

Konzeption und Inbetriebnahme einer Anlage und Modellbildung zur Raumheizungsregelung für den Betrieb mit Modellprädiktiver Regelung

Conception and startup operations of a technical system and model development for
a space heating control to run with model predictive control

Master-Thesis
im Studiengang Wirtschaftsingenieurwesen

zur Erlangung des akademischen Grades
Master of Science (M.Sc.)

vorgelegt von

Daniel Johannes Mayer
aus Sulzfeld

Erstkorrektor: Prof. Dr. Angelika Altmann-Dieses

Zweitkorrektor: Prof. Dr.-Ing. Marco-Braun

Matr.-Nr.: 51968

E-Mail: daniel-j-mayer@gmx.de

Bearbeitungszeitraum: 17.03.2015 – 29.02.2016

Tag der Einreichung: 17.03.2016

Fakultät für Wirtschaftswissenschaften
Hochschule Karlsruhe – Technik und Wirtschaft
2016

Kurzfassung

Abstract

This text should be *italic*

Schlüsselwortliste: Modellprädiktive Regelung, JModelica.org, Modellbildung, Modellica, Kommunikation technischer Systeme, Modbus RTU, Modbus TCP, Raumtemperaturregelung, Solarstrahlung

Keywords: *Test1, Test2, Test3*

Danksagung

Mein besonderer Dank gilt Frau PROF. DR. ANGELIKA ALTMANN-DIESES und Herrn ADRIAN BÜRGER.

Frau PROF. DR. ANGELIKA ALTMANN-DIESES danke ich für Betreuung der Arbeit und für die persönliche Unterstützung.

Bei Herrn ADRIAN BÜRGER bedanke ich mich besonders für die umfassende und herausragende Unterstützung, ohne die insbesondere der Aufbau und die Einrichtung der Softwareumgebung in diesem Umfang nicht möglich gewesen wäre.

Für die Unterstützung und kurze Einführung in die Modbus Protokolle, möchte ich mich bei Herrn bedanke ich mich bei Herrn CHRISTIAN SCHMIDT.

Außerdem möchte Herrn PROF. DR. MARCO BRAUN für die Übernahme der Zweitkorrektur danken.

Herr XXX KIT Wettermessdaten freundlich und kurzfristig zur Verfügung Für die finanzielle und ideelle Förderung, die mein Studium durch zahlreiche Akademien, Tagungen und ein weiteres Auslandssemester im kommenden Semester bereichert hat, gebührt mein Dank besonders der STUDIENSTIFTUNG DES DEUTSCHEN VOLKES E.V.

Nicht zuletzt gilt mein Dank auch meiner Lebensgefährtin HENRIETTE BLANK und TOBIAS RUHLAND für das Korrektur lesen dieser Arbeit und alle weiteren Leuten, die mich während des gesamten Studiums und bei der Umsetzung dieser Arbeit unterstützt haben.

Inhaltsverzeichnis

Tabellenverzeichnis	III
Abbildungsverzeichnis	IV
Quelltextverzeichnis	VI
Symbolverzeichnis	VII
1 Einleitung	1
1.1 Motivation und Problemstellung	1
1.2 Zielsetzung und Aufbau der Arbeit	3
2 Theoretische Grundlagen	4
2.1 Grundlagen zur Modellprädiktiven Regelung	4
2.1.1 Optimalsteuerung	4
2.1.2 Modellprädiktive Regelung	6
2.2 Technische Grundlagen zur Kommunikation mit Bussystemen	6
2.2.1 Bussysteme	7
2.2.1.1 Informationsaustausch	7
2.2.1.2 Netzwerk und Topologie	7
2.2.1.3 Buszugriffsverfahren	10
2.2.1.4 Datensicherung	11
2.2.1.5 Schnittstellen	12
2.2.2 OSI-Kommunikationsmodell	16
2.2.3 Modbus Kommunikationstechnologie	21
2.3 Technische Grundlagen zur Modellbildung	28
2.3.1 Thermodynamische Systeme	28
2.3.2 Erster Hauptsatz der Thermodynamik	30
2.3.3 Wärmeübertragung	31
2.4 Technische Grundlagen zur Solar- und Gebäudetechnik	33
2.4.1 Außenklima und Komponenten	33
2.4.1.1 Die Außenlufttemperatur	34
2.4.1.2 Sonnenstrahlung	34
2.4.2 Gebäudetechnik Glas und Wärmedurchgangskoeffizienten	34
3 Anlagendesign	35
3.1 Analyse der Anforderungen	35
3.1.1 Einsatzziele und Rahmenbedingungen	35
3.1.2 Definition der Anforderungen	36

3.2 Ziel und die Idee der Anlage	41
3.2.1 Aufgabe der Anlage	41
3.2.2 Idee der Anlage	41
3.3 Konzept und Planung	42
3.3.1 Netzwerkarchitektur	43
3.3.2 Erfassung der Raumtemperatur	43
3.3.3 Steuerung des Heizkörpers	44
3.4 Installation und Inbetriebnahme	46
3.4.1 Hardware	46
3.4.2 Software	48
3.4.3 Inbetriebnahme mit Zweipunktregler	54
4 Modellbildung und Simulation	59
4.1 Modellbildung	59
4.1.1 Anforderungen an das Raummodell	59
4.1.2 Das Grundmodell des Raumes	60
4.1.3 Modellerweiterung durch Berücksichtigung der realen Umgebung .	61
4.1.4 Modellerweiterung durch Berücksichtigung der räumlichen Gegebenheiten	63
4.1.5 Das Heizkörpermodell	64
4.1.6 Modellerweiterung durch Berücksichtigung der Sonneneinstrahlung und Störgrößen	66
4.2 Simulation und Modellanpassung	70
4.2.1 Simulation und Validierung des Modells	70
4.2.2 Anpassung des Modells	73
5 Schlussbetrachtung	78
5.1 Fazit	78
5.2 Ausblick und Ansatzpunkte für weitere Arbeiten	78
A Modelle, Programme, Messdaten	79
B Modelle	80
B.1 Raummodell	80
Literaturverzeichnis	86

Tabellenverzeichnis

Tab. 3.1: Umsetzung der Ziele in Anforderungen der Anlage	37
Tab. 3.2: Netzwerkkonfiguration der Ports des EX9132M-MTCP Gateways	47
Tab. 4.1: Eigenschaften des Raummodells	61
Tab. 4.2: Weitere Eigenschaften des Raummodells	63
Tab. 4.3: Eigenschaften des Heizkörpermodells	65

Abbildungsverzeichnis

Abb. 2.1:	Variablen und Nebenbedingungen eines optimalen Steuerungsproblems	5
Abb. 2.2:	Bus-Struktur	8
Abb. 2.3:	Impulsverzerrung auf einer Leitung	9
Abb. 2.4:	Baumstruktur	10
Abb. 2.5:	<i>Cyclic Redundancy Check</i>	13
Abb. 2.6:	Parallele und serielle Datenübertragung	14
Abb. 2.7:	Spannungspiegel und Stecker der EIA 232-Schnittstelle	15
Abb. 2.8:	Spannungspiegel und Stecker der EIA 485-Schnittstelle	16
Abb. 2.9:	Die sieben Schichten des Open System Interconnection Modells	18
Abb. 2.10:	Die vier Dienstvorgänge	19
Abb. 2.11:	Die Modbus Kommunikation im OSI-Referenzmodell	22
Abb. 2.12:	Allgemeiner Rahmen für Telegramme nach dem Modbus Anwendungsprotokoll	23
Abb. 2.13:	Transaktion mit dem Modbus Protokoll	23
Abb. 2.14:	Datenmodell und Adressierung nach dem Modbus Protokoll	24
Abb. 2.15:	Serielle Kommunikation über Modbus RTU	26
Abb. 2.16:	Die Modbus TCP/IP Kommunikationsarchitektur	27
Abb. 2.17:	Angepasster Rahmen für Telegramme nach dem Modbus TCP/IP Protokoll	27
Abb. 2.18:	Komponenten des Außenklimas	34
Abb. 3.1:	Raumskizze K004A vom K Gebäude der Hochschule Karlsruhe – Technik und Wirtschaft	38
Abb. 3.2:	Prinzipskizze eines technischen Systems zur Raumtemperaturregelung des Raumes K004b	42
Abb. 3.3:	Aufbau des Netzwerks	43
Abb. 3.4:	Verteilung der Raumtemperaturfühler	44
Abb. 3.5:	Schaltplan der Raumtemperaturregelungsanlage	47
Abb. 3.6:	UML Klassendiagramm der zentralen Anlagensteuerung	57
Abb. 4.1:	Grundmodell eines Raumes	60
Abb. 4.2:	Erweitertes Raummodell	64
Abb. 4.3:	Erweitertes Raummodell	67
Abb. 4.4:	Simulation des Raummodells ohne Einsatz der Heizung	71
Abb. 4.5:	Simulation des Raummodells mit Einsatz des Heizkörpers	72
Abb. 4.6:	Parameterschätzung des Wärmedurchgangskoeffizienten der Wand von zwei Intervallen	74
Abb. 4.7:	Parameterschätzung	75
Abb. 4.8:	Simulation des Raummodells ohne Einsatz der Heizung mit geschätztem Parameter	76

Quelltextverzeichnis

3.1 Klasse SensorsHttp zur Abfrage der Temperaturen vom Webthermograph8x	48
3.2 Klasse zum Verbindungsaufbau und Für die Grundfunktionen über Modbus TCP	50
3.3 Klasse zum Auslesen über Modbus TCP	51
3.4 Klasse zum Auslesen über Modbus TCP	52
3.5 Die Actutators und Sensors Klassen zur Bedienung der Anlage	53
3.6 Zweipunktregler Programm zur Inbetriebnahme der Anlage	55
4.1 Einfaches Gleichungssystem für das Grundmodell des Raumes in Modelica	61
4.2 Erweitertes Gleichungssystem Modell des Raumes unter Berücksichtigung der realen Umgebung in Modelica	62
4.3 Erweitertes Gleichungssystem des Raumes unter Berücksichtigung der räumlichen Gegebenheiten in Modelica	63
4.4 Auszüge des Modelica Modells vom Heizkörper in K004b	65
4.5 Erweitertes Gleichungssystem Modell des Raumes unter Berücksichtigung der Sonneneinstrahlung und Störgrößen	67
4.6 Programm zur Umrechnung der Globalstrahlung in die effektive Solarstrahlung an der Fensterfront am Rum K004b	68
4.7 Fenster als Subkomponente des Raummodells	69
listings/room_model_backup.mo	80

Symbolverzeichnis

Hinweis: Bei der Angabe der Symbole soll sich auf die Wesentlichen beschränkt werden. Die jeweils zutreffende Bedeutung ergibt sich entweder aus dem Kontext oder ist explizit im Text angegeben.

Formelzeichen

$A_{exchange}$	Wärmeaustauschoberfläche [m^2]
c_p	Spezifische Wärmekapazität eines Stoffes [$\frac{J}{kg*K}$]
E	Gesamtenergie eines Systems [J]
E_{kin}	Kinetische Energie eines Systems [J]
E_{pot}	Potenzielle Energie eines Systems [J]
f_{max}	Maximale Übertragungsfrequenz [Hz]
T_0	Celsius Nullpunkt bei [273, 15K]
T	Kelvin Temperatur [K]
m	Masse [kg]
\dot{m}	Massenstrom [$\frac{kg}{s}$]
m_{sys}	Masse eines Systems [kg]
P	Leistung [W]
p_{high}	High-Pegel für ein Signal, entspricht logischer „1“
p_{low}	Low-Pegel für ein Signal, entspricht logischer „0“
Q	Wärme [J]
\dot{Q}	Wärmestrom [W]
t	Celsius Temperatur [$^{\circ}C$]
Δt_{Imp}	Impulsverzerrung [s]
$U - Wert$	Materialabhängiger Wärmedurchgangskoeffizient [$\frac{W}{K*m^2}$]
U	Innere Energie eines Systems [J]
W	Arbeit [J]

Griechische Buchstaben

κ	Erwartete Umsetzungsschwierigkeit
----------	-----------------------------------

Ω	Systemelement (Raum, Organisation, Technik)
ρ_n	Gewichtung des Wandlungspotentialmerkmals n
$\lambda_{abs,n}$	Absolute Teilweichtigkeit des Merkmals n
$\lambda_{rel,n}$	Relative Teilweichtigkeit des Merkmals n

Lateinische Buchstaben

kt	Kundentakt
$rw_\emptyset(t)$	Mittlere Lagerreichweite in Periode
t_{AZ}	Verfügbare Arbeitszeit in Periode
$T_{\emptyset BZ}$	Mittlere Gesamtbearbeitungszeit
$T_{\emptyset DLZ}$	Mittlere Gesamtdurchlaufzeit eines Produkts
$T_{\emptyset PZ}$	Mittlere Gesamtprozesszeit
$x_{\emptyset B}(t)$	Mittlerer Bedarf in Periode
$x_{\emptyset LB}$	Mittlerer Lagerbestand

Mathematische Operatoren

$(a; b]$	Halboffenes Intervall
\Leftrightarrow	Genau dann, wenn
\cong	Näherungsweise
\forall	Für alle
\in	Ist Element von
\mathbb{N}^+	Menge der natürlichen Zahlen ohne 0
\mathbb{R}	Menge der reellen Zahlen

„Erfolgreich zu sein setzt zwei Dinge voraus: Klare Ziele und den brennenden Wunsch, sie zu erreichen.“
— JOHANN WOLFGANG VON GOETHE

1 Einleitung

1.1 Motivation und Problemstellung

Die neueren Entwicklungen in den südlichen und östlichen Staaten Asiens, allen voran China und Indien, haben einen starken Einfluss auf den enormen Anstieg des weltweiten, zukünftigen Energiebedarfs. In einem zentralen Szenario ihrer Prognosen, schätzt die Internationale Energie-Agentur (IEA) den Energieverbrauch im Jahr 2040 um ein Drittel höher ein als im vergangenen Jahr 2015 [Agency, 2015, S. 1].

Zudem sieht das Intergovernmental Panel on Climate Change (IPCC) den Menschen als eine der Hauptursachen für den Klimawandel, insbesondere für die globale Erderwärmung [Core Writing Team u. , eds., S. V]. Eine Schlüsselrolle bei der globalen Erwärmung spielt der Ausstoß von Treibhausgasen. Einen großen Anteil davon entsteht bei der Erzeugung von Elektrizitäts- und Wärmeenergie, wobei große Mengen an CO₂ freigesetzt werden[Core Writing Team u. , eds., S. 47].

Daher wird die Bedeutung der Energieerzeugung auch in Zukunft weiter zunehmen. Der Energieerzeugung aus erneuerbaren Energien, welche ohne den Ausstoß schädlicher Treibhausgase auskommen, bietet sich damit eine enorme Chance. Dies wird beispielsweise durch die Zielsetzung der Bundesregierung in Deutschland belegt, deren Ziel es ist, im Jahre 2050 den Energiebedarf zu 80 % aus Erneuerbaren Energien zu decken [bi1, 2015, S. 2]. Als Erneuerbare Energien werden Energiequellen bezeichnet, die nach menschlichem Zeithorizont unerschöpflich sind, worin sich ein gewaltiges Potenzial begründet. Sie umfassen die Planetenenergie durch Gravitation und geothermische Energie, das jedoch mit Abstand größte Energieangebot bietet die Sonnenenergie. Das Angebot an Sonnenenergie übersteigt den gesamten weltweiten Energiebedarf um ein Vielfaches und könnte diesen daher theoretisch vollständig decken [Quaschning, 2011, S. 34f.].

Bevor das Potenzial zur Deckung des Weltenergiebedarfs aus Regenerativen Energien genutzt werden kann, gilt es jedoch noch einige Probleme zu lösen. Eines davon ist die Frage, wie die sich die technische Umwandlung der Solarenergie in eine nutzbare Energieform, wie beispielsweise Wärme oder Elektrizität, möglichst effizient realisieren lässt und bestehende Technologien weiter optimiert werden können. Wichtige Technologien bisher sind solarthermische Kraftwerke, welche die Solarenergie zunächst in Wärme und anschließend in elektrische Energie umwandeln, sowie Solarkollektoren und -zellen zur CO₂-freien Erzeugung von Wärme und Strom [Quaschning, 2011, S. 36f.].

Ein weiteres großes Problem ist der steigende globale Energiebedarf, welchem durch

eine Erhöhung der Energieeffizienz entgegengewirkt werden kann. Die größten Einsparpotenziale bestehen dabei nicht in privaten Haushalten, sondern im Energieverbrauch der Industrie. Die Bundesregierung in Deutschland sieht darin zudem einen Investitionsmotor, da durch Einsparungen ein größerer monetärer Spielraum für Investitionen der Industrie und Konsum der Privathaushalte besteht [bi1, 2015, S. 2].

Damit ganzheitliche Lösungsansätze entwickelt und die Energiewende erfolgreich ge-
managt werden kann, ist die Forschung von zentraler Bedeutung. Die Forschung zur
Energiewende umfasst die Entdeckung neuer Technologien sowie die kontinuierliche
Verbesserung bestehender Technologien, unter anderem durch eine erhöhte Energie-
effizienz. Die Bundesregierung maßt im Zuge der Energiewende Deutschland eine
Vorreiterrolle zu und versucht diese durch verschiedene Forschungsprogramme zu
untermauern. Sie umfassen Projekte zur Energieversorgung aus Erneuerbaren Energie-
quellen, der Energiespeicherung und der Verbesserung der Energieeffizienz [bi1, 2015,
S. 11].

Einen Teil zu dieser Forschung trägt die Hochschule Karlsruhe mit einem Forschungs-
projekt bei, welches die Modellprädiktive Regelung (MPR)¹ einer Anlage zur solaren
Klimatisierung eines Fakultätsgebäude zum Ziel hat.

Die Modellprädiktive Regelung beschäftigt sich damit, wie allgemein ein Prozess, unter
Zuhilfenahme eines mathematischen Modells desselben und anhand eines gewählten
Optimalitätskriteriums, optimal – im mathematisch exakten Sinne – geregelt werden
kann. Wird als Optimalitätskriterium der minimale Verbrauch von Energie gewählt,
insbesondere von nicht-erneuerbar erzeugten Energien, trägt die MPR damit zur
Verbesserung der Energieeffizienz bei. Bei der Bildung der benötigten, komplexen und
physikalisch-motivierten Modelle ist außerdem ein grundlegendes Verständnis für die
einzelnen Bauteile und die ablaufenden Prozesse erforderlich, wodurch Verbesserungs-
potenziale einzelner Prozesse und Komponenten aufgedeckt werden können.

Konkret umfasst das Forschungsprojekt der Hochschule Karlsruhe den Aufbau einer
solaren Anlage, welche die in Solarkollektoren gewonnene Wärmeenergie nutzt, um
eine Adsorptionskälteanlage anzutreiben und damit das K Gebäude zu kühlen. Die
Planung und Installation der Anlage hat sich aufgrund einiger Probleme verzögert
und befindet sich daher derzeit noch im Aufbau Hochschule Karlsruhe Technik und
Wirtschaft.

Um bereits vorab nützliche Erfahrungen für die Inbetriebnahme der großen Solaranlage
zu sammeln sowie erste Forschungsergebnisse zur Modellprädiktive Regelung zu
erzielen, soll diese durch eine kleine Anlage mit solarer Anwendung ergänzt werden.
Diese kleine Anlage soll komplementäre Eigenschaften solaren Klimatisierung besitzen
und sich ebenfalls für eine Modellprädiktive Regelung eignen.

Diese Arbeit übernimmt diese Aufgabe und versucht durch die Installation einer kleinen
Anlage nützliche Erfahrungen zu sammeln, welche die Inbetriebnahme der großen

¹ Im Englischen und in der einschlägigen Literatur wird die Modellprädiktive Regelung auch als
Model Predictive Control (MPC) bezeichnet.

Solaranlage vereinfachen und die Anlaufzeit verkürzen können. Weiterhin soll durch die Modellbildung für die Modellprädiktive Regelung, zusammen mit der kleinen Anlage, ein komplementärer Forschungsbeitrag zur großen Solaranlage realisiert werden.

1.2 Zielsetzung und Aufbau der Arbeit

Das übergeordnete Ziel dieser Arbeit ist es also, die Forschung der Hochschule Karlsruhe auf dem Gebiet der Modellprädiktiven Regelung von solaren Anwendungen zu erweitern. Konkret sollen durch die Konzeption, Planung und Installation einer kleinen Anlage mit solarer Anwendung jene Chancen genutzt werden, welche die große Anlage nicht bietet. Als einfache solare Anwendung, die mit wenig Aufwand realisierbar ist, bietet sich die Regelung der Temperatur innerhalb eines sonnenbestrahlten Raumes an.

Als Problemstellung dieser Arbeit ergibt sich damit die Forschungsfrage, wie eine Anlage zur Raumtemperaturregelung und ein mathematisches Modell derselben aufgebaut sein müssen, um eine Modellprädiktive Regelung der Raumtemperatur zu ermöglichen.

Aus der Forschungsfrage lässt sich das konkrete Ziel ableiten, eine entsprechende Anlage zu Konzipieren, zu Planen und Umzusetzen. Des Weiteren erfolgt die Bildung eines Modells für die Nutzung mit Modellprädiktiver Regelung. Damit soll also eine Forschungsumgebung geschaffen werden, um verschiedene Steuerungs- und Regelungssystematiken zu untersuchen sowie eigene Regelungsalgorithmen zu entwickeln. Weiterhin soll mit dieser Arbeit Know-How generiert werden, dass bei der weiteren Forschung von Nutzen ist. Die konkreten Einsatzziele der Anlage wurden gemeinsam mit den beiden Forschungsprojektverantwortlichen der Hochschule Karlsruhe, in Person von Herrn ADRIAN BÜRGER und MARKUS BOHLAYER, definiert und sind in Kapitel 3.1 detailliert beschrieben.

Im Rahmen dieser Arbeit werden – anschließend an die Einleitung in 1 – die theoretischen Grundlagen in 2 ausgeführt. Zunächst wird die grundlegenden Theorie zu Model Predictive Control vorgestellt bevor anschließend weitere technische Grundlagen erklärt, welche für das weitere Verständnis dieser Arbeit benötigt werden.

Danach wird das technische System in Kapitel 3 Schritt für Schritt entwickelt, ausgehend von der Idee und den räumlichen Gegebenheiten/Nebenbedingungen, und weiter konkretisiert bis zur realisierten Umsetzung in eine funktionierende Anlage.

Dementsprechend werden zunächst das Konzept, die Planung und die technische Umsetzung der konkreten Anlage dargestellt, bevor anschließend die theoretischen Grundlagen von Model Predictive Control und zur Modellbildung erläutert werden. Anschließend wird das Modell für Model Predictive Control gebildet und ein erstes grobes Konzept zur Steuerung der Raumtemperatur vorgestellt. Abschließend wird eine Validierung des Modells versucht und findet eine Anpassung des Modells statt damit es künftig mit Model Predictive Control genutzt werden kann.

„Theorie ist die Mutter der Praxis.“

— LOUIS PASTEUR

2 Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen erläutert die benötigt werden, um die Forschungsfrage zu beantworten. Zunächst MPC dann eher technisch

2.1 Grundlagen zur Modellprädiktiven Regelung

Die Modellprädiktive Regelung ist ein Teilgebiet der angewandten Mathematik zur Regelung von Prozessen oder technischen Systemen hinsichtlich gewählter Optimierungskriterien. Hierzu bedient sie sich mathematischer Modelle des zu steuernden Prozesses sowie einem Teilbereich der angewandten Mathematik, der Optimalsteuerung. Zur optimalen Regelung eines Prozesses, werden wiederholt Optimalsteuerungsprobleme gelöst, die jedoch besondere Anforderungen an das Modell erfordern [Diehl, 2014, S. 10]. Daher erfolgt zunächst eine kurze Einführung in die Optimalsteuerung bevor die Modellprädiktive Regelung genauer beschrieben wird.

2.1.1 Optimalsteuerung

Die Optimalsteuerung ist ebenfalls ein Teilgebiet der angewandten Mathematik und dient dazu, dynamische Systeme optimal zu steuern hinsichtlich spezifizierter Optimalitätskriterien. Dazu sollen also Steuersignale bestimmt werden, sodass das System eventuell vorgegebenen physikalischen und nutzerdefinierten Rahmen- sowie Randbedingungen genügt und zudem ein Leistungskriterium minimiert beziehungsweise maximiert wird [Kirk, 2004, S. 3f.]. Die modellierten Prozesse dynamischer Systeme lassen sich durch Zustände und Parameter beschreiben. Das Modell dann beschreibt das Verhalten des Prozesses, welches durch Rand- und Nebenbedingungen beschränkt sein kann. Weiterhin werden die Zustände von Steuergrößen beeinflusst. Diese Steuergrößen können frei oder innerhalb bestimmter Grenzen wählbar sein oder durch reale Einschränkungen fest vorgegeben, stellen jedoch einen Freiheitsgrad dar. Des Weiteren wird das Leistungskriterium durch die Kostenfunktion zum Ausdruck gebracht. Das Optimalsteuerungsproblem lässt sich dann durch die Kostenfunktion, Zustände, Parameter, Rand- und Nebenbedingungen und Steuergrößen wie folgt beschreiben [Diehl, 2014, S. 61]:

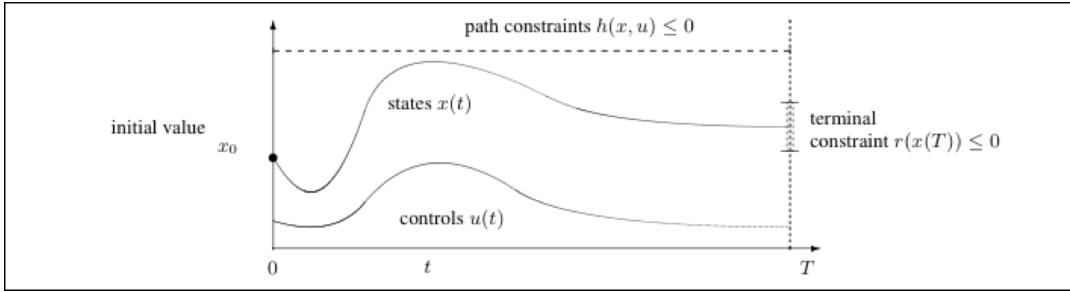


Abb. 2.1: Variablen und Nebenbedingungen eines optimalen Steuerungsproblems [Diehl, 2014, S.61]

$$\underset{\vec{x}(\cdot), \vec{u}(\cdot)}{\text{minimize}} \quad \int_0^T L(\vec{x}(t), \vec{u}(t)) dt + E(\vec{x}(T)) \quad , t \in [0, T] \quad (\text{Gl. 1})$$

subject to

$$\begin{aligned} \vec{x}(0) - \vec{x}_0 &= 0, && \text{(fixed initial value)} \\ \dot{\vec{x}} - f(\vec{x}(t), \vec{u}(t)) &= 0, && \text{(ODE model)} \\ h(\vec{x}(t), \vec{u}(t)) &\leq 0, && \text{(path constraints)} \\ r(\vec{x}(T)) &\leq 0, && \text{(terminal constraints)}. \end{aligned}$$

Die Zustände werden vom Zustandsvektor \vec{x} und die Steuergrößen vom Vektor \vec{u} , welche häufig als Controls bezeichnet werden, beschrieben. Die Parameter und das Prozessverhalten sind implizit im Modell beschrieben, welches durch ein Gleichungssystem aus gewöhnlichen Differenzialgleichungen (ODE Model) besteht. Die Kostenfunktion, die das Optimalitätskriterium hinsichtlich der Minimierung beschreibt, wird auch als Bolza Ziel bezeichnet und besteht aus dem Lagrange- und den Mayer-Term. Die Randbedingungen werden durch die Initialwerte (*fixed initial value*) und die Endbedingungen (*terminal constraints*) berücksichtigt. Die physikalischen und nutzerdefinierten Nebenbedingungen sind in den path constraints enthalten. Anschaulich werden diese Zusammenhänge in Abb. 2.1 dargestellt.

Für die Lösung dieses zeitkontinuierlichen Optimierungsproblems, müssen Bedingungen für die erste und zweite Ableitung erfüllt sein. Daher ist es nötig, dass das nichtlineare Optimalsteuerungsproblem zweifach stetig differenzierbar ist und damit auch das mathematische Modell des Prozesses [Diehl, 2014, S. 21ff.]. Konkret können mehrere, numerische Ansätze zur Lösung des Optimalsteuerungsproblems verfolgt werden, die sich in den Zustandsraum, die indirekten und direkten Verfahren einteilen lassen. Für reale Probleme, welche sich meist durch eine Vielzahl an Nebenbedingungen auszeichnen, eignen sich insbesondere die direkten Verfahren, die auch in der Praxis am weitesten verbreitet sind [Diehl, 2014, S. 63]. Die Lösung beschreibt einen hinsichtlich der gewählten Kriterien optimalen Einsatz von Steuergrößen Lösung und wird als Optimalsteuerungsplan bezeichnet. Da die Modellprädiktive Regelung nur den Rahmen

dieser Arbeit beschrieben wird nicht näher darauf eingegangen und für eine theoretische fundierte Einführung in die gesamte Thematik und die einzelnen Lösungsmethoden von Optimalsteuerungsproblemen auf Diehl [2014] verwiesen.

2.1.2 Modellprädiktive Regelung

Da zwischen realen Modellen und den tatsächlichen Prozessen immer Diskrepanzen bestehen, wird sich der reale Prozess durch die Umsetzung des berechneten Optimalsteuerungsplans mit höchster Wahrscheinlichkeit auch nicht exakt der Berechnung folgen. Dies wird auch als Model-Plant-Mismatch bezeichnet. Die Modellprädiktive Regelung eignet sich hervorragend, um diese Modellfehler sowie Störungen auszugleichen ohne eine umfassende Echtzeitregelung einzusetzen. Eine Echtzeitregelung lässt sich aufgrund des enormen Rechenaufwands ohnehin nur durch Näherungen und Vereinfachungen für sehr einfache Anwendungen umsetzen. Diese Lücke wird durch Modellprädiktive Regelung geschlossen, die ein beschränktes, immer gleich weit in die Zukunft schauendes Zeitintervall betrachtet. Zu diskret wiederholten Zeitpunkten wird das Verhalten eines Prozesses durch Kenntnis des aktuellen Zustandes und mit Hilfe eines Modells in dem Intervall vorhergesagt und versucht, dieses mit Optimalsteuerungsplänen dem gewählten Optimierungskriterium entsprechend zu beeinflussen. Dadurch kann auf Abweichungen, egal ob durch Modellabweichungen oder Störungen des Prozesses, zum Optimalsteuerungsplan zu den diskreten Zeitpunkten reagiert werden [Diehl, 2014, S. 71].

Im Rahmen dieser Arbeit wird eine Anlage aufgebaut und ein Modell gebildet, die eine Raumtemperaturregelung während des laufenden Betriebs mit Modellprädiktiver Regelung ermöglicht. Im ersten Schritt werden auf Basis von aktuellen Messungen des Systemzustandes und Kenntnis der nicht beeinflussbaren Steuergrößen zu jedem diskreten Steuerungszeitpunkt optimale Steuerungspläne bestimmt. Im nächsten Schritt soll die Modellprädiktive Regelung Vorhersagewerte für den äußeren, nicht beeinflussbaren Steuergrößen zur Bestimmung der optimalen Steuersignale nutzen. Die Anforderungen die sich daraus an ein Modell ableiten, werden bei der Modellbildung in Kapitel 4 explizit spezifiziert.

2.2 Technische Grundlagen zur Kommunikation mit Bussystemen

In diesem Kapitel werden die Grundlagen von Hard- und Software beleuchtet die für die Kommunikation der Steuerung mit den einzelnen Anlagenteilen benötigt werden. Diese umfassen zunächst Bussysteme im Allgemeinen, dann das OSI Modell für techn. Kommunikation und werden anhand des spezifischen/konkreten Anwendungsfalls Modbus erläutert.

Die Einführung wird an der Struktur von Schnell u. Wiedemann [2006] anlehnen.

2.2.1 Bussysteme

Um allgemein Prozesse zu überwachen, zu steuern oder regeln zu können, müssen zwischen den verschiedenen Prozessbeteiligten/einheiten Informationen ausgetauscht werden. Im Kontext dieser Arbeit gilt es einen Prozess zur Temperatursteuerung/Halten zu regeln. Die Prozessbeteiligten sind hierbei technische Bauteile, Aktoren und Sensoren sowie ein Steuerungsrechner/Controller, die zusammen ein technisches System bilden, dass im Folgenden als Anlage bezeichnet wird. Die Anlage zeichnet sich dadurch aus, dass sie eine eigenständige funktionale Einheit bildet einen eigenen Zweck verfolgt, die Raumtemperaturregelung und einen Mehrwert gegenüber ihrer Einzelteile hat, was dem Zusammenspiel der einzelnen Bauteile und Geräte entspricht. Um den Mehrwert zu realisieren und den Zweck zu erfüllen, werden Kommunikationssysteme benötigt. Mit Hilfe derer kann Kommunikation erfolgen und die Anlage ihren Zweck erreichen. Diese Kommunikationssysteme werden von technischer Seite oftmals als Bussysteme realisiert/umgesetzt. Bussysteme lassen sich anhand ihrer verschiedenen Ausprägungen von Merkmalen klassifizieren. Im folgenden Abschnitt werden zunächst die Merkmale von Bussystemen zum allgemeinen Verständnis dargestellt bevor auf die konkreten Ausprägungen des später eingesetzten Modbus-Bussystems eingegangen wird.

2.2.1.1 Informationsaustausch

Die Informationen über einen Prozess und dessen Zustand, werden auf höherer Ebene durch Daten und auf unterster Ebene durch einzelne Bits repräsentiert. Der Austausch dieser Daten zwischen den einzelnen Geräten findet in Form von Telegrammen statt. Ein Telegramm besteht grundsätzlich aus den zu übertragenden Daten und zusätzlich aus Informationen zur Übertragung. Die Daten werden vor der Übertragung in Rahmen, sogenannte data frames, eingeteilt, deren genauer Aufbau abhängig vom verwendeten Kommunikationsprotokoll innerhalb des Netzwerks ist [Schnell u. Wiedemann, 2006, S. 11f.]. Der Aufbau eines Telegramms am Beispiel des Modbus-Kommunikationsprotokolls wird in Abschnitt 2.2.3 erläutert und ist in Abb. 2.12 graphisch dargestellt.

2.2.1.2 Netzwerk und Topologie

Werden einzelne Prozesseinheiten miteinander über sogenannte Verbindungsleitungen, die zur Übertragung Informationen genutzt werden können, verknüpft, entstehen dabei Netzwerke und die Prozesseinheiten werden als Teilnehmer des Netzwerks bezeichnet. Ein Netzwerk lässt sich in einzelne Segmente einteilen und kann je nach Ausführung der Verbindungsleitungen und Anzahl der Teilnehmer unterschiedlich ausgeprägt sein. Anhand der geometrischen Anordnung lassen sich die folgenden, verschiedenen Netzwerktopologien unterscheiden.

Die einfachste Art, um eine Verbindung zwischen zwei Teilnehmern eines Netzwerks

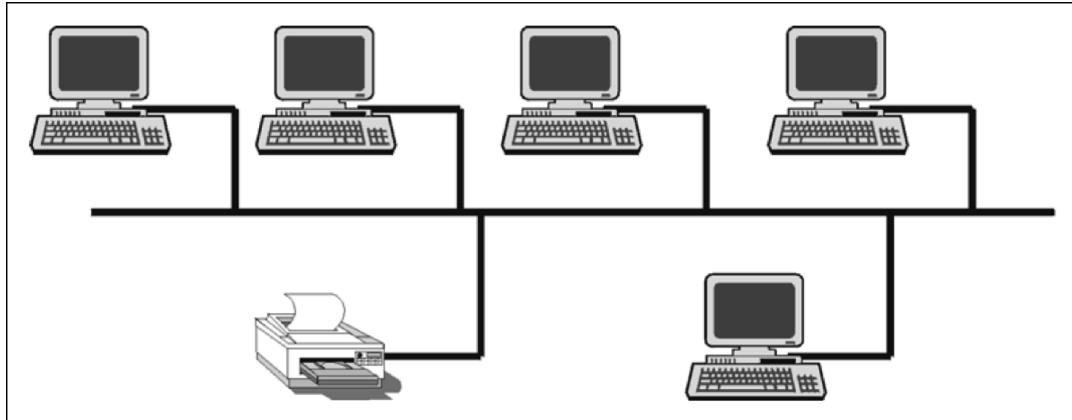


Abb. 2.2: Bus-Struktur aus [Schnell u. Wiedemann, 2006, S. 3]

herzustellen, ist die sogenannte Zweipunktverbindung. Dazu sind die Netzwerkeinnehmer durch eine direkte Leitung miteinander verbunden. Jedoch steigt mit der Anzahl von Teilnehmern auch der Verbindungsaufwand überproportional an, um bei solchen vermaschten Netzwerk alle Teilnehmer miteinander zu verbinden. Dies hat für große, vermaschte Netze zur Folge, dass eine unübersichtliche große Anzahl von Schnittstellen, ein extrem hoher Verkabelungsaufwand und damit verbundene hohe Kosten entstehen. Um diese Kosten zu vermeiden, ergeben sich noch verschiedene andere Möglichkeiten zur Anordnung von Teilnehmern in Netzwerken [Schnell u. Wiedemann, 2006, S. 1f.].

Um dem hohen Verkabelungsaufwand zu vermeiden, wird bei großen Netzwerken zu einer Linienstruktur übergegangen, die auch als Bus-Struktur bezeichnet wird und in Abb. 2.2 visualisiert ist. Charakteristisch für die Bus-Struktur ist, dass alle Teilnehmer entlang einer langen Verbindungsleitung, dem sogenannten Buskabel, angeordnet sind. Sie sind mit Hilfe von kurzen Stichleitungen an das gemeinsame Buskabel angebunden, über das die gesamte Kommunikation im Netzwerk erfolgt. Durch diese Anordnung wird der Verkabelungsaufwand sowie die Anzahl an Schnittstellen, insbesondere für sehr große Netzwerke, stark reduziert. Jedoch wird durch Nutzung einer gemeinsamen Kommunikationsleitung die gleichzeitige Kommunikation von Teilnehmern erschwert und es müssen sogenannte Buszugriffsverfahren definiert werden, welche lediglich Regeln für Zugriff auf den Bus festlegen. Weiterhin müssen durch die Parallelschaltung alle Teilnehmer ständig alle Sendungen mitverfolgen, wodurch der Sender stark belastet wird. Die Busleitungslängen sind meist sehr lange² und da die Länge auf die zu übertragende Wellenlänge bezogen nicht mehr vernachlässigbar klein ist, müssen Reflexionen durch Leitungsabschlusswiderstände an den beiden Enden der Busleitung unterbunden werden. Außerdem werden die Leitungslängen und die Teilnehmer je Netzwerksegment begrenzt??S. 3f.]schn06.

Ein weiterer begrenzender Faktor für die Leitungslänge ist der Fakt, dass die maximale Übertragungslänge und die maximale Übertragungsrate miteinander verknüpft sind und sich gegenseitig beschränken. Der Leitungs- und Kapazitätswiderstand einer Le-

² Diese reichen von mehreren hundert Metern bis teilweise in den Kilometerbereich, je nach Art und Einsatzort der Anwendung.

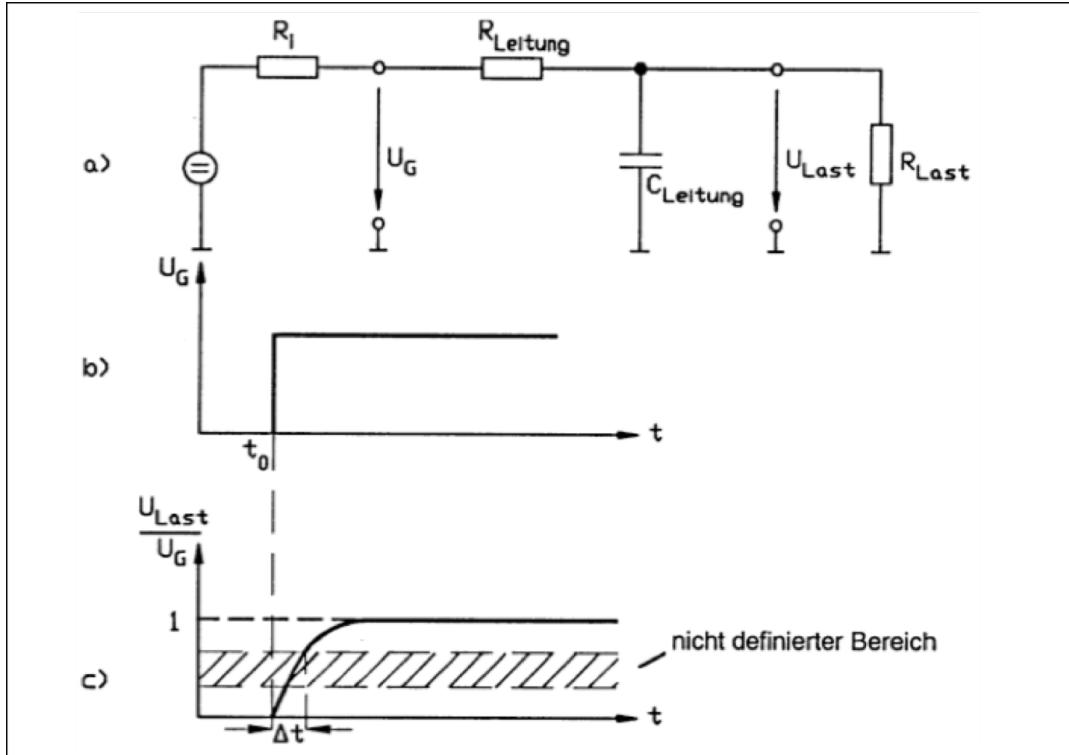


Abb. 2.3: Impulsverzerrung auf einer Leitung: a) Ersatzschaltbild der Anordnung b) Ausgangsspannung des Generators c) Empfängerspannung aus [Schnell u. Wiedemann, 2006, S. 4]

tung hängen von der Länge der Leitung ab und lassen sich durch das Ersatzschaltbild eines RC-Gliedes repräsentieren, wie in Abbildung Abb. 2.3 a) zu sehen ist. Durch die beiden Widerstände entsteht auf der Leitung eine Impulsverzerrung Δt_{Imp} , die somit mittelbar von der Leitungslänge abhängt. Je länger die Leitung wird, desto größer werden auch beiden Widerstände. Durch die erhöhte Leitungskapazität C_{Leitung} erhöht sich die Ladezeit und gleichzeitig sinkt durch den erhöhten Leitungswiderstand R_{Leitung} die Lastspannung U_G . Damit vergrößert sich die Impulsverzerrung Δt_{Imp} , wie in Abb. 2.3 b) und Abb. 2.3 c) dargestellt. Dadurch wird die maximale Frequenz f_{\max} der Datenübertragung auf den Kehrwert der Impulsverzerrung $f_{\max} = \frac{1}{\Delta t_{\text{Imp}}}$ beschränkt, da ansonsten der Empfänger den Wechsel des logischen Zustandes nicht mehr registrieren kann. In der Praxis bedeutet dies, dass die maximale Übertragungslänge und die maximale Übertragungsrate miteinander verknüpft sind und sich gegenseitig beschränken [Schnell u. Wiedemann, 2006, S. 4f.].

Um die Begrenzung der Leitungslängen zu korrigieren, wurde die Bus-Struktur zu einer Baumstruktur weiterentwickelt, welche in Abb. 2.4 dargestellt ist. Darin werden einzelne Netzwerk-Segmente, also einzelne Bus-Strukturen, durch Verstärkerelemente, sogenannte Repeatere, zu einem großen Netzwerk verknüpft. Um damit jedoch größere Flächen als mit der Bus-Struktur zu vernetzen und gleichzeitig die maximale Leitungslänge und die maximale Anzahl der Busteilnehmer zu vergrößern, wird jedoch eine galvanische Trennung der Teilnehmer voneinander benötigt [Schnell u. Wiedemann, 2006, S. 5 f.]. Die Besonderheit in dieser Struktur liegt also darin, dass sich durch

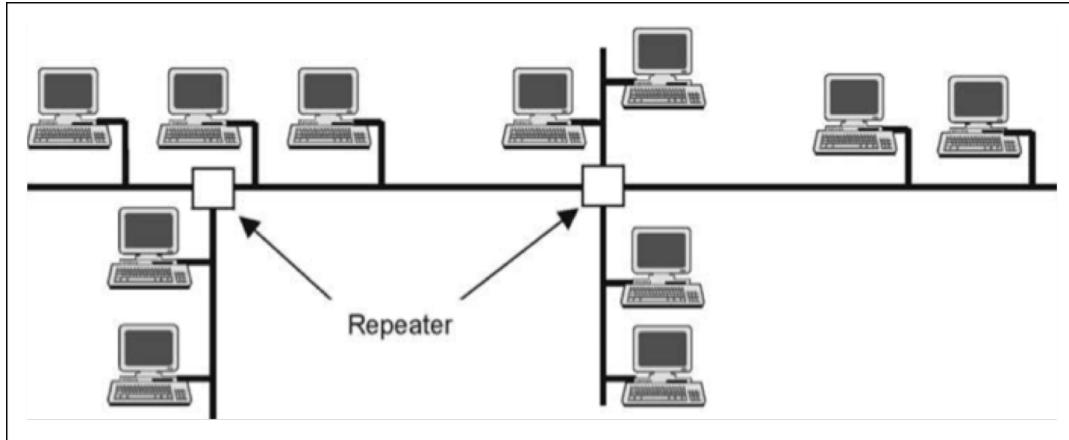


Abb. 2.4: Baumstruktur aus [Schnell u. Wiedemann, 2006, S. 5]

Ihren Aufbau bestehende Bus-Strukturen auch nachträglich einfach erweitern oder miteinander verknüpfen lassen. Die Bauteile zur Erweiterung von Netzwerken werden im Abschnitt 2.2.1.5 Schnittstellen vorgestellt.

Weitere wichtige Netzwerk Topologien, die für das Verständnis dieser Arbeit keine weitere Relevanz haben, sind die Ring- und die Stern-Struktur. Die Ring-Struktur ist durch einen physikalischer Ring von Zweipunktverbindungen aufgebaut und gekennzeichnet durch die Kommunikation der Teilnehmer übereinander hinweg. Die Stern-Topologie hingegen ist um eine Zentralstation herum ausgebaut, die mit allen Teilnehmer verbunden ist und über die gesamte Kommunikation abläuft. Der interessierte Leser findet in [Schnell u. Wiedemann, 2006, S. 6f.] detailliertere Ausführungen.

2.2.1.3 Buszugriffsverfahren

Die meisten Netzwerktopologien kommunizieren gemeinsam über eine Verbindungsleitung. Daher werden Regeln für den Zugriff definiert, um eine reibungslose Kommunikation zu ermöglichen. Die Buszugriffsverfahren lassen sich in zwei Gruppen aufteilen, der kontrollierten und zufälligen Verfahren [Schnell u. Wiedemann, 2006, S. 19].

Bei den kontrollierten Verfahren ist der Sender bereits vor Sendebeginn eindeutig bestimmt und eine Zuteilung des Busses ist nicht notwendig. Der Buszugriff findet entweder zentral innerhalb einer Zentralstation statt, bei sogenannten Master/Slave-Verfahren, oder wird dezentral durch Steuereinheiten vorgenommen, wie z.B. beim Tokenring und Tokenbus. Ein solches Verfahren heißt echtzeitfähig, wenn die Zykluszeit zur Datenübertragung berechenbar ist, aufgrund einer Beschränkung der Länge des Übertragungsintervalls und der maximalen Datenlänge. Bei zufälligen Buszugriffsverfahren greifen die Teilnehmer bei Bedarf auf die Verbindungsleitung zu und müssen sicherstellen, dass diese nicht gerade von einem anderen Teilnehmer belegt ist. Da nicht vorhergesehen werden kann, an welchem Zeitpunkt Informationen übertragen werden kann keine Echtzeitfähigkeit erreicht werden [Schnell u. Wiedemann, 2006,

S. 19].

Das Master/Slave-Verfahren besteht in der Regel aus einer Bussteuerungseinheit, dem sogenannten Master, und mehreren passiven Teilnehmern, den Slaves. Die Kommunikation wird ausschließlich vom Master initiiert, der die Verbindung zu den Slaves aktiv durch ein Request herstellt, in welchem die angeforderten Daten spezifiziert sind. Die Slaves treten nur nach Anfragen in Aktion und antworten darauf unmittelbar mit einer Response, die die angeforderten Daten des Masters enthält. In der Regel erfolgt die Kommunikation zyklisch zu allen Slaves gleichzeitig (Polling), damit der Master ein umfassendes und aktuelles Bild über den Systemzustand bekommt. Dadurch ergeben sich einfache Slaves, die günstig in den Bus eingebunden werden können, weil die gesamte „Intelligenz“ im Master implementiert ist. Jedoch gilt es bei diesem Verfahren zu beachten, dass der Informationsaustausch zwischen verschiedenen Slaves längere Zeit in Anspruch nehmen kann und bei einem Ausfall des Masters das gesamte Bussystem stillliegt [Schnell u. Wiedemann, 2006, S. 19ff.]. Auf eine weitere Beschreibung der übrigen Verfahren wird aufgrund der fehlenden Relevanz für diese Arbeit verzichtet, der interessierte Leser findet jedoch bei Schnell u. Wiedemann [2006] im Kapitel Buszugriffsverfahren ausführliche Informationen.

2.2.1.4 Datensicherung

Bei der Übertragung von Informationen besteht die Gefahr von Störungen, welche sich als Fehler in einer Nachricht durch Invertierung von Bits äußern. Störungen sind in der Regel technischer Art, wie zum Beispiel elektromagnetische Störsignale, Rauschen oder Potentialdifferenzen, und gegen einen Großteil lassen verschiedene Vorkehrungen treffen. Dadurch ergibt sich die Möglichkeit Störungen vorzubeugen oder diese nach Einsatz zu beseitigen. Der erste Ansatz ist also eine Verminderung des Auftretens durch technische Vorkehrungen, wie zum Beispiel Schirmung der Kabel, galvanische Trennung von Netzwerken oder differenzielle Signalübertragung. Der zweite Ansatz beschäftigt sich mit der Überwachung des Nachrichtenverkehrs und dem Ausbessern/Gegenmaßnahmen bei Fehlern [Schnell u. Wiedemann, 2006, S. 30].

Auf dem Gebiet der Buskommunikation entspricht eine Information oder Nachricht oder Telegramm codiert. Es werden stets Transparente Codes betrachtet, welche bitorientiert sind. Dabei sind jegliche Kombination von Bits erlaubt und man kann allein aus der Folge der Blitz nicht auf einen Fehler schließen. Definition Telegramm und Bit(0 und 1)

Die Vorkehrungen technischer Art können in aller Regel in den Spezifikationen der einzelnen Bussysteme enthalten, daher werden diese nicht weiter beschrieben und lediglich die Überwachung und Ausbesserung des Nachrichtenverkehrs erläutert. Bei der Übermittlung von Nachrichten können drei Arten von Fehlern auftreten: Der Fehler kann erkennbar und korrigierbar sein, erkennbar und nicht korrigierbar oder nicht erkennbar und damit auch nicht korrigierbar.

Kann der Fehler erkannt werden ist bereits ein großer Teil der Arbeit getan.

Fehlermaße sind die Bitfehlerrate $p = \text{Anzahl fehlerhafter bits} / \text{gesamtzahl gesendete bits}$, schlechtester Wert ist $p=0,5$, da durch invertierung immer wieder umstellbar. üblich in Technik $p = 10^{-4}$. ARQ(Error detection and automatic request repeat) ist normale Reaktion auf erkannte Fehler, dass eine einfache Wiederholung der Übertragung.

Restfehlerrate R ist eine wichtige Kennzahl weil, sie die unerkannten, fehlerhaften Bitfolgen die nach der Anwendung von Fehlererkennungsstrategien noch verbleiben misst. $R = \text{Anzahl unerkannt fehlerhafter Bitkombinationen} / \text{Gesamtlänge in Bits der Information}$. -> Maß für Unversehrtheit der Daten

Telegrammeffizienz auch wichtig, weil sie eine Aussage erlaubt wie viel Infomrtionen transportiert werden können und sie steht im Gegensatz zur Restfehlerquote, je sicherer Übertragung desto weniger Effizienz. $E = \text{fehlerfreie Infobits} / \text{Gesamtzahl übertragene bits} (\text{incl. Adresse Erro check usw.})$ [Schnell u. Wiedemann, 2006, S. 31ff.]

Die genauen Berechnungen der Wahrscheinlichkeiten und der mittleren Zeit zwischen zwei Fehlern sind und noch viel mehr Details finden sich im Kapitel Datensicherung in [Schnell u. Wiedemann, 2006, S. 31f.] zu finden.

Um Fehler systematisch zu erkennen, existieren verschiedene Fehlererkennungsstrategien. Die einfachste ist der Paritätsbit, der lediglich die Quersumme des Telegramms angibt, $P=0$ für eine ungerade und $p=1$ für eine gerade Quersumme. Damit können diejenigen Fehler entdeckt werden, die eine ungerade Anzahl an Bitflips haben, die mit geraden leider nicht. Eine Erweiterung der Paritätssicherung ist die Blocksicherung, bei der die Paritäten über ein Array aus mehreren Telegrammen überprüft werden kann [Schnell u. Wiedemann, 2006, S. 34f.].

Beim sogenannten Cyclic Redundancy Check, in der Literatur häufig als CRC-Check bezeichnet, wird ein Telegramm als Zahl aufgefasst. Im Sender wird diese Zahl durch das Generatorpolynom G geteilt. Das Ergebnis wird verworfen, lediglich der Rest bei der Division wird an das Telegramm angehängt. Der Empfänger dividiert das empfangene Telegramm durch dasselbe Polynom G und falls sich ein Rest von 0 ergibt, war die Übertragung fehlerfrei. Dadurch lassen sich abhängig vom Generatorpolynom G unterschiedliche Güten der Fehlererkennung realisieren. In Abb. 2.5 sind die Vorgänge beim Cyclic Redundancy Check graphisch zusammengefasst.

2.2.1.5 Schnittstellen

Die physikalische Übertragung der Daten/Telegramme kann über verschiedenste Schnittstellen geschehen und erfpigt binär. Je nach Topologie des Bussystems und Protokoll, kann die Datenübertragung seriell oder parallel geschehen wie in Abb. 2.6 zu sehen. Bei einer parallelen Datenübertragung werden immer mehrere Bits gleichzeitig übertragen, weshalb eine hohe Übertragungsgeschwindigkeit erreicht werden kann, jedoch eine aufwändige im Sinne eines vermaschten Netzes notwendig ist. Daher erfolgt

CRC (Cyclic Redundancy Check)	
1. Die Nachricht sei I	
Beispiel dezimal: $I = 14$	binär: $I = 110101$
2. Das Prüfpolynom sei G	
$G = \dots a_4 \cdot x^4 + a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0 \cdot x^0$	
d.: $\dots 0 \cdot x + 3 \cdot x^0 = 3$	b.: $\dots 1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x + 1 \cdot x^0 = 1011$
3. Der Hilfsvektor ist dann $H = 100 \dots 0$	
d.: $H = 10$	b.: $H = 10000$
4. Die Information I wird mit H multipliziert:	
$B = I \cdot H$	
d.: $B = 140$	b.: $B = 1101010000$
5. Das Produkt B wird durch G dividiert:	
$\frac{B}{G} = Q + \frac{R}{G}$	
d.: $\frac{B}{G} = \frac{140}{3} = 46 + \frac{2}{3}$	b.: $\frac{B}{G} = \frac{1101010000}{1011}$ $= 1111011 + \frac{101}{1011}$
6. Der Rest R wird B hinzugefügt und das Ganze gesendet	
d.: $B + R = 142$	b.: $B + R = 1101010101$
7. Der Empfänger bildet	
$(B - R) : G$	
d.: $(B - R) : G = 138 : 3$	b.: $(B - R) : G$ $= (B + R) : G = 1101010101 : 1011$ $= 46, R = 0$ $= 1111011, R = 0$
8. Es bedeutet $R = 0$ fehlerfreie Übertragung	

Abb. 2.5: Cyclic Redundancy Check aus [Schnell u. Wiedemann, 2006, S. 38]

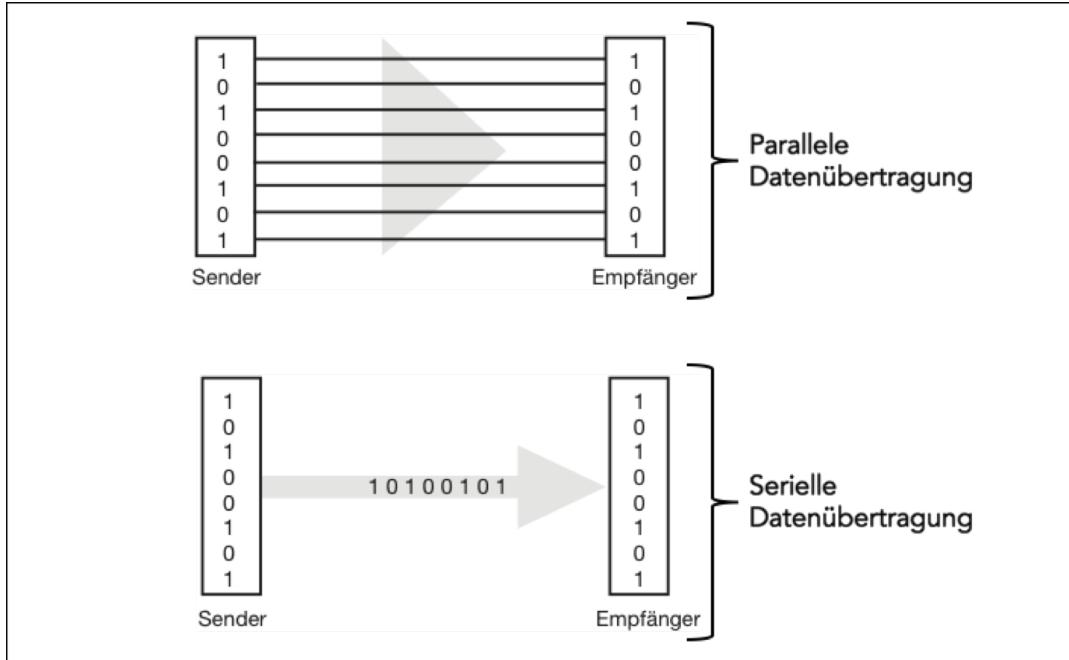


Abb. 2.6: Parallele und serielle Datenübertragung aus [Schleicher, 2008, S. 13]

in der Praxis eher eine serielle Datenübertragung, also die einzelne Übertragung von bits nacheinander über eine Leitung, wie sie zum Beispiel mit einer Bus-Struktur möglich ist [Schleicher, 2008, S. 13].

Bei der binären Datenübertragung werden lediglich zwei Zustände unterschieden, deren Signalwerte einen High-Pegel p_{high} und einen Low-Pegel p_{low} besitzen, die durch einen Bereich, in dem das Signal nicht definiert ist, voneinander getrennt sind [Schleicher, 2008, S. 9]. Die Geschwindigkeit der Datenübertragung wird in der Einheit Baud gemessen und definiert als übertragene Bits pro Sekunde (Bps)[Schleicher, 2008, S. 22].

Die gängigsten Übertragungsverfahren sind elektrische und optische Schnittstellen. Die elektrischen Schnittstellen gliedern sich wiederum in Strom- und Spannungsschnittstellen auf. Im weiteren Verlauf der Arbeit sind lediglich die Spannungsschnittstellen EIA³ 232 und EIA 485, welche auch als EIA 232 und EIA 485 bezeichnet werden, relevant, weshalb der interessierte Leser weitere Informationen zu weiteren Schnittstellen in [Schleicher, 2008, S. 13ff.] und [Schnell u. Wiedemann, 2006, S. 57ff.] findet.

Die EIA 232 Schnittstelle ist wie bereits erwähnt eine Spannungsschnittstelle, die für Punkt-zu-Punkt Verbindungen geeignet ist und deren Pegel in Abb. 2.7 zu sehen sind. Die Pegel sind für Spannungen zwischen $-3V < p_{high} < -15V$ als logische „1“, der Low-Pegel für Spannungen zwischen $-3V < p_{low} < -15V$ als die logische „0“ definiert. Im Intervall $[-3V, 3V]$ ist das Signal nicht definiert, weshalb dieser Bereich möglichst schnell durchlaufen werden sollte. Da der Signalpegel von der Datenleitung hin zur Masse gemessen wird kann er nicht symmetrisch sein und ist damit erdunsymmetrisch.

³ Abkürzung der Normen und Standards die von der Electronic Industries Alliance entwickelt wurden.

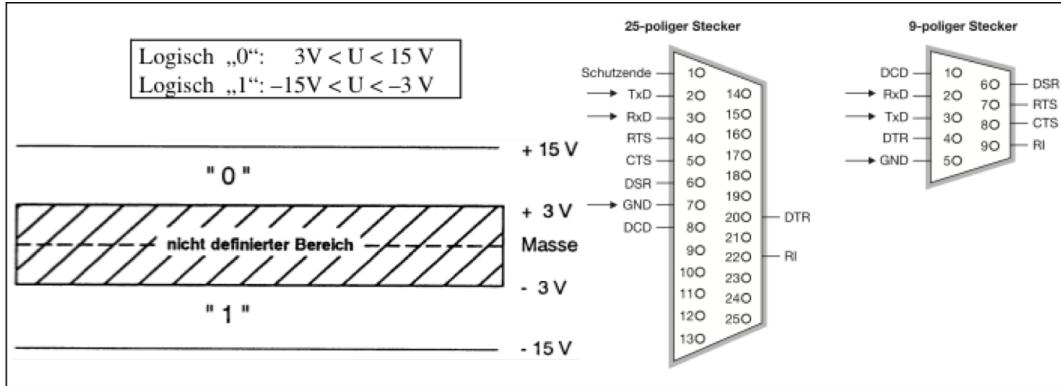


Abb. 2.7: Links: Spannungspegel EIA 232-Schnittstelle aus [Schnell u. Wiedemann, 2006, S. 57]
Rechts: Stecker EIA 232-Schnittstelle aus [Schleicher, 2008, S. 14]

[Schnell u. Wiedemann, 2006, S. 57f.]

Typischerweise steht eine solche Schnittstelle jedem Rechner als COM-Port zur Verfügung und in der Automatisierungstechnik werden nur die RxD (Receive Data), TxD (Transmit Data) und GND Leitungen, die ein gemeinsames Bezugspotenzial definiert, verwendet. Wichtig bei der Verkabelung ist, dass die übertragende Leitung TxD immer mit der empfangenden Leitung RxD verbunden ist [Schleicher, 2008, S. 14f.].

Die EIA 485 Schnittstelle ist ebenfalls eine Spannungsschnittstelle, die jedoch für Mehrpunktverbindungen geeignet ist und die in der Norm ISO 8482 beschrieben ist. Die Signalübertragung erfolgt über zwei Übertragungsleitungen, die in der Regel als verdrilltes und abgeschirmtes Zweidrahtleitung ausgeführt sind. Die Signalpegel entsprechen der Differenzspannung U_{AB} zwischen den beiden Leitungen, die innerhalb des Intervalls von $[-7V, 12V]$ bezogen auf die Masse liegen muss. Durch das verdrillte Leitungspaar, wirken sich mögliche Störgrößen auf die Spannung beider Leitungen gleichermaßen aus, wodurch die Spannungsdifferenz unverändert bleibt und eine erhöhte Störfestigkeit gegenüber der EIA 232-Schnittstelle erreicht wird. Bei der Pegelfestlegung werden für Empfänger und Sender gibt es verschiedene Vorgaben, wie in Abb. 2.8 graphisch dargestellt. So müssen die Sender eine Differenzspannung zwischen $-1,5V < U_{AB} < -5V$ für logische „1“ und $1,5V < U_{AB} < -5V$ für logische „0“ leisten können. Empfänger hingegen müssen in der Lage sein Spannungsdifferenzen von $U_{AB} < -0,3V$ als logische „1“ und $0,3 < U_{AB}$ als logische „0“ zu detektieren und zu interpretieren [Schnell u. Wiedemann, 2006, S. 59ff.].

Es besteht ein geringer Installationsaufwand und die Datenübertragung erfolgt über die TxD/RxD + und TxD/RxD - Leitungen, ein gemeinsames Bezugspotenzial über die GND Leitung wird nicht zwingend benötigt, jedoch wird ein Leitungsabschluss an beiden Enden des Buskabels benötigt. [Schleicher, 2008, S. 19f.].

Um Netzwerk zu erweitern, können verschiedene Bauteile eingesetzt werden. Sogenannte Repeater sind aktive Bauteile, die lediglich einen kurze Zeitverzögerung verursachen und nur Netzwerke der selben Schnittstelle miteinander verbinden kann. Er ist ein aktives Bauteil, dass das Datensignal lediglich verstärkt, indem er die empfangenen

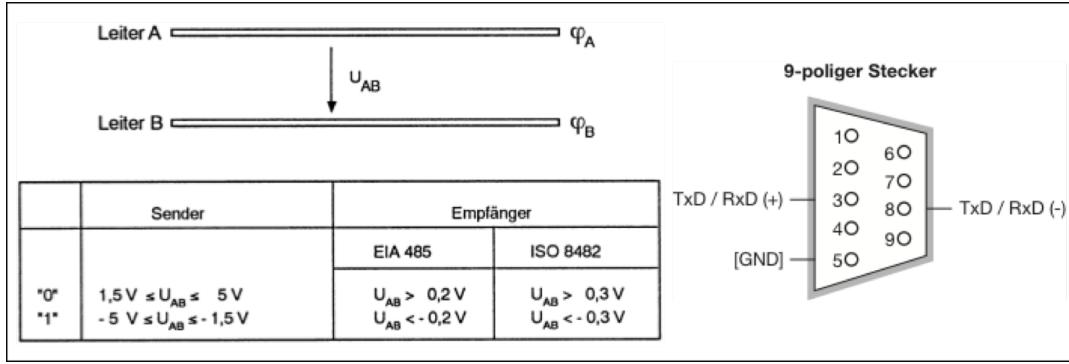


Abb. 2.8: Links: Spannungspegel EIA 485-Schnittstelle aus [Schnell u. Wiedemann, 2006, S. 60]
Rechts: Stecker EIA 485-Schnittstelle aus [Schleicher, 2008, S. 19]

Bits blind kopiert und verstärkt auf das angeschlossene Netzwerk überträgt, weshalb er für die Kommunikationsteilnehmer unsichtbar ist [Schnell u. Wiedemann, 2006, S. 79f.].

Eine Möglichkeit um Netzwerke verschiedener Art zu verbinden bieten Bridges und Gateways. Erstere werden auch als Schnittstellenumsetzer bezeichnet und kommen zum Einsatz, wenn trotz gleichem Übertragungsprotokoll genutzt unterschiedliche physikalische Schnittstellen oder Übertragungsmedien genutzt werden sollen [Schnell u. Wiedemann, 2006, S. 80f.]. Er erkennt die Datenflussrichtung automatisch und wandelt die Pegel der Schnittstellen in den jeweils anderen um [Schleicher, 2008, S. 21]. Die sogenannten Gateways dienen der Kopplung von Netzwerken, die verschiedene Architekturen aufweisen, also neben unterschiedlichen physikalischen Schnittstellen auch andere Übertragungsprotokolle verwenden. Das Gateway ist demnach umfassender als die Bridge und erweitert deren Funktionen um die Übersetzung der Signale von einem Übertragungsprotokoll in das jeweils andere [Schnell u. Wiedemann, 2006, S. 84f.].

2.2.2 OSI-Kommunikationsmodell

Aufgrund der großen Anzahl verschiedener technischer Systeme existieren auch viele verschiedene Arten der Kommunikation untereinander. Bei der genaueren Betrachtung der Kommunikation wird ersichtlich, dass diese oftmals ähnlich abläuft und sich durch ein Meta-Schema beschreiben lässt [Schnell u. Wiedemann, 2006, S. 8]. Um die Kommunikation auch über verschiedenen Systemen hinweg zu ermöglichen und sie zu formalisieren, wurde von der *International Organization for Standardization* 1984 ein abstraktes Referenz-Modell entwickelt, dass in der *ISO-Norm 7498-1* beschrieben ist. Es dient der Entwicklung und Verbesserung von Standards für den Informationsaustausch sowie als Referenz für bestehende Standard um eine gewisse Konsistenz zu wahren [osi, 1996, S. 1]. Das Ziel bei dem Entwurf des Modells war es, eine Menge von Standards zu schaffen um autonomen Systemen die Kommunikation untereinander zu ermöglichen [osi, 1996, S. 4].

Das sogenannte Open System Interconnection Modell wird zunächst allgemein erläutert, da die Kommunikation von technischen Systemen im Rahmen der Arbeit eine zentrale Rolle spielt, und wird anschließend im Anwendungskontext mit den eingesetzten Protokollen und Schnittstellen referenziert.

Zunächst wird im Standard definiert, womit sich das Modell beschäftigt und abgegrenzt welche Aspekte im Modell keine Berücksichtigung finden [osi, 1996, S. 3]:

*„OSI is concerned with the exchange of information between open systems
(and not the internal functioning of each individual real open system).“*

Das OSI-Modell beschäftigt sich also zentral mit dem Austausch von Informationen zwischen verschiedenen offenen Systemen und allen dabei anfallenden Aktivitäten. Diese sind sehr umfangreich und lassen sich in folgende Bereiche gliedern [osi, 1996, S. 3f.]:

- Der Austausch von Informationen zwischen offenen Systemen,
- die physischen Medien zur Verbindung von offenen Systemen und deren Transportmöglichkeit von Informationen,
- die Vernetzung von offenen Systemen,
- die Interaktion zwischen offenen Systemen und deren Fähigkeit zur Kooperation bei der Datenübertragung.

Bezogen auf den Austausch von Informationen überschneiden sich die physische Verbindung und die Vernetzung und entsprechen zusammen der Infrastruktur und deren Architektur, die zur Übertragung zur Verfügung steht. Die Interaktion umfasst weitaus mehr Aufgaben: Neben der Synchronisation der Prozesse, die Daten austauschen wollen, muss auch die Darstellung der auszutauschenden Daten und eventuell notwendige Transformationen beachtet werden, um eine Kompatibilität unterschiedlicher Systeme zu erreichen. Weitere wichtige Aufgaben sind die Datenspeicherung, deren Integrität und die Sicherheit beim Austausch hinsichtlich Fehler und Einsicht von Außen [osi, 1996, S. 4]. Es ist leicht zu erkennen, dass die technische Kommunikation einen sehr umfangreichen und komplizierten Prozess darstellt. Daher wird der Kommunikationsprozess im OSI-Modell stark abstrahiert und in sieben abstrakte Ebenen gegliedert. Die einzelnen Ebenen sind in Abb. 2.9 dargestellt und dienen dazu verschiedene Aufgaben des Kommunikationsprozesses in Teilaufgaben zusammenzufassen.

Die Ebenen werden Schichten genannt und haben klar definierte Aufgaben und Schnittstellen zu ihren Nachbarschichten. An diesen Schnittstellen werden Dienste bereitgestellt, die von den anderen Ebenen genutzt werden können. Durch diesen Aufbau können einzelne Schichten einfach bearbeitet oder ausgetauscht werden, ohne die Gesamtfunktionalität zu gefährden. Außerdem kann ein System auch aus Komponenten verschiedener Hersteller zusammengesetzt werden, womit diese Architektur nachweislich als Basis für offene Systeme dient. In Abb. 2.9 ist ebenfalls dargestellt, dass die Schichten eins bis vier auch als Übertragungsschichten beziehungsweise Trans-

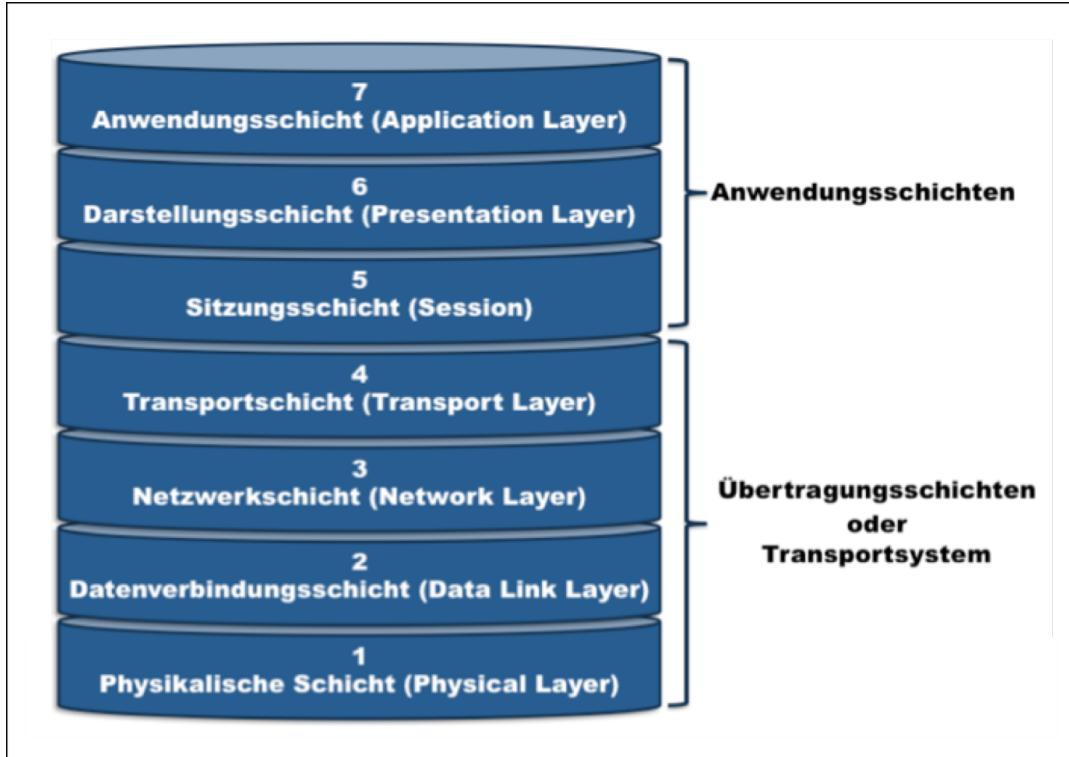


Abb. 2.9: Die sieben Schichten des *Open System Interconnection* Modells verändert nach [Schnell u. Wiedemann, 2006, S. 10] und [osi, 1996, S. 28]

portsystem zusammengefasst werden, weil sie für die Datenübertragung zwischen Systemen als gemeinsame Aufgabe haben. Die Schichten fünf bis sieben werden als Anwendungsschichten bezeichnet weil sie bei der Datenübertragung die Zusammenarbeit zwischen der Anwendersoftware und dem Betriebssystem sicherstellen [Schnell u. Wiedemann, 2006, S. 8f.].

Die Schnittstellen/Dienste zwischen den Schichten werden als Service Access Points bezeichnet und besitzen jeweils eine eindeutige Adresse, die oberhalb liegende Schicht ist der Service user, da er den Service daer unterhalb liegenden Schicht nutzt, dem service provider. Die Dienste können in verbindungsorientierte und verbindungsunabhängige unterschieden werden. Für den Datenausatausch stehen folgende Dienste zur Verfügung Bei der Abhandlung der Dienstaufgaben stehen die vier Dienstvorgänge der Client/Server Architektur zur Verfügung, die zusammengefasst in Abb. 2.10 abgebildet sind [mod, 2006a, S. 2f.] :

request - Anforderung indication - Meldung response - Antwort confirmation - Bestätigung Bestätigten Diensten stehen alle vier Vorgänge zur Verfügung, unbestätigten lediglich die Anforderung und Meldung. Typische Dienste sind Connect, disconnect, data

[Schnell u. Wiedemann, 2006, S. 14f.].

Im Folgenden wird kurz auf die einzelnen Schichten von Unten nach oben eingegangen bevor das Zusammenwirken der einzelnen Schichten anhand eines Beispiels verdeutlicht wird.

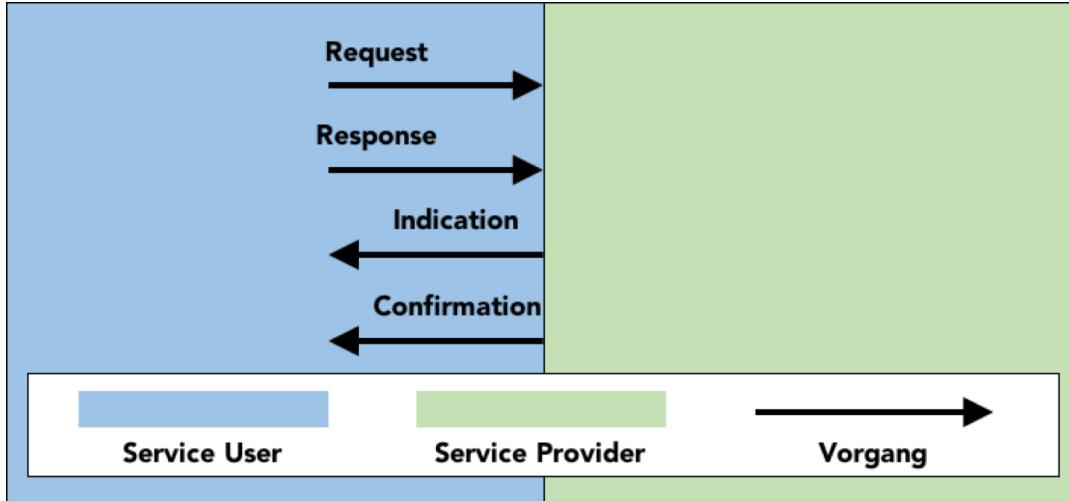


Abb. 2.10: Die vier Dienstvorgänge

Die erste, physikalische Schicht stellt die mechanischen und elektrischen Möglichkeiten zur physischen Verbindung von Systemen zur Verfügung, um die Datenübertragung der einzelnen Bits zu ermöglichen [osi, 1996, S. 49f.]. Sie legt also die mechanischen und elektrischen Eigenschaften der Übertragung fest, also die Endsystemkopplung (Stecker), die Kabelspezifikationen und die Zuordnung der Anschlüsse sowie die Art der Codierung und die Spannungspegel zur Übertragung. In der Regel werden dazu bestehende Normen genutzt, wie zum Beispiel die elektrischen Übertragungsstrecke nach EIA 485-Norm, welche im Folgenden noch erläutert wird.

Ein wichtiger Aspekt der Schicht ist es, dass die Spezifikation der Strecke und nicht das physikalische Medium selbst Teil der Schicht eins ist, denn die Kommunikation ist unabhängig von der konkreten Ausprägung der Schicht [Schnell u. Wiedemann, 2006, S. 9].

Die zweite Schicht betrachtet Kommunikation zwischen zwei systemen. Deshalb stellt die Datenverbindungsschicht, stellt funktionale und prozedurale Möglichkeiten für den verbindungsaufbau/trennung erhaltung und den Transfer von Dateneinheiten Verfügung. Ermöglicht dem Netzwerkschicht die Kontrolle über die Verbindung von Data circuits physikalisch, sowie fehlerabfangen der physikalischen schicht [osi, 1996, S. 46f.] Aufgabe ist sicherer transport von Station zu station. Datensicherung -> Verpacken um Übertragungsfehler erkenntlich zu machen in data frames. In Frames sind die maximale Anzahl Datenbits für Rohdaten spezifiziert, weiterhin wird Information zur Übertragung hinzugefügt. Die zusatzinfo kann Prüfsumme und Anfang und Ende des Rahmens enthalten oder quittierung eines telegramms und dient dazu fehlerhafte Übertragung oder etwas verloren gegangen zu überprüfen. MAC mit Schicht eins, LLC mit Schicht drei. Wuelle Wiki bisher MAC regelt den Zugriff auf das ohyische Medium zur Kommunikation, kontrolliert oder konkurriert. LLC verteilt die Daten passend in Schicht drei und gibt die Daten von schihct drei an passende MAC für schicht eins weiter und fügt Diese Infos von oben hinzu (Adressen Empfänger und Sender und zusatzinfo wie control für Steuerung von Datenfluss oder so).

Wichtig Die schicht hat jedoch keine Kenntnis über Inhalte der Daten![Schnell u. Wiedemann, 2006, S. 9ff.]

verbindungsloser Dienst heisst keine Verbindung zwischen Kommunikationspartnern, Datenpakete werden wie Brief ganz in Netzwerk gespeist mit Zieladresse versehen und weitertransportiert, ohne beeinflussung des transportweges durch benutzer des netzwerkdienstes. Später ist Modbus ein Beispiel dazu erklärt. verbindungsorientierte Dienste heisst ein virtueller Kanal zwischen kommunikationspartnern wird zur Verfüfung gestellt, eingeriechitet: Verbindungsaufbau, Datenaustausch, Verbindungsabbau, wie telefongespräch. [Schnell u. Wiedemann, 2006, S. 11f.]

Die dritte Schicht beschäftigt sich mit dem Netzwerk als Ganzes. Die Aufgaben der Netzwerkschicht hängen ein wenig ab von verbindungsorientierung Daher beschäftigt sie sich mit dem Aufbau , der erhaltung und Datenaustasuch und dem trennen von Netzwerk Verbindungen zwischen offenen systemen i Netzwerk, also schnittstellen. weietrhin ist sie für den Transport von Daten im netzwerk zustaändig, also insbeonsdere auch für die Festlegung der Route(Wegsteuerung) der Daten im Netzwerk [osi, 1996, S. 41f.]. Also Kontrolle von Verkehr im Netzwerk, d.h. Anzahl Pakete im Netzwerk, Staus [Schnell u. Wiedemann, 2006, S. 11f.]

Die vierte Schicht ist die Transportschicht und zuständig für die transparente Übertragung von Daten zwischen Prozessen und ist völlig unabähngig bzw losgelöst von den Gedanken an Kosten und Verlässlichkeit der Datenübertragung, da dies aufgaben der unteren schichten sind. Sie kümmert sich um die optimale nutzung von Netzwerk services/Nutzung [osi, 1996, S. 37f.]. Adressierung der Teilnehmer, Aufbau und Abbau für Transportverbindung wzischen Kom.Partner prozessen(Sammel Einzel Mehrere), Fehlerbehalndlung verbidnung und flusskontrolle, Synchronisierung der dtaenaustauschenden Prozesse.Zerlegung der Daten aus Sitzugnsschicht in Transportierbare Einheiten. Internetworking, Umsetzung verschiedener Protokolle Gateway Aufgaben. Aufbau Verbindung legt Art fest, Punkt zu Punkt oder Broadcast/Multicast (Alle bzw einige Teilnehmer gleichzeitig) [Schnell u. Wiedemann, 2006, S. 12f.].

Die fünfte Schicht, die Sitzungsschicht, startet eine Sitzungsverbindung mit bestimmter Adresse wenn dieser Prozess von einer höheren Schicht angefordert wird. Diese Verbindung dient dazu, den dialog von kooperiendnen Porzesse auf der eines höhreren Darstellungsbene durch eine Sitzungsverbindung zu synchronisieren und deren Datenaustausch zu organisieren. verknüpft die Sitzungsadressen mit den Transportadressen, also die Anwendungsschiten mit dem Transportsystem [osi, 1996, S. 35]. Benutzung des Transportsystems über die Schnittstelle zur Transportschicht. Je nach Funktionen der höheren Schihcten entsprechender Funktionsumfang BCS Basic Combined Subset - Verbindungssteuerung und Datenübertragung BAS Basic Activity Subset - Aktivitätsverwaltung BSS Basic synchronized Subset - Synchronisierung [Schnell u. Wiedemann, 2006, S. 13].

Die sechste Schicht, die Darstellungsschicht ist nach [osi, 1996, S. 33f.] für die Darstellung der Daten die von Anwendung-Entitäten entweder kommuniziert oder bei deren

Kommunikation referenziert werden. Sie stellt außerdem eine gemeinsame Represan-
tion der übertragenen Daten dar zwischen Anwendungs-Entitäten und befreit diese
dadurch von Syntaxabhängigkeiten. [Schnell u. Wiedemann, 2006, S. 13f.] stellt fest,
dass die Dienste die der Darstellung der transferierten Daten dienen wie die Codierung
der zu übertragenden Daten, der verwendete Zeichensatz und die Darstellung der
Daten auf dem Bildschirm oder Drucker. Semantik/Syntax beim Nachrichtenaustausch
und der beiden Kommunizierenden Prozesse. Evtl Komprimierung um Zeit und Kosten
zu sparen .

Die siebte und letzte Schicht stellt lediglich eine Möglichkeit für Anwendungsprozesse
zur Verfügung um auf die OSI Umgebung zuzugreifen. Jeder Anwendung stellt
im OSI genau einen Anwendungsprozess dar, verschiedene Anwendungsprozesse für
verschiedene Anwendungen und vice versa [osi, 1996, S. 32] stellt Funktionen bereit,
mit denen der Benutzer auf das Kommunikationssystem zugreifen kann, wobei der
Benutzer idR ein Computerprogramm und kein Mensch ist. [Schnell u. Wiedemann,
2006, S. 14]

Im folgenden werden die verwendeten Modbus Protokolle und Spezifikation ind An-
wendung Berzug zum OSI Modell gebracht um den praktischen Nutzen davon klar zu
machen.

2.2.3 Modbus Kommunikationstechnologie

Zunächst erfolgt die Einordnung ins OSI Modell und anschließend eine Klassifizierung
der Spezifikationen gemäß der zuvor von Bussystemen.

Das Modbus Protokoll teilt sich auf verschiedene Protokolle auf, zum einen auf das Ap-
plication Layer Messaging Protocol auf oberster Ebene, welches auf das Modbus Over
Serial Line Protocol sowie das Ethernet TCP/IP Protokoll aufbaut. Das Application
Layer Messaging Protocol lässt sich im OSI Referenzmodell in die siebte und oberste
Schicht(Anwendungsschicht) einordnen wie in Abb. 2.11 dargestellt. Bei der Nutzung
des Modbus over Serial line Protocol implementiert diese die zweite Schicht und die
Ebenen drei bis sechs sind leer implementiert. Als physikalische Schicht werden die
Übertragungsstandards nach EIA 485 oder nach EIA 232 [mod, 2006b, S. 2]. Anstatt
des Over serial line kann auch das Modbus Messaging On TCP/IP Protocol verwendet,
dann implementiert dieses zusammen mit dem dem TCP/IP Standard zusammen
die Netzwerkschicht im OSI-Referenzmodell. Das Ethernet implementiert dabei die
Datensicherungsschicht Nummer zwei und dessen physikalische Spezifikationen die
unterste physikalische Schicht und deren Schnittstellen. Die Ebenen vier bis sechs
sind auch bei Nutzung dieser Kommunikationsweise leer implementiert [mod, 2012,
S. 2f.]. Eine Einführung zum Ethernet und TCP/IP Standard findet sich in Schnell u.
Wiedemann [2006], detaillierte Ausführungen dazu in Furrer [2003].

Nun wird das Modbus Application Layer Messaging Protocol erläutert. Anschlie-
ßend werden die Modbus Protokolle der unteren Ebenen erläutert bis hin zu den

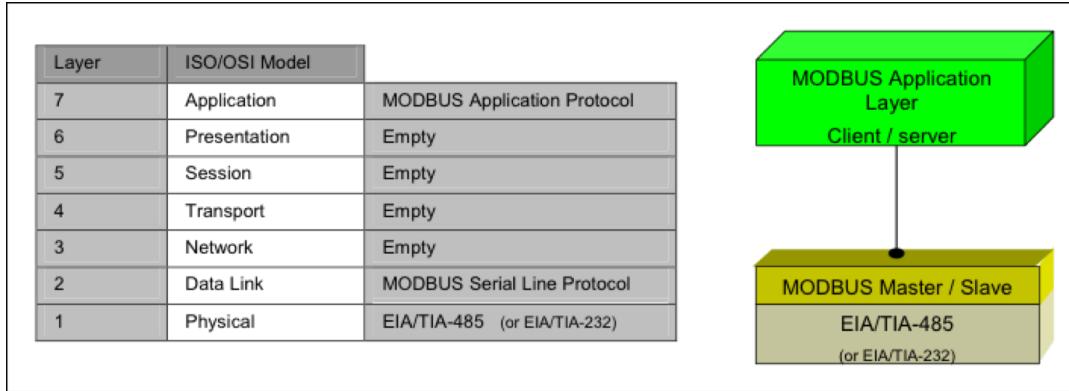


Abb. 2.11: Die Modbus Kommunikation im OSI-Referenzmodell aus [mod, 2006b, S. 5]

physikalischen Schnittstellen.

Das Modbus Application Layer Messaging protocol ist ein Protokoll, dass verschiedene Netzwerke und Bussysteme zur Master/Slave Kommunikationen von verbundenen Geräten nutzen kann [mod, 2012, S. 2f.], jedoch übernimmt im Rahmen des Protokolls der Master die Rolle des Clients und die Slaves die Rollen als Server, da der Client alleinig die Anfragen stellt und die Server lediglich auf Anfragen antworten. Das Modbus Protokoll definiert eine gemeinsame Telegrammstruktur, bezogen auf Inhalte und Rahmen der Nachricht. Damit ermöglicht es die Kommunikation zwischen verschiedenen Geräten innerhalb eines Netzwerks, welche unabhängig von Art und Typ des darunter liegenden Netzwerks sind. Es beschreibt außerdem wie die Kommunikation abläuft, wie der Client eine Anfrage an ein Server stellt und wie diese auf die Anfragen reagieren und antworteten und beschreibt wie Fehler bei der Übertragung entdeckt und darauf hingewiesen werden [MODICON, 96, S. 2f.]. Die Kommunikation kann dabei über eine serielle Leitung nach EIA 485 sowie über Ethernet Netzwerk erfolgen. Über Gateways, wie in Abschnitt 2.2 bereits erläutert, kann die Kommunikation auch über verschiedene Typen von Bussystemen oder Netzwerken geschehen [mod, 2012, S. 3f.].

Das Modbus Protokoll stellt ein allgemeines Gerüst für Telegramme zur Verfügung, welches in Abb. 2.12 dargestellt ist. Die einfache Protocol Data Unit (PDU) enthält die eigentlichen Informationen die ausgetauscht werden sollen und sind daher unabhängig vom Netzwerk oder dem Bussystems. Zu den Informationen gehören die eigentlichen Daten, welche leer sein können oder Daten enthalten die der Slave, in Rahmen von Modbus auch als Server bezeichnet, benötigt, und ein Funktionscode, der ein Byte groß ist und beschreibt welche Art der Aktion Reaktion gefordert ist/wird. Die sogenannte Application Data Unit (ADU) enthält zusätzlich zur PDU weitere Informationen zur Adressierung im Netzwerk und zur Fehlererkennung und ist daher abhängig vom Netzwerk. Der Buszugriff ist damit als Master-Slave geregelt.

Für die Kommunikation wird innerhalb des Masters, der im Rahmen von Modbus auch als Client bezeichnet wird, die Kommunikation durch eine ADU initialisiert. Das genaue Format der ADU wird nach Modbus Protokollspezifikation festgelegt. Das

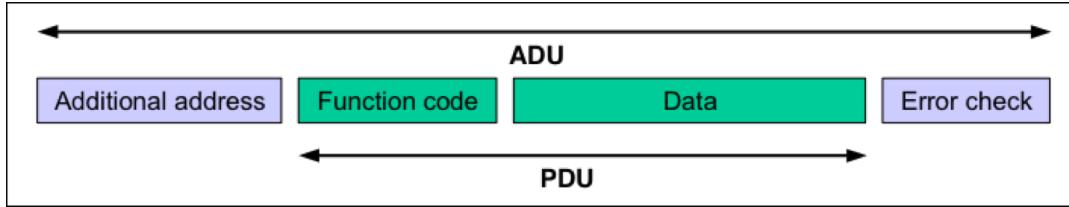


Abb. 2.12: Allgemeiner Rahmen für Telegramme nach dem Modbus Anwendungsprotokoll aus [mod, 2012, S. 3]

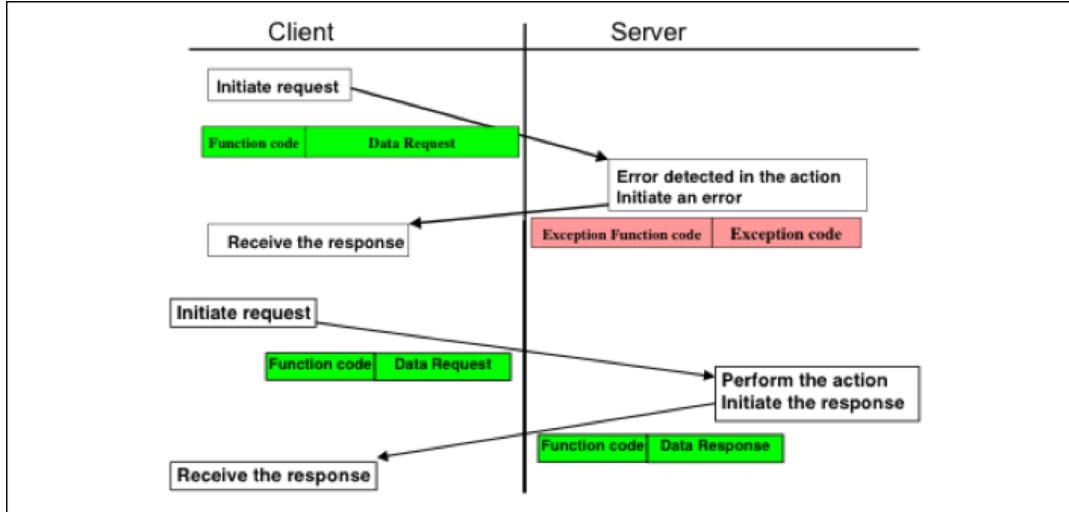


Abb. 2.13: Transaktion mit dem Modbus Protokoll nach [mod, 2012, S. 4]

Schema einer Transaktion läuft nach dem Prinzip in Abb. 2.13 ab. Wenn eine ADU fehlerfrei empfangen wurde nutzt der Server das Funktioncode Feld um anzugeben, ob seine Antwort ebenfalls eine normale, fehlerfrei Antwort, durch eine einfaches Echo des empfangenen Funktionscodes, ist oder ob ein irgendein Fehler aufgetreten ist durch einen Exception code. Die Größe der PDU ist grundsätzlich durch die serielle Kommunikation begrenzt auf 256 Bytes, da jedoch noch zwei Bytes für einen Cyclic Redundancy Check und ein Byte für die Server Adresse reserviert werden müssen ist die PDU auf 253 Bytes begrenzt. Ein weiterer, wichtiger Aspekt ist, dass Modbus die „big-Endian“ Codierung/Repräsentation für Daten verwendet verwendet, falls der numerische Wert größer als ein einzelner Byte ist [mod, 2012, S. 3ff.]. Anhand des Beispielder Uhrzeit kann die Bedeutung der Big-Endian Repräsentation einfach erläutert werden: Die Daten werden so aufgeteilt, dass zunächst die Daten mit der höchsten Wertigkeit, also den Stunden zuerst gesendet werden, anschließend den Minuten und zum Schluss die Sekunden, unabhängig von deren numerischem Wert. Also werden nacheinander die Code 03, 50, 12 empfangen bedeutet dies dass die Uhrzeit 03:50:12 ist. Der interessierte Leser wird für eine weitere Ausführungen auch zu little-Endian Darstellung in Bertrand Blanc [2005].

Das Modbus Datenmodell basiert darauf, dass auf die Daten in vier verschiedenen Tabellen mit unterschiedlichen Funktionen und Datenobjekten zugegriffen werden kann:

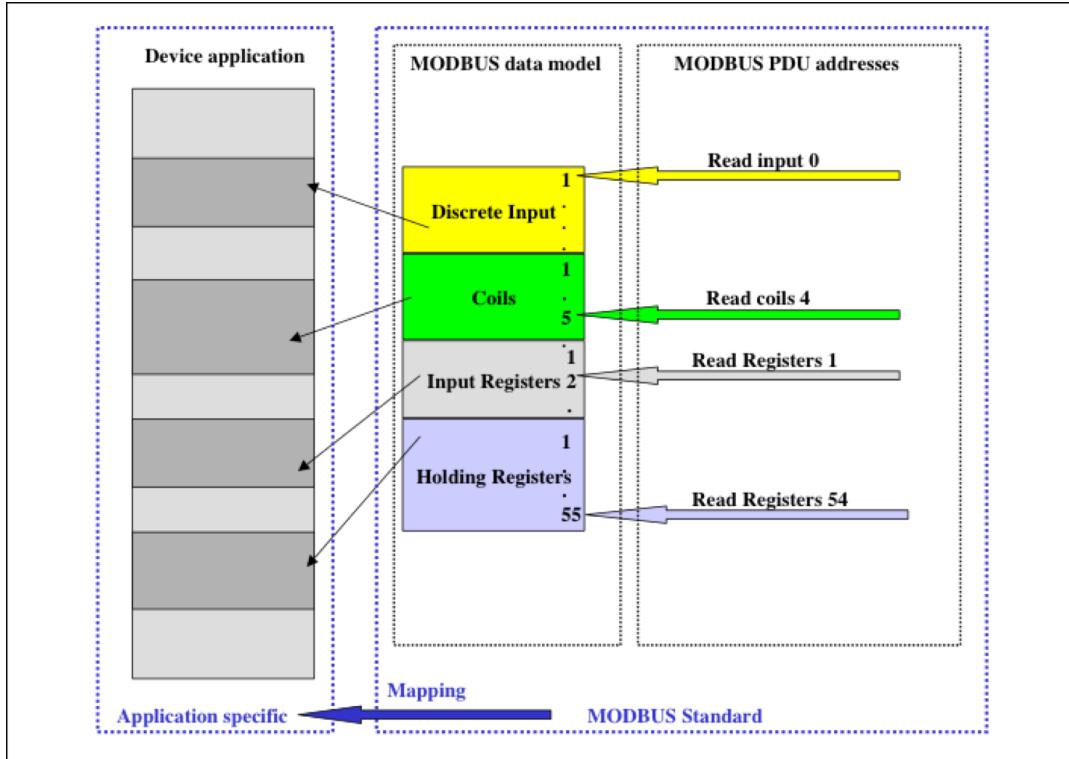


Abb. 2.14: Datenmodell und Adressierung nach dem Modbus Protokoll aus [mod, 2012, S. 8]

- In die Discretes Input, welche Single Bit Objekte enthält und lediglich gelesen werden kann,
- die Coils, welche ebenfalls Single Bit Objekte enthält jedoch gelesen und beschrieben werden darf,
- die Input Registers, die Datenobjekte als 16-Bit Wort enthält und wiederum nur gelesen werden kann
- und die Holding Registers, dessen 16-Bit Wort Objekte wiederum gelesen und beschrieben werden dürfen.

16 bit wort entspricht einer Folge von 16 bits/binärzeichen das also einen dezimalen Zahlenwert zwischen 0 und 65.536. Nach der IEC 61131-3 Norm entspricht es in iim Rechner einem Integer Wert.

Die Daten selbst können auch innerhalb einer Tabelle abgelegt sein, müssen lediglich über die vier angegebenen Tabellen zugreifbar sein, also die Referenz auf die Daten muss gegeben sein, der Rest ist egal. Jede Tabelle besitzt Daten adressiert zwischen 0 und 65535 und ist damit auch nach oben begrenzt was Daten angeht. Welche Daten wo genau stehen, also unter welcher Adresse kann von Gerät zu Gerät unterschiedlich sein und wird vom Gerätehersteller festgelegt. [mod, 2012, S. 6ff.]

Die Funktionscodes starten bei 1 und können bis 255 genutzt werden und sind wie bereits angesprochen in der PDU enthalten und definieren welche Aktion ein Server ausführen soll. In erster Linie dienen sie dem Datenzugriff in den Tabellen, können aber auch für Diagnosen oder nutzerdefinierte Aktionen benutzt werden. Daher lassen

sich die Funktionscodes in drei große Gruppen aufteilen, die öffentlichen, nutzerdefinierbaren und die reservierten Funktionscodes. Die öffentlichen sind wohldefiniert, unique, dokumentiert und sind auf Konformität getestet und sind daher einfach, schnell und sicher nutzbar. Die vom Nutzer definierbaren Funktionscodes können genutzt werden um von den öffentlich bereitgestellten nicht bereitgestellte Funktionen eigens zu definieren/nutzbar zu machen. Die reservierten Codes sind von wenigen Unternehmen, welche an der Entwicklung des Modbus Protokolls beteiligt waren, für deren hinterlassene Produkte reserviert und daher nicht öffentlich nutzbar[mod, 2012, S. 10ff.].

Als nächstes folgen die beiden verschiedenen Modbus Over Serial Line Protocol und Modbus Messaging On TCP/IP Protocol

Beim Modbus Over Serial Line Protocol können die Daten über zwei unterschiedliche Modi übertragen werden, der RTU und ASCII Modus. Näheres Der ASCII Modus ist optional und wird in mod [2006b] detailliert beschrieben. Der RTU Modus wird von allen modbusfähigen Komponenten unterstützt und spezifiziert das folgende Format zur Übertragung der einzelnen Bytes: Jede Byteübertragung beginnt mit einem Startbit, auf das zu übertragende Byte, bestehend aus acht einzelnen Bits, folgt, bevor die Übertragung optional von einem Paritätsbit und einem Stoppsbit beziehungsweise lediglich von zwei Stoppsbits abgeschlossen wird. Dabei wird jedes zu übertragende Byte als zwei 4-bit hexadezimales Zeichen übertragen[mod, 2006b, S. 12f.]. Die Paritätsprüfung ist optional und dient der Fehlerüberprüfung des Telegramms, wie bereits in Abschnitt 2.2 erläutert. Der Rahmen eines gesamten Modbus RTU Telegramms besteht aus der Slave Adresse, die für jeden Slave eindeutig ist und zwischen 1 und 247 liegt, und dem Function Code, die jeweils aus einem Byte bestehen. Darauf folgen die eigentlichen Informationen für die 0 bis 252 Bytes vorgesehen sind. Abgeschlossen wird der Rahmen durch ein CRC Feld, dass aus einem CRC Low und einem CRC High byte aufgebaut ist und dazu dient das Telegramm auf Fehler zu überprüfen. Der Ablauf und Vorgang des CRC Checks ist detailliert in mod [2006b] beschrieben. Die Übertragung eines Telegramms erfolgt byteweise, wie zuvor beschrieben. Die Datensicherung findet also durch Parität und CRC auf verschiedenen Ebenen statt. Die genaue Übertragungszeit eines Bytes und einer Nachricht hängt von der Baudrate ab. Um den Beginn und den Abschluss eines RTU Rahmens eindeutig zu definieren, geschieht dies in Abhängigkeit von der Übertragungsgeschwindigkeit. Zwischen einzelnen Bytes innerhalb eines Rahmens folgt ein stilles Intervall, dass je nach Länge angibt ob das Telegramm beendet ist. Auf eine Intervall kleiner gleich der anderthalbfachen Übertragung eines Bytes folgt eine weiteres Byte. Ist das stille Intervall länger als die dreieinhalfache Byteübertragungszeit markiert dies das Ende eines Telegramms und den Beginn eines nächsten Telegramms [mod, 2006b, S. 13]. Diese Zusammenhänge sind zur Veranschaulichung in Abb. 2.15 zusammengefasst.

Der Implementierungsleitfaden legt auch die Spezifikationen der physikalischen Schicht fest, die nun folgen. Er schlägt vor die EIA 438 Schnittstelle als elektrisches/physikalisches Interface zu verwenden, erlaubt aber auch weiterhin die Implementierung durch

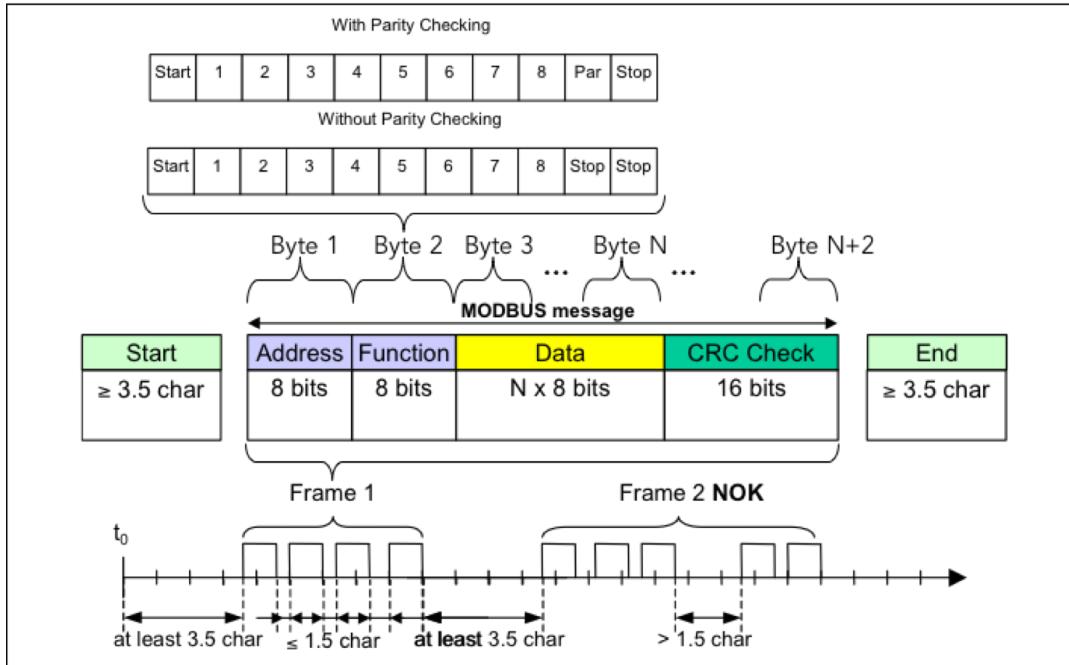


Abb. 2.15: Serielle Kommunikation über Modbus RTU nach [mod, 2006b, S. 12f.]

die EIA 232 Schnittstelle, beides über ein verdrilltes Leiterpaar. Weiterhin werden die Datenraten von 9.600 und 192.000 bps und eine Even Parität bei der Byteübertragung als Standard festgelegt. Die Standardverdrahtung der Komponenten erfolgt bei beiden elektrischen Standards über ein verdrilltes Leiterpaar und einer gemeinsamen Verbindungsleitung common. Die beiden Leitungen des verdrillten Paars werden mit D1, welche auch als D+ oder A Leitung bezeichnet wird, und D0, welche auch als D- oder B Leitung bezeichnet wird, bezeichnet. Ein Standard Netzwerk besteht aus maximal 32 Teilnehmern, dass durch den Einsatz von Repeatern auch vergrößert werden kann. Außerdem wird die Bus-Struktur als Topologie beschrieben, nach der die einzelnen Komponenten im Netzwerk angeordnet werden, unter der Voraussetzung, dass die Busleitung an beiden Enden durch einen Widerstand von 150 Ohm zwischen der D0 und D1 Leitung abgeschlossen werden [mod, 2006b, S. 20ff.]. Die Verbindung der Kabel kann im einfachsten Fall durch Schraubklemmen erfolgen, jedoch können auch genormte mechanische Interfaces, also Standard Steckverbindungen genutzt werden, deren Einsatz und Verkabelung/Anschlüsse in [mod, 2006b, S. 29ff.] detailliert beschrieben sind.

XXXXXXXX

Beim Modbus Messaging on TCP/IP Protocol stellt die Möglichkeit zur Verfügung, Geräte, die über ein Ethernet miteinander verbunden sind, über ein Client/Server Modell kommunizieren zu lassen. Des Weiteren erlaubt es dieses Protokoll explizit über Bridges, Gateways oder Router Netzwerke miteinander zu verbinden. Dabei dürfen auch serielle Subnetzwerke zu verbinden und erlaubt auch zwischen diesen Endgeräten die Kommunikation [mod, 2006a, S. 2f.]. Diese Kommunikationsarchitektur ist auf in Abb. 2.16 dargestellt.

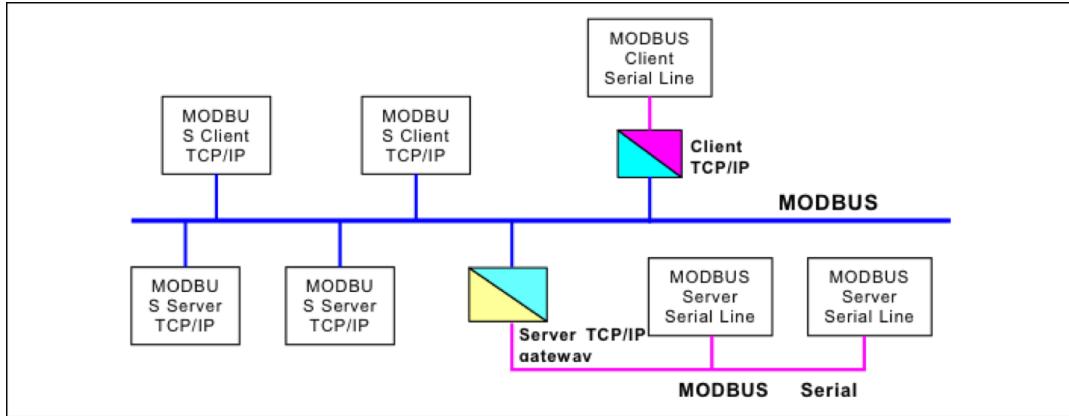


Abb. 2.16: Die Modbus TCP/IP Kommunikationsarchitektur aus [mod, 2006a, S. 4]

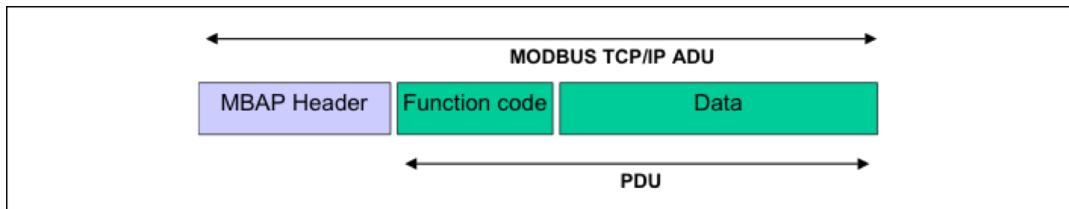


Abb. 2.17: Angepasster Rahmen für Telegramme nach dem Modbus TCP/IP Protokoll aus [mod, 2006a, S. 4]

Außerdem ist eine leicht verschiedene ADU vorhanden, wie in Abbildung REF ADU TCP zu sehen. Modbus Application Protocol Header MBAP Header ist 7 bytes lang und enthält unit identifier ähnlich slave adress/id, adresse für modbus routing, ein bytecount, der die Länge des folgenden Telegramms angibt (inklusive Unit identifier und Daten) auch wenn gesplittet, CRC-32 error check, protocol identifier mit modbus 0, transaction identifier vom client, der nur kopiert wird vom server um Transaktionen einander zuzuordnen. Alle Kommunikation erfolgt über TCP Port 502 [mod, 2006a, S. 4f.]

Alle Modbus /TCP ADU werden via TCP zum Modbus registrierten Port 502 geschickt.

Modbus TCP Komponenten können sowohl Client als auch Server Interface haben. [mod, 2006a, S. 7f.]

Der Modbus Client erlaubt den Informationsaustausch indem er eine ADU erstellt
Der Modbus Server wartet auf Anfragen über den TCP Port 502

Transmission control protocol TCP und IP ist Internet Protocol. TCP als Transport-schicht verbindungsorientiert: Teilt Daten in Datenblöcke, sogenannte Pakete, zum Transport. IP regelt Netzwerkaufgaben, siehe OSI Modell Netzwerkschicht, und versendet die Daten über Telegrammservice und packt den MBAP Header an jedes Paket dran dran. Der Port erlaubt die parallele Nutzung von Ethernet Netzwerken, da er den Übertragungsprozess eindeutig kennzeichnet. Verbindungsorientiert heißt über ein Socket werden zwei Prozesse miteinander verbunden und die empfangenen Telegrammen werden/müssen quittiert [Schnell u. Wiedemann, 2006, S. 16ff.]. Der interessierte

Leser findet eine detaillierte Beschreibung des Ethernet TCP/IP Standards in Furrer [2003].

TCP übernimmt Netzwerkmanagement - TCP Management: Hauptaufgabe ist das connection management. Das managen von Verbindungen kann entweder durch ein Modul erfolgen oder durch die Nutzeranwendung selbst durch die Zugriffsüberwachung der sockets. Der Port 502 ist für Modbus Kommunikation reserviert, jedoch können auch andere Ports genutzt werden falls die Modbusgeräte eine Portkonfiguration unterstützen. Weiterhin wird der Datenfluss und der Einsatz der Netzwerkressourcen überwacht. [mod, 2006a, S. 7ff.]

Generell, Verbindungsmanagement wichtig, da Modbus/TCP Kommunikation zwischen einem Server und einem Clienten eine Verbindung benötigt. Hinweise gibt der Guide, das die Verbindungen nicht dauernd geöffnet und geschlossen werden und auch mehrere viele Modbus Transaktionen während der Verbindung stattfinden. Außerdem sollte sich auf ein Minimum von Verbindungen beschränkt werden für den gleichen Server. [mod, 2006a, S. 9f.] Das Nutzer TCP Management umfasst folgende Aufgaben, die aktive und passive Herstellung von Verbindungen sowie das schließen dieser und das festlegen von maximalen Verbindungen [mod, 2006a, S. 11ff.]. Verbindungsherstellung über Ethernet IP, also der eindeutigen Adresse, des Geräts und Portnummer und die Socket Nummer. Der Socket ist ein Endpunkt innerhalb eines Rechners, über den die Kommunikation läuft und der einem Port eindeutig zugewiesen ist [mod, 2006a, S. 15f.] .

Physikalisch bedient sich der Ethernet Schnittstelle also einem normalen Netzwerkabel.

Diese Kommunikationstechnologie und die verschiedenen Protokolle finden im Rahmen der Anlage in Kapitel dann Anwendung

2.3 Technische Grundlagen zur Modellbildung

In diesem Kapitel werden die technischen Grundlagen zur Bildung eines mathematischen Modells des Raumes erläutert. Themrodym systeme 1. HS thermo Wärmeübertragung

2.3.1 Thermodynamische Systeme

Im Raummodell müssen Energieströme, genauer betrachtet Wärmeströme, untersucht werden. Um dieses thermodynamischen Vorgänge mit Hilfe von Bilanzierungsgleichungen zu beschreiben, folgt zunächst ein kurze Einführung in die Thermodynamische Systembildung nach [Baehr u. Kabelac, 2012, S. 11ff.].

Thermodynamische Systeme werden durch den zu untersuchenden Raum abgegrenzt. Sie dienen dem Zweck der Bilanzierung von Massen- und Energieströmen und alles

was diesen abgegrenzten Raum an den Systemgrenzen umgibt wird als Umgebung bezeichnet. Die begrenzenden Flächen können gedanklicher, physischer oder beider Natur zugleich sein, wichtig ist jedoch das die Systemgrenzen eindeutig festgelegt sind [Baehr u. Kabelac, 2012, S. 11].

Anhand der Eigenschaften von den Systemgrenzen lassen sich die thermodynamischen Systeme weiter differenzieren. Solche Systeme, deren Grenzen undurchlässig für Materie sind, werden als *geschlossene Systeme* bezeichnet und werden durch eine konstante Stoffmenge innerhalb des Systems gekennzeichnet. Die Grenzen eines geschlossenen Systems sind meistens räumlich anhand eines fixen Volumens definiert, können aber auch beweglich sein, wie z.B. das Volumen einer vorgegebenen Stoffmenge unabhängig von dessen räumlicher Ausdehnung [Baehr u. Kabelac, 2012, S. 12].

Sind die Grenzen von thermodynamischen Systemen für Materie durchlässig, werden diese als *offene Systeme* bezeichnet. In der Regel werden diese von Stoffströmen durchflossen und durch räumlich festgelegte Grenzen beschrieben. Diese werden in der Literatur auch als *Kontrollraum* oder *Kontrollvolumen* bezeichnet [Baehr u. Kabelac, 2012, S. 12].

Ein *abgeschlossenes System* umfasst in der Regel mehrere Systeme oder ein einzelnes System und dessen Umgebung, so dass es zwischen den Grenzen des abgeschlossenen Systems und seiner Umgebung keine Wechselwirkungen gibt. Die Systemgrenzen werden also so gelegt, dass über sie hinweg keine beziehungsweise keine relevanten⁴ Flüsse von Materie und Energie [Baehr u. Kabelac, 2012, S. 13].

Nach der Abgrenzung folgt die *Beschreibung* von thermodynamischen Systemen und dessen *Eigenschaften*. Diese erfolgt durch *Variablen* und *physikalische Größen* die ein System kennzeichnen. Falls die Variablen feste Werte annehmen werden diese als *Zustandsgrößen* bezeichnet, da sie den *Zustand* eines Systems bestimmen [Baehr u. Kabelac, 2012, S. 13]. Im Rahmen der Modellbildung in Kapitel 4 ist es ausreichend die Vorgänge und Effekte auf systemischer Ebene zu betrachten, wodurch sich Modelle mit wenigen Variablen und physikalischen Größen beschreiben lassen.

Die Variablen lassen sich in *äußere Größen*, welche den mechanischen Zustand eines Systems beschreiben⁵, und *innere Größen* gliedern, welche den thermodynamischen Zustand, also die Eigenschaften der Materie innerhalb der Systemgrenzen, beschreiben [Baehr u. Kabelac, 2012, S.13 f.].

Innerhalb der Grenzen eines thermodynamischen Systems, und damit implizit auch für das Raummodell⁶ wird *Homogenität* angenommen. Dies bedeutet, dass die physikalischen Eigenschaften, wie zum Beispiel Temperatur und Druck, sowie die chemische Zusammensetzung an jeder Stelle innerhalb des Systems homogen ist, also die gleiche Ausprägung besitzt [Baehr u. Kabelac, 2012, S.15].

Da wir im Rahmen der Modellbildung Zustände betrachten müssen auch deren

⁴ Relevant im Sinne von kaum messbarer Fluss und nicht messbare Auswirkung auf das System.

⁵ Zum Beispiel die Koordinaten im Raum oder die relative Geschwindigkeit zum Beobachter)

⁶ Diese Annahme wird im Kapitel 5 noch überprüft und kritisch hinterfragt werden müssen

Änderungen genauer untersucht werden. Die *Zustandsänderungen* eines Systems werden durch Änderungen von Energie oder Materie über dessen Grenzen hinweg bedingt und finden meist im Austausch der Umgebung statt. Während einer solchen Änderung des Systemzustands wird ein Prozess durchlaufen, der eine zeitliche Abfolge von Ereignissen ist. Eine Änderung des Zustands eines Systems mit der gleichen Wirkung kann also durch verschiedene Prozesse bewirkt werden. Daher beschreibt ein *Prozess* nicht nur die Veränderung des Zustands sondern viel mehr die Beziehungen zwischen einem System und seiner Umgebung [Baehr u. Kabelac, 2012, S.21 f.].

Ein Prozess kann aber auch innerhalb eines Systems stattfinden, dass heißt ohne äußere Einwirkungen. Dies geschieht zum durch das Aufheben innerer Hemmungen oder dem Wegfall Zwängen von Außen. Diese Prozesse laufen in abgeschlossenen Systemen meist von selbst ab und streben als Ziel einen ausgeglichenen, also homogenen, Endzustand an. *Ausgleichsprozesse* dienen somit dazu, einen *Gleichgewichtszustand* zu erreichen und repräsentieren Wechselwirkungen zwischen verschiedenen Teilen eines abgeschlossenen Systems. Dabei gleichen sich die Zustandsgrößen von einzelnen Subsystemen wie zum Beispiel der Druck oder die Temperatur einander an. Der Gleichgewichtszustand wird also durch die Zustände in den einzelnen Subsystemen bestimmt und ist dadurch charakterisiert, dass ein System diesen Zustand nicht von sich aus sondern nur durch äußere Eingriffe verlässt, zum Beispiel durch eine Veränderungen in der Umgebung. Die Erfahrung lehrt, dass ein System einem Gleichgewichtszustand entgegen strebt, wenn es sich selbst überlassen wird [Baehr u. Kabelac, 2012, S.22 f.]. Im Rahmen der Modellbildung in Kapitel 4 nehmen diese *Ausgleichsprozesse* eine zentrale Rolle ein, weil der Großteil an Änderungen von einzelnen Zustandsgrößen innerhalb des Raumes darauf zurückgeführt werden können.

2.3.2 Erster Hauptsatz der Thermodynamik

Der erste Hauptsatz der Thermodynamik wird im Folgenden als allgemeiner Energieerhaltungssatz formuliert und anschließend angewendet um eine Energiebilanzgleichung für geschlossene thermodynamische Systeme zu erhalten.

Der erste Hauptsatz der Thermodynamik erweitert den mechanischen Energieerhaltungssatz um die Energieformen Wärme und innere Energie. Er handelt ganz allgemein vom Prinzip der Energieerhaltung und dient er der Bilanzierung von Systemen [Baehr u. Kabelac, 2012, S. 43].

Die Gesamtenergie eines Systems E setzt sich zusammen aus der potenziellen E_{pot} und kinetischen Energie E_{kin} wie in der Mechanik und wird durch die innere Energie U ergänzt [Baehr u. Kabelac, 2012, S. 49]:

$$E := E_{pot} + E_{kin} + U \quad (\text{Gl. 2})$$

Im weiteren Verlauf der Arbeit werden nur ortsfeste Systeme betrachtet die sich da-

durch auszeichnen, dass deren potenzielle Energie E_{pot} in etwa konstant ist. Weiterhin erfahren sie im betrachteten Intertialsystem Erde auch nur sehr kleine Änderungen in ihrer Geschwindigkeit, weshalb auch die kinetische Energie E_{kin} in etwa konstant ist. Da die Änderungen der mechanischen Energien in Bezug auf die Änderung der inneren Energie sehr klein sind werden im Folgenden nicht weiter betrachtet und die Gesamtenergie eines Systems E vereinfacht und lediglich aus der inneren Energie bestehend angenommen.

Die innere Energie hängt von der spezifischen Wärmekapazität c_p , der Masse eines Systems m_{sys} und der Temperatur t beziehungsweise T ab [Baehr u. Kabelac, 2012, S. 54]:

$$U := m * c_p * T = m * c_p * t + u_0, \text{ mit } t = T - T_0 \quad (\text{Gl. 3})$$

Nach dem Prinzip der Energieerhaltung, kann die Energie eines Systems also weder erzeugt noch vernichtet werden sondern lediglich durch den Energietransport über dessen Grenzen hinweg verändert werden. Daraus ergeben sich folgende qualitative Formen des Energietransports [Baehr u. Kabelac, 2012, S. 48f.]:

- Die Arbeit W , die entweder von oder an einem System verrichtet wird, in differentieller Form die Leistung P .
- Die Wärme Q , die entweder in das System hinein- oder herausfließt, in differentieller Form der Wärmestrom \dot{Q} .
- Der Transport von Materie, also das Einbringen oder Wegnehmen von Masse m eines Systems, in differentieller Form die Materialflüsse \dot{m} .

Mit der zuvor getroffenen Annahme, dass die innere Energie des Systems entspricht, und unter Beachtung der Vorzeichenkonvention, welche besagt dass zugeführte Energie positiv und abgeführte Energie negativ zu bewerten ist, lassen sich die Änderungen der Energie eines Systems mit der folgenden Gleichung quantitativ beschreiben [Baehr u. Kabelac, 2012, S. 54]:

$$\Delta U = Q + W + m_{in} * c_p * T_{in} - m_{out} * c_p * T_{out} \\ \text{beziehungsweise in differentieller Form} \quad (\text{Gl. 4})$$

$$\frac{dU}{dt} = \dot{U} = \dot{Q} + P + \sum \dot{m}_{in} * c_p * T_{in} - \sum \dot{m}_{out} * c_p * T_{out}$$

2.3.3 Wärmeübertragung

Wärmeströme spielen bei der Modellbildung in Kapitel 4 eine wichtige Rolle, daher ist eine genauere Betrachtung dieser unumgänglich und im Folgenden werden die Grundlagen dazu erläutert.

Die Definition von Wärmeübertragung ist nach [Böckh u. Wetzel, 2014, S. 1] „[...] der

Transfer der Energieform Wärme aufgrund einer Temperaturdifferenz. „Die Definition umfasst also einen zuvor beschriebenen Ausgleichsprozess und eine Änderung der inneren Energie eines thermodynamischen Systems. Die Wärmeübertragung kann nach Nußelt⁷ grundsätzlich durch zwei verschiedene Arten stattfinden [Böckh u. Wetzel, 2014, S. 3f.]:

- Durch Strahlung, bei der die Übertragung von Wärme ohne stofflichen Träger durch elektromagnetische Wellen zwischen Oberflächen erfolgt. Weil diese Art der Wärmeübertragung keine Relevanz für die weiteren Betrachtungen hat wird er interessierte Leser an dieser auf Böckh u. Wetzel [2014] verwiesen der diese Thematik detailliert ausführt.
- Durch Wärmeleitung, die sich wiederum in die Wärmeübertragung zwischen ruhenden Stoffen, und die Konvektion, die eine Wärmeübertragung zwischen einem ruhenden und einem strömenden Fluid beschreibt, aufteilen lässt.

Die übertragene Wärmemenge ist bei der reinen Wärmeleitung lediglich von den Stoffeigenschaften und der Temperaturdifferenz abhängig, bei der Konvektion hingegen, unabhängig davon ob erzwungen oder frei, hängt sie von der Strömung der Fluide ab. Die Konvektion ist ein Effekt zusätzlich zur reinen Wärmeleitung auftritt und ist im weiteren Verlauf der Arbeit nicht relevant und wird deshalb nicht detaillierter ausgeführt [Böckh u. Wetzel, 2014, S. 3f.]. Erfolgt der Wärmetransport stationär, dass heißt er ist von äußeren Anregungen bedingt und unabhängig von der Zeit, lässt er sich qualitativ einfach als konstanter Wärmestrom \dot{Q} beschreiben und gibt an wie viel Wärme pro Sekunde übertragen wird [Böckh u. Wetzel, 2014, S. 5ff.]. Der Wärmestrom ist wie zuvor bereits erwähnt von den Stoffeigenschaften abhängig, welche von der Wärmedurchgangszahl U – Wert⁸ und der Austauschoberfläche $A_{exchange}$, an der der Wärmeaustausch stattfindet. Typische U-Werte für verschiedene Materialien und Komponenten finden in der einschlägigen Literatur und beziehen sich bei der Übertragung durch eine Wand im europäischen Raum auf die Außenfläche [Böckh u. Wetzel, 2014, S. 28]. Damit lässt sich der Wärmestrom unter Berücksichtigung der Abhängigkeiten durch die kinetische Kopplungsgleichung quantifizieren [Böckh u. Wetzel, 2014, S. 6f.]:

$$\dot{Q} := u * A * (t_1 - t_2) \quad (\text{Gl. 5})$$

Unterschiedliche geometrische Ausprägungen, wie zum Beispiel ein Wärmeaustausch durch eine Wand oder ein Rohr hindurch, finden damit implizit bei der Austauschoberfläche Berücksichtigung.

⁷ Beschrieben in seinem Aufsatz „Das Grundgesetz des Wärmeüberganges“, 1915.

⁸ Der U-Wert wurde bis zu der Umstellung auf die europäischen Prüfnormen 2003 als k-Wert bezeichnet und ist unter dieser Bezeichnung noch häufig in der Literatur zu finden [Sack, 2004, S.1 f.]

2.4 Technische Grundlagen zur Solar- und Gebäudetechnik

2.4.1 Außenklima und Komponenten

Der Begriff Außenklima wird häufig im Zusammenhang mit dem Thema Umwelt und deren Einflüsse auf Gebäude gebraucht. Der allgemeine Begriff des Klimas wird von [Peter Häupl, 2013, S. 295] definiert als:

„die Summe aller Umweltfaktoren, die unmittelbar oder mittelbar Einfluss nehmen auf die Gesundheit und das Befinden von Menschen und Tieren, auf die Entwicklung von Pflanzen sowie auf den Zustand von Lagergütern, Produktionsverfahren, Maschinen, Apparaten und Bauwerken.“

Daraus lässt sich ableiten, dass das Außenklima ein Aspekt des Klimas ist und den meteorologischen Umweltzustand außerhalb von Gebäuden, an einem bestimmten, lokalen Ort meint. Abhängig vom Außenklima stellt sich innerhalb von Gebäuden ein Innenklima ein, welches direkten Einfluss auf das Wohlbefinden von Menschen hat und wodurch der mittelbare Einfluss des Außenklimas gegeben ist. Um den Zustand zu beschreiben werden viele Zustandsgrößen herangezogen. Um einen Überblick zu bekommen, lassen sich diese in verschiedene Bereiche gliedern [Peter Häupl, 2013, S. 295f.] : Schall Licht Temperatur und Feuchte

Im Hinblick auf die Modellbildung in Kapitel 4 sind lediglich die Größen zur Beschreibung der Temperatur und des Lichts von Interesse, eine detaillierte Ausführung in die Bereiche Schall und Feuchte und deren Zustandsgrößen ist in Peter Häupl [2013] gegeben.

Je nach Größe des Gebietes wird von einem Regional- bzwziehungsweise Makroklima, das große Gebiete umfasst, oder von einem Lokal- beziehungsweise Mikroklima gesprochen, dass kleine Gebiete wie eine Straße oder einen Park umfasst und von deren Besonderheiten abhängig ist. So kann z.B. die Außentemperatur je nach Verschattungsgrad einer Straße lokal erhöht oder erniedrigt sein. Das Klima folgt in verschiedenen Regionen der Erde bestimmten Charakteristiken, welche sich in Klimazonen zusammenfassen lassen. Die Erde besteht aus vierzehn verschiedenen Klimazonen und in Europa wird von einem Übergangsklima gesprochen [Peter Häupl, 2013, S. 296f.].

Eine Übersicht der Außenklimakomponenten, die einen Einfluss auf die Gebäude-technik und damit auch auf die Raumtemperatur haben ist in Abb. 2.18 gegeben. Für die Modellbildung ist weiterhin eine Quantifizierung der relevanten Größen des Außenklimas in den Bereichen Licht und Temperatur erforderlich. Wie bereits im vorherigen Abschnitt erwähnt, werden Wärmeströme durch Temperaturdifferenzen bedingt, weshalb die Außenlufttemperatur *HierSymbol* einen großen Einfluss auf die Raumtemperatur hat und durch Messung quantifiziert werden muss. Des Weiteren werden die lichttechnischen Größen der kurzwelligen direkten und diffusen Strahlung durch die beiden Strahlungsintensitäten G_{dif} und G_{dir} beschrieben, da sie Energie durch einen Wärmestrom in ein System einbringen und damit auch einen Einfluss

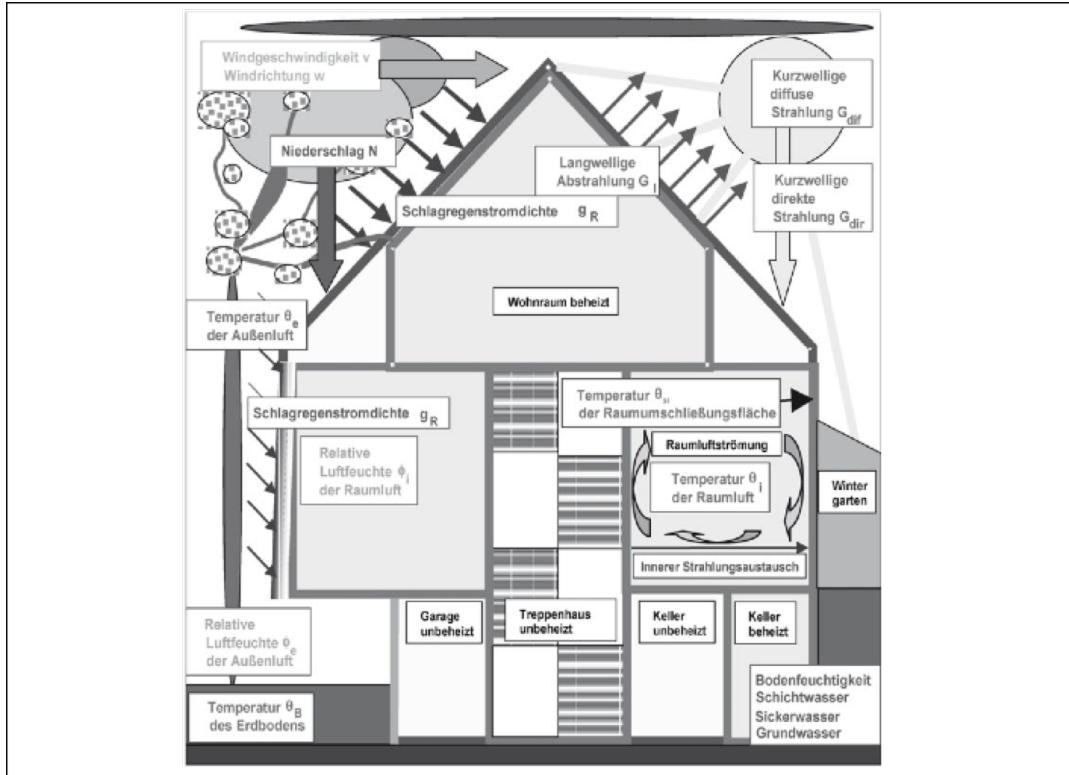


Abb. 2.18: Komponenten des Außenklimas aus [Peter Häupl, 2013, S. 298]

ausüben.

2.4.1.1 Die Außenlufttemperatur

Eine Übersicht über die Außenlufttemperatur

2.4.1.2 Sonnenstrahlung

Gemessen wird immer Globalstarhlung.

Nicolai [2013] [Quaschning, 2011, S. 63ff.] [Kaltschmitt, 2013, S. 61ff.] [Peter Häupl, 2013, S. 315ff.] Bild

Algorithmus nach Reda [2008]

Nutzung Berechnung/Implementierung von pys

2.4.2 Gebäudetechnik Glas und Wärmedurchgangskoeffizienten

Auf gehts

Glas nach [Peter Häupl, 2013, S. 61ff.] Durchlassgrad nach [Peter Häupl, 2013, S. 604ff.] Transmissionsgrad

„Design is the appropriate combination of materials in
order to solve a problem.“
— CHARLES EAMES

3 Anlagendesign

Ziel dieses Kapitel ist es, eine Anlage zur Raumtemperaturregelung für den Betrieb mit Modellprädiktiver Regelung zu konzipieren, zu konkretisieren und im letzten Schritt umzusetzen. Dazu werden zunächst die Anforderungen an die Anlage analysiert und weiterhin die Vorgaben und Rahmenbedingungen zur Anlage von Seiten der Hochschule Karlsruhe spezifiziert und ausgeführt. Daraus wird eine Idee abgeleitet, die anschließend zu einem Konzept weiterentwickelt und in ein konkretes Anlagendesign umgesetzt wird. Dabei werden die einzelnen Anlagenteile und deren Funktionsweisen näher beschrieben und auf die realen Einsatzbedingungen ausgelegt. Abschließend wird die Installation und dabei aufgetretenen Besonderheiten der Anlage beschrieben.

Die These, welche es in diesem Kapitel zu belegen gilt, ist, dass sich die beschriebene und umgesetzte Anlage dazu eignet, um die Temperatur innerhalb eines zu regeln. Im nächsten Schritt gilt es zu zeigen, dass sich die Anlage gemeinsam mit dem Einsatz eines in Kapitel 4 gebildeten Modells, für die Modellprädiktive Regelung eignet.

3.1 Analyse der Anforderungen

3.1.1 Einsatzziele und Rahmenbedingungen

Um die Anforderungen an eine Anlage zu bestimmen, die sich für die Anwendung mit Modellprädiktiver Regelung eignet, wird zunächst der Zweck und die Einsatzziele der Anlage untersucht und definiert. In Kapitel 1.1 wurde bereits darauf hingewiesen, dass es die Vorgabe von Seiten der Hochschule ist, die Einsatzziele in Einklang mit der großen Anlgae zu bringen und komplementär zu wählen. Daher wurden im Dialog mit den Projektverantwortlichen⁹ für die Forschung im Bereich solarer Anwendungen an der Hochschule Karlsruhe gemeinsam konkrete Einsatzziele der Anlage erarbeitet. Als Ergebnis wurden die folgenden, konkreten Ziele vereinbart:

- Die Einarbeitung in die Themen Modellbildung, Kommunikation von technischen Systemen und Modellprädiktive Regelung soll durch eine praktisches Anwendung unterstützt werden.
- Es soll Know-how im Bereich der Kommunikation von technischen Systemen aufgebaut werden, insbesondere im Umgang mit der Software, der Hardware und zahlreichen Schnittstellen.
- Die Anlage soll eine hohe Funktionalität, also möglichst wartungsarm, und eine

⁹ In Person von Herrn ADRIAN BÜRGER und MARKUS BOHLAYER

hohe Robustheit gegenüber Fehlern und Beschädigungen besitzen, da bei der Einarbeitung eine erhöhte Wahrscheinlichkeit der Fehlbedienung besteht und Schäden dadurch vermieden werden sollen.

- Es soll ein Vergleich verschiedener Regelungsmethodiken beim Einsatz von Modellprädiktiver Regelung ermöglicht werden.
- Außerdem soll ein Vergleich von Ergebnissen bei der Variation von Steuerungsparametern sowie beim Einsatz verschiedener Steuerungs- und Regelungsalgorithmen ermöglicht werden.
- Des Weiteren soll die Anlage möglichst flexibel ansteuerbar und erweiterbar sein, damit der Grad der Komplexität anpassbar ist und die Anlage um weitere Funktionen oder Features ergänzt werden kann.
- Der temperaturerhöhende Effekt der Sonneneinstrahlung auf die Raumtemperatur soll untersucht werden können.
- Im Rahmen der Anwendungsforschung soll der Raum zur Temperaturregelung möglichst nahe an der Realität sein, also Störgrößen beinhalten und nicht ungenutzt beziehungsweise leerstehend sein.

Zusammenfassend wurde festgehalten, dass die Anlage als Forschungsumgebung für Entwicklungs-, Test- und Anwendungszwecke von verschiedenen Steuerungen und Regelungen dienen soll.

Weiterhin wurden von Seiten der Hochschule Karlsruhe¹⁰ Rahmenbedingungen definiert, die im Folgenden zusammengefasst sind:

- Der Raum K004a im K Gebäude der Hochschule Karlsruhe wird zur Installation der Anlage und Einrichtung der Forschungsumgebung zur Verfügung gestellt.
- Die Installation der Anlage muss mit minimalem baulichem und finanziellem Aufwand zu realisieren sein.
- Für die Kommunikation innerhalb der Anlage soll die Modbus Kommunikationstechnologie mit mindestens zwei verschiedenen Übertragungsprotokollen genutzt werden.
- Die Modellprädiktive Regelung soll mit Hilfe der Plattform JModelica.org erfolgen.

3.1.2 Definition der Anforderungen

Diese Einsatzziele und Rahmenbedingungen definieren implizit Anforderungen an eine Anlage, welche im Nachfolgenden explizit ausgeführt werden und aus Gründen der Übersichtlichkeit die wichtigsten in Tabelle Tab. 3.1 zusammengefasst sind.

¹⁰ In Person von Frau Professor ANGELIKA ALTMANN-DIESES, Herrn Professor MARCO BRAUN und Herrn ADRIAN BÜRGER

Einsatzziele & Rahmenbedingungen	Anforderungen
Raum K004a als Umgebung	<ul style="list-style-type: none"> – Die Anpassung der Anlage an K004a.
Minimaler baulicher Aufwand	<ul style="list-style-type: none"> – Die Nutzung bestehender Heizkörper anstatt einer Klimatisierung des Raumes.
Minimaler finanzieller Aufwand	<ul style="list-style-type: none"> – Die Beschränkung auf eine minimale Funktionalität und Anzahl der einzelnen Komponenten.
Modellprädiktive Regelung mit JModelica.org und CasADi	<ul style="list-style-type: none"> – Die Modellbildung erfolgt in Modelica. – Die Ansteuerung und Kommunikation innerhalb der Anlage soll in Python stattfinden.
Einsatz der Modbus Kommunikationstechnologie	<ul style="list-style-type: none"> – Die Kommunikation der Anlage erfolgt gemäß den Modbus RTU und TCP Protokollspezifikationen. – über Modbus TCP soll die Ansteuerung der Anlage innerhalb des gesamten lokalen Netzwerks möglich sein.
Flexible Ansteuerung der Anlage	
Einarbeitung in die Thematiken:	
<ul style="list-style-type: none"> – Modellbildung, – Kommunikation technischer Systeme, – und Modellprädiktive Regelung. 	<ul style="list-style-type: none"> – Komplexität ist notwendig, darf jedoch nicht zu hoch sein. – Es sind möglichst wenige thematische überschneidungen erwünscht, daher wird eine klare Struktur mit möglichst scharfer Trennung benötigt.
Know-how für Kommunikation technischer Systeme	
Vergleich von Ergebnissen durch:	
<ul style="list-style-type: none"> – die Variation von Steuerungsparametern, – den Einsatz verschiedener Regelungsmethodiken, – und den Einsatz verschiedener Algorithmen. 	<ul style="list-style-type: none"> – Die Reaktion des Systems muss schnell sowie günstig und einfach zu erfassen sein. – Der Einsatz von robusten und einfachen Bauteile. – Nutzung eines wartungsarmen Systems. – Eine einfache, modulare Erweiterbarkeit des Systems für weitere Schritte muss gegeben sein.
Hohe Funktionalität und Robustheit	
Erweiterbarkeit der Anlage	

Tab. 3.1: Umsetzung der Ziele in Anforderungen der Anlage

Die grundlegendste Anforderung an die Planung, ist die Anpassung an die Lage und die Gegebenheiten des Raumes K004a, welche in Abb. 3.1 skizziert sind. Der Raum befindet auf dem Campus der Hochschule Karlsruhe, an der südwestlichen Ecke des K Gebäudes im Erdgeschoss. Die südliche und westliche Wand teilt sich der Raum mit der Außenumgebung und wird im Folgenden als Außenwand bezeichnet. Die östliche und nördliche Wand sowie die Decke und der Boden des Raumes grenzen an andere Räume im K Gebäude. Außerdem ist in der südlichen Außenwand eine Fensterfront mit

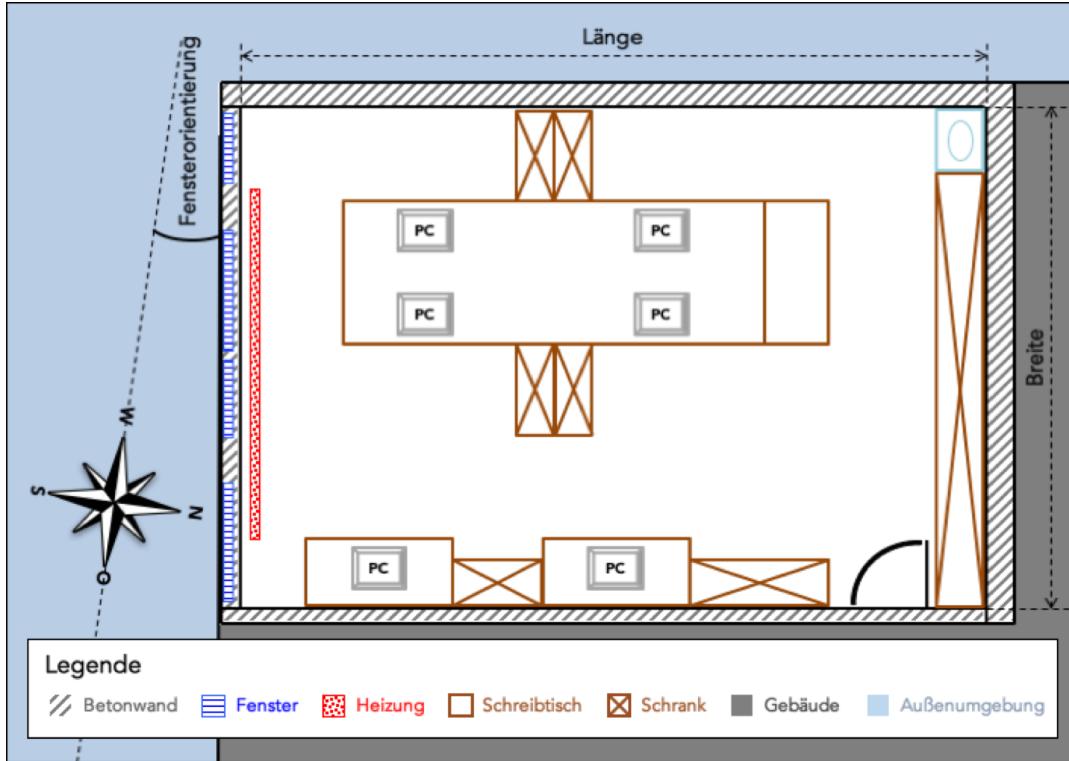


Abb. 3.1: Raumskizze K004A vom K Gebäude der Hochschule Karlsruhe – Technik und Wirtschaft

Jalousien zur Verschattung eingebaut und direkt darunter ein Heizkörper installiert.

Durch die Raumwahl werden bereits zwei wichtige Anforderungen erfüllt, denn durch die Fensterfront kann der Einfluss der Sonneneinstrahlung auf die Raumtemperatur untersucht werden und der bestehende Heizkörper kann in die Anlage integriert werden, um einen minimalen baulichen Aufwand sicherzustellen. Damit wird zunächst eine Klimatisierung zur Temperaturregelung des Raumes ausgeschlossen, weil diese mit einem erheblichen finanziellen und baulichen Aufwand verbunden ist. Um der Forderung nach einer Erweiterbarkeit nachzukommen, wird jedoch die Möglichkeit der Nachrüstung einer Klimatisierung und der Ansteuerung der Jalousien bei der Planung explizit berücksichtigt.

Durch die Nutzung des Raumes als Büro für wissenschaftliche Mitarbeiter befinden sich in K004a sechs Computerarbeitsplätze sowie das typische Büro-Mobiliar, wie in Abb. 3.1 abgebildet. Damit wird auch die Anforderung einer anwendungsnahen Umgebung erfüllt und durch die Menschen und Rechner sind verschiedene Störgrößen zu berücksichtigen.

Eine weitere, sehr einschränkende Vorgabe ist es, dass die Modellprädiktive Regelung unter Zuhilfenahme der Plattform JMODELICA.ORG erfolgen soll. JMODELICA.ORG ist eine kostenlose Open-Source Plattform zur Analyse, Simulation und Optimierung von komplexen, dynamischen Systemen, die auf der Modellierungssprache Modelica basiert. Aufbauend auf den mathematischen Modellen physikalischer Systeme in Modelica, lassen sich, dank der Unterstützung der Spracherweiterung Optimica, Optimierungsprobleme durch einfache Konstrukte das Optimierungsintervall, die Kostenfunktion

und die Nebenbedingungen einfach formulieren. JModelica.org wird über eine Python Nutzerschnittstelle genutzt und besitzt eine eigene Klasse für die Modellprädiktive Regelung, auch wenn diese bisher noch experimenteller Art ist [AB, 2015, S. 1f.]. Der Compiler kann die Modelle und Optimierungsprobleme in verschiedene Formate übersetzen. Zum einen in direkt ausführbaren C Code, der die Modellgleichungen und Optimierungsparameter enthält, und ein XML Code, der die Meta-Daten des Modells enthält. Zum anderen kann das Modell aber auch in ein *OptimizationProblem* Objekt transferiert werden, welches eine symbolische Repräsentation des Optimierungsproblems ist [AB, 2015, S. 12ff.].

Die *OptimizationProblem* Objekte können anschließend direkt mit den Optimierungs werkzeugen von CASADI bearbeitet und damit zur Lösung des Optimierungsproblems eingesetzt werden. CASADI ist ein Open-Source Softwaretool, das einzelne Bausteine für die numerische Optimierung im Allgemeinen und für die Optimalsteuerung im Speziellen zur Verfügung stellt. Es eignet sich besonders für die gradientenbasierte, numerische Optimierung von nichtlinearen Problemen, aufgrund seiner effizienten Ableitungserzeugung durch die Algorithmische Differentiation und der Möglichkeit zur Integration von gewöhnlichen Differentialgleichungen und differential-algebraischer Gleichungen. Die Interaktion mit dem Nutzer soll aus Stabilitätsgründen über die Python Schnittstelle erfolgen [Joel Andersson, 2015, S. 5f.]. Daraus resultieren weitere Anforderungen an das Modell, welche vor Beginn der Modellbildung im Kapitel 4 erörtert werden.

Weil die vorgegebene Softwareplattform zur Modellprädiktiven Regelung und deren Komponenten eine allesamt eine Schnittstelle für Python besitzen, soll die Ansteuerung und Kommunikation der gesamten Anlage in Python erfolgen. Python eignet sich hervorragend für diese Aufgabe, da es frei erhältlich und modular aufgebaut ist. Durch die Open-Source Lizenzierung ist es frei nutzbar und bietet sowohl durch die Standardbibliothek als auch durch eine Vielzahl an nutzerentwickelten Bibliotheken, die auch als Pakete bezeichnet werden, unzählige Anwendungsmöglichkeiten, zum Beispiel das pysolar Paket, das im Rahmen der Modellbildung eine Umrechnung der gemessenen Solarstrahlung in die wirkende am Fenster ermöglicht [van Rossum u. the Python development team, 2016, S. 2f.].

Der Vorgabe der Modbus Kommunikationsprotokolle für die Kommunikation ist von ebenfalls von großer Relevanz, da die komplementäre Forschungsanlage zur solaren Klimatisierung dasselbe Kommunikationsprotokoll unterstützt und durch die gewonnenen Erkenntnisse eine beschleunigte Inbetriebnahme erfolgen kann. Um das Know-how breit zu fächern, sollen die beiden Modbus RTU und Modbus TCP Protokolle Anwendung finden. Außerdem soll das Kommunikationsnetzwerk aus mehreren Subnetzwerken aufgebaut sein, die durch verschiedene elektrische und mechanische Schnittstellen implementiert werden. Für die Kommunikation über Modbus werden in den Python Paketen pymodbus, minimalmodbus und modbus-tk bereits Bausteine zur Verfügung gestellt.

Die Anlage sollte möglichst wenig Komplexität aufweisen, damit die Einarbeitung in die einzelnen Themengebiete Modellbildung, Kommunikation von technischen Systemen und Modellprädiktive Regelung vereinfacht wird. Dies soll durch eine klare Abgrenzung der verschiedenen Anlagenteile und deren Funktionen sowie durch die Strukturierung und den Aufbau der Anlage erreicht werden. Im Gegensatz dazu steht die Forderung nach dem Aufbau von Know-how auf dem Gebiet der Kommunikation technischer Systeme. Deshalb muss ein Kompromiss zwischen Verständlichkeit und Komplexität gefunden werden, um beiden Anforderungen gerecht zu werden. Dieser besteht darin, dass die Kommunikation nur über Modbus stattfindet, jedoch unter Nutzung von verschiedenen Hard- und Software-Schnittstellen, um einen gewissen Grad an Komplexität zu erreichen. Zudem können dadurch bereits Erfahrungen gesammelt werden, die für die Inbetriebnahme der solaren Klimatisierungsanlage sehr nützlich sind.

Um das Vergleichen von Ergebnissen zu ermöglichen, soll das System auf eine änderung der Steuergrößen schnell¹¹ reagieren und die Reaktionen sollen möglichst einfach messbar sein. Konkret lässt sich daraus ableiten, dass die Aktorik auf Steuersignale möglichst ohne zeitliche Verzögerung agieren sowie die gesamte Anlage einen möglichst direkten Einfluss auf die Raumtemperatur haben soll. Die Messung der Raumtemperatur kann durch einfache Temperatursensoren erfolgen die ohne großen technischen und monetären Aufwand auskommen und damit die Anforderung erfüllen.

Es soll außerdem eine hohe Funktionalität erreicht werden, um die wissenschaftliche Arbeit – durch den Ausschluss von Fehlerquellen außerhalb der Forschung – zu erleichtern. Zusätzlich ist eine Robustheit gegenüber Bedienungsfehlern und Beschädigungen erforderlich, da durch den Test- und Entwicklungsauftrag der Anlage hierzu eine erhöhte Gefahr besteht. Dies lässt sich durch wartungsarme, robuste und einfache Bauteile sowie den Aufbau der Anlage berücksichtigen, was wiederum im Einklang mit der Forderung nach einer minimalen finanziellen Belastung steht.

Wie an passender Stelle bereits erwähnt, stellt Python Schnittstellen und Funktionalitäten für die Aufgaben der Anlage zur Verfügung. Darüberhinaus bietet sich Python, durch die weitgehende Unabhängigkeit vom Betriebssystem, als zentrales Steuerungstool an, um den Controller in einem nächsten Schritt durch einen vergleichsweise günstigen Einplatinenrechner zu realisieren, wie zum Beispiel ein Raspberry Pi. Daher basiert die folgende Planung und Auslegung der Anlage auf einer zentralen Steuerung in Python.

¹¹ Im Kontrast zu den langsamen Reaktionen der solaren Klimatisierungsanlage, welche im oberen Minutenbereich liegen, bedeutet schnell in diesem Kontext im unteren Minutenbereich.

3.2 Ziel und die Idee der Anlage

3.2.1 Aufgabe der Anlage

Das Ziel ist es, die Temperatur innerhalb eines Raumes mit Hilfe eines technischen Systems zu regeln, welches den zuvor ausgeführten Anforderungen genügt.

Dazu gilt es zunächst, die Raumtemperatur über einfache Raumtemperaturfühler zu erfassen. Ist die Raumtemperatur bekannt so soll diese mit Hilfe eines Heizkörpers beeinflusst werden, damit sie einer vorgegebener Temperaturkurve folgen kann.

Die Heizleistung wiederum hängt von weiteren Eigenschaften des Heizkörpers ab und wird über das Einlassventil gesteuert. Sie wird von einem Wärmemengenzähler gemessen und indirekt über den Massenstrom und die Temperaturen des Heizwassers am Ein- und Auslass bestimmt. Dazu werden ein Durchflusssensor sowie zwei Temperatursensoren an den jeweiligen Enden des Heizkörpers eingesetzt. Um die Heizleistung zu steuern, muss also der Massenstrom gemessen und gesteuert werden sowie beide Temperaturen am Heizkörper bekannt sein. über das Ventil kann der Massenstrom mit Hilfe eines Stellantriebs eingestellt werden. Zusammen mit der Temperatur am Einlass und den Eigenschaften des Heizkörpers, lässt sich die Temperatur am Auslass und damit die aktuelle Heizleistung bestimmen. Dadurch wird eine gezielte Beeinflussung der Raumtemperatur ermöglicht.

Nachdem die Raumtemperatur bekannt ist und eine Möglichkeit für deren Manipulation besteht, wird weiterhin eine intelligente Steuerung benötigt. Dieser soll den Heizkörper möglichst ressourcenschonend einsetzen, um das Ziel der Temperaturregelung zu erreichen.

3.2.2 Idee der Anlage

Diese umfassende Aufgabe wird von einer Anlage übernommen, die sich dazu grob in drei Teile gliedern lässt. Den größten Umfang besitzt die Sensorik, die zur quantitativen Erfassung der Zustandsgrößen dient. Die aktive Beeinflussung dieser Größen erfolgt durch die Aktorik. Die Regelungsaufgabe wird von einem Controller durch seine interne Logik und der Koordination des Zusammenspiels zwischen Sensorik und Aktorik gelöst.

Um den Anforderungen gerecht zu werden, muss der logische Controller als zentrale Komponente der Anlage Python unterstützen und die gesamten Steuerungs- und Kommunikationsaufgaben übernehmen. Weiterhin müssen im Rahmen der Modellprädiktiven Regelung in regelmäßigen Zeitabständen wiederholt Optimalsteuerungsprobleme gelöst werden, weshalb eine ausreichende große Rechenkapazität benötigt wird. Um diesen Aufgaben übernehmen zu können und um keinen zusätzlichen finanziellen Aufwand zu generieren, wird zunächst ein freier Rechner der Hochschule Karlsruhe als Controller genutzt.

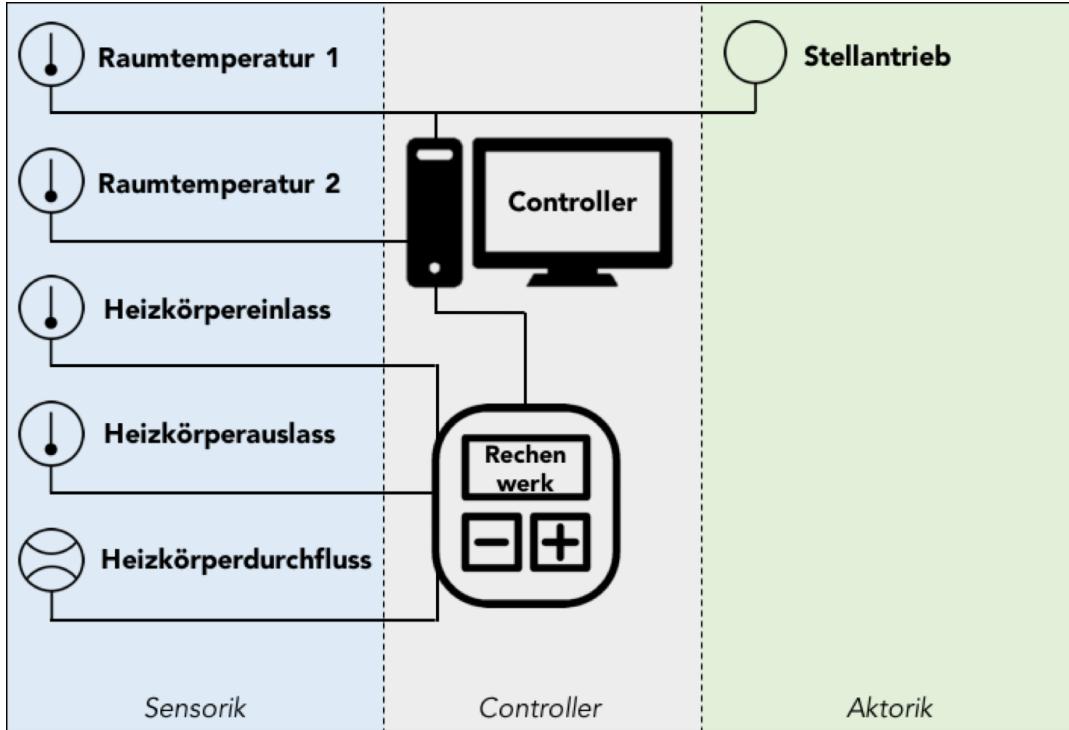


Abb. 3.2: Prinzipskizze eines technischen Systems zur Raumtemperaturregelung des Raumes K004b

Wie bereits beschrieben hat die Sensorik den größten Umfang, da sie neben der Raumtemperatur auch den Massenstrom und die Temperaturen am Heizkörper erfasst. Die Temperatur innerhalb des Raumes wird durch mehrere Sensoren erfasst, um die unterstellte Homogenität zu überprüfen und für die Modellberechnungen herzustellen. Für die Erfassung der Heizleistung wird ein Wärmemengenzähler eingesetzt, der wiederum aus einem Rechenwerk, zwei weiteren Temperatursensoren und einem Durchflusssensor aufgebaut ist. Für die Modellprädiktive Regelung werden die einzelnen Werte aller Sensoren benötigt, weshalb das Rechenwerk einen Zugriff darauf ermöglichen muss.

Die Aktorik umfasst lediglich die Ansteuerung des Heizungsventils. Wie zuvor bereits erwähnt wird dazu ein Stellantrieb genutzt, um das Ventil stufenlos zu öffnen und zu schließen und damit den Massenstrom bis hin zu einem Maximalwert zu steuern.

Eine Prinzipskizze dieser Idee ist in Abb. 3.2 graphisch dargestellt.

3.3 Konzept und Planung

Das folgende Konzept erfüllt die in Abschnitt 3.1 definierten Anforderungen und konkretisiert die zuvor geschilderte Idee. Zunächst das Netzwerk, dann xxx bis xxx

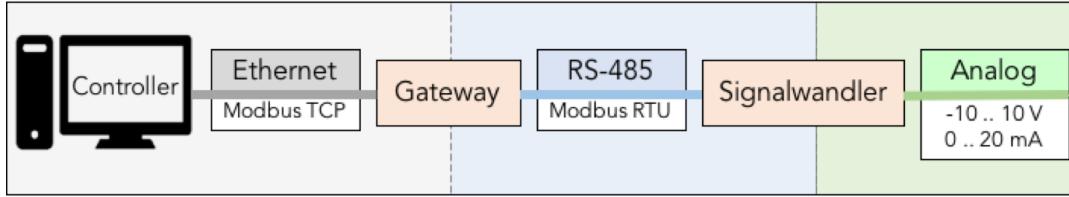


Abb. 3.3: Aufbau des Netzwerks

3.3.1 Netzwerkarchitektur

Der Rechner stellt das Zentrum der Anlage dar, deshalb werden darauf aufbauend die Kommunikationsleitungen und der Aufbau des Netzwerks geplant. Die Kommunikation erfolgt gemäß den Spezifikationen der beiden Modbus Protokolle RTU und TCP, deren Besonderheiten im Abschnitt 2.2.3 beschrieben wurden. Da der Rechner standardmäßig mit einem Ethernet-Netzwerkanschluss ausgestattet ist, wird der Rechner über ein Netzwerkkabel mit dem ersten Subnetzwerk verbunden. Das erste Netzwerk ist also ein lokales Netzwerk, das gemäß dem Ethernet/IP Protokoll ausgeführt ist und damit die Kommunikation über das Modbus TCP Protokoll ermöglicht.

Das zweite Subnetz ist ein serielles RS 485 Netzwerk und erlaubt somit den Einsatz des Modbus RTU Protokolls. Die beiden Netzwerke werden durch ein Gateway miteinander verbunden, welches die Übersetzung der Kommunikation in beiderlei Richtungen übernimmt. Die Übersetzung beinhaltet im Detail das Umwandeln der elektrischen Signale und der verschiedenen Modbus Telegrammformate ineinander.

Das eingesetzte EX9132C-2-MTCP Gateway von EXPERTDAQ bietet hierzu eine einfache und kostengünstige Möglichkeit, um von einem Modbus TCP/IP Clienten aus mit Modbus RTU Servern zu kommunizieren. Es verfügt über zwei serielle Ports verfügt, jeweils ein EIA-232 und ein EIA 485 Port. Die genauen Spezifikationen können dem Datenblatt im Anhang ?? entnommen werden.

Im dritten Subnetz wird über analoge Spannungssignale und Stromsignale kommuniziert, welche von einem Signalwandler erzeugt werden. Der Signalwandler wiederum ist kein Gateway, da die Spannungssignale nicht implizit übersetzt, sondern explizit durch Modbus RTU Befehlstelegramme festgelegt werden. Eine graphische Zusammenfassung der Netzwerkarchitektur ist in Abb. 3.3 abgebildet.

3.3.2 Erfassung der Raumtemperatur

Die Raumtemperatur kann einfach und günstig mit Raumtemperaturfühlern gemessen werden. Zudem soll der gemessene Wert über eine der beiden Modbus Schnittstellen dem Controller zur Verfügung gestellt werden. Innerhalb der Anlage kommen zwei THERMASGARD RTM1-MODBUS Raumtemperaturfühler ohne Display von S+S REGELTECHNIK zum Einsatz, weil sich diese durch weitere Eigenschaften, wie zum Beispiel der Kalibrierfähigkeit, auszeichnen. Die detaillierten Funktionen und Daten können dem Datenblatt im Anhang ?? entnommen werden.

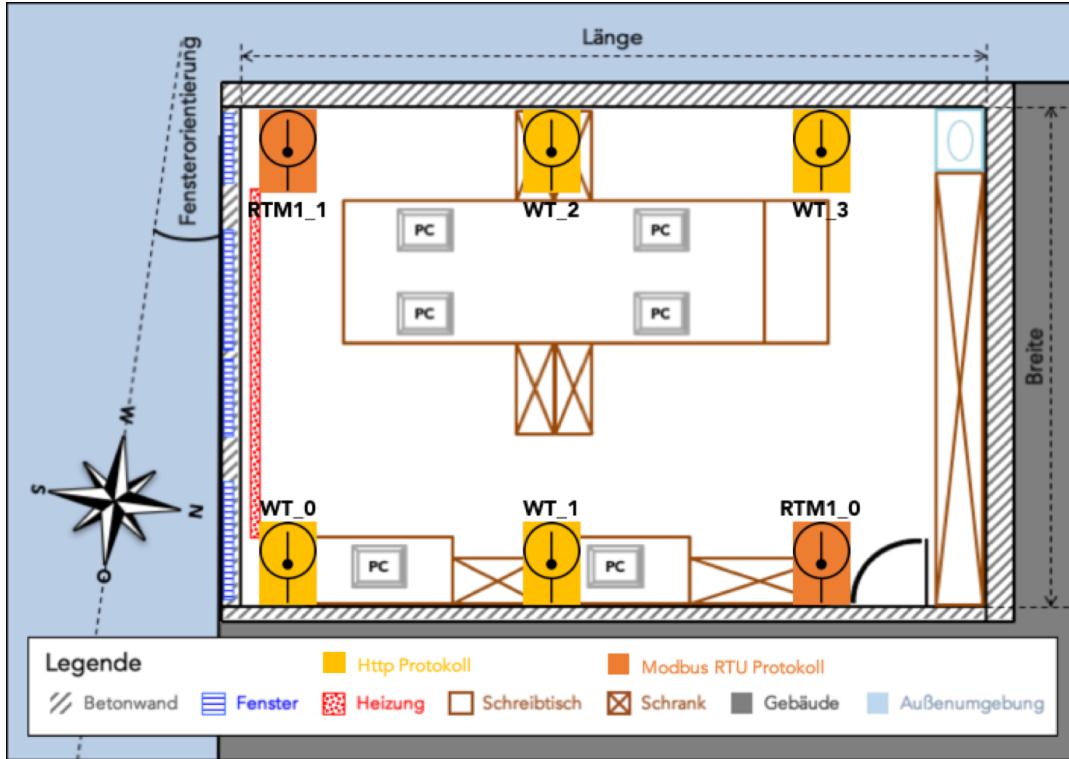


Abb. 3.4: Verteilung der Raumtemperaturfühler

Ursprünglich sollten die beiden Temperatursensoren auf einer mittig gedachten Achse im Raum in nord-südlicher Richtung platziert werden, jeweils am Ende des mittigen Schreibtischs. Jedoch wurden während der Installation der Anlage noch vier weitere Temperatursensoren zusammen mit einem Messumformer von Seiten der Hochschule Karlsruhe zur Verfügung gestellt. Bei den Sensoren handelt es sich um vier PT1000 Temperatursensoren die über den WEBTHERMOGRAPH 8X von WUT die Messwerte zur Verfügung stellen. Der Webthermograph besitzt keine Modbus Schnittstelle, jedoch kann er über ein Netzwerkkabel in das Ethernet Netzwerk integriert werden und unter Verwendung des HTTP Protokolls vom Controller ausgelesen werden. Um die Messwerte mit dem Controller auszulesen, bietet sich das Python Paket *httplib* an, welches das HTTP Protokoll implementiert und einfache Funktionen zur Verfügung stellt. Die Anordnung der Temperatursensoren wurde daher angepasst, sodass jeweils drei Sensoren an gegenüberliegenden Wänden angeordnet sind. Die Sensoren wurden entlang der westlichen Außenwand und der östlichen Innenwand gleichmäßig verteilt und auf einer Höhe von 2 m installiert. Die Identifizierung der Sensoren erfolgt über ID's, die zusammen mit der Anordnung in Abb. 3.4 visualisiert sind.

3.3.3 Steuerung des Heizkörpers

Um die Heizung mit den benötigten Sensoren auszustatten, bietet sich wie bereits erwähnt ein Wärmemengenzähler an. Die Temperaturmessung am Ein- und Auslass der Heizung gestaltet sich jedoch etwas aufwendiger als zuvor. Dazu werden Tauchhülsen im Vor- und Rücklauf des Heizkörpers benötigt, welche vom Heizwasser direkt

umflossen werden. Die Tauchhülsen bedingen geringe bauliche Änderungen und können beispielsweise in einen Kugelhahn eingeschraubt werden. Sie sind jeweils mit einem Temperaturfühler bestückt und ermöglichen dadurch die Messung der Heizwassertemperatur. Der Einbau des Durchflusssensors erfordert ebenfalls bauliche Maßnahmen. Dazu wird entweder im Vor- oder Rücklauf, je nach Spezifikation Herstellers, ein Stück Rohrleitung entfernt und durch den Durchflusssensor ersetzt. Dieser kann beim Einbau entweder fest integriert oder durch eine Anschlussverschraubung beziehungsweise einen Flanschanschluss einfach demontierbar eingebaut werden.

In der Anlage kommt der Wärmemengenzähler MULTICAL 602 von KAMSTRUP zum Einsatz, da er neben einem geforderten Modbus Kommunikationsmodul weitere besondere Eigenschaften mit sich bringt. Die Temperaturen werden von zwei PT500 Sensoren erfasst und der Durchfluss von einem ULTRAFLOW 54 Ultraschallsensor gemessen. Die gemessenen und berechneten Werte der Heizungsleistung lassen sich einfach einzeln durch das Rechenwerk einfach auslesen. Der MULTICAL 602 ist zum einen kostengünstig und umfasst alle benötigten Sensoren, die außerdem bereits präzise aufeinander abgestimmt sind. Zum anderen ist er wartungsfrei, da der Durchfluss über kontaktlose Ultraschallmessungen bestimmt wird, indem er die Laufzeitdifferenz zwischen wechselseitig gesendeten und empfangenen Signalen zweier integrierter Ultraschallsensoren auswertet. Zudem ist der Durchflusssensor ULTRAFLOW 54 bereits mit einer Tauchhülse ausgestattet, sodass lediglich eine weitere Tauchhülse montiert werden muss.

Die Steuerung des Durchflusses im Heizkörper erfolgt über das Heizungsventil, welches mit Hilfe eines Stellantriebs geöffnet und geschlossen wird. Dieser kann von einem Elektromotor angetrieben werden, der sich durch eine schnelle Reaktion aber hohe Anschaffungskosten auszeichnet. Alternativ kann ein thermoelektrisches Element als Antrieb genutzt werden, welches sich durch geringe Anschaffungskosten auszeichnet jedoch eine langsamere Reaktionszeit besitzt. Die Ansteuerung des Stellantriebs erfolgt üblicherweise unabhängig von der Ausführung durch ein digitales oder analoges Spannungs- oder Stromsignal.

Um dem Controller die Steuerung zu ermöglichen, wird ein Signalwandler eingesetzt, der Modbustelegramme in elektrische Signale umwandeln kann. Dazu wird innerhalb der Anlage das *EX9024-M* Modul von EXPERTDAQ eingesetzt, welches sich leicht in ein serielles RS485-Netzwerk einbinden lässt. Das Modul ist Modbus RTU-fähig und besitzt vier analoge Ausgänge die verschiedene Spannungs und Strombereiche ausgeben können. Da für den Stellantrieb lediglich ein Ausgang genutzt wird, besteht explizit die Möglichkeit einer einfachen Erweiterung der Anlage um 3 weitere, analoge Komponenten, beispielsweise einem Schalter zur Bedienung der Jalousien.

Der thermoelektrische Stellantrieb *ABNM-LIN* von *Danfoss* wird zur stufenlosen Betätigung des Heizkörperventils eingesetzt. Die Ansteuerung erfolgt über ein analoges 0-10 V Signal, welches innerhalb der Anlage vom EX9024-M bereitgestellt wird. Der Stellantrieb wandelt das angelegte Signal in einen proportionalen, linearen Stellweg

um und steuert damit den Durchfluss im Heizkörper.

Das beschriebene Konzept zeichnet sich dadurch aus, dass es eine kostengünstige Umsetzung erlaubt und eine größtmögliche Kompabilität der Anlageteile und Bauteile sicherstellt. Die explizit Forderung nach der Erweiterbarkeit wird explizit durch die offene Architektur und die eingesetzten Komponenten ebenfalls berücksichtigt, sodass langfristig die Möglichkeit besteht die Raumtemperatur beispielsweise unter Zuhilfenahme einer Raumklimatisierung und der Ansteuerung der Jalousien ganzjährig zu regeln zu können.

3.4 Installation und Inbetriebnahme

Eine Übersicht über den Aufbau der Anlage findet sich in Abbildung REFXXXXX. Die Installation gliedert sich in zwei Bereiche, zunächst der Hardware Installation, die die Montage der einzelnen Komponenten, deren Stromversorgung und die Verkabelung im Netzwerk umfasst. Der zweite Teil beschäftigt sich mit der Inbetriebnahme der Anlage durch die Softwareseitige Ansteuerung und Benutzung der Anlage.

3.4.1 Hardware

Der Aufbau des Netzwerks wurde bereits in Abb. 3.3 dargestellt. Als Ethernet-Netzwerk wird das lokale Netzwerk der Hochschule Karlsruhe genutzt. Dadurch ist die flexible Ansteuerung der Anlage innerhalb des gesamten lokalen Netzwerks und über eine VPN-Verbindung auch außerhalb vom Campus möglich, auch wenn dadurch zusätzlich ein Passwortschutz erforderlich wird. Der Rechner ist bereits im Netzwerk integriert, sodass lediglich das Gateway beim IZ der Hochschule Karlsruhe registriert werden musste, um eine IP Adressierung gemäß dem Ethernet/IP Protokoll zu erhalten. Für den Webthermographen wurde dasselbe Procedere durchlaufen, um diesen in die Anlage zu integrieren.

Der Schaltplan des seriellen RS-485 Netzwerk wird in Abb. 3.5 dargestellt. Zur Verkabelung des Netzwerks wird eine Busleitung von LappKabel eingesetzt, welche zwei jeweils paarweise verdrillte Leitungspaare besitzt die durch ein Kupfergeflecht von Störungen abgeschirmt sind. Das Datenblatt der Busleitung ist in nahang XXXX zu finden. Gemäß der Spezifikationen im Modbus seriell Protokoll ZITAT erfolgt die Verkabelung der D0 Leitung die braune sowie der D1 Leitung die gelbe Ader zugewiesen. Die GNC Common wird über die weißgraue Ader verdrahtet. Das grüne Kabel kann für die gemeinsame Stromversorgung der Komponenten genutzt werden, was im Rahmen dieser Anlage durch die verschiedenen benötigten Spannungen nicht genutzt wurde. Daher hat jede Komponente ihre eigene Stromversorgung durch jeweils ein eigenes Netzteil gemäß dem Schaltplan.

Die Kabel wurden gemäß der Übersicht in REFFFFXXX verlegt und mit den Bau teilen gemäß dem Schaltplan in reffig:schaltplan verbunden. Die kurzen Stichleitungen

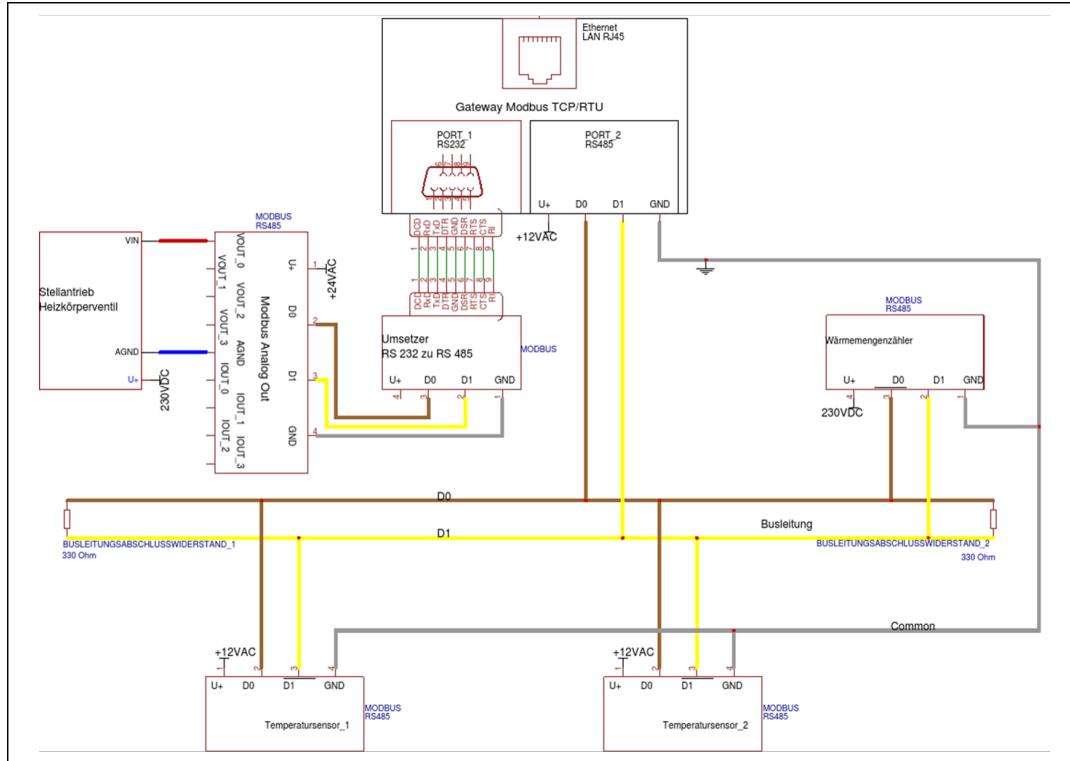


Abb. 3.5: Schaltplan der Raumtemperaturregelungsanlage

wurden über flexibel einsetzbare sowie wiederverwendbare Hebelklemmen an die lange Busleitung angeschlossen.

Die Konfiguration des RS 485 Netzwerks geschieht über das Gateway und dessen Einstellungen sind in Tabelle Tab. 3.2 zusammengefasst.

Konfigurationsmerkmal	Werte Port 1	Werte Port 2
Port	502	
Baudrate	19200 [$\frac{\text{bits}}{\text{s}}$]	9600 [$\frac{\text{bits}}{\text{s}}$]
Parität	Gerade	Keine
Datenbitlänge	8	8
Stoppbitlänge	1	1
Timeoutzeit	10 [mS]	10 [mS]
Betriebsweise	Modbus TCP zu RTU Server	Modbus TCP zu RTU Server

Tab. 3.2: Netzwerkkonfiguration der Ports des EX9132M-MTCP Gateways

Bei der Inbetriebnahme der Anlage ergab sich jedoch eine Schwierigkeit bei der Kommunikation. Es hat sich herausgestellt, dass der Signalwandler EX9024M nicht streng nach den Modbus Spezifikationen für serielle Kommunikation hergestellt wird. Dadurch ergibt sich keine kompatible Netzwerkkonfiguration, um diesen zusammen mit den beiden Raumtemperaturfühlern in einem Netzwerk zu betreiben. Da das Gateway jedoch mit zwei Ports ausgestattet ist, konnte der zweite genutzt werden um ein

weiteres serielles Netzwerk aufzubauen und damit das Problem zu lösen. Hierfür wurde ein Bridge in der Form eines einfachen Adapters eingesetzt, der die Spannungspegel und Steckerbelegungen des RS 232 Ports auf die einer RS 485 Schnittstelle umsetzt, wie im Schaltplan in Abb. 3.5 dargestellt.

3.4.2 Software

Um die zentral Steuerung vom Controller aus vorzunehmen, müssen die Schnittstellen der einzelnen Komponenten und deren Protokolle genutzt werden um alle von einer Python Konsole aus genutzt werden zu können. Wie bereits erwähnt bietet Python sehr viele Pakete die Funktionalitäten bereitstellen. Nichtsdestotrotz muss eine eigene anlagenspezifische Software entwickelt werden, die diese Funktionalitäten jedoch nutzen kann. Der gesamte folgende Programmcode erfolgt objektorientiert und versucht eine klare Struktur und Wiederverwendbarkeit der einzelnen Methoden und Klassen zu gewährleisten. Das oberste Ziel der Programmierung ist es, eine einfache Bedienbarkeit der Anlage zu ermöglichen, also eine Bedienung ohne Kenntnisse über die zugrundeliegende Kommunikationstechnik zu ermöglichen.

Um die vier Temperatursensoren über den Webthermograph8x auszulesen wurde das Python Paket `httplib` genutzt. Die Abfrage der einzelnen Temperatursensoren über das HTTP Protokoll wurde in einer eigenen Klasse `SensorsHttp` implementiert und nutzt das `HTTPConnection` Objekt aus der `httplib`. Die konkrete Umsetzung der Klasse ist in Listing 3.1 dargestellt.

Um die Verbindung zum Thermograph8x herzustellen wird dessen IP Adresse benötigt und die Port Nummer, über die die Telegramme gesendet und empfangen werden. Aus Gründen des Zugriffsschutzes sind diese Werte nicht im Code hinterlegt, sondern in den Umgebungsvariablen `server-address` und `server-port` der Pythonkonsole des Controllers. Bei der Initialisierung wird mit der `init`-Methode ein `HTTPConnection` Objekt `server` erstellt, welches als Host und Port die Adressen des Webthermographen übergeben bekommt. Des Weiteren wird jeweils eine Methode zur Herstellung und zur Trennung der Verbindung definiert sowie eine abstrakte Methode `get-wt`, welche die allgemeine Abfragesyntax der Messwerte enthält. Die Messwerte können lediglich einzeln abgefragt werden, über eine einfache Serveranfrage `request`, welche die Messwerte durch einen einfachen Befehls-string, „`GET\Single`“ zusammen mit der ID des gewünschten Temperatursensors, anfordert. Nachdem der Messwert in einen `float` Datentyp umgewandelt zurückgegeben wurde wird am Ende die Verbindung zum Server wieder getrennt. Die Methoden zum Auslesen der einzelnen Messwerte rufen lediglich die abstrakte Methode auf und übergeben dabei die ID des gewünschten Temperatursensors.

```
1from httplib import HTTPConnection  
  
3class SensorsHttp():  
    @property  
5        def server_address(self):
```

```

    return self.server.host

7
@property
9 def server_port(self):
    return str(self.server.port)

11 def __init__(self, server_address, server_port):
13     self.server = HTTPConnection(host=server_address, port=int(server_port))

15 def __connect_to_server(self):
    self.server.connect()

17 def __disconnect_from_server(self):
19     self.server.close()

21 def __get_wt(self, unit):
    self.__connect_to_server()
    self.server.request('GET', '/Single' + str(unit))
    response_string = self.server.getresponse().read()
25    temperature = float(response_string.split(';')[-1][-4:].replace(',', '.'))
    return temperature
27 finally:
    self.__disconnect_from_server()

29 def get_wt_0(self):
31     return self.__get_wt(unit=1)

33 def get_wt_1(self):
35     return self.__get_wt(unit=2)

37 def get_wt_2(self):
39     return self.__get_wt(unit=3)

41 def get_wt_3(self):
43     return self.__get_wt(unit=4)

```

Listing 3.1: Klasse SensorsHttp zur Abfrage der Temperaturen vom Webthermograph8x

Für die Kommunikation mit den Modbus Komponenten stehen drei verschiedene Python Pakete zur Verfügung. Da Minimalmodbus leider keine Unterstützung für das Modbus TCP Protokoll bietet, kann es nicht zum Einsatz kommen. Nach ausführlicher Recherche über die beiden verbleibenden Pakete, fiel die Entscheidung pymodbus einzusetzen, da dessen Funktionsumfang viel umfangreicher und Dokumentation ausführlicher gestaltet und mit einfachen Beispielen gespickt ist. Mit Hilfe der des pymodbus Pakets wurde daher zunächst wieder eine eigene Klasse ModbusConnection für die Kommunikation programmiert, welche in 3.2 zu sehen ist.

Damit über das Modbus TCP Protokoll kommuniziert werden kann, muss auch hier zunächst eine Verbindung zwischen Client/Master und Server/Slave hergestellt werden. Für die Kommunikation wird das ModbusClient Objekt genutzt, da der Controller als Client über das Modbus TCP Protokol Anfangan an die Server stellt. Konkret stellt der Controller lediglich an das Gateway als Server Anfragen, welches die Telegramme in das serielle Modbus RTU Protokoll übersetzt um mit den Komponenten in den seriellen Netzwerken zu kommunizieren. Bei der Initialisierung werden wiederum die IP Adresse client-adress und der Port client-port benötigt über den die Kommunikation stattfindet. Des Weiteren wurden eigene Methoden zum Verbinden und Trennen

implementiert, welche bei der Initialisierung genutzt werden. Eine Besonderheit stellt die delete-Methode dar, die im Falle eines Fehlers oder sterben einer Instanz der Klasse die Verbindung trennt. Zudem sind Methoden für die benötigten Zugriffe gemäß dem Modbus Datenmodell auf die Datentabellen implementiert. Diese umfassen das Auslesen von input- und holding-registers und liefern ein register zurück, was je nach Datenmodell einen Datentyp besitzt. Für die beiden Lesemethoden wird ein maximales probieren von 10 mal vorgesehen, bei einem Fehlversuch wird eine Pause von 1,5 sekunden gemacht bevor der nächste versuch gestartet wird. Für die Abfrage wird die Adresse in der Tabelle benötigt, die Anzahl der auszulesenden Worte/register/Datentypen sowie die Slave/Server-ID. Die letzte Methode um Register zu schreiben wird auf 3 Versuche begrenzt, ebenfalls mit Pause bei Fehler. Auch hier wird die Adresse in der Tabelle sowie die Slave/Server ID um den ebenfalls mitzugebenden Wert zu schreiben benötigt. Damit sind die grundlegenden Funktionen für die Modbus Kommunikation implementiert und können zum Auslesen und Schreiben von Werten genutzt werden.

```

1 from pymodbus.client.sync import ModbusTcpClient as ModbusClient
  from time import sleep
3
  class ModbusConnection(object):
5      _max_retries_read = 10
      _max_retries_write = 3
7
      @property
9       def max_retries_read(self):
          return self._max_retries_read
11
      @property
13       def max_retries_write(self):
          return self._max_retries_write
15
      @property
17       def client_address(self):
          return self.client.host
19
      @property
21       def client_port(self):
          return str(self.client.port)
23
      def __init__(self, client_address, client_port):
25          self.client = ModbusClient(host=client_address, port=int(client_port))
          self.connect_to_client()
27
      def __del__(self):
29          self.disconnect_from_client()
31
      def connect_to_client(self):
          self.client.connect()
33
      def disconnect_from_client(self):
          self.client.close()
35
37      def read_input_registers(self, address, count, unit):
          k = 0
          while k < self.max_retries_read:
              try:
41                  return self.client.read_input_registers(address=address, count=count, unit=unit)

```

```

        = unit). registers
    except:
43        k += 1
        sleep(1.5)

45    def read_holding_registers(self, address, count, unit):
46        k = 0
47        while k < self.max_retries_read:
48            try:
49                return self.client.read_holding_registers(address=address, count=count,
50                                                unit=unit). registers
51            except:
52                k += 1
53                sleep(1.5)

55    def write_register(self, address, unit, value):
56        k = 0
57        while k < self.max_retries_write:
58            try:
59                return self.client.write_register(address=address, unit=unit, value=
60                                                value)
61            except:
62                k += 1
63                sleep(1.5)

```

Listing 3.2: Klasse zum Verbindungsauflauf und Für die Grundfunktionen über Modbus TCP

Diese Grundfunktionen werden von der SensorsModbus Klasse genutzt, um die Messwerte einzeln von den Sensoren abzufragen. Der Code dazu ist in Listing 3.3 abgebildet. Eine Besonderheit ist beim Auslesen der Werte aus dem Wärmemengenzähler gegeben, da dessen einzelne Messwerte die Größe eines Registers übersteigt. Wie in Abschnitt XXXXX bereits erwähnt, werden die Daten dann gemäß der Big-Endian Codierung verschlüsselt und müssen decodiert werden. Dazu bietet pymodbus ebenfalls eine entsprechende Funktion durch den BinaryPayload Decodierer mit der Besonderheit an, dass die Codierung little und big vertauscht sind bzw dass das gewünschte format angegeben werden muss. Die Länge von 2 Registern der Daten ist im Count angegeben und die ID im seriellen Netzwerk des Wärmemengenzählers ist ebenfalls angegeben. Die konkreten einzelnen Messwerte haben verschiedene Adressen in der Datentabelle welche in den Methoden gespeichert sind. Die Adressen finden sich in Tabelle im Datenblatt im Anhang bzw in Tabelle. Die beiden Temperatursensoren geben den Temperaturwert als integer ganzzahligen Wert zurück um eine Zehnerpotenz erhöht. Daher müssen auch dessen Werte decodiert werden durch eine Division durch zehn. Sie stehen jeweils in derselben Tabelle unter der gleichen Adresse, jedoch haben die beiden Temperatursensoren unterschiedliche IDs zugewiesen bekommen, durch die Schalterstellung gemäß Datenblatt/Adressmapping.

```

from pymodbus.client.sync import ModbusTcpClient as ModbusClient
2from modbus_connection import ModbusConnection
from pymodbus.constants import Endian
4from pymodbus.payload import BinaryPayloadDecoder

6class SensorsModbus(ModbusConnection):
    def __get_mc602(self, address):
8        registers = self.read_input_registers(address=address, count=2, unit=65)
        decoder = BinaryPayloadDecoder.fromRegisters(registers, endian=Endian.Little)

```

```

10         float_value = decoder.decode_32bit_float()
11         return float_value
12
13     def __get_rtm1(self, unit):
14         registers = self.read_input_registers(address = 0, count = 1, unit = unit)
15         temperature = registers[0] / 10.0
16         return temperature
17
18     def get_mc602_temperature_inlet(self):
19         return self.__get_mc602(address = 16)
20
21     def get_mc602_temperature_outlet(self):
22         return self.__get_mc602(address = 20)
23
24     def get_mc602_flowrate(self):
25         return self.__get_mc602(address = 4)
26
27     def get_rtm1_0(self):
28         return self.__get_rtm1(unit = 3)
29
30     def get_rtm1_1(self):
31         return self.__get_rtm1(unit = 4)

```

Listing 3.3: Klasse zum Auslesen über Modbus TCP

Die Klasse AcuatorsModbus erweitert ebenfalls die ModbusConnection Klasse, allerdings um die Aktuatoren anzusteuern. Dazu wird der Spannungsausgang am EX9024M mithilfe der dezimalen Registerwerten zwischen 8191 und 16383 gesteuert, die sich aus den Hexadezimalwerten im Datenblatt REFXXX ergeben. Der Maximalwert entspricht einer Ausgangsspannung von 10 V, der Minimalwert entsprechend 0 V. Im Rahmen der Anlagensteuerung soll jedoch nicht die Ausgangsspannung, sondern die Ventilstellung gesetzt werden. Um die Ventilstellung opening-level direkt anzugeben, 0 für Ventil zu und 1 für voll offen, erfolgt eine Umrechnung, zunächst in den entsprechenden Spannungswert und anschließend in den Registerwert der in die Datentabelle geschrieben wird. Außerdem wurde eine Methode zur Abfrage der Ventilstellung implementiert.

```

1 from pymodbus.client.sync import ModbusTcpClient as ModbusClient
2 from modbus_connection import ModbusConnection
3
4     class ActuatorsModbus(ModbusConnection):
5         _ex9024_min_register_value = 8191
6         _ex9024_max_register_value = 16383
7
8         @property
9         def ex9024_min_register_value(self):
10             return self._ex9024_min_register_value
11
12         @property
13         def ex9024_max_register_value(self):
14             return self._ex9024_max_register_value
15
16         def __set_ex9024(self, address, voltage_value):
17             register_value = 8191 + 819.2 * voltage_value
18             registers = self.write_register(address = address, unit = 8, value =
19                 register_value)
20             return registers
21
22         def __get_ex9024(self, address):
23             registers = self.read_holding_registers(address = address, count = 1, unit = 8)

```

```

23         voltage_value = (float(registers[0]) - 8191)/819.2
24         return voltage_value
25
26     def set_ex9024(self, opening_level):
27         opening_level = float(opening_level)
28         if opening_level < 0 or opening_level > 1:
29             raise ValueError("Only values between 0 and 1 allowed for opening level
30                           definition .")
31         voltage_value = opening_level * 10
32         self.__set_ex9024(address = 2, voltage_value = voltage_value)
33
34     def get_ex9024(self):
35         voltage_value = self.__get_ex9024(address = 2)
36         opening_level = voltage_value/10.0
37         return opening_level

```

Listing 3.4: Klasse zum Auslesen über Modbus TCP

Um nun eine Bedienoberfläche zu schaffen die Plattformunabhängig bzw Kommunikationstechnologieunabhängig ist, wurden zwei weitere Klasse implementiert, die die jeweils vorausgegangenen Sensor und Aktor Klassen ein letztes Mal erweitert. Die Klassen Sensors und Actuators schaffen dies, indem sie die zuvor definierten Klassen importieren und bei der Initialisierung jeweils eine Instanz der Klasse bilden/instanzieren. Um die Unabhängigkeit von Kommunikationstechnologie zu erreichen, wird für jeden einzelnen Messwert eine eigene Methode mit eindeutiger Bezeichnung der Funktion und des Geräts definiert ohne dass ersichtlich ist welche Kommunikation darunter steckt. Damit ist ein Interface bze eine Oberfläche geschaffen die von Steuerungen genutzt werden können.

```

1 from actuators_modbus import ActuatorsModbus
2 from httpplib import HTTPConnection
3 from sensors_modbus import SensorsModbus
4 from sensors_http import SensorsHttp

6 class Actuators():
8
8     def __init__(self, client_address, client_port):
9         self.actuators_modbus = ActuatorsModbus(client_address, client_port)
10
11    def set_valve_position(self, opening_level):
12        return self.actuators_modbus.set_ex9024(opening_level)
13
14    def get_valve_position(self):
15        return self.actuators_modbus.get_ex9024()
16
18 class Sensors():
19
20     def __init__(self, client_address, client_port, server_address, server_port, port,
21                  baudrate, parity, stopbits, bytesize, timeout):
22         self.sensors_modbus = SensorsModbus(client_address, client_port)
23         self.sensors_http = SensorsHttp(server_address, server_port)
24
25    def get_temperature_northeast(self):
26        return self.sensors_modbus.get_rtm1_0()
27
28    def get_temperature_northwest(self):
29        return self.sensors_http.get_wt_3()
30
31    def get_temperature_west(self):
32

```

```

30     return self.sensors_http.get_wt_2()

32 def get_temperature_southwest(self):
33     return self.sensors_modbus.get_rtm1_1()

34 def get_temperature_southeast(self):
35     return self.sensors_http.get_wt_0()

36 def get_temperature_east(self):
37     return self.sensors_http.get_wt_1()

38 def get_radiator_flowrate(self):
39     return self.sensors_modbus.get_mc602_flowrate()

40 def get_radiator_temperature_inlet(self):
41     return self.sensors_modbus.get_mc602_temperature_inlet()

42 def get_radiator_temperature_outlet(self):
43     return self.sensors_modbus.get_mc602_temperature_outlet()

```

Listing 3.5: Die Actutators und Sensors Klassen zur Bedienung der Anlage

3.4.3 Inbetriebnahme mit Zweipunktregler

Um die Funktion der Anlage zu überprüfen und bereits Messdaten zu sammeln, wurde ein erstes, einfaches Programm zur Regelung erstellt, welches bis zum Einsatz der Modellprädiktiven Regelung die Raumtemperaturregelung übernimmt. Anhand dessen Programmablauf wird im Folgenden das Zusammenspiel der Anlage und die Ansteuerung der Anlagekomponenten erläutert.

Das Programm ist in Listing 3.6 dargestellt und implementiert einen einfachen Zweipunktregler, der nach dem bekannten Backofenprinzip arbeitet und eine Schalthysterese von einem Grad Celsius besitzt. Zunächst werden die beiden Bedienoberflächen Sensors und Actuators importiert, gemeinsam mit einem Paket zum einfügen der Messwerte in eine Datenbank. Anschließend werden die verschiedenen Verbindungsparameter aus den Umgebungsvariablen der Pythonkonsole ausgelesen und zum Aufbau der Verbindung zu den Anlagenteilen und zur Datenbank genutzt. Dies geschieht durch die Instanzierung der Bedienoberflächeklassen Sensors und Actuators, die den Variablen `s` und `a` zugeiwsen werden. Des Weiteren werden die beiden Ventilstellungen für die Schaltpunkte definiert. Nun startet der eigentliche Steueralgorithmus innerhalb einer while-Schleife. Die dauerhafte Laufzeit wird über die Schleifenbedingung `true` erreicht, welche durch eine Tastenkombination oder bei Auftreten eines Fehlers gestoppt wird. Dabei werden die Verbindungen zu den Anlagenteilen trennt damit wird eine schnelle Wiederaufnahme des Betriebs gewährleistet wird und die Komponenten nicht durch eine bestehende Verbindung blockiert sind. Zu Beginn werden alle Messwerte innerhalb der Anlage ausgelesen und das arithmetische Mittel der Raumtemperaturen einfach ermittelt. Die angenommene Raumtemperatur lässt sich über den Gewichtungsvektor `weightings` auch als gewichtetes arithmetisches Mittel berechnen. Zum Auslesen der Werte, werden die Methoden des Sensors Objekts `s` und des Actuators Objekts `a` aufgerufen. Das Abfangen von Fehlern wurde bereits in den unteren Klassen implementiert,

sodass die Methoden zu Anlagenbedeutung einfach und robust sind. Außerdem wurden möglich eFehlebbdieungen ebenfalls ausgeschlossen an gegebenen Stellen, sodass z.B. unmögliche Ventilstellungen oder Werte nicht geschrieben oder angefordert werden können.

Anhand des Mittelwerts wird dann entschieden, ob der Heizkörper zur Raumerwärmung eingeschalten, weiter genutzt oder abgeschalten werden soll. Ist die Temperatur unter 21 Grad Celsius gestiegen und der Heizkörper wird noch nicht genutzt, erfolgt eine Ausgabe in der Konsole, dass es zu kalt ist und das Heizungsventil wird vollständig geöffnet. Ist die Temperatur über 22 Grad Celsius gestiegen und der Heizkörper wird genutzt, erfolgt eine Ausgabe, dass es zu warm ist, und das Heizungsventil wird geschlossen. Danach erfolgt ein Update der Ventilstellung, bevor die Messdaten in der Konsole ausgegeben und in eine Datenbank eingefügt werden. Abschließend wartet die Steuerung für 60 Sekunden, bevor der Algorithmus von vorne beginnt/abläuft.

```

from sensors_actuators_k004b.sensors import Sensors
2from sensors_actuators_k004b.actuators import Actuators
from mdb_communication_k004b.dbinsert import DBInsert
4import time
import os
6import numpy as np
# Modbus Connection Parameter
8sensor_client_address = os.environ["EX9132_ADDRESS"]
sensor_client_port = os.environ["EX9132_PORT_1"]
10actuator_client_address = os.environ["EX9132_ADDRESS"]
actuator_client_port = os.environ["EX9132_PORT_2"]
12# http Connection Parameter
server_address = os.environ["WT_ADDRESS"]
14server_port = os.environ["WT_PORT"]
# DB Connection parameter
16db_name = os.environ["DB_NAME"]
db_host = os.environ["DB_HOST"]
18db_port = os.environ["DB_PORT"]
db_user = os.environ["DB_USER"]
20user_password = os.environ["USER_PASSWORD"]

22s = Sensors(sensor_client_address, sensor_client_port, server_address, server_port)
a = Actuators(actuator_client_address, actuator_client_port)
24
dbi = DBInsert(db_name = db_name, db_host = db_host, db_port = db_port, db_user =
db_user, user_password = user_password)
26
valve_open = 1.0
28valve_closed = 0.0

30try:
    while True:
32        time_measurement = time.time()
        rtm1_0 = s.get_temperature_northeast()
34        wt_3 = s.get_temperature_northwest()
        wt_2 = s.get_temperature_west()
36        rtm1_1 = s.get_temperature_southwest()
        wt_0 = s.get_temperature_southeast()
38        wt_1 = s.get_temperature_east()
        ex9024_level = a.get_valve_position()
40        flowrate = s.get_flowrate()
        temperature_inlet = s.get_radiator_temperature_inlet()
42        temperature_outlet = s.get_radiator_temperature_outlet()

```

```

44     temperatures = [rtm1_0, rtm1_1, wt_0, wt_1, wt_2, wt_3]
45     weightings = [1, 1, 1, 1, 1, 1]
46     temperatures_weighted = 0
47     for j, temperature in enumerate(temperatures):
48         temperatures_weighted += weightings[j] * temperature
49
50     temperature_room = temperatures_weighted / sum(weightings)
51
52     if (temperature_room < 21.0) and (abs(ex9024_level_valve_open) > 0.05):
53         print "too cold"
54         a.set_valve_position(opening_level = valve_open)
55
56     elif (temperature_room > 22.0) and (abs(ex9024_level_valve_closed) > 0.05):
57         print "too hot!"
58         a.set_valve_position(opening_level = valve_closed)
59
60     ex9024_level_new = a.get_valve_position()
61
62     data_set = [time.ctime(), flowrate, temperature_inlet, temperature_outlet,
63                 ex9024_level_new, rtm1_0, rtm1_1, wt_0, wt_1, wt_2, wt_3, round(
64                     temperature_room, 2)]
65     print data_set
66
67     dbi.insert_temperature_1(time_measurement, rtm1_0)
68     dbi.insert_temperature_2(time_measurement, wt_3)
69     dbi.insert_temperature_3(time_measurement, wt_2)
70     dbi.insert_temperature_4(time_measurement, rtm1_1)
71     dbi.insert_temperature_5(time_measurement, wt_0)
72     dbi.insert_temperature_6(time_measurement, wt_1)
73     dbi.insert_flowrate(time_measurement, flowrate)
74     dbi.insert_temperature_inlet(time_measurement, temperature_inlet)
75     dbi.insert_temperature_outlet(time_measurement, temperature_outlet)
76     dbi.insert_valve_position(time_measurement, ex9024_level)
77     dbi.insert_control_method(time_measurement, 1)
78
79     time.sleep(60.0)
80
81 except KeyboardInterrupt:
82     del a
83     del s
84     raise

```

Listing 3.6: Zweipunktregler Programm zur Inbetriebnahme der Anlage

Aus Gründen der Übersichtlichkeit sind die Abhängigkeiten zwischen den Klassen in Abbildung in einem UML Klassendiagramm in Abb. 3.6 graphisch dargestellt. Eine alternative Modellprädiktive Regelung würde

Die Anlage konnte aufgrund von Lieferschwierigkeiten des Wärmemengenzählers erst am 16.12.2015 erfolgreich in Betrieb genommen werden. Seither sind lediglich kleinere Fehler aufgetreten, wodurch die Programmierung der Anlage stetig und sukzessive verbessert wurde. Seit Mitte Januar läuft der Betrieb der Anlage durchgängig stabil und fehlerfrei und leistet einen sehr zuverlässigen Dienst. Damit wird der Forderung nach einer hohen Funktionalität genüge getragen. Zudem wird auch die gewünschte Robustheit erfüllt, durch die fehlerabfangende Software.

Der gemessene Raumtemperaturverlauf über den Zeitraum von Mitte Januar bis Mitte April ist in REFXXXXX dargestellt und die These bestätigt werden, dass

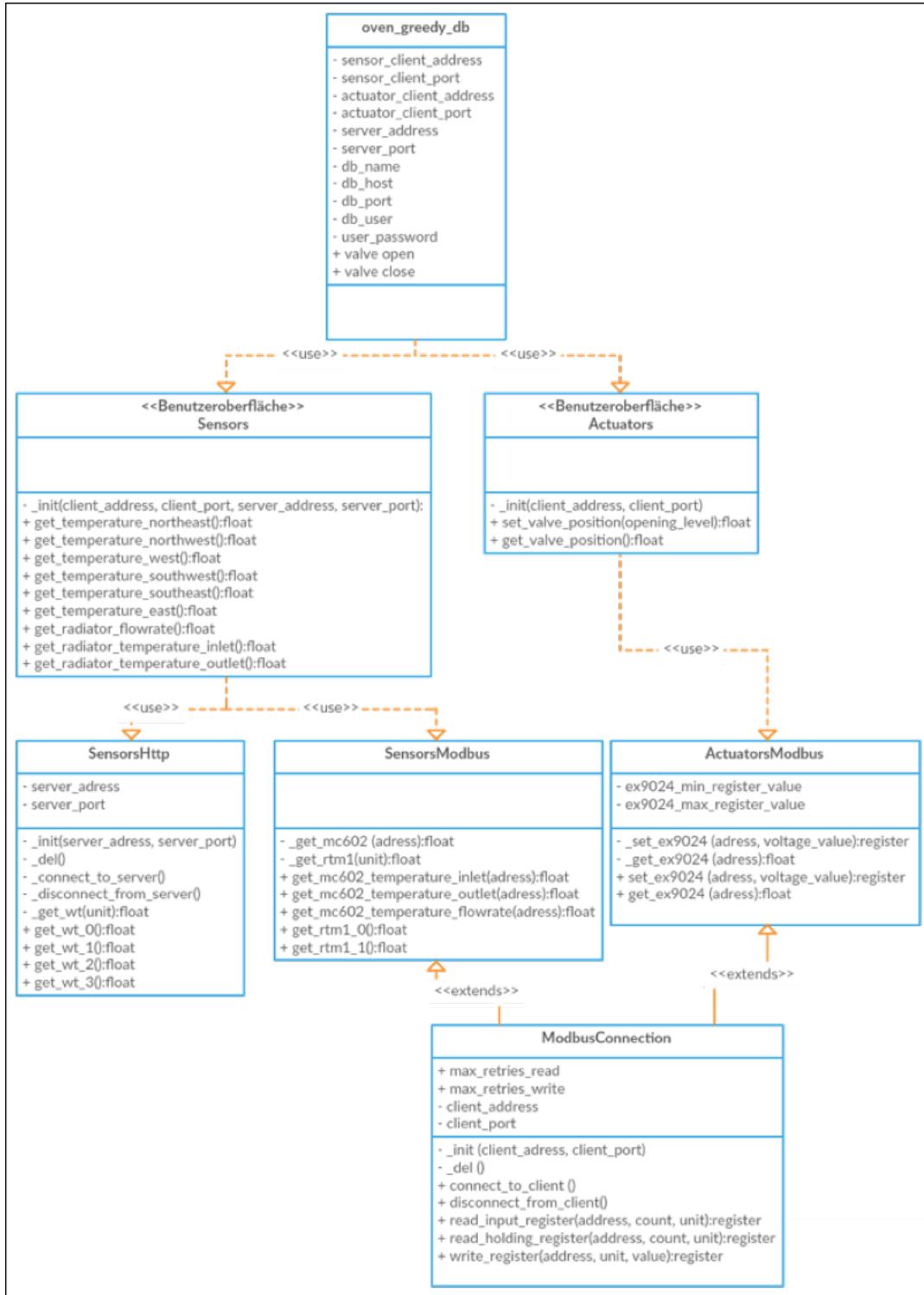


Abb. 3.6: UML Klassendiagramm der zentralen Anlagensteuerung

die Anlage eine Raumtemperatur regeln kann. Aufgrund der zentralen Steuerung der Anlage, welche in Python umgesetzt wurde und fehlerfrei funktioniert, sind die Voraussetzungen für eine Regelbarkeit mit Hilfe von Modellprädiktiver Regelung erfüllt. Um die Eignung zu überprüfen, wird jedoch noch ein Modell des Raumes und der Anlage benötigt, welches im nachfolgenden Kapitel gebildet wird.

„Das beste Modell für eine Katze ist eine Katze; möglichst dieselbe Katze.“
— NORBERT WIENER

4 Modellbildung und Simulation

Ziel dieses Kapitels ist es, ein hinreichend exaktes Modell zur Berechnung der Raumtemperatur in K004b zu bilden. Die Modellbildung soll physikalisch motiviert sein und daher auf den thermodynamischen Prozessen mit der Außenumgebung und der Anlage aus Kapitel 3 basieren. Des Weiteren soll das Modell speziellen Anforderungen genügen, um eine Modellprädiktive Regelung der Anlage zu ermöglichen.

Dazu wird zunächst ein einfaches Grundmodell für einen hypothetischen Raum gebildet, dass anschließend schrittweise an den bestehenden Raum erweitert angepasst wird, bis eine die Qualität/Güte des Modells ausreichend ist.

4.1 Modellbildung

4.1.1 Anforderungen an das Raummodell

Die triviale Aufgabe des Modells ist eine hinreichend genaue Beschreibung der realen Vorgänge und des Temperaturverlaufs. Hinreichend bedeutet, dass das Modell eine ausreichende Güte für den Einsatz mit Modellprädiktiver Regelung besitzt. In Kapitel 2.1 wurden bereits die Grundlagen zur Modellprädiktiven Regelung erläutert, wodurch sich Einschränkungen bei der Modellbildung ergeben. Dabei wurde festgestellt, dass die Lösung von Optimalsteuerungsproblemen gradientenbasiert erfolgt und die zweifache Erzeugung von Ableitung erforderlich. Damit ergibt sich die Anforderung, dass das Modell keine Unstetigkeiten aufweist und sich zweimal stetig differenzieren lässt. Zudem ist Berechnung von Lösungen für Optimalsteuerungsprobleme sehr rechenintensiv und muss während des laufenden Betriebs der Anlage wiederholt stattfinden. Da die Lösung zudem wiederholt stattfinden muss, wird eine erhöhte Rechenkapazität benötigt, um die Lösung in ausreichender Zeit zu berechnen.

Daraus ergibt sich eine weitere Anforderung, denn um den Rechenbedarf möglichst gering zu halten, soll das Modell möglichst wenig Komplexität besitzen. Wie bereits bei den Anforderungen an die Anlage in Abschnitt 3.1 festgestellt wurde, stellt die Umgebung zur Optimalsteuerung einen begrenzenden Faktor dar, wodurch das Modell mit der Modellierungssprache Modelica gebildet wird.

Da diese Anforderungen teils gegenläufig sind, gilt es einen Kompromiss zu finden, ein stetig differenzierbares Modell, um eine ausreichende Modellgüte zu erhalten, gleichzeitig jedoch keine hohe Komplexität oder einen großen Umfang an Gleichungen besitzt. Ziel: Hohe Modellgüte bei gleichzeitig geringer Komplexität und Stetigkeit

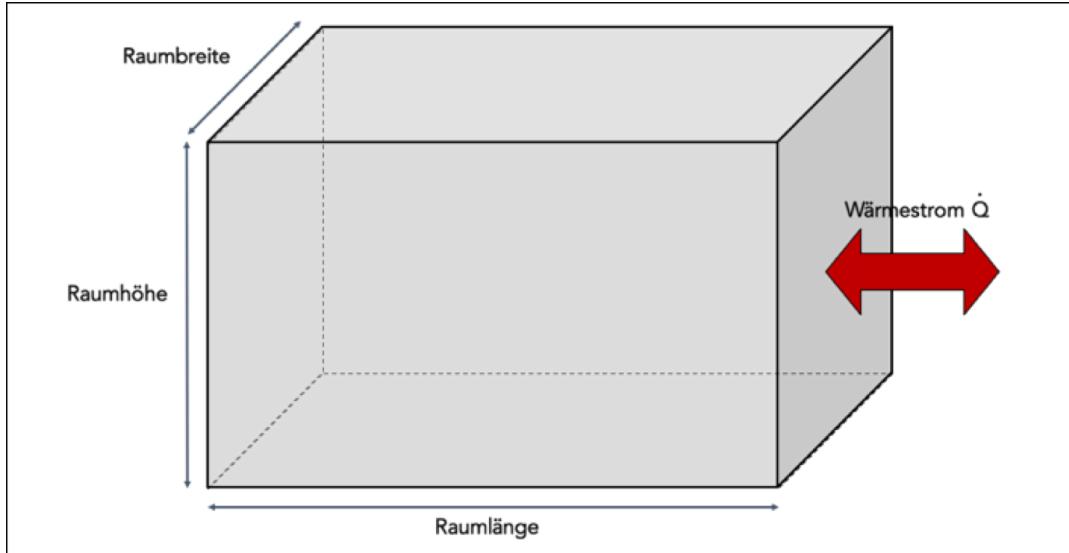


Abb. 4.1: Grundmodell eines Raumes

Daher wird im Folgenden zunächst ein simples Raummodell gebildet, dass anschließend sukzessive erweitert und damit die Komplexität des Modells schrittweise erhöht wird, bis eine ausreichende Beschreibung der Realität mit dem Modell möglich ist.

4.1.2 Das Grundmodell des Raumes

Um ein möglichst einfaches Grundmodell zu erhalten, wird zunächst ein hypothetischer Raum betrachtet. Dieser Raum bildet zusammen mit der ihn umgebenden Luft ein abgeschlossenes thermodynamisches System, wie in Kapitel 2.3 beschrieben. Der Raum ist selbst mit Luft gefüllt und wird zu allen sechs Seiten hin durch Wände begrenzt. Damit bildet der Raum ein geschlossenes System, da keine Massenströme über die Grenzen hinweg fließen können. An den Grenzflächen kann also lediglich Wärme zwischen der Umgebung und dem Raum ausgetauscht werden. Des Weiteren wird eine homogene Temperatur innerhalb des Raumes und der Umgebung angenommen, welche in der Realität eingeschwungenen Gleichgewichtszuständen innerhalb der beiden Teilsysteme entspricht. Um die Annahme für den Raum zu überprüfen, muss noch festgestellt werden auf welcher zeitlichen Skala der Einschwingvorgang für eine homogene Temperatur innerhalb des Raumes stattfindet und ob dieser damit eine Relevanz für die Modellbildung besitzt.

Zur Bestimmung der Temperatur innerhalb des Raumes, ausgehend von einer initialen Raumtemperatur und dem externen Steuerungsparameter der Umgebungstemperatur, muss der Ausgleichsprozess zwischen Raum und Umgebung untersucht werden, konkret der ausgetauschte Wärmestrom. Um diesen nach Gl. 5 zu berechnen, müssen zunächst die verschiedene modellrelevanten Eigenschaften des Raumes durch physikalische Größen und Variablen beschrieben werden. Zur Berechnung der Austauschoberfläche wird die Raumbreite, -länge und -höhe benötigt und weiterhin sind der U-Wert einer Betonwand, die spezifische Wärmekapazität und Dichte von Luft für die Bestimmung

des Wärmestroms relevant. Diese modellrelevanten Eigenschaften sind allesamt mit ihren Zahlenwerten in Tabelle Tab. 4.1 zusammengefasst.

Modellrelevante Eigenschaften	Wert	Einheit
Raumbreite	7,81 ¹⁾	[m]
Raumlänge	5,78 ¹⁾	[m]
Raumhöhe	2,99 ¹⁾	[m]
Wärmedurchgangskoeffizient Betonwand	1,0 ²⁾	[$\frac{W}{m^2 \cdot K}$]
Spezifische Wärmekapazität von Luft	1.000,0 ³⁾	[$\frac{J}{kg \cdot K}$]
Dichte von Luft	1,25 ³⁾	[$\frac{kg}{m^3}$]

¹⁾Werte durch eigene Vermessung des Raumes K004b vom 07.12.2015.

²⁾Schätzwert, geschätzt nach [Recknagel, 2013, S. 409] mit Richtwerten aus [Recknagel, 2013, S. 194ff].

³⁾Tabellenwert aus [Peter Häupl, 2013, S. 68].

Tab. 4.1: Eigenschaften des Raummodells

Erfolgt nun die Bilanzierung des Raumes mit Hilfe des ersten Hauptsatzes der Thermodynamik nach Gl. 4 und die Berechnung der inneren Energie des Raumes nach Gl. 3 ergibt sich folgendes, einfaches Gleichungssystem zur Bestimmung der Raumtemperatur in Abhängigkeit vom Steuergrößen Außentemperatur im Grundmodell in Modelica:

```

equation
2  /* calculate room volume */
  room_volume = room_length * room_height * room_breadth;
4  /* calculate room mass */
  room_mass = room_volume * rho_air;
6  /* calculate surface of heat exchange */
  exchange_surface = 2 * (room_length * room_breadth) + 2 * (room_length * room_height) +
    2 * (room_breadth * room_height);
8  /* calculate inner energy*/
  room_u = room_mass * cp_air * room_temperature;
10 /* calculate derivative of the inner energy */
  der(room_u) = environment_qdot;
12 /* calculate heatflow between room and environment */
  environment_qdot = u_wall * exchange_surface * (environment_temperature -
    room_temperature);

```

Listing 4.1: Einfaches Gleichungssystem für das Grundmodell des Raumes in Modelica

Damit ist ein Grundmodell für einen Raum gebildet, wie in Abb. 4.1 graphisch dargestellt, um die Temperatur innerhalb eines Raumes zu berechnen. Dieses wird im Folgenden nun schrittweise erweitert und zum Abschluss überprüft, ob es der Realität genüge zu trägt.

4.1.3 Modellerweiterung durch Berücksichtigung der realen Umgebung

Im nächsten Schritt wird das einfache Raummodell zunächst an die reale Umgebung des Raums K004b angepasst. Die Lage von K00b ist in Abb. 3.1 ersichtlich und es ist zu

erkennen, dass der Raum lediglich zwei Außenwände besitzt, die an die Umgebungsluft grenzen: Die Wände in Richtung Süden und Westen. Die anderen beiden Wände, sowie die Decke und der Boden, grenzen an weitere Gebäudeteile des K Gebäudes. Somit entspricht der Raum im Modell nach wie vor einem geschlossenen System und bildet weiterhin, zusammen mit dem umgebenden K Gebäude und der Umgebungsluft, ein abgeschlossenes System. Jedoch müssen nun potenziell verschiedene Wärmeströme zwischen dem Raum und der Außenumgebung sowie dem Raum und dem K Gebäude betrachtet werden. Da die fließenden Wärmeströme im Vergleich zur sehr großen Energie innerhalb des gesamten K Gebäudes und der Umgebung nur verschwindend gering sind, wird der erwärmende beziehungsweise kühlende Effekt der Wärmeströme auf die beiden Teilsysteme vernachlässigt und es wird von konstanten, homogenen Temperaturen beider ausgegangen.

Durch diese Erweiterung des Modells hängt die Raumtemperatur nun von zwei Wärmeströmen und damit indirekt von zwei externen Steuergrößen, den Temperaturen in der Umgebung und im K Gebäude, ab. Um die Wärmeströme separat berechnen zu können, wird die gesamte Oberfläche zum Wärmeaustausch aufgeteilt in die Austauschoberfläche mit der Umgebung und die Austauschoberfläche mit dem K K GebäudeGebäude. Des Weiteren werden im Modell die Temperatur der Außenumgebung und die Temperatur innerhalb des K Gebäudes als externe Steuergrößen berücksichtigt. Das Gleichungssystem des Grundmodells in 4.1 erweitert sich also um folgende Änderungen:

```

1 equation
2   [...]
3   /* calculate surface of heat exchange with the environment */
4   environment_surface = room_length * room_height + room_breadth * room_height;
5   /* calculate surface of heat exchange with the remaining building */
6   building_surface = 2 * (room_length * room_breadth) + room_length * room_height +
7                      room_breadth * room_height;
8   /* calculate derivative of the inner energy */
9   der(room_u) = environment_qdot + building_qdot;
10  /* calculate heatflow between room and environment */
11  environment_qdot = u_wall * environment_surface * (environment_temperature -
12                     room_temperature);
13  /* calculate heatflow between room and building */
14  building_qdot = u_wall * building_surface * (building_temperature - room_temperature);

```

Listing 4.2: Erweitertes Gleichungssystem Modell des Raumes unter Berücksichtigung der realen Umgebung in Modelica

Damit wurde das Raummodell an die reale Umgebung angepasst und um die Temperatur innerhalb des Raumes zu bestimmen, wird nun neben der Ausgangstemperatur im Raum und der Umgebungstemperatur noch die Temperatur innerhalb des restlichen K Gebäudes berücksichtigt. Im nächsten Schritt werden die realen, räumlichen Gegebenheiten im Modell abgebildet.

4.1.4 Modellerweiterung durch Berücksichtigung der räumlichen Gegebenheiten

Um das Modell an die realen Gegebenheiten des Raumes K004b anzupassen, müssen zwei bauliche Gegebenheiten beachtet werden. Wie in Abb. 3.1 bereits dargestellt ist, ist in der südlichen Außenwand eine Fensterfront vorhanden. Da der U-Wert eines Fensters erheblich von dem U-Wert einer Wand abweicht, entsteht ein zusätzlicher Wärmestrom zwischen dem Raum und der Umgebung durch das Fenster hindurch. Das Öffnen und Schließen der Fenster mit daraus resultierenden Massenströmen wird zunächst nicht explizit berücksichtigt, weshalb das Raummodell weiterhin als geschlossenes System betrachtet wird. Des Weiteren ist es möglich, den Raum über einen Heizkörper zu beheizen. Mit dem Heizkörper, der zunächst als einfache Wärmequelle im Modell ergänzt wird, erhöht sich die Anzahl der externen Steuergrößen erneut, da die Temperatur innerhalb des Raumes auch von dieser abhängig ist.

Durch diese Erweiterungen werden auch weitere physikalische Größen zur Beschreibung der Eigenschaften des Raummodells benötigt. Wie bereits erwähnt werden die Eigenschaften um den U-Wert eines Fensters, sowie die Breite und Höhe der Fensterfront ergänzt, wie in Tabelle Tab. 4.2 zusammengefasst.

Modellrelevante Eigenschaften	Wert	Einheit
Fensterbreite	7,0 ¹⁾	[m]
Fensterhöhe	2,08 ¹⁾	[m]
Wärmedurchgangskoeffizient Glas	2,0 ²⁾	[$\frac{W}{m^2 \cdot K}$]

¹⁾Werte durch eigene Vermessung des Raumes K004b vom 07.12.2015.

²⁾Tabellenwert, geschätzt nach [Hauser, 2000, S. 270ff.].

Tab. 4.2: Weitere Eigenschaften des Raummodells

Durch diese Anpassung verändert sich die Austauschoberfläche mit der Umgebung, die sich nun auf zwei Flächen mit verschiedenen Wärmedurchgangskoeffizienten verteilt. Des Weiteren wird eine Wärmequelle für die Heizung ergänzt, so dass sich folgende Änderungen des Gleichungssystems im Vergleich zum bisherigen Modell in 4.1 ergeben:

```

equation
2   [...]
3   /* calculate surface of heat exchange with the environment */
4   environment_surface = room_length * room_height + room_breadth * room_height -
      window_surface;
5   /* calculate surface of heat exchange with the remaining building */
6   building_surface = 2 * (room_length * room_breadth) + room_length * room_height +
      room_breadth * room_height;
7   /* calculate surface of window with the environment */
8   window_surface=(window_length*window_height);
9   /* calculate derivative of the inner energy */
10  der(room_u)=environment_qdot + building_qdot + window_qdot + radiator_qdot;
    /* calculate heatflow between room and environment through the walls */

```

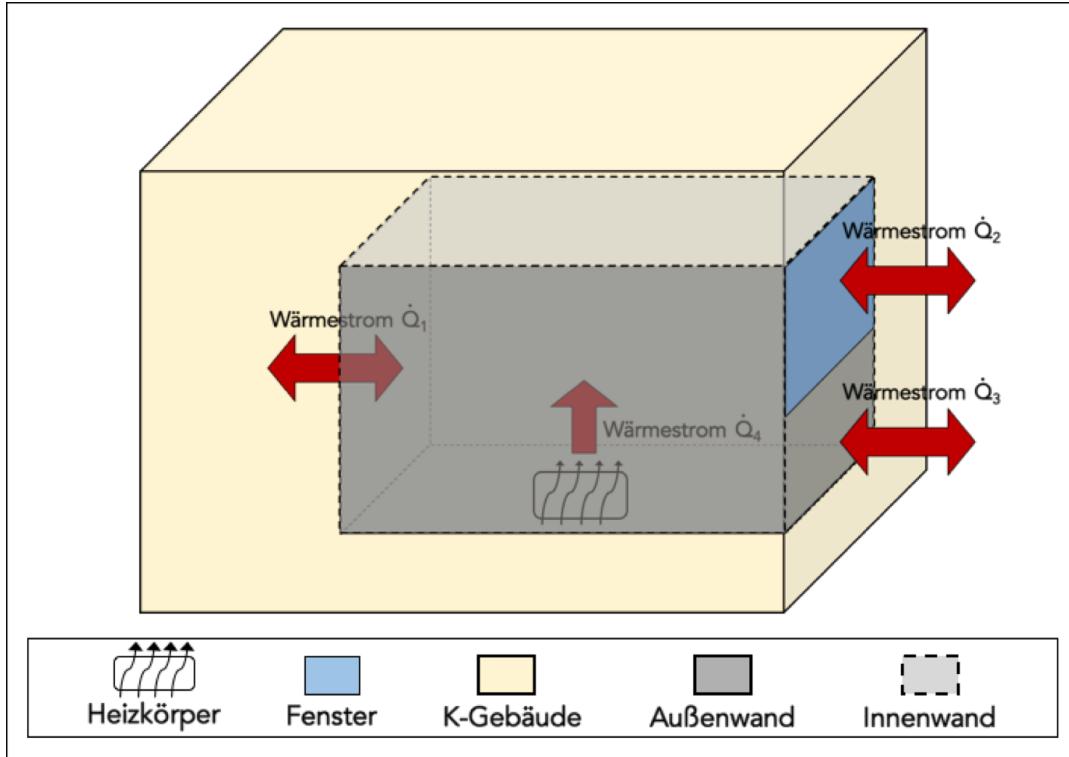


Abb. 4.2: Erweitertes Raummodell

```

12   environment_qdot = u_wall * environment_surface * (environment_temperature -
           room_temperature);
13   /* calculate heatflow between room and environment through the window */
14   building_qdot = u_glass * window_surface * (environment_temperature - room_temperature);
15   /* calculate heatflow between room and building */
16   building_qdot = u_wall * building_surface * (building_temperature - room_temperature);

```

Listing 4.3: Erweitertes Gleichungssystem des Raumes unter Berücksichtigung der räumlichen Gegebenheiten in Modelica

Damit ist das Modell auch an die räumlichen Gegebenheiten angepasst und beschreibt dadurch die realen Zusammenhänge in groben Zügen. Die bisherigen Zusammenhänge des Modells sind in Abb. 4.2 graphisch dargestellt. Allerdings kann ein Controller die Heizung nicht beliebig als einfache Wärmequelle einsetzen. Daher wird im folgenden Abschnitt ein detaillierteres Modell des Heizkörpers gebildet, um die Steuerung für den Controller zu ermöglichen.

4.1.5 Das Heizkörpermodell

Der Heizkörper im Raum K004b lässt sich nach [Recknagel, 2013, S. 824f.] als Stahlrohradiator identifizieren. Er besteht aus genormten zwei-säuligen Gliedern, die jeweils am Sammler oben und unten miteinander verschweißt den Heizkörper bilden. Dabei sind die Temperatur des Heizwassers am Einlass des Heizkörpers und der Massenstrom wiederum Steuergrößen für das Modell. Die Temperatur am Einlass wird durch die Heizanlage vorgegeben und ist damit eine externe Steuergröße. Der Massenstrom wird direkt durch den Stellantrieb gesteuert und ist daher eine Steuergröße, die vom Control-

ler mittelbar genutzt werden kann, um aktiv einen Einfluss auf die Raumtemperatur zu nehmen. Da der eingebrachte Wärmestrom nicht linear von der Temperaturdifferenz abhängt, erfolgt eine Diskretisierung des Heizkörpers in Volumenelemente, um den Heizkörper adäquat im Modell abzubilden.

Zur Berechnung des in den Raum eingebrachten Wärmestroms, erfolgt wiederum eine Bilanzierung von jedem diskreten Volumenelement durch den ersten Hauptsatz der Thermodynamik nach Gl. 4. Der gesamte Wärmestrom berechnet sich dann aus der Summe der einzelnen Wärmestrome zwischen Raum und den Volumenelementen, welche sich nach Gl. 5 berechnen. Bei der Diskretisierung wird mit Hilfe der Heizwassertemperatur am Einlass des Volumenelements die Temperatur am Auslass berechnet und zusammen mit dem Massenstrom an das nächste Volumenelement weitergereicht. Allerdings müssen für die Berechnung zunächst noch weitere Eigenschaften in das Raummodell mit aufgenommen werden, welche in Tab. 4.3 zusammen mit ihren Werten dargestellt sind.

Modellrelevante Eigenschaften	Wert	Einheit
Zahl der Glieder	106 ¹⁾	Stück
Säulen pro Glied	2 ¹⁾	Stück
Rohrdurchmesser Vertikal	0,0255 ¹⁾	[m]
Rohrdurchmesser Horizontal	0,05 ²⁾	[m]
Höhe der Glieder	0,4 ¹⁾	[m]
Länge des Glieds	0,045 ²⁾	[m]
Wassermasse innerhalb eines Gliedes	0,35 ²⁾	[kg]
Spezifische Wärmekapazität von Wasser	4.200,0 ⁴⁾	[$\frac{J}{kg*K}$]
Wärmedurchgangskoeffizient Heizkörper	16 ³⁾	[$\frac{W}{m^2*K}$]
Anzahl Volumenelemente	10	Stück

¹⁾Wert aus eigener Vermessung des Raumes K004b vom 07.12.2015.

²⁾Tabellenwert, entnommen aus [Recknagel, 2013, S. 825].

³⁾Tabellenwert, geschätzt nach [Recknagel, 2013, S. 191ff.].

⁴⁾Tabellenwert, entnommen aus [Baehr u. Kabelac, 2012, S. 619].

Tab. 4.3: Eigenschaften des Heizkörpermodells

Die realen Eigenschaften des Heizkörpers werden eins zu eins im Modell abgebildet und anschließend im Heizkörpermodell zentral auf die einzelnen diskretisierten Volumenelemente umgerechnet, wie in Listing 4.4 in Zeile zu sehen. Dazu gehört die Zahl und Wassermasse der Glieder, die Höhe des Heizkörpers sowie die Anzahl, Durchmesser und Längen der verschiedenen Rohre, um die Wärmeaustauschoberfläche zu bestimmen. Zudem werden die spezifische Wärmekapazität des Heizwassers und der Wärmedurchgangskoeffizient des Heizkörpers benötigt.

```

model CV_Radiator "control volume for a discretized radiator"
2 [...]
equation
4   /* calculate inner energy within one control volume*/

```

```

6   cv_u = cv_m * cp_water * cv_temperature_out;
7   /* calculate derival of the inner energy of one control volume*/
8   der(cv_u) = mdot * cp_water * (cv_temperature_in - cv_temperature_out) - cv_qdot;
9   /* calculate heatflowrate of the control volume*/
10  cv_qdot = u_radiator * exchange_surface_cv * (cv_temperature_out - room_temperature_cv
11      );
12  /* commit calculated temperature */
13  outlet.t = cv_temperature_out;
14 end CV_Radiator;

15 model Radiator "model for a discretized radiator within a room"
16 [...]
17 equation
18   /* calculate exchange surface of one control volume */
19   exchange_surface_cv = (radiator_element_number * radiator_element_length * 2 *
20     Modelica.Constants.pi * tube_diameter_horizontal + radiator_element_number *
21     radiator_tubes_element * tube_length_vertical * Modelica.Constants.pi *
22     tube_diameter_vertical)/cv_number;
23   /* calculate mass within one control volume */
24   cv_m = (radiator_elment_mass * radiator_element_number)/cv_number;
25   /* commit temperature of radiator fluid inlet to the first control volume */
26   cv_radiator [1]. inlet.t = radiator_inlet ;
27   /* connect the control volumes within the radiator */
28   for i in 1 : (cv_number-1) loop
29     connect( cv_radiator[i]. outlet, cv_radiator[i+1]. inlet );
30   end for;
31   /* calculate and commit the heatflow which is leaving the radiator */
32   radiator_qdot = sum(cv_radiator.cv_qdot);
33 end Radiator;

```

Listing 4.4: Auszüge des Modelica Modells vom Heizkörper in K004b

Das Modell des Heizkörpers ist als Subkomponente in den Raum integriert und die Steuergrößen werden über einen eigenen Port am Raummodell an den Heizkörper übergeben. Mit dem Modell des Heizkörpers wird ein physikalisch motivierter, mittelbarer Zusammenhang zwischen der Heizleistung und dem Massenstrom hergestellt, welcher insbesondere von einem modellprädiktiv regelnden Controller genutzt werden kann.

Bereits bei den Einsatzzielen der Anlage in Abschnitt 3.1 war gefordert, den Zusammenhang zwischen der Sonneneinstrahlung und der Raumtemperatur zu untersuchen sowie Störgrößen explizit in Kauf zu nehmen. Daher ist es passend, dass die Fensterfront in Richtung Süden ausgerichtet ist und der Raum K004b als Büro genutzt wird. Um jedoch das Modell darauf anzupassen, erfolgt im nächsten Abschnitt erfolgt weitere Erweiterung des Modells.

4.1.6 Modellerweiterung durch Berücksichtigung der Sonneneinstrahlung und Störgrößen

Der Raum K004b wird regulär als Büro genutzt, weshalb verschiedene Faktoren als Störgrößen in Bezug auf die Raumtemperatur betrachtet werden können. Zum einen wird durch die Menschen und deren Rechner weitere Wärme in den Raum eingebracht und zum Anderen werden die Fenster und die Türen geöffnet und geschlossen. Dabei werden Massenströme zwischen Raum und Außenumgebung sowie K Gebäude

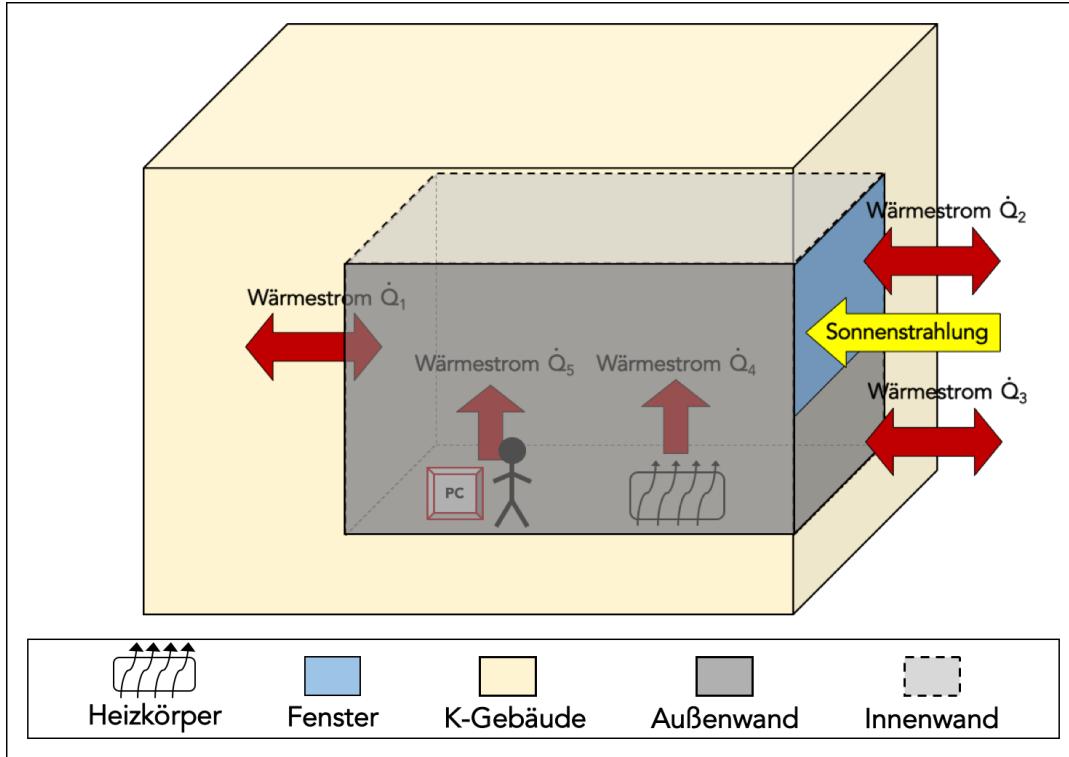


Abb. 4.3: Erweitertes Raummodell

ausgetauscht, welche jedoch zunächst nicht modelliert werden sollen, sondern als Störgröße betrachtet die es durch die Modellprädiktive Regelung zu auszugleichen gilt. Die Wärme von Rechnern und Menschen werden als einfache Wärmequelle im Modell berücksichtigt. Zudem trifft auf die südseitigen Fenster Sonnenstrahlung, welche ebenfalls einen Wärmestrom in den Raum einbringt und damit ebenfalls einen Einfluss auf die Raumtemperatur hat. Dieser wird zunächst ebenfalls als eine einfache Wärmequelle aufgefasst. Damit ergeben sich zwei weitere äußere Steuergrößen, die von der Anlage beziehungsweise dem Modell nicht beeinflusst werden können. Damit erweitert sich das Modell wie folgt:

```

equation
2   [...]
3   /* calculate derivative of the inner energy */
4   der(room_u) = environment_qdot + building_qdot + window_qdot + radiator_qdot +
                 sun_qdot + otherfactors_qdot;

```

Listing 4.5: Erweitertes Gleichungssystem Modell des Raumes unter Berücksichtigung der Sonneneinstrahlung und Störgrößen

Damit lassen sich die Zusammenhänge des Modells wie in Abb. 4.3 visualisieren. Der Einfluss der Sonnenstrahlung wurde bereits in Kapitel 2.4.1.2 erläutert und muss nun an die Gegebenheiten des Raumes K004b angepasst werden. Als Messwerte stehen Werte der Globalstrahlung zur Verfügung, welche umgerechnet werden müssen in die effektive Strahlungsintensität an der Fenstersfront.

Da zur Berechnung der effektiven Strahlungsintensität neben der Globalstrahlung auch der Azimut und Höhenwinkel der Sonne bekannt sein muss, werden komplexe

Berechnungen nötig. Da die Sonnenstrahlung eine Steuergröße darstellt und um die Komplexität des Modells zu schonen, werden die Effektivwerte der Sonnenstrahlung von einem Python Skript berechnet und anschließend dem Modell übergeben. Zudem hat dies den Vorteil, dass im pysolar Package bereits sehr exakte Algorithmen implementiert sind. Das Programm ist in Listing 4.6 dargestellt. Zur Berechnung des Azimuth- und Sonnenhöhenwinkels wird der Längen- und Breitengrad des K-Gebäudes sowie die Höhe über dem Meeresspiegel benötigt. Um die effektive Solarstrahlung zu bestimmen wird zusätzlich die Ausrichtung der Fensterfläche benötigt. Die Werte sind in Listing 4.6 in Zeile sieben bis zehn zu finden und wurden mit Hilfe von Google Inc. ermittelt.

```

1  from pysolar.solar import *
2  from pysolar import *
3  import datetime
4  from math import cos, sin, radians
5  import numpy as np
6
7  latitude_deg = 49.01305
8  longitude_deg = 8.39207      # Negativ Richtung Westen gemessen von Greenwich, England aus
9  elevation = 116.0
10 beta = 8.0      # Ausrichtung der Fläche in Bodenebene. Norden ist Nullpunkt mit positiver
11   Richtung im Uhrzeigersinn
12
13 data = np.loadtxt("globalstrahlung.csv", delimiter = ",")
14 t_start = datetime.datetime(year,month,day,hour,minute)
15 timemeasure = np.asarray([t_start + datetime.timedelta(minutes=(distance_measurements*dt))
16                           for dt in range(data.size)])
17
18 qdotmeasure = data
19 qdotsun_effective_list = []
20 azimuth_list = []
21 altitude_list = []
22
23 for i in range(data.size):
24     azimuth = get_azimuth(latitude_deg, longitude_deg, timemeasure[i], elevation)
25     altitude = get_altitude(latitude_deg, longitude_deg, timemeasure[i], elevation)
26
27     if (altitude <= 0.0) or (beta - 270 <= azimuth <= beta - 90.0) or (altitude == 90.0):
28         # Sonne noch nicht aufgegangen
29         # oder Fläche verschattet
30         qdotsun_effective = 0.0
31
32     elif (qmeasure[i] > 0.0) and (0.0 < altitude < 5.0):
33         # Vermeide unrealistisch hohe Strahlungsintensitäten bei niedrigem Sonnenhöhenwinkel
34         qdotsun_effective = 10.0
35
36     else:
37         qdotsun_effective = qdotmeasure[i] * cos(radians(azimuth - beta)) * (cos(radians(
38             altitude))/sin(radians(altitude)))
39
40     azimuth_list.append(azimuth)
41     altitude_list.append(altitude)
42     qdotsun_effective_list.append(qdotsun_effective)
43
44 np.savetxt("qdotsun_effective.csv", np.c_[np.asarray(qdotsun_effective_list)], delimiter = ",")
```

Listing 4.6: Programm zur Umrechnung der Globalstrahlung in die effektive Solarstrahlung an der Fensterfront am Rum K004b

Außerdem wird der Startzeitpunkt und der Abstand zwischen den Messpunkten für die gemessenen Daten benötigt, da der Stand der Sonne von der Uhrzeit abhängig ist. Nach dem Einlesen der Messdaten aus der Datei globalstrahlung.csv wird anschließend für jeden einzelnen Messwert und -zeitpunkt der Azimuth- und Sonnenhöhenwinkel berechnet. Darauf basierend wird geprüft, ob die Sonne senkrecht am Himmel steht, noch nicht aufgegangen ist oder die Fläche verschattet ist. Ist dies der Fall trifft keine effektive Strahlung auf das Fenster. Des Weiteren würden sich für niedrige Sonnenstände sehr hohe und unrealistische effektive Strahlungswerte ergeben, weshalb diese für Sonnenstandswinkel kleiner als fünf Grad begrenzt werden. Ansonsten wird der effektive Strahlungswert nach REFXXXXgleichung berechnet und in der Datei qdotsun-effective gespeichert. Dieses Programm kann in leicht abgewandelter Form einfach zur Umrechnung einzelner Messwerte genutzt werden, indem das Einlesen und Speichern der umgerechneten Daten durch eine Abfrage eines Messwertes und Übergabe einer Variable mit umgerechnetem Messwert ersetzt wird.

Der effektive Strahlungswert wird allerdings durch den Transmissionsgrad des Fensters weiter abgeschwächt, der durch einen weiteren Parameter/ eine weitere Eigenschaft der *window_transmission* beschrieben wird. Der Schätzwert für eine zweischeibige Verglasung des Fensters stammt aus [Peter Häupl, 2013, S. 63]. Dadurch erweitert sich das Modell um ein Fenster als Subkomponente:

```

1 model Window
2   Modelica.SIunits.DensityOfHeatFlowRate qdotsun_effective;
3   Modelica.SIunits.HeatFlowRate qdot_effective;
4   parameter Real window_transmission = 0.5;
5   Modelica.SIunits.Area window_surface;
6   equation
7     qdot_effective = qdotsun_effective * window_transmission * window_surface;
8 end Window;
```

Listing 4.7: Fenster als Subkomponente des Raummodells

Die Modellkomplexität ist bis zu diesem Punkt noch überschaubar und weiterhin berücksichtigt das Modell die physikalischen Effekte mit dem größten bzw. mit relevanten Einfluss. Damit ist die Modellbildung abgeschlossen und das gesamte Modell des Raumes findet sich im Anhang B.1. Bei der Modellbildung wurde auf eine Programmierung mit klarer Struktur Wert gelegt. Daher sind die Variablen der Modelle strikt nach Parametern, Controls und Zuständen gegliedert/sortiert.

Das Schaltbild in Dymola, welches als Umgebung zur Modellbildung und Simulation genutzt wurde ist in REFXXXXX abgebildet. Darauf wird auch deutlich, dass das Modell 6 verschiedene Steuergrößen und damit Freiheitsgrade besitzt. Jedoch werden fünf davon durch äußere Bedingungen festgelegt, durch die Außentemperatur, die Temperatur im K Gebäude, die Temperatur am Einlass der Heizung, die anderen Faktoren sowie die Solarstrahlung. Damit kann der Controller lediglich den Massenstrom zur Beeinflussung der Raumtemperatur nutzen um auf Änderungen der anderen Steuergrößen zu reagieren.

FIG MOPDELICA

Weitere Effekte die explizit nicht berücksichtigt wurden sind die Energie innerhalb der Wände und des Mobiliars von Raum K004b, welche bei abrupter Abkühlung der Raumlufttemperatur einen stark erwärmenden Einfluss durch die Abstrahlung von Wärme, also Wärmekonvektion wie in Kapitel 2 beschrieben. Da sich eine solch starke Abkühlung der Raumluft jedoch über einen längeren Zeithorizont erstreckt, wird diese als Störgröße wahrgenommen die ein Modellprädiktiver Regler ausgleichen können sollte.

4.2 Simulation und Modellanpassung

In diesem Abschnitt erfolgt zunächst die Simulation der Raumtemperatur unter Verwendung des Modells. Dazu werden für die Steuergrößen reale Messwerte vorgegeben, um die Güte des Modells abzuschätzen zu können, durch einen Vergleich der berechneten und tatsächlichen gemessenen Werte. Abschließend erfolgt eine Anpassung der Modellparameter mit Hilfe von Bürger [2016], um die Modellgüte zu verbessern.

4.2.1 Simulation und Validierung des Modells

Die Validation des Modells wird in zwei Schritte aufgeteilt. Zunächst wird das Raummodell ohne Verwendung des Heizkörpers simuliert, um das Grundmodell des Raumes zu überprüfen und ein Gefühl dafür zu bekommen ob alle wichtigen physikalischen Effekte berücksichtigt wurden. Im nächsten Schritt wird erfolgt eine Simulation, bei der der Heizkörper zum Einsatz kommt, die dazu dienen soll die Güte der Abbildung des Heizkörpers zu bestimmen. Somit gilt es für den ersten Schritt Zeitintervall zu identifizieren, welches möglichst langer Dauer ist und währenddessen möglichst wenig Störfaktoren aufgetreten sind und weiterhin der Heizkörper nicht genutzt wurde. Ein solches Zeitintervall findet sich über die Weihnachten vom 23.12.2015 bis zum 28.12.2015, da aufgrund der Feiertage das Büro nicht genutzt wurde. Das Intervall umfasst mehr als 5 Tage, von denen an den ersten Tage eine sehr milde Außentemperatur zwischen 8°C und 16°C vorgeherrscht hat. Erst am letzten Tag nähert sich die Außenlufttemperatur dem Gefrierpunkt von 0°C . Außerdem hat die Sonne gegen mittag geschienen und den Raum dadurch erwärmt. Die genauen Messdaten in 10 minütigen Intervallen der Globalstrahlung und Außentemperatur für diesen Zeitraum wurden von Mühr [2016] zur Verfügung gestellt und sind in REFXXXXBILD visualisiert sowie auf der angehängten CD in der Datei XXXXXX zu finden. Der Simulation wurde außerdem eine Temperatur von 22°C im K Gebäude und die gemessene Initialtemperatur von $22,8^{\circ}\text{C}$ zugrunde gelegt. Die Ergebnisse der Simulation sind im Plot in Abbildung Abb. 4.4 abgebildet und werden im Folgenden analysiert.

Zu beachten ist die Skalierung der Abzisse, welche in Minutenschritte eingeteilt ist und die Simulation erstreckt über einen Zeitraum von 7500 Minuten. Im Rahmen der Modellprädiktiven Regelung werden die Simulationszeiträume deutlich kürzer betrachtet, jedoch soll durch diese Simulation ein Abgleich des Modells mit der Realität

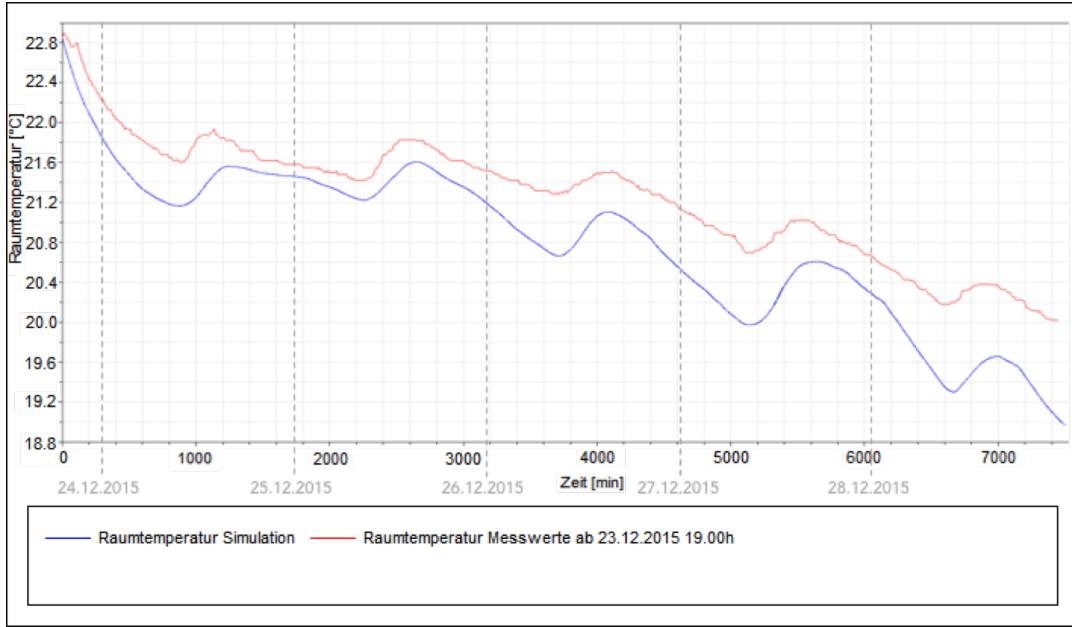


Abb. 4.4: Simulation des Raummodells ohne Einsatz der Heizung

ermöglicht werden, worin sich die lange Periode begründet.

Der erste Blick lässt bereits erkennen, dass die simulierte Temperaturkurve eine ähnliche Dynamik wie der gemessene Temperaturverlauf aufweist. Von Beginn der Simulation bis hin zum 26.02.2016 stimmt das Systemverhalten sehr gut mit der Realität überein. Danach erfolgt ein leichter Knick bei der simulierten Kurve, der bis Ende der Simulation erhalten bleibt und nicht weiter erklärt werden kann. Außerdem ist eine minimale zeitliche Verzögerung des Modells zur Realität erkennbar, was auf einen Unterschied in der Trägheit zwischen Realität und Modell hindeutet. Die simulierte Temperaturkurve liegt während des gesamten Intervalls unterhalb der Messwerte, was einen Hinweis auf einen zu hohen Wärmeverlust im Modell liefert. Des Weiteren lässt sich der Einfluss der Solarstrahlung auf die Raumtemperatur deutlich erkennen, der täglich etwa zur Mittagszeit einsetzt und den Raum dadurch bis Nachmittags erwärmt.

Insgesamt ist zu hervorheben, dass die simulierte von der tatsächlich gemessenen Temperatur innerhalb der ersten beiden Tag um nicht mehr als $0,4^{\circ}\text{C}$ und während der gesamten Simulationsdauer um nicht mehr als 1°C abweicht. Dabei besitzen die Temperatursensoren des WEBTHERMOGRAPH8X eine Messabweichung von $\pm 0,26^{\circ}\text{C}$ und die beiden RTM1 Temperaturfühler eine Messabweichung von $\pm 0,5^{\circ}\text{C}$. Außerdem haben die Messwerte der Wetterstation auf dem Physikhochhaus des Karlsruher Instituts für Technologie eine 10-minütige Auflösung, wohingegen die Raumtemperatur minütlich gemessen wurde. Durch die örtliche Trennung von 1km Luftlinie zwischen der Messstation und dem K Gebäude, ist es durchaus in Betracht zu ziehen, dass Messwerte durch eine Bewölkung die Steuergröße Solarstrahlung am Raum nicht vollständig, also mit fehlern behaftet beschreibt.

Im Rahmen der Modellprädiktiven Regelung werden jedoch wie zuvor erwähnt kürzere

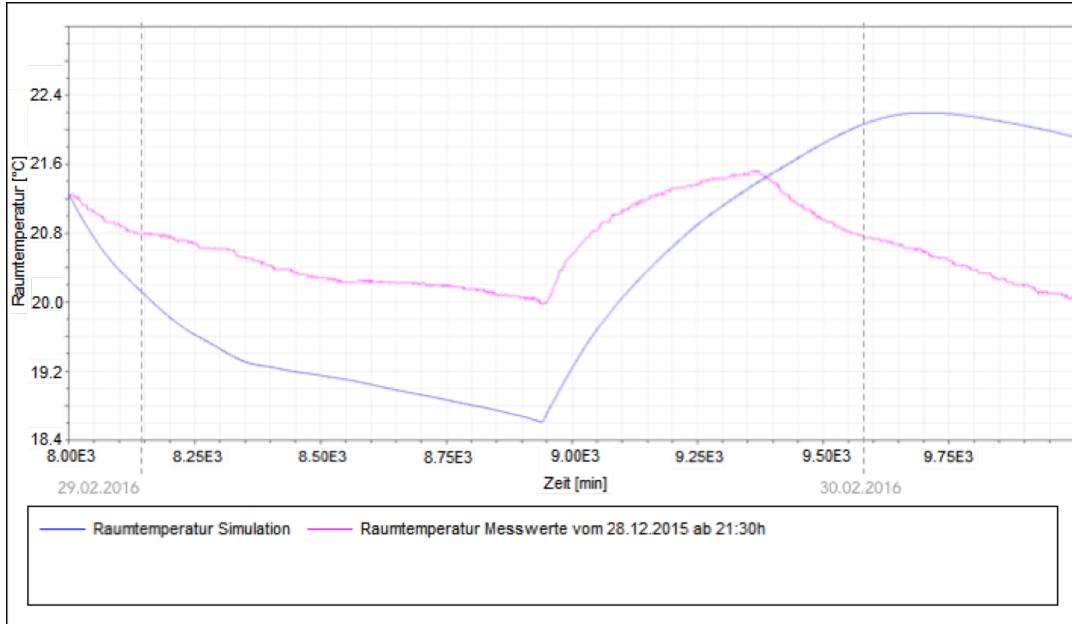


Abb. 4.5: Simulation des Raummodells mit Einsatz des Heizkörpers

Simulationszeiträume im unteren Stunden beziehungsweise höheren Minutenbereich betrachtet, weshalb die Güte des Modells ohne Einsatz des Heizkörpers dazu ausreichend ist.

Im nächsten Schritt findet eine Simulation mit Einsatz des Heizkörpers statt, um das vollständige Modell mit der Realität abzulegen. Der Heizkörper wurde für die folgende Modellsimulation in vier Volumenelemente diskretisiert. Für diese Simulation wurde ein Zeitintervall gesucht, indem die Raumtemperatur unter die untere Schalttemperatur von 20°C fällt, damit der Controller den Heizkörper zum Erwärmen des Raumes bis zum oberen Schaltpunkt von $21,5^{\circ}\text{C}$ nutzt und der Raum nach schließen des Heizkörperventils wieder abkühlt. Ein passendes Intervall findet sich vom 28.12.2015 bis zum 30.12.2015, dass auch an diesen Tagen das Büro nicht genutzt wurde. Die Messdaten für die Globalstrahlung und Außenlufttemperatur wurden auch hier in 10 minütigen Intervallen von Mühr [2016] zur Verfügung gestellt und sind auf der angehängten CD in der Datei XXX zu finden. Die Sonne hat gegen Nachmittag des 29.12.2016 leicht geschienen und die Außenlufttemperatur hat sich zwischen 6 Grad leicht unter den Gefrierpunkt bewegt. Die Heizkörper wird gegen Nachmittag des 29.12.2015 bis gegen Abend genutzt. Auch bei dieser Simulation wurde eine Temperatur von 22°C im K Gebäude und die gemessene Initialtemperatur von $21,25^{\circ}\text{C}$ zugrunde gelegt. Die Simulationsergebnisse sind im Plot Abb. 4.5 abgebildet und werden wiederum im Folgenden analysiert.

Der Simulationszeitraum beginnt gegen späten Abend am 28.02.2015 und erstreckt sich über 2000 Minuten bis zum frühen Morgen des 30.12.2015. Die Simulationsergebnisse bestätigen zunächst die vorherigen Ergebnisse, indem das Modell zunächst schneller ausköhlt als der reale Raum. Dies liefert wiederum einen Hinweis auf einen zu hohen Wärmeverlust im Modell. Die Systemdynamik des Modells folgt in etwa der Realität

bis zur Nutzung des Heizkörpers nach etwa 950 simulierten Minuten. Das Öffnen des Heizkörperventils ist sowohl im Modell, als auch in den Messwerten sehr gut zu sehen und führt zu einem signifikanten Anstieg der Raumtemperatur. Es wird zudem deutlich, dass die Modellreaktion zum einen gleichzeitig einsetzt, jedoch deutlich langsamer/träger abläuft als in der Realität. Zum anderen wird im Modell weitaus mehr Wärme in den Raum eingebracht als in der Realität, was auf eine erhöhte Wärmezufuhr durch den Heizkörper oder durch die Solarstrahlung hinweist, da wie zuvor erwähnt die lokale Bewölkung für Fehler sorgen könnte. Die Trägheit des Modells kann auf eine unzureichende Diskretisierung hinweisen, weshalb die Anzahl der Volumenelemente auf 10 erhöht wurde. Die anschließenden Untersuchungen zeigen, dass die Modellgüte damit gesteigert werden konnte.

Insbesondere im Zusammenhang mit den zuvor festgestellten Unsicherheiten bei den Messwerten lässt sich feststellen, dass die Simulationskurve die Dynamik des realen Systems erkennbar widerspiegelt, wenn auch mit einer gewissen Trägheit verbunden durch die Nutzung der Heizung.

Zusammenfassend lässt sich feststellen, dass das Modell die grundlegende Systemdynamik der Realität beschreibt, jedoch noch gewisse Abweichungen im Modell vorhanden sind. Dies wurde insbesondere durch die sehr langen Simulationsintervalle deutlich, die für den späteren Einsatz des Modells nicht von weiterer Relevanz sind.

Um die Modellgüte insbesondere hinsichtlich des Einsatzes der Heizung zu verbessern, wird im Folgenden eine Parameterschätzung vorgenommen und abschließend überprüft, ob das Modell den Anforderungen für eine Modellprädiktive Regelung mit JModelica.org genügt.

4.2.2 Anpassung des Modells

Ziel des Abschnittes ist eine Verbesserung des Raummodells, welche durch eine Parameterschätzung mit Bürger [2016], einer freien Softwareumgebung für die optimale Versuchsplanung und Parameterschätzung. Wie bereits erwähnt, werden beim Einsatz des Modells mit Modellprädiktiver Regelung kürzere Intervalle betrachtet, weshalb die Modelloptimierung anhand von kürzeren Intervallen erfolgt. Die Durchführung der Parameterschätzung erfolgte mit freundlicher Unterstützung von Herrn ADRIAN BÜRGER, der das Übersetzen des Modell und die Ausführung übernommen hat.

Wie bereits zuvor, erfolgt auch die Parameterschätzung systematisch, um die einzelnen Modellparameter anzupassen. Als zu schätzende Parameter wird der Wärmedurchgangskoeffizient der Raumwände und des Heizkörpers sowie der Transmissionsgrad der Fensterscheiben betrachtet, welche im Folgenden in drei Schritten berechnet werden. Der Wärmedurchgangskoeffizient der Glasscheibe wird als fester Wert angenommen, da sich aufgrund der offensichtlichen Korrelation zwischen den beiden Wärmedurchgangskoeffizienten voneinander nicht-physikalische Schätzwerte ergeben können.

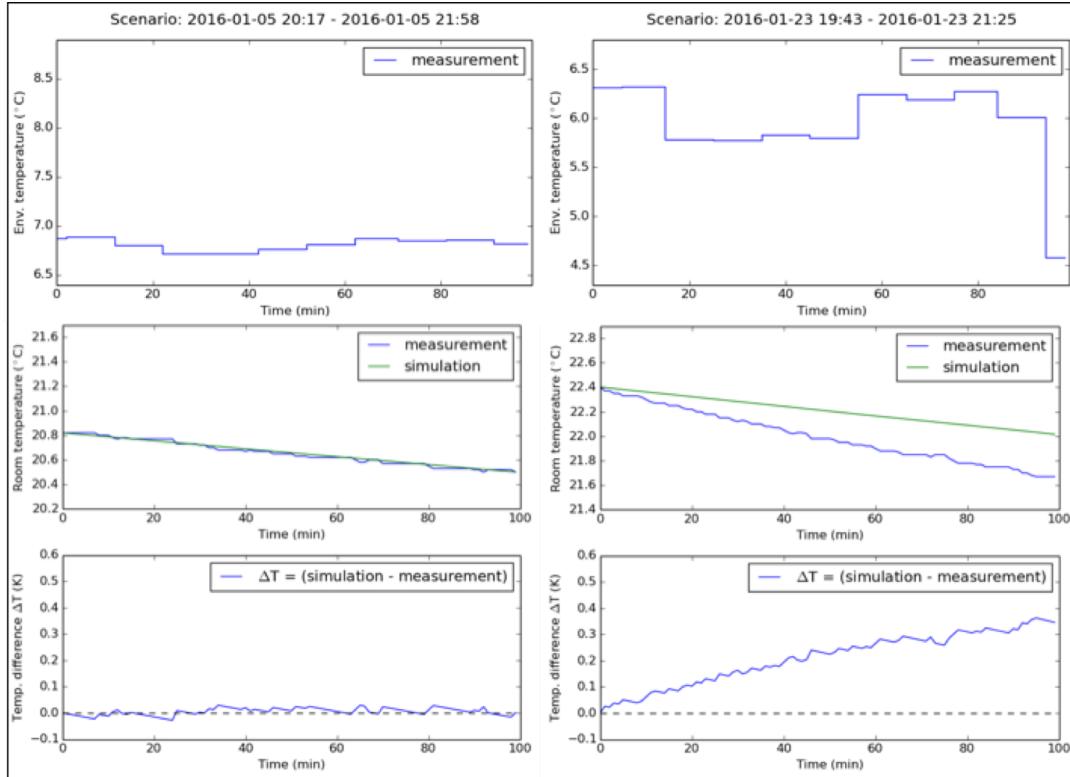


Abb. 4.6: Parameterschätzung des Wärmedurchgangskoeffizienten der Wand von zwei Intervallen

Es muss weiterhin beachtet werden, dass die Erhöhung der Modellgüte durch den Einsatz realer Messwerte ebenfalls mit einer Abweichung der physikalischen Schätzparameter zur Realität einhergehen können, da Störgrößen und Messfehler implizit berücksichtigt werden.

Zunächst wird der Wärmedurchgangskoeffizient der Wand bestimmt, wozu Intervalle genutzt wurden die möglichst ohne Störgrößen sind und ohne Einsatz der Heizung auskommen. Dazu bieten sich Zeitintervalle am Abend an, an denen das Büro nicht mehr genutzt wird und keine Solarstrahlung mehr auf die Fenster trifft. Das Ergebnis der Parameterschätzung für zwei Intervalle ist in den Plots in Abb. 4.6 dargestellt. Das Skript zur Parameterschätzung findet sich im Anhang A in der Datei *room_{pe}step1night.py*.

Die Plots auf der linken Seite stellen das Ergebnis für ein 100-minütiges Intervall vom Abend des 05.01.2016, auf der rechten Seite vom Abend des 23.01.2016 dar. Mit dem vorab fixierten U-Wert für Glas ergab sich ein Schätzwert für den Wärmedurchgangskoeffizient der Wand von $U_{wall} = 0.612986$. Die oberen beiden Plots enthalten die Verläufe der Steuergröße Außenlufttemperatur im Intervall, wobei die Solarstrahlung und die Heizung keinen Einfluss hatten. Die genauen Messdaten sind den Dateien auf der angehängten CD in A im Ordner enthalten. Die mittleren Plots zeigen einen Vergleich der gemessenen Raumtemperatur mit der Simulation des geschätzten Intervalls über 100 Minuten. Darin ist gut zu erkennen, dass die Dynamik des Modells durch eine Anpassung des Parameters weiter verbessert werden konnte. Die unteren Plots zeigen die Abweichung zwischen Simulation und Messwerten. Im linken Intervall beschreibt die Simulation bis auf minimale Abweichungen das reale System sehr genau,

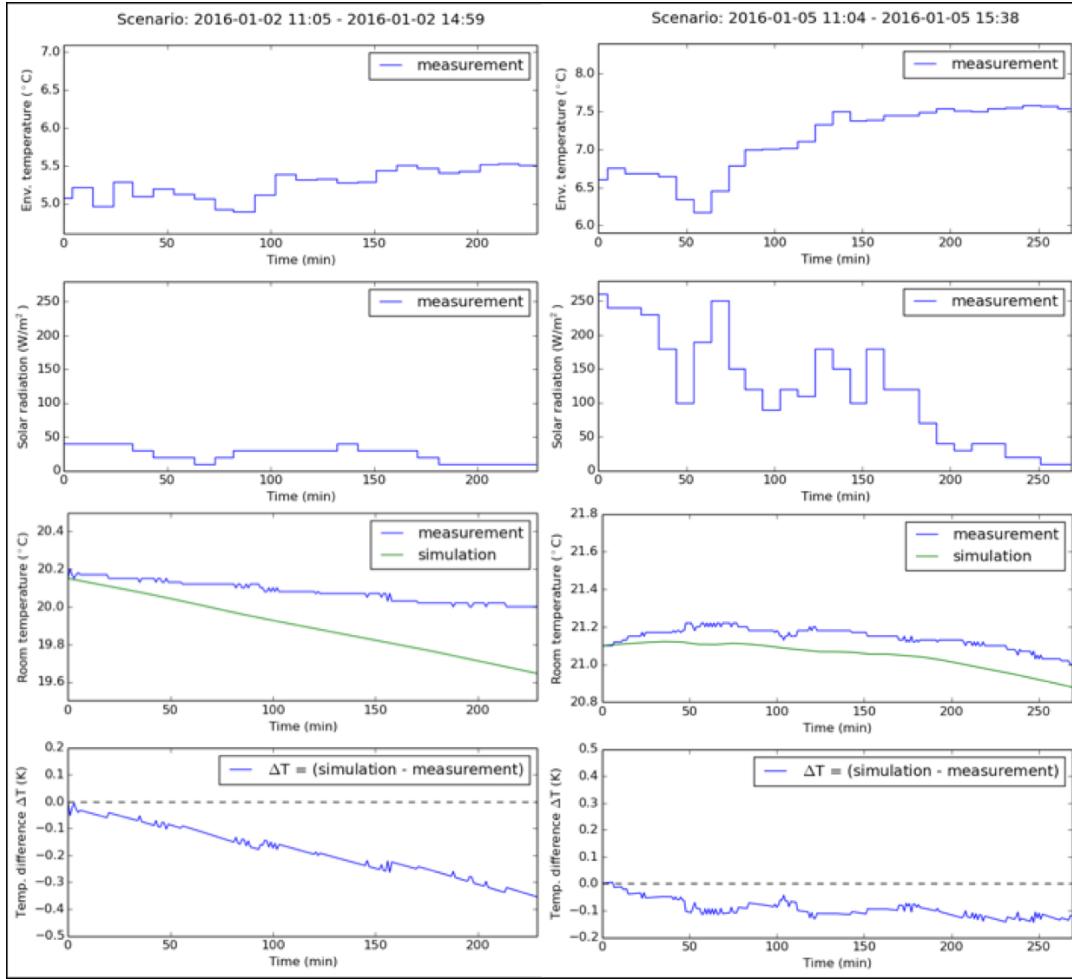


Abb. 4.7: Parameterschätzung

im Rechten ergibt sich bei der Simulation ein leicht erhöhte Temperatur im Vergleich zur Messung, die mit einer Abweichung von etwa $0,35^{\circ}\text{C}$ nach 100 Minuten immer noch eine ausreichende Güte besitzt.

Im nächsten Schritt wird der Transmissionsgrad der Fensterscheiben geschätzt. Dazu wurden wiederum Intervalle identifiziert, an denen die Sonne geschienen hat und die Störgrößen und den Einsatz der Heizung möglichst ausschließen. Dazu bieten sich zwei Intervalle Anfang Januar an, am 02.01.2016 und 05.01.2016 gegen die Mittagszeit, an denen das Büro nur teilweise genutzt wurde. Die Ergebnisse der Schätzung für die knapp 250 Minuten umfassenden Intervalle sind in Abb. 4.7 zusammengefasst. Das Skript zur Parameterschätzung findet sich im Anhang A in der Datei *room_pstep2day.py*.

window-transmission = 0.00687213

Einen besseren Vergleich der Modellgüte ermöglicht die erneute Simulation des ersten Intervalls ohne Heizkörper. Die Ergebnisse sind in Abb. 4.8 dargestellt.

Abgesehen von einem Fehler bei der Initialisierung der Raumtemperatur zu Beginn der Simulation beschreibt das Modell ohne Einsatz der Heizung ist eindeutig zu erkennen, dass es das Verhalten des realen Systems sehr gut beschreibt.

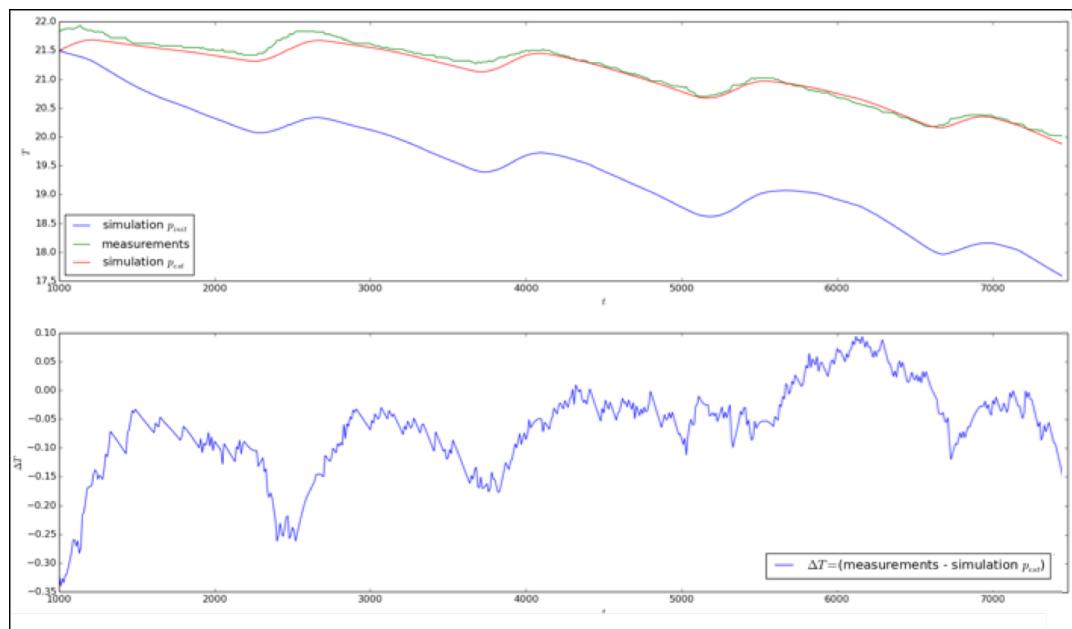
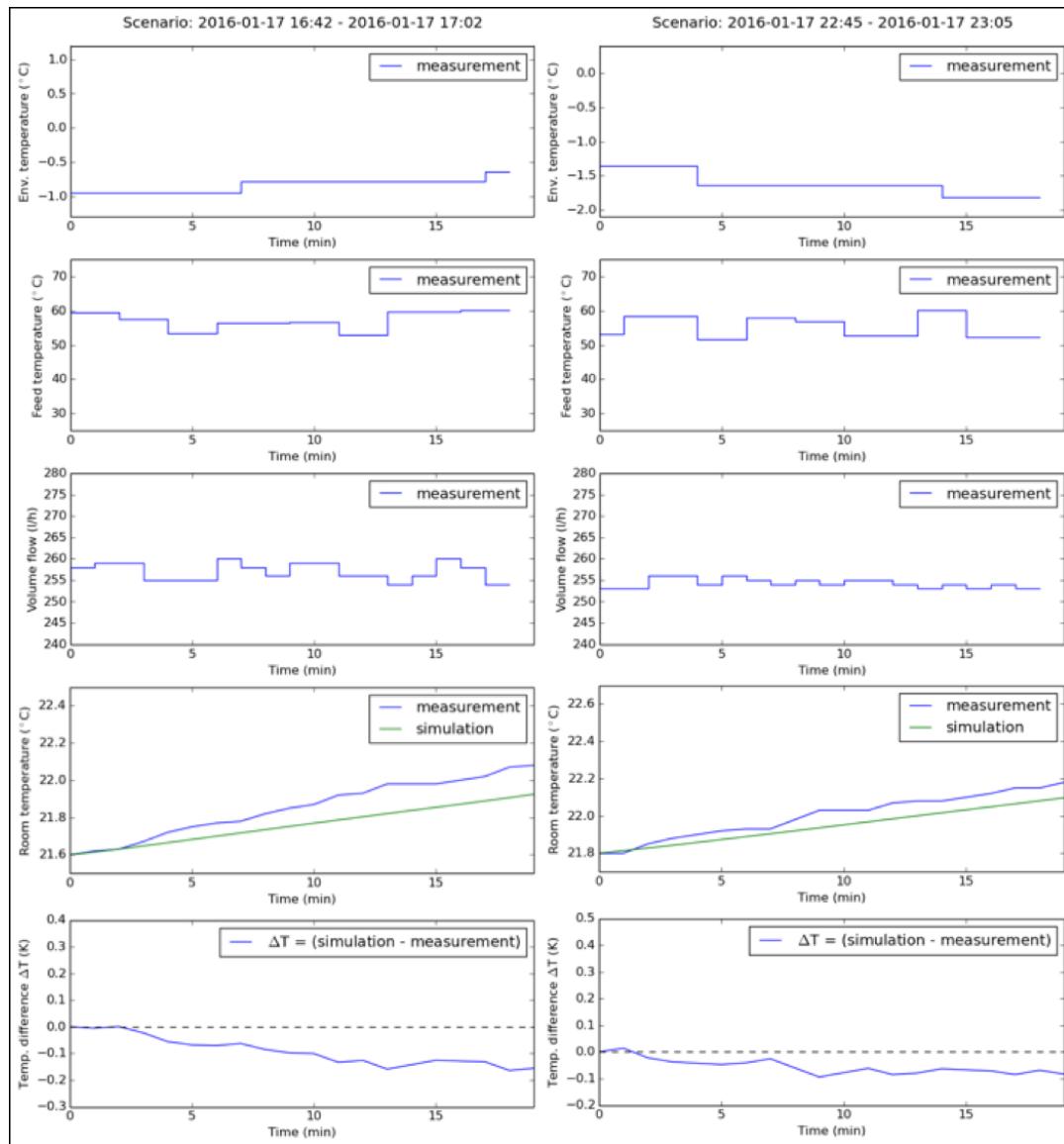


Abb. 4.8: Simulation des Raummodells ohne Einsatz der Heizung mit geschätztem Parameter

step3 schätzung mit heizung und teilweise störgrößen

u-radiator = 12.9872

JMODELICA fähig beweisen, also nächster Schritt MPC implementierung prüfen.

**Abb. 4.9:** Parameterschätzung

„Knowing where things are, and why, is essential to rational decision making.“
— JACK DANGERMOND, Esri

5 Schlussbetrachtung

5.1 Fazit

5.2 Ausblick und Ansatzpunkte für weitere Arbeiten

Welche Art der Verwendung? MPC mit JModelica.org also deren mpc Klasse eigene in casadi etc?

Vergleich mit bestehenden Modellen möglich z.B. das THERAKLES Modell

Annahme Temperatur und homogen im Raum untersuchen

Sonnenstarhlung genauer untersuchen und für MPc auf Vorhersagewerte gehen. Implementierung MPC, Länge Intervall, Kostenfunktionen versch. Kriterien untersuchen

A Modelle, Programme, Messdaten

Auf der beiliegenden CD befinden sich:

- die Modelle
- Skripte zur Parameterschätzung
- Messdaten

B Modelle

B.1 Raummodell

```

package room_model_backup
2

4   connector Temperature
5     Modelica.SIunits.Conversions.NonSIunits.Temperature_degC t;
6   end Temperature;

8
9   connector HeatFlow
10    Modelica.SIunits.HeatFlowRate qdot;
11  end HeatFlow;

12

14  model CV_Radiator "control volume for a discretized radiator"
15    /* parameter */
16    /* central parameter */
17    outer parameter Modelica.SIunits.CoefficientOfHeatTransfer u_radiator
18      "heat transfer coefficient of the radiator";
19    outer parameter Modelica.SIunits.SpecificHeatCapacity cp_water
20      "specific heat capacity of water";
21    outer Modelica.SIunits.Mass cv_m "mass within one control volume";
22    outer Modelica.SIunits.Area exchange_surface
23      "surface of one control volume at which heat transfer takes place";
24    /* calculated parameter*/
25    Modelica.SIunits.Energy cv_u "inner energy of the control volume";
26    Modelica.SIunits.HeatFlowRate cv_qdot
27      "heatflowrate over the borders of the control volume";
28    Modelica.SIunits.Conversions.NonSIunits.Temperature_degC cv_temperature_out(start=21.2,
29      fixed=true)
30      "temperature of the fluid leaving the control volume";
31
32    /* states */
33    outer Modelica.SIunits.Conversions.NonSIunits.Temperature_degC room_temperature_cv
34      "temperature within the room";

36    /* controls */
37    Modelica.SIunits.Conversions.NonSIunits.Temperature_degC cv_temperature_in=inlet.t
38      "temperature of the fluid streaming in the control volume";
39    outer Modelica.SIunits.MassFlowRate mdot "massflowrate within the radiator";

40  equation
41    /* calculate inner energy */
42    cv_u = cv_m * cp_water * cv_temperature_out;
43    /* calculate derival of the inner energy */
44    der(cv_u) = mdot * cp_water * (cv_temperature_in - cv_temperature_out) - cv_qdot;
45    /* calculate heatflowrate */
46    cv_qdot = u_radiator * exchange_surface * (cv_temperature_out - room_temperature_cv);
47    /* commit calculated temperature */
48    outlet.t = cv_temperature_out;
49  end CV_Radiator;
50

52  model Radiator "model for a discretized radiator within a room"
53
54    /* parameter */
55    /* parameter of the radiator */
56    parameter Real radiator_element_number=106 "number of normed elements of which the

```

```

radiator consists";
parameter Real radiator_tubes_element=2 "number of parallel tubes in one element";
58 parameter Integer cv_number = 20 "number of control volumes in which the radiator is
discretized ";
parameter Modelica.Slunits.Length radiator_element_length=0.045 "length of one element
depending on type (Recknagel 2013/2014: Heizung und Klimatechnik S.815ff.)";
60 parameter Modelica.Slunits.Height tube_length_vertical=0.4 "height of one element
depending on type (Recknagel 2013/2014: Heizung und Klimatechnik S.815ff.)";
parameter Modelica.Slunits.Diameter tube_diameter_horizontal=0.05 "diameter of the
horizontal tubes depending on type (Recknagel 2013/2014: Heizung und Klimatechnik
S.815ff.)";
62 parameter Modelica.Slunits.Diameter tube_diameter_vertical=0.0255 "diameter of the
vertical tubes depending on type (Recknagel 2013/2014: Heizung und Klimatechnik
S.815ff.)";
parameter Modelica.Slunits.Density rho_water = 1000 "density of water";
64 parameter Modelica.Slunits.Mass radiator_element_mass=0.35 "mass within one element of the
radiator depending on type (Recknagel 2013/2014: Heizung und Klimatechnik S.815ff.)";

66 inner parameter Modelica.Slunits.CoefficientOfHeatTransfer u_radiator = 12.9872 heat
transfer coefficient of the radiator";
inner parameter Modelica.Slunits.SpecificHeatCapacity cp_water = 4182 "specific heat
capacity of water";
68 CV_Radiator[cv_number] cv_radiator "array of control volumes to discretize the radiator";

70 /* calculated parameter */
Modelica.Slunits.HeatFlowRate radiator_qdot_out
72   "heatflow which is leaving the radiator";
Modelica.Slunits.Conversions.NonSlunits.Temperature_degC radiator_temperature_out
74   "calculated (predicted) temperature of the water leaving the radiator";
//Modelica.Slunits.Volume cv_v "volume within one control volume";

76 /* parameter of the control volumes */
78 inner Modelica.Slunits.Mass cv_m "mass within one control volume";
inner Modelica.Slunits.Area exchange_surface
80   "surface of one control volume at which h";
82 /** states */
/* states of the radiator variable */
84 inner Modelica.Slunits.Conversions.NonSlunits.Temperature_degC room_temperature_cv "
temperature within the room for the control volume";
outer Modelica.Slunits.Conversions.NonSlunits.Temperature_degC room_temperature "
temperature within the room from the room";

86 /** controls */
88 inner Modelica.Slunits.MassFlowRate mdot=inlet.mdot "massflowrate within the radiator";
Modelica.Slunits.Conversions.NonSlunits.Temperature_degC radiator_inlet = inlet.t "
temperature of the inflowing fluid ";

90 equation
92   /* calculate surface of one control volume */
exchange_surface = (radiator_element_number * radiator_element_length * 2 *
Modelica.Constants.pi * tube_diameter_horizontal + radiator_element_number *
radiator_tubes_element * tube_length_vertical * Modelica.Constants.pi *
tube_diameter_vertical)/cv_number;
94   /* calculate mass within one control volume */
cv_m = (radiator_element_mass * radiator_element_number)/cv_number;
96   /* calculate volume within one control volume */
//cv_v = (radiator_element_number * radiator_element_volume) / cv_number;
98   /* commit temperature of radiator fluid inlet to the first control volume */
cv_radiator [1]. inlet.t = radiator_inlet ;
100  /* commit fluid temperatures within the radiator control volumes */
for i in 1 : (cv_number-1) loop
102    connect( cv_radiator [i]. outlet, cv_radiator [i+1]. inlet );
end for ;

```

```

104  /* save fluid temperature of the last radiator control volumes */
105  radiator_temperature_out=cv_radiator[cv_number].outlet.t;
106  /* calculate and save the heatflowrate which is leaving the radiator */
107  radiator_qdot_out = sum(cv_radiator.cv_qdot);
108  /* commit roomtemperature*/
109  room_temperature=room_temperature_cv;
110  outlet.mdot = mdot;
111  outlet.t = radiator_temperature_out;
112 end Radiator;

114
115  model Room_radiator_window "model of a room for mpc purpose with JModelica.org"
116
117  /** parameter p */
118  /* parameter of the room */
119  parameter Modelica.SIunits.Length room_length=7.81 "length of the room";
120  parameter Modelica.SIunits.Breadth room_breadth=5.78 "breadth of the room";
121  parameter Modelica.SIunits.Height room_height=2.99 "height of the room";
122  parameter Modelica.SIunits.Length window_length=7 "length of the window";
123  parameter Modelica.SIunits.Height window_height=2.08 "height of the window";
124  parameter Modelica.SIunits.Density rho_air = 1.2 "density of air";
125  parameter Modelica.SIunits.CoefficientOfHeatTransfer u_glass=2.0 "heat transfer
126   coefficient for glass ";
127  parameter Modelica.SIunits.CoefficientOfHeatTransfer u_wall=0.612986 "heat transfer
128   coefficient for the walls of the room";
129  parameter Modelica.SIunits.SpecificHeatCapacity cp_air=1005 "specific heat capacity of
130   air";
131  parameter Modelica.SIunits.SpecificHeatCapacity cp_water=4182 "specific heat capacity of
132   water";
133  Radiator heating "instance of a radiator";
134  Window window "instance of a window";
135  /* calculated parameter */
136  Modelica.SIunits.Volume room_volume "volume of the room";
137  Modelica.SIunits.Mass room_mass "mass of air within the room";
138  Modelica.SIunits.Area building_surface "sum of contacting surfaces (walls) with other
139   rooms of the building";
140  Modelica.SIunits.Area environment_surface "sum of contacting surfaces (walls) with the
141   environment";
142  inner Modelica.SIunits.Area window_surface "sum of contacting surfaces (windows) with
143   the environment";
144  Modelica.SIunits.Energy room_u "inner energy of the system room";
145  Modelica.SIunits.HeatFlowRate building_qdot "rate of heat flow with other rooms within
146   the building";
147  Modelica.SIunits.HeatFlowRate environment_qdot "rate of heat flow with the environment";
148  Modelica.SIunits.HeatFlowRate environment_qdot_wall "rate of heat flow with the
149   environment through the wall";
150  Modelica.SIunits.HeatFlowRate environment_qdot_window "rate of heat flow with the
151   environment through the window";
152  Modelica.SIunits.HeatFlowRate qdot_loss "summed up rate of heatflow leaving the system";
153  Modelica.SIunits.HeatFlowRate radiator_qdot "heat flow rate at the radiator surfaces
154   streaming into the room";

144
145  /** states x */
146  inner Modelica.SIunits.Conversions.NonSIunits.Temperature_degC room_temperature(start
147   =24, fixed=true)
148  "temperature within the room ( Initially 24 degree celsius )";

148
149  /** controls u */
150  Modelica.SIunits.MassFlowRate mdot=inlet_radiator.mdot
151  "commitment of the massflowrate to the radiator";
152  Modelica.SIunits.Conversions.NonSIunits.Temperature_degC environment_temperature=
153   inlet_environment.t
154  "temperature of the environment";
155  Modelica.SIunits.Conversions.NonSIunits.Temperature_degC building_temperature=

```

```

    inlet_building.t
    "temperature of the rest of the building ";
156  Modelica.SIunits.HeatFlowRate qdot_sun
    "rate of heat flow brought in by the sun";
158  Modelica.SIunits.HeatFlowRate qdot_otherfactors=inlet_other.qdot
    "rate of heat flow brought in by other factors (e.g. people, computer)";
160

equation
162  /* calculate room volume */
    room_volume=room_length*room_height*room_breadth;
164  /* calculate room mass */
    room_mass=room_volume*rho_air;
166  /* calculate surface of the room with other rooms of the building */
    building_surface=(room_length*room_breadth*2)+(room_length*room_height*2);
168  /* calculate wall surface of the room with the environment */
    environment_surface=(room_length*room_height)+(room_breadth*room_height)-
        window_surface;
170  /* calculate window surface of the room with the environment */
    window_surface=(window_length*window_height);
172  /* calculate inner energy*/
    room_u = room_mass * cp_air * room_temperature; //?berpr?fen ob ?berhaupt n?tig?!?!
174  /* calculate derival of the inner energy */
    der(room_u) = radiator_qdot + qdot_loss + qdot_sun + qdot_otherfactors;
176  /* sum up the lost heat flow */
    qdot_loss = building_qdot+environment_qdot;
178  /* calculate lost heatflow with other rooms of the building */
    building_qdot = u_wall * building_surface * (building_temperature-room_temperature);
180  /* sum up the lost heatflow with the environment */
    environment_qdot = environment_qdot_window + environment_qdot_wall;
182  /* calculate lost heatflow with the environment through the window */
    environment_qdot_window = u_glass * window_surface * (environment_temperature -
        room_temperature);
184  /* calculate lost heatflow with the environment through the wall */
    environment_qdot_wall = u_wall * environment_surface * (environment_temperature -
        room_temperature);
186  /* commit the inflowing heat flow of the radiator */
    radiator_qdot=heating.radiator_qdot_out;
188  /* connect radiator with the inlet */
    connect( heating.inlet, inlet_radiator );
190  connect( inlet_sun, window.inlet_sun );
    qdot_sun = window.outlet_room.qdot;
192 end Room_radiator_window;

194
connector MassTemperature
196
    Modelica.SIunits.Conversions.NonSIunits.Temperature_degC t;
    Modelica.SIunits.MassFlowRate mdot;
end MassTemperature;
200

202 model SourceTemp
204
    Modelica.SIunits.Conversions.NonSIunits.Temperature_degC t=inlet ;
    Modelica.Blocks.Interfaces.RealInput inlet ;
206
equation
208
    outlet.t =t;
end SourceTemp;
210

212 model SourceHeat
214
    Modelica.SIunits.HeatFlowRate qdot=inlet;

```

```
    Modelica.Blocks.Interfaces.RealInput  inlet ;
216
217 equation
218   outlet.qdot = qdot;
219 end SourceHeat;
220
221
222 model SourceTempMass
223
224   Modelica.SIunits.Conversions.NonSIunits.Temperature_degC t=inlet_t;
225   Modelica.SIunits.MassFlowRate mdot=inlet_mdot;
226   Modelica.Blocks.Interfaces.RealInput  inlet_t ;
227   room_model_backup.MassTemperature outlet;
228   Modelica.Blocks.Interfaces.RealInput  inlet_mdot;
229
230 equation
231   outlet.t =t;
232   outlet.mdot=mdot;
233 end SourceTempMass;
234
235
236 model RoomRadiator
237
238   Modelica.Blocks.Sources.RealExpression  realExpression1 (y=0);
239   Modelica.Blocks.Sources.RealExpression  realExpression2 (y=0);
240   Modelica.Blocks.Sources.RealExpression  realExpression4 (y=20);
241   Modelica.Blocks.Sources.RealExpression  realExpression6 (y=0.1);
242   Modelica.Blocks.Sources.RealExpression  realExpression3 (y=60);
243   SourceHeat Sonne;
244   Modelica.Blocks.Sources.RealExpression  realExpression5 (y=800);
245   SourceTempMass Heizkorper;
246   SourceTemp UmgebungA;
247   SourceTemp Gebaude;
248   SourceSun sourceSun;
249   Room_radiator_window room_radiator_window;
250
251 equation
252   connect( realExpression3.y, Heizkorper.inlet_t );
253   connect( realExpression6.y, Heizkorper.inlet_mdot );
254   connect( realExpression4.y, Gebaude.inlet );
255   connect( realExpression2.y, UmgebungA.inlet);
256   connect( realExpression1.y, Sonne.inlet );
257   connect(room_radiator_window.inlet_other, Sonne.outlet);
258   connect(room_radiator_window.inlet_sun, sourceSun.outlet);
259   connect(UmgebungA.outlet, room_radiator_window.inlet_environment);
260   connect( Gebaude.outlet, room_radiator_window.inlet_building);
261   connect( Heizkorper.outlet, room_radiator_window.inlet_radiator);
262   connect( realExpression5.y, sourceSun.inlet );
263 end RoomRadiator;
264
265
266 model Window
267
268   Modelica.SIunits.DensityOfHeatFlowRate radiation_arriving =inlet_sun.radiation_sun ;
269   Modelica.SIunits.HeatFlowRate qdot_effective ;
270   parameter Real window_transmission = 0.00687213;
271   outer Modelica.SIunits.Area window_surface "sum of contacting surfaces (windows) with the
272   environment";
273   HeatFlow outlet_room;
274   RadiantEnergyFluenceRate inlet_sun;
275
276 equation
277   qdot_effective = radiation_arriving * window_transmission * window_surface;
278   qdot_effective = outlet_room.qdot;
```

```
278 end Window;  
  
280 connector RadiantEnergyFluenceRate  
  Modelica.Slunits.DensityOfHeatFlowRate radiation_sun;  
end RadiantEnergyFluenceRate;  
  
284  
model SourceSun  
  Modelica.Slunits.DensityOfHeatFlowRate radiation_sun = inlet;  
  Modelica.Blocks.Interfaces.RealInput inlet;  
  RadiantEnergyFluenceRate outlet;  
  
290 equation  
  outlet.radiation_sun = radiation_sun;  
end SourceSun;  
  
294 annotation (uses(Modelica(version="3.2.1")));  
end room_model_backup;
```

Literaturverzeichnis

- [pys] *Pysolar by pingswept*. <https://github.com/pingswept/pysolar>, Abruf: Abruf vom 15.03.2016
- [osi 1996] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: ISO 7498-1: Information technology – Open Systems Interconnection – Basic Reference Model: The basic model. Genf, Switzerland, März 1996. – ISO 7498-1 Standard
- [mod 2006a] MODBUS ORGANIZATION: Modbus messaging on TCP/IP implementation guide v1.0b. Hopkinton, USA, Oktober 2006. – Forschungsbericht
- [mod 2006b] MODBUS ORGANIZATION: Modbus over serial line specification and implementation guide V1.02. Hopkinton, USA, Dezember 2006. – Forschungsbericht
- [mod 2012] MODBUS ORGANIZATION: Modbus application protocol specification v1.1b3. Hopkinton, USA, April 2012. – Forschungsbericht
- [bi1 2015] *Bilanz zur Energiewende 2015*. 2015
- [AB 2015] AB, Modelon (Hrsg.): *JModelica.org User Guide - Version 1.17*. Ideon Science Park, SE-223 70 Lund : Modelon AB, 2015
- [Agency 2015] AGENCY, International E.: World Energy Outlook 2015 - Zusammenfassung / International Energy Agency. 2015. – Forschungsbericht
- [Baehr u. Kabelac 2012] BAEHR, Hans ; KABELAC, Stephan: *Thermodynamik: Grundlagen und technische Anwendungen*. 15. Auflage 2012. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012
- [Bertrand Blanc 2005] BERTRAND BLANC, Bob M.: Endianness or Where is Byte 0? 2005. – Forschungsbericht
- [Böckh u. Wetzel 2014] BÖCKH, Peter v. ; WETZEL, Thomas: *Wärmeübertragung: Grundlagen und Praxis*. 5. überarbeitete und erweiterte Auflage 2014. Berlin, Heidelberg : Springer Vieweg, 2014
- [Bürger 2016] BÜRGER, Adrian: *Casadi Interface for Optimum experimental design and Parameter Estimation and Identification Applications*. <http://casiopeia.readthedocs.org/en/latest/>, 2016
- [Core Writing Team u. (eds.) 2014] CORE WRITING TEAM, R.K. P. ; (EDS.), L.A. M.: *Climate Change 2014: Synthesis Report*. IPCC, Geneva, Switzerland, 2014
- [Diehl 2014] DIEHL, Moritz: Lecture Notes on Optimal Control and Estimation / Institut für Mikrosystemtechnik – IMTEK Albert-Ludwigs-Universität Freiburg. 2014. – Forschungsbericht
- [Furrer 2003] FURRER, Frank J.: *Industrieautomation mit Ethernet-TCP/IP und Web-Technologie*. 3. neu bearbeitete und erweiterte Auflage. Heidelberg: Hüthig, 2003
- [Google Inc.] GOOGLE INC., California Mountain V. Mountain View: *Googlemaps - Bismarckstraße 12, 76133 Karlsruhe*. <https://www.google.de/maps/place/Bismarckstra\T1\sse+12,+76133+Karlsruhe/@49.01312,8.3918928,19z/data=!3m1!4b1!4m2!3m1!1s0x47970656caeb24c5:0x630ef97b3685becd?hl=de>, Abruf: Abruf vom 12.12.2015

- [Hauser 2000] HAUSER, Höttges K G.: Bauphysik in Kürze: U-Werte von Fenstern. In: *Bauphysik* 22 (2000), S. S.270–273
- [Joel Andersson 2015] JOEL ANDERSSON, Moritz D. Joris Gillis G. Joris Gillis: *User Documentation for CasADi v2.2.0+1.cf4be18*. Joel Andersson, Joris Gillis, Moritz Diehl, 2015
- [Kaltschmitt 2013] KALTSCHMITT, Wolfgang ;Wiese A. Martin ;Streicher ;. Martin ;Streicher: *Erneuerbare Energien : Systemtechnik, Wirtschaftlichkeit, Umweltaspekte*. 5. erweiterte Auflage. Springer Berlin Heidelberg, 2013
- [Kirk 2004] KIRK, Donald E.: *Optimal control theory : an introduction*. Mineola, N.Y. : Dover Publications, 2004
- [MODICON 96] MODICON, Inc.: *Modicon Modbus Protocol Reference Guide*. One High Street, North Andover, Massachusetts 01845: MODICON, Inc., Industrial Automation Systems, June 96
- [Mühr 2016] MÜHR, Bernhard: Außentemperatur und Globalstrahlung der Wetterstation Physikhochhaus vom 01.12.2015 bis 28.02.2016 / Institut für Meteorologie und Klimaforschung (IMK-TRO), Karlsruher Institut für Technologie (KIT). 2016. – Forschungsbericht
- [Nicolai 2013] NICOLAI, Dr. A.: Physikalische Grundlagen des thermischen Raummodells THE-RAKLES / Technische Universität Dresden, Fakultät Architektur, Institut für Bauklimatik. 2013. – Forschungsbericht
- [Peter Häupl 2013] PETER HÄUPL, Christian Kölzow Olaf Riese Anton Maas Gerrit Höfker Christian N. Martin Homann ; WILLEMS, Wolfgang (Hrsg.): *Lehrbuch der Bauphysik : Schall - Wärme - Feuchte - Licht - Brand - Klima*. Springer Vieweg, 2013
- [Quaschning 2011] QUASCHNING, Volker: *Regenerative Energiesysteme : Technologie, Berechnung, Simulation*. Bd. 7. aktualisierte Auflage. München : Hanser, 2011
- [Recknagel 2013] RECKNAGEL, Schramek Sprenger: *(Taschenbuch für Heizung + Klimatechnik) : 76.2013/14*. Oldenbourg Industrieverlag, 2013
- [Reda 2008] REDA, A. I. ; A. I. ; Andreas: Solar Position Algorithm for Solar Radiation Applications (Revised) / National Renewable Energy Laboratory (NREL), Golden, CO. 2008. – Forschungsbericht
- [van Rossum u. the Python development team 2016] ROSSUM, Guido van ; TEAM the Python d.: *Python Frequently Asked Questions*. Release 2.7.11. Python Software Foundation, 2016
- [Sack 2004] SACK, Dipl.-Phys. N.: Von k zu U - Was ändert sich bei Fensterrahmen und -profilen? / ift Rosenheim Bauphysik. 2004. – Forschungsbericht
- [Schleicher 2008] SCHLEICHER, Manfred: *Digitale Schnittstellen und Bussysteme*. JUMO, 2008
- [Schnell u. Wiedemann 2006] SCHNELL, Gerhard ; WIEDEMANN, Bernhard: *Bussysteme in der Automatisierungs- und Prozesstechnik : Grundlagen, Systeme und Trends der industriellen Kommunikation*. 6. überarbeitete und aktualisierte Auflage. Wiesbaden : Vieweg+Teubner, 2006
- [Hochschule Karlsruhe Technik und Wirtschaft] WIRTSCHAFT, Adrian B.: *Kühlen mit Wärme - Solare Klimatisierung des K-Baus*. <https://www.hs-karlsruhe.de/fakultaeten/w/projekte.html>, Abruf: Abruf vom 24.03.2016

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Quellen und Hilfsmittel angefertigt habe. Alle Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Quellen entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegen.

Karlsruhe, den 29. März 2016

Daniel Johannes Mayer