

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 – 31.03.2016

Tag der Einreichung: 31.03.2016

Kurzfassung

Im Rahmen dieser Masterarbeit erfolgt die Konzeption, der Aufbau und die Inbetriebnahme einer technischen Anlage zur Regelung einer Raumtemperatur für den Betrieb mit Modellprädiktiver Regelung. Zunächst werden dazu die Anforderungen analysiert und festgelegt, bevor die Planung der Anlage ausgeführt wird. Nach einer Beschreibung der Installation und deren Funktionsweise wird die Eignung der Anlage zur Raumtemperaturregelung aufgezeigt. Der anschließende Teil dient der Bildung eines Raummodells. Dieses wird schrittweise an die realen Gegebenheiten angepasst und nach einer erfolgten Parameterschätzung evaluiert. Abschließend wird überprüft, ob sich das Modell für einen Einsatz mit der Modellprädiktiven Regelung in JMODELICA.ORG eignet.

Abstract

In the course of this master thesis a technical system for a space heating control is designed,

Schlüsselwortliste: Modellprädiktive Regelung, Model Predictive Control, JMODELICA.ORG, CASADI, Modellbildung, Modelica, Kommunikation technischer Systeme, Modbus RTU, Modbus TCP, Raumtemperaturregelung

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 der Aufbau und die Einrichtung der Anlage in diesem Umfang nicht möglich gewesen wäre.

Für die Unterstützung bei technischen Fragen und die kurze Einführung in die Modbus Protokolle, möchte ich mich bei Herrn STEFAN OSTROWSKI von EXPERTDAQ.

Herrn BERNHARD MÜHR vom Institut für Meteorologie und Klimaforschung vom Karlsruher Institut für Technologie danke ich für die freundliche Kooperationsbereitschaft und die schnelle Bearbeitung der Anfragen.

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

Für die finanzielle und ideelle Förderung, die mein Studium durch zahlreiche Akademien, Tagungen und ein weiteres Auslandssemester im kommenden Semester bereichert hat, gebührt mein besonderer Dank der STUDIENSTIFTUNG DES DEUTSCHEN VOLKES E.V.

Nicht zuletzt gilt mein Dank auch meiner Freundin HENRIETTE BLANK und TOBIAS RUHLAND für die Übernahme der Korrektur dieser Arbeit und allen weiteren Personen, die mich während des gesamten Studiums und bei der Umsetzung dieser Arbeit unterstützt haben.

Inhaltsverzeichnis

Tabellenverzeichnis	IV
Abbildungsverzeichnis	V
Quelltextverzeichnis	VII
Symbolverzeichnis	VIII
1 Einleitung	1
1.1 Motivation und Problemstellung	1
1.2 Zielsetzung und Aufbau der Arbeit	3
2 Theoretische Grundlagen	5
2.1 Grundlagen zur Modellprädiktiven Regelung	5
2.1.1 Optimalsteuerung	5
2.1.2 Modellprädiktive Regelung	7
2.2 Technische Grundlagen zur Kommunikation mit Bussystemen	8
2.2.1 Bussysteme	8
2.2.1.1 Informationsaustausch	8
2.2.1.2 Netzwerk und Topologie	9
2.2.1.3 Buszugriffsverfahren	11
2.2.1.4 Datensicherung	13
2.2.1.5 Schnittstellen	15
2.2.2 OSI-Kommunikationsmodell	18
2.2.3 Modbus Kommunikationstechnologie	22
2.3 Technische Grundlagen zur Modellbildung	29
2.3.1 Thermodynamische Systeme	29
2.3.2 Erster Hauptsatz der Thermodynamik	31
2.3.3 Wärmeübertragung	32
3 Anlagendesign	34
3.1 Analyse der Anforderungen	34
3.1.1 Einsatzziele und Rahmenbedingungen	34
3.1.2 Definition der Anforderungen	35
3.2 Idee und Ziel der Anlage	40
3.2.1 Aufgabe der Anlage	40
3.2.2 Idee der Anlage	40
3.3 Konzept und Planung	41
3.3.1 Netzwerkarchitektur	42
3.3.2 Erfassung der Raumtemperatur	42

3.3.3 Steuerung des Heizkörpers	44
3.4 Installation und Inbetriebnahme	45
3.4.1 Hardware	45
3.4.2 Software	47
3.4.3 Inbetriebnahme mit einem 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
4.3 Modellprädiktive Regelung	79
5 Schlussbetrachtung	81
5.1 Fazit	81
5.2 Ausblick und Ansatzpunkte für weitere Arbeiten	82
A Modelle, Programme, Messdaten	83
B Das Raummodell	84
C Datenblätter	89
C.1 EX9132C-2-MTCP Gateway von ExpertDAQ	89
C.2 EX9024M Signalwandler von ExpertDAQ	95
C.3 THERMASGARD RTM1-Modbus Raumtemperaturfühler von S+S Regeltechnik	102
C.4 Webthermograph 8x von WuT	103
C.5 MULTICAL 602 Wärmemengenzähler von Kamstrup	107
C.6 ABNM-LIN Stellantrieb von Danfoss	113
C.7 UNITRONIC Busleitung von LappKabel	117
D Modbusadressmapping	118
D.1 EX9024M Signalwandler von ExpertDAQ	118
D.2 THERMASGARD RTM1-Modbus Raumtemperaturfühler von S+S Regeltechnik	119
D.3 MULTICAL 602 Wärmemengenzähler von Kamstrup	122

Literaturverzeichnis	128
-----------------------------	-----

Tabellenverzeichnis

Tab. 3.1: Umsetzung der Ziele in Anforderungen der Anlage	36
Tab. 3.2: Netzwerkkonfiguration der Ports des EX9132M-MTCP Gateways . .	46
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, Neben- und Randbedingungen eines optimalen Steuerungsproblems	7
Abb. 2.2:	Bus-Struktur	10
Abb. 2.3:	Impulsverzerrung auf einer Leitung	11
Abb. 2.4:	Baumstruktur	12
Abb. 2.5:	<i>Cyclic Redundancy Check</i>	14
Abb. 2.6:	Parallele und serielle Datenübertragung	15
Abb. 2.7:	Spannungspiegel und Stecker der EIA 232-Schnittstelle	16
Abb. 2.8:	Spannungspiegel und Stecker der EIA 485-Schnittstelle	17
Abb. 2.9:	Die sieben Schichten des Open System Interconnection Modells	19
Abb. 2.10:	Die vier Dienstvorgänge	20
Abb. 2.11:	Die Modbus Kommunikation im OSI-Referenzmodell	23
Abb. 2.12:	Allgemeiner Rahmen für Telegramme nach dem Modbus Anwendungsprotokoll	23
Abb. 2.13:	Transaktion mit dem Modbus Protokoll	24
Abb. 2.14:	Datenmodell und Adressierung nach dem Modbus Protokoll	25
Abb. 2.15:	Serielle Kommunikation über Modbus RTU	27
Abb. 2.16:	Eine Modbus TCP/IP Kommunikationsarchitektur	27
Abb. 2.17:	Angepasster Rahmen für Telegramme nach dem Modbus TCP/IP Protokoll	28
Abb. 3.1:	Raumskizze K004b vom K Gebäude der Hochschule Karlsruhe – Technik und Wirtschaft	37
Abb. 3.2:	Prinzipskizze eines technischen Systems zur Raumtemperaturregelung des Raumes K004b	41
Abb. 3.3:	Aufbau des Netzwerks	42
Abb. 3.4:	Verteilung der Raumtemperaturfühler	43
Abb. 3.5:	Schaltplan der Raumtemperaturregelungsanlage	46
Abb. 3.6:	Schaltplan der Raumtemperaturregelungsanlage	47
Abb. 3.7:	UML Klassendiagramm der zentralen Anlagensteuerung	57
Abb. 3.8:	Raumtemperturverlauf für ein 14-tägiges Intervall im Februar 2016	58
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
Abb. 4.9: Parameterschätzung	77
Abb. 4.10: Simulation des Raummodells mit Einsatz des Heizkörpers mit geschätztem Parameter	78
Abb. 4.11: Simulation des Raummodells mit Einsatz des Heizkörpers mit geschätztem Parameter	78
Abb. 4.12: JMODELICA.ORG Kompatibilität des Modells	80

Quelltextverzeichnis

3.1 Klasse <i>SensorsHttp</i> zur Abfrage der Temperaturen vom Webthermograph8x	48
3.2 Klasse zum Verbindungsaufbau und für die Grundfunktionen über Modbus TCP	49
3.3 Klasse zum Auslesen über Modbus TCP	51
3.4 Klasse zum Auslesen über Modbus TCP	52
3.5 Die <i>Actutators</i> und <i>Sensors</i> 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
4.8 Programm zur Übersetzung des Raummodells in Optimierungsfähige Objekte in JMODELICA.ORG	79
listings/room_model_listing.mo	84

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]

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

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 als im Jahr 2015 [Agency, 2015, S. 1].

Das Intergovernmental Panel on Climate Change (IPCC) betrachtet den Menschen als eine der Hauptursachen des Klimawandels, insbesondere jedoch als Verursacher für die globale Erderwärmung [Core Writing Team u. , eds., S. V]. Eine Schlüsselrolle bei der globalen Erwärmung spielt der Ausstoß von Treibhausgasen, wovon ein großer Anteil bei der Erzeugung von Elektrizitäts- und Wärmeenergie entsteht und nebenbei große Mengen an CO₂ in die Erdatmosphäre entweichen[Core Writing Team u. , eds., S. 47].

Die Bedeutung der Energieerzeugung wird daher auch in Zukunft immer weiter zunehmen. Da die Energieerzeugung aus Erneuerbaren Energien ohne den Ausstoß schädlicher Treibhausgase auskommen, bietet der Ausbau der erneuerbaren Energien eine enorme Chance. Dies wird beispielsweise durch die Zielsetzung der Bundesregierung aus Deutschland belegt, die sich vorgenommen hat 80% des Energiebedarfs im Jahre 2050 aus Erneuerbaren Energien zu decken [bi1, 2015, S. 2]. Als Erneuerbare Energien werden Energiequellen bezeichnet, die aus Sicht des menschlichen Zeithorizonts unerschöpflich sind und daher ein gewaltiges Potenzial mit sich bringen. Die Erneuerbaren Energien 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 noch einige Problemstellungen zu lösen. Eine davon ist die Frage, wie sich die technische Umwandlung der Solarenergie in eine nutzbare Energieform, wie beispielsweise Wärme oder Elektrizität, möglichst effizient realisieren lässt. Hiermit verbunden ist ebenfalls die Frage, wie bereits bestehende Technologien weiter optimiert werden können. Eine dieser Technologien, die von Bedeutung sind, sind solarthermische Kraftwerke, welche die Solarenergie zunächst in Wärme und anschließend in elektrische Energie umwandeln. Eine weitere derartige Technologie

stellen die Solarkollektoren und -zellen dar, die zur CO₂-freien Erzeugung von Wärme und Strom eingesetzt werden [Quaschning, 2011, S. 36f.].

Ein zweites großes Problem ist der steigende globale Energiebedarf, welchem durch eine Erhöhung der Energieeffizienz entgegengewirkt werden kann. Die größten Einsparpotenziale sind dabei nicht in dem Bereich privater Haushalte zu finden, sondern in der Industrie. Die Bundesregierung von Deutschland sieht in den besagten Einsparpotenzialen einen Investitionsmotor. Werden diese realisiert, bestehen ein größerer monetärer Spielraum für Investitionen der Industrie und den Konsum der Privathaushalte [bi1, 2015, S. 2].

Damit ganzheitliche Lösungsansätze entwickelt und die Energiewende erfolgreich gemanagt werden kann, ist die Forschung von zentraler Bedeutung. Sie umfasst die kontinuierliche Verbesserung bestehender und die Entwicklung neuer Technologien. Die Bundesregierung misst Deutschland im Zuge der Energiewende eine Vorreiterrolle zu und versucht diese durch verschiedene Forschungsprogramme zu untermauern. Hierunter fallen Projekte zur Energieversorgung aus Erneuerbaren Energiequellen, der Energiespeicherung und der Verbesserung der Energieeffizienz [bi1, 2015, S. 11].

Auch die Hochschule Karlsruhe trägt durch eines ihrer Forschungsprojekte einen Teil dazu bei, welches die Modellprädiktive Regelung (MPR)¹ einer Anlage zur solaren Klimatisierung eines Fakultätsgebäudes zum Ziel hat.

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

Konkret umfasst das Forschungsprojekt der Hochschule Karlsruhe den Aufbau einer solaren Anlage. Sie nutzt die in Solarkollektoren gewonnene Wärmeenergie, um eine Adsorptionskälteanlage anzutreiben und dadurch für die Kühlung des K Gebäudes zu sorgen. Die Installation der Anlage hat sich aufgrund einiger Probleme verzögert und befindet sich derzeit noch im Aufbau Hochschule Karlsruhe Technik und Wirtschaft.

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

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

Diese Arbeit beschäftigt sich mit der Konzeption, Umsetzung und Inbetriebnahme einer kleinen Anlage, mit deren Hilfe nützliche Erfahrungen für die Inbetriebnahme der großen Solaranlage gesammelt werden sollen, um diese zu vereinfachen und deren Anlaufzeit zu verkürzen. Darüber hinaus soll durch diese Arbeit komplementärer Forschungsbeitrag zur großen Solaranlage im Bereich der Modellprädiktive Regelung realisiert werden.

1.2 Zielsetzung und Aufbau der Arbeit

Das übergeordnete Ziel dieser Arbeit ist es, die Forschung der Hochschule Karlsruhe auf dem Gebiet der Modellprädiktiven Regelung von solaren Anwendungen voranzutreiben. 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 vergleichsweise 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 kann, um eine Modellprädiktive Regelung der Raumtemperatur zu ermöglichen.

Aus jener Forschungsfrage lässt sich das konkrete Ziel ableiten, eine entsprechende Anlage zu planen und zu installieren. Zudem bedarf es der Bildung eines Modells für die Modellprädiktive Regelung der kleinen Anlage. Es soll also eine Forschungsumgebung zur Untersuchung verschiedener Steuerungen und Regelungssystematiken sowie zur Entwicklung eigener Regelungsalgorithmen geschaffen werden. Die Eigenschaften und Einsatzziele der Anlage wurden gemeinsam mit den beiden Projektverantwortlichen der Hochschule Karlsruhe, in Person von Herrn ADRIAN BÜRGER und MARKUS BOHLAYER, festgelegt und sind in Kapitel 3.1 detailliert aufgeführt.

Auf die Einleitung folgt eine Einführung in die theoretischen Grundlagen, die dem Verständnis beim Aufbau der Anlage und der Bildung des Modells dienen.

Um den Rahmen der Arbeit abzugrenzen, werden zunächst die Theorien der Modellprädiktive Regelung und der Optimalsteuerung erläutert, mit dem Ziel die Anforderungen an die Anlage und Modellbildung zu identifizieren. Nach der Abgrenzung des Rahmens werden die allgemeinen Grundlagen zur Kommunikation von technischen Systemen thematisiert, welche bestimmte Grundlagen aus den Gebieten der Informatik, Elektro- und Nachrichtentechnik umfassen. Es wird ein theoretisches Fundament gelegt, auf welches im Anschluss die praktische Anwendung der Modbus Kommunikationstechnologie aufgebaut wird. Der darauffolgende Abschnitt beschreibt die thermodynamischen Grundlagen, die bei der Beschreibung realer Prozesse in der Modellbildung in Kapitel 4 Anwendung finden. Abschließend erfolgt eine allgemeine Einführung in die relevanten physikalischen und bautechnischen Grundlagen zur Solarstrahlung und Gebäudetechnik. Diese zielt darauf ab, ein Grundverständnis für den Einfluss der

Solarstrahlung auf einen Raum zu entwickeln.

Das dritte Kapitel dient der Beschreibung des Aufbaus und der Funktionsweise der Anlage. Dazu werden zunächst die Anforderungen an die geplante Anlage definiert. Darauf basierend wird der Aufbau der Anlage von der Idee, bis hin zur abschließenden Beschreibung der installierten Anlage und deren Bedienung sukzessiv konkretisiert. Schließlich soll durch eine funktionierende Zweipunktregelung die Hypothese belegt werden, dass die Anlage fähig ist eine Raumtemperatur zu regeln.

Das Ziel des vierten Kapitels ist es, ein Modell der Anlage für die Modellprädiktive Regelung der Raumtemperatur zu entwickeln. Es wird ein einfaches Grundmodell aufgestellt, dessen Komplexität durch eine schrittweise Anpassung an die reale Umgebung stetig erhöht wird. Anschließend erfolgt die Simulation und Validierung des Modells, um eine Abschätzung für die Modellgüte zu erhalten. Im nachfolgenden Abschnitt erfolgt eine erneute Anpassung des Modells durch eine Parameterschätzung und abschließende Untersuchung der Modellgüte. Zuletzt wird die Eignung des Modells für den Einsatz mit Modellprädiktive Regelung untersucht.

Die Schlussbetrachtung soll eine Antwort auf die Forschungsfrage liefern, indem die Ergebnisse der Arbeit zusammengefasst werden. Zum Abschluss der Arbeit wird noch ein Ausblick gegeben, bei dem mögliche Ansatzpunkte für weitere Arbeiten aufgezeigt werden.

„Theorie ist die Mutter der Praxis.“
— LOUIS PASTEUR, französischer Chemiker und
Mikrobiologe

2 Theoretische Grundlagen

Die folgenden Abschnitte dienen dazu, die grundlegende Zusammenhänge verschiedener Fachgebiete zu beschreiben, um dem weiteren Verlauf der Arbeit einfach folgen zu können.

Dazu erfolgt zunächst eine allgemeine Einführung in die *Theorie der Optimalsteuerung* und der darauf aufbauenden *Modellprädiktive Regelung*, um die Rahmenbedingungen für die folgenden Kapitel zu definieren.

Im darauffolgenden Abschnitt werden die *Grundlagen zur Kommunikation über Bus-systeme* ausführlich beschrieben. Sie umfassen elektrotechnische Grundlagen zur Datenübertragung und Vernetzung von technischen Systemen sowie Aspekte der Nachrichtentechnik und Informatik. Anschließend werden die Grundlagen mithilfe der *Modbus Kommunikationstechnologie* in einen Anwendungskontext gebracht, damit sie in Kapitel 3 bei der Installation der Anlage angewendet werden können.

Weiterhin erfolgt eine Beschreibung der *thermodynamischen Grundlagen*, die bei der Beschreibung der Prozesse und Modellbildung in Kapitel 4 angewandt werden.

Abschließend erfolgt eine allgemeine *Einführung in die Solar- und Gebäudetechnik*, um den Einfluss der Solarstrahlung auf die Raumtemperatur in Abschnitt 4.1.6 beschreiben zu können.

2.1 Grundlagen zur Modellprädiktiven Regelung

Die *Modellprädiktive Regelung* ist ein Teilgebiet der angewandten Mathematik, welche sich mit der Regelung von Prozessen oder technischen Systemen hinsichtlich gewählter Optimalitätskriterien beschäftigt. Hierzu bedient sie sich mathematischer Modelle des zu steuernden Prozesses sowie den *Grundlagen der Optimalsteuerung*. Zur optimalen Regelung eines Prozesses werden wiederholt Optimalsteuerungsprobleme gelöst, wodurch sich besondere Anforderungen an das Modell ergeben [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 anhand von spezifizierten Optimalitätskriterien optimal zu steuern. Durch die Lösung eines Optimalsteuerungsproblems werden

Steuersignale bestimmt, sodass die spezifizierten Leistungskriterien minimiert beziehungsweise maximiert werden und das Systeme eventuell vorgegebenen physikalischen und nutzerdefinierten Neben- sowie Randbedingungen genügt [Kirk, 2004, S. 3f.]. Die Prozesse dynamischer Systeme lassen sich durch *Zustände* und *Parameter* modellhaft beschreiben. Erstgenannte stellen die zu steuernden Größen dar und gemeinsam mit den Parametern beschreiben sie das *Systemverhalten*. Das Modell kann wie zuvor erwähnt durch *Rand- und Nebenbedingungen* beschränkt werden. Weiterhin können die Zustände von *Steuergrößen* beeinflusst werden, welche frei wählbar oder vorgegeben sein können. Diese werden auch als *Controls* bezeichnet und können durch Grenzen eingeschränkt werden. Sie stellen einen Freiheitsgrad des Systems dar. Des Weiteren wird das Leistungskriterium durch die *Kostenfunktion* zum Ausdruck gebracht. Das Optimalsteuerungsproblem lässt sich dann wie folgt beschreiben [Diehl, 2014, S. 61]:

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

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 Controls vom Vektor \vec{u} beschrieben. Die Parameter sind implizit im Modell enthalten, welches durch ein Gleichungssystem aus gewöhnlichen Differenzialgleichungen (ODE Model) beschrieben wird. Die Kostenfunktion wird auch als *Bolza Ziel* bezeichnet und besteht aus dem *Lagrange-* und den *Mayer-Term*. Letzterer beschreibt die Kosten zum Endzeitpunkt T des Optimierungsintervalls wohingegen der Lagrange-Term den integralen Kostenanteil umfasst. Die Randbedingungen werden von den Initialwerten $\vec{x}(0) - \vec{x}_0 = 0$ (Fixed Initial Value) und den Endbedingungen $r(\vec{x}(T)) \leq 0$ (Terminal Constraints) berücksichtigt. Die physikalischen und nutzerdefinierten Nebenbedingungen sind durch die Path Constraints $h(\vec{x}(t), \vec{u}(t)) \leq 0$ beschrieben. Anschaulich sind diese Zusammenhänge in Abb. 2.1 dargestellt.

Um eine *optimale Lösung* dieses zeitkontinuierlichen Optimierungsproblems zu bestimmen, muss die erste Ableitung die *notwendigen Karush-Kuhn-Tucker* Bedingungen erfüllen. Weiterhin muss die zweite Ableitung einer *hinreichenden* Bedingung genügen. Daher ist es notwendig, dass das nichtlineare Optimalsteuerungsproblem zweifach stetig differenzierbar ist und damit auch das mathematische Modell des Prozesses [Diehl, 2014, S. 21ff.].

Eine *optimale Lösung* beschreibt einen optimalen Einsatz von Steuergrößen – hinsichtlich der formulierten Kostenfunktion – zur Beeinflussung des Systems und wird auch

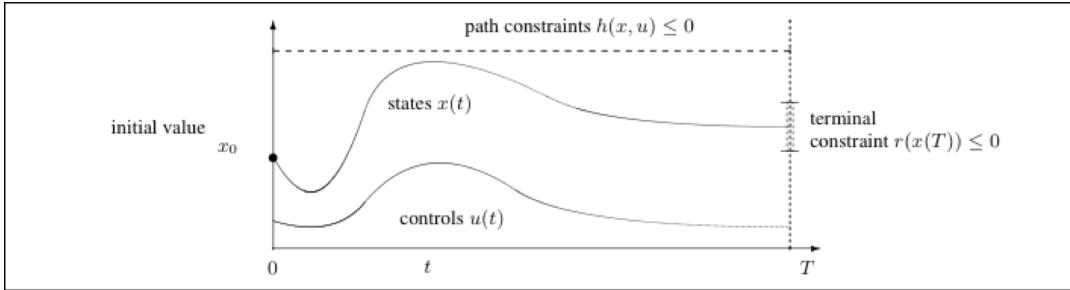


Abb. 2.1: Variablen, Neben- und Randbedingungen eines optimalen Steuerungsproblems, entnommen aus [Diehl, 2014, S.61]

als *Optimalsteuerungsplan* bezeichnet.

Konkret können mehrere, numerische Ansätze zur Lösung des Optimalsteuerungsproblems verfolgt werden, die sich in den *Zustandsraum*, die *indirekten* sowie *direkten Verfahren* einteilen lassen. Reale Probleme zeichnen sich durch eine Vielzahl an Nebenbedingungen aus. Diese eignen sich besonders zur Optimierung mit direkten Verfahren, weshalb sie in der Praxis am weitesten verbreitet sind [Diehl, 2014, S. 63]. Da die Modellprädiktive Regelung nur den Rahmen dieser Arbeit bildet, wird nicht näher auf die Optimalsteuerung eingegangen. Der interessierte Leser findet in Diehl [2014] eine theoretische fundierte Einführung zur Optimalsteuerung sowie eine ausführliche Beschreibung der einzelnen Lösungsverfahren.

2.1.2 Modellprädiktive Regelung

Da zwischen Modellen und den realen Prozessen immer Diskrepanzen bestehen, wird der Prozess durch den Einsatz eines berechneten Optimalsteuerungsplans mit höchster Wahrscheinlichkeit nicht exakt dem vorausberechneten Weg folgen. Dieses Phänomen wird auch als *Model-Plant-Mismatch* bezeichnet. Die Modellprädiktive Regelung eignet sich hervorragend, um diesen Model-Plant-Mismatch sowie Störungen des Prozesses auszugleichen, ohne den Einsatz einer umfassenden *Echtzeitregelung*. Eine solche Echtzeitregelung lässt sich aufgrund des enormen Rechenaufwands ohnehin nur durch Näherungen und Vereinfachungen für sehr einfache Anwendungen umsetzen. Diese Lücke wird durch die Modellprädiktive Regelung geschlossen. Sie betrachtet ein beschränktes, immer gleich weit in die Zukunft hinein ragendes Zeitintervall und versucht zu diskret wiederholten Zeitpunkten das Verhalten des Systems vorherzusagen. Dies geschieht durch eine Vorausberechnung mithilfe eines mathematischen Modells des Prozesses, aufbauend auf der Kenntnis des aktuellen Zustandes. Dabei wird gleichzeitig durch die periodisch Berechnung von Optimalsteuerungsplänen versucht, die projizierten Zustände dem gewählten Optimalitätskriterium entsprechend zu beeinflussen. Dadurch kann auf Abweichungen des Prozesses zum ursprünglichen Optimalsteuerungsplan zu diskreten Zeitpunkten reagiert, unabhängig davon ob diese durch eine Modellabweichungen oder Störungen bedingt sind [Diehl, 2014, S. 71].

Im Rahmen dieser Arbeit wird eine Anlage aufgebaut und ein Modell gebildet, die

eine Modellprädiktive Raumtemperaturregelung während des laufenden Betriebs ermöglichen sollen. Die hieraus abgeleiteten Anforderungen an eine Anlage werden im Kapitel 3 bei der Anforderungsanalyse beschrieben. Aus dieser Theorie ergeben sich weitere Anforderungen, welche bei der Modellbildung in Kapitel 4 dargelegt werden.

2.2 Technische Grundlagen zur Kommunikation mit Bussystemen

In diesem Abschnitt werden die technischen Grundlagen von Hard- und Software beleuchtet, die zur Kommunikation technischer Systeme benötigt werden. Die Kommunikation findet zumeist über Bussysteme statt, deren Grundlagen zuerst erklärt werden. Anschließend wird das allgemeine Open System Interconnection Modell (OSI-Modell) zur Kommunikation beschrieben. Abschließend werden die zuvor erläuterten Grundlagen anhand der *Modbus Kommunikationstechnologie* in einen Anwendungsbezug gesetzt.

2.2.1 Bussysteme

Um ganz allgemein Prozesse zu überwachen, zu steuern oder regeln zu können, müssen zwischen Prozessbeteiligten Informationen ausgetauscht werden. Die Prozessbeteiligten können technische Bauteile, Aktoren oder Sensoren sowie Controller sein, die zusammen ein technisches System bilden, welches im Folgenden als Anlage bezeichnet wird. Eine Anlage ist eine eigenständige funktionale Einheit, welche einen eigenen Zweck verfolgt. Sie zeichnet sich durch einen Mehrwert gegenüber der Summe der einzelnen Komponenten aus, der durch ein Zusammenspiel der Komponenten erreicht wird. Diese Emergenz wird durch die Kommunikation realisiert und ermöglicht der Anlage ihren Zweck zu erreichen. Es zeichnet sich allerdings ein Trend hin zur drahtlosen Kommunikation ab, die jedoch noch stark vom Einsatzort und der Störanfälligkeit begrenzt wird. Im Kontext von technischen Systemen erfolgt die Kommunikation zumeist über Bussysteme, da sie eine hohe Robustheit gegenüber Störungen besitzen und auch in unfreundlichen Umgebungen eingesetzt werden können. Sie lassen sich anhand von verschiedenen Merkmalen klassifizieren. Im Folgenden werden die wichtigsten Merkmale beschrieben in Anlehnung an die Struktur von Schnell u. Wiedemann [2006].

2.2.1.1 Informationsaustausch

Die *Informationen* über einen Prozess und dessen Zustand werden auf hoher Ebene durch *Daten* repräsentiert. Auf der untersten Ebene wird eine Information durch mehrere *Bits* repräsentiert. Der Austausch von Daten findet in Form von *Telegrammen* statt. Ein Telegramm besteht grundsätzlich aus zwei Teilen, den zu übertragenden Daten und den Informationen zur Übertragung. Die Daten werden vor der Übertra-

gung in *Rahmen*, sogenannte *Data Frames*, eingeteilt. Der detaillierte Aufbau eines Telegramms wird vom eingesetzten Kommunikationsprotokoll innerhalb des Netzwerks definiert [Schnell u. Wiedemann, 2006, S. 11f.].

Im Detail wird die Struktur eines Telegramms in Abschnitt 2.2.3 am Beispiel von zwei Modbus Kommunikationsprotokollen erläutert. Eine graphische Illustration dazu findet sich in Abb. 2.12.

2.2.1.2 Netzwerk und Topologie

Einzelne Komponenten der Anlage werden zur Kommunikation miteinander über *Verbindungsleitungen* miteinander verbunden, welche zur Übertragung von Telegrammen genutzt werden. Dabei entstehen *Netzwerke*, wobei die Komponenten der Anlage auch *Netzwerkteilnehmer* bezeichnet werden. 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 zwei Teilnehmer eines Netzwerks miteinander zu verbinden ist die sogenannte *Zweipunktverbindung*. Dazu werden die Netzwerkteilnehmer jeweils durch eine direkte Leitung miteinander verbunden. Mit der Anzahl von Teilnehmern steigt jedoch auch der Verbindungsaufwand überproportional an, um in solch *vermaschten Netzwerken* alle Teilnehmer miteinander zu verbinden. Dies hat für große vermaschte Netzwerke zur Folge, dass eine unübersichtlich große Anzahl von Schnittstellen und ein extrem hoher Verkabelungsaufwand entsteht, der wiederum mit hohen Kosten verbunden ist. Um diese Kosten zu vermeiden, ergeben sich noch weitere Möglichkeiten zur Verbindung und Anordnung von Netzwerkteilnehmern [Schnell u. Wiedemann, 2006, S. 1f.].

Um dem hohen Verkabelungsaufwand zu entgehen, wird bei großen Netzwerken zu einer *Linienstruktur* übergegangen. Diese wird auch als *Bus-Struktur* bezeichnet und ist in Abb. 2.2 veranschaulicht. Charakteristisch für die Bus-Struktur ist, dass alle Teilnehmer entlang einer langen Verbindungsleitung, dem sogenannten *Buskabel*, angeordnet sind. Sie sind mithilfe von kurzen *Stichleitungen* an das gemeinsame Buskabel angebunden, über das die gesamte Kommunikation im Netzwerk erfolgt.

Durch diese Netzwerktopologie wird der Verkabelungsaufwand sowie die Anzahl an Schnittstellen stark reduziert, insbesondere für sehr große Netzwerke. Jedoch wird durch die Nutzung einer gemeinsamen Kommunikationsleitung eine gleichzeitige Kommunikation von mehreren Teilnehmern erschwert. Daher wurden sogenannte *Buszugriffsverfahren* eingeführt, welche Zugriffsregeln auf die Busleitung definieren und im nachfolgenden Abschnitt beschrieben werden. Weiterhin müssen aufgrund der Parallelschaltung alle Teilnehmer ständig alle Telegramme mitverfolgen, wodurch der Sender stark belastet wird.

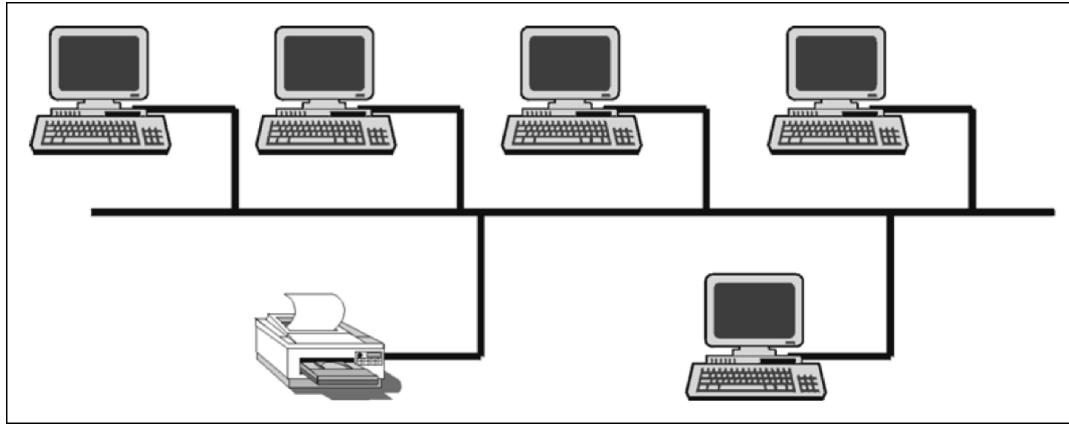


Abb. 2.2: Bus-Struktur, entnommen aus [Schnell u. Wiedemann, 2006, S. 3]

Die Busleitungslängen sind außerdem meist sehr lange². Daher ist die Länge der Leitung, bezogen auf die zu übertragende Wellenlänge, nicht mehr vernachlässigbar klein und die Busleitung muss zur Vermeidung von Reflexionen durch Leitungsabschlusswiderstände an beiden Enden abgeschlossen werden. Außerdem werden die Leitungslängen und die Teilnehmer je Netzwerksegment begrenzt [Schnell u. Wiedemann, 2006, S. 3f.].

Der Leitungs- und Kapazitätswiderstand einer Busleitung hängen unmittelbar von ihrer Länge 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 eine Impulsverzerrung Δt_{Imp} , die somit mittelbar von der Leitungslänge abhängt.

Je länger die Leitung wird, desto größer werden auch die beiden Widerstände. Die Ladezeit steigt durch die erhöhte Leitungskapazität $C_{Leitung}$, gleichzeitig sinkt die Lastspannung U_G durch den vergrößerten Leitungswiderstand $R_{Leitung}$. Dadurch verstärkt sich die Impulsverzerrung Δt_{Imp} , wie in Abb. 2.3 b) und Abb. 2.3 c) dargestellt.

Die maximale Frequenz f_{max} der Datenübertragung wird auf den Kehrwert der Impulsverzerrung $f_{max} = \frac{1}{\Delta t_{Imp}}$ beschränkt, sodass der Empfänger den Wechsel des logischen Zustandes weiterhin registrieren kann. In der Praxis bedeutet dies, dass die Leitungslänge und die Übertragungsrate miteinander verknüpft sind und sich gegenseitig beeinträchtigen [Schnell u. Wiedemann, 2006, S. 4f.].

Um die Beschränkung der Leitungslänge aufzulockern, wird die Bus-Struktur zu einer *Baum-Struktur* weiterentwickelt. In Abb. 2.4 ist zu erkennen, dass mehrere Netzwerk-Segmente durch *Verstärkerelemente* zu einem großen Netzwerk verknüpft werden. Die verschiedenen Segmente wiederum bestehen aus einzelnen Bus-Strukturen. Es wird eine galvanische Trennung der Teilnehmer benötigt, um große Flächen zu vernetzen und damit die Einschränkungen der Bus-Struktur zu umgehen [Schnell u. Wiedemann, 2006, S. 5 f.]. Die Besonderheit dieser Struktur liegt darin, dass sie durch

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, entnommen aus [Schnell u. Wiedemann, 2006, S. 4]

ihren Aufbau nachträgliche Erweiterungen mühelos ermöglicht. Die verschiedenen Bauteile zur Erweiterung eines Netzwerkes werden im nachfolgenden Abschnitt 2.2.1.5 vorgestellt.

Weitere bedeutende Netzwerktopologien sind die *Ring-* und die *Stern-Struktur*, die für das Verständnis dieser Arbeit keine weitere Relevanz besitzen. Die Ring-Struktur besteht aus einem physikalischen Ring von Zweipunktverbindungen und ist durch eine Kommunikation der Teilnehmer übereinander hinweg gekennzeichnet. Bei der Stern-Topologie hingegen, sind die Netzwerkelemente um eine Zentralstation herum angeordnet. Die gesamte Kommunikation läuft über die zentrale Komponente, die mit jedem einzelnen Teilnehmer verbunden ist. Der interessierte Leser wird für detaillierte Beschreibungen auf Schnell u. Wiedemann [2006] verwiesen.

2.2.1.3 Buszugriffsverfahren

Bei den meisten Netzwerktopologien erfolgt die Kommunikation über eine gemeinsame Verbindungsleitung. Daher müssen Regeln für den Zugriff definiert werden, die eine reibungslose Kommunikation gewährleisten. *Buszugriffsverfahren* lassen sich in zwei Gruppen untergliedern: in *kontrollierte* und *zufällige Verfahren* [Schnell u. Wiedemann, 2006, S. 19].

Bei den kontrollierten Verfahren ist der Sender bereits vor Sendebeginn eindeutig bestimmt, daher ist eine Zuteilung des Busses nicht erforderlich. Der Zugriff

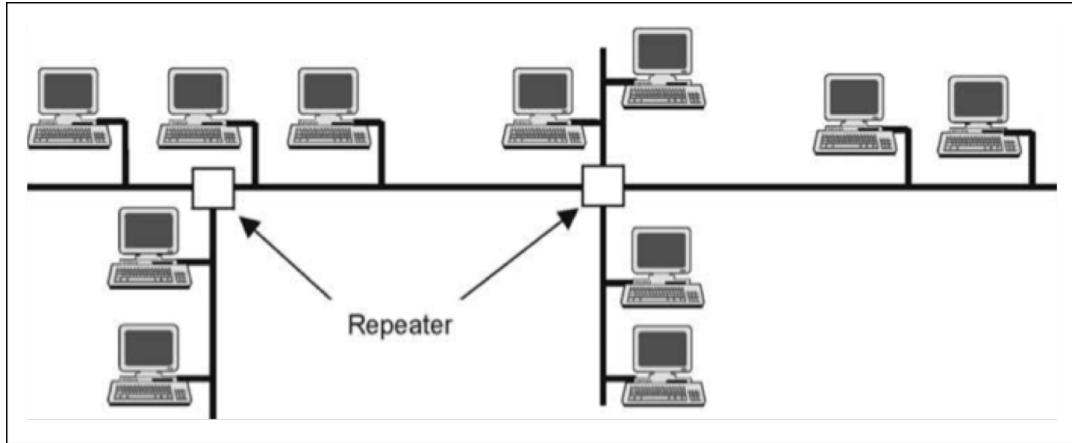


Abb. 2.4: Baumstruktur, entnommen aus [Schnell u. Wiedemann, 2006, S. 5]

findet entweder zentral statt, bei den sogenannten *Master/Slave-Verfahren*, oder wird dezentral durch Steuereinheiten geregelt, wie beispielsweise bei dem *Tokenring* und *Tokenbus*. Ein solches Verfahren heißt *echtzeitfähig*, falls die Zykluszeit der Datenübertragung durch eine Beschränkung der maximalen Telegrammlänge und damit des Übertragungsintervalls bestimmt ist. Bei zufälligen Buszugriffsverfahren greifen die Teilnehmer bei Bedarf auf die Verbindungsleitung zu. Dabei müssen sie sicherstellen, dass die Busleitung nicht gleichzeitig von einem anderen Teilnehmer belegt ist. Es kann nicht bestimmt werden zu welchen Zeitpunkten Telegramme übertragen werden, wodurch keine Echtzeitfähigkeit vorliegt [Schnell u. Wiedemann, 2006, S. 19].

Beim Master/Slave-Verfahren kommen eine *Bussteuerungseinheit*, der sogenannte *Master*, sowie mehrere *passive Teilnehmer*, die sogenannten *Slaves*, zum Einsatz. Die Kommunikation wird ausschließlich vom Master initiiert, indem er aktiv eine Verbindung zu den Slaves herstellt und *Requests* sendet, in welchen die angeforderten Daten spezifiziert sind. Die Slaves werden lediglich durch Anfragen aktiv und antworten unmittelbar mit einer *Response*, welche die angeforderten Daten enthält. Damit der Master ein umfassendes und aktuelles Bild über den Systemzustand bekommt, erfolgt die Kommunikation in der Regel in periodischen Intervallen zeitgleich mit allen Slaves (*Polling*). Als Slaves können einfache und günstige Bauteile eingesetzt werden, da im Master die benötigte „Intelligenz“ implementiert ist. Es gilt bei diesem Verfahren zu beachten, dass der Informationsaustausch zwischen verschiedenen Slaves längere Zeit in Anspruch nehmen kann und dass ein Ausfall des Masters das gesamte Bussystem stilllegt [Schnell u. Wiedemann, 2006, S. 19ff.].

Auf eine Beschreibung der übrigen Verfahren wird aufgrund der fehlenden Relevanz für diese Arbeit verzichtet. Weitere Ausführungen finden sich bei Schnell u. Wiedemann [2006].

2.2.1.4 Datensicherung

Bei der Übertragung von Informationen besteht die Gefahr von Störungen, welche sich als Fehler in einem Telegramm durch eine Invertierung einzelner Bits äußern. Störquellen sind in der Regel technischer Art, wie zum Beispiel durch elektromagnetische Störsignale, Rauschen oder Potentialdifferenzen. Gegen einen Großteil von Störfaktoren lassen sich daher Vorkehrungen treffen. Damit ergibt sich die Möglichkeit Störungen vorzubeugen oder nach Auftreten zu beseitigen. Der erste Ansatz dient einer Verminderung des Auftretens durch technische Vorkehrungen, wie beispielsweise eine Abschirmung der Kabel oder die galvanische Trennung von Netzwerken. Der zweite Ansatz beschäftigt sich mit der Überwachung des Telegrammverkehrs und den Gegenmaßnahmen beim Auftreten von Fehlern [Schnell u. Wiedemann, 2006, S. 30].

Wie zuvor erwähnt, bestehen Telegramme auf unterster Ebene aus Bitfolgen. Dabei ist jegliche Kombination von Bits erlaubt, wodurch allein aus der Folge der Bits nicht auf einen Fehler geschlossen werden kann. Bei der Übermittlung von Telegrammen können drei Arten von Fehlern auftreten:

- Der Fehler ist erkennbar und kann korrigiert werden.
- Der Fehler ist erkennbar, lässt sich jedoch nicht korrigieren.
- Der Fehler ist nicht erkennbar und damit auch nicht korrigierbar.

Falls der Fehler erkannt werden kann, ist bereits ein großer Teil der Arbeit getan. Die normale Reaktion auf erkannte Fehler, ist eine einfache Wiederholung der Übertragung, die auch als *Error Detection and Automatic Aquest Repeat* (ARQ) bezeichnet wird. Ein Fehlermaß bei der Datenübertragung ist die die *Bitfehlerrate*, welche den prozentualen Anteil fehlerhafter Bits bezogen auf die Anzahl der gesamten gesendeten Bits angibt. In der Technik ergibt sich durchschnittlich eine Bitfehlerrate von etwa $p = 10^{-4}$. Eine weitere wichtige Kennzahl ist die *Restfehlerrate*, welche die unerkannten, fehlerhaften Bitfolgen nach der Anwendung von Fehlerkennungsstrategien misst. Sie ist ein Maß für die Unversehrtheit von Telegrammen und gibt den prozentualen Anteil von fehlerhaften Bits in Bezug auf die gesamte Anzahl der Bits eines Telegramms an [Schnell u. Wiedemann, 2006, S. 31ff.].

Um Fehler systematisch zu erkennen, existieren verschiedene Fehlererkennungsstrategien. Eine einfache Möglichkeit stellt die Nutzung eines *ParitätsBits* dar. Dieses gibt lediglich an, ob die Quersumme der Bitfolge gerade oder ungerade ist. Damit können jedoch nur Fehler entdeckt werden, die eine ungerade Anzahl an Bitflips besitzen. Um die Anzahl der erkannten Fehler zu erhöhen, kann das Paritätsbit zur *Blocksicherung* erweitert werden, bei der die Paritäten über ein Array aus mehreren Telegrammen überprüft wird [Schnell u. Wiedemann, 2006, S. 34f.].

Beim sogenannten Cyclic Redundancy Check (CRC) wird ein gesamtes Telegramm als Zahl aufgefasst. Im Sender wird diese Zahl durch ein bestimmtes Generatorpolynom G geteilt. Das Ergebnis wird verworfen, lediglich der Rest bei der Division wird an das Telegramm angehängt. Der Empfänger wiederum dividiert das empfangene

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, entnommen aus [Schnell u. Wiedemann, 2006, S. 38]



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

Telegramm durch dasselbe Polynom G. Falls sich ein Divisionsrest von 0 ergibt, war die Übertragung fehlerfrei, ansonsten hat sich ein Fehler bei der Übertragung ereignet. Abhängig vom Generatorpolynom G, lassen sich unterschiedliche Güten bei 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 Telegramme kann über verschiedenste elektrische oder optische Schnittstellen geschehen und läuft bitweise ab. Je nach Topologie des Bussystems und eingesetztem Übertragungsprotokoll kann die Datenübertragung *seriell* oder *parallel* erfolgen. Bei der parallelen Datenübertragung werden mehrere Bits gleichzeitig übertragen, womit eine hohe Übertragungsgeschwindigkeit erreicht werden kann. Dazu wird eine aufwändige Netzwerktopologie, im Sinne eines vermaschten Netzes, benötigt. Daher erfolgt die Datenübertragung in der Praxis seriell, also einzelne Bits nacheinander über eine gemeinsame Kommunikationsleitung, wie zum Beispiel bei einer Bus-Struktur [Schleicher, 2008, S. 13].

Bei der *binären Datenübertragung* wird lediglich zwischen zwei Zuständen unterschieden. Die beiden Zustände werden durch Signalbereiche definiert, welche wiederum durch einen High-Pegel p_{high} und einen Low-Pegel p_{low} festgelegt sind. Der Bereich zwischen den beiden Pegeln dient der Hysterese, weshalb der Zustand für Signale in diesem Bereich nicht definiert ist [Schleicher, 2008, S. 9]. Die Geschwindigkeit der Datenübertragung wird in übertragenen Bits pro Sekunde $\frac{Bits}{s}$ gemessen und häufig auch als *Baud-Rate* bezeichnet [Schleicher, 2008, S. 22].



Abb. 2.7: Links: Spannungspegel EIA 232-Schnittstelle, entnommen aus [Schnell u. Wiedemann, 2006, S. 57]

Rechts: Stecker EIA 232-Schnittstelle, entnommen aus [Schleicher, 2008, S. 14]

Die gängigsten Übertragungsverfahren basieren auf elektrischen und optischen Schnittstellen. Die elektrischen Schnittstellen wiederum lassen sich weiter in *Strom-* und *Spannungs-Schnittstellen* untergliedern. Für den weiteren Verlauf der Arbeit sind lediglich die Spannungsschnittstellen *EIA³ 232* und *EIA 485* relevant, welche auch als *RS 232* und *RS 485* bezeichnet werden. Nähere Beschreibungen von weiteren wichtigen Schnittstellen sind in [Schleicher, 2008, S. 13ff.] und [Schnell u. Wiedemann, 2006, S. 57ff.] zu finden.

Die *RS 232 Schnittstelle* ist eine Spannungsschnittstelle, deren definierte Signalpegel 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 wird als erdunsymmetrisch bezeichnet. [Schnell u. Wiedemann, 2006, S. 57f.]

Typischerweise steht eine RS 232 Schnittstelle bei handelsüblichen Rechnern als *COM-Port* zur Verfügung. In der Automatisierungstechnik werden zumeist nur die *RxD* (Receive Data), *TxD* (Transmit Data) und *GND*, welche ein gemeinsames Bezugspotenzial definiert, Leitungen verwendet. Wichtig bei der Verkabelung ist, dass die übertragende Leitung *TxD* mit der empfangenden Leitung *RxD* verbunden wird [Schleicher, 2008, S. 14f.].

Die *RS 485 Schnittstelle* ist ebenfalls eine Spannungsschnittstelle, die in der Norm *ISO 8482* ausführlich beschrieben ist. Die Signalübertragung erfolgt mithilfe von zwei Übertragungsleitungen, die in der Regel als verdrillte und abgeschirmte Zweidrahtleitung ausgeführt sind. Die Signalpegel entsprechen der Differenzspannung U_{AB} zwischen den beiden Leitungen, die innerhalb des Intervalls von $[-7V, 12V]$ 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

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



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

Pegelfestlegung werden für Empfänger und Sender verschiedene Vorgaben gemacht, welche in Abb. 2.8 graphisch dargestellt sind. So müssen die Sender eine Differenzspannung zwischen $-1,5V < U_{AB} < -5V$ für die logische „1“ und $1,5V < U_{AB} < -5V$ für die logische „0“ leisten können. Empfänger hingegen müssen in der Lage sein, Spannungsdifferenzen von $U_{AB} < -0,3V$ als logische „1“ und $0,3V < U_{AB} < -0,3V$ als logische „0“ zu detektieren [Schnell u. Wiedemann, 2006, S. 59ff.].

Die Datenübertragung erfolgt über die *TxD/RxD+* und *TxD/RxD-* Leitung, ein gemeinsames Bezugspotential über die GND Leitung wird nicht zwingend benötigt. Allerdings wird zusätzlich ein Leitungsabschluss an beiden Enden des Buskabels benötigt. [Schleicher, 2008, S. 19f.].

Um ein Netzwerk zu erweitern, können verschiedene Arten von Bauteile eingesetzt werden. Sogenannte *Repeaters* sind aktive Bauteile, die eine kurze Zeitverzögerung verursachen und Netzwerke derselben Schnittstelle miteinander verbinden. Es werden lediglich die einzelnen Leitungen miteinander verbunden und das Datensignal verstärkt, indem die empfangenen Bits blind kopiert und auf das angeschlossene Netzwerksegment übertragen werden. Daher ist er für die Kommunikationsteilnehmer unsichtbar [Schnell u. Wiedemann, 2006, S. 79f.].

Eine Möglichkeit, um Netzwerke verschiedener Art miteinander zu verbinden, bieten *Bridges* und *Gateways*. Erstere werden auch als *Schnittstellenumsetzer* bezeichnet und kommen zum Einsatz, wenn das gleiche Übertragungsprotokoll bei unterschiedlichen physikalischen Schnittstellen genutzt wird [Schnell u. Wiedemann, 2006, S. 80f.]. Eine Bridge erkennt die Datenflussrichtung automatisch und wandelt die Signale gemäß den Pegeln des jeweils anderen Netzwerks um [Schleicher, 2008, S. 21]. Die sogenannten Gateways dienen der Kopplung von Netzwerken, die verschiedene Architekturen aufweisen. Sie werden benötigt, falls neben unterschiedlichen physikalischen Schnittstellen auch verschiedene Übertragungsprotokolle verwendet werden. Das Gateway ist demnach umfassender als eine Bridge und erweitert deren Funktionen, um die Übersetzung der Telegramme 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 verschiedene Systeme hinweg zu ermöglichen und zu formalisieren, wurde von der International Organization for Standardization (IOS) 1984 ein abstraktes Referenz-Modell entwickelt, das in der *ISO-Norm 7498-1* beschrieben ist. Es dient der Entwicklung und Verbesserung von Standards des Informationsaustausch sowie der Wahrung einer gewissen Konsistenz, als Referenz für bestehende Standards [osi, 1996, S. 1]. Das Ziel bei dem Entwurf des Modells war es, eine Menge an Standards zu schaffen, um autonomen Systemen die Kommunikation untereinander zu ermöglichen [osi, 1996, S. 4].

Das sogenannte Open System Interconnection Modell (OSI-Modell) wird im Folgenden erläutert, und abschließend im Anwendungskontext der Modbus Kommunikationstechnologie referenziert.

Zunächst wird im Standard definiert, womit sich das Modell im Allgemeinen 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 vier Bereiche gliedern [osi, 1996, S. 3f.]:

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

Bezogen auf den Austausch von Informationen, überschneidet sich die physische Verbindung und die Vernetzung zur Infrastruktur und Architektur. Die Architektur steht dann für die Übertragung zur Verfügung. Die Interaktion umfasst weitaus mehr Aufgaben: Neben der Synchronisation der Prozesse, welche Daten austauschen wollen, muss auch die Darstellung der Daten und eventuell notwendige Transformationen beachtet werden, um eine Kompatibilität unterschiedlicher Systeme zu erreichen. Weitere wichtige Aufgaben sind die Datenspeicherung und -integrität sowie die Sicherheit beim Austausch hinsichtlich Fehlern und Datenschutz [osi, 1996, S. 4]. Es



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

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 gedankliche Ebenen gegliedert. Die einzelnen abstrakten Ebenen sind in Abb. 2.9 dargestellt und fassen verschiedene Aufgaben des Kommunikationsprozesses in Teilaufgaben zusammen.

Die Ebenen werden als Schichten bezeichnet 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, wodurch diese Architektur sehr gut als Basis für offene Systeme dient. In Abb. 2.9 ist ebenfalls dargestellt, dass die Schichten eins bis vier gemeinsam auch als Übertragungsschichten beziehungsweise Transportsystem zusammengefasst werden, weil sie die Aufgabe der Datenübertragung zwischen Systemen übernehmen. 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 zwischen den Schichten werden als *Service Access Points* (SAP) bezeichnet und besitzen jeweils eine eindeutige Adresse. Die obere Schicht ist der *Service User*, der den Dienst der darunter liegenden Schicht nutzt, die des *Service*



Abb. 2.10: Die vier Dienstvorgänge

Providers. Für den Datenaustausch stehen Dienste zur Verfügung, welche in verbindungsorientierte und verbindungslose unterschieden werden. Verbindungslose Dienste benötigen keine Verbindung zwischen den Kommunikationspartnern zur Datenübertragung. Verbindungsorientierte Dienste benötigen für den Datenaustausch zunächst einen virtuellen Kanal zwischen Kommunikationspartnern. Typische Dienste sind beispielsweise das Aufbauen und Abbauen von Verbindungen sowie der eigentliche Datenaustausch [Schnell u. Wiedemann, 2006, S. 11f.].

Bei der Abhandlung der Dienstaufgaben anhand der Client/Server-Architektur, stehen vier grundlegende Dienstvorgänge zur Verfügung, die zusammengefasst in Abb. 2.10 abgebildet sind [mod, 2006a, S. 2f.]:

- Das Stellen einer Anforderung, ein Request,
- das Senden einer Meldung, eine Indication,
- das Senden einer Antwort, eine Response,
- und das Senden einer Bestätigung, eine Confirmation.

Die unterste physikalische Schicht definiert die mechanischen und elektrischen Schnittstellen zur physischen Verbindung von Systemen und zur Übertragung der einzelnen Bits [osi, 1996, S. 49f.]. Sie legt also die mechanischen und elektrischen Eigenschaften der Übertragung fest, also die Endsystemkopplung durch Stecker, die Kabelspezifikationen und die Zuordnung der Anschlüsse. Außerdem wird die Art der Codierung und die Spannungspegel der Übertragung spezifiziert. In der Regel werden dazu bestehende Normen genutzt, wie zum Beispiel die zuvor beschriebene elektrische Übertragungsstrecke nach EIA 485-Norm [Schnell u. Wiedemann, 2006, S. 14f.]. Ein wichtiger Aspekt der Ebene ist es, dass die Spezifikation der Schnittstelle und nicht das physikalische Medium selbst Teil der Schicht eins ist, da die Kommunikation unabhängig von der konkreten Ausprägung abläuft [Schnell u. Wiedemann, 2006, S. 9].

Die zweite Schicht widmet sich der Kommunikation zwischen zwei Systemen. Deshalb

stellt die Datenverbindungsschicht funktionale und prozedurale Hilfsmittel für den Aufbau und die Trennung einer Verbindung sowie den Transfer von Dateneinheiten zur Verfügung. Außerdem wird der darüber liegenden Netzwerkschicht eine Kontrolle über die Verbindung von physikalischen Netzwerken ermöglicht [osi, 1996, S. 46f.]. Eine weitere umfangreiche Aufgabe ist die Sicherung der Daten. Durch die Einteilung der Daten in Rahmen und die Nutzung von Strategien zur Fehlererkennung und -vermeidung, soll die Sicherheit gewährleistet werden. Dies kann durch die zuvor beschriebenen Methodiken, wie beispielsweise dem CRC geschehen. Wichtig hierbei ist, dass die Schicht keinerlei Kenntnis über Inhalte der Daten hat [Schnell u. Wiedemann, 2006, S. 9ff.]

Die dritte Schicht beschäftigt sich mit dem Netzwerk als Ganzes. Die Aufgaben der Netzwerkschicht hängen davon ab, ob der Datenaustausch verbindungsorientiert oder verbindungslos stattfindet. Daher steuert sie den Aufbau, die Erhaltung und das Trennen von Verbindungen sowie den eigentlichen Datenaustausch [osi, 1996, S. 41f.]. Weiterhin ist sie für den Transport der Daten innerhalb des Netzwerks zuständig. Sie kontrolliert den Datenverkehr im Netzwerk und legt eventuell Routen für die einzelnen Kommunikationsvorgänge fest [Schnell u. Wiedemann, 2006, S. 11f.].

Die vierte Schicht ist die Transportschicht und ist zuständig für eine transparente Übertragung der Daten. Sie beschäftigt sich nicht mit Transportrouten oder der Verlässlichkeit der Datenübertragung, sondern hat das Ziel einer optimalen Nutzung der Services der darunter liegenden Netzwerkschicht [osi, 1996, S. 37f.]. Typische Aufgaben sind die Adressierung der Teilnehmer, der Auf- und Abbau von Transportverbindungen sowie die Synchronisierung der datenaustauschenden Systeme. Außerdem werden die Daten aus der Sitzungsschicht in transportierbare Einheiten zerlegt [Schnell u. Wiedemann, 2006, S. 12f.].

Die fünfte Schicht – die Sitzungsschicht – startet eine Sitzungsverbindung mit eindeutiger Adresse, wenn diese von einer höheren Schicht angefordert wird. Diese Verbindung dient dazu, den Datenaustausch auf höherer Ebene zu organisieren, indem Sitzungsadressen mit den Transportadressen verknüpft werden [osi, 1996, S. 35]. Sie verbindet demnach die Anwendungsschichten mit dem Transportsystem über eine Schnittstelle [Schnell u. Wiedemann, 2006, S. 13].

Die sechste Schicht ist nach [osi, 1996, S. 33f.] mit der Darstellung der Daten betraut, die von beiden Kommunikationsanwendungen entweder verstanden oder referenziert werden. Dadurch wird eine gemeinsame Repräsentation der zu übertragenen Daten geschaffen. [Schnell u. Wiedemann, 2006, S. 13f.] stellt fest, dass diese Dienste eine Syntaxfreiheit der Daten beim Nachrichtenaustausch schaffen und durch eine Kompression der Daten Zeit- und Kostenvorteile der Übertragung generiert werden.

Die siebte und letzte Schicht stellt einem Benutzer lediglich eine Zugriffsmöglichkeit auf das OSI Kommunikationssystem zur Verfügung [osi, 1996, S. 32]. Benutzer sind in der Regel Softwareanwendungen [Schnell u. Wiedemann, 2006, S. 14].

Im nachfolgenden Abschnitt werden die später angewandten Modbus Protokolle und

deren Spezifikation beschrieben und in Bezug zum OSI-Modell gebracht, um den praktischen Nutzen davon zu verdeutlichen.

2.2.3 Modbus Kommunikationstechnologie

Zunächst wird die Modbus Technologie in das Open System Interconnection Modell eingeordnet, bevor eine Beschreibung der genauen Spezifikationen und Protokolle anhand der zuvor dargelegten Merkmalen erfolgt.

Das Modbus Protokoll teilt sich in verschiedene Protokolle auf, zum einen auf das *Application Layer Messaging Protocol* auf oberster Ebene, welches auf das *Modbus Over Serial Line Protocol* sowie das *Ethernet* und *TCP/IP Protokoll* aufbauen kann. 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 gemeinsamen Nutzung mit dem Modbus Over Serial Line Protocol, wird letzteres in die zweite Schicht eingeordnet und die Ebenen drei bis sechs sind leer implementiert. Als physikalische Schicht werden die Übertragungsstandards nach EIA 485 oder nach EIA 232 genutzt [mod, 2006b, S. 2].

Anstatt des seriellen Modbus Protokolls kann auch das *Modbus Messaging On TCP/IP Protocol* verwendet werden, dass dabei zusammen mit dem TCP/IP Protokoll zusammen die Netzwerkschicht im OSI-Referenzmodell implementiert. Die Ethernet Technologie implementiert dann die Datensicherungsschicht und definiert die Spezifikationen der untersten physikalischen Schicht und deren Schnittstellen. Die Ebenen vier bis sechs sind auch bei Nutzung dieser Kommunikationsweise leer implementiert [mod, 2012, S. 2f.]. Eine allgemeine Einführung zur Ethernet Technologie und zum TCP/IP Standard findet sich in Schnell u. Wiedemann [2006]. Für eine ausführliche Darstellung wird jedoch auf Furrer [2003] verwiesen.

Die Funktionen der einzelnen Ebenen werden im folgenden ausführlich dargestellt. Zunächst wird das Modbus Application Layer Messaging Protocol erläutert, dass von beiden unteren Modbus Protokollen genutzt wird. Anschließend werden letztere bis hin zu den physikalischen Schnittstellen erläutert.

Das Modbus Application Layer Messaging Protocol kann von verschiedenen Netzwerken und Bussystemen zur Master/Slave Kommunikationen genutzt werden [mod, 2012, S. 2f.]. Das Protokoll definiert eine gemeinsame Telegrammstruktur, bezogen auf die Inhalte und den Rahmen des Telegramms. Diese Definition ermöglicht die Kommunikation zwischen verschiedenen Geräten innerhalb eines Netzwerks, unabhängig von Art und Typ des darunter liegenden Netzwerks. Es beschreibt außerdem wie die Kommunikation abläuft. Zum einen, wie der Master beziehungsweise Client eine Anfrage an einen Server beziehungsweise Slave stellt, zum anderen deren Reaktionen und Antworten auf Anfragen. Außerdem beschreibt es die Überwachung auf Übertragungsfehler [MODICON, 96, S. 2f.]. Die Kommunikation kann über eine serielle Leitung nach EIA 485 beziehungsweise EIA 232 sowie über ein lokales Ethernet-Netzwerk



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

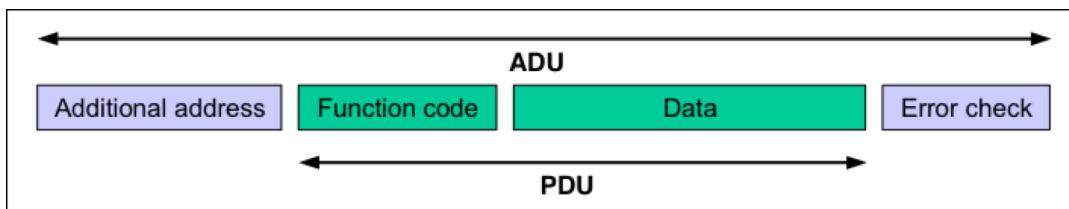


Abb. 2.12: Allgemeiner Rahmen für Telegramme nach dem Modbus Anwendungsprotokoll, entnommen aus [mod, 2012, S. 3]

erfolgen. Über Gateways, wie in Abschnitt 2.2 erläutert, kann die Kommunikation auch über verschiedene Typen von Bussystemen oder Netzwerken geschehen [mod, 2012, S. 3f.].

Das allgemeine Gerüst für Modbus Telegramme ist in Abb. 2.12 dargestellt. Die Protocol Data Unit (PDU) enthält die auszutauschenden Informationen und ist unabhängig vom Netzwerk oder dem Bussystem. Zu den Informationen zählen die eigentlichen Daten, welche auch leer sein können, sowie ein Funktionscode, der beschreibt welche Art der Reaktion gefordert wird. Die sogenannte Application Data Unit (ADU) enthält neben der PDU weitere Informationen zur Adressierung im Netzwerk und zur Fehlererkennung und ist daher abhängig von der zugrundeliegenden Netzwerkstruktur.

Die Kommunikation erfolgt anhand einer *Server/Client Architektur*, bei der der *Client* die Rolle des Masters und die *Server* die Rolle der Slaves übernehmen. Sie wird innerhalb des Clients durch eine ADU initialisiert. Das genaue Format der ADU wird durch die Wahl des Modbus RTU oder TCP Protokolls festgelegt. Das Schema einer Transaktion läuft jedoch bei beiden nach demselben Prinzip ab, welches in Abb. 2.13 graphisch dargestellt ist. Wenn eine ADU fehlerfrei empfangen wurde nutzt der Server den Funktion Code, um anzugeben ob ein Fehler aufgetreten ist. Ist dies der Fall, wird dies durch einen Exception Function Code angezeigt. Ansonsten sendet der Server eine normale Antwort, zusammen mit einem einfachen Echo des empfangenen Funktionscodes. Die Größe der PDU ist aufgrund der seriellen Kommunikation begrenzt auf 256 Bytes. Da zwei Bytes für einen Cyclic Redundancy Check und ein Byte für die Server Adresse reserviert sind, ist die PDU auf eine Länge von 253 Bytes begrenzt.



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

Ein weiterer, wichtiger Aspekt ist, dass das Modbus Anwendungsprotokoll die „big-Endian“ Codierung für Daten verwendet verwendet, deren numerischer Wert größer als ein einzelnes Byte ist [mod, 2012, S. 3ff.]. Anhand des Beispiels einer Uhrzeit, kann die Bedeutung der Big-Endian Repräsentation einfach erläutert werden: Die Daten werden zum Transport so aufgeteilt, dass zunächst die Daten mit der höchsten Wertigkeit, also den Stunden zuerst gesendet werden. Anschließend folgen die Minuten und zum Schluss die Sekunden, unabhängig von deren numerischem Wert. Werden nun nacheinander die Telegramme 03, 50, 12 empfangen, werden diese als 03:50:12h interpretiert. Der interessierte Leser wird für nähere Ausführungen, auch zu little-Endian Codierung, an Bertrand Blanc [2005] verwiesen.

Die Funktionscodes umfassen ein Byte und sind wie bereits angesprochen in der PDU enthalten. Es existieren 255 verschiedene Codes die von 1 aufwärts gezählt werden. Sie definieren welche Aktion ein Server ausführen soll. In erster Linie dienen sie dem Datenzugriff in verschiedenen Tabellen, können aber auch für Diagnosen oder nutzerdefinierte Aktionen benutzt werden. Weiterhin lassen sie sich in drei große Gruppen untergliedern, die öffentlichen, nutzerdefinierbaren und die reservierten Funktionscodes. Die öffentlichen Codes sind wohldefiniert und dokumentiert sowie auf Konformität getestet und sind daher einfach, schnell und sicher nutzbar. Die vom nutzerdefinierbaren Funktionscodes können genutzt werden um die öffentlichen Codes durch eigene Funktionaltäten zu ergänzen. Die reservierten Codes sind für Altprodukten einiger Unternehmen reserviert, welche an der Entwicklung des Modbus Protokolls beteiligt waren [mod, 2012, S. 10ff.].

Das Modbus Datenmodell basiert darauf, dass durch den Funktionscode und auf die Daten in vier verschiedenen Tabellen für unterschiedliche Funktionen und Datenobjekte zugegriffen werden kann:

- In die *Discrete Input* Tabelle, welche Single Bit Objekte enthält und lediglich gelesen werden kann,



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

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

Ein 16-Bit Wort entspricht einer Folge von 16 binären Symbolen. Nach der *IEC 61131-3 Norm* entspricht es im Rechner einem Integer Wert und es lassen sich damit ganze Zahlen zwischen 0 und 65.536 beziehungsweise -32.768 und 32.767 darstellen. Die Daten selbst können auch innerhalb einer einzigen Tabelle abgelegt sein, es muss lediglich Interface für den Zugriff auf die vier verschiedenen Tabellen möglich sein. Jede Tabelle ist begrenzt auf maximal 65536 Einträge, deren Adressierung bei beginnt 0 und bei 65535 endet [mod, 2012, S. 6ff.]. An welcher Stelle welche Daten zu finden wird vom Hersteller eines modbusfähigen Geräts definiert und üblicherweise durch ein Modbusadressmapping beschrieben. Ein Beispiel dazu findet sich in den Datenblätter zu den Komponenten der Anlage in Anhang ???. Um also auf die Daten einer modbusfähigen Komponente zugreifen zu können, muss durch den Funktionscode die richtige Tabelle und durch die Adresse wiederum die richtige Stelle in der Tabelle referenziert werden.

Beim *Modbus Over Serial Line Protocol* können die Daten über zwei unterschiedliche Modi übertragen werden, dem *RTU* und *ASCII Modus*. Der ASCII Modus stellt eine

exotische Anwendung dar und wird in mod [2006b] detailliert beschrieben. Der RTU Modus findet bei der Kommunikation der Anlage in Kapitel 3 Anwendung und wird von allen seriellen modbusfähigen Komponenten unterstützt. Das Protokoll spezifiziert das folgende Format zur Übertragung der einzelnen Bytes: Jede Byteübertragung beginnt mit einem *Startbit*, auf dass dass zu übertragende Byte, bestehend aus acht einzelnen Bits, folgt. Abgeschlossen wird die Übertragung optional von einem *Paritätsbit* und einem *Stoppbit* beziehungsweise lediglich von zwei StoppBits. 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 bei der Byteübertragung, wie bereits zuvor in Abschnitt 2.2 erläutert. Eine graphische Darstellung dieses Vorgangs findet sich in Abb. 2.15. Der *Rahmen* eines Modbus RTU Telegramms besteht aus der *Slave Adresse*, die für jeden Slave innerhalb eines Netzwerks eindeutig ist und einen numerischen Wert zwischen 1 und 247 besitzt. Anschließend folgt der *Function Code*, der den Tabellenzugriff definiert. Darauf folgen die eigentlichen Informationen für die 0 bis maximal 252 Bytes vorgesehen sind. Abgeschlossen wird der Rahmen durch ein *CRC Feld* zur Fehlerüberprüfung. Der Ablauf und die Vorgänge des CRC Checks beim RTU Protokoll sind detailliert in mod [2006b] beschrieben. Die Übertragung eines Telegramms erfolgt byteweise, wie zuvor beschrieben. Die Datensicherung findet also durch eine Paritätsprüfung und einen CRC auf verschiedenen Ebenen statt. Die *Übertragungszeit* eines Bytes und eines Telegramms hängt wiederum von der *Baudrate* ab. Um den Beginn und den Abschluss eines RTU Rahmens eindeutig zu definieren, geschieht dies in Abhängigkeit von der Baudrate. Zwischen einzelnen Bytes folgt ein stilles Intervall, dass je nach Länge angibt, ob das Telegramm beendet ist oder fortgesetzt wird. Auf ein stilles Intervall das kleiner oder gleich der anderthalbfachen Übertragungszeit eines Bytes ist, folgt eine weiteres Byte. Ist das stille Intervall länger als die dreieinhalbfache Byteübertragungszeit markiert dies das Ende eines Telegramms und den Beginn des nächsten Telegramms [mod, 2006b, S. 13]. Diese Zusammenhänge sind zur Veranschaulichung in Abb. 2.15 zusammengefasst.

Der Implementierungsleitfaden innerhalb des Protokolls legt auch die Spezifikationen der physikalischen Schicht fest, die im Folgenden beschrieben werden. Er wird eine EIA 483 Schnittstelle als elektrisches Interface vorgeschlagen, erlaubt aber weiterhin die Nutzung einer EIA 232 Schnittstelle zur Datenübertragung. Bei beiden erfolgt die Datenübertragung über ein verdrilltes Leiterpaar. Weiterhin wird eine Baudrate von 9.600 und 192.000 bei einer Even Parität als *Standardkonfiguration* des seriellen Netzwerks festgelegt. Die Verkabelung der Komponenten erfolgt bei beiden elektrischen Standards standmäßig durch ein verdrilltes Leiterpaar und einer gemeinsamen Verbindungsleitung. Die beiden Leitungen des verdrillten Paars werden mit *D1*, welche auch *D+* oder *Modbus A* Leitung genannt wird, und *D0*, welche auch *D-* oder *Modbus B* Leitung genannt wird, bezeichnet. Ein Standard-Netzwerk besteht aus maximal 32 Teilnehmern, dass durch den Einsatz von Repeatern vergrößert werden kann. Das Netzwerk wird durch eine Bus-Struktur ausgeführt, nach der die einzelnen



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

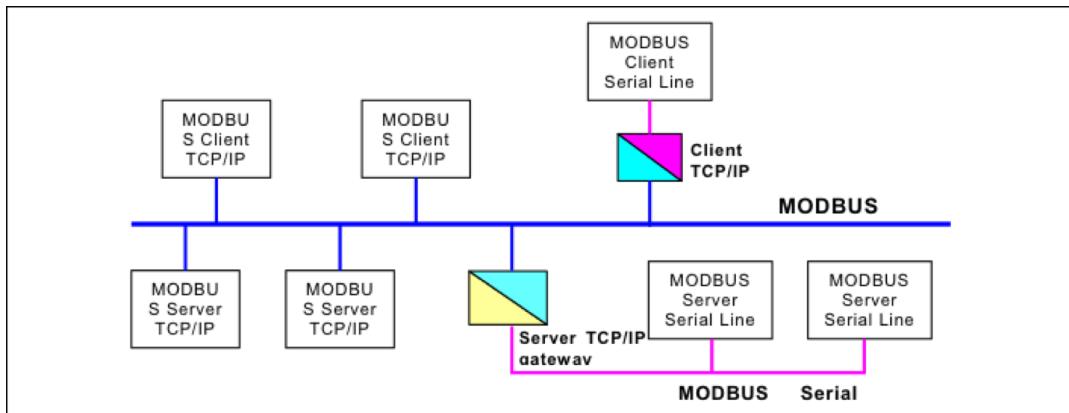


Abb. 2.16: Eine Modbus TCP/IP Kommunikationsarchitektur, entnommen aus [mod, 2006a, S. 4]

Komponenten im Netzwerk angeordnet werden. Die beiden Enden der Busleitung werden durch einen Widerstand von 150 Ohm zwischen der D0 und D1 Leitung abgeschlossen [mod, 2006b, S. 20ff.]. Die Verbindung der Kabel kann im einfachsten Fall durch einfache Schraubklemmen erfolgen, jedoch können auch genormte mechanische Interfaces genutzt werden, deren Ausführung und Verkabelung in [mod, 2006b, S. 29ff.] detailliert beschrieben sind.

Das *Modbus Messaging on TCP/IP Protocol* ermöglicht die Kommunikation von Geräten, die über ein Ethernet miteinander verbunden sind. Des Weiteren ermöglicht dieses Protokoll explizit Netzwerksegmente über Bridges, Gateways oder Router Netzwerke miteinander zu verbinden. Das Protokoll erlaubt das Einbinden serieller Netzwerksegmente und eine übergreifende Kommunikation [mod, 2006a, S. 2f.]. Eine beispielhafte Kommunikationsarchitektur ist in Abb. 2.16 dargestellt.

Das Modbus TCP Protokoll nutzt ein Master/Slave Buszugriffsverfahren, die Kom-



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

munikation hingegen ist anhand einer Client/Server Architektur aufgebaut. Dabei stellen lediglich die Clienten Anfragen, die vom angesprochenen Server beantwortet werden. Der Master schlüpft hierbei in die Rolle eines alleinigen Clienten. Außerdem weicht die ADU leicht von der zuvor vorgestellten Version ab, wie in Abbildung Abb. 2.17 dargestellt. Die ADU enthält neben der PDU einen *Modbus Application Protocol Header* (MBAP), der sieben Bytes lang ist und die zusätzlichen Informationen zur Datenübertragung enthält. An erster Stelle im Header steht der *Transaction Identifier*, der der Identifikation der Modbus TCP Telegramme dient. Darauf folgt der *Protocol Identifier* der einen Wert von Null für das Modbus Protokoll annimmt. Das *Length* Feld gibt durch einen Bytecount an, aus wie vielen Bytes das Telegramm besteht. Der Header wird durch den *Unit Identifier*, analog zur Slave ID, abgeschlossen. Der Port 502 ist für Modbus Kommunikation reserviert, jedoch können auch andere Ports genutzt werden falls die Modbusgeräte eine Portkonfiguration unterstützen [mod, 2006a, S. 4f.]

Der Modbus Client initiiert den Informationsaustausch, indem er eine ADU erstellt und an den TCP Port 502 übermittelt. Die Modbus Server warten auf Anfragen über den TCP Port 502 [mod, 2006a, S. 7f.]. Das *Transmission Control Protocol* (TCP) und das *Internet Protokoll* (IP) übernehmen die Netzwerkaufgaben, entsprechend der Netzwerkschicht des OSI Modells. Dazu gehört die Bereitstellung eines Telegrame-services sowie das Einteilen der Daten in Transportdatenblöcke, wobei der MBAP Headers an jedes Paket angehängt wird [mod, 2006a, S. 7ff.].

Das Verbindungsmanagement wird vom TCP übernommen und ist sehr wichtig, da die Kommunikation verbindungsorientiert erfolgt. Das Management kann durch ein externes Modul oder durch die Nutzeranwendung selbst erfolgen und umfasst den aktiven und passiven Auf- und Abbau von Verbindungen [mod, 2006a, S. 11ff.]. Zum Datenaustausch erfolgt der Aufbau einer Verbindung über das Ethernet IP Protokoll. Diese wird über die eindeutige IP Adresse des Gerätes im Netzwerk, die Port und *Socket* Nummer hergestellt. Der Socket ist lediglich ein Endpunkt innerhalb eines Rechners, der einem Port eindeutig zugewiesen ist und über den die Kommunikation abläuft [mod, 2006a, S. 15f.]. Der interessierte Leser findet eine fundierte Einführung in die beiden Standards bei Furrer [2003].

Der Leitfaden zur Implementierung des Modbus TCP Protokolls weist außerdem darauf hin, dass eine Verbindung nicht für jede einzelne Transaktion auf- und abgebaut werden muss, sondern endlos viele Modbus Transaktionen während einer Sitzung stattfinden

können. Außerdem sollte die Anzahl an Verbindungen eines Server auf eine Minimum beschränkt werden [mod, 2006a, S. 9f.]. Die physikalischen Schnittstellen werden durch das Ethernet als handelsüblicher Netzwerkanschluss spezifiziert.

2.3 Technische Grundlagen zur Modellbildung

In diesem Abschnitt werden die technischen Grundlagen zur Bildung eines mathematischen Modells eines Raumes in Kapitel 4 erläutert.

2.3.1 Thermodynamische Systeme

Für das Raummodell müssen Energieströme – genauer betrachtet Wärmeströme – untersucht werden. Um diese thermodynamischen Vorgänge mithilfe 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, der an den Grenzen von der Umgebung umgeben ist. Sie dienen dem Zweck der Bilanzierung von Massen- und Energieströmen. Die begrenzenden Flächen können gedanklicher, physischer oder beider Art zugleich sein, wichtig ist 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 beispielsweise 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 so gelegt, dass über sie hinweg keine beziehungsweise keine relevanten⁴ Flüsse von Materie und Energie stattfinden [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

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

System kennzeichnen. Die Variablen, welche den *Zustand* eines Systems bestimmen, werden diese als *Zustandsgrößen* bezeichnet [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 Raummodell 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 die gleiche Ausprägung besitzt [Baehr u. Kabelac, 2012, S.15].

Da wir im Rahmen der Modellbildung Zustände betrachten, müssen auch deren 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 Zustandes eines Systems mit einer 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 Beispiel durch das Aufheben innerer Hemmungen oder dem Wegfall von äußerer Zwängen. Diese Prozesse laufen in abgeschlossenen Systemen meist von selbst ab und streben einen ausgeglichenen, also homogenen, Endzustand an. *Ausgleichsprozesse* dienen 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, beispielsweise 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.].

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

2.3.2 Erster Hauptsatz der Thermodynamik

Der erste Hauptsatz der Thermodynamik wird zunächst als allgemeiner Energieerhaltungssatz formuliert und anschließend angewandt, 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 dadurch 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 wird 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 weder erzeugt noch vernichtet, 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 der 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]:

$$\Delta U = Q + W + m_{in} * c_p * T_{in} - m_{out} * c_p * T_{out}$$

beziehungsweise in differentieller Form

$$\frac{dU}{dt} = \dot{U} = \dot{Q} + P + \sum \dot{m}_{in} * c_p * T_{in} - \sum \dot{m}_{out} * c_p * T_{out}$$
(Gl. 4)

2.3.3 Wärmeübertragung

Wie zuvor erwähnt haben Wärmeströme eine zentrale Bedeutung bei der Modellbildung, weshalb ein genauere Betrachtung folgt.

Die Definition von Wärmeübertragung ist nach [Böckh u. Wetzel, 2014, S. 1] „[...] der Transfer der Energieform Wärme aufgrund einer Temperaturdifferenz. „. Die Wärmeübertragung kann nach Nußelt⁶ grundsätzlich durch zwei verschiedene Arten stattfinden [Böckh u. Wetzel, 2014, S. 3f.]:

- Wärmeübertragung durch Strahlung, bei der die Übertragung von Wärme ohne stofflichen Träger, lediglich durch elektromagnetische Wellen zwischen Oberflächen erfolgt. Weil diese Art der Wärmeübertragung keine Relevanz für die weiteren Betrachtungen hat wird er interessierte Leser für eine fundierte Einführung an Böckh u. Wetzel [2014] verwiesen.
- Wärmeübertragung 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 hängt sie von der Strömung der Fluide ab, unabhängig davon ob erzwungene oder freie Konvektion. Die Konvektion ist ein Effekt, der zusätzlich zur reinen Wärmeleitung auftritt ebenfalls nicht relevant für den weiteren Verlauf ist und detaillreich von Böckh u. Wetzel [2014] beschrieben wird. Erfolgt der Wärmetransport stationär, dass heißt von äußeren Anregungen bedingt und unabhängig von der Zeit, lässt er sich qualitativ einfach als konstanter Wärmestrom \dot{Q} beschreiben. Er gibt, an wie viel Wärme pro Sekunde übertragen werden [Böckh u. Wetzel, 2014, S. 5ff.]. Der Wärmestrom ist wie zuvor bereits erwähnt von den Stoffeigenschaften abhängig, welche von der Wärmedurchgangszahl U – Wert⁷ und der Austauschoberfläche $A_{exchange}$, an welcher

⁶ Beschrieben in seinem Aufsatz „Das Grundgesetz des Wärmeüberganges“, 1915.

⁷ 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,

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.

„*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 allgemeingültigen Anforderungen an eine Anlage analysiert und schließlich verschiedene Vorgaben von Seiten der Hochschule Karlsruhe spezifiziert und ergänzt. Im darauffolgenden Schritt wird eine Idee abgeleitet, die anschließend zu einem Konzept weiterentwickelt und in ein konkretes Anlagendesign umgesetzt wird. Einzelnen Anlagenteile und deren Funktionsweisen werden zunächst näher beschrieben und im Anschluss auf die realen Einsatzbedingungen ausgelegt. Abschließend wird die Installation und dabei aufgetretene Besonderheiten der Anlage erläutert.

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, werden zunächst die Einsatzziele der Anlage definiert. Wie in Kapitel 1.1 erwähnt, ist die hier betrachtete Anlage als Teil einer großen Anlage gedacht, daher sollten die Einsatzziele kompatibel zueinander gewählt werden. Um die Kompatibilität zu gewährleisten, wurde mit den Projektverantwortlichen⁸ der 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 den zahlreichen Schnittstellen.
- Die Anlage soll eine hohe Funktionalität und Robustheit gegenüber Fehlern und Beschädigungen vorweisen, da bei der Einarbeitung eine erhöhte Wahrscheinlichkeit der Fehlbedienung besteht und Schäden dadurch vermieden werden sollen.

⁸ In Person von Herrn ADRIAN BÜRGER und MARKUS BOHLAYER

- Es soll ein Vergleich verschiedener Methodiken 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.
- Die Anlage soll flexibel ansteuerbar und erweiterbar sein, sodass der Grad der Komplexität verändert werden kann.
- Die Anlage soll ermöglichen den Einfluss der Sonneneinstrahlung auf die Raumtemperatur zu untersuchen.
- Im Rahmen der Anwendungsforschung soll das Modell des betrachteten Raumes zur Temperaturregelung möglichst nahe der Realität entsprechen.

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

Neben den oben genannten Einsatzzielen wurden von Seiten der Hochschule Karlsruhe⁹ weitere Rahmenbedingungen definiert, die im Folgenden dargestellt sind:

- Der Raum K004b 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 realisierbar sein.
- Für die Kommunikation zwischen den Anlageteilen 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

Aus den Einsatzzielen und Rahmenbedingungen lassen sich die Anforderungen an die Anlage ableiten, welche im Nachfolgenden explizit aufgeführt sind. Aus Gründen der Übersichtlichkeit sind die wichtigsten in der nachfolgenden Tabelle Tab. 3.1 zusammengefasst.

⁹ In Person von Frau Professor ANGELIKA ALTMANN-DIESES, Herrn Professor MARCO BRAUN und Herrn ADRIAN BÜRGER

Einsatzziele und Rahmenbedingungen	Anforderungen
Raum K004b als Umgebung	<ul style="list-style-type: none"> - Anpassen der Anlage an Raum K004b.
Minimaler baulicher Aufwand	<ul style="list-style-type: none"> - Nutzen bestehender Heizkörper anstatt einer Klimatisierung des Raumes.
Minimaler finanzieller Aufwand	<ul style="list-style-type: none"> - Die Anlage soll auf die wesentlichen Funktionalitäten und eine minimale Anzahl an Komponenten beschränkt sein.
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. - Die Ansteuerung der Anlage soll innerhalb des gesamten lokalen Netzwerks über Modbus TCP möglich sein.
Flexible Ansteuerung der Anlage	
Einarbeitung in die Thematiken:	
<ul style="list-style-type: none"> - Modellbildung - Kommunikation technischer Systeme - Modellprädiktive Regelung 	<ul style="list-style-type: none"> - Komplexität ist notwendig, sie darf jedoch nicht zu hoch sein. - Eine klare Struktur mitsamt einer scharfen Trennung ist erwünscht. Diese soll durch wenige thematische Überschneidungen erreicht werden.
Know-how für Kommunikation technischer Systeme	
Vergleich von Ergebnissen durch:	
<ul style="list-style-type: none"> - Die Variation von Steuerungsparametern - Den Verwendung verschiedener Regelungsmethodiken - Den Einsatz verschiedener Algorithmen. 	<ul style="list-style-type: none"> - Die Reaktion des Systems soll einfach und zugleich kostengünstig erfasst werden können. - Einfachen robusten Bauteile sollen zum Einsatz kommen. - Ziel ist die Erstellung eines wartungsarmen Systems.
Hohe Funktionalität und Robustheit	<ul style="list-style-type: none"> - Das Systems soll modular und ohne großen Aufwand für weitere Schritte erweiterbar sein.
Erweiterbarkeit der Anlage	

Tab. 3.1: Umsetzung der Ziele in Anforderungen der Anlage

Die grundlegendste Anforderung der Planung, ist die Anpassung an die Gegebenheiten des Raumes K004b, welche in Abb. 3.1 skizziert sind. Der Raum befindet sich 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 daher im Folgenden als Außenwand bezeichnet. Die östliche und nördliche Wand sowie die Decke und der Boden des Raumes grenzen an weitere Innenräume des K Gebäudes. In der südlichen Außenwand ist eine Fensterfront mit Jalousien eingebaut, direkt darunter ist ein Heizkörper installiert.



Abb. 3.1: Raumskizze K004b vom K Gebäude der Hochschule Karlsruhe – Technik und Wirtschaft

Durch die Raumwahl werden bereits zwei wichtige Anforderungen erfüllt, denn durch die Fensterfront kann der Einfluss der Sonneneinstrahlung auf die Raumtemperatur untersucht werden. Darüber hinaus kann der bestehende Heizkörper in die Anlage integriert werden und so einen minimalen baulichen Aufwand sicherstellen. Da eine Klimatisierung zur Raumtemperaturregelung mit einem erheblichen finanziellen und baulichen Aufwand verbunden wäre, kann diese zunächst ausgeschlossen werden. Um der Forderung nach einer erweiterbaren Anlage nachzukommen, soll bei der Planung ein möglicher Nachrüstungsvorgang durch eine Klimatisierung und eine Ansteuerung der Jalousien trotz alledem berücksichtigt werden.

Der Raum K004b wird als Büro von wissenschaftlichen Mitarbeitern der Hochschule Karlsruhe genutzt, daher befinden sich darin, wie in Abb. 3.1 abgebildet, sechs Computerarbeitsplätze und das übliche Büro-Mobiliar. Durch den Einfluss von Mensch und Rechner, sogenannten Störgrößen welche im Modell berücksichtigt werden müssen, wird die Anforderung einer möglichst anwendungsnahen Umgebung erfüllt.

Als weitere restriktive Vorgabe soll die Modellprädiktive Regelung unter Zuhilfenahme der Plattform JMODELICA.ORG erfolgen. 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, welche bisher noch experimenteller Art ist [AB, 2015, S. 1f.]. Der Compiler kann die Modelle und Optimierungsprobleme in verschiedene Formate übersetzen. Zum einen in einen 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 in ein *OptimizationProblem* Objekt transferiert werden, welches eine symbolische Repräsentation des Optimierungsproblems darstellt [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 gut für die gradientenbasierte numerische Optimierung von nichtlinearen Problemen aufgrund seiner effizienten Ableitungserzeugung durch die Algorithmische Differentiation und der Möglichkeit zur Integration von gewöhnlichen Differenzialgleichungen. Die Interaktion mit dem Nutzer soll aus Gründen der Stabilität über die Schnittstelle in Python erfolgen [Joel Andersson, 2015, S. 5f.]. Daraus lassen sich weitere Anforderungen an das Modell ableiten, die in Kapitel 4 näher erläutert werden.

Die vorgegebene Softwareplattform zur Modellprädiktiven Regelung und deren einzelne Komponenten besitzen allesamt eine Schnittstelle für Python, daher liegt es nahe, dass sowohl die Ansteuerung als auch die Kommunikation der gesamten Anlage in Python erfolgen. Python eignet sich hervorragend für diese Aufgaben, da es frei erhältlich ist und einen modularen Aufbau vorweist. Durch die Open-Source Lizenzierung ist die Programmiersprache frei nutzbar und bietet sowohl durch ihre Standardbibliothek als auch durch eine Vielzahl an von Nutzern entwickelten Bibliotheken, sogenannten Paketen, unzählige Anwendungsmöglichkeiten. Ein Beispiel für ein derartiges Paket ist das pysolar Paket, das im Rahmen der Modellbildung eine Umrechnung der an einem Fenster gemessenen in die wirkende Solarstrahlung ermöglicht [van Rossum u. the Python development team, 2016, S. 2f.].

Die Vorgabe der Modbus Kommunikationsprotokolle für die Kommunikation ist 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 einen möglichst geringen Komplexitätsgrad aufweisen, damit eine

Einarbeitung in die einzelnen Themengebiete der Modellbildung, der Kommunikation von technischen Systemen und der Modellprädiktive Regelung mit einem überschaubaren Aufwand ermöglicht wird. Dies soll durch eine klare Abgrenzung der verschiedenen Anlagenteile und deren Funktionen, wie auch durch die Aufbaustruktur der Anlage erreicht werden. Im Gegensatz hierzu steht die Forderung nach dem Aufbau von Know-how auf dem Gebiet der Kommunikation technischer Systeme. Es muss daher ein Kompromiss zwischen Verständlichkeit und Komplexität gefunden werden. Um beiden Forderungen gerecht zu werden, findet die Kommunikation der Anlageteile ausschließlich über Modbus und unter Nutzung von verschiedenen Hard- und Software-Schnittstellen statt. Dadurch ist es möglich bereits erste Erfahrungen zu sammeln, die für die Inbetriebnahme der solaren Klimatisierungsanlage hilfreich sind. Um eine Vergleichbarkeit von Ergebnissen zu gewährleisten, soll das System auf eine Veränderung der Steuergrößen eine schnelle¹⁰ und zugleich unmittelbar messbare Reaktion zeigen. Daraus lässt sich folgern, dass die Aktorik ohne zeitliche Verzögerung auf Steuersignale ansprechen und die gesamte Anlage einen unmittelbaren Einfluss auf die vorherrschende Raumtemperatur haben soll. Die Messung der Raumtemperatur kann mithilfe von handelsüblichen Temperatursensoren erfolgen. Diese erfüllen die vorgegebenen Anforderungen der Hochschule Karlsruhe, da sie keinen allzu großen technischen und monetären Aufwand darstellen.

Als Basis für fortführende wissenschaftliche Tätigkeiten soll ein möglichst fehlerfreies System mit hoher Funktionalität erschaffen werden, sodass Fehlerquellen innerhalb des Systems von Vornherein ausgeschlossen werden können und damit nicht den Fokus von den eigentlichen Forschungsaktivitäten lenken. Zusätzlich ist eine hohe Robustheit der Anlage gegenüber Bedienungsfehlern und Beschädigungen erforderlich, da sie für Test- und Entwicklungszwecke genutzt werden soll. Dies lässt sich durch den Einsatz von wartungsarmen, einfachen und robusten Bauteilen sowie den Aufbau der Anlage berücksichtigen und steht zugleich im Einklang mit der Forderung nach einer minimalen finanziellen Belastung.

Python stellt, wie bereits erwähnt, Schnittstellen und Funktionalitäten für die Aufgaben der Anlage zur Verfügung. Darüberhinaus bietet sich Python, durch ihre weitgehende Unabhängigkeit vom Betriebssystem, als zentrales Steuerungstool an. Hierdurch ist es einem nachfolgenden Schritt möglich einen Controller durch einen vergleichsweise günstigen Einplatinenrechner, wie beispielsweise durch einen Raspberry Pi, zu realisieren. Aus den eben genannten Gründen basiert die folgende Planung und Auslegung der Anlage auf einer zentralen Steuerung in Python.

¹⁰ Im Kontrast zu den langsamen Reaktionen der solaren Klimatisierungsanlage, welche im oberen Minutenbereich liegen, bedeutet schnell in diesem Kontext im unteren Minutenbereich.

3.2 Idee und Ziel der Anlage

3.2.1 Aufgabe der Anlage

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

Dazu gilt es zunächst die Temperatur innerhalb des Raumes K004b 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 wird über das Einlassventil gesteuert, hängt jedoch auch von weiteren Eigenschaften des Heizkörpers ab. Sie wird von einem Wärmemengenzähler gemessen und mittelbar über den Massenstrom und die Temperaturen des Heizwassers am Ein- und Auslass bestimmt. Zur Messung werden ein Durchflusssensor und zwei Temperatursensoren an den jeweiligen Enden des Heizkörpers eingesetzt. Um die Heizleistung zu steuern muss demnach der Massenstrom des Heizwassers gemessen und gesteuert werden und zugleich beide Temperaturen am Heizkörper bekannt sein. Der Massenstrom kann mit Hilfe eines Stellantriebs über ein Ventil eingestellt werden. Aus der Temperatur am Einlass und den Eigenschaften des Heizkörpers, lässt sich die Temperatur am Auslass und damit die aktuelle Heizleistung bestimmen. Durch diesen Zusammenhang wird eine gezielte Beeinflussung der Raumtemperatur ermöglicht.

Nachdem nun die Raumtemperatur bekannt ist und eine Möglichkeit einer Manipulation besteht, wird weiterhin eine intelligente Steuerung benötigt. Durch diese soll der Heizkörper ressourcenschonend eingesetzt und das übergeordnete Ziel der Temperaturregelung erreicht werden.

3.2.2 Idee der Anlage

Die oben genannte, umfassende Aufgabe wird von einer Anlage übernommen, die sich 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.

Als zentrale Komponente der Anlage muss der logische Controller Python unterstützen und die gesamten Steuerungs- und Kommunikationsaufgaben übernehmen. Darüber hinaus müssen im Rahmen der Modellprädiktiven Regelung in regelmäßigen Zeitabständen wiederholt Optimalsteuerungsprobleme gelöst werden, weshalb eine ausreichend große Rechenkapazität benötigt wird. Um diese 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 zuvor beschrieben besitzt die Sensorik den größten Umfang, da sie neben der Raumtemperatur auch den Massenstrom und die Temperaturen am Heizkörper erfasst. Innerhalb des Raumes K004b wird die Innentemperatur durch mehrere Sensoren erfasst, um die unterstellte Homogenität zu überprüfen und eine belastbare Raumtemperatur für die Modellberechnungen herauszufinden. Für die Erfassung der Heizleistung kommt ein Wärmemengenzähler zum Einsatz, der 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 auf diese 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 Ziele und konkretisiert die zuvor geschilderte Idee. Das Konzept gliedert sich zunächst in die Netzwerkarchitektur und die Teifunktionen der Anlage, sowie in die Beschreibung der eingesetzten Bauteile. Die Teifunktionen der Anlage beinhalten die Erfassung der Raumtemperatur wie auch die Steuerung des Heizkörpers. Abschließend wird die Um-

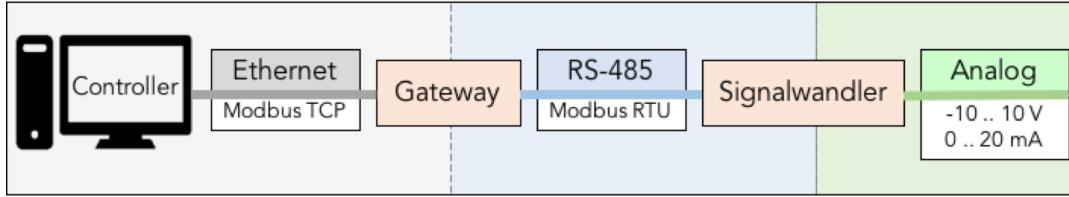


Abb. 3.3: Aufbau des Netzwerks

setzung und die softwareseitige Steuerung der Anlage am Beispiel der Inbetriebnahme näher erläutert.

3.3.1 Netzwerkarchitektur

Der Rechner stellt den Kern der Anlage dar, deshalb werden darauf aufbauend die Kommunikationsleitungen und der Aufbau des Netzwerk geplant. Die Kommunikation erfolgt gemäß den Spezifikationen der beiden Modbus Protokolle RTU und TCP, deren Besonderheiten im Abschnitt 2.2.3 näher beleuchtet wurden. Da der Rechner standardmäßig mit einem Ethernet-Netzwerkanschluss ausgestattet ist, wird er über ein Netzwerkkabel mit dem ersten Subnetzwerk verbunden. Dieses Netzwerk stellt ein lokales Netzwerk dar, 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. Es erlaubt somit den Einsatz des Modbus RTU Protokolls. Die beiden Netzwerke werden durch ein Gateway miteinander verbunden, welches die Übersetzung der Kommunikation in beide Richtungen übernimmt. Die Übersetzung umfasst die Umwandlung der elektrischen Signale und der verschiedenen Modbus Telegrammformate ineinander.

Das eingesetzte EX9132C-2-MTCP Gateway von EXPERTDAQ bietet hierzu eine einfache und zugleich kostengünstige Möglichkeit, um von einem Modbus TCP/IP Clienten mit Modbus RTU Servern zu kommunizieren. Es verfügt über zwei serielle Ports, einem EIA-232 und einem EIA 485 Port. Die genauen Spezifikationen können dem Datenblatt im Anhang C.1 entnommen werden.

Im dritten Subnetz findet die Kommunikation über analoge Spannungssignale und Stromsignale statt, welche von einem Signalwandler erzeugt werden. Der Signalwandler wiederum ist kein Gateway, da die Spannungssignale nicht 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 preiswert mithilfe von Raumtemperaturführlern gemessen werden. Der Messwert soll anschließend über eine der beiden Modbus Schnittstellen dem Controller zur Verfügung gestellt werden. Innerhalb der Anlage

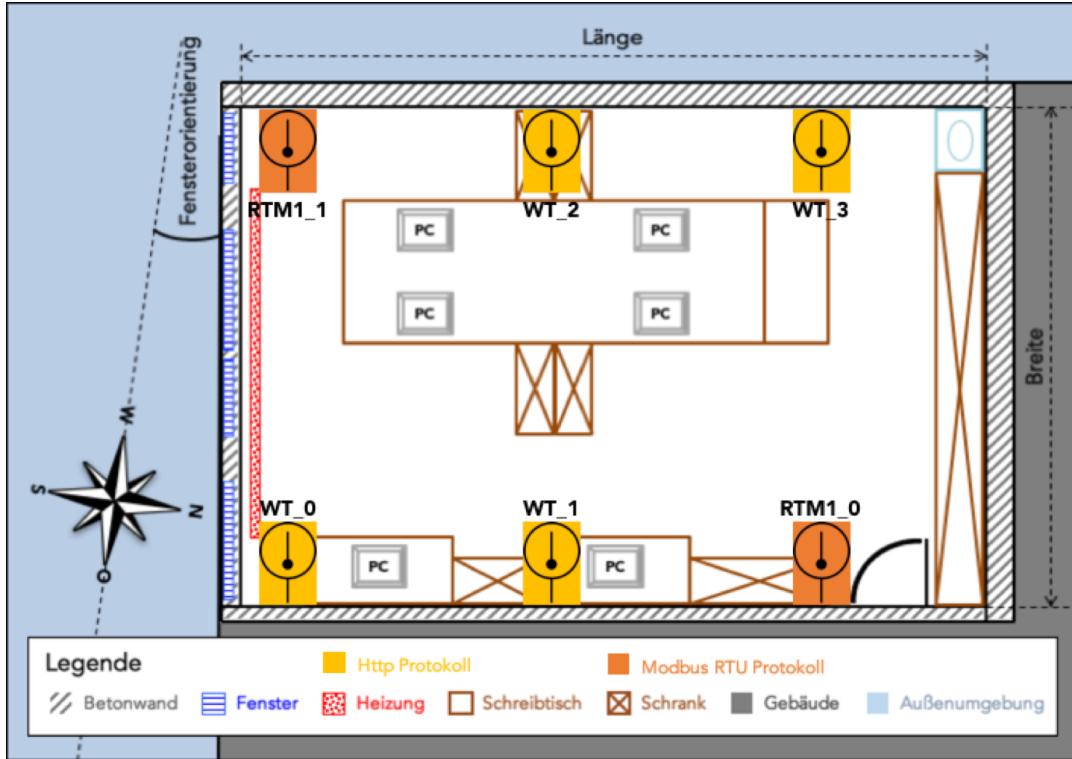


Abb. 3.4: Verteilung der Raumtemperaturfühler

kommen zwei THERMASGARD RTM1-MODBUS Raumtemperaturfühler ohne Display von S+S REGELTECHNIK zum Einsatz, da sich diese aufgrund weiterer vorteilhafter Eigenschaften, wie zum Beispiel ihrer Kalibrierfähigkeit, auszeichnen. Die detaillierten Funktionen und Daten der Raumtemperaturfühler können dem Datenblatt im Anhang C.3 entnommen werden.

Ursprünglich sollten die beiden Temperatursensoren auf einer mittig gedachten Achse im Raum in nord-südlicher Richtung jeweils am Ende des in der Mitte stehenden Schreibtischs platziert werden. Da jedoch während der Installation der Anlage von Seiten der Hochschule Karlsruhe vier weitere Temperatursensoren zusammen mit einem Messumformer zur Verfügung gestellt wurden, wurde eine veränderte Anordnung der Sensoren gewählt. Bei den Sensoren handelt es sich um vier PT1000 Temperatursensoren, die über den WEBTHERMOGRAPH 8X von WUT die Messwerte zur Verfügung stellen. Da der Webthermograph keine Modbus Schnittstelle besitzt, wird er über ein Netzwerkkabel in das Ethernet Netzwerk integriert und kann unter Verwendung des HTTP Protokolls vom Controller ausgelesen werden. Um Messwerte mit dem Controller auszulesen, bietet sich das Python Paket *httplib* an, welches das HTTP Protokoll implementiert und Grundfunktionen zur Verfügung stellt. Als veränderte Anordnung der Temperatursensoren wurden jeweils drei Sensoren an den gegenüberliegenden Wänden Richtung Osten und Westen platziert. 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 ihrer Anordnung in Abb. 3.4 visualisiert sind.

3.3.3 Steuerung des Heizkörpers

Um die Heizung mit den benötigten Sensoren auszustatten, bietet sich wie in Abschnitt 3.2.2 erwähnt ein Wärmemengenzähler an. Die Temperaturmessung am Ein- und Auslass der Heizung gestaltet sich jedoch etwas aufwendiger als zuvor. Es werden Tauchhülsen im Vor- und Rücklauf des Heizkörpers benötigt, welche vom Heizwasser direkt umflossen werden. Aufgrund der Tauchhülsen werden geringfügige bauliche Änderungen nötig, beispielsweise können diese 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 des Herstellers, ein Stück Rohrleitung entfernt und durch den Durchflusssensor ersetzt. Dieser kann beim Einbau entweder fest integriert oder durch eine Anschlussverschraubung, beziehungsweise einen Flanschanschluss, 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 Temperaturmessung erfolgt durch zwei PT500 Sensoren, wohingegen der Durchfluss von einem ULTRAFLOW 54 Ultraschallsensor erfasst wird. Die gemessenen, wie auch die berechneten Werte der Heizungsleistung lassen sich einzeln durch das Rechenwerk auslesen. Der MULTICAL 602 ist zum einen kostengünstig und umfasst alle benötigten Sensoren, die bereits präzise aufeinander abgestimmt sind. Zum anderen ist er wartungsfrei, da er den Durchfluss über kontaktlose Ultraschallmessungen bestimmt, indem er die Laufzeitdifferenz zwischen wechselseitig gesendeten und empfangenen Signalen zweier integrierter Ultraschallsensoren auswertet. Ein klarer Vorteil des Durchflusssensors ULTRAFLOW 54 ist, dass er bereits mit einer Tauchhülse ausgestattet ist, sodass lediglich eine weitere Tauchhülse montiert werden muss.

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

Um eine Steuerung durch den Controller zu ermöglichen wird ein Signalwandler eingesetzt, der Modbustelegramme in elektrische Signale umwandeln kann. Hierzu wird innerhalb der Anlage das EX9024-M Modul von EXPERTDAQ integriert, 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 die Möglichkeit einer einfachen Erweiterung der Anlage um drei weitere analoge Komponenten, wie beispielsweise einen Schalter zur Bedienung der Jalousien.

Der thermoelektrische Stellantrieb *ABNM-LIN* von *Danfoss* wird zur stufenlosen Be-tätigung des Heizkörperventils eingesetzt. Seine 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, womit er den Durchfluss im Heizkörper steuert.

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

3.4 Installation und Inbetriebnahme

Eine Übersicht über den Aufbau und die Installation der Anlage findet sich in Abb. 3.5. Die Installation lässt sich in die zwei Bereiche Hardware und Software untergliedern. Die Hardware Installation umfasst die Montage der einzelnen Komponenten, deren Stromversorgung und die Verkabelung im Netzwerk. Der zweite Teil beschäftigt sich mit der Inbetriebnahme der Anlage durch eine softwareseitige Ansteuerung 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 eine flexible Ansteuerung der Anlage innerhalb des gesamten lokalen Netzwerks und über eine VPN-Verbindung auch außerhalb vom Campus möglich. Der Rechner ist bereits im Netzwerk integriert, sodass lediglich das Gateway beim Rechenzentrum der Hochschule Karlsruhe registriert werden musste, damit ihm eine IP Adresse gemäß dem Ethernet/IP Protokoll zugewiesen werden konnte. Um den Webthermographen in die Anlage einzubinden, wurde dasselbe Prozedere durchlaufen.

Der Schaltplan des seriellen RS-485 Netzwerks wird in Abb. 3.6 dargestellt. Zur Verkabelung des Netzwerks wird eine Busleitung von LAPPKABEL verwendet, welche aus zwei jeweils paarweise verdrillten Leitungspaaren besteht, die durch ein Kupfergeflecht gegen Störungen abgeschirmt sind. Das Datenblatt der Busleitung ist im Anhang C.7 zu finden. Die Verkabelung erfolgt gemäß der Spezifikationen im Modbus seriell Protokoll mod [2006b], daher wird die D0 Leitung der braunen Ader, die D1 Leitung hingegen der gelben Ader und die GNC Common Leitung der weißgrauen Ader zuge-

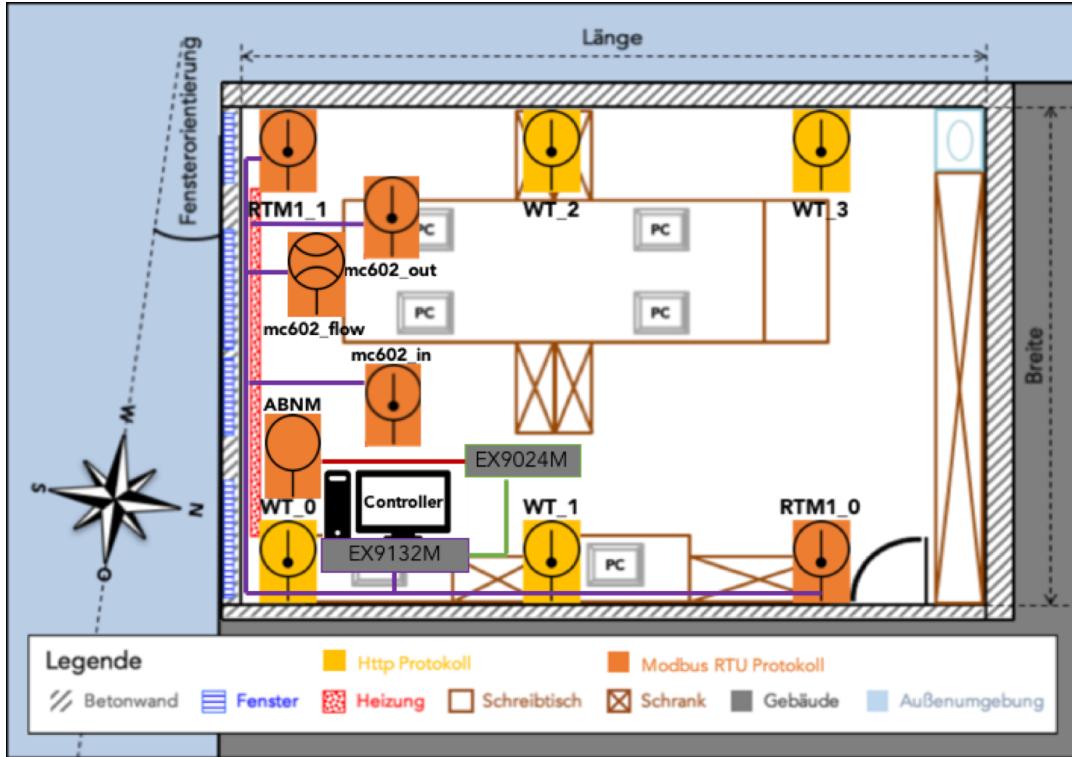


Abb. 3.5: Schaltplan der Raumtemperaturregelungsanlage

wiesen. Das grüne Kabel kann für eine gemeinsame Stromversorgung der Komponenten eingesetzt werden, was im Rahmen dieser Anlage aufgrund der vielen unterschiedlichen benötigten Spannungen nicht genutzt wurde. Folglich hat jede Komponente, wie im Schaltplan dargestellt, ein eigenes Netzteil und daher eine eigene Stromversorgung.

Die Kabel wurden gemäß der Übersicht in Abb. 3.5 verlegt und mit den Bauteilen entsprechend dem Schaltplan in Abb. 3.6 verbunden. Die kurzen Stichleitungen wurden über flexibel einsetzbare und zugleich wiederverwendbare Hebelklemmen an die lange Busleitung angeschlossen.

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

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
Stoppbits	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

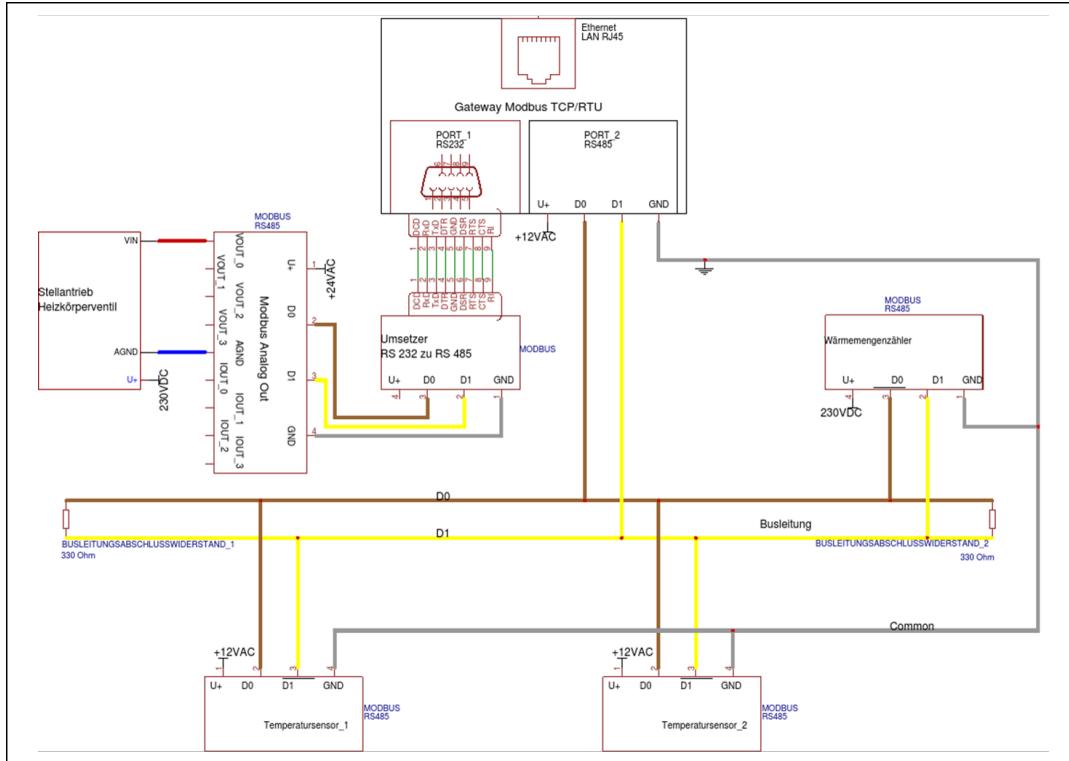


Abb. 3.6: Schaltplan der Raumtemperaturregelungsanlage

Bei der Inbetriebnahme der Anlage kam es zu Schwierigkeiten bei der Kommunikation. Es stellte sich heraus, dass der Signalwandler EX9024M nicht streng nach den Modbus Spezifikationen für serielle Kommunikation hergestellt wird. Infolgedessen kann dieser nicht mit den beiden Raumtemperaturfühlern innerhalb eines Netzwerks betrieben werden, da keine kompatible Netzwerkkonfiguration zustande kommt. Das Gateway ist mit zwei Ports ausgestattet, wovon der zweite zum Aufbau eines weiteren seriellen Netzwerks genutzt wird und dadurch das Problem beseitigt wurde. Hierfür wurde ein Bridge, in der Form eines einfachen Adapters, genutzt, der die Spannungspegel und Steckerbelegungen des RS 232 Ports auf die einer RS 485 Schnittstelle umsetzt. Hierzu findet sich eine Darstellung im Schaltplan in Abb. 3.6.

3.4.2 Software

Um eine zentrale Steuerung vom Controller aus zu gewährleisten, müssen verschiedene Schnittstellen zu Python von einzelnen Komponenten der Anlage genutzt werden. Wie bereits erwähnt, bietet Python verschiedene Pakete an, die Funktionalitäten bereitstellen. Nichtsdestotrotz muss eine eigene anlagenspezifische Software entwickelt werden, die zu einem gewissen Teil auf die von Python bereitgestellten Funktionalitäten zugreift. Der im Folgenden beschriebene Programmcode erfolgt vollständig objektorientiert und versucht eine klare Struktur und Wiederverwendbarkeit einzelner Methoden und Klassen zu gewährleisten. Das oberste Ziel der Programmierung war es, eine einfache Bedienbarkeit der Anlage, sprich eine Bedienung ohne Kenntnisse über die zugrundeliegende Kommunikationstechnik, zu ermöglichen.

Zum Auslesen der vier Temperatursensoren über den Webthermograph8x wurde das Python Paket `httplib` genutzt. Die Abfrage der Temperatursensoren über das HTTP Protokoll wurde in einer eigenen Klasse `SensorsHttp` implementiert, welche das `HTTPConnection` Objekt aus der `Bits` verwendet. Die konkrete Umsetzung der Klasse ist in Listing 3.1 dargestellt.

Um eine Verbindung zum Thermograph8x herzustellen wird dessen IP Adresse und Port Nummer benötigt, über die Telegramme gesendet und empfangen werden. Die Werte hierfür sind aus Gründen des Zugriffsschutzes nicht im Programmcode hinterlegt, sondern finden sich in den Umgebungsvariablen `server-address` und `server-port` der Pythonkonsole des Controllers wieder. Mithilfe der `init`-Methode wird bei der Initialisierung ein `HTTPConnection` Objekt `server` erstellt, dem als Host und Port die Adressen des Webthermographen übergeben werden. 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 durch eine Serveranfrage abgefragt werden, welche die Messwerte durch einen einfachen Befehls-string „GET\Single“ zusammen mit der ID des gewünschten Temperatursensors anfordert. Nach erfolgter Rückgabe des Messwerts, der in einen float Datentyp umgewandelt wurde, wird die Verbindung zum Server wieder getrennt. Die Methoden, die das Auslesen der einzelnen Messwerte veranlassen, rufen lediglich die abstrakte Methode auf und übergeben dabei die ID des gewünschten Temperatursensors.

```

1 from httplib import HTTPConnection
2
3 class SensorsHttp():
4     @property
5         def server_address( self ):
6             return self . server . host
7
8     @property
9         def server_port( self ):
10            return str( self . server . port )
11
12    def __init__(self, server_address, server_port):
13        self . server = HTTPConnection(host = server_address, port = int(server_port))
14
15    def __connect_to_server(self):
16        self . server . connect()
17
18    def __disconnect_from_server(self):
19        self . server . close()
20
21    def __get_wt(self, unit):
22        self . __connect_to_server()
23        self . server . request('GET', "/Single" + str(unit))
24        response_string = self . server . getresponse().read()
25        temperature = float(response_string . split(';') [-1][:4]. replace(",","." ))
26        return temperature
27
28    finally :
29        self . __disconnect_from_server()
30
31    def get_wt_0(self):
32        return self . __get_wt(unit = 1)

```

```

33     def get_wt_1(self):
34         return self.__get_wt(unit = 2)
35
36     def get_wt_2(self):
37         return self.__get_wt(unit = 3)
38
39     def get_wt_3(self):
40         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. Eines davon heißt *minimalmodbus*, da es jedoch keine Unterstützung für das Modbus TCP Protokoll bietet soll diesem keine weitere Beachtung zukommen. Nach einer ausführlichen Recherche über die beiden verbleibenden Pakete, fiel die Entscheidung auf das sogenannte *pymodbus* Paket. Es konnte durch seinen größeren Funktionsumfang und seine ausführlichere Dokumentation, die mit einfachen Beispielen veranschaulicht ist, überzeugen. Um eine Kommunikation über Modbus zu ermöglichen, wurde mithilfe des *pymodbus* Pakets die Klasse *ModbusConnection* programmiert, die in 3.2 zu sehen ist.

Für eine Kommunikation über das Modbus TCP Protokoll, bedarf es zunächst einer Verbindung zwischen Client und Server. Hierzu wird das *ModbusClient* Objekt genutzt, da der Controller als Client über das Modbus TCP Protokoll Anfragen an die Server stellt. Faktisch stellt der Controller ausschließlich Anfragen an das Gateway, welches die Telegramme in das serielle Modbus RTU Protokoll übersetzt, um mit den Slaves im seriellen Netzwerken zu kommunizieren. Bei der Initialisierung werden erneut die IP Adresse *client-adress* und der Port *client-port* benötigt, über die die Kommunikation stattfindet. Auch hier wurden eigene Methoden zum Aufbauen und Trennen einer Verbindung implementiert, welche bei der Initialisierung genutzt werden. Eine Besonderheit stellt die *delete*-Methode dar, die im Falle eines Fehlers oder beim Sterben einer Instanz der Klasse die Verbindung trennt. Außer den oben genannten Methoden, sind weitere für die benötigten Zugriffe auf die verwendeten Datentabellen implementiert. Die Methoden ermöglichen das Auslesen von input- und holding-registers und liefern ein register zurück, welches in Abhängigkeit des jeweiligen betrachteten Datenmodells einen bestimmten Datentyp besitzt. Für die beiden Lesemethoden wird eine Anfrage maximal zehn Mal wiederholt, wobei nach jedem Fehlversuch eine Pause von 1,5 Sekunden vorgesehen ist. Für die Abfrage wird die Datenadresse der jeweiligen Tabelle, die Anzahl der auszulesenden Register sowie die Slave-ID benötigt. Die letzte Methode zum Schreiben in Register wird auf drei Fehlversuche begrenzt, die ebenfalls durch eine Pause von 1,5 Sekunden voneinander getrennt sind. Auch hier wird die Datenadresse und die Slave-ID benötigt, um den mitgegebenen Wert in die Tabelle zu schreiben. Damit sind die grundlegenden Funktionen für eine Modbus Kommunikation implementiert.

```

1from pymodbus.client.sync import ModbusTcpClient as ModbusClient
2from time import sleep

```

```
3 class ModbusConnection(object):
5     _max_retries_read = 10
7     _max_retries_write = 3
9
11    @property
13        def max_retries_read(self):
14            return self._max_retries_read
16
18    @property
20        def max_retries_write(self):
21            return self._max_retries_write
23
25    @property
27        def client_address(self):
28            return self.client.host
30
32    @property
34        def client_port(self):
35            return str(self.client.port)
37
39    def __init__(self, client_address, client_port):
40        self.client = ModbusClient(host=client_address, port=int(client_port))
41        self.connect_to_client()
43
45    def __del__(self):
46        self.disconnect_from_client()
48
50    def connect_to_client(self):
51        self.client.connect()
53
55    def disconnect_from_client(self):
56        self.client.close()
58
59    def read_input_registers(self, address, count, unit):
60        k = 0
61        while k < self.max_retries_read:
62            try:
63                return self.client.read_input_registers(address=address, count=count, unit=unit).registers
64            except:
65                k += 1
66                sleep(1.5)
68
69    def read_holding_registers(self, address, count, unit):
70        k = 0
71        while k < self.max_retries_read:
72            try:
73                return self.client.read_holding_registers(address=address, count=count, unit=unit).registers
74            except:
75                k += 1
76                sleep(1.5)
78
79    def write_register(self, address, unit, value):
80        k = 0
81        while k < self.max_retries_write:
82            try:
83                return self.client.write_register(address=address, unit=unit, value=value)
84            except:
85                k += 1
86                sleep(1.5)
```

Listing 3.2: Klasse zum Verbindungsaufbau und für die Grundfunktionen über Modbus TCP

Die Grundfunktionen werden von der *SensorsModbus* Klasse genutzt, um diverse Messwerte von den verschiedenen Sensoren abzufragen. Der zugehörige Code ist in Listing 3.3 abgebildet. Beim Auslesen der Werte aus dem Wärmemengenzähler findet sich eine Besonderheit, da dessen Messwerte die Größe eines einzelnen Registers übersteigen. Wie in Abschnitt 2.2.3 erwähnt, werden die Daten gemäß der Big-Endian Codierung verschlüsselt und müssen daher decodiert werden. Dazu bietet *pymodbus* mit dem BinaryPayload Decodierer eine passende Funktion an. Hierbei muss berücksichtigt werden, dass die Decodierung von Little- und Big-Endian in *pymodbus* vertauscht sind. Zum Auslesen der Messwerte im Wärmemengenzähler werden der Count, der die Anzahl der zu lesenden Register angibt, und die ID des Bauteils im seriellen Netzwerk angegeben. Die konkreten Messwerte unterscheiden sich durch ihre Adressen in den Datentabellen und sind in den entsprechenden Methoden hinterlegt. Eine Übersicht der Tabellen-Adressen der einzelnen Bauteile findet sich in den Datenblättern der Bauteile im Anhang D. Die beiden Temperatursensoren geben die Temperatur in Form eines ganzzahligen Werts zurück, der um eine Zehnerpotenz erhöht ist. Daher müssen die abgefragten Werte durch eine Division durch zehn decodiert werden. Die Messwerte der zum Zeitpunkt der Abfrage herrschende Raumtemperatur, stehen unter der gleichen Tabellenadresse, müssen jedoch anhand ihrer jeweiligen Sensor ID aus den zugehörigen Tabellen ausgelesen werden. Die IDs wurden den Temperatursensoren durch die Schalterstellung gemäß dem Datenblatt in C.3 zugewiesen.

```

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

6 class SensorsModbus(ModbusConnection):
7     def __get_mc602(self, address):
8         registers = self.read_input_registers(address = address, count = 2, unit = 65)
9         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)

```

```
30  def get_rtm1_1(self):
    return self.__get_rtm1(unit = 4)
```

Listing 3.3: Klasse zum Auslesen über Modbus TCP

Die Klasse *ActuatorsModbus* 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 D.1 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.

```
1from pymodbus.client.sync import ModbusTcpClient as ModbusClient
2from modbus_connection import ModbusConnection
3
4class 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)
24        voltage_value = (float(registers[0]) - 8191)/819.2
25        return voltage_value
26
27    def set_ex9024(self, opening_level):
28        opening_level = float(opening_level)
29        if opening_level < 0 or opening_level > 1:
30            raise ValueError( " Only values between 0 and 1 allowed for opening level
31                           definition . " )
32        voltage_value = opening_level * 10
33        self.__set_ex9024(address = 2, voltage_value = voltage_value)
34
35    def get_ex9024(self):
36        voltage_value = self.__get_ex9024(address = 2)
37        opening_level = voltage_value/10.0
38        return opening_level
```

Listing 3.4: Klasse zum Auslesen über Modbus TCP

Es wurden zwei weitere Klassen implementiert, die die vorausgegangenen *Sensors* und

Aktors Klassen ein letztes Mal erweitern. Sie dienen dazu eine Bedienoberfläche zu schaffen, die sowohl unabhängig von ihrer Plattform als auch von ihrer Kommunikationstechnologie ist. Die Klassen *Sensors* und *Actuators* bewerkstelligen dies, indem sie die zuvor definierten Klassen importieren und bei der Initialisierung jeweils eine Instanz der Klasse bilden. Um eine Unabhängigkeit von der 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 dahinter steckt. Damit ist ein Interface geschaffen, das für Steuerungen genutzt werden kann.

```

from actuators_modbus import ActuatorsModbus
2from httplib import HTTPConnection
from sensors_modbus import SensorsModbus
4from sensors_http import SensorsHttp

6class Actuators():
8    def __init__(self, client_address, client_port):
        self.actuators_modbus = ActuatorsModbus(client_address, client_port)
10   def set_valve_position( self ,opening_level):
        return self.actuators_modbus.set_ex9024(opening_level)

14   def get_valve_position( self ):
        return self.actuators_modbus.get_ex9024()
16
18 class Sensors():
19    def __init__(self, client_address, client_port, server_address, server_port, port,
20                 baudrate, parity, stopbits, bytesize, timeout):
21        self.sensors_modbus = SensorsModbus(client_address, client_port)
22        self.sensors_http = SensorsHttp(server_address, server_port)

24    def get_temperature_northeast(self):
        return self.sensors_modbus.get_rtm1_0()

26    def get_temperature_northwest(self):
        return self.sensors_http.get_wt_3()

28    def get_temperature_west(self):
        return self.sensors_http.get_wt_2()

30    def get_temperature_southwest(self):
        return self.sensors_modbus.get_rtm1_1()

32    def get_temperature_southeast(self):
        return self.sensors_http.get_wt_0()

34    def get_temperature_east(self):
        return self.sensors_http.get_wt_1()

36    def get_radiator_flowrate( self ):
        return self.sensors_modbus.get_mc602_flowrate()

38    def get_radiator_temperature_inlet( self ):
        return self.sensors_modbus.get_mc602_temperature_inlet()

40    def get_radiator_temperature_outlet( self ):
        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 einem Zweipunktregler

Ein erstes, vergleichsweise einfach gehaltenes Programm zur Regelung soll die Funktion der Anlage überprüfen und dabei erste Messdaten sammeln. Anhand des Programmablaufs soll im Folgenden die Ansteuerung der Anlage und das Zusammenspiel der Komponenten veranschaulicht werden.

Eine Darstellung des Programms findet sich in Listing 3.6. Es zeigt die Implementierung eines simplen Zweipunktreglers, der nach dem bekannten Backofenprinzip arbeitet. Der Zweipunktregler besitzt eine Schalthysterese von einem Grad Celsius. Zunächst werden die beiden Bedienoberflächen *Sensors* und *Actuators*, gemeinsam mit einem Paket zum Einfügen von Messwerten in eine Datenbank, importiert. Anschließend werden die verschiedenen Verbindungsparameter aus den Umgebungsvariablen der Pythonkonsole ausgelesen und zum Aufbau einer Verbindung mit den Anlagenteilen und der Datenbank genutzt. Dies geschieht durch die Instanziierung der Bedienoberflächenklassen *Sensors* und *Actuators*, die den Variablen *s* beziehungsweise *a* zugewiesen werden. Es folgt die Deklaration der beiden Ventilstellungen für die Schaltpunkte.

Der eigentliche Steueralgorithmus beginnt mit einer while-Schleife. Eine dauerhafte Laufzeit des Algorithmus wird über die Schleifenbedingung *true* erreicht, welche nur durch eine Tastenkombination oder durch das Auftreten eines Fehlers gestoppt werden kann. Tritt eines der beiden Abbruchkriterien ein, werden die Verbindungen zu den Anlagenteilen getrennt. Dieses Vorgehen soll eine schnelle Wiederaufnahme des Betriebs gewährleisten, da die Komponenten auf diese Weise nicht durch eine bestehende Verbindung blockiert werden. Der Algorithmus beginnt damit alle Messwerte innerhalb der Anlage ausgelesen und daraus das arithmetische Mittel der Raumtemperaturen bestimmen. Die angenommene Raumtemperatur lässt sich ebenfalls über den Gewichtungsvektor *weightings* 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 zur Bedienung der Anlage einfach und robust aufgebaut sind. Außerdem wurden an gegebenen Stellen darauf geachtet, mögliche Fehlbedienungen von vornherein auszuschließen. Ein Beispiel hierfür ist, dass nur real mögliche Ventilstellungen gesetzt und keine unmöglichen Werte angefordert werden können. Anhand des berechneten Mittelwerts wird entschieden, ob der Heizkörper zur Raumerwärmung eingeschaltet, weiterhin genutzt oder abgeschaltet werden soll. Liegt die Raumtemperatur unterhalb von 21 Grad Celsius und der Heizkörper ist ausgeschaltet, so erfolgt eine Ausgabe in der Konsole. Diese besagt, dass es zu kalt ist, woraufhin das Öffnen des Heizungsventils veranlasst wird. Sobald die Temperatur über 22 Grad Celsius steigt und der Heizkörper zeitgleich eingeschaltet ist, erfolgt eine erneute Ausgabe. In dieser heißt es, dass es zu warm ist, wodurch das Heizungsventil

umgehend geschlossen wird. Bevor die Messdaten in der Konsole ausgegeben und in eine Datenbank eingefügt werden, erfolgt ein Update der Ventilstellung. Ehe der Algorithmus wieder von vorne beginnt, ist in der Steuerung eine Unterbrechung von 60 Sekunden vorgesehen.

```

from sensors_actuators_k004b.sensors import Sensors
2from sensors_actuators_k004b.actuators import Actuators
3from mdb_communication_k004b.dbinsert import DBInsert
4import time
5import os
6import numpy as np
# Modbus Connection Parameter
7sensor_client_address = os.environ["EX9132_ADDRESS"]
8sensor_client_port = os.environ["EX9132_PORT_1"]
9actuator_client_address = os.environ["EX9132_ADDRESS"]
10actuator_client_port = os.environ["EX9132_PORT_2"]
11# http Connection Parameter
12server_address = os.environ["WT_ADDRESS"]
13server_port = os.environ["WT_PORT"]
# DB Connection parameter
14db_name = os.environ["DB_NAME"]
15db_host = os.environ["DB_HOST"]
16db_port = os.environ["DB_PORT"]
17db_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()
33        rtm1_0 = s.get_temperature_northeast()
34        wt_3 = s.get_temperature_northwest()
35        wt_2 = s.get_temperature_west()
36        rtm1_1 = s.get_temperature_southwest()
37        wt_0 = s.get_temperature_southeast()
38        wt_1 = s.get_temperature_east()
39        ex9024_level = a.get_valve_position()
40        flowrate = s.get_flowrate()
41        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

50        temperature_room = temperatures_weighted / sum(weightings)

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)

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)

```

```

58     print "too hot!"
      a.set_valve_position(opening_level = valve_closed)

60     ex9024_level_new = a.get_valve_position()

62     data_set = [time.ctime(), flowrate, temperature_inlet, temperature_outlet,
                  ex9024_level_new, rtm1_0, rtm1_1, wt_0, wt_1, wt_2, wt_3, round(
                      temperature_room, 2)]
     print data_set

64     dbi.insert_temperature_1(time_measurement, rtm1_0)
66     dbi.insert_temperature_2(time_measurement, wt_3)
68     dbi.insert_temperature_3(time_measurement, wt_2)
70     dbi.insert_temperature_4(time_measurement, rtm1_1)
72     dbi.insert_temperature_5(time_measurement, wt_0)
74     dbi.insert_temperature_6(time_measurement, wt_1)
    dbi.insert_flowrate(time_measurement, flowrate)
    dbi.insert_temperature_inlet(time_measurement, temperature_inlet)
    dbi.insert_temperature_outlet(time_measurement, temperature_outlet)
    dbi.insert_valve_position(time_measurement, ex9024_level)
    dbi.insert_control_method(time_measurement, 1)

76     time.sleep(60.0)

78 except KeyboardInterrupt:
80     del a
81     del s
82     raise

```

Listing 3.6: Zweipunktregler Programm zur Inbetriebnahme der Anlage

Aus Gründen der Übersichtlichkeit sind die Abhängigkeiten der Klassen untereinander in einem UML Klassendiagramm in Abbildung Abb. 3.7 graphisch dargestellt.

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 werden konnte. Seit Mitte Januar läuft die Anlage durchgängig stabil und fehlerfrei und leistet einen sehr zuverlässigen Dienst. Die Forderung einer hohen Funktionalität der Anlage konnte demnach erfüllt werden. Durch die Programmierung einer fehlertoleranten Software konnte ebenfalls dem Wunsch einer hohen Robustheit der Anlage nachgekommen werden.

Der Raumtemperaturverlauf eines 14-tägigen Intervalls im Februar 2016 ist in Abb. 3.8 dargestellt. Er wurde bei laufendem Betrieb der Anlage gemessen, wobei der untere Schaltpunkt bei 21 und der obere bei 23 Grad Celsius lag. Es ist deutlich zu erkennen, dass die Anlage die Raumtemperatur auf einen Wert zwischen von 20 und 24 Grad Celsius regeln konnte. Zusammen mit dem Fakt, dass die Anlage seit Mitte Januar fehlerfrei läuft, lässt sich die Hypothese bestätigen, dass die Anlage eine Raumtemperatur regeln kann.

Aufgrund der zentralen Steuerung der Anlage, welche in Python umgesetzt wurde und fehlerfrei funktioniert, sind weitere 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

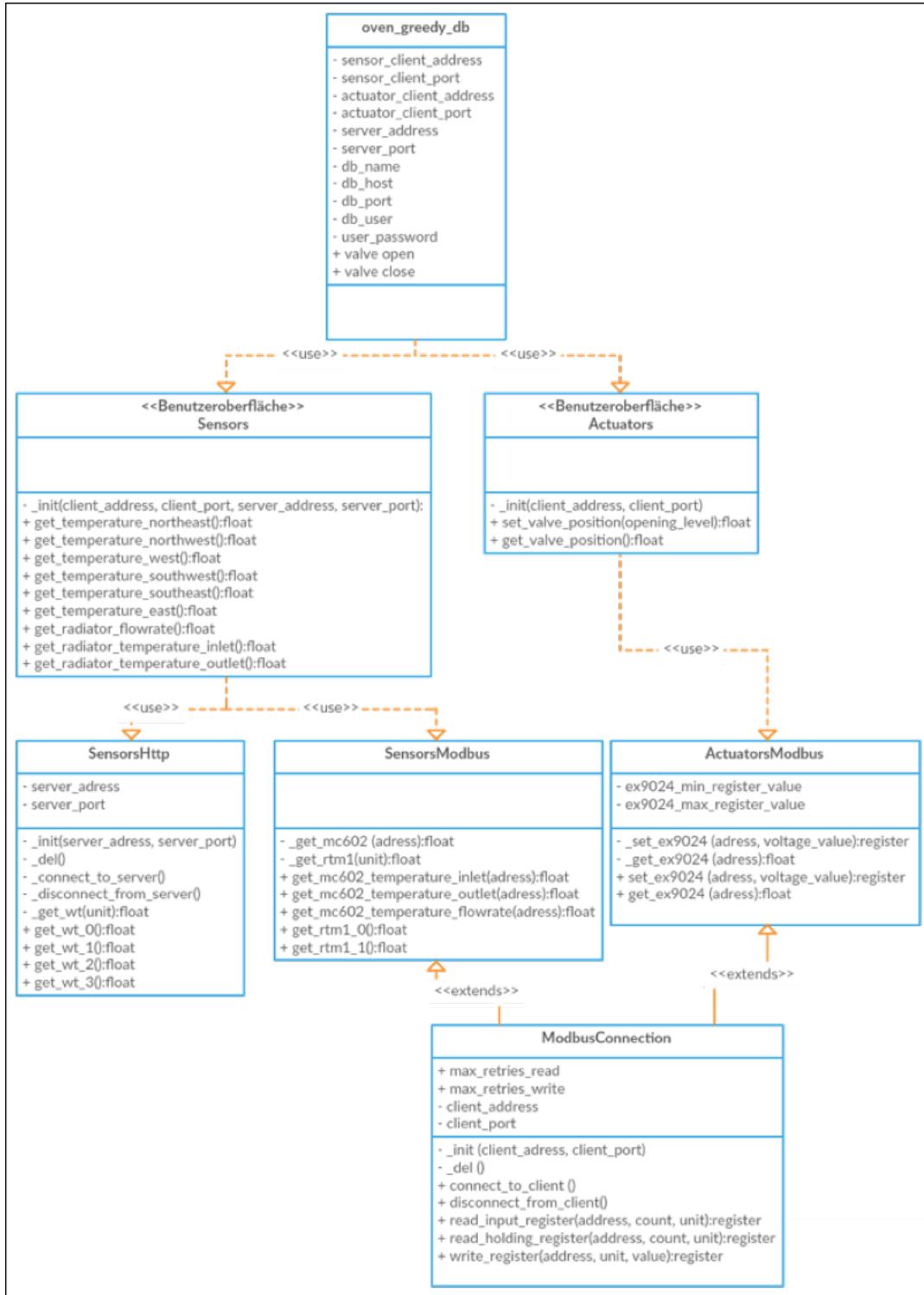


Abb. 3.7: UML Klassendiagramm der zentralen Anlagensteuerung

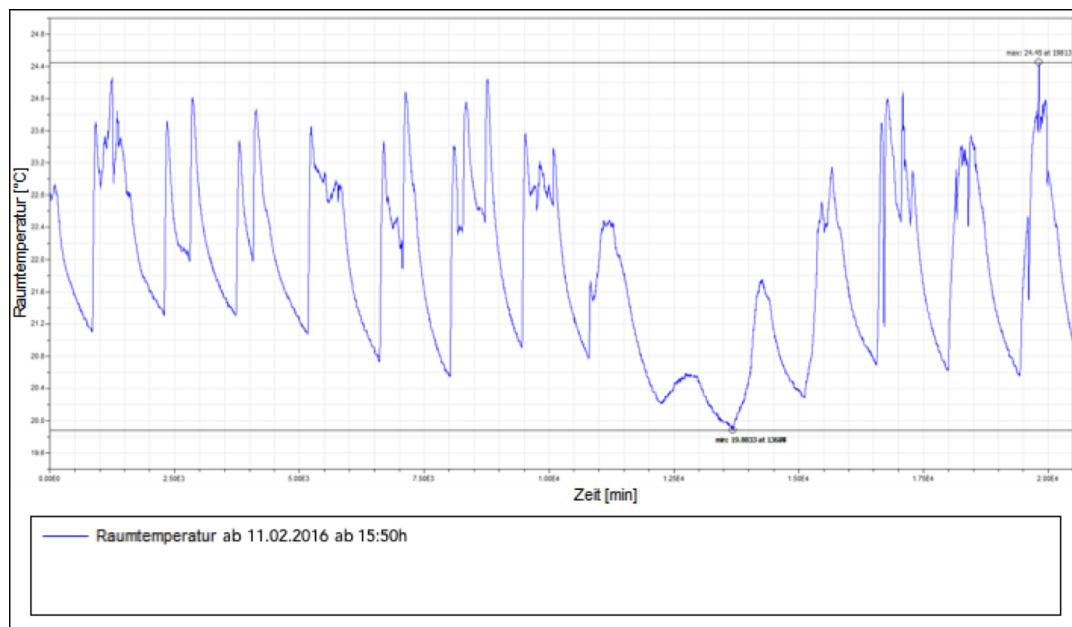


Abb. 3.8: Raumtemperaturverlauf für ein 14-tägiges Intervall im Februar 2016

Kapitel gebildet wird.

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

— NORBERT WIENER, US-amerikanischer Mathematiker

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 erfordert. 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

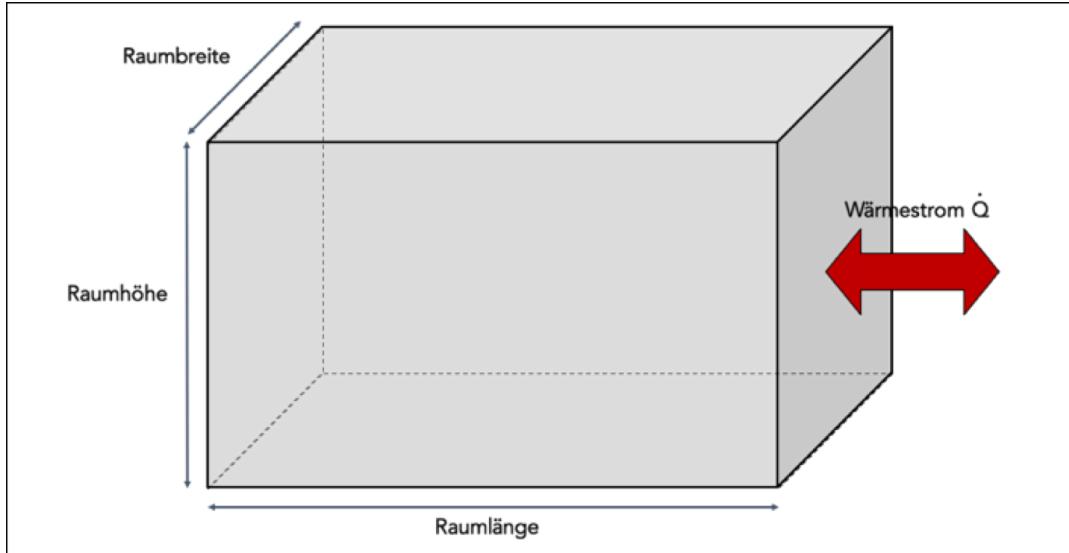


Abb. 4.1: Grundmodell eines Raumes

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 modellrelevante 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 K004b 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 Gebäude. 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 */

```

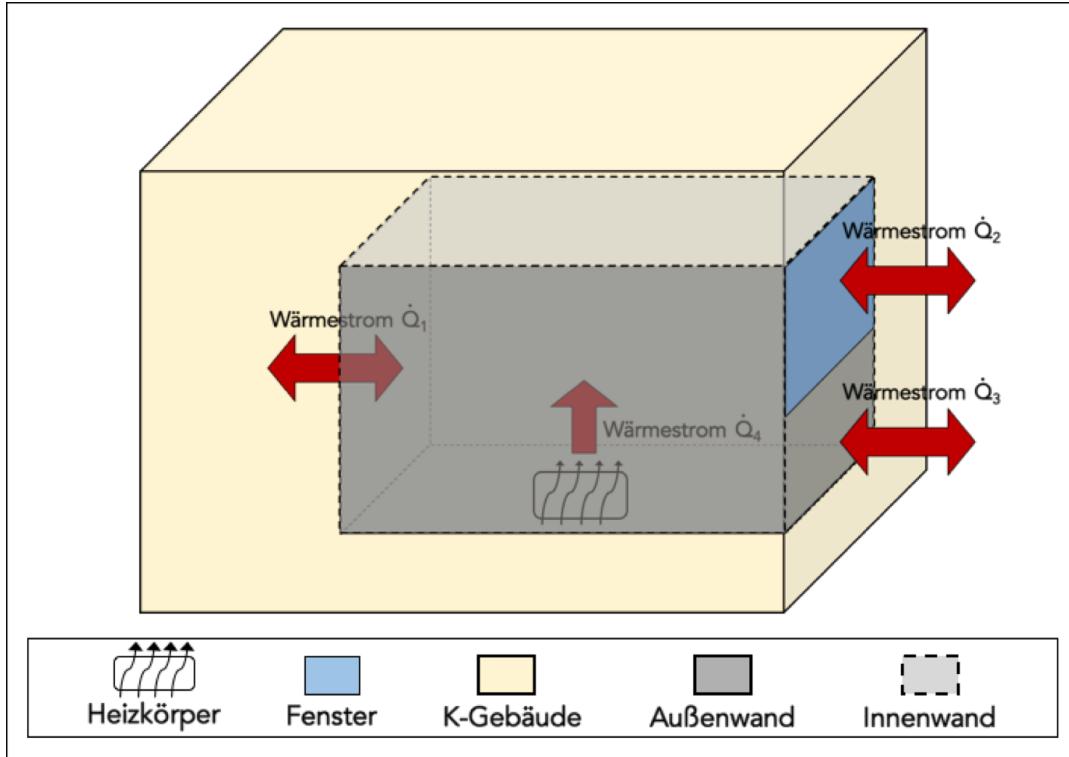


Abb. 4.2: Erweitertes Raummodell

```

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 ¹⁾	[m]
Höhe der Glieder	0,4 ¹⁾	[m]
Länge des Glieds	0,045 2 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*/

```

```

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

14 model Radiator "model for a discretized radiator within a room"
[...]
16 equation
    /* calculate exchange surface of one control volume */
18 exchange_surface_cv = (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;
    /* calculate mass within one control volume */
20 cv_m = (radiator_elment_mass * radiator_element_number)/cv_number;
    /* commit temperature of radiator fluid inlet to the first control volume */
22 cv_radiator [1].inlet.t = radiator_inlet ;
    /* connect the control volumes within the radiator */
24 for i in 1 : (cv_number-1) loop
    connect( cv_radiator[i].outlet, cv_radiator[i+1].inlet );
end for;
    /* calculate and commit the heatflow which is leaving the radiator */
28 radiator_qdot = sum(cv_radiator.cv_qdot);
end Radiator;

```

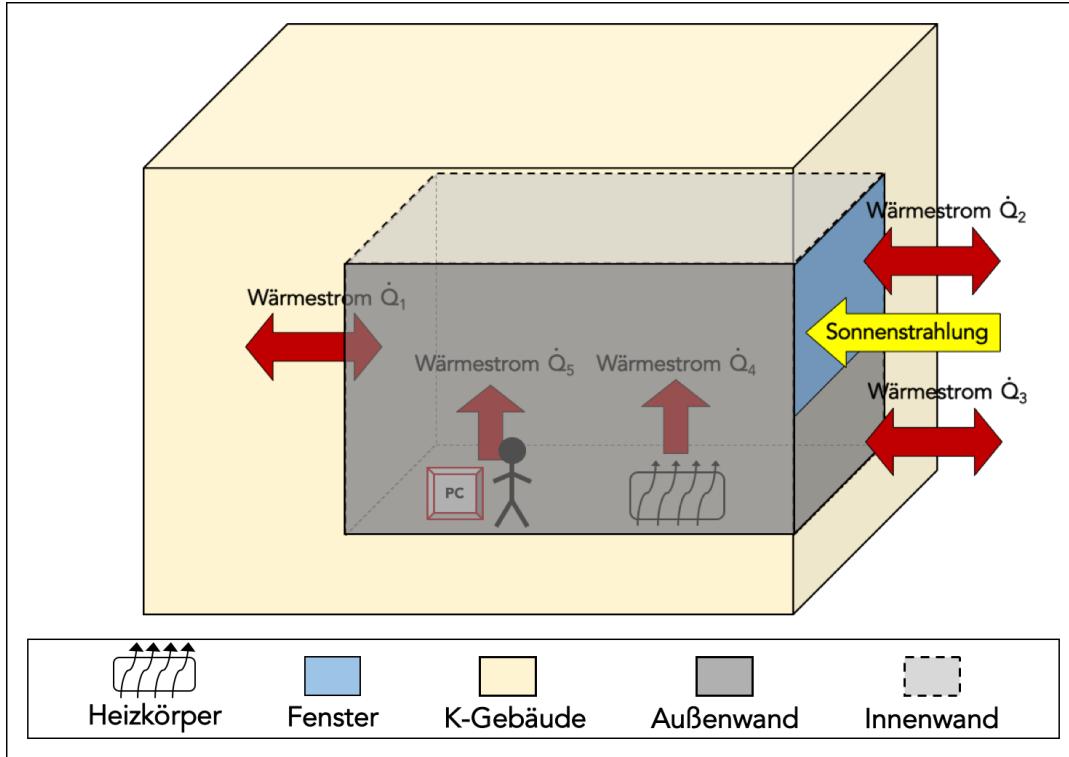
Listing 4.4: Auszüge des Modelica Modells vom Heizkörper in K004b

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  [...]
  /* 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 ?? 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 werden 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. 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.

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

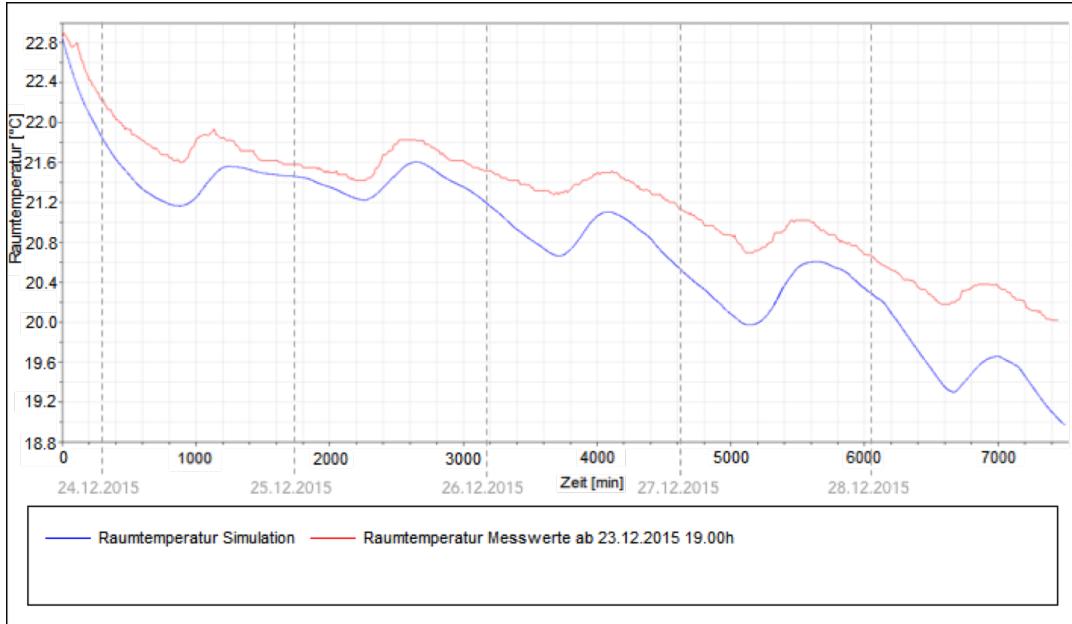


Abb. 4.4: Simulation des Raummodells ohne Einsatz der Heizung

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

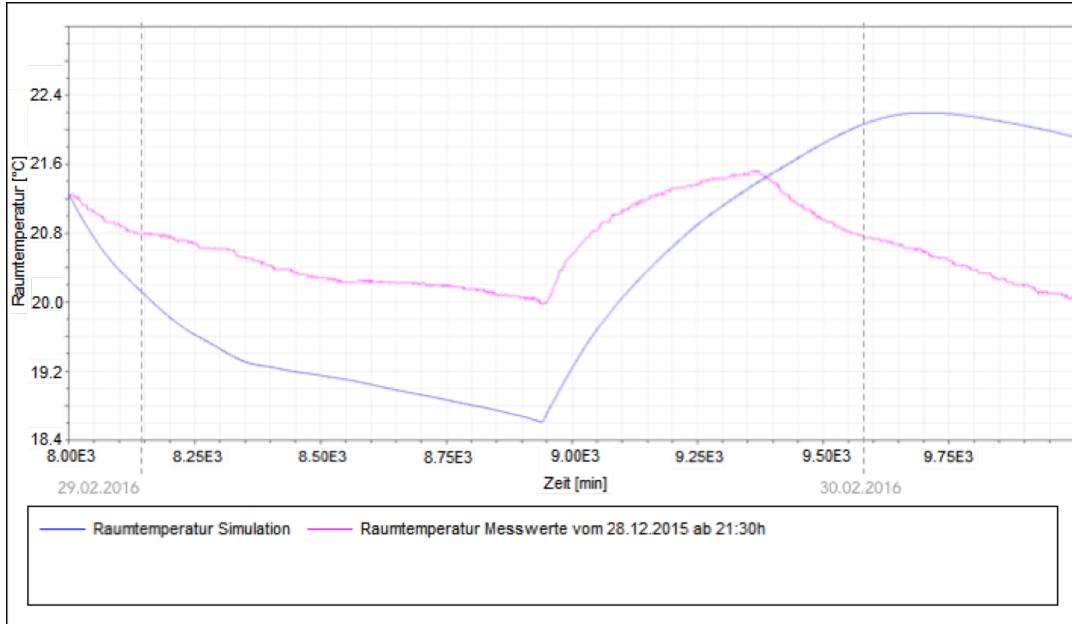


Abb. 4.5: Simulation des Raummodells mit Einsatz des Heizkörpers

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.

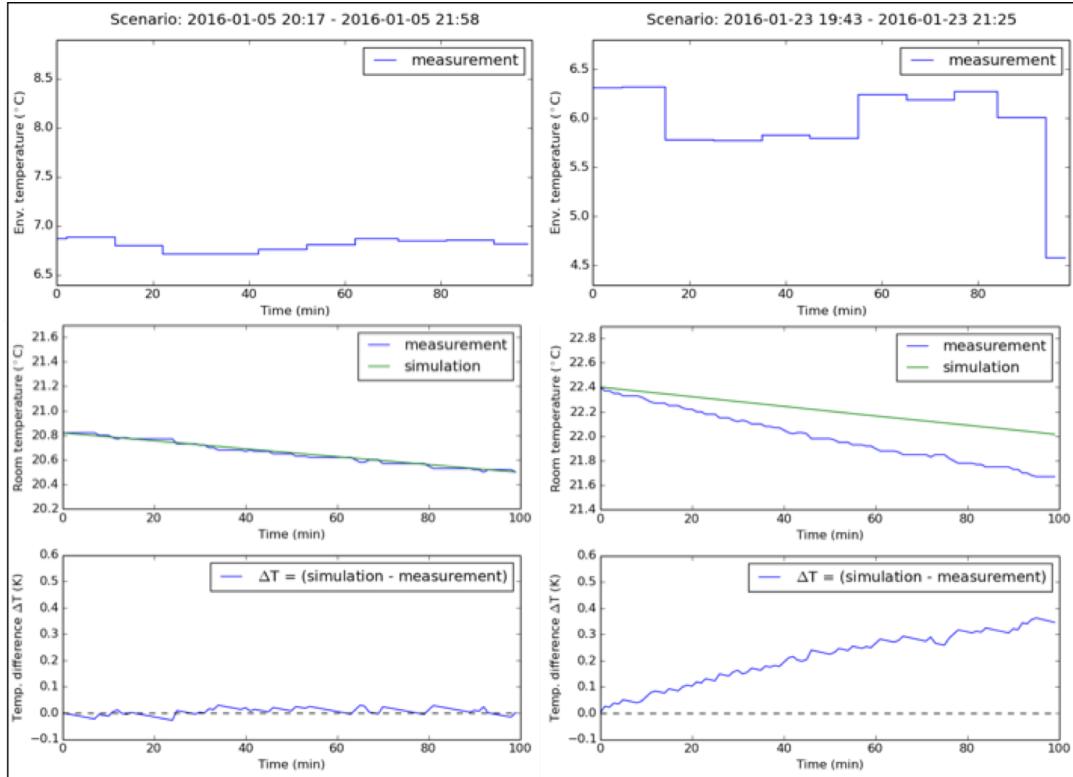


Abb. 4.6: Parameterschätzung des Wärmedurchgangskoeffizienten der Wand von zwei Intervallen

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_pestep1night.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

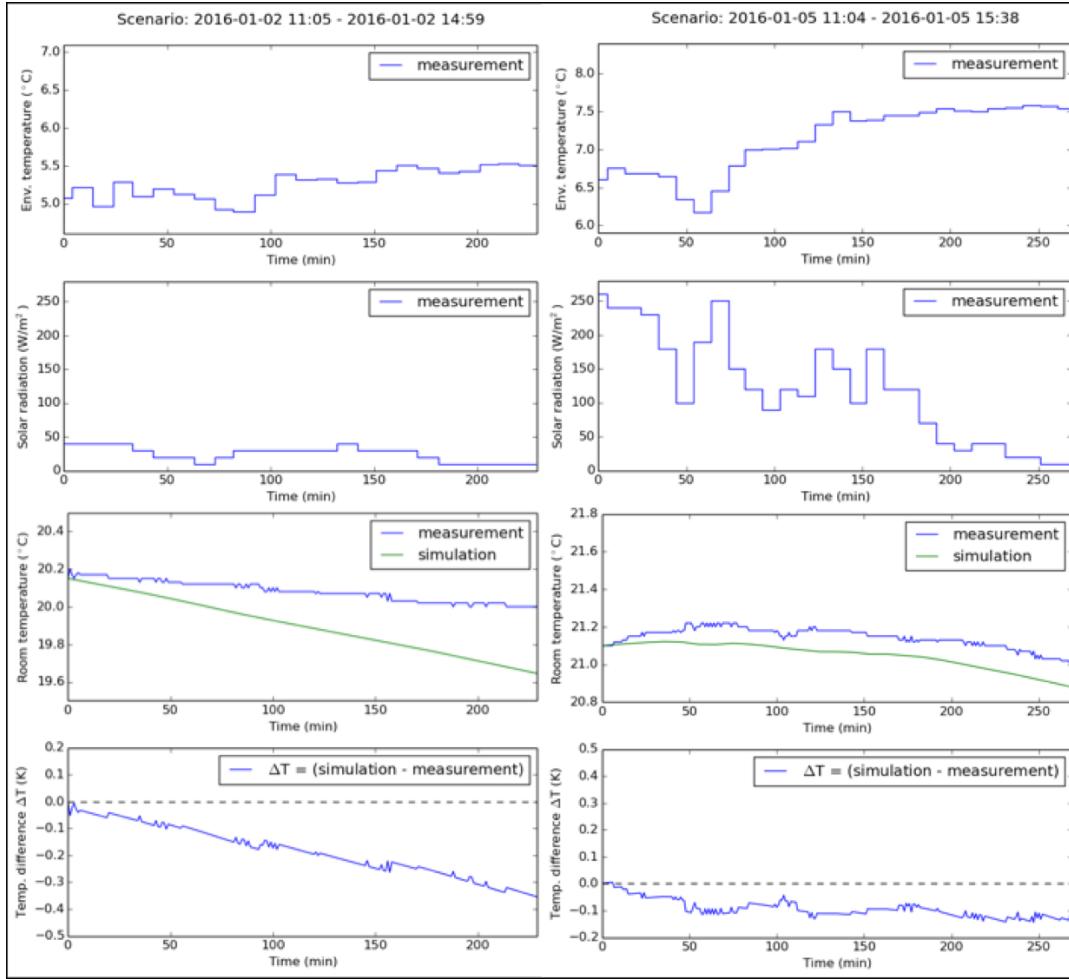


Abb. 4.7: Parameterschätzung

Simulation und Messwerten. Im linken Intervall beschreibt die Simulation bis auf minimale Abweichungen das reale System sehr genau, 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 sind in Abb. 4.7 zusammengefasst. Das Skript zur Parameterschätzung findet sich im Anhang A in der Datei *room_pes_tep2day.py*.

Die Plots auf der linken Seite stellen das Ergebnis für ein 220-minütiges Intervall am 02.01.2016, auf der rechten Seite für ein 270 minütiges Intervall 05.01.2016 dar. Die Parameterschätzung ergab einen geschätzten Transmissionsgrad von $window_{transmission} = 0.00687213$. Dieser Wert scheint zunächst unrealistisch klein. Bei einer genaueren Betrachtung ist zu beachten, dass wie bereits eingangs erwähnt bei der Parameter-

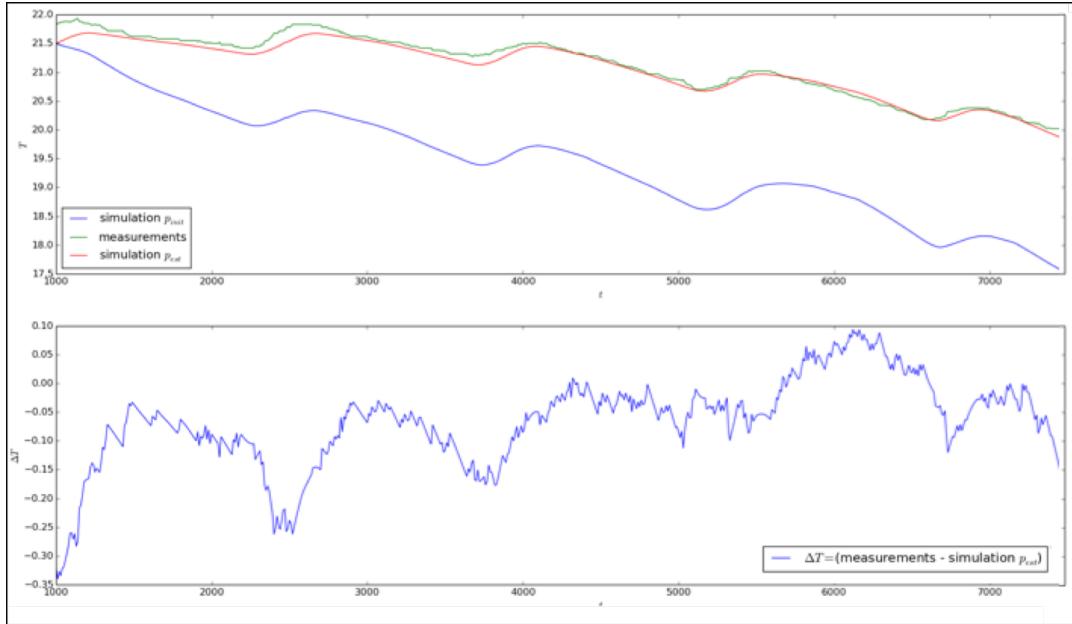


Abb. 4.8: Simulation des Raummodells ohne Einsatz der Heizung mit geschätztem Parameter

schätzung eine Anpassung des Modells an die Messwerte stattfindet, inklusive deren Störgrößen und Messfehler. Weiterhin ist der tatsächliche Transmissionsgrad der Fensterscheiben in K004b unbekannt, da der Fensterhersteller SCHÜCO lediglich die Rahmen produziert. Das Unternehmen, welches die Glasscheiben eingesetzt und die Fenster montiert hat und damit die Information besitzt, konnte leider nicht ermittelt werden.

In den oberen Plots sind wiederum die Verläufe der Steuergrößen im Schätzintervall zu sehen. Die Außentemperaturen waren in beiden Intervallen mild zwischen fünf und acht Grad Celsius. Im ersten Intervall wurde eine schwache, im zweiten hingegen wurde eine starke Globalstrahlung gemessen. Auf beiden Ergebnisplots ist zu erkennen, dass der Einfluss der Strahlung zum Ende des Intervalls hin immer weiter unterschätzt wird. Die Temperaturabweichung liegt bei beiden Intervallen unterhalb einem Betrag von $0,4^{\circ}\text{C}$. Im rechten Modellabgleich ist durch die leicht angedeuteten Senken sehr schön zu erkennen, dass das Verhalten des Modells mit der Realität trotz Abweichungen sehr gut abgebildet wird. Unter der Berücksichtigung der Messfehler und da insbesondere die ersten Minuten der Simulation eine ausreichende Beschreibung der Realität liefern, sollte die Güte des Modells für eine Anwendung mit Modellprädiktiver Regelung ausreichen.

Um die Verbesserung der Modellgüte einordnen zu können, erfolgt eine erneute Simulation des ersten Validierungsintervalls ohne den Einsatz eines Heizkörper. Die Ergebnisse sind im Plot in Abb. 4.8 dargestellt.

Abgesehen von einem Fehler bei der Initialisierung der Raumtemperatur zu Beginn der Simulation, ist eine erheblich Verbesserung zu erkennen. Über ein fünf-tägiges Intervall ist die Abweichung der Simulation sehr gering und die Dynamik des Systems wird sehr gut beschrieben.

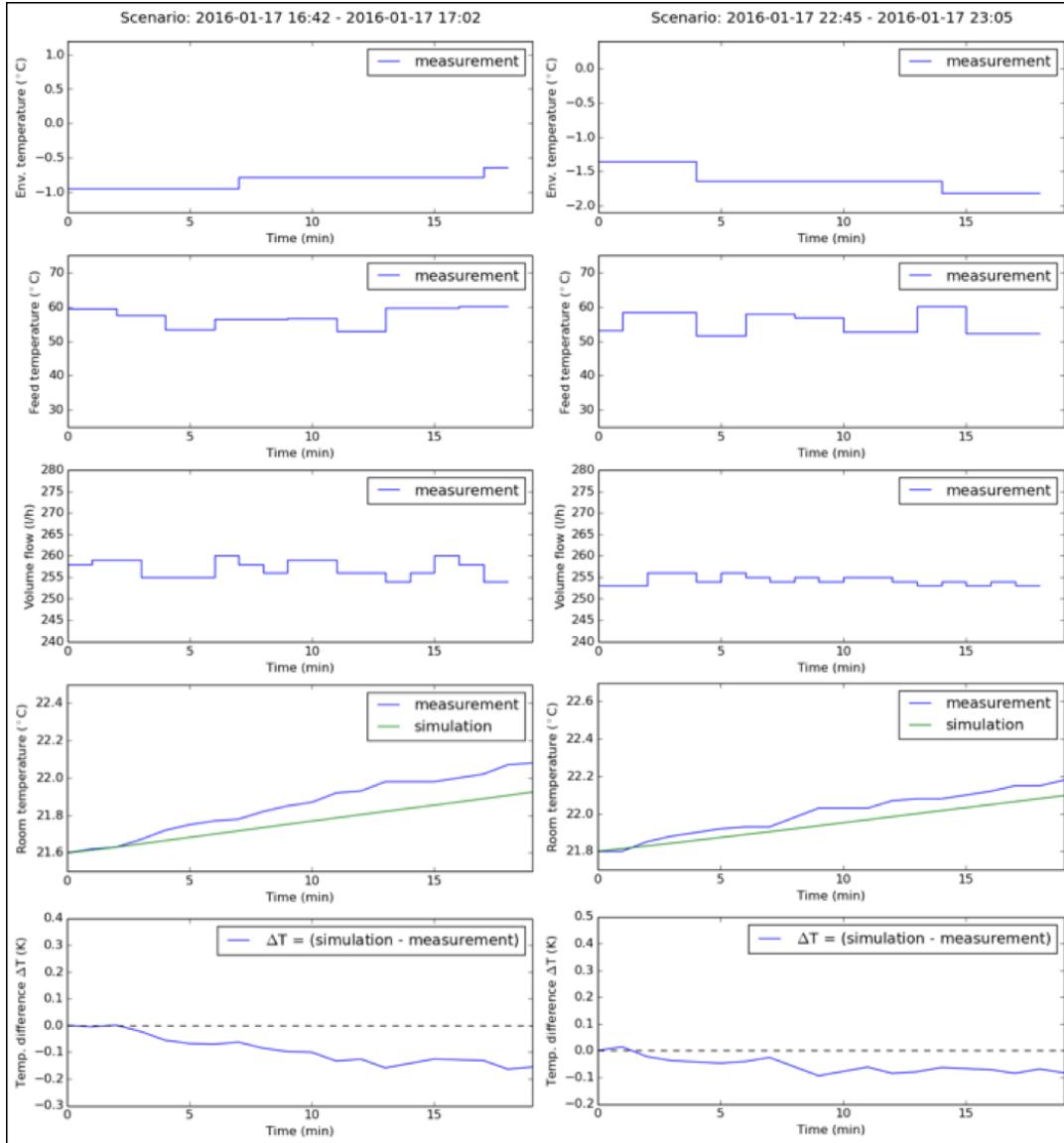


Abb. 4.9: Parameterschätzung

Im letzten Schritt wird die Schätzung des Wärmedurchgangskoeffizienten des Heizkörpers durchgeführt. Es wurden Schätzintervalle identifiziert, an denen keine Solarstrahlung zu finden war und Störgrößen möglichst ausgeschlossen werden können. Dazu bieten sich zwei Intervalle am Abend des 17.01.2016, an dem die Heizung von der Anlage genutzt wurde. Die Ergebnisse der Parameterschätzung für die 20 Minuten umfassenden Intervalle sind in Abb. 4.9 zusammengefasst. Das Skript zur Parameterschätzung findet sich im Anhang A in der Datei *room_{pe}step3_radiator_night.py*.

Die Plots stellen das Ergebnis für die beiden 20-minütigen Intervalle dar. Die Parameterschätzung ergab einen realistischen Schätzwert für den Wärmedurchgangskoeffizienten des Heizkörpers von $u_{=radiator} = 12.9872$. Wie zuvor sind in den oberen Plots die Verläufe der Steuergrößen im Schätzintervall zu sehen. Die Außentemperaturen waren in beiden Intervallen knapp unterhalb des Gefrierpunkts und in keinem der beiden Intervalle wurde eine Globalstrahlung gemessen. Außerdem wurde die Heizung mit

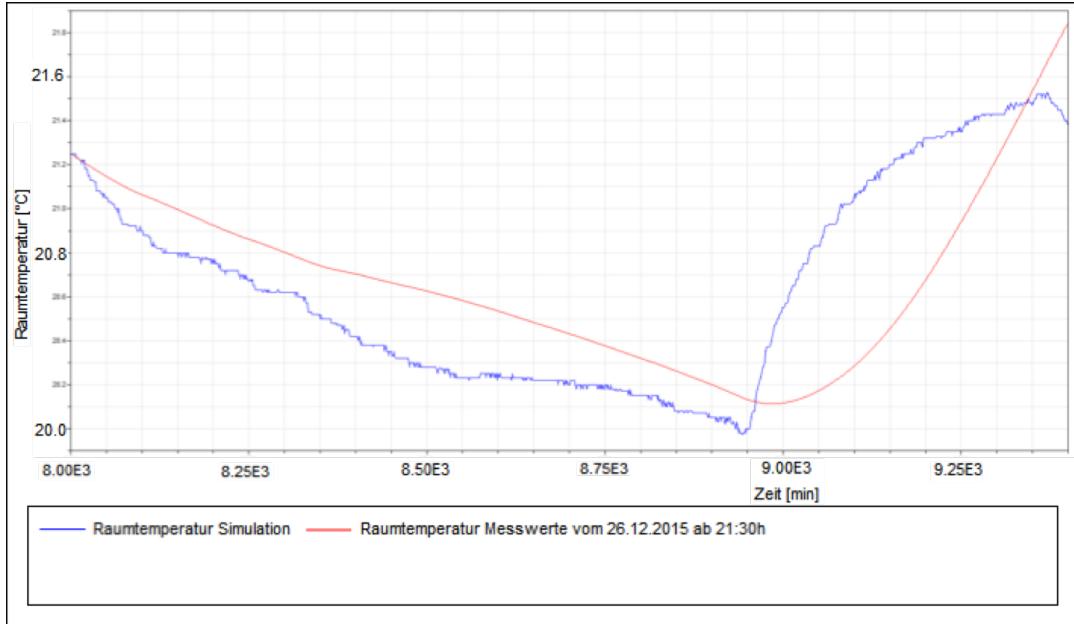


Abb. 4.10: Simulation des Raummodells mit Einsatz des Heizkörpers mit geschätztem Parameter

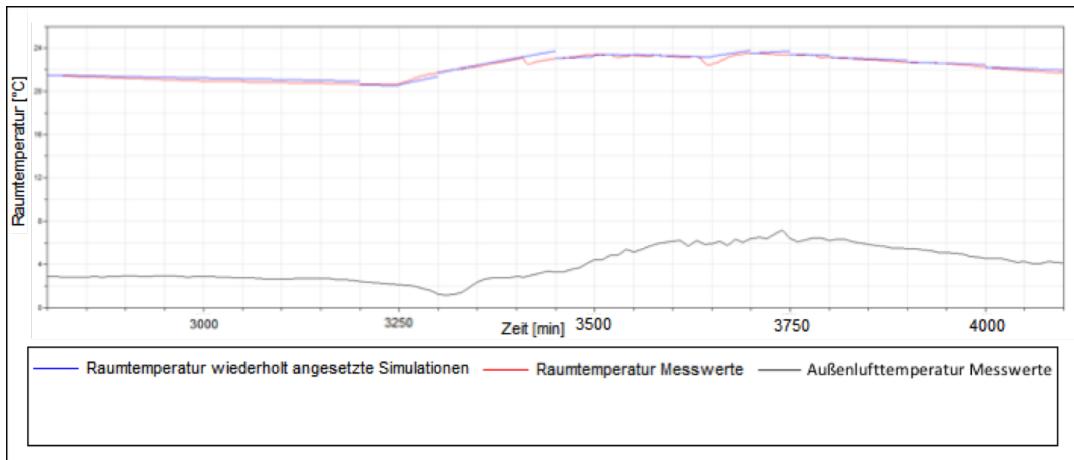


Abb. 4.11: Simulation des Raummodells mit Einsatz des Heizkörpers mit geschätztem Parameter

voller Leistung betrieben. Auf beiden Ergebnisplots folgt die Simulation der Temperaturkurve sehr gut, wenn auch die Diskrepanz zum Ende des Intervalls hin zunimmt. Mit einer maximalen Temperaturabweichung bei beiden Intervallen mit einem Betrag von $0,2^{\circ}\text{C}$ beschreibt das Modell das reale Verhalten damit ausreichend.

Eine erneute Simulation des vorherigen Simulationsintervalls mit dem Einsatz des Heizkörper verdeutlicht eine Verbesserung des Modells. Die Ergebnisse sind im Plot in Abb. 4.10 dargestellt.

Die Abweichungen der Kurve sind deutlich verbessert sowie die Verzögerung hat sich verkürzt. Die Dynamik hat sich für das eintägige Intervall ebenfalls verbessert, wenn auch nur leicht.

Abschließend erfolgt eine Modellprädiktive Regelungsnahe Simulation des Modells. Das Ergebnis ist in Abb. 4.11 dargestellt.

Es wurden wiederholt kürzere Simulationen mit einer angepassten Initialtemperatur angesetzt, um eine Struktur ähnlich der Modellprädiktiven Regelung zu erhalten. Die Außenlufttemperatur bewegte sich während des Zeitraums knapp über dem Gefrierpunkt und es wurde lediglich eine schwache Globalstrahlung detektiert. Beim Einsatz der Heizung wurden kürzere Simulationsintervalle von 20 Minuten angesetzt, ansonsten wurden die Simulationen für längere Intervalle durchgeführt. Nach 800 simulierten Minuten setzt die Heizung ein, was durch das Modell gut abgebildet wird. Es bestehen lediglich Abweichungen an zwei kleineren Abfällen der gemessenen Temperatur, die sich durch Störgrößen wie beispielsweise das Öffnen der Fenster erklären lassen, und daher vom Modell nicht berücksichtigt wurden.

Zusammenfassend lässt sich feststellen, dass das Modell eine solide Beschreibung des Verhaltens von K004b darstellt und sich damit für den Einsatz in der Modellprädiktiven Regelung eignet. Im letzten Abschnitt wird überprüft, ob das Modell den Anforderungen für den Einsatz mit JMODELICA.ORG genügt.

4.3 Modellprädiktive Regelung

Um eine Modellprädiktive Regelung in JMODELICA.ORG anhand des Raummodells zu ermöglichen, wird das Modell in ein Optimierungsfähiges Objekt übersetzt. Dazu wird das Modell mithilfe des JMODELICA.ORG Compilers in ein JMU und Optimization Object transferiert anhand des Programms in Listing 4.8.

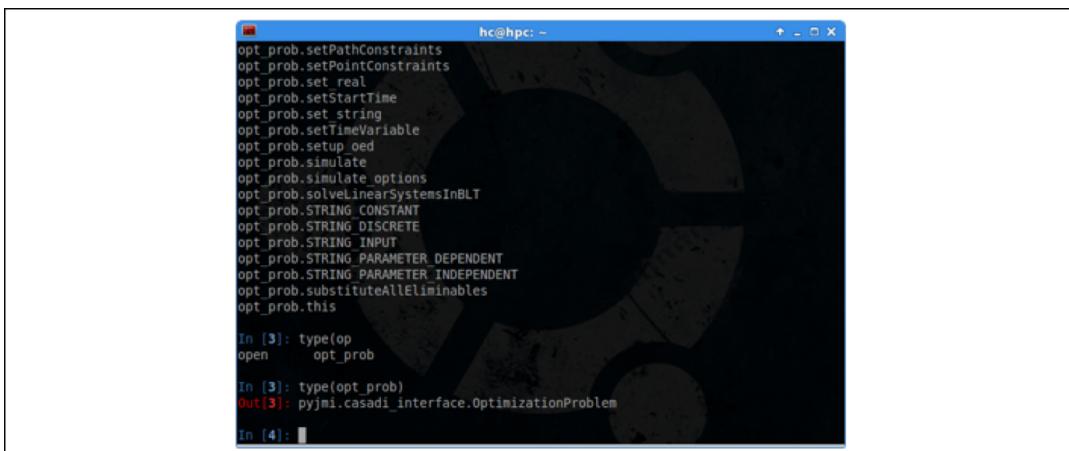
```
from pymodelica import compile_jmu
room_jmu=compile_jmu('room_model_backup.RoomRadiator', '/home/hc/jmtest/
room_model_backup.mo')

from pyjmi import transfer_optimization_problem
opt_prob = transfer_optimization_problem('room_model_backup.RoomRadiator', '/home/hc/
jmtest/room_model_backup.mo', accept_model=True)
```

Listing 4.8: Programm zur Übersetzung des Raummodells in Optimierungsfähige Objekte in JMODELICA.ORG

Trotz der Ausgabe von kleineren Warnungen, lässt sich das Modell übersetzen und damit für Optimierungszwecke in JMODELICA.ORG nutzen. Eine Bestätigung in der Python Konsole für das erfolgreich übersetzte Problem findet sich in Abb. 4.12.

Die Hypothese, dass sich das Modell für den Einsatz mit JMODELICA.ORG eignet, kann also bestätigt werden. Damit sind die Voraussetzungen für eine Modellprädiktive Regelung der Raumtemperatur in K004b geschaffen.



The screenshot shows a terminal window titled "hc@hpc: ~" with a black background and white text. The window displays a list of Python functions and their descriptions:

```
opt_prob.setPathConstraints
opt_prob.setPointConstraints
opt_prob.set real
opt_prob.setStartTime
opt_prob.set_string
opt_prob.setTimeVariable
opt_prob.setup_oed
opt_prob.simulate
opt_prob.simulate_options
opt_prob.solveLinearSystemsInBLT
opt_prob.STRING CONSTANT
opt_prob.STRING DISCRETE
opt_prob.STRING INPUT
opt_prob.STRING PARAMETER_DEPENDENT
opt_prob.STRING PARAMETER_INDEPENDENT
opt_prob.substituteAllEliminables
opt_prob.this

In [3]: type(op
open      opt_prob

In [3]: type(opt_prob)
Out[3]: pyjmi.casadi_interface.OptimizationProblem

In [4]:
```

Abb. 4.12: JMODELICA.ORG Kompabilität des Modells

„Alles Wissen besteht in einer sicheren und klaren Erkenntnis.“

— RENÉ DESCARTES, französischer Philosoph,
Mathematiker und Naturwissenschaftler

5 Schlussbetrachtung

5.1 Fazit

Das Ziel der Arbeit war die Untersuchung, wie eine Anlage zur Raumtemperaturregelung und ein mathematisches Modell derselben aufgebaut sein kann, um eine Modellprädiktive Regelung der Raumtemperatur zu ermöglichen. Zur Beantwortung dieser Frage wurde im Verlauf der Arbeit eine konkrete Anlage installiert und ein Modell entwickelt.

Zunächst wurden im dritten Kapitel die besonderen Anforderungen für eine Modellprädiktive Regelung analysiert und daraus eine erste Idee entwickelt. Die Idee bestand in der Automatisierung eines Raumes der Hochschule Karlsruhe, der als zukünftige Forschungsumgebung genutzt werden kann. Durch die Nutzung von einfachen Temperatursensoren und eines Stellantriebs zur Ansteuerung des bestehenden Heizkörpers, wurde die Idee in die Praxis umgesetzt. Die Kommunikation innerhalb der Anlage wurde durch eine klare Struktur möglichst einfach und übersichtlich gehalten. Durch den Einsatz von Interfaces wird weiterhin eine einfache Ansteuerung der Anlage erlaubt, auch ohne detaillierte Fachkenntnisse zur Anlage und den verwendeten Kommunikationstechnologien. Abschließend wurde eine Zweipunktregelung zur Inbetriebnahme der Anlage eingesetzt, anhand derer die Fähigkeit zur Regelung einer Raumtemperatur gezeigt wurde.

Damit konnte die Hypothese belegt werden, dass sich die spezifizierte Anlage zur Regelung einer Raumtemperatur eignet.

Der anschließende Teil beschäftigte sich mit der Bildung eines Raummodells. Das Modell wurde für den Betrieb einer Modellprädiktiven Regelung der zuvor installierten Anlage entwickelt. Die daraus resultierenden Anforderungen an das Modell wurden beim der schrittweisen Aufbau beachtet. Das Modell wurde auf Validität untersucht, wobei festgestellt wurde, dass die grundlegende Dynamik der realen Prozesse ausreichend beschrieben wird. Anschließend wurde eine Parameterschätzung mithilfe von Bürger [2016] durchgeführt, um die Modellgüte zu verbessern. Nachdem das Modellverhalten anhand von weiteren Simulationen untersucht wurde, konnte auch eine prinzipielle Eignung des Modells einen Einsatz mit der Modellprädiktiven Regelung festgestellt werden. Abschließend wurde durch eine Übersetzung des Modells in ein Optimierungsfähiges Objekt gezeigt, dass es für die Modellprädiktive Regelung in der Plattform JMODELICA.ORG geeignet ist.

Dadurch konnte eine weitere Hypothese belegt werden, dass das Modell für die Modellprädiktive Regelung mithilfe der Plattform *JModelica.org* geeignet ist.

Aufgrund der vorgestellten Ergebnisse und durch die Bestätigung der beiden Thesen, kann die Beantwortung auf die Forschungsfrage erfolgen.

und ein Modell

5.2 Ausblick und Ansatzpunkte für weitere Arbeiten

Dazu werden im ersten Schritt auf Basis von aktuellen Messungen des Systemzustandes und der vorgegebenen Steuergrößen zu jedem diskreten Steuerungszeitpunkt optimale Steuerungspläne bestimmt werden.. Im nächsten Schritt soll die Modellprädiktive Regelung Vorhersagewerte für den äußeren, nicht beeinflussbaren Steuergrößen zur die Bestimmung der optimalen Steuersignale nutzen.

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

Annahme homogene Raumtemperatur bzw einschwingvorgang überprüfen ob modellrelevant Sonnenstrahlung 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:

- das Raummodell,
- die vorgestellten Programme,
- und die Messdaten zu den Simulationen

B Das Raummodell

```

1 package room_model_backup

3   connector Temperature
4     Modelica.Slunits.Conversions.NonSlunits.Temperature_degC t;
5   end Temperature;

7   connector HeatFlow
8     Modelica.Slunits.HeatFlowRate qdot;
9   end HeatFlow;

11  connector MassTemperature
12    Modelica.Slunits.Conversions.NonSlunits.Temperature_degC t;
13    Modelica.Slunits.MassFlowRate mdot;
14  end MassTemperature;

15  connector RadiantEnergyFluenceRate
16    Modelica.Slunits.DensityOfHeatFlowRate radiation_sun ;
17  end RadiantEnergyFluenceRate;

19  model CV_Radiator "control volume for a discretized radiator"
20

21  /** parameter */
22  outer parameter Modelica.Slunits.CoefficientOfHeatTransfer u_radiator
23    "heat transfer coefficient of the radiator";
24  outer parameter Modelica.Slunits.SpecificHeatCapacity cp_water
25    " specific heat capacity of water";
26  outer Modelica.Slunits.Mass cv_m
27    "mass within one control volume";
28  outer Modelica.Slunits.Area exchange_surface
29    "surface of one control volume at which heat transfer takes place";
30  Modelica.Slunits.Energy cv_u
31    "inner energy of the control volume";
32  Modelica.Slunits.HeatFlowRate cv_qdot
33    "heatflowrate over the borders of the control volume";
34  Modelica.Slunits.Conversions.NonSlunits.Temperature_degC cv_temperature_out(start=21.2,
35    fixed=true)
36    "temperature of the fluid leaving the control volume";

37  /** states */
38  outer Modelica.Slunits.Conversions.NonSlunits.Temperature_degC room_temperature_cv
39    "temperature within the room";

41  /** controls */
42  Modelica.Slunits.Conversions.NonSlunits.Temperature_degC cv_temperature_in=inlet.t
43    "temperature of the fluid streaming in the control volume";
44  outer Modelica.Slunits.MassFlowRate mdot
45    "massflowrate within the radiator";
46  [...]
47  equation
48    /* calculate inner energy within one control volume*/
49    cv_u = cv_m * cp_water * cv_temperature_out;
50    /* calculate derival of the inner energy of one control volume*/
51    der(cv_u) = mdot * cp_water * (cv_temperature_in - cv_temperature_out) - cv_qdot;
52    /* calculate heatflowrate of the control volume*/
53    cv_qdot = u_radiator * exchange_surface_cv * (cv_temperature_out - room_temperature_cv
54      );
55    /* commit calculated temperature */
56    outlet.t = cv_temperature_out;
57  end CV_Radiator;

```

```

59 model Radiator "model for a discretized radiator within a room"

61  /** parameter */
62  parameter Real radiator_element_number=106
63   "number of normed elements of which the radiator consists";
64  parameter Real radiator_tubes_element=2
65   "number of parallel tubes in one element";
66  parameter Integer cv_number = 20
67   "number of control volumes in which the radiator is discretized";
68  parameter Modelica.SIunits.Length radiator_element_length=0.045
69   "length of one element depending on type (Recknagel 2013/2014: Heizung und Klimatechnik
70    S.815ff.)";
71  parameter Modelica.SIunits.Height tube_length_vertical=0.4
72   "height of one element depending on type (Recknagel 2013/2014: Heizung und Klimatechnik
73    S.815ff.)";
74  parameter Modelica.SIunits.Diameter tube_diameter_horizontal=0.05
75   "diameter of the horizontal tubes depending on type (Recknagel 2013/2014: Heizung und
76    Klimatechnik S.815ff.)";
77  parameter Modelica.SIunits.Diameter tube_diameter_vertical=0.0255
78   "diameter of the vertical tubes depending on type (Recknagel 2013/2014: Heizung und
79    Klimatechnik S.815ff.)";
80  parameter Modelica.SIunits.Density rho_water = 1000
81   "density of water";
82  parameter Modelica.SIunits.Mass radiator_element_mass=0.35
83   "mass within one element of the radiator depending on type (Recknagel 2013/2014:
84    Heizung und Klimatechnik S.815ff.)";
85  inner parameter Modelica.SIunits.CoefficientOfHeatTransfer u_radiator = 12.9872
86   "heat transfer coefficient of the radiator";
87  inner parameter Modelica.SIunits.SpecificHeatCapacity cp_water = 4200
88   "specific heat capacity of water";
89  CV_Radiator[cv_number] cv_radiator
90   "array of control volumes to discretize the radiator";
91  Modelica.SIunits.HeatFlowRate radiator_qdot_out
92   "heatflow which is leaving the radiator";
93  Modelica.SIunits.Conversions.NonSIunits.Temperature_degC radiator_temperature_out
94   "calculated (predicted) temperature of the water leaving the radiator";
95  inner Modelica.SIunits.Mass cv_m
96   "mass within one control volume";
97  inner Modelica.SIunits.Area exchange_surface
98   "surface of one control volume at which h";

99  /** states */
100 inner Modelica.SIunits.Conversions.NonSIunits.Temperature_degC room_temperature_cv
101  "temperature within the room for the control volume";
102 outer Modelica.SIunits.Conversions.NonSIunits.Temperature_degC room_temperature
103  "temperature within the room from the room";

104  /** controls */
105 inner Modelica.SIunits.MassFlowRate mdot=inlet.mdot
106  "massflowrate within the radiator";
107  Modelica.SIunits.Conversions.NonSIunits.Temperature_degC radiator_inlet = inlet.t
108  "temperature of the inflowing fluid";
109  [...]
110  equation
111   /* calculate exchange surface of one control volume */
112   exchange_surface_cv = (radiator_element_number * radiator_element_length * 2 *
113    Modelica.Constants.pi * tube_diameter_horizontal + radiator_element_number *
114    radiator_tubes_element * tube_length_vertical * Modelica.Constants.pi *
115    tube_diameter_vertical)/cv_number;
116   /* calculate mass within one control volume */
117   cv_m = (radiator_elment_mass * radiator_element_number)/cv_number;
118   /* commit temperature of radiator fluid inlet to the first control volume */
119   cv_radiator [1]. inlet.t = radiator_inlet ;
120   /* connect the control volumes within the radiator */

```

```

115  for i in 1 : (cv_number-1) loop
116    connect( cv_radiator[i].outlet, cv_radiator[i+1].inlet );
117  end for;
118  /* calculate and commit the heatflow which is leaving the radiator */
119  radiator_qdot = sum(cv_radiator.cv_qdot);
120 end Radiator;
121
122 model Room_radiator_window "model of a room for mpc purpose with JModelica.org"
123
124  /** parameter p */
125  parameter Modelica.Slunits.Length room_length=7.81
126    "length of the room";
127  parameter Modelica.Slunits.Breadth room_breadth=5.78
128    "breadth of the room";
129  parameter Modelica.Slunits.Height room_height=2.99
130    "height of the room";
131  parameter Modelica.Slunits.Length window_length=7
132    "length of the window";
133  parameter Modelica.Slunits.Height window_height=2.08
134    "height of the window";
135  parameter Modelica.Slunits.Density rho_air = 1.2
136    "density of air";
137  parameter Modelica.Slunits.CoefficientOfHeatTransfer u_glass=2.0
138    "heat transfer coefficient for glass ";
139  parameter Modelica.Slunits.CoefficientOfHeatTransfer u_wall=0.612986
140    "heat transfer coefficient for the walls of the room";
141  parameter Modelica.Slunits.SpecificHeatCapacity cp_air=1000
142    "specific heat capacity of air";
143 Radiator heating
144  "instance of a radiator";
145 Window window
146  "instance of a window";
147 Modelica.Slunits.Volume room_volume
148  "volume of the room";
149 Modelica.Slunits.Mass room_mass
150  "mass of air within the room";
151 Modelica.Slunits.Area building_surface
152  "sum of contacting surfaces (walls) with other rooms of the building";
153 Modelica.Slunits.Area environment_surface
154  "sum of contacting surfaces (walls) with the environment";
155 inner Modelica.Slunits.Area window_surface
156  "sum of contacting surfaces (windows) with the environment";
157 Modelica.Slunits.Energy room_u
158  "inner energy of the system room";
159 Modelica.Slunits.HeatFlowRate building_qdot
160  "rate of heat flow with other rooms within the building";
161 Modelica.Slunits.HeatFlowRate environment_qdot
162  "rate of heat flow with the environment";
163 Modelica.Slunits.HeatFlowRate environment_qdot_wall
164  "rate of heat flow with the environment through the wall";
165 Modelica.Slunits.HeatFlowRate environment_qdot_window
166  "rate of heat flow with the environment through the window";
167 Modelica.Slunits.HeatFlowRate qdot_loss
168  "summed up rate of heatflow leaving the system";
169 Modelica.Slunits.HeatFlowRate radiator_qdot
170  "heat flow rate at the radiator surfaces streaming into the room";
171
172  /** states x */
173 inner Modelica.Slunits.Conversions.NonSlunits.Temperature_degC room_temperature(start
174    =24, fixed=true)
175    "temperature within the room ( Initially about 24 degree celsius )";
176
177  /** controls u */
178 Modelica.Slunits.MassFlowRate mdot=inlet_radiator.mdot

```

```

179   "commitment of the massflowrate to the radiator";
Modelica.SIunits.Conversions.NonSIunits.Temperature_degC environment_temperature=
    inlet_environment.t
  "temperature of the environment";
181 Modelica.SIunits.Conversions.NonSIunits.Temperature_degC building_temperature=
    inlet_building.t
  "temperature of the rest of the building";
183 Modelica.SIunits.HeatFlowRate qdot_sun
  "rate of heat flow brought in by the sun";
185 Modelica.SIunits.HeatFlowRate qdot_otherfactors=inlet_other.qdot
  "rate of heat flow brought in by other factors (e.g. people, computer)";

187 equation
189 /* calculate room volume */
room_volume=room_length*room_height*room_breadth;
191 /* calculate room mass */
room_mass=room_volume*rho_air;
193 /* calculate surface of the room with other rooms of the building */
building_surface=(room_length*room_breadth*2)+(room_length*room_height*2);
195 /* calculate wall surface of the room with the environment */
environment_surface=(room_length*room_height)+(room_breadth*room_height)-
    window_surface;
197 /* calculate window surface of the room with the environment */
window_surface=(window_length*window_height);
199 /* calculate inner energy*/
room_u = room_mass * cp_air * room_temperature;
201 /* calculate derival of the inner energy */
der(room_u) = radiator_qdot + qdot_loss + qdot_sun + qdot_otherfactors;
203 /* sum up the lost heat flow */
qdot_loss = building_qdot+environment_qdot;
205 /* calculate lost heatflow with other rooms of the building */
building_qdot = u_wall * building_surface * (building_temperature-room_temperature);
207 /* sum up the lost heatflow with the environment */
environment_qdot = environment_qdot_window + environment_qdot_wall;
209 /* calculate lost heatflow with the environment through the window */
environment_qdot_window = u_glass * window_surface * (environment_temperature -
    room_temperature);
211 /* calculate lost heatflow with the environment through the wall */
environment_qdot_wall = u_wall * environment_surface * (environment_temperature -
    room_temperature);
213 /* commit the inflowing heat flow of the radiator */
radiator_qdot=heating.radiator_qdot_out;
215 /* connect radiator and sun with room */
connect( heating.inlet, inlet_radiator );
217 connect(inlet_sun, window.inlet_sun);
qdot_sun = window.outlet_room.qdot;
219 end Room_radiator_window;

221 model SourceTemp
  Modelica.SIunits.Conversions.NonSIunits.Temperature_degC t=inlet;
  Modelica.Blocks.Interfaces.RealInput inlet ;
equation
225 outlet.t =t;
end SourceTemp;

227 model SourceHeat
  Modelica.SIunits.HeatFlowRate qdot=inlet;
  Modelica.Blocks.Interfaces.RealInput inlet ;
equation
231 outlet.qdot=qdot;
end SourceHeat;

235 model SourceTempMass
  Modelica.SIunits.Conversions.NonSIunits.Temperature_degC t=inlet_t;

```

```
237  Modelica.SIunits.MassFlowRate mdot=inlet_mdot;
238  Modelica.Blocks.Interfaces.RealInput inlet_t ;
239  Modelica.Blocks.Interfaces.RealInput inlet_mdot;
240  equation
241    outlet.t =t;
242    outlet.mdot=mdot;
243  end SourceTempMass;

245  model Window
246    Modelica.SIunits.DensityOfHeatFlowRate radiation_arriving =inlet_sun.radiation_sun ;
247    Modelica.SIunits.HeatFlowRate qdot_effective ;
248    parameter Real window_transmission = 0.00687213;
249    outer Modelica.SIunits.Area window_surface;
250    equation
251      qdot_effective = radiation_arriving * window_transmission * window_surface;
252      qdot_effective = outlet_room.qdot;
253  end Window;

255  model SourceSun
256    Modelica.SIunits.DensityOfHeatFlowRate radiation_sun = inlet ;
257    Modelica.Blocks.Interfaces.RealInput inlet ;
258    RadianEnergyFluenceRate outlet;
259    equation
260      outlet.radiation_sun = radiation_sun;
261  end SourceSun;
262
263 end room_model_backup;
```

C Datenblätter

C.1 EX9132C-2-MTCP Gateway von ExpertDAQ

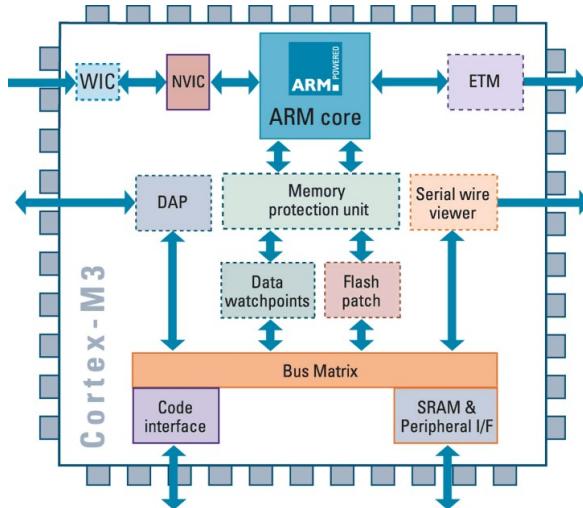
Operation Manual of EX9132C-2-MTCP

Modbus TCP to Modbus RTU/ASCII Converter



NOTE: Please notify your sales representative if any of the above items is missing or damaged.

1.3 Block Diagram



Low-cost devices usually are equipped with low speed processors and limited memories. In reality, they have neither the capability nor practicality to manage complicated network TCP/IP protocols. The ARM Cortex™-M3 32-bit processor has been specifically developed to provide a high-performance, low-cost platform for a broad range of applications including microcontrollers, automotive body systems, and industrial control systems, networking by converting data stream between network TCP/IP and popular serial port signals.

Instead of processing TCP/IP packets directly, devices need only deal with those interface signals, which greatly simplifies the complexity of TCP/IP network in linkage. The ARM Cortex-M3 processor provides outstanding computational performance and exceptional system response to interrupt while meeting low cost requirements through small core footprint, industry leading code density enabling smaller memories, reducing pin count, and low power consumption.

The central core of ARM Cortex-M3 processor, based on a 3-stage pipeline Harvard bus architecture, incorporates advanced features including single cycle multiply and hardware divide to deliver an outstanding efficiency of 1.25 DMIPS/MHz. The ARM Cortex-M3 processor also implements the new Thumb®-2 instruction set architecture, which combined with features such as unaligned data storage and atomic bit manipulation delivers 32-bit performance at a cost equivalent to modern 8- and 16-bit devices.

1.4 Product Features

- Data Conversion between RS232 and RS422/RS485 and Ethernet
EX9132C-2-MTCP converter device (RS232 • 1 port, RS232/RS422/RS485 • 1 port) data/signal into the Modbus TCP/IP package data/signal and send them out with the Ethernet Data Stream; or convert the Modbus TCP/IP package data/signal into serial device data/signal.
- Socket Communication
EX9132C-2-MTCP is provided one socket connection.
- Digital I/O Activating (Optional)
EX9132C-2-MTCP provides eight TTL of digital I/O.

Convert the sensors' statuses (the sensors are connected to the converter) into the TCP/IP package data and send them out with the Ethernet Data Stream; or use the TCP/IP package data to activate/deactivate the specified digital outputs.

- Dynamic IP Configuration
Support DHCP client mode, simplifying network address configuration and management.
- Dual LAN Speed
Support 10/100 Mbps Ethernet, auto-detected.
- Server / Client Dual Modes
EX9132C-2-MTCP converter device can be configured as network server or network client. In the client mode, it can be installed in network which is protected by NAT router or firewall without a real IP address.
- Web-based Setup
Parameters setup is based on HTTP protocol by using standard browsers (IE and Netscape). No special software would be required.
- Built-in Security Control
Security protect by login password to prevent intruders.
- Remote Update
Firmware can be updated directly via Ethernet network to keep up with latest network standards.

1.5 Product Specifications

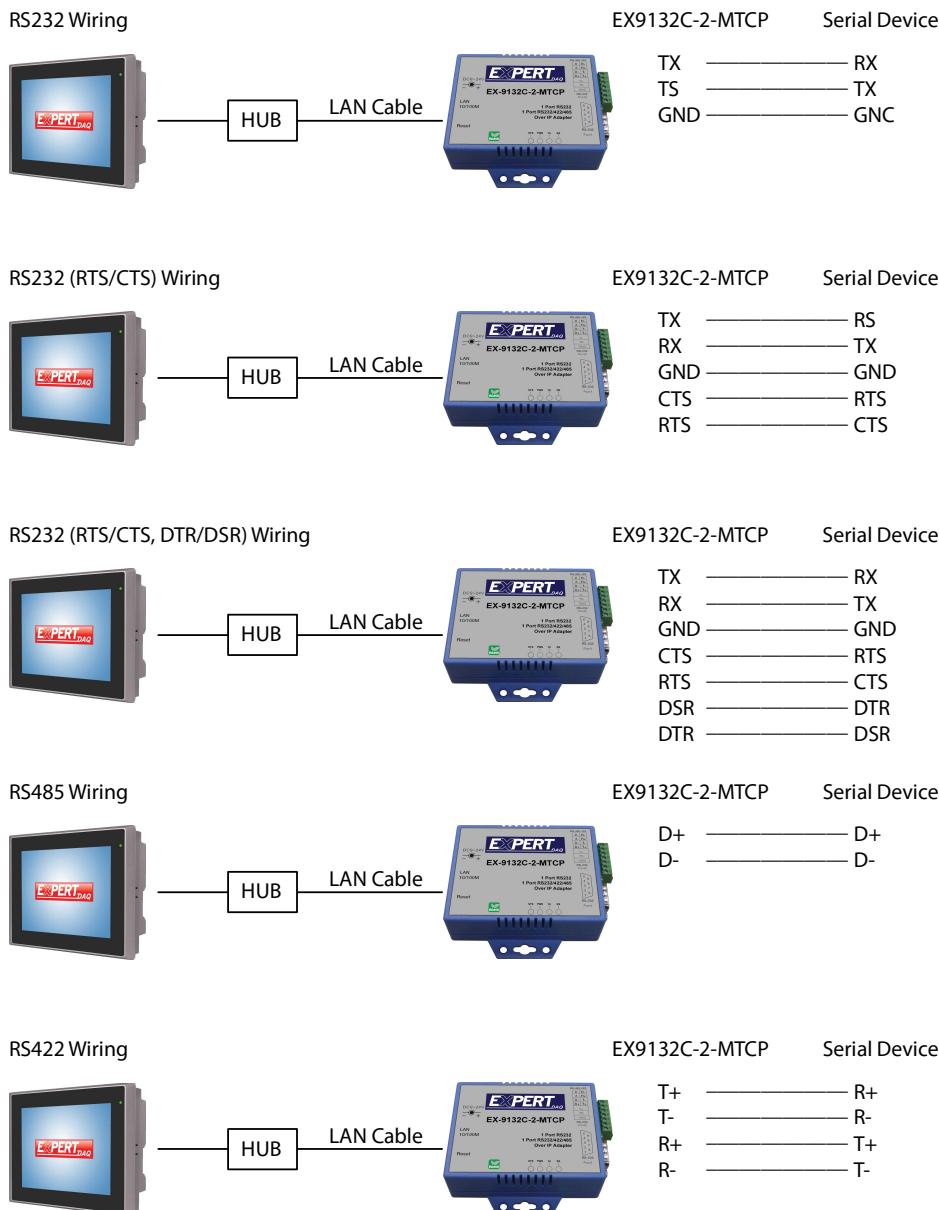
- CPU: ARM Cortex™-M3 32-bit processor, 50MHz
- RAM: 64K Bytes SRAM
- ROM: 256K Bytes Flash ROM
- Ethernet
 - Port Type: RJ-45 Connector
 - Speed: 10 /100 M bps (Auto Detect)
 - Protocol:
 - HTTP, DHCP
 - Modbus-TCP Master to Modbus-RTU Slave
 - Modbus-TCP Master to Modbus-ASCII Slave
 - Modbus-RTU Master to Modbus-TCP Slave
 - Modbus-ASCII Master to Modbus-TCP Slave
 - Mode: TCP Server/TCP Client/UDP Client / Virtual COM / Pairing
 - Setup: HTTP Browser Setup (IE & Netscape)
 - Security: Login Password

- Protection: Built-in 1.5KV Magnetic Isolation
- Serial Port:
 - RS232 • 1 Port
 - RS232/RS422/RS485(Auto-Detect) * 1 Port
 - Speed: 300 bps-115.2K bps
 - Parity: None, Odd, Even, Mark, Space
 - Data Bit: 5, 6, 7, 8
 - Stop Bit: 1, 2
- Port 1
 - RS232 Port
 - RS232 Signal: Rx, Tx, GND, RTS, CTS, DTR, DSR, DCD
 - Connector: DB9 male
- Port 2
 - RS232/RS422 /RS485 Port (Auto-Detect)
 - Built-in RS422/RS485 Pull High-Low Resistor
 - RS232 Signal: Rx, Tx, GND
 - RS422 Signal: Rx+, Rx-, Tx+, Tx- (Surge & Over Current Protection)
 - RS485 Signal: Data+, Data- (Surge & Over Current Protection)
 - Connector: Terminal Block
- Socket Connection: 1 Connection
- 15KV ESD for all signal
- Watch Dog Function
- Virtual Support Windows 2000 /2003 / XP / Vista / 7 / 10
- Firmware On-line Updated Via Ethernet
- Power: DC 9 - 24V, 500mA (5,5 mm DC Jack / optional: Terminal Block)
- LED Lamp
 - RX/TX Port 1 (Red)
 - RX/TX Port 2 (Green)
 - Di/Do (Red - no function)
 - SYS (Green)

- Environment
 - Operating Temperature: +0°C to 70°C
 - Storage Temperature: -10°C to 80°C
- Dimensions: 115 * 90 * 27 mm (W * D * H)
- Weight: 140 g
- RoHS: Compliant with RoHS
- Regulatory Approvals: FCC, CE
- Warranty: 1 year

2.7 Wiring Architecture

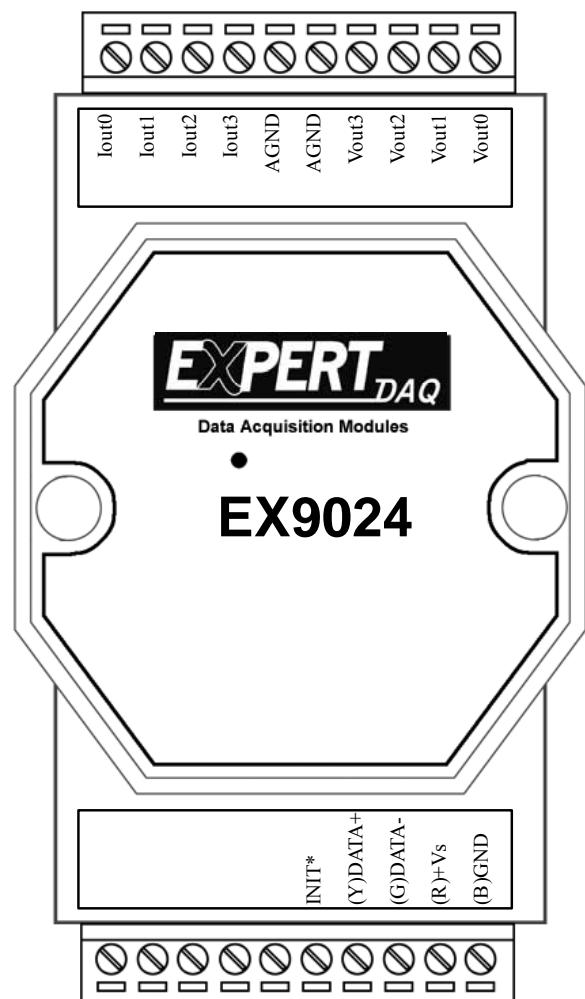
RS422/RS485 Wiring Architecture

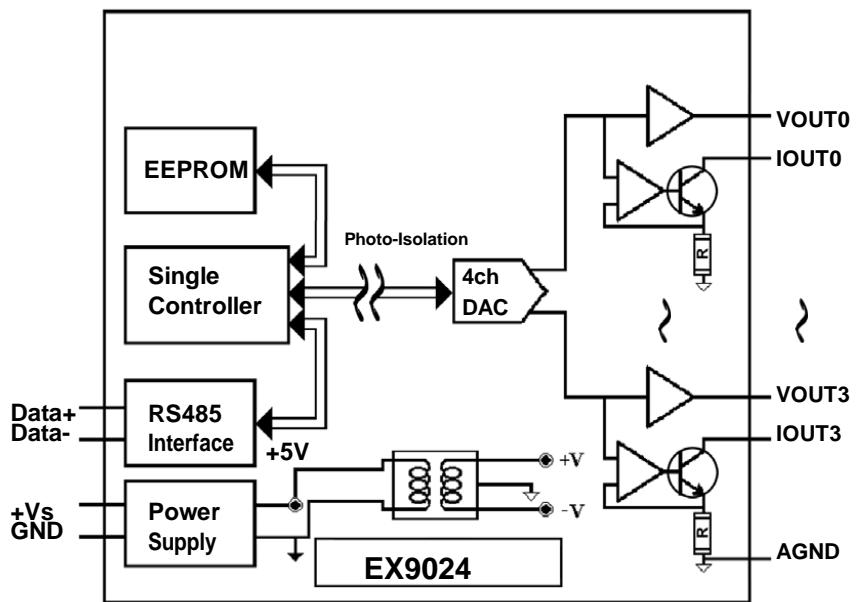


When you finish the steps mentioned above and the LED indicators are as shown in above diagram, the converter is installed correctly. You can use the Setup Tool "EXBrowser.exe" to setup the IP Address.

To proceed the advanced parameter setup, please use a web browser (IE or Netscape) to continue the detailed settings.

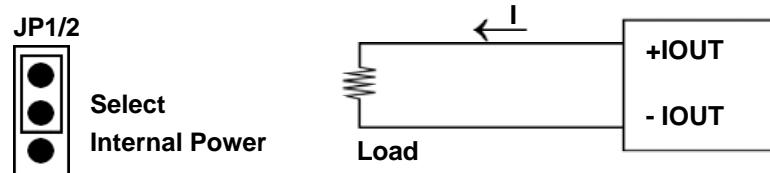
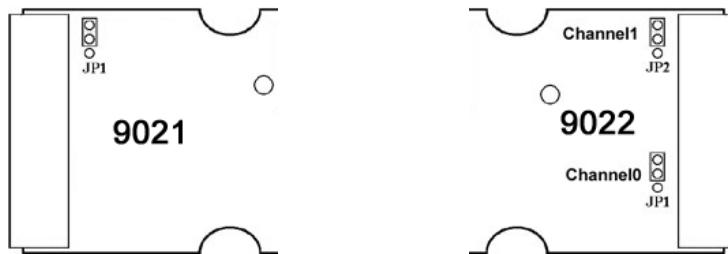
C.2 EX9024M Signalwandler von ExpertDAQ



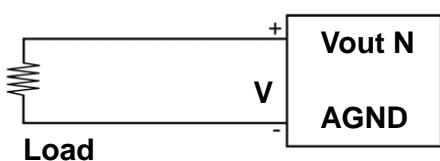


1.4 Jumper Setting & Wire Connection

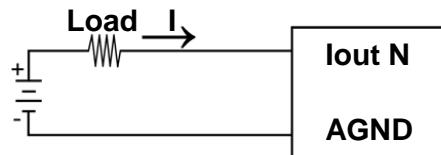
EX9021/21P/22 Current output wire connection



EX9024 Voltage output wire connection



EX9024 Current output wire connection



EX9024-M Quick Start

- 1. The default setting is MODBUS mode after Power On.**
- 2. Using INIT pin to contact with GND pin then Power On will enter Normal mode.**
- 3. Command: \$00P0 is set EX9024-M to Normal mode after Repower On. On normal mode, user can set other setting like address, Baudrate, (Please check the EX9000 user manual).**
- 4. Command: \$AAP1 is set to MODBUS mode after Repower On.**
- 5. Under Normal mode that Command: \$AAP can check which mode it is after Repower On.**

Response:

!AA10=Normal

!AA11=MODBUS

The Modbus protocol was originally developed for Modicon controllers by Modicon Inc. Detailed information can be found at <http://www.modicon.com/techpubs/toc7.html>. Visit <http://www.modbus.org> to find more valuable information.

9000M series modules support the Modbus RTU protocol. The communication Baud Rates range from 1200bps to 115200bps. The parity, data bits and stop bits are fixed as no parity, 8 data bits and 1stop bit. The following Modbus functions are supported.

03(0x03) Read Back Multiple Channel Output Value

Request

00	Address	1 Byte	1-247
01	Function code	1 Byte	0x03
02~03	Starting channel	2 Bytes	0x0000~0x0003
04~05	Channel numbers	2 Bytes	0x0001~0x0004

Response

00	Address	1 Byte	1-247
01	Function code	1 Byte	0x03
02	Byte count	1 Byte	N* x 2
03	Output channel read back value	N* x 2 Byte	0x0000~0x3FFF

N*=Number of output channels

Error Response

00	Address	1 Byte	1-247
01	Function code	1 Byte	0x83
02	Exception code	1 Byte	Refer to the Modbus standard for more details.

06(0x06) Write Single Channel Output

Request

00	Address	1 Byte	1-247
01	Function code	1 Byte	0x06
02~03	Starting channel	2 Bytes	0x0000~0x0003
04~05	Output channel value	2 Bytes	0x0001~0x3FFF Refer Output type & Data Format Table

Response

00	Address	1 Byte	1-247
01	Function code	1 Byte	0x06
02	Starting channel	2 Byte	0x0000~0x0003
03	Output channel value	2 Byte	0x0001~0x3FFF Refer Table A Output type & Data Format

Error Response

00	Address	1 Byte	1-247
01	Function code	1 Byte	0x86
02	Exception code	1 Byte	Refer to the Modbus standard for more details.

Output type & Data Format Table

Type Code	Output Range	Data Format	Max.	Min.
30	0 to 20mA	Hexadecimal	3FFF	1FFF
31	4 to 20 mA	Hexadecimal	3FFF	2665
32	0 to 10V	Hexadecimal	3FFF	1FFF
33	-10V to +10V	Hexadecimal	3FFF	0
34	0 to +5V	Hexadecimal	2FFF	1FFF
35	-5V to +5V	Hexadecimal	2FFF	0FFF

**Channel output value should be in hexadecimal form and should between range of maximum & minimum value that depend on each type code.

16(0x10) Write Multiple Channel Output

Request

00	Address	1 Byte	1-247
01	Function code	1 Byte	0x10
02~03	Starting channel	2 Bytes	0x0000~0x0003
04~05	Output channel numbers	2 Bytes	0x0000~0x0004
06	Byte count	1 Byte	2 x N*
07~	Output channel value	N* x 2 Byte	0x0001~0x3FFF Refer Output type & Data Format Table

N*= Output channel numbers

Response

00	Address	1 Byte	1-247
01	Function code	1 Byte	0x10
02~03	Starting channel	2 Bytes	0x0000~0x0003
04~05	Output channel numbers	2 Bytes	0x0000~0x0004

Error Response

00	Address	1 Byte	1-247
01	Function code	1 Byte	0x90
02	Exception code	1 Byte	Refer to the Modbus standard for more details.

Output type & Data Format Table

Type Code	Output Range	Data Format	Max.	Min.
30	0 to 20mA	Hexadecimal	3FFF	1FFF
31	4 to 20 mA	Hexadecimal	3FFF	2665
32	0 to 10V	Hexadecimal	3FFF	1FFF
33	-10V to +10V	Hexadecimal	3FFF	0
34	0 to +5V	Hexadecimal	2FFF	1FFF
35	-5V to +5V	Hexadecimal	2FFF	0FFF

**Channel output value should be in hexadecimal form and should between range of maximum & minimum value that depend on each type code.

C.3 THERMASGARD RTM1-Modbus

Raumtemperaturfühler von S+S Regeltechnik

THERMASGARD® RTM1- Modbus

Raumbedien-Temperaturfühler ($\pm 3\%$), Aufputz, für Temperatur, relative Feuchte, Taupunkt, kalibrierfähig, mit Modbus-Anschluss

TECHNISCHE DATEN

Spannungsversorgung:	24 V AC ($\pm 20\%$) und 15...36 V DC ($\pm 10\%$)
Leistungsaufnahme:	< 1,0VA / 24 V DC < 2,2VA / 24 V AC
Sensor:	digitaler Feuchtesensor mit integriertem Temperatursensor, kleine Hysterese, hohe Langzeitstabilität
Datenpunkte:	Temperatur, relative Feuchte, Taupunkt, Sollwertpotentiometer
Messbereich:	0...+50 °C
Abweichung Temperatur:	$\pm 0,5K$ bei +20 °C
Nullpunkt-Offset:	$\pm 10^\circ C$, über Potentiometer einstellbar
Umgebungstemperatur:	Lagerung -35...+85 °C; Betrieb 0...+50 °C
Medium:	saubere Luft und nicht aggressiv, nicht brennbare Gase
Signalfilterung:	4 s / 32 s
Prozessanschluss:	mittels Schrauben
Gehäuse:	Kunststoff, Werkstoff ABS, Farbe Reinweiß (ähnlich RAL 9010)
Abmaße:	85 x 85 x 27 mm (Baldur 1)
Montage:	Wandmontage oder auf UP-Dose, Ø 55 mm, Unterteil mit 4-Löch, für Befestigung auf senkrecht oder waagerecht installierten UP-Dosen für Kabeleinführung hinten, mit Sollbruchstelle für Kabeleinführung oben/unten bei AP
zulässige Luftfeuchte:	< 95% r. H., nicht kondensierende Luft
Schutzklasse:	III (nach EN 60 730)
Schutztart:	IP 30 (nach EN 60 529)
Normen:	CE-Konformität, elektromagnetische Verträglichkeit nach EN 61326, nach EMV-Richtlinie 2004 / 108 / EC
Optional:	Display mit Beleuchtung, zweizeilig, programmierbar, Ausschnitt ca. 36 x 15 mm (B x H), zur Anzeige der Ist-Temperatur oder eines individuell programmierbaren Anzeigewertes (Über die Modbuschnittstelle kann das Display sowohl im 7-Segment-Bereich, als auch im Dot-Matrix-Bereich individuell beschrieben werden.)

RTM1-Modbus
Standard

programmierbares
Display

RTM1-Modbus

RTM1-Modbus

Schaltbild

Detailed description of the circuit diagram:
 The diagram shows the internal connections of the RTM1-Modbus module. It includes power inputs (+UB 24VAC/DC, -UB GND), a display socket (Stecker Display), a DIP switch for address (DIP A), a DIP switch for parameters (DIP B), a mode switch, an offset potentiometer, and an RS485 port (Modbus A and Modbus B). A legend defines the symbols used in the diagram.

DIP A: Busadresse
 DIP B: Busparameter (Baudrate, Parity ...)
 Telegramm-Anzeige
 Empfang (LED grün)
 Fehler (LED rot)
 LED (interner Status)
 Offset-Korrektur
 Temperatur: $\pm 10^\circ C$
 Stecker für Display
 Kontaktseite: rechts

C.4 Webthermograph 8x von W&T



Datenblatt:

Web-Thermograph 8x

Artikelnummer: 57608



Messen, loggen und melden...

Der Web-Thermograph 8x ist ein Messgerät welches **acht Temperaturwerte** erfasst und die Werte im Netzwerk zur Verfügung stellt. Das Gerät verfügt über einen integrierten Datenlogger und über zahlreiche Web- und Netzwerkdienste zur manuellen Abfrage der Messwerte, oder selbstständigem Versand von Meldungen.

Eigenschaften:

Sensor:

- **Acht Temperaturmesseingänge Pt100/Pt1000:**
 - Messbereich W&T Fühler: -50°C...180°C
 - Messeingang: -200°C...650°C

Konnektivität:

- Per Browser Temperaturen und Verläufe überwachen
- **Alarm und Berichtsfunktion:**
 - E-Mail zur Alarmierung oder als Berichtsfunktion
 - SNMP-Abfragen /-Alarm Traps
 - Bis zu 8 Alarmmeldungen konfigurierbar
- **Dynamische Integration in andere Webseiten:**
 - Direktzugriff auf aktuelle Messwerte via AJAX, JavaScript und Java-Applet
- **Weitere Software-Schnittstellen zur Einbindung in Ihre Systeme/Datenbanken:**

MULTICAL® 602

DATENBLATT

Verzeichnis

Rechenwerksfunktionen	3
Impulsaus- und Eingänge auf Modulen	10
Kabinetteinrichtung	11
Zugelassene Zählerdaten	12
Elektrische Daten	12
Mechanische Daten	15
Werkstoffbezeichnungen	15
Bestellvorschrift	16
Toleranzband	17
Maßskizzen	18
Zubehör	20

MULTICAL® 602

DATENBLATT

Rechenwerksfunktionen

Energieberechnung

MULTICAL® 602 berechnet die thermische Energie gemäß prEN 1434-1:2009, die die internationale Temperaturskala von 1990 (ITS-90) und die Druckdefinition von 16 bar verwendet.

Die Energieberechnung kann in vereinfachter Form wie folgt ausgedrückt werden:

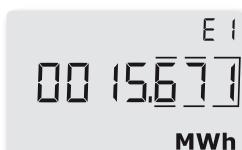
$$\text{Energie} = V \times \Delta\Theta \times k$$

V ist das zugeführte Wasservolumen

$\Delta\Theta$ ist die gemessene Temperaturdifferenz

k ist der Wärmekoeffizient des Wassers

Das Rechenwerk berechnet immer die Energie in [Wh]. Hiernach erfolgt die Umrechnung auf die gewählte Maßeinheit.



E [Wh] =	$V \times \Delta\Theta \times k \times 1000$
E [kWh] =	E [Wh] / 1.000
E [MWh] =	E [Wh] / 1.000.000
E [GJ] =	E [Wh] / 277.780
E [Gcal] =	E [Wh] / 1.163.100

Applikationstypen

MULTICAL® 602 arbeitet mit neun verschiedenen Energieformeln E1...E9, die alle bei jeder Integration parallel berechnet werden, unabhängig von der Konfiguration des Zählers.

Die Energiertypen E1 zu E9 werden wie folgt berechnet:

E1=V1(T1-T2)k Wärmeenergie (V1 in Vor- oder Rücklauf)

E2=V2(T1-T2)k Wärmeenergie (V2 in Rücklauf)

E3=V1(T2-T1)k Kälteenergie (V1 in Vor- oder Rücklauf)

E4=V1(T1-T3)k Vorlaufenergie

E5=V2(T2-T3)k Rücklaufenergie oder Zapfen von Rücklauf

E6=V2(T3-T4)k Zapfwasserenergie, separat

E7=V2(T1-T3)k Zapfwasserenergie von Vorlauf

E8=m³xT1 Die Grundlage für die Berechnung von volumenbasierten Durchschnittstemperaturen in der Vorlauf T1

E9=m³xT2 Die Grundlage für die Berechnung von volumenbasierten Durchschnittstemperaturen in der Rücklauf T2

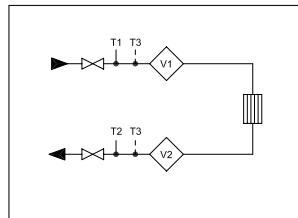
Somit kann MULTICAL® 602 die Wärme- und Kälteenergie der meisten Applikationen, sowohl geschlossener als offener Anlagen, berechnen.

Alle Energiertypen werden protokolliert und können konfigurationsabhängig angezeigt werden.

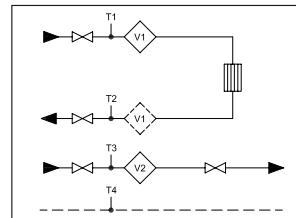
MULTICAL® 602

DATENBLATT

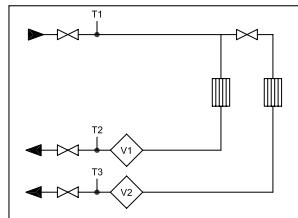
Rechenwerksfunktionen



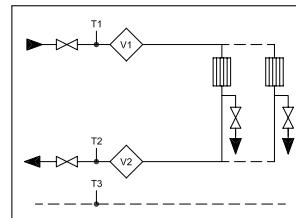
*Beispiel 1:
Geschlossenes thermisches System mit
einem Durchflusssensor*



*Beispiel 2:
Offenes Zweistrangsystem mit zwei
Durchflusssensoren*



*Beispiel 3:
Zwei Wärmekreise mit gemeinsamem
Vorlauf*



*Beispiel 4:
Offenes System mit zwei Durchflusssen-
soren*

Durchflusssmessung

MULTICAL® 602 berechnet den aktuellen Wasserdurchfluss nach zwei verschiedenen Prinzipien abhängig vom angeschlossenen Durchfluszzählertyp:

- Die Durchflussanzeige bei angeschlossenen elektronischen Durchfluszzählern wird alle 10 Sekunden aktualisiert.
- Die Durchflussanzeige bei angeschlossenen mechanischen Durchfluszzählern, normalerweise mit Reed-Schalter, wird auf der Basis einer Periodenzeitzählung berechnet und wird bei jedem Volumenimpuls aktualisiert.



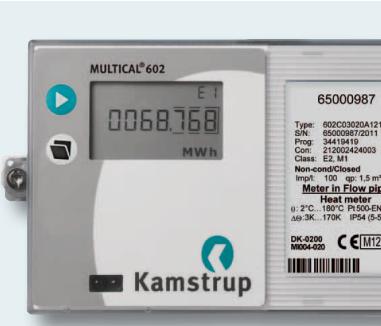
C.5 MULTICAL 602 Wärmemengenzähler von Kamstrup

MULTICAL® 602

DATENBLATT

- Komplettes Sortiment von Kommunikationsmodulen
- High Power FunkRouter-Modul
- Datenlogger
- Info-Logger
- Datenbackup bei Stromausfall

EN 1434 MID-2004/22/EG



Wärme- und Kältezähler mit unbegrenzter Kommunikation

Anwendung

MULTICAL® 602 ist ein universelles Rechenwerk zur Wärme- und Kältemessung zusammen mit den meisten impulsgebenden Durchflusssensoren sowie einem 2- oder 4-Leiter Temperaturfühlerpaar. In Kombination mit dem Kamstrup Ultraschalldurchflusssensor ULTRAFLOW® bietet der Zähler noch erweiterte Funktionen. Dank seiner hohen Messgenauigkeit registriert der Zähler den genauen Verbrauch über die ganze Lebensdauer des Zählers. Der Zähler ist wartungsfrei, hat eine lange Lebensdauer, und garantiert so- mit minimale jährliche Betriebskosten.

MULTICAL® 602 wird zur Wärme-, Kälte- und kombinierten Wärme-/Kältemessung in allen wasserbasierten Anlagen mit Temperaturen von 2°C bis 180°C für Wärme und 2°C bis 50°C für Kälte verwendet.

Funktion

In Wärmeapplikationen wird MULTICAL® 602 zusammen mit dem Durchflusssensor ULTRAFLOW® 54 verwendet. Die Durchflussgrößen decken den Bereich von qp 0,6 m³/h bis qp 1.000 m³/h.

In Kälteapplikationen bis zu qp 100 m³/h wird MULTICAL® 602 zusammen

mit ULTRAFLOW® 14 und von qp 150 m³/h bis qp 1.000 m³/h zusammen mit ULTRAFLOW® 54 verwendet.

Das Rechenwerk kann an Durchflusssensoren bis zu qp 3.000 m³/h angeschlossen werden.

MULTICAL® 602 auszeichnet sich durch die komplette Auswahl von Kommunikationsmodulen und die eingebaute RTC (Echtzeituhr), die es leicht machen, den Zähler in allen Applikationen anzupassen, unabhängig von der Auslesemethode.

Für drahtgebundene Kommunikation kann der Zähler mit LON, SIOX, M-Bus, Datenmodul sowie den neuen Lösungen Metasys N2 und Ethernet/IP ausgestattet werden. Wird der Zähler in ein drahtloses Netzwerk integriert, können Sie Funk, Wireless M-Bus oder eines der neuen Module: GSM/GPRS oder High Power RadioRouter wählen.

Die InfoCodes und Datenlogger des Rechenwerks stellen ein unschätzbares Werkzeug zur Fehlersuche, Fehlerberichtigung und Analyse des Energieverbrauchs dar. Das Infologger überwacht ständig eine Reihe wichtiger Funktionen im Zähler, z.B. Fehler im Messsystem, Stromausfall, Leckage, Bersten, oder Montage des Durchflusssensors mit falscher Durch-

flussrichtung installiert ist. In solchen Fällen erscheinen eine blinkende "INFO" und eine InfoCode im Display.

MULTICAL® 602 speichert die Verbrauchsdaten jährlich, monatlich, täglich und stündlich, welches eine komplette Betriebsanalyse erleichtert.

Betriebsoptimierung

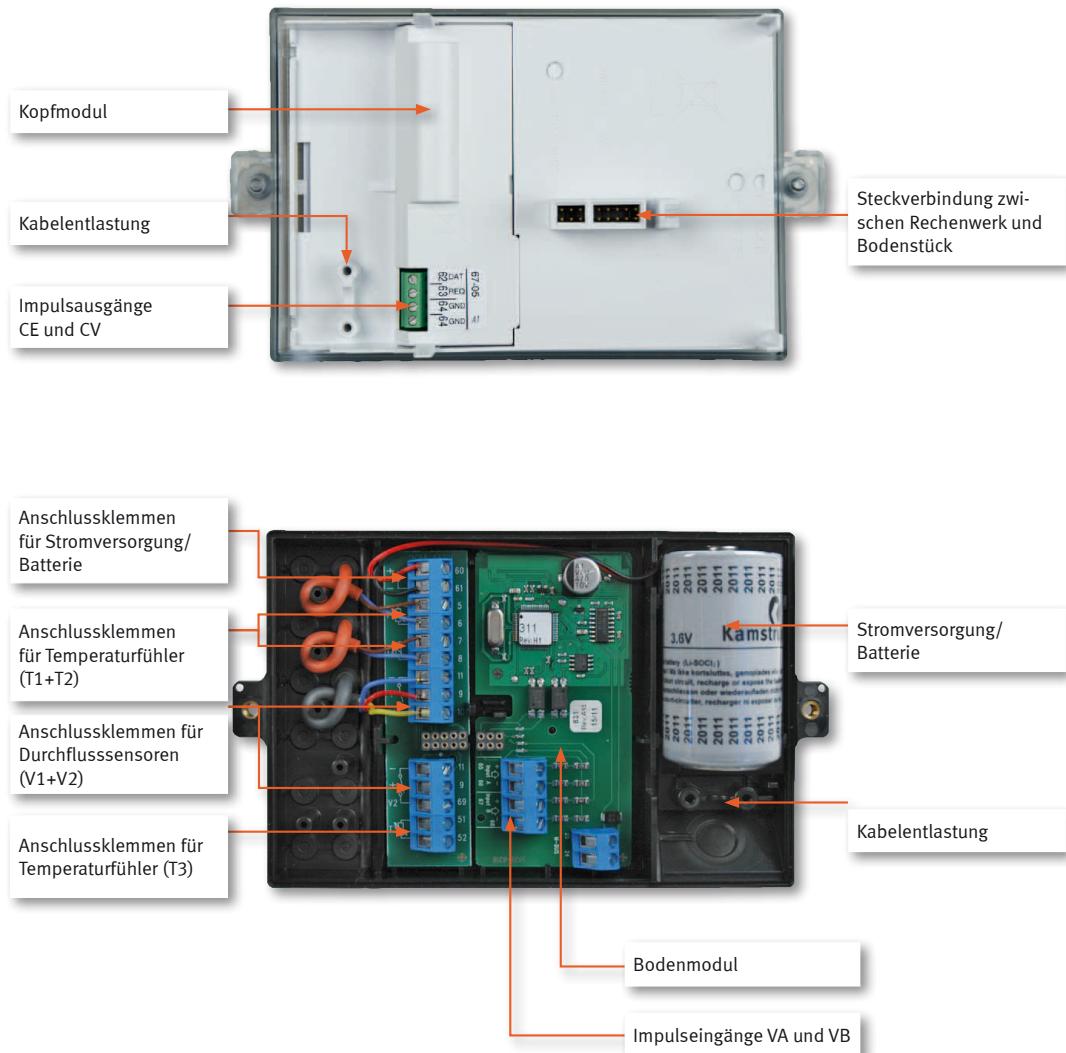
Bei eventuellem Stromausfall werden die Daten gespeichert, und die Abrechnung der Verbrauchsdaten wird somit sichergestellt. Durch die Batterieversorgung des Zählers wird die Lebensdauer auf bis zu 13 Jahre verlängert, einschl. Wireless M-Bus.

Schließlich garantiert MULTICAL® 602 mit ULTRAFLOW® und den genau gepaarten Temperaturfühlern präzise Messergebnisse eben bei minimalen Temperaturunterschieden. Die Langzeitstabilität und Genauigkeit des Durchflusssensors wird nicht von Durchflussgeschwindigkeit, Durchflusstörungen und Verschleiß beeinflusst, was für einen optimalen Betrieb sorgt.

MULTICAL® 602

DATENBLATT

Kabinetteinrichtung



MULTICAL® 602

DATENBLATT

Zugelassene Zählerdaten

Zulassung Norm: prEN 1434:2009 und OIML R75:2002

EU-Direktiven

- MID (Measuring Instruments Directive)
- LVD (Low Voltage Directive)
- EMC (Electromagnetic Compatibility Directive)

Wärmezähler

- Zulassung DK-0200-MI004-020
- Temperaturbereich 0: 2°C...180°C
- Differenzbereich ΔΘ: 3 K...170 K

Die angeführten Mindesttemperaturen sind nur auf die Bauartzulassung bezogen.

Der Zähler hat keine Abschirmung gegen tiefe Temperaturen und misst damit bis zu 0,01°C und 0,01 K.

Kältezähler

- Temperaturbereich 0: 2°C...50°C
- Differenzbereich ΔΘ: 3 K...40 K

Genauigkeit

$$E_c \pm (0,5 + \Delta\Theta_{min}/\Delta\Theta)\%$$

Temperaturfühler

- Typ 602-A Pt100 EN 60 751, Zweileiteranschluss
- Typ 602-B+602-D Pt500 EN 60 751, Vierleiteranschluss
- Typ 602-C Pt500 EN 60 751, Zweileiteranschluss

Durchflusssensortypen

- ULTRAFLOW®
- Elektronische Zähler mit aktivem 24 V Impulsausgang
- Mechanische Zähler mit elektronischer Abtastung
- Mechanische Zähler mit Reed-Schalter

Durchflusssensorgrößen

- [kWh] q_p , 0,6 m³/h... q_p , 15 m³/h
- [MWh] q_p , 0,6 m³/h... q_p , 1500 m³/h
- [GJ] q_p , 0,6 m³/h... q_p , 3000 m³/h

EN 1434 Bezeichnung

Umgebungsklasse A und C

MID Bezeichnung

- Mechanische Umgebung Klasse M1
- Elektromagnetische Umgebung Klasse E1 und E2

Elektrische Daten

Rechenwerksdaten

Typische Genauigkeit

- Rechenwerk $E_c \pm (0,15 + 2/\Delta\Theta)\%$
- Fühlersatz $E_r \pm (0,4 + 4/\Delta\Theta)\%$

Display

LCD – 7 (8) Ziffern mit 7,6 mm Ziffernhöhe

Auflösung

9999,999 – 99999,99 – 999999,9 – 9999999

Energieeinheiten

MWh – kWh – GJ – Gcal

MULTICAL® 602

DATENBLATT

Elektrische Daten

Datenlogger (EEPROM)	
– Standardmässig	1392 Stunden, 460 Tage, 36 Monate, 15 Jahre, 50 Infocodes
– Option	Datenlogger mit programmierbarem Intervall
Uhr/Kalender	Uhr, Kalender, Schaltjahr-Kompensation, Stichtag, Realzeituhr mit Batterie-Backup
Datenkommunikation	KMP Protokoll mit CRC16 wird zur optischen Kommunikation sowie für Kopf- und Bodenmodule verwendet
Leistung von Temperaturfühlern	< 10 µW RMS
Versorgungsspannung	3,6 VDC ± 0,1 VDC
Batterie	3,65 VDC, D-Zelle Lithium
Ruhestrom	< 15 µA ausschl. Durchflusszähler
Austauschintervall	
– Wandmontage	12 + 1 Jahre @ $t_{BAT} < 30^\circ\text{C}$
– Kompaktmontage	10 Jahre @ $t_{BAT} < 40^\circ\text{C}$
	Die Anwendung der Datenmodule, häufige Datenkommunikation und hohe Umgebungstemperatur sind Faktoren, die das Austauschintervall reduzieren werden
Netzversorgung	230 VAC +15/-30%, 50/60 Hz 24 VAC ±50%, 50/60 Hz
Isolationsspannung	4 kV
Leistungsverbrauch	< 1 W
Backup Netzversorgung	Eingebauter SuperCap eliminiert Betriebsstillstand bei kurzfristigem Netzausfall (Nur Versorgungsmodule Typ 602-0000-7 und Typ 602-0000-8)
EMC Daten	Erfüllt prEN 1434-4:2009 Klasse C (MID Klasse E2)
Temperaturmessung	
Fühlereingänge T1, T2, T3	
– Messbereich	0,00...185,00°C
Temperatur T3, T4	
– Voreingestellter Bereich	0,01...180,00°C
Höchstkabellängen	
– Pt100, Zweileiter	2 x 0,25 mm²: 2,5 m 2 x 0,50 mm²: 5 m
– Pt500, Zweileiter	2 x 0,25 mm²: 10 m 2 x 0,50 mm²: 20 m
– Pt500, Vierleiter	4 x 0,25 mm²: 100 m

MULTICAL® 602

DATENBLATT

Elektrische Daten

Durchflussmessung V1 und V2	ULTRAFLOW® V1: 9-10-11 und V2: 9-69-11	Reed-Schalter V1: 10-11 und V2: 69-11	24 V aktive Impulse V1: 10B-11B und V2: 69B-79B
EN 1434 Impulsklasse	IC	IB	(IA)
Impulseingang	680 kΩ Pullup bis zu 3,6 V	680 kΩ Pullup bis zu 3,6 V	12 mA bei 24 V
Impuls EIN	< 0,4 V in > 0,5 mSek.	< 0,4 V in > 100 mSek.	< 4 V in > 3 mSek.
Impuls AUS	> 2,5 V in > 10 mSek.	> 2,5 V in > 100 mSek.	> 12 V in > 10 mSek.
Impulsfrequenz	< 128 Hz	< 1 Hz	< 128 Hz
Integrationsfrequenz	< 1 Hz	< 1 Hz	< 1 Hz
Elektrische Isolation	Nein	Nein	2 kV
Höchstkabellänge	10 m	25 m	100 m

Impulseingänge ohne Prelldämpfung VA und VB VA: 65-66 und VB: 67-68	Wasserzähleranschluss FF(VA) und GG(VB) = 71...90	E-Zähleranschluss FF(VA) und GG(VB) = 50...60
Impulseingang	680 kΩ Pullup bis zu 3,6 V	680 kΩ Pullup bis zu 3,6 V
Impuls EIN	< 0,4 V in > 30 mSek.	< 0,4 V in > 30 mSek.
Impuls AUS	> 2,5 V in > 100 mSek.	> 2,5 V in > 100 mSek.
Impulsfrequenz	< 1 Hz	< 3 Hz
Elektrische Isolation	Nein	Nein
Höchstkabellänge	25 m	25 m
Anforderungen an externen Schalter	Verluststrom bei Funktion offen < 1 µA	

Impulseingänge mit Prelldämpfung VA und VB VA: 65-66 und VB: 67-68	Wasserzähleranschluss FF(VA) und GG(VB) = 01...40
Impulseingang	680 kΩ Pullup bis zu 3,6 V
Impuls EIN	< 0,4 V in > 200 mSek.
Impuls AUS	> 2,5 V in > 500 mSek.
Impulsfrequenz	< 1 Hz
Elektrische Isolation	Nein
Höchstkabellänge	25 m
Anforderungen an externen Schalter	Verluststrom bei Funktion offen < 1 µA

Impulsausgänge CE und CV	Über Kopfmodul 67-OB	Über Kopfmodul 602-OC
Typ	Opto FET	Offener Kollektor (OB)
Impulslänge	32 mSek. oder 100 mSek.	
Externe Spannung	5...48 VDC	5...30 VDC
Strom	1...50 mA	1...10 mA
Restspannung	$R_{ON} \leq 40 \Omega$	$U_{CE} \approx 1 \text{ V bei } 10 \text{ mA}$
Elektrische Isolation	2 kV	2 kV
Höchstkabellänge	25 m	25 m

MULTICAL® 602

DATENBLATT

Mechanische Daten

Umweltklasse	Erfüllt EN 1434 Klasse A und C
Umgebungstemperatur	5...55°C, nicht-kondensierend, geschlossene Position (Inneninstallation)
Schutzart	IP54
Lagertemperatur	-20...60°C (leerer Durchflusszähler)
Gewicht	0,4 kg ausschl. Fühler und Durchflusszähler
Anschlussleitungen	Ø3,5...6 mm
Versorgungsleitung	Ø5...10 mm

Werkstoffbezeichnungen

Oberdeckel	PC
Anschlussbodenstück	ABS mit TPE Dichtungen (thermoplastisches Elastomer)
Platinenkasten	ABS
Wandbeschlag	PC + 30% Glas

C.6 ABNM-LIN Stellantrieb von Danfoss



Datenblatt

Thermischer Stellantrieb ABNM-LOG/LIN für AB-QM, 0-10 Vdc, proportional

Anwendung



Der Antrieb ABNM ist ein thermoelektrischer Stellantrieb zum Öffnen und Schließen von Ventilen im Bereich Heizungs-, Lüftungs- und Klimatechnik (HLK).

Die Regelung erfolgt über ein 0-10 Vdc Signal, das entweder von einem Raumthermostat oder in den meisten Fällen von der zentralen Leittechnik (DDC) bereitgestellt wird. Der Stellantrieb wandelt das 0-10 Vdc Signal in einen proportionalen Stellweg um, der linear oder logarithmisch sein kann.

- Idealer Einsatzbereich sind Heizungs-/Kühlanlagen sowie in Kombination mit der zentralen Leittechnik (DDC) in Gebäudemanagementsystemen (GLT).
- **ABNM LOG** zur Betätigung von Ventilen, die den Durchfluss bei Luft-Wasser-Wärmeüberträgern steuern, z. B. bei Ventilatorkonvektoren oder Klimageräten.
- **ABNM LIN** zur Betätigung von Ventilen, die den Durchfluss bei Wasser-Wasser-Wärmeüberträgern regeln.

Funktion

Der Stellmechanismus des ABNM Stellantriebs arbeitet mit einem PTC-beheizten Wachs-Dehnstoffelement und einer Druckfeder. Das Dehnstoffelement wird erhitzt, indem die Betriebsspannung angelegt wird, und bewegt so den integrierten Kolben. Die durch diese Bewegung erzeugte Kraft wird über den Kolben übertragen und öffnet oder schließt so das Ventil.

Die Schließkraft der Druckfeder (100 N) ist auf die Schließkraft der Ventile abgestimmt und hält das Ventil im stromlosen Zustand geschlossen. Nach Anlegen der Steuerspannung (0-10 Vdc) wird das Dehnstoffelement elektronisch geregelt erwärmt. Aktiv regelt der Stellantrieb in einem festgelegten Bereich (siehe Kennlinie von 0,5 bis 10 Vdc).

Zwischen 0 und 0,5 Vdc ist der Stellantrieb im Ruhezustand. Dadurch werden eventuell auftretende Brummspannungen auf langen Leitungen im unteren Steuerspannungsbereich ignoriert. Das Verhältnis zwischen Steuerspannung und Antriebsbewegung wird durch optische Hubmessung angepasst, wodurch eine sehr genaue Positionierung möglich ist. Wenn die Steuerspannung außerhalb des aktiven Bereichs liegt, wird das Ventil durch die Schließkraft der Druckfeder geschlossen gehalten.

First-Open-Funktion (nur bei stromlos geschlossener Ausführung)

Im Auslieferungszustand wird der ABNM im stromlosen Zustand durch die First-Open-Funktion geöffnet gehalten. Dies ermöglicht den Heiz-/Kühlwasser-Durchfluss durch das Ventil während der Bauphase, auch wenn die elektrische Installation noch nicht abgeschlossen ist. Bei der späteren elektrischen Inbetriebnahme wird die First-Open-Funktion durch Anlegen der Betriebsspannung (mindestens 6 Minuten) außer Kraft gesetzt und der Stellantrieb ABNM ist dann voll funktionsstüchtig.

Automatische Kalibrierung

Während der elektrischen Inbetriebnahme wird der Schließpunkt des Ventils erfasst. Dies gewährleistet eine optimale Anpassung auf das jeweils verwendete Ventil.

Funktionsanzeige

Durch die rundum erkennbare Funktionsanzeige am ABNM kann auf den ersten Blick festgestellt werden, ob das Ventil sich im geöffneten oder im geschlossenen Zustand befindet.

**Datenblatt****Thermischer Stellantrieb ABNM-LOG/LIN für AB-QM, 0-10 Vdc, proportional****Artikelnummern und
Technische Daten**

Typ	Versorgungs- spannung	Steuerspan- nung	Ventilfunktion	Kabellän- ge	Artikel-Nr.
ABNM LOG mit Adapter VA50	24 V AC	0-10 V DC	NC (stromlos geschlossen)	1 m	082F1191
ABNM LOG mit Adapter VA50	24 V AC	0-10 V DC	NC (stromlos geschlossen)	5 m	082F1192
ABNM LIN mit Adapter VA50	24 V AC	0-10 V DC	NC (stromlos geschlossen)	1 m	082F1193
ABNM LOG ohne Adapter	24 V AC	0-10 V DC	NC (stromlos geschlossen)	Nein	082F1198
ABNM LIN ohne Adapter	24 V AC	0-10 V DC	NC (stromlos geschlossen)	Nein	082F1199

Hinweis: Diebstahlsicherung auf Anfrage

Zubehör**Ventiladapter**

Anschluss	Artikel-Nr.
VA50 für Danfoss AB-QM	082F1075

Kabel (halogenfrei)

Länge	Artikel-Nr.
1 Meter	082F1081
5 Meter	082F1082
10 Meter	082F1083

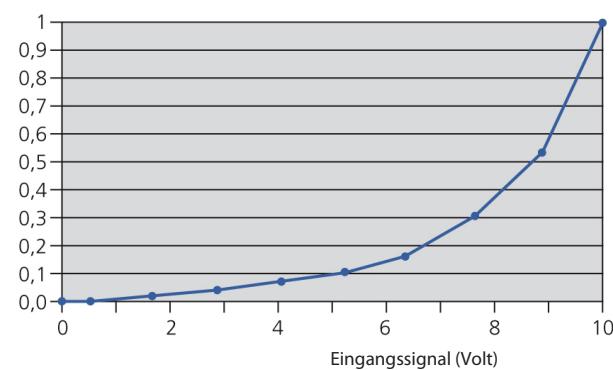
Daten

Version	Stromlos geschlossen
Spannung	24 VAC 50/60 Hz (-10 % bis +20 %)
Max. Einschaltstrom	<300 mA, Dauer ca. 2 Min.
Betriebsstrom	90 mA
Betriebsleistung	0,4 W
Steuerspannung	0-10 V DC
Proportionaler Umwandlungsbereich der Steuerspannung	0,5-10 V DC
Eingangswiderstand	100 kΩ (10 kΩ optional auf Anfrage)
Stellweg	4,5 mm (minus Überhub); max. 4 mm
Mittlere Stellzeit	30 s/mm
Stellkraft	100 N +/- 5 %
Betriebstemperatur	0-60 °C
Medientemperatur	0-100 °C
Lagertemperatur	-25 bis 65 °C
Umgebungstemperatur	0 bis 60 °C
Relative Luftfeuchtigkeit	max. 80 %
Schutzart/Schutzklaasse	IP54/Schutzkleinspannung
CE-Konformität	60730
Gehäuse/Gehäusefarbe	Polyamid/Weiß RAL 9003
Gewicht	100 g ohne Adapter und Kabel
Anschlusskabel/Kabellänge	3 x 0,22 mm ² , Weiß/1 Meter/30 g

Kennlinien

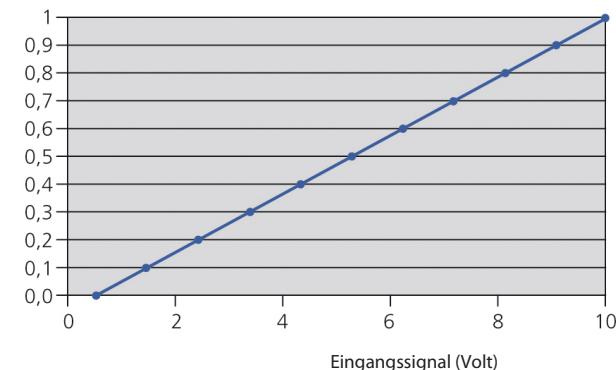
ABNM-LOG, Transformationskurve

Relativer Hub



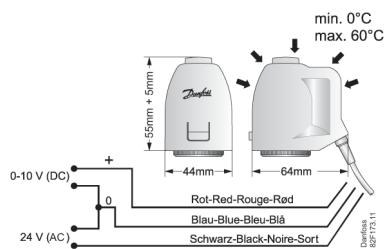
ABNM-LIN, Transformationskurve

Relativer Hub



Stellantrieb ABNM

Der Antrieb wandelt die 0-10 V Steuerspannung in einen proportionalen Stellweg von 0-4,5 mm um.

Abmessungen, Elektrischer Anschluss

Transformator

Vereinfachte Berechnungsformel zur Dimensionierung des Transformators:

$$P_{\text{Transformator}} = 6 \text{ W} \times \text{Anzahl ABNM-Antriebe}$$

Berechnung der max. Kabellänge (Kupferkabel)

$$L = K \times A / n$$

A: Leiterquerschnitt in mm²

n: Anzahl ABNM-Antriebe

K: Konstante für Kupfer (269 m/mm²)

L: Kabellänge in m

C.7 UNITRONIC Busleitung von LappKabel



DATENBLATT		2170203
UNITRONIC® BUS LD ... x 2 x 0,22 mm²		gültig ab : 27.10.2009

Verwendung

UNITRONIC® BUS LD ist eine kapazitätsarme Datenleitung für Feldbussysteme mit Übertragungsraten bis 10 MBit/s. Die Leitungskreise der Feldbusleitung sind gut entkoppelt und weisen günstige Nebensprechdämpfungswerte auf. Die Paarverteilung und der Schirm aus dem engmaschigen Kupferdrahtgeflecht bieten einen wirkungsvollen Schutz gegen äußere elektromagnetische Störbeeinflussungen. Die Leitung ist für bedingt flexiblen Einsatz und zur festen Verlegung in trockenen und feuchten Räumen ausgelegt.

Verwendete Steckverbinder: D-Sub-Stecker, 9-polig; Rundsteckverbinder, 9-polig (Schutzart IP 65)

Aufbau

Leiter	Mehrdrähtige Litze aus blanken Kupferdrähten
Isolierhülle	Kunststoffmischung auf PE-Basis
Aderfarben	nach DIN 47100
Verseilung	Paarverseilung, Paare gemeinsam verseilt
Bewicklung	Kunststofffolie
Abschirmung	Geflecht aus verzинnten Kupferdrähten
Mantel	Kunststoffmischung auf PVC-Basis
Mantel Farbe	violett, ähnlich RAL 4001
Außendurchmesser 1paarig:	ca. 5,7 mm (Art. Nr. 2170203)
Außendurchmesser 2paarig:	ca. 7,1 mm (Art. Nr. 2170204)
Außendurchmesser 3paarig:	ca. 7,2 mm (Art. Nr. 2170205)

Elektrische Eigenschaften bei 20°C

Leiterwiderstand (Schleife)	max. Ω/km	186
Isolationswiderstand	min. GΩ x km	5
Betriebskapazität bei 800 Hz	max. nF/km	60
Wellenwiderstand	Ω	100 - 120
Leitungsdämpfung bei 100 kHz	nom. dB/100m	0,9
Leitungsdämpfung bei 1 MHz	nom. dB/100m	2,5
Nahnebensprechdämpfung bei 1 MHz	min. dB	50
Nahnebensprechdämpfung bei 10 MHz	min. dB	40
Kopplungswiderstand bei 30 MHz	max. mΩ/m	250
Signalausbreitungsgeschwindigkeit	nom.	0,66 c
Signallaufzeit	nom. ns/m	5,06
Betriebsspitzenspannung (nicht für Starkstromzwecke)	V	250
Prüfspannung (Ader/Ader)	V	1500
Prüfspannung (Ader/Schirm)	V	1000

Mechanische und thermische Eigenschaften

Temperaturbereich festverlegt	°C	-40 bis +80
Temperaturbereich bewegt	°C	-5 bis +70
Mindestbiegeradius festverlegt	LeitungsØ x	8
Mindestbiegeradius bewegt	LeitungsØ x	20
Brennverhalten	flammwidrig nach IEC 60 332-1-2	

Konformität

Die Leitungen sind konform zur RoHSRichtlinie (2002/95/EG)

ausgearbeitet von: Petra Samek, PDC	Dokument: DB2170203DE04	Blatt 1 von 1
--	----------------------------	---------------

D Modbusadressmapping

D.1 EX9024M Signalwandler von ExpertDAQ

9017-M				
Address	Hex	Channel	Content	Attribute
30001	0H	0	Analog input Value	Read
30002	1H	1	Analog input Value	Read
30003	2H	2	Analog input Value	Read
30004	3H	3	Analog input Value	Read
30005	4H	4	Analog input Value	Read
30006	5H	5	Analog input Value	Read
30007	6H	6	Analog input Value	Read
30008	7H	7	Analog input Value	Read

9024-M				
Address	Hex	Channel	Content	Attribute
40001	0H	0	Analog output value	Read/Write
40002	1H	1	Analog output value	Read/Write
40003	2H	2	Analog output value	Read/Write
40004	3H	3	Analog output value	Read/Write

9033-M				
Address	Hex	Channel	Content	Attribute
30001	0H	0	Analog input Value	Read
30002	1H	1	Analog input Value	Read
30003	2H	2	Analog input Value	Read

9036/15-M				
Address	Hex	Channel	Content	Attribute
30001	0H	0	Analog input Value	Read
30002	1H	1	Analog input Value	Read
30003	2H	2	Analog input Value	Read
30004	3H	3	Analog input Value	Read
30005	4H	4	Analog input Value	Read
30006	5H	5	Analog input Value	Read

9000-M DIO Series				
Address	Hex	Channel	Content	Attribute
00001~00032	0H~1FH	0~31	Digital Output	Read/Write
00033~00048	0H~1FH	0~31	Digital Input	Read

D.2 THERMASGARD RTM1-Modbus

Raumtemperaturfühler von S+S Regeltechnik

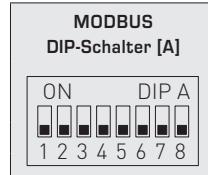
(D) THERMASGARD® RTM 1-Modbus | Konfiguration

Rev. 2016-V12 DE intro

BUSADRESSE

Busadresse (binärcodiert, Wertigkeit 1 bis 247 einstellbar)							
DIP 1	DIP 2	DIP 3	DIP 4	DIP 5	DIP 6	DIP 7	DIP 8
128	64	32	16	8	4	2	1
ON	ON	OFF	OFF	OFF	OFF	OFF	ON

Beispiel zeigt $128 + 64 + 1 = 193$ als Modbus-Adresse.



Die **Geräteadresse** im Bereich von **1 bis 247** (Binärformat) wird über den DIP-Schalter [A] eingestellt. Schalterstellung Pos. 1 bis 8 – siehe Tabelle auf Rückseite!

Die Adresse 0 ist für Broadcast-Meldungen reserviert, die Adressen größer 247 dürfen nicht belegt werden und werden vom Gerät ignoriert. Die DIP-Schalter sind binärcodiert mit folgender Wertigkeit:

DIP 1 = **128** DIP 1 = **ON**

DIP 2 = **64** DIP 2 = **ON**

DIP 3 = **32** DIP 3 = **OFF**

DIP 4 = **16** DIP 4 = **OFF**

DIP 5 = **8** DIP 5 = **OFF**

DIP 6 = **4** DIP 6 = **OFF**

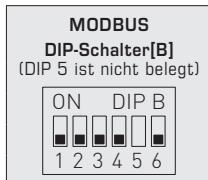
DIP 7 = **2** DIP 7 = **OFF**

DIP 8 = **1** DIP 8 = **ON**

folgt die Modbus-Adresse **128 + 64 + 1 = 193**

BUSPARAMETER

Baudrate (einstellbar)	DIP 1	DIP 2
9600 Baud	ON	OFF
19200 Baud	ON	ON
38400 Baud	OFF	ON
reserviert	OFF	OFF



Parity (einstellbar)	DIP 3
EVEN (gerade)	ON
ODD (ungerade)	OFF

Parity-Sicherung (mit/ohne einstellbar)	DIP 4
aktiv (1 Stopbit)	ON
inaktiv (2 Stopbits)	OFF

Busabschluss (mit/ohne einstellbar)	DIP 6
aktiv	ON
inaktiv	OFF

Die **Baudrate** (Übertragungsgeschwindigkeit) wird über Pos. 1 und 2 des DIP-Schalters [B] eingestellt. Einstellbar sind **9600 Baud**, **19200 Baud** oder **38400 Baud** – siehe Tabelle!

Die **Parity** wird über Pos. 3 des DIP-Schalters [B] eingestellt.

Einstellbar sind **EVEN (gerade)** oder **ODD (ungerade)** – siehe Tabelle!

Die **Parity-Sicherung** wird über Pos. 4 des DIP-Schalters [B] aktiviert.

Einstellbar ist Parity-Sicherung **aktiv (1 Stopbit)** oder **inaktiv (2 Stopbits)**, d.h. keine Parity-Sicherung – siehe Tabelle!

Pos. 5 des DIP-Schalters [B] ist nicht belegt!

Der **Busabschluss** wird über Pos. 6 des DIP-Schalters [B] aktiviert.

Einstellbar ist **aktiv** (Busabschlusswiderstand von 120 Ohm) oder **inaktiv** (ohne Busabschluss) – siehe Tabelle!

Bei Änderung der Busparameter und Busadresse werden bei Geräten mit **Displayanzeige** die entsprechenden Einstellungen im Display für ca. 30 Sekunden angezeigt.

KOMMUNIKATIONSANZEIGE

Die Kommunikation wird über 2 LED-Anzeigen signalisiert. Fehlerfrei empfangene Telegramme werden unabhängig von der Geräteadresse durch Aufleuchten der grünen Anzeige signalisiert. Fehlerhafte Telegramme oder ausgelöste Modbus Exception-Telegramme werden durch das Aufleuchten der roten Anzeige dargestellt.

DIAGNOSE

Fehlerdiagnosefunktion mitintegriert

 **THERMASGARD® RTM 1-Modbus** | Telegramme

Function 04 Read Input Register

Register	Parameter		Data Type	Value	Range
3x0001	Temperatur	4 s Abtastung	Signed 16 Bit	-350...+800	-35.0 ... +80.0 °C
3x0002	Temperatur	Filterung	Signed 16 Bit	-350...+800	-35.0 ... +80.0 °C
3x0003	Sollwert Potentiometer		Signed 16 Bit	0...1000	0...100.0 %
3x0004	Taupunkt	Berechneter Wert	Signed 16 Bit	0... 500	0...+50.0 °C
3x0005	Relative Feuchte	Filterung	Signed 16 Bit	0...1000	0...100.0 % r.H.

Function 08 Diagnostics

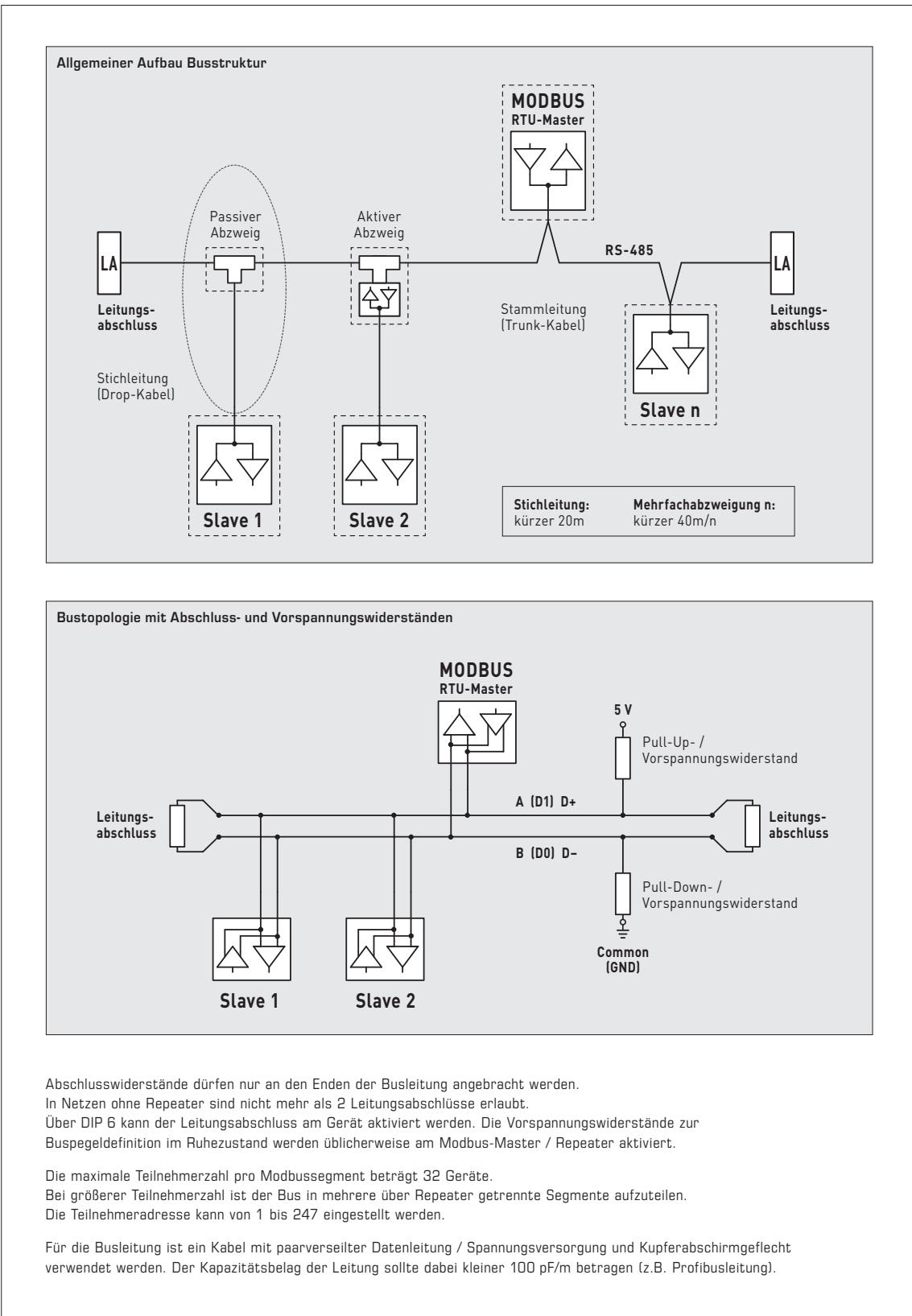
Folgende Sub Function Codes werden unterstützt

Sub Function Code	Parameter	Data Type	Antwort
00	Echo der Sendedaten (Loopback)		Echodata
01	Neustart Modbus (Reset Listen Only Mode)		Echo Telegramm
04	Aktivierung Listen Only Mode		Keine Antwort
10	Lösche Zähler		Echo Telegramm
11	Zähler Bustelegramme	Unsigned 16 Bit	alle gültigen Bustelegramme
12	Zähler Kommunikationsfehler (Parity, CRC, Framefehler, etc.)	Unsigned 16 Bit	fehlerhafte Bustelegramme
13	Zähler Exception-Meldungen	Unsigned 16 Bit	Fehlerzähler
14	Zähler Slave-Telegramme	Unsigned 16 Bit	Slave-Telegramme
15	Zähler Telegramme ohne Antwort	Unsigned 16 Bit	Broadcastmeldungen (Adresse 0)

Function 17 Report Slave ID

Aufbau Antworttelegramm

Byte Nr.	Parameter	Data Type	Antwort
00	Byteanzahl	Unsigned 8 Bit	6
01	Slave ID (Device Typ)	Unsigned 8 Bit	4 = THERMASGARD® MODBUS
02	Slave ID (Device Class)	Unsigned 8 Bit	10 = THERMASGARD® / THERMASREG®
03	Status	Unsigned 8 Bit	255 = RUN, 0 = STOP
04	Versionsnummer (Release)	Unsigned 8 Bit	1...9
05	Versionsnummer (Version)	Unsigned 8 Bit	1...99
06	Versionsnummer (Index)	Unsigned 8 Bit	1



D.3 MULTICAL 602 Wärmemengenzähler von Kamstrup

Datenblatt

Modbus RTU Slave-Modul

**Modbus-Kommunikationsmodul für
MULTICAL® 62/601/602/6L2/6M2/801**

- RTU-Kommunikation basiert auf RS-485
- Übertragungsgeschwindigkeit von bis zu 76.800 Bits/Sek.
- Einstellung von Programmierbaren Daten, der Kommunikationsgeschwindigkeit und der Parität
- Zwei Impulseingänge für zusätzliche Wasser- und Stromzähler
- RS-485 galvanisch getrennt vom Zähler



Beschreibung

Anwendung

Modbus ist ein offenes, weit verbreitetes und gut etabliertes, serielles Kommunikationsprotokoll, das im Rahmen der Gebäudeautomatisierung verwendet wird.

Das Modbus-Bodenmodul für MULTICAL® sichert eine einfache Integration von Wärme-, Kälte- und Wasserzählern mit Modbus-basierten Systemen.

Das Modbus-Modul wird in MULTICAL® installiert und zur Datenübertragung aus MULTICAL® Wärme-, Kälte- und Wasserzählern auf ein Modbus-System verwendet.

Funktionalität

Das Modbus-Modul kommuniziert als ein RTU*-Slave-Gerät auf RS-485.

Das Modbus-Modul überträgt eine Reihe von sowohl aktuellen als auch akkumulierten Daten.

Darüber hinaus können MULTICAL®-Infocodes für allgemeine Alarne, Durchflussfehler, Temperaturfehler, Wasserlecks, Rohrbrüche, Luft im System und falsche Durchflussrichtung auf das Modbus-System übertragen werden.

Die beiden Impulseingänge ermöglichen den Anschluss und die Auslesung von zwei zusätzlichen Zählern für z.B. Wasser und Strom mit Impulsausgang.

Betriebszuverlässigkeit

Der RS-485-Port des Modbus-Moduls ist vom Spannungspotenzial des Zählers galvanisch getrennt, was einen reibungslosen Betrieb sichert. Gleichzeitig ist die Gefahr einer Beeinflussung des Zählers durch Auswirkungen des RS-485-Ports auf ein Minimum reduziert.

Das Modbus-Modul ist in voller Übereinstimmung mit und in der MID-Zulassung für MULTICAL® enthalten.

Adressenbereich

Das Modul kann als einen Slave im Bereich 1-247 adressiert werden.

Als Standard entspricht die Modbus-Adresse den letzten drei Ziffern der Kundennummer.

Wenn die Kundennummer des Zählers eine Adresse ergibt, die größer als 247 ist, werden nur die beiden letzten Ziffern für die Modbus-Adresse des Moduls verwendet.

Zur Beachtung: Wenn die Kundennummer des Zählers mit 000 endet, wechselt das Modbus-Modul automatisch auf die Adresse 247.

*]) RTU: Remote Terminal Unit

Technische Daten

Modbus-Funktionen

- Kommuniziert mit dem Bus mittels RS-485 [Standard 19200, 8, E, 1]
- Unterstützte Baudaten: 300, 2400, 9600, 19200, 38400, 76800
- Unterstützte Paritätseinstellungen: keine, gerade und ungerade
- Unterstützte Stoppbit-Einstellungen: ein und zwei
- Unterstützte Datenbits: 8
- Das Modul kann als einen Slave im Bereich 1-247 adressiert werden.
- Dateninhalt, Baudrate, Parität und Adresse werden in MULTICAL® gespeichert und können über einen optischen Lesekopf und das PC-Programm MULTICAL® Module Programmer geändert werden.
- Unterstützt RTU-Übertragungsmodus
- Unterstützte Funktionscodes und ihre möglichen Ausnahmecodes:
 - 0x03 Read Holding Registers with exception codes:
 - 0x02 – Illegal data address
 - 0x03 – Illegal data value
 - 0x04 Read Input Registers with exception codes:
 - 0x02 – Illegal data address
 - 0x03 – Illegal data value
 - 0x08 Diagnostics with exception code:
 - 0x01 – Illegal function
 - Subcode 0x01 Restart with exception code:
 - 0x03 – Illegal data value
 - 0x2B Encapsulated interface transport with exception code:
 - 0x01 – Illegal function
 - Subcode 0x0E Read Device Identification with exception code:
 - 0x03 – Illegal data value
 - 0x41 und 0x42 Reserviert als spezifischer Funktionscode.

Modbus-Datenmodellmapping von MULTICAL®-Werten

Der PDU-Adressenbereich von 0 bis 168 wird nachfolgend näher beschrieben. Die PDU-Adresse ist in Bytes angegeben, sodass ein 32-Bit Register ergeben wird, dass das folgende Register auf einer Adresse 4 Stellen höher platziert werden soll. Die Daten sind in 6 verschiedene Tabellen gegliedert. Die Daten in den jeweiligen Tabellen sind alle von der gleichen Größe und sollten in gleicher Weise ausgelegt werden. Eine genauere Beschreibung der einzelnen Spalten werden später gegeben.

Datenmodellmapping für Byte-adressierten Bereich, Standard-Datagramm

Memory	Memory (hex)	Individual description	Size in bytes	Table	Contents	Data type	Update status
0	0x0000	Heat energy E1	4	1	Values in float	IEEE Float - 32 bit	Dynamic
4	0x0004	Actual flow	4	1	Values in float	IEEE Float - 32 bit	Dynamic
8	0x0008	Volume V1	4	1	Values in float	IEEE Float - 32 bit	Dynamic
12	0x000C	Actual power	4	1	Values in float	IEEE Float - 32 bit	Dynamic
16	0x0010	Inlet temperature T1	4	1	Values in float	IEEE Float - 32 bit	Dynamic
20	0x0014	Outlet temperature T2	4	1	Values in float	IEEE Float - 32 bit	Dynamic
24	0x0018	Pulse input A	4	1	Values in float	IEEE Float - 32 bit	Dynamic
28	0x001C	Pulse input B	4	1	Values in float	IEEE Float - 32 bit	Dynamic
32	0x0020	Heat energy E1	2	2	Units	Word - 16 bit	Dynamic
34	0x0022	Actual flow	2	2	Units	Word - 16 bit	Dynamic
36	0x0024	Volume V1	2	2	Units	Word - 16 bit	Dynamic
38	0x0026	Actual power	2	2	Units	Word - 16 bit	Dynamic
40	0x0028	Heat energy E1	4	3	Values in integer	Double Word - 32 bit	Dynamic
44	0x002C	Actual flow	4	3	Values in integer	Double Word - 32 bit	Dynamic
48	0x0030	Volume V1	4	3	Values in integer	Double Word - 32 bit	Dynamic
52	0x0034	Actual power	4	3	Values in integer	Double Word - 32 bit	Dynamic
56	0x0038	Inlet temperature T1	4	3	Values in integer	Signed Double Word - 32 bit	Dynamic
60	0x003C	Outlet temperature T2	4	3	Values in integer	Signed Double Word - 32 bit	Dynamic
64	0x0040	Pulse input A	4	3	Values in integer	Double Word - 32 bit	Dynamic
68	0x0044	Pulse input B	4	3	Values in integer	Double Word - 32 bit	Dynamic
72	0x0048	Heat energy E1	2	4	Decimal	Word - 16 bit	Dynamic
74	0x004A	Actual flow	2	4	Decimal	Word - 16 bit	Dynamic
76	0x004C	Volume V1	2	4	Decimal	Word - 16 bit	Dynamic
78	0x004E	Actual power	2	4	Decimal	Word - 16 bit	Dynamic
80	0x0050	Pulse input A	2	4	Decimal	Word - 16 bit	Dynamic
82	0x0052	Pulse input B	2	4	Decimal	Word - 16 bit	Dynamic
84	0x0054	Version	2	5	Program version	Word - 16 bit	Static
86	0x0056	Info code	2	6	Info code	Word - 16 bit	Dynamic
88	0x0058	Reserved	4	N/A		IEEE Float - 32 bit	
92	0x005C	Cooling energy E3	4	8	Values in float	IEEE Float - 32 bit	Dynamic
96	0x0060	Volume - V2	4	8	Values in float	IEEE Float - 32 bit	Dynamic
100	0x0064	Temperature T3	4	8	Values in float	IEEE Float - 32 bit	Dynamic
104	0x0068	Cooling energy E3	2	9	Units	Word - 16 bit	Dynamic

Modbus-Datenmodellmapping von MULTICAL®-Werten

Memory	Memory (hex)	Individual description	Size in bytes	Table	Contents	Data type	Update status
106	0x006A	Volume - V2	2	9	Units	Word - 16 bit	Dynamic
108	0x006C	Cooling energy E3	4	10	Values in integer	Double Word - 32 bit	Dynamic
112	0x0070	Volume - V2	4	10	Values in integer	Double Word - 32 bit	Dynamic
116	0x0074	Temperature T3	4	10	Values in integer	Signed Double Word - 32 bit	Dynamic
120	0x0078	Cooling energy E3	2	11	Decimal	Word - 16 bit	Dynamic
122	0x007A	Volume - V2	2	11	Decimal	Word - 16 bit	Dynamic
124	0x007C	Max power	4	12	Values in float	IEEE Float - 32 bit	Dynamic
128	0x0080	Tarif 2	4	12	Values in float	IEEE Float - 32 bit	Dynamic
132	0x0084	Tarif 3	4	12	Values in float	IEEE Float - 32 bit	Dynamic
136	0x0088	Tarif limit 2	4	12	Values in float	IEEE Float - 32 bit	Static
140	0x008C	Tarif limit 3	4	12	Values in float	IEEE Float - 32 bit	Static
144	0x0090	Meter type	4	13	Parameters	Double Word - 32 bit	Static
148	0x0094	Meter number 1	4	13	Parameters	Double Word - 32 bit	Static
152	0x0098	Serial number	4	13	Parameters	Double Word - 32 bit	Static
156	0x009C	Program number	4	13	Parameters	Double Word - 32 bit	Static
160	0x00A0	Config number 1	4	13	Parameters	Double Word - 32 bit	Static
164	0x00A4	Config Number 2	4	13	Parameters	Double Word - 32 bit	Static
168	0x00A8	Hour counter	4	13	Parameters	Double Word - 32 bit	Dynamic

Inhalt

Beschreibt welche Arten von Daten, die im spezifischen Register gespeichert sind.
Alle Register in einer Tabelle haben den gleichen Inhalt.

- Werte im Float-Format
 - Daten sollten als ein 32-Bit IEEE-Float-Format ausgelegt werden.
- Werte im integer
 - Daten sollten als eine 32-Bit vorzeichenlose Ganzzahl ausgelegt werden.
- Einheiten
 - Daten sind in einem 16-Bit vorzeichenlosen Wort gespeichert und sollten auf folgender Weise ausgelegt werden:

Dezimale	Hexadezimal	Einheit
1	0x0001	kW
2	0x0002	MW
17	0x0011	kWh
18	0x0012	MWh
33	0x0021	I
34	0x0022	m³
35	0x0023	m³x10
49	0x0031	I/h
50	0x0032	m³/h
65	0x0041	Tonne

- Dezimale
 - Daten sind in einem 16-Bit vorzeichenlosen Wort gespeichert und geben die Anzahl der Dezimalen an.
- Parameter
 - Daten sind in einem 32-Bit vorzeichenlosen Doppelwort gespeichert.
- Infocode
 - Daten sind in einem 16-Bit vorzeichenlosen Wort gespeichert und geben den MULTICAL®-Infocode an.
- Programm-Version
 - Daten sind in einem 16-Bit vorzeichenlosen Wort gespeichert und repräsentieren die offizielle Programm-Version im Modul.

Update status (Aktualisierungsstatus)

Dynamic gibt an, dass der Speicherplatz laufend aktualisiert wird (alle 30 Sekunden für MULTICAL® 601 und alle 10 Sekunden für andere Zähler), während static bedeutet, dass der Speicherplatz nur einmal aktualisiert wird.

Literaturverzeichnis

- [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/Bismarckstraße+12,+76133+Karlsruhe/@49.01312,8.3918928,19z/data=!3m1!4b1!4m2!3m1!1s0x47970656cae24c5:0x630ef97b3685becd?hl=de>, 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
- [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
- [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
- [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 31. März 2016

Daniel Johannes Mayer