunde.', time.strftime("
                                                %d.%m.%Y %H:%M:%S"))
        time.sleep(3600)
        print('Neue Werte:')
        x_r, x_l, x_m, temp_soll, temp_ist, vlt_ist, vlt_soll_haw,
                                                vlt_sek = get_data()
        dt = temp_soll - temp_ist
        print("dT:", dt)

    # Schleife 7
    i = 2
    while (x_l >= 40) and (x_l < 80) and (x_m >= 80) and (x_r >= 75):
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('Schlafe in Schleife 7 fuer eine Stunde.', time.strftime("
                                                %d.%m.%Y %H:%M:%S"))
        time.sleep(3600)
        print('Neue Werte:')
        x_r, x_l, x_m, temp_soll, temp_ist, vlt_ist, vlt_soll_haw,
                                                vlt_sek = get_data()
        dt = temp_soll - temp_ist
        print("dT:", dt)
```

```
# Schleife 8
i = 2
while (x_m >= 40) and (x_m < 80) and (x_r >= 75) and (x_r >= 75):
    vlt_soll_neu = vlt_soll_haw - i
    if vlt_soll_neu > vlt_sek:
        vlt_soll_neu = vlt_sek
    if vlt_soll_neu < 30:
        vlt_soll_neu = 30
    write_vlt_soll_in_DB(vlt_soll_neu)
    print('Schlafe in Schleife 8 fuer eine Stunde.', time.strftime("
                                        %d.%m.%Y %H:%M:%S"))
    time.sleep(3600)
    print('Neue Werte:')
    x_r, x_l, x_m, temp_soll, temp_ist, vlt_ist, vlt_soll_haw,
                                        vlt_sek = get_data()
    dt = temp_soll - temp_ist
    print("dT:", dt)

# Schleife 9
i = 2
while (x_r >= 40) and (x_r < 75) and (x_l >= 40) and (x_l < 80) and
                                    (x_m >= 40) and (x_m < 80):
    vlt_soll_neu = vlt_soll_haw - i
    if vlt_soll_neu > vlt_sek:
        vlt_soll_neu = vlt_sek
    if vlt_soll_neu < 30:
        vlt_soll_neu = 30
    write_vlt_soll_in_DB(vlt_soll_neu)
    print('Schlafe in Schleife 9 fuer eine Stunde.', time.strftime("
                                        %d.%m.%Y %H:%M:%S"))
    time.sleep(3600)
    print('Neue Werte:')
    x_r, x_l, x_m, temp_soll, temp_ist, vlt_ist, vlt_soll_haw,
                                        vlt_sek = get_data()
    dt = temp_soll - temp_ist
    print("dT:", dt)

# Schleife 10
i = 4
while (x_r < 40) or (x_l < 40) or (x_m < 40):
```

```
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('Schlafe in Schleife 10 fuer eine Stunde.', time.strftime(
                                            "%d.%m.%Y %H:%M:%S"))
        time.sleep(3600)
        print('Neue Werte:')
        x_r, x_l, x_m, temp_soll, temp_ist, vlt_ist, vlt_soll_haw,
                                            vlt_sek = get_data()
        dt = temp_soll - temp_ist
        print("dT:", dt)

# Aktuelle Datenpunkte aus InfluxDB lesen
def read_last_point_from_influx_field(measurement, field_key):
    """
    Request last point of a field from an InfluxDB measurement
    :param measurement: Name of the measurement
    :param field_key: Field key to query
    :return: Latest value of measurement
    """
    select_query = f'select last("{field_key}") from "{measurement}"'
    result_query: influxdb.resultset.ResultSet = influx_client.query(
        query=select_query)
    result = list(result_query.get_points(measurement=measurement))[0]['
                                        last']
    return result


# Neuen Sollwert fuer VLT ueberschreiben
def write_vlt_soll_in_DB(vlt_soll_neu):
    name = "ccu3"
    value = vlt_soll_neu
    data_point = "HK_Nord_Sollwert_VLT_HAW"
    data = f'{name} {data_point}={value}'
    response = requests.post(INFLUX_WRITE_URL, data=data, auth=
                                        INFLUX_AUTH)
    try:
        response.raise_for_status()
```

```
    except requests.HTTPError:
        print(f"Error: Write request to InfluxDB for measurement '{name}
                                                    '"
              f' failed: {response.status_code}: {response.reason}')
    print()
    print("data=", data, INFLUX_WRITE_URL)


def get_data():
    x_r = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_012_RECHTS,
        field_key=INFLUX_FIELD_XR)
    print("Ventil rechts:", x_r)
    x_l = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_012_LINKS,
        field_key=INFLUX_FIELD_XL)
    print("Ventil links:", x_l)
    x_m = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_012_MITTE,
        field_key=INFLUX_FIELD_XM)
    print("Ventil mitte:", x_m)
    temp_soll = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_012_MITTE,
        field_key=INFLUX_FIELD_TEMP_SOLL)
    print("Soll-T:", temp_soll)
    temp_ist = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_012_MITTE,
        field_key=INFLUX_FIELD_TEMP_IST)
    print("Ist-T:", temp_ist)
    vlt_ist = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_WAGO_SPS,
    field_key=INFLUX_FIELD_VLT_IST)
    print("VLT_ist:", vlt_ist)
    vlt_soll_haw = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_CCU3,
        field_key=INFLUX_FIELD_VLT_SOLL_HAW)
    print("VLT_soll", vlt_soll_haw)
    vlt_sek = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_WAGO_SPS,
    field_key=INFLUX_FIELD_VLT_SEK)
    print("VLT_sek", vlt_sek)
```

```
    return x_r, x_l, x_m, temp_soll, temp_ist, vlt_ist, vlt_soll_haw,
                                            vlt_sek


while True:
    if __name__ == '__main__':
        main()
```

A.4.3 Referenzraum 0.32

```
# Imports
import time
from influxdb import InfluxDBClient
import influxdb.resultset
import requests

# Umgebungsvariablen
host = os.environ['INFLUX_HOST']
username = os.environ['INFLUX_USERNAME']
password = os.environ['INFLUX_PASSWORD']
database = os.environ['INFLUX_DATABASE']
INFLUX_WRITE_URL = f'http://{INFLUX_HOST}:8086' \
                   f'/write?db={INFLUX_DATABASE}' \
                   f'&precision=s'
INFLUX_AUTH = (INFLUX_USERNAME, INFLUX_PASSWORD)
influx_client = InfluxDBClient(
    host=INFLUX_HOST,
    username=INFLUX_USERNAME, password=INFLUX_PASSWORD,
    database=INFLUX_DATABASE)

# Hype Query strings
INFLUX_MEASUREMENT_032 = "Thermostat_0.32_Buero_Haustechnik"
INFLUX_FIELD_X = 'current_valve_position'
INFLUX_FIELD_TEMP_SOLL = 'current_set_point'
INFLUX_FIELD_TEMP_IST = 'current_temperature'
INFLUX_MEASUREMENT_WAGO_SPS = "Wago_SPS"
INFLUX_FIELD_VLT_IST = 'Hzk_BH_Nord_VLT'
INFLUX_FIELD_VLT_SEK = 'VLT_sek'
INFLUX_MEASUREMENT_CCU3 = 'ccu3'
```

```
INFLUX_FIELD_VLT_SOLL_HAW = 'HK_Nord_Sollwert_VLT_HAW'

def main():
    # Gather data from InfluxDB
    x, temp_soll, temp_ist, vlt_soll_haw, vlt_sek, vlt_ist = get_data()

    # Anfangssollwert setzen
    if vlt_soll_haw == 0:
        vlt_soll_haw = vlt_ist
        vlt_soll_neu = vlt_soll_haw
        write_vlt_soll_in_DB(vlt_soll_neu)

    # Temperaturdifferenz
    dt = temp_soll - temp_ist

    # Optimierung der Vorlauftemperatur
    # Schleife 1
    i = 3
    while (x >= 80) and (dt > 2):
        vlt_soll_neu = vlt_soll_haw + i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('Schlafe in Schleife 1 fuer eine Stunde.', time.strftime("
                                                %d.%m.%Y %H:%M:%S"))
        time.sleep(3600)
        print('Neue Werte:')
        x, temp_soll, temp_ist, vlt_soll_haw, vlt_sek, vlt_ist =
                                                get_data()
        dt = temp_soll - temp_ist
        print("dT:", dt)

    # Schleife 2
    while (x >= 80) and (dt <= 2):
        print('Schlafe in Schleife 2 fuer eine Stunde.', time.strftime("
                                                %d.%m.%Y %H:%M:%S"))
        time.sleep(3600)
        print('Neue Werte:')
```

```
        x, temp_soll, temp_ist, vlt_soll_haw, vlt_sek, vlt_ist =
                                                get_data()
        dt = temp_soll - temp_ist
        print("dT:", dt)

    # Schleife 3
    i = 2
    while (x >= 40) and (x < 80):
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('Schlafe in Schleife 3 fuer eine Stunde.', time.strftime("
                                                %d.%m.%Y %H:%M:%S"))
        time.sleep(3600)
        print('Neue Werte:')
        x, temp_soll, temp_ist, vlt_soll_haw, vlt_sek, vlt_ist =
                                                get_data()
        dt = temp_soll - temp_ist
        print("dT:", dt)

    # Schleife 4
    i = 4
    while (x < 40):
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('Schlafe in Schleife 4 fuer eine Stunde.', time.strftime("
                                                %d.%m.%Y %H:%M:%S"))
        time.sleep(3600)
        print('Neue Werte:')
        x, temp_soll, temp_ist, vlt_soll_haw, vlt_sek, vlt_ist =
                                                get_data()
        dt = temp_soll - temp_ist
        print("dT:", dt)
```

```
# Aktuelle Datenpunkte aus InfluxDB lesen
def read_last_point_from_influx_field(measurement, field_key):
    """
    Request last point of a field from an InfluxDB measurement
    :param measurement: Name of the measurement
    :param field_key: Field key to query
    :return: Latest value of measurement
    """
    select_query = f'select last("{field_key}") from "{measurement}"'
    result_query: influxdb.resultset.ResultSet = influx_client.query(
        query=select_query)
    result = list(result_query.get_points(measurement=measurement))[0]['
                                        last']
    return result


# Neuen Sollwert fuer VLT ueberschreiben
def write_vlt_soll_in_DB(vlt_soll_neu):
    name = "ccu3"
    value = vlt_soll_neu
    data_point = "HK_Nord_Sollwert_VLT_HAW"
    data = f'{name} {data_point}={value}'
    response = requests.post(INFLUX_WRITE_URL, data=data, auth=
                                        INFLUX_AUTH)
    try:
        response.raise_for_status()
    except requests.HTTPError:
        print(f"Error: Write request to InfluxDB for measurement '{name}
                                            '"
              f' failed: {response.status_code}: {response.reason}')
    print()
    print("data=", data, INFLUX_WRITE_URL)


def get_data():
    x = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_032,
        field_key=INFLUX_FIELD_X)
    print("Ventiloeffnung:", x)
    temp_soll = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_032,
```

```
        field_key=INFLUX_FIELD_TEMP_SOLL)
    print("Soll-T:", temp_soll)
    temp_ist = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_032,
        field_key=INFLUX_FIELD_TEMP_IST)
    print("Ist-T:", temp_ist)
    print("dT:", temp_soll - temp_ist)
    vlt_soll_haw = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_CCU3,
        field_key=INFLUX_FIELD_VLT_SOLL_HAW)
    print("VLT_soll", vlt_soll_haw)
    vlt_sek = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_WAGO_SPS,
        field_key=INFLUX_FIELD_VLT_SEK)
    print("VLT_sek", vlt_sek)
    vlt_ist = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_WAGO_SPS,
    field_key=INFLUX_FIELD_VLT_IST)
    print("VLT_ist:", vlt_ist)
    return x, temp_soll, temp_ist, vlt_soll_haw, vlt_sek, vlt_ist



while True:
    if __name__ == '__main__':
        main()
```

A.4.4 Referenzräume 0.11, 0.12 und 0.32

```
# Imports
import time
from influxdb import InfluxDBClient
import influxdb.resultset
import requests

# Umgebungsvariablen
host = os.environ['INFLUX_HOST']
username = os.environ['INFLUX_USERNAME']
password = os.environ['INFLUX_PASSWORD']
```

```
database = os.environ['INFLUX_DATABASE']
INFLUX_WRITE_URL = f'http://{INFLUX_HOST}:8086' \
                   f'/write?db={INFLUX_DATABASE}' \
                   f'&precision=s'
INFLUX_AUTH = (INFLUX_USERNAME, INFLUX_PASSWORD)
influx_client = InfluxDBClient(
    host=INFLUX_HOST,
    username=INFLUX_USERNAME, password=INFLUX_PASSWORD,
    database=INFLUX_DATABASE)

# Hype Query strings
# 0.11
INFLUX_MEASUREMENT_011_RECHTS = "Thermostat_0.11_Seminarraum_rechts"
INFLUX_FIELD_XR011 = 'current_valve_position'
INFLUX_MEASUREMENT_011_LINKS = "Thermostat_0.11_Seminarraum_links"
INFLUX_FIELD_XL011 = 'current_valve_position'
INFLUX_FIELD_TEMP_SOLL011 = 'current_set_point'
INFLUX_FIELD_TEMP_IST011 = 'current_temperature'
# 0.32
INFLUX_MEASUREMENT_032 = "Thermostat_0.32_Buero_Haustechnik"
INFLUX_FIELD_X = 'current_valve_position'
INFLUX_FIELD_TEMP_SOLL032 = 'current_set_point'
INFLUX_FIELD_TEMP_IST032 = 'current_temperature'
# 0.12
INFLUX_MEASUREMENT_012_MITTE = "Thermostat_0.12_Seminarraum_mitte"
INFLUX_FIELD_XM = 'current_valve_position'
INFLUX_MEASUREMENT_012_LINKS = "Thermostat_0.12_Seminarraum_links"
INFLUX_FIELD_XL = 'current_valve_position'
INFLUX_MEASUREMENT_012_RECHTS = "Thermostat_0.
                                  12_Seminarraum_rechts_Erker"
INFLUX_FIELD_XR = 'current_valve_position'
INFLUX_FIELD_TEMP_SOLL = 'current_set_point'
INFLUX_FIELD_TEMP_IST = 'current_temperature'
# alle
INFLUX_MEASUREMENT_WAGO_SPS = "Wago_SPS"
INFLUX_FIELD_VLT_IST = 'Hzk_BH_Nord_VLT'
INFLUX_FIELD_VLT_SEK = 'VLT_sek'
INFLUX_MEASUREMENT_CCU3 = 'ccu3'
INFLUX_FIELD_VLT_SOLL_HAW = 'HK_Nord_Sollwert_VLT_HAW'

def main():
```

```
# Gather data from InfluxDB
# 0.11
xr_011 = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_011_RECHTS,
    field_key=INFLUX_FIELD_XR011)
print("Ventil rechts 011:", xr_011)
xl_011 = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_011_LINKS,
    field_key=INFLUX_FIELD_XL011)
print("Ventil links 011:", xl_011)
temp_soll_011 = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_011_RECHTS,
    field_key=INFLUX_FIELD_TEMP_SOLL011)
print("Soll-T 011:", temp_soll_011)
temp_ist_011 = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_011_RECHTS,
    field_key=INFLUX_FIELD_TEMP_IST011)
print("Ist-T 011:", temp_ist_011)
# 0.32
x = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_032,
    field_key=INFLUX_FIELD_X)
print("Ventiloeffnung 0.32:", x)
temp_soll_032 = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_032,
    field_key=INFLUX_FIELD_TEMP_SOLL032)
print("Soll-T 0.32:", temp_soll_032)
temp_ist_032 = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_032,
    field_key=INFLUX_FIELD_TEMP_IST032)
print("Ist-T 0.32:", temp_ist_032)
# 0.12
x_r = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_012_RECHTS,
    field_key=INFLUX_FIELD_XR)
print("Ventil rechts:", x_r)
x_l = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_012_LINKS,
    field_key=INFLUX_FIELD_XL)
print("Ventil links:", x_l)
x_m = read_last_point_from_influx_field(
```

```
    measurement=INFLUX_MEASUREMENT_012_MITTE,
    field_key=INFLUX_FIELD_XM)
print("Ventil mitte:", x_m)
temp_soll = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_012_MITTE,
    field_key=INFLUX_FIELD_TEMP_SOLL)
print("Soll-T:", temp_soll)
temp_ist = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_012_MITTE,
    field_key=INFLUX_FIELD_TEMP_IST)
print("Ist-T:", temp_ist)
# alle
vlt_soll_haw = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_CCU3,
    field_key=INFLUX_FIELD_VLT_SOLL_HAW)
print("VLT_soll", vlt_soll_haw)
vlt_sek = read_last_point_from_influx_field(
    measurement=INFLUX_MEASUREMENT_WAGO_SPS,
    field_key=INFLUX_FIELD_VLT_SEK)
print("VLT_sek", vlt_sek)

# Temperaturdifferenz
dt_032 = temp_soll_032 - temp_ist_032
dt_011 = temp_soll_011 - temp_ist_011
dt = temp_soll - temp_ist


if (dt_011 > dt-2) or (dt_011 == dt-2) and (dt_011 > dt_032-2):
    print("Raum 0.11")
    # Raum 0.11
    # Schleife 1
    i = 3
    while (xr_011 >= 80) and (xl_011 >= 80) and (dt_011 > 0):
        if dt_011 > 2:
            print('Schlafe eine Stunde wegen geoeffnetem Fenster')
            time.sleep(3600)
        vlt_soll_neu = vlt_soll_haw + i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
```

```
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('0.11: Schlafe in Schleife 1 fuer eine Stunde.', time.
                                            strftime("%d.%m.%Y %H:%M
                                            :%S"))
        time.sleep(3600)
        print('0.11 Neue Werte:')
        xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                            temp_soll_032,
                                            temp_ist_032, x_r, x_l,
                                            x_m, temp_soll, temp_ist
                                            , vlt_soll_haw, vlt_sek
                                            = get_data()
        dt_032 = temp_soll_032 - temp_ist_032
        dt_011 = temp_soll_011 - temp_ist_011
        dt = temp_soll - temp_ist
        print("0.12: dT:", dt)
        print("0.32: dT:", dt_032)
        print("0.11: dT:", dt_011)

    # Schleife 2
    while (xr_011 >= 80) and (xl_011 >= 80) and (dt_011 <= 0):
        print('0.11 Schlafe in Schleife 2 fuer halbe Stunde.', time.
                                            strftime("%d.%m.%Y %H:%M
                                            :%S"))
        time.sleep(3600)
        print('0.11 Neue Werte:')
        xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                            temp_soll_032,
                                            temp_ist_032, x_r, x_l,
                                            x_m, temp_soll, temp_ist
                                            , vlt_soll_haw, vlt_sek
                                            = get_data()
        dt_032 = temp_soll_032 - temp_ist_032
        dt_011 = temp_soll_011 - temp_ist_011
        dt = temp_soll - temp_ist
        print("0.12: dT:", dt)
        print("0.32: dT:", dt_032)
        print("0.11: dT:", dt_011)
        if (dt_032-2 > dt_011) and (dt_032 > 2):
            break
        elif (dt-2 > dt_011) and (dt > 2):
```

```
            break
        else:
            continue

    # Schleife 3
    i = 2
    while (xr_011 >= 40) and (xr_011 < 80) and (xl_011 >= 80):
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('0.11 Schlafe in Schleife 3 fuer eine Stunde.', time.
                                            strftime("%d.%m.%Y %H:%M
                                            :%S"))
        time.sleep(3600)
        print('0.11 Neue Werte:')
        xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                            temp_soll_032,
                                            temp_ist_032, x_r, x_l,
                                            x_m, temp_soll, temp_ist
                                            , vlt_soll_haw, vlt_sek
                                            = get_data()
        dt_032 = temp_soll_032 - temp_ist_032
        dt_011 = temp_soll_011 - temp_ist_011
        dt = temp_soll - temp_ist
        print("0.12: dT:", dt)
        print("0.32: dT:", dt_032)
        print("0.11: dT:", dt_011)
        if (dt_032-2 > dt_011) and (dt_032 > 2):
            break
        elif (dt-2 > dt_011) and (dt > 2):
            break
        else:
            continue

    # Schleife 4
    i = 2
    while (xl_011 >= 40) and (xl_011 < 80) and (xr_011 >= 80):
        vlt_soll_neu = vlt_soll_haw - i
```

```
            if vlt_soll_neu > vlt_sek:
                vlt_soll_neu = vlt_sek
            if vlt_soll_neu < 30:
                vlt_soll_neu = 30
            write_vlt_soll_in_DB(vlt_soll_neu)
            print('0.11 Schlafe in Schleife 4 fuer eine Stunde.', time.
                                                strftime("%d.%m.%Y %H:%M
                                                :%S"))
            time.sleep(3600)
            print('0.11 Neue Werte:')
            xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                                temp_soll_032,
                                                temp_ist_032, x_r, x_l,
                                                x_m, temp_soll, temp_ist
                                                , vlt_soll_haw, vlt_sek
                                                = get_data()
            dt_032 = temp_soll_032 - temp_ist_032
            dt_011 = temp_soll_011 - temp_ist_011
            dt = temp_soll - temp_ist
            print("0.12: dT:", dt)
            print("0.32: dT:", dt_032)
            print("0.11: dT:", dt_011)
            if (dt_032-2 > dt_011) and (dt_032 > 2):
                break
            elif (dt-2 > dt_011) and (dt > 2):
                break
            else:
                continue

        # Schleife 5
        i = 2
        while (xr_011 >= 40) and (xr_011 < 80) and (xl_011 >= 40) and (
                                                xl_011 < 80):
            vlt_soll_neu = vlt_soll_haw - i
            if vlt_soll_neu > vlt_sek:
                vlt_soll_neu = vlt_sek
            if vlt_soll_neu < 30:
                vlt_soll_neu = 30
            write_vlt_soll_in_DB(vlt_soll_neu)
            print('0.11 Schlafe in Schleife 5 fuer eine Stunde.', time.
                                                strftime("%d.%m.%Y %H:%M
```

```
                                        :%S"))
        time.sleep(3600)
        print('0.11 Neue Werte:')
        xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                        temp_soll_032,
                                        temp_ist_032, x_r, x_l,
                                        x_m, temp_soll, temp_ist
                                        , vlt_soll_haw, vlt_sek
                                        = get_data()
        dt_032 = temp_soll_032 - temp_ist_032
        dt_011 = temp_soll_011 - temp_ist_011
        dt = temp_soll - temp_ist
        print("0.12: dT:", dt)
        print("0.32: dT:", dt_032)
        print("0.11: dT:", dt_011)
        if (dt_032-2 > dt_011) and (dt_032 > 2):
            break
        elif (dt-2 > dt_011) and (dt > 2):
            break
        else:
            continue

    # Schleife 6
    i = 4
    while (xr_011 < 40) or (xl_011 < 40):
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('0.11 Schlafe in Schleife 6 fuer eine Stunde.', time.
                                        strftime("%d.%m.%Y %H:%M
                                        :%S"))
        time.sleep(3600)
        print('0.11 Neue Werte:')
        xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                        temp_soll_032,
                                        temp_ist_032, x_r, x_l,
                                        x_m, temp_soll, temp_ist
                                        , vlt_soll_haw, vlt_sek
```

```
                                                = get_data()
            dt_032 = temp_soll_032 - temp_ist_032
            dt_011 = temp_soll_011 - temp_ist_011
            dt = temp_soll - temp_ist
            print("0.12: dT:", dt)
            print("0.32: dT:", dt_032)
            print("0.11: dT:", dt_011)
            if (dt_032-2 > dt_011) and (dt_032 > 2):
                break
            elif (dt-2 > dt_011) and (dt > 2):
                break
            else:
                continue
    # 0.32
    if (dt_032 > dt) or (dt_032 == dt) and (dt_032-2 > dt_011) or (
                                        dt_032-2 == dt_011):
        print("Raum 0.32")
        # Schleife 1
        i = 3
        while (x >= 80) and (dt_032 > 2):
            if dt_032 > 3:
                print('Schlafe eine Stunde wegen geoeffnetem Fenster')
                time.sleep(3600)
            vlt_soll_neu = vlt_soll_haw + i
            if vlt_soll_neu > vlt_sek:
                vlt_soll_neu = vlt_sek
            write_vlt_soll_in_DB(vlt_soll_neu)
            print('0.32 Schlafe in Schleife 1 fuer eine Stunde.', time.
                                                strftime("%d.%m.%Y %H:%M
                                                :%S"))
            time.sleep(3600)
            print('0.32 Neue Werte:')
            xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                                temp_soll_032,
                                                temp_ist_032, x_r, x_l,
                                                x_m, temp_soll, temp_ist
                                                , vlt_soll_haw, vlt_sek
                                                = get_data()
            dt_032 = temp_soll_032 - temp_ist_032
            dt_011 = temp_soll_011 - temp_ist_011
            dt = temp_soll - temp_ist
```

```
        print("0.12: dT:", dt)
        print("0.32: dT:", dt_032)
        print("0.11: dT:", dt_011)

    # Schleife 2
    while (x >= 80) and (dt_032 <= 2):
        print('0.32 Schlafe in Schleife 2 fuer eine Stunde.', time.
                                            strftime("%d.%m.%Y %H:%M
                                            :%S"))
        time.sleep(3600)
        print('0.32 Neue Werte:')
        xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                            temp_soll_032,
                                            temp_ist_032, x_r, x_l,
                                            x_m, temp_soll, temp_ist
                                            , vlt_soll_haw, vlt_sek
                                            = get_data()
        dt_032 = temp_soll_032 - temp_ist_032
        dt_011 = temp_soll_011 - temp_ist_011
        dt = temp_soll - temp_ist
        print("0.12: dT:", dt)
        print("0.32: dT:", dt_032)
        print("0.11: dT:", dt_011)
        if (dt_011 > dt_032-2) and (dt_011 > 0):
            break
        elif (dt > dt_032) and (dt > 2):
            break
        else:
            continue

    # Schleife 3
    i = 2
    while (x >= 40) and (x < 80):
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('0.32 Schlafe in Schleife 3 fuer eine Stunde.', time.
                                            strftime("%d.%m.%Y %H:%M
```

```
                                        :%S"))
        time.sleep(3600)
        print('0.32 Neue Werte:')
        xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                        temp_soll_032,
                                        temp_ist_032, x_r, x_l,
                                        x_m, temp_soll, temp_ist
                                        , vlt_soll_haw, vlt_sek
                                        = get_data()
        dt_032 = temp_soll_032 - temp_ist_032
        dt_011 = temp_soll_011 - temp_ist_011
        dt = temp_soll - temp_ist
        print("0.12: dT:", dt)
        print("0.32: dT:", dt_032)
        print("0.11: dT:", dt_011)
        if (dt_011 > dt_032-2) and (dt_011 > 0):
            break
        elif (dt > dt_032) and (dt > 2):
            break
        else:
            continue

    # Schleife 4
    i = 4
    while (x < 40):
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('0.32 Schlafe in Schleife 4 fuer eine Stunde.', time.
                                        strftime("%d.%m.%Y %H:%M
                                        :%S"))
        time.sleep(3600)
        print('0.32 Neue Werte:')
        xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                        temp_soll_032,
                                        temp_ist_032, x_r, x_l,
                                        x_m, temp_soll, temp_ist
                                        , vlt_soll_haw, vlt_sek
```

```
                                                = get_data()
            dt_032 = temp_soll_032 - temp_ist_032
            dt_011 = temp_soll_011 - temp_ist_011
            dt = temp_soll - temp_ist
            print("0.12: dT:", dt)
            print("0.32: dT:", dt_032)
            print("0.11: dT:", dt_011)
            if (dt_011 > dt_032-2) and (dt_011 > 0):
                break
            elif (dt > dt_032) and (dt > 2):
                break
            else:
                continue
    # 0.12
    if (dt > dt_011) or (dt == dt_011) and (dt > dt_032):
        # Schleife 1
        i = 3
        while (x_r >= 70) and (x_l >= 80) and (x_m >= 80) and (dt > 2):
            if dt > 3:
                print('Schlafe eine Stunde wegen geoeffnetem Fenster')
                time.sleep(3600)
            vlt_soll_neu = vlt_soll_haw + i
            if vlt_soll_neu > vlt_sek:
                vlt_soll_neu = vlt_sek
            write_vlt_soll_in_DB(vlt_soll_neu)
            print('0.12: Schlafe in Schleife 1 fuer eine Stunde.', time.
                                                strftime("%d.%m.%Y %H:%M
                                                :%S"))
            time.sleep(3600)
            print('0.12: Neue Werte:')
            xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                                temp_soll_032,
                                                temp_ist_032, x_r, x_l,
                                                x_m, temp_soll, temp_ist
                                                , vlt_soll_haw, vlt_sek
                                                = get_data()
            dt_032 = temp_soll_032 - temp_ist_032
            dt_011 = temp_soll_011 - temp_ist_011
            dt = temp_soll - temp_ist
            print("0.12: dT:", dt)
            print("0.32: dT:", dt_032)
```

```
        print("0.11: dT:", dt_011)

    # Schleife 2
    while (x_r >= 70) and (x_l >= 80) and (x_m >= 80) and (dt <= 2):
        print('0.12: Schlafe in Schleife 2 fuer eine Stunde.', time.
                                            strftime("%d.%m.%Y %H:%M
                                            :%S"))
        time.sleep(3600)
        print('0.12: Neue Werte:')
        xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                            temp_soll_032,
                                            temp_ist_032, x_r, x_l,
                                            x_m, temp_soll, temp_ist
                                            , vlt_soll_haw, vlt_sek
                                            = get_data()
        dt_032 = temp_soll_032 - temp_ist_032
        dt_011 = temp_soll_011 - temp_ist_011
        dt = temp_soll - temp_ist
        print("0.12: dT:", dt)
        if (dt_011 > dt-2) and (dt_011 > 0):
            break
        elif (dt_032 > dt) and (dt_032 > 2):
            break
        else:
            continue

    # Schleife 3
    i = 2
    while (x_r >= 40) and (x_r < 70) and (x_m >= 40) and (x_m < 80)
                                        and (x_l >= 80):
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('0.12: Schlafe in Schleife 3 fuer eine Stunde.', time.
                                            strftime("%d.%m.%Y %H:%M
                                            :%S"))
        time.sleep(3600)
        print('0.12: Neue Werte:')
```

```
        xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                            temp_soll_032,
                                            temp_ist_032, x_r, x_l,
                                            x_m, temp_soll, temp_ist
                                            , vlt_soll_haw, vlt_sek
                                            = get_data()
        dt_032 = temp_soll_032 - temp_ist_032
        dt_011 = temp_soll_011 - temp_ist_011
        dt = temp_soll - temp_ist
        print("0.12: dT:", dt)
        print("0.32: dT:", dt_032)
        print("0.11: dT:", dt_011)
        if (dt_011 > dt-2) and (dt_011 > 0):
            break
        elif (dt_032 > dt) and (dt_032 > 2):
            break
        else:
            continue

    # Schleife 4
    i = 2
    while (x_l >= 40) and (x_l < 80) and (x_m >= 40) and (x_m < 80)
                                        and (x_r >= 70):
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('0.12: Schlafe in Schleife 4 fuer eine Stunde.', time.
                                            strftime("%d.%m.%Y %H:%M
                                            :%S"))
        time.sleep(3600)
        print('0.12: Neue Werte:')
        xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                            temp_soll_032,
                                            temp_ist_032, x_r, x_l,
                                            x_m, temp_soll, temp_ist
                                            , vlt_soll_haw, vlt_sek
                                            = get_data()
        dt_032 = temp_soll_032 - temp_ist_032
```

```
        dt_011 = temp_soll_011 - temp_ist_011
        dt = temp_soll - temp_ist
        print("0.12: dT:", dt)
        print("0.32: dT:", dt_032)
        print("0.11: dT:", dt_011)
        if (dt_011 > dt-2) and (dt_011 > 0):
            break
        elif (dt_032 > dt) and (dt_032 > 2):
            break
        else:
            continue

    # Schleife 5
    i = 2
    while (x_r >= 40) and (x_r < 70) and (x_l >= 40) and (x_l < 80)
                                        and (x_m >= 80):
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('0.12: Schlafe in Schleife 5 fuer eine Stunde.', time.
                                            strftime("%d.%m.%Y %H:%M
                                            :%S"))
        time.sleep(3600)
        print('0.12: Neue Werte:')
        xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                            temp_soll_032,
                                            temp_ist_032, x_r, x_l,
                                            x_m, temp_soll, temp_ist
                                            , vlt_soll_haw, vlt_sek
                                            = get_data()
        dt_032 = temp_soll_032 - temp_ist_032
        dt_011 = temp_soll_011 - temp_ist_011
        dt = temp_soll - temp_ist
        print("0.12: dT:", dt)
        print("0.32: dT:", dt_032)
        print("0.11: dT:", dt_011)
        if (dt_011 > dt-2) and (dt_011 > 0):
            break
```

```
        elif (dt_032 > dt) and (dt_032 > 2):
            break
        else:
            continue

    # Schleife 6
    i = 2
    while (x_r >= 40) and (x_r < 70) and (x_m >= 80) and (x_l >= 80)
                                        :
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('0.12: Schlafe in Schleife 6 fuer eine Stunde.', time.
                                            strftime("%d.%m.%Y %H:%M
                                            :%S"))
        time.sleep(3600)
        print('0.12: Neue Werte:')
        xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                            temp_soll_032,
                                            temp_ist_032, x_r, x_l,
                                            x_m, temp_soll, temp_ist
                                            , vlt_soll_haw, vlt_sek
                                            = get_data()
        dt_032 = temp_soll_032 - temp_ist_032
        dt_011 = temp_soll_011 - temp_ist_011
        dt = temp_soll - temp_ist
        print("0.12: dT:", dt)
        print("0.32: dT:", dt_032)
        print("0.11: dT:", dt_011)
        if (dt_011 > dt-2) and (dt_011 > 0):
            break
        elif (dt_032 > dt) and (dt_032 > 2):
            break
        else:
            continue

    # Schleife 7
    i = 2
```

```
        while (x_l >= 40) and (x_l < 80) and (x_m >= 80) and (x_r >= 70)
                                                :
            vlt_soll_neu = vlt_soll_haw - i
            if vlt_soll_neu > vlt_sek:
                vlt_soll_neu = vlt_sek
            if vlt_soll_neu < 30:
                vlt_soll_neu = 30
            write_vlt_soll_in_DB(vlt_soll_neu)
            print('0.12: Schlafe in Schleife 7 fuer eine Stunde.', time.
                                                strftime("%d.%m.%Y %H:%M
                                                :%S"))
            time.sleep(3600)
            print('0.12: Neue Werte:')
            xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                                temp_soll_032,
                                                temp_ist_032, x_r, x_l,
                                                x_m, temp_soll, temp_ist
                                                , vlt_soll_haw, vlt_sek
                                                = get_data()
            dt_032 = temp_soll_032 - temp_ist_032
            dt_011 = temp_soll_011 - temp_ist_011
            dt = temp_soll - temp_ist
            print("0.12: dT:", dt)
            print("0.32: dT:", dt_032)
            print("0.11: dT:", dt_011)
            if (dt_011 > dt-2) and (dt_011 > 0):
                break
            elif (dt_032 > dt) and (dt_032 > 2):
                break
            else:
                continue

        # Schleife 8
        i = 2
        while (x_m >= 40) and (x_m < 80) and (x_r >= 70) and (x_l >= 80)
                                                :
            vlt_soll_neu = vlt_soll_haw - i
            if vlt_soll_neu > vlt_sek:
                vlt_soll_neu = vlt_sek
            if vlt_soll_neu < 30:
                vlt_soll_neu = 30
```

```
            write_vlt_soll_in_DB(vlt_soll_neu)
            print('0.12: Schlafe in Schleife 8 fuer eine Stunde.', time.
                                                strftime("%d.%m.%Y %H:%M
                                                :%S"))
            time.sleep(3600)
            print('0.12: Neue Werte:')
            xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                                temp_soll_032,
                                                temp_ist_032, x_r, x_l,
                                                x_m, temp_soll, temp_ist
                                                , vlt_soll_haw, vlt_sek
                                                = get_data()
            dt_032 = temp_soll_032 - temp_ist_032
            dt_011 = temp_soll_011 - temp_ist_011
            dt = temp_soll - temp_ist
            print("0.12: dT:", dt)
            print("0.32: dT:", dt_032)
            print("0.11: dT:", dt_011)
            if (dt_011 > dt-2) and (dt_011 > 0):
                break
            elif (dt_032 > dt) and (dt_032 > 2):
                break
            else:
                continue

        # Schleife 9
        i = 2
        while (x_r >= 40) and (x_r < 70) and (x_l >= 40) and (x_l < 80)
                                            and (x_m >= 40) and (x_m <
                                            80):
            vlt_soll_neu = vlt_soll_haw - i
            if vlt_soll_neu > vlt_sek:
                vlt_soll_neu = vlt_sek
            if vlt_soll_neu < 30:
                vlt_soll_neu = 30
            write_vlt_soll_in_DB(vlt_soll_neu)
            print('0.12: Schlafe in Schleife 9 fuer eine Stunde.', time.
                                                strftime("%d.%m.%Y %H:%M
                                                :%S"))
            time.sleep(3600)
            print('0.12: Neue Werte:')
```

```
        xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                            temp_soll_032,
                                            temp_ist_032, x_r, x_l,
                                            x_m, temp_soll, temp_ist
                                            , vlt_soll_haw, vlt_sek
                                            = get_data()
        dt_032 = temp_soll_032 - temp_ist_032
        dt_011 = temp_soll_011 - temp_ist_011
        dt = temp_soll - temp_ist
        print("0.12: dT:", dt)
        print("0.32: dT:", dt_032)
        print("0.11: dT:", dt_011)
        if (dt_011 > dt-2) and (dt_011 > 0):
            break
        elif (dt_032 > dt) and (dt_032 > 2):
            break
        else:
            continue

    # Schleife 10
    i = 4
    while (x_r < 40) or (x_l < 40) or (x_m < 40):
        vlt_soll_neu = vlt_soll_haw - i
        if vlt_soll_neu > vlt_sek:
            vlt_soll_neu = vlt_sek
        if vlt_soll_neu < 30:
            vlt_soll_neu = 30
        write_vlt_soll_in_DB(vlt_soll_neu)
        print('0.12: Schlafe in Schleife 10 fuer eine Stunde.', time
                                            .strftime("%d.%m.%Y %H:%
                                            M:%S"))
        time.sleep(3600)
        print('0.12: Neue Werte:')
        xr_011, xl_011, temp_soll_011, temp_ist_011, x,
                                            temp_soll_032,
                                            temp_ist_032, x_r, x_l,
                                            x_m, temp_soll, temp_ist
                                            , vlt_soll_haw, vlt_sek
                                            = get_data()
        dt_032 = temp_soll_032 - temp_ist_032
        dt_011 = temp_soll_011 - temp_ist_011
```

```
            dt = temp_soll - temp_ist
            print("0.12: dT:", dt)
            print("0.32: dT:", dt_032)
            print("0.11: dT:", dt_011)
            if (dt_011 > dt-2) and (dt_011 > 0):
                break
            elif (dt_032 > dt) and (dt_032 > 2):
                break
            else:
                continue

    else:
        print("Keine passende Schleife gefunden")
        time.sleep(1800)

# Aktuelle Datenpunkte aus InfluxDB lesen
def read_last_point_from_influx_field(measurement, field_key):
    """
    Request last point of a field from an InfluxDB measurement
    :param measurement: Name of the measurement
    :param field_key: Field key to query
    :return: Latest value of measurement
    """
    select_query = f'select last("{field_key}") from "{measurement}"'
    result_query: influxdb.resultset.ResultSet = influx_client.query(
        query=select_query)
    result = list(result_query.get_points(measurement=measurement))[0]['
                                          last']
    return result


# Neuen Sollwert fuer VLT ueberschreiben
def write_vlt_soll_in_DB(vlt_soll_neu):
    name = "ccu3"
    value = vlt_soll_neu
    data_point = "HK_Nord_Sollwert_VLT_HAW"
    data = f'{name} {data_point}={value}'
    response = requests.post(INFLUX_WRITE_URL, data=data, auth=
                                          INFLUX_AUTH)
    try:
        response.raise_for_status()
```

```
    except requests.HTTPError:
        print(f"Error: Write request to InfluxDB for measurement '{name}
                                                    '"
              f' failed: {response.status_code}: {response.reason}')
    print()
    print("data=", data, INFLUX_WRITE_URL)


def get_data():
    # 0.11
    xr_011 = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_011_RECHTS,
        field_key=INFLUX_FIELD_XR011)
    print("Ventil rechts 011:", xr_011)
    xl_011 = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_011_LINKS,
        field_key=INFLUX_FIELD_XL011)
    print("Ventil links 011:", xl_011)
    temp_soll_011 = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_011_RECHTS,
        field_key=INFLUX_FIELD_TEMP_SOLL011)
    print("Soll-T 011:", temp_soll_011)
    temp_ist_011 = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_011_RECHTS,
        field_key=INFLUX_FIELD_TEMP_IST011)
    print("Ist-T 011:", temp_ist_011)
    # 0.32
    x = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_032,
        field_key=INFLUX_FIELD_X)
    print("Ventiloeffnung 0.32:", x)
    temp_soll_032 = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_032,
        field_key=INFLUX_FIELD_TEMP_SOLL032)
    print("Soll-T 0.32:", temp_soll_032)
    temp_ist_032 = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_032,
        field_key=INFLUX_FIELD_TEMP_IST032)
    print("Ist-T 0.32:", temp_ist_032)
    # 0.12
    x_r = read_last_point_from_influx_field(
```

```
        measurement=INFLUX_MEASUREMENT_012_RECHTS,
        field_key=INFLUX_FIELD_XR)
    print("Ventil rechts:", x_r)
    x_l = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_012_LINKS,
        field_key=INFLUX_FIELD_XL)
    print("Ventil links:", x_l)
    x_m = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_012_MITTE,
        field_key=INFLUX_FIELD_XM)
    print("Ventil mitte:", x_m)
    temp_soll = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_012_MITTE,
        field_key=INFLUX_FIELD_TEMP_SOLL)
    print("Soll-T:", temp_soll)
    temp_ist = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_012_MITTE,
        field_key=INFLUX_FIELD_TEMP_IST)
    print("Ist-T:", temp_ist)
    # alle
    vlt_soll_haw = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_CCU3,
        field_key=INFLUX_FIELD_VLT_SOLL_HAW)
    print("VLT_soll", vlt_soll_haw)
    vlt_sek = read_last_point_from_influx_field(
        measurement=INFLUX_MEASUREMENT_WAGO_SPS,
        field_key=INFLUX_FIELD_VLT_SEK)
    print("VLT_sek", vlt_sek)
    return xr_011, xl_011, temp_soll_011, temp_ist_011, x, temp_soll_032
                                        , temp_ist_032, x_r, x_l, x_m,
                                        temp_soll, temp_ist,
                                        vlt_soll_haw, vlt_sek


while True:
    if __name__ == '__main__':
        main()
```

A.5 Diagramme Raum 0.11 Testphase 2 bis 4

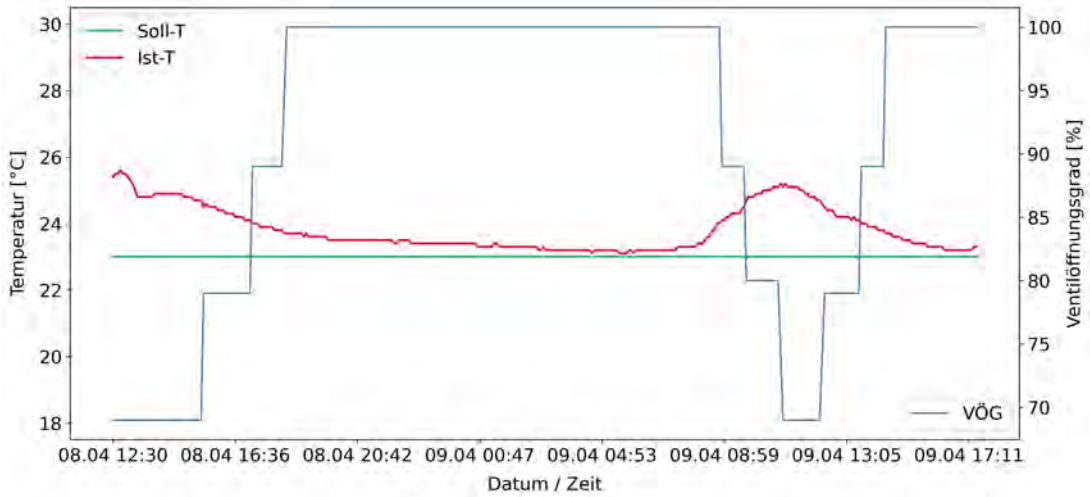


Abbildung A.10: Testphase 2: Referenzraum 0.11

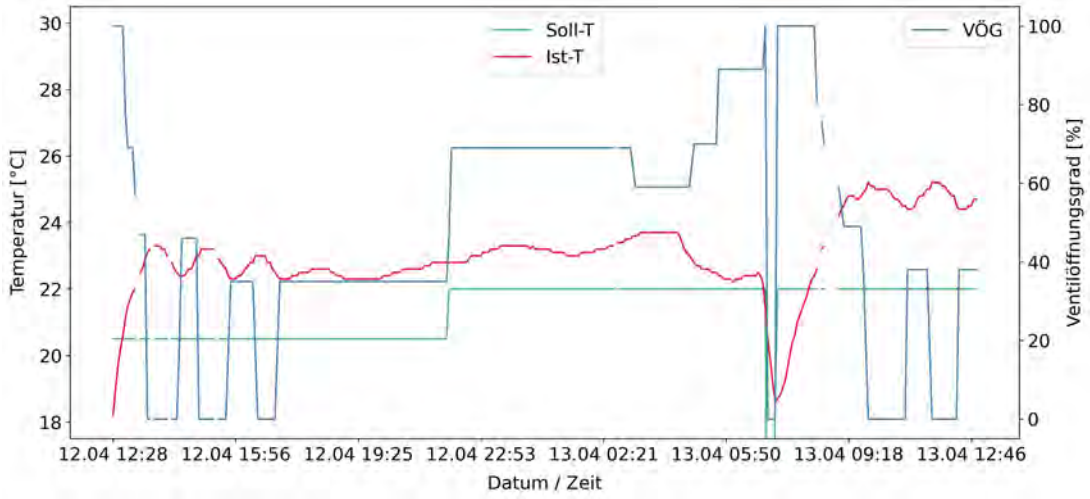


Abbildung A.11: Testphase 3: Raum 0.11

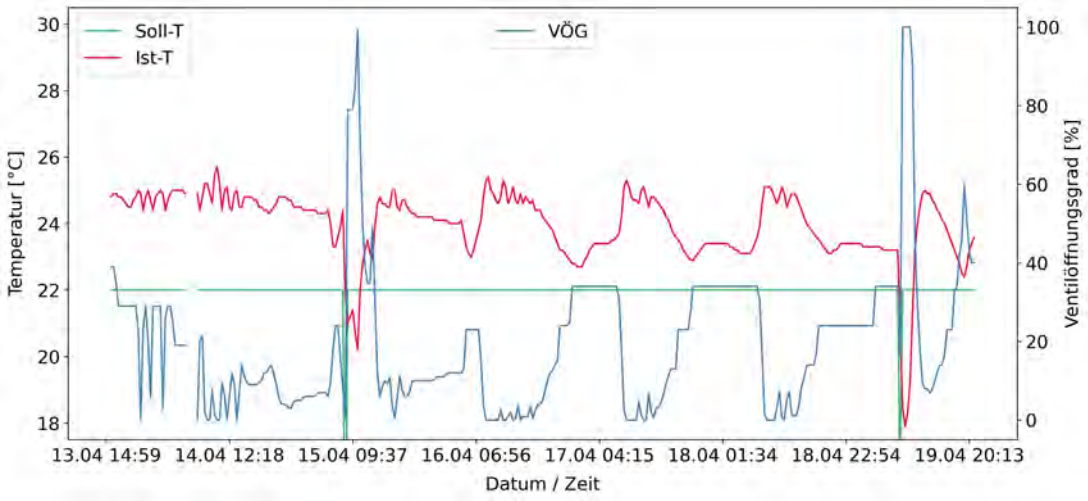


Abbildung A.12: Testphase 4: Raum 0.11

A.6 Ergebnisse der Raumtemperaturen im Heizkreis Nord

Abbildung A.13: Finale Testphase: Temperaturen in Raum 0.13

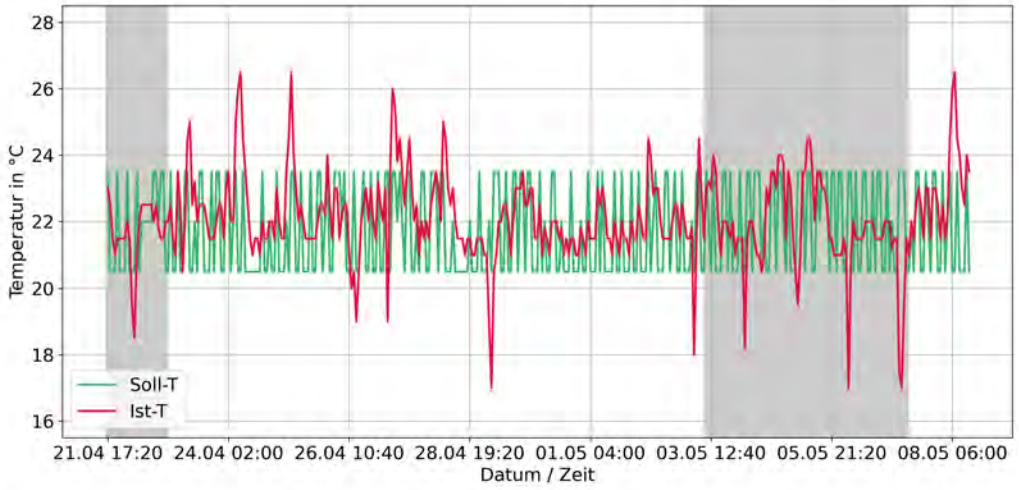


Abbildung A.14: Finale Testphase: Temperaturen in Raum 0.14

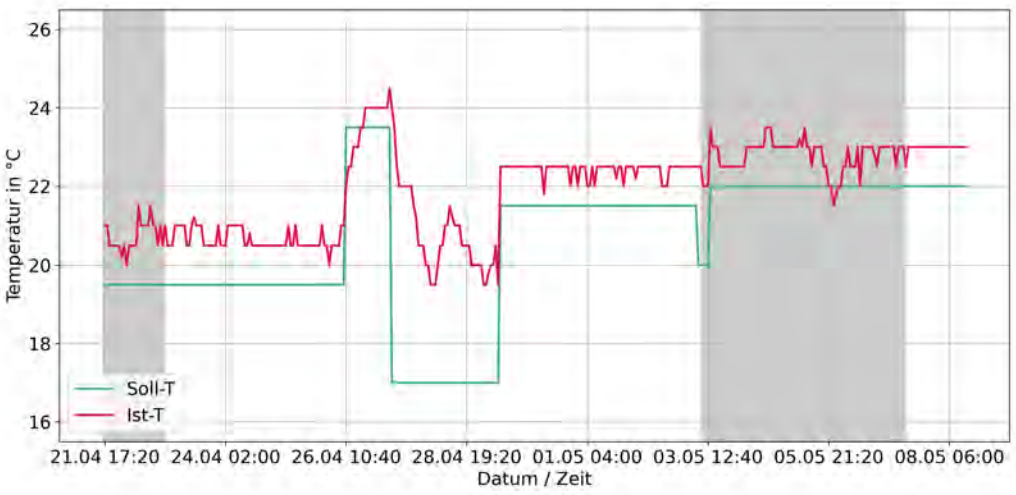


Abbildung A.15: Finale Testphase: Temperaturen in Raum 0.15

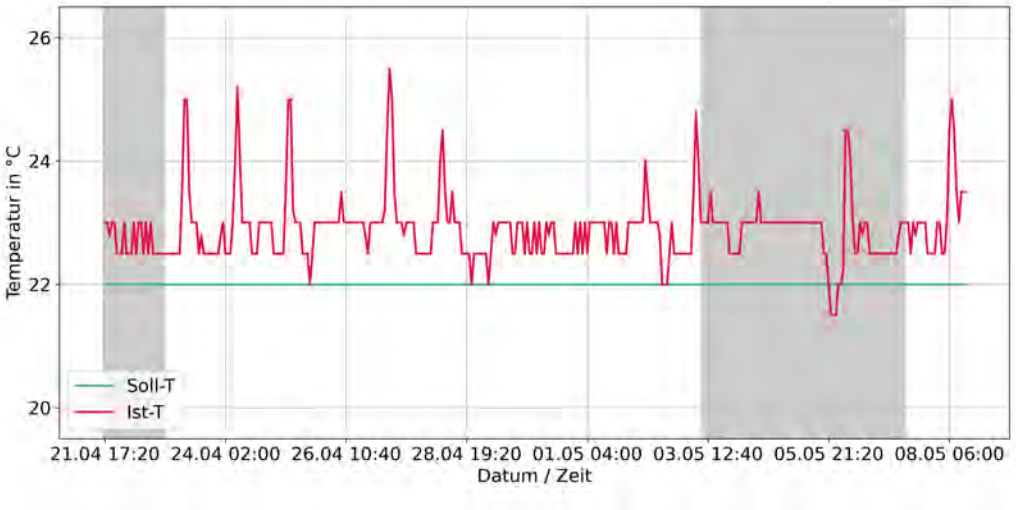


Abbildung A.16: Finale Testphase: Temperaturen in Raum 0.18

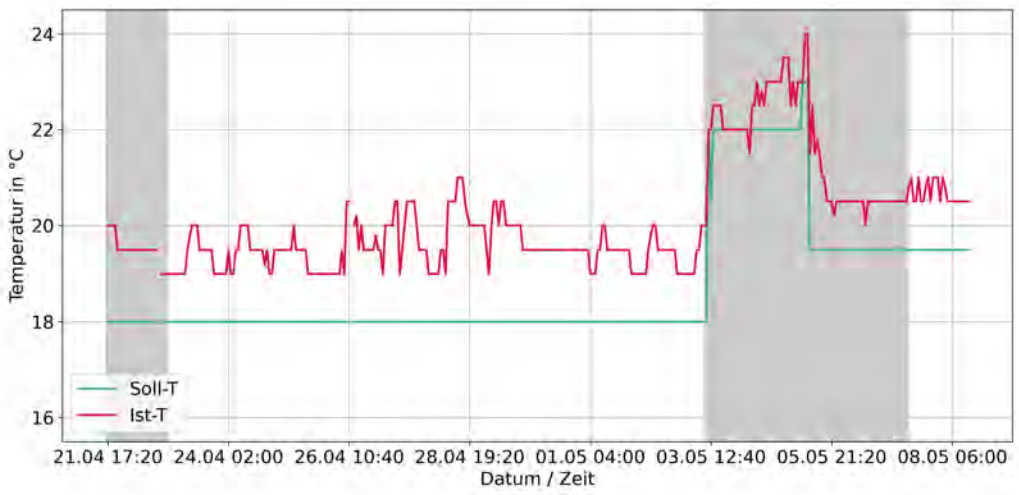


Abbildung A.17: Finale Testphase: Temperaturen in Raum 0.21

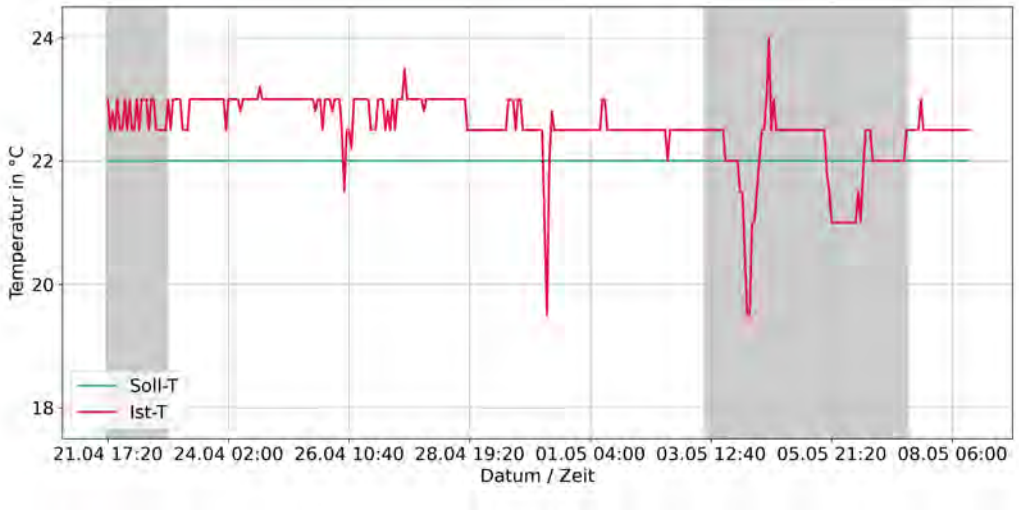


Abbildung A.18: Finale Testphase: Temperaturen in Raum 1.10

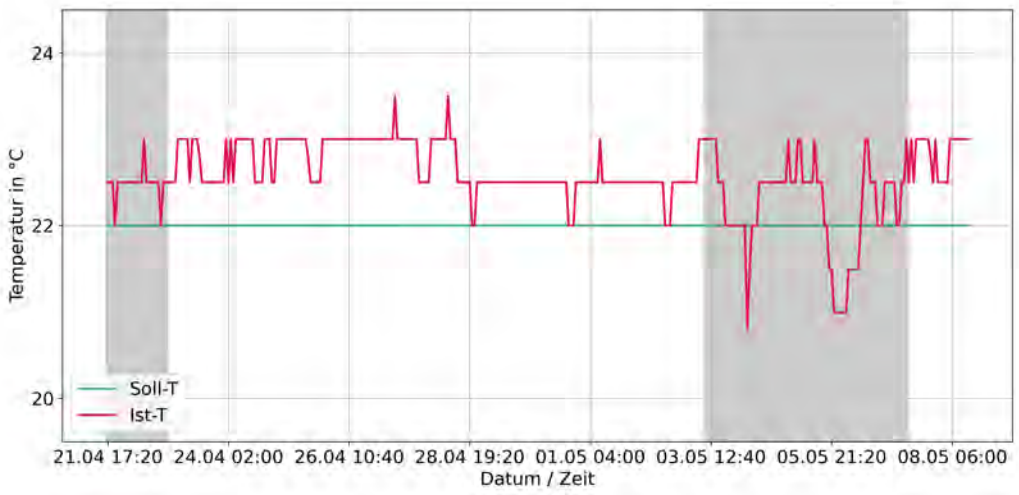


Abbildung A.19: Finale Testphase: Temperaturen in Raum 1.11

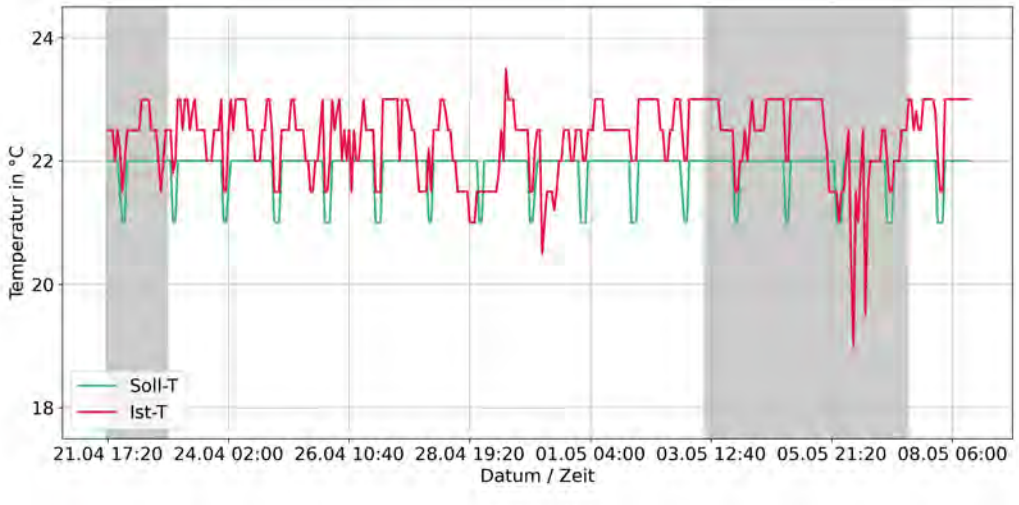


Abbildung A.20: Finale Testphase: Temperaturen in Raum 1.12

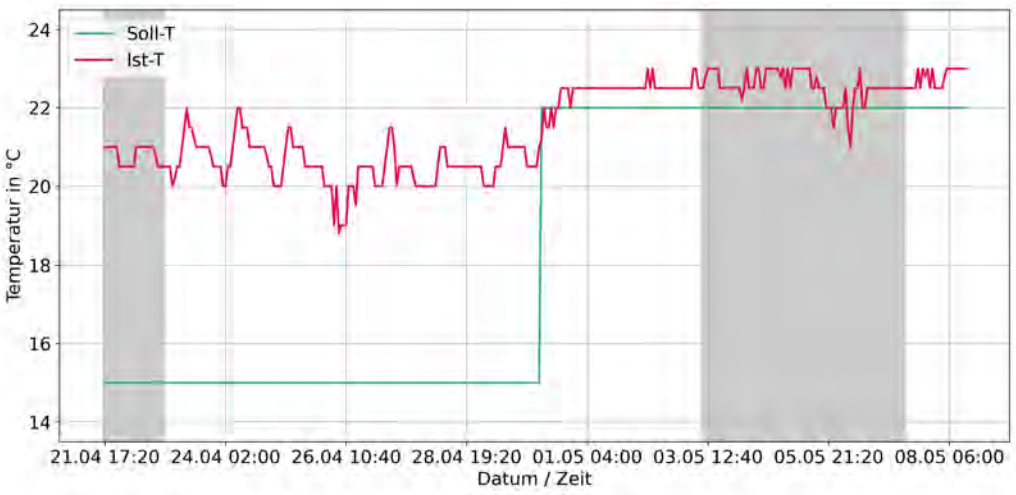


Abbildung A.21: Finale Testphase: Temperaturen in Raum 1.13

Abbildung A.22: Finale Testphase: Temperaturen in Raum 1.14

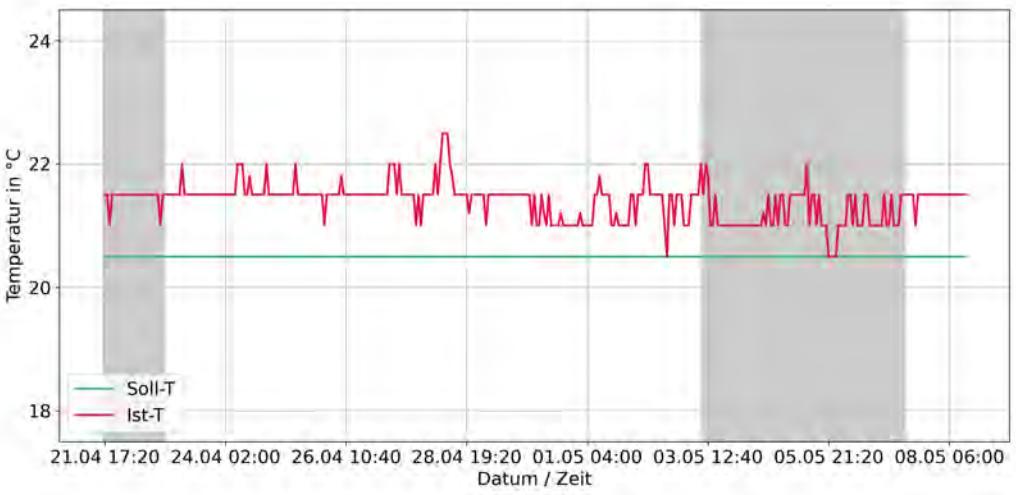


Abbildung A.23: Finale Testphase: Temperaturen in Raum 1.18

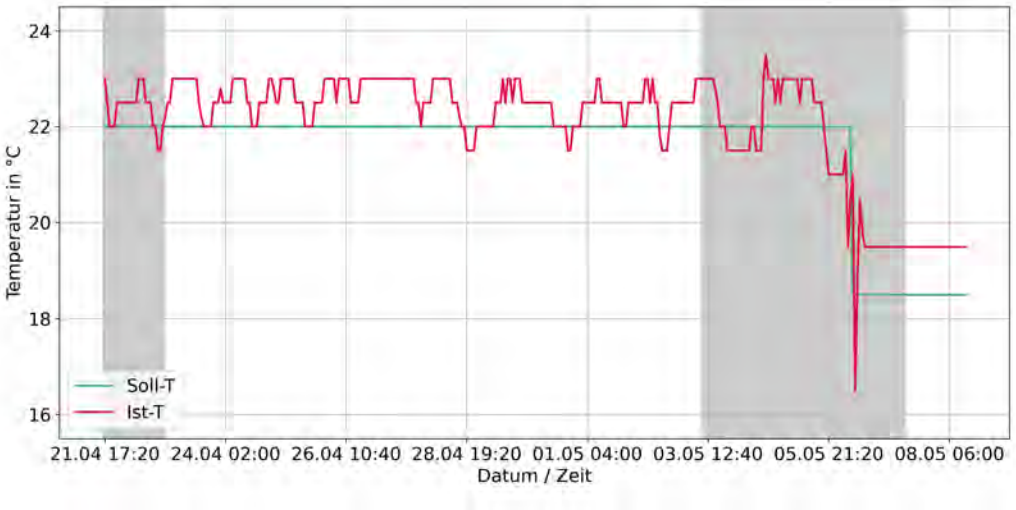


Abbildung A.24: Finale Testphase: Temperaturen in Raum 1.21

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Alice Heeb

dass ich die vorliegende Bachelorarbeit mit dem Thema:

Smart Pro HeaT: Entwurf und Umsetzung einer bedarfsgeführten Regelung zur Optimierung der Vorlauftemperatur

ohne Fremde Hilfe selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 20. Juli 2021

Ort, Datum

Unterschrift