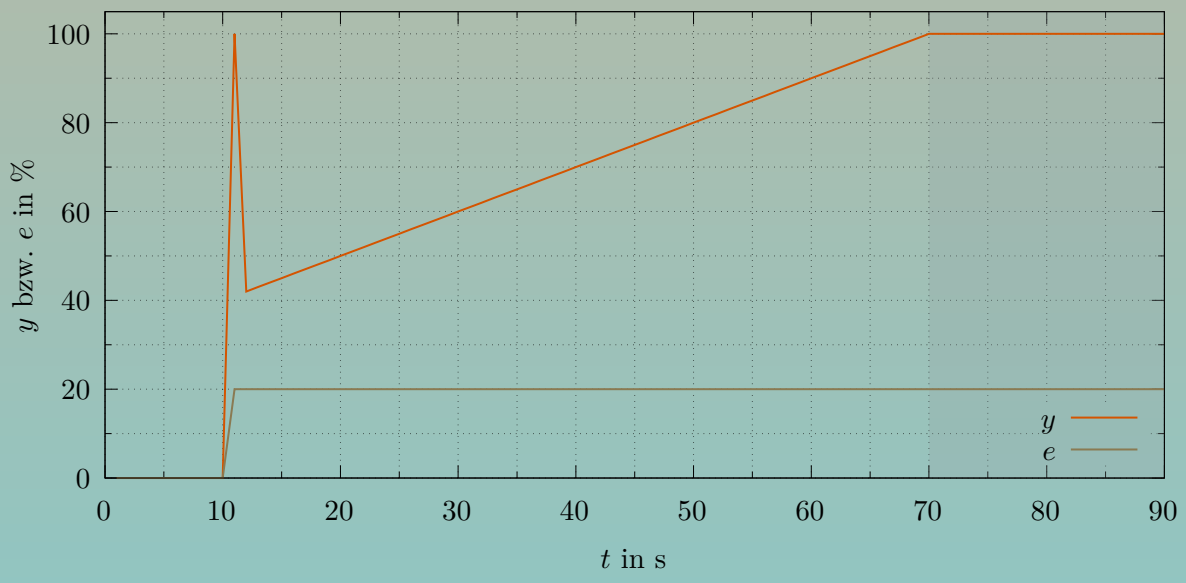
Praktische Regelungstechnik mit Arduino

am Beispiel einer Temperaturregelstrecke

mit Anregungen für einen handlungsorientierten Technikunterricht

StR Dipl.-Phys. Volker Neises

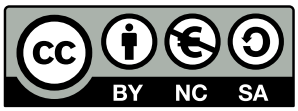
```
38 void loop() {
39   if (millis() / 1000.0 - t >= dt) {
40     tMem = t;
41     t = millis() / 1000.0;
42
43     sensors.requestTemperatures();
44     x = sensors.getTempCByIndex(0);
45
46     eMem = e;
47     e = 100.0 * (w - x) / R;
48
49     yp = Kp * e;
50     yd = Kd * (e - eMem) / (t - tMem);
51     dyi = Ki * e * (t - tMem);
52     yi += dyi;
```



Volker Neises
Kerschensteinerschule Wiesbaden
Welfenstraße 10
65189 Wiesbaden
volker.neises@schule.hessen.de

Version vom 12. Dezember 2023

Dieses Werk ist lizenziert unter einer Creative Commons „Namensnennung – Nicht-kommerziell – Weitergabe unter gleichen Bedingungen 4.0 International“ Lizenz.



Sie dürfen:

Teilen – das Material in jedwedem Format oder Medium vervielfältigen und weiterverbreiten
Bearbeiten – das Material remixen, verändern und darauf aufbauen
Der Lizenzgeber kann diese Freiheiten nicht widerrufen, solange Sie sich an die Lizenzbedingungen halten.

Unter folgenden Bedingungen:

Namensnennung – Sie müssen angemessene Urheber- und Rechteangaben machen, einen Link zur Lizenz beifügen und angeben, ob Änderungen vorgenommen wurden. Diese Angaben dürfen in jeder angemessenen Art und Weise gemacht werden, allerdings nicht so, dass der Eindruck entsteht, der Lizenzgeber unterstütze gerade Sie oder Ihre Nutzung besonders.
Nicht kommerziell – Sie dürfen das Material nicht für kommerzielle Zwecke nutzen.
Weitergabe unter gleichen Bedingungen – Wenn Sie das Material remixen, verändern oder anderweitig direkt darauf aufbauen, dürfen Sie Ihre Beiträge nur unter derselben Lizenz wie das Original verbreiten.
Keine weiteren Einschränkungen – Sie dürfen keine zusätzlichen Klauseln oder technische Verfahren einsetzen, die Anderen rechtlich irgendetwas untersagen, was die Lizenz erlaubt.

Hinweise:

Sie müssen sich nicht an diese Lizenz halten hinsichtlich solcher Teile des Materials, die gemeinfrei sind, oder soweit Ihre Nutzungshandlungen durch Ausnahmen und Schranken des Urheberrechts gedeckt sind.
Es werden keine Garantien gegeben und auch keine Gewähr geleistet. Die Lizenz verschafft Ihnen möglicherweise nicht alle Erlaubnisse, die Sie für die jeweilige Nutzung brauchen. Es können beispielsweise andere Rechte wie Persönlichkeits- und Datenschutzrechte zu beachten sein, die Ihre Nutzung des Materials entsprechend beschränken.

Inhaltsverzeichnis

1. Vorbemerkung und Motivation

Ein Ziel dieser Arbeit ist es zu zeigen, dass Regelungstechnik keine „*Raketenwissenschaft*“ ist. Dies wird in erster Linie am Beispiel eines realen Temperaturregelkreises diskutiert. Auch wenn in Vorlesungen zu Ingenieursstudiengängen und auf der entsprechenden Wikipedia-Seite[1] mit *Differentialgleichungen*, *inversen Laplace-Transformationen*, *Übertragungsfunktionen*, ... gearbeitet wird, soll hier dargelegt werden, dass z. B. auch stetige Proportional-Integral-Differential-Regler (PID) auf Basis der Programmierung eines Arduino-Mikrocontrollers[2] unter ausschließlicher Verwendung der Grundrechenarten (+, −, ·, /) realisiert werden können. Anstatt Stabilitätskriterien mathematisch zu beurteilen, liegt der Fokus dieser Arbeit auf der Untersuchung von realen Regelstrecken. Anhand der Beobachtung von Regel- und Stellgrößenverläufen sowie Veränderungen an der Regelstrecke und an Regelparametern soll ein intuitives Verständnis für die Dynamik von Regelprozessen gewonnen werden. Hierzu wird insbesondere auf das Erreichen von Gleichgewichtszuständen eingegangen. Dabei werden auch Analogien zu weiteren (mechanischen) Modellen gesucht.

Die Arbeit richtet sich in erster Linie zunächst an Lehrende und soll Anregungen dazu geben, wie durch geeignete didaktische Reduktionen und mit verhältnismäßig einfachen Mitteln Regelungstechnik im Unterricht handlungsorientiert umgesetzt werden kann. Da Lernsituationen allerdings immer auf die konkrete Lerngruppe und u. a. auch auf den zur Verfügung stehenden Zeitrahmen abgestimmt sein müssen, werden Formulierungen dabei recht allgemein gehalten. Es wird immer nur grob angerissen, wie die Phasen einer vollständigen Handlung abgebildet werden könnten. Dies soll zumindest Anregungen für eine eigene Ausgestaltung geben. Ergänzend werden auch Ideen für Aufgabenstellungen formuliert. Es ist nicht geplant, die Regelungstechnik vollständig bis ins letzte Detail zu behandeln. Theoretische Grundlagen und Fachbegriffe werden nur soweit erklärt, dass die verwendeten Beispiele nachvollzogen werden können. Detaillierte Programmierkenntnisse werden nicht vorausgesetzt. Die einzelnen Codebeispiele werden ausführlich besprochen und schrittweise ergänzt[3]. Dabei können ggf. auch grundlegende Programmiertechniken, z. B. das Deklarieren von Variablen sowie Wertzuweisungen, das Arbeiten mit Schleifen und das Prüfen von Bedingungen über Vergleichsoperatoren vermittelt werden. Zur Übung und Vertiefung ist es u. a. möglich, die Beispiele zu verändern, zu erweitern oder Codestücke verschiedener Programme miteinander zu kombinieren. Die Programmierung soll aber nicht zwingend im Vordergrund stehen, sondern eher ein Werkzeug zur Umsetzung einer Aufgaben- bzw. Problemstellung sein.

Die Arbeit ist in die folgenden Abschnitte gegliedert: In **Abschnitt 2** wird der Aufbau des umgesetzten Regelkreises vorgestellt und am Beispiel des darin verwendeten Temperatursensors ein erster Arduino-Code geschrieben, welcher als Grundlage für fast alle weiteren Codes dienen wird. Details zur im Regelkreis verbauten Regelstrecke werden in **Abschnitt 3** diskutiert und mit dem Sprungantwortverfahren erste Testmessungen durchgeführt und ausgewertet. In **Abschnitt 4** wird der Regelkreis zum ersten Mal geschlossen und mit Zwei- und Dreipunktregelungen unstetige Regelalgorithmen formuliert. Auch hier werden verschiedene Messreihen aufgenommen, ausgewertet und verglichen, um die Eigenschaften und Grenzen solcher Regelungsarten nachvollziehen und verstehen zu können. **Abschnitt 5** stellt den Schwerpunkt der Arbeit dar. Hier werden stetige Regelungsarten umgesetzt, indem ein PID-Regelungsalgorithmus abgeleitet und mit dem Sprungantwortverfahren getestet wird, bevor auch hierzu diverse Messungen durchgeführt und analysiert werden. In den **Anhängen** finden sich dann noch Alternativen, Erweiterungen und Vertiefendes. Diese Informationen könnten für Interessierte eventuell auch noch nützlich sein.

[1] `https://de.wikipedia.org/wiki/Regelungstechnik` Abfragedatum: 20.02.2022

[2] Andere Plattformen, z. B. ESP32, Teensy, Raspberry Pi Pico ... können alternativ natürlich auch verwendet werden.

[3] Personen, die schon über Programmierkenntnisse verfügen und mit der Arduino-Plattform vertraut sind, können sich natürlich auch gleich auf die Codebeispiele stürzen. Eine Liste aller verwendeter Codes findet sich auf Seite 72.

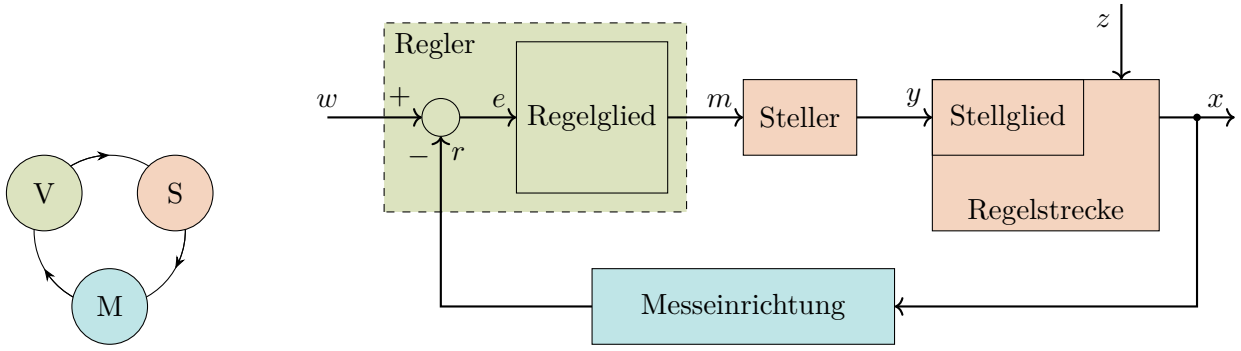
2. Anforderungen und Aufbau des Regelkreises

2.1. Definition einer Regelung

In der Norm DIN IEC 60051-351 findet sich folgende Definition für den Begriff Regelung:

„Vorgang, bei dem fortlaufend eine variable Größe, die Regelgröße, erfasst, mit einer anderen variablen Größe, der Führungsgröße, verglichen und im Sinne einer Angleichung an die Führungsgröße beeinflusst wird. Kennzeichen für das Regeln ist der geschlossene Wirkungsablauf, bei dem die Regelgröße im Wirkungsweg des Regelkreises fortlaufend sich selbst beeinflusst.“

In vereinfachter Weise ausgedrückt bedeutet dies, dass im Regelkreis (vgl. Abb. 2.1a) ein Istwert über eine Messeinrichtung fortlaufend gemessen und mit einem Sollwert verglichen wird. Sofern eine Abweichung festgestellt wird, wird über eine geeignete Stelleinrichtung nachgestellt.
Als **Beispiel** kann man das Autofahren z. B. auf einer Autobahn betrachten, bei der man die Geschwindigkeit möglichst konstant halten möchte. Die momentane Geschwindigkeit wird auf dem Tacho von der fahrenden Person fortlaufend beobachtet und mit der gewünschten Geschwindigkeit verglichen. Bei Abweichungen kann über das Gas- oder Bremspedal nachgestellt werden.
Abbildung 2.1b zeigt einen erweiterten Regelkreis nach DIN IEC 60050-351. Die Bedeutung der darin verwendeten, genormten Formelzeichen ist in Tabelle 2.1 angegeben. Am Beispiel des in dieser Arbeit verwendeten Regelkreises wird in Abschnitt 2.2 eine konkrete Zuordnung vorgenommen.



(a) *vereinfachter Regelkreis*

(b) *Regelkreisschema nach DIN IEC 60050-351*

Abbildung 2.1: *Aufbau eines Regelkreises*

Tabelle 2.1: *Bezeichnungen zum Regelkreis*

w :	Führungsgröße
x :	Regelgröße
r :	Rückführgröße
$e = w - r$:	Regeldifferenz
y :	Stellgröße
m :	Reglerausgangsgröße
z :	Störgröße

2.2. Der verwendete Regelkreis

In Abbildung 2.2 ist ein Aufbau eines kompletten Temperaturregelkreises zu sehen, so wie er für die Tests und Messungen in den folgenden Abschnitten dieser Arbeit verwendet wird. Folgende Anforderungen wurden gestellt:

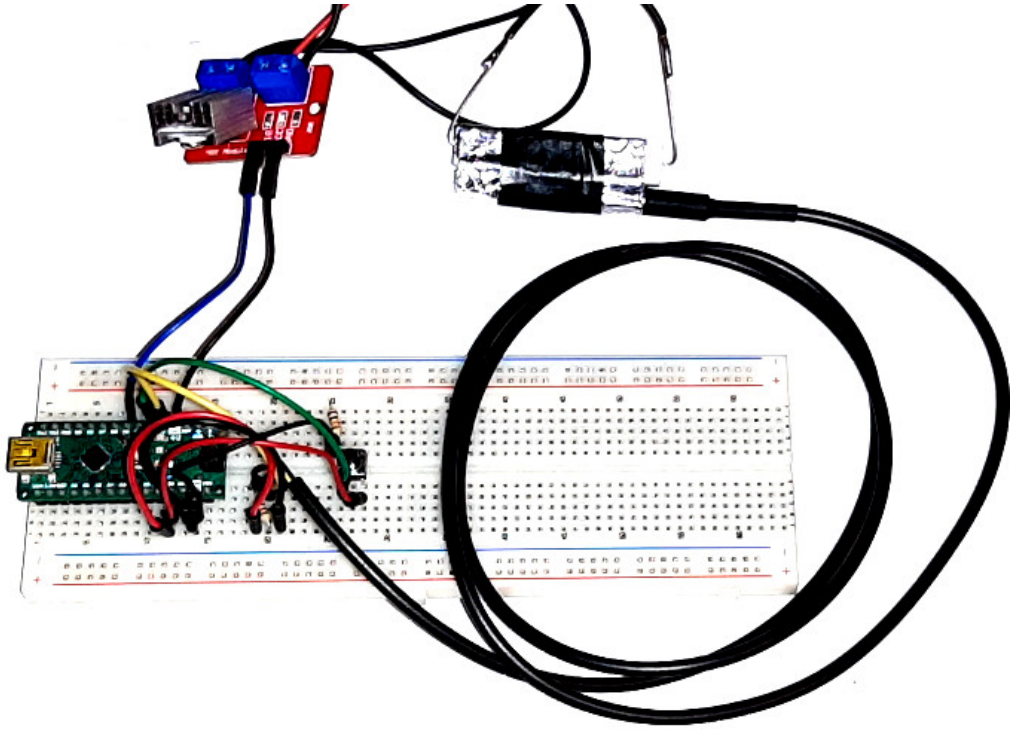
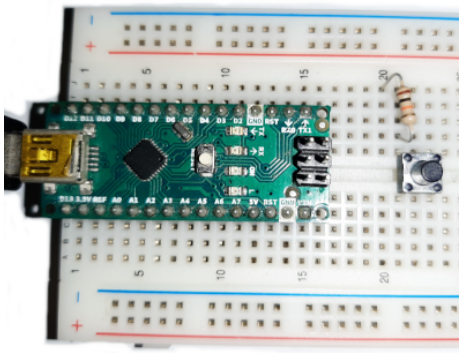


Abbildung 2.2: *Der verwendete Regelkreis*

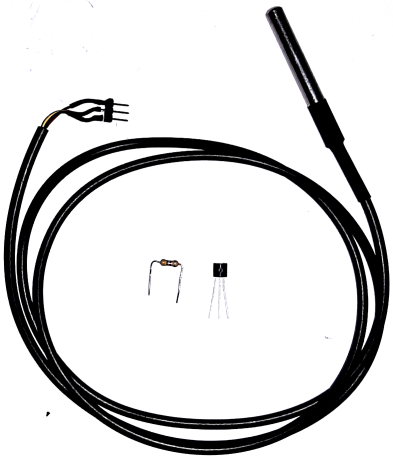
- Der Regelkreis sollte (zunächst) möglichst übersichtlich und einfach gehalten werden.
- Die einzelnen Bestandteile sollten von ihrem Aufbau und ihrer Wirkung durch die Lernenden nachvollzogen werden können.
- Die Kosten sollten insgesamt möglichst gering gehalten werden und die Bauteile möglichst einfach zugänglich sein. Insgesamt sollte der Aufbau klein gehalten werden, insbesondere auch was die Heizleistung betrifft, sodass einer sicheren Umsetzung im Unterricht nichts im Wege steht.
- Die Implementierung der Regelalgorithmen sollte möglichst allgemein gehalten werden, sodass durch geringe Veränderungen/Anpassungen die Übertragung des Gelernten auch auf andere Regelaufgaben bzw. Regelstrecken möglich ist.
- Veränderungen/Variationen an der Hardware und insbesondere an der Regelstrecke sollten möglich bleiben[1].

Die einzelnen Regelkreiselemente sind in Abbildung 2.3 detailliert gezeigt und werden im Folgenden dem Regelkreisschema aus Abbildung 2.1b zugeordnet.
Ein Arduino-Mikrocontroller-Board (Abb. 2.3a[2]) übernimmt die Aufgabe des Reglers. Verschiedene Regelalgorithmen können darauf programmiert werden, wobei insbesondere über die Führungsgröße w der Sollwert von außen vorgegeben werden kann. Zudem wird über die Programmierung die Bestimmung der Regeldifferenz e realisiert.

[1] Eine Anpassung auf eine Temperaturregelung über ein Peltier-Element wird in Anhang C.1 gezeigt.
[2] Gezeigt wird ein Arduino Nano, es kann aber auch jedes andere Modell, z. B. ein Arduino Uno, verwendet werden.

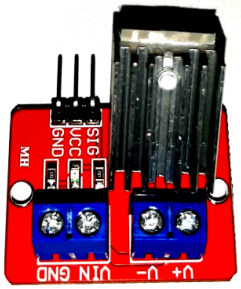


(a) *Arduino-Mikrocontroller als Regler*



(b) *DS18B20-Temperatursensor im Schutzmantel*

(c) *Heizwiderstand*



(d) *Transistor-Board*

Abbildung 2.3: *Die Regelkreiselemente im Detail*

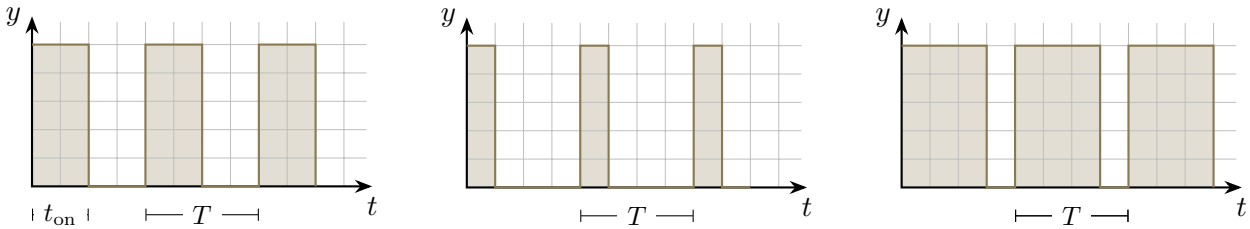
Die Reglerausgangsgröße m ist ein elektrisches (PWM-)Signal (vgl. später mit Abb. 2.4 in Abschnitt 2.2.1), welches an den Feldeffekt-Transistor (MOSFET) (Abb. 2.3d) als Steller übertragen wird. Die eigentliche Stellgröße y ist die Heizleistung, welche sich aus der Stromstärke durch den Widerstand und der am Widerstand anliegenden Spannung ergibt. Diese beiden Größen werden von einem externen Netzteil bereitgestellt (vgl. später mit Abschnitt 3.2).
Der Widerstandsdraht im Heizwiderstand (Abb. 2.3c) ist das Stellglied, welches die Heizleistung auf die Regelstrecke überträgt. Entlang der Regelstrecke läuft der zu regelnde Prozess mit seinen Massen- und Energieströmen ab. Hierzu gehören alle Bestandteile, die während des Regelvorgangs Wärme aufnehmen oder abgeben können. Für die gezeigte Regelstrecke sind dies z. B. der Keramikmantel des Widerstands, die Aluminiumfolie, mit der der Widerstand und der Temperatursensor umwickelt sind (vgl. Abb. 2.2), sowie das Klebeband, aber insbesondere auch der metallische Schutzmantel des Temperatursensors. Weitere Energiespeicher können flexibel in Kontakt mit der Regelstrecke gebracht werden. Alle anderen Regelkreisbestandteile neben der Regelstrecke, welche der Signalverarbeitung dienen, werden unter dem Begriff der Regeleinrichtung zusammengefasst.
Die Temperatur stellt die Regelgröße x im Regelkreis dar und wird vom `DS18B20`-Temperatursensor (Abb. 2.3b) erfasst[1]. Dieser überträgt ein digitales Signal als Rückführgröße r an den Arduino, sodass der Regelkreis geschlossen ist.
Störgrößen z wirken von außen auf die Regelstrecke und beeinflussen die Regelgröße in ungewollter Weise. Hier können insbesondere Schwankungen der Umgebungstemperatur genannt werden. Es können aber auch Störungen provoziert werden, indem kurzfristig andere Materialien in Kontakt[2] mit der Regelstrecke gebracht werden.

[1] In Anhang A wird noch ein alternativer Temperatursensor vorgestellt
[2] Vorsicht bei Berührung (z. B. mit der Hand) bei hoher Temperatur!

2.2.1. Limitierungen des verwendeten Regelkreises

Der verwendete Regelkreis weißt Limitierungen auf, die bedacht werden müssen, insbesondere wenn es um das konkrete Design der Regelstrecke geht (siehe Abschnitt 3.2). So können nur in kurzen Zeitintervallen mit einer begrenzten Abtastrate Daten verarbeitet werden. Hierauf wird im Abschnitt 2.3 näher eingegangen. Hinzu kommt, dass beim verwendeten Arduino-Board die Reglerausgangsgröße als pulsweitenmoduliertes 8 bit-Signal ausgegeben wird und damit nur $2^8 = 256$ verschiedene Zustände[1] annehmen kann. Insgesamt kann nur von einer quasi-kontinuierlichen Regelung gesprochen werden.
Abbildung 2.4 illustriert das Verfahren der Pulsweitenmodulation. Die Stellgröße wechselt mit einer erhöhten Frequenz[2] zwischen den Zuständen an ($y = 0\,\%$) und aus ($y = 100\,\%$). Der zeitliche Mittelwert der Stellgröße wird dabei über den Tastgrad D (engl. duty cycle), d. h. dem Verhältnis aus Einschaltzeit t_{on} zu Periodendauer T bestimmt. Dies hat im Regelkreis folgenden Effekt auf die Stellgröße:
Liegen über den entsprechenden Pin des Arduino-Boards am Gate des Transistors 5 V an, so schaltet dieser nahezu komplett durch und es fällt praktisch die komplette Netzteilspannung U_{b} am Heizwiderstand R_{H} ab (vgl. Abb. 2.5). Die Form des Spannungsverlaufs am Heizwiderstand folgt somit jener des PWM-Signals. Für den zeitlichen Mittelwert der Leistung $\overline{P}$ ergibt sich bei einer ohmschen Last wie dem Heizwiderstand folgende Gleichung:

$$\overline{P} = \frac{U_{\mathrm{b}}^2}{R_{\mathrm{H}}} \cdot \frac{t_{\mathrm{on}}}{T} = \frac{U_{\mathrm{b}}^2}{R_{\mathrm{H}}} \cdot D \tag{2.1}$$



(a) *Tastgrad von* $D = 50\,\%$ (b) *Tastgrad von* $D = 25\,\%$ (c) *Tastgrad von* $D = 75\,\%$

Abbildung 2.4: *Illustrationen zur Pulsweitenmodulation*

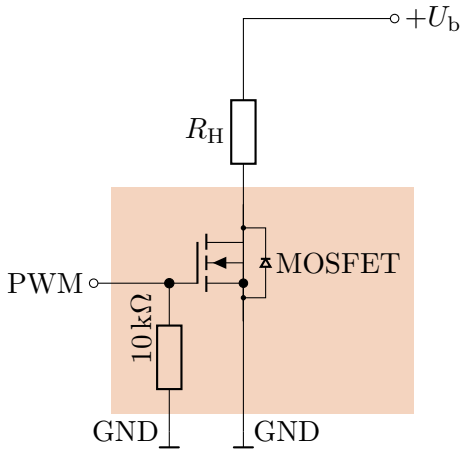


Abbildung 2.5: *Schaltskizze zur Regulierung des Heizstroms durch den Lastwiderstand als Heizelement Der* 10 kΩ*-Pulldown-Widerstand ist auf dem Transistor-Board bereits integriert.*

[1] von 0 bis einschließlich 255
[2] z. B. $f = 980\,\mathrm{Hz}$ an Pin 5 oder 6 beim Arduino Uno oder Nano

2.3. Nutzung des DS18B20-Temperatursensors

Die Kenndaten des verwendeten `DS18B20`-Temperatursensors sind in Tabelle 2.2 aufgelistet.

Tabelle 2.2: *Eigenschaften des `DS18B20`-Temperatursensors*

Betriebsspannung:	3,0 V bis 5,0 V
Anzeigebereich:	−55 °C bis 125 °C
Messgenauigkeit:	±0,5 °C (im Messbereich zwischen −10 °C und 85 °C)
Auflösung:	9 bit bis 12 bit (voreingestellt sind 12 bit)
Identifikation:	eindeutiger 64 bit-Code auf einem onboard ROM

Die Verschaltung, so wie sie in dieser Arbeit verwendet wird, ist in Abbildung 2.6 gezeigt. Der `DS18B20`-Sensor wird mit dem Arduino-Board über Ground (Gnd) und den 5 V Festspannungsausgang verbunden. Der Data-Ausgang wird mit einem digitalen Eingang des Arduino-Boards (Sensor-Pin) verbunden. Zwischen den 5 V Festspannungsausgang und den Sensor-Pin muss zusätzlich noch ein 4,7 kΩ-Pullup-Widerstand geschaltet werden.
Durch den auf dem onboard ROM hinterlegten, eindeutigen 64 bit-Code können mehrere dieser Temperatursensoren an dieselbe Datenleitung (denselben Sensor-Pin) angeschlossen und nacheinander ausgelesen werden[1]. Deshalb ist die Einbindung der `OneWire`-Bibliothek[2] von Jim Stud et al. notwendig. Zur Nutzung des Funktionsumfangs des `DS18B20`-Sensors wird die Bibliothek DallasTemperature[3] von Miles Burton verwendet.

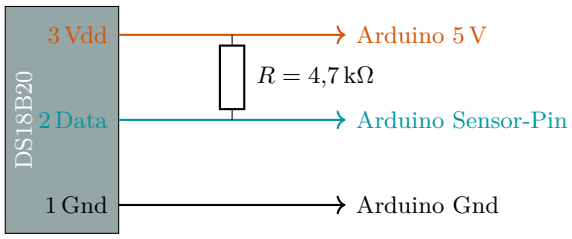


Abbildung 2.6: *Verschaltung des `DS18B20`-Temperatursensors*

Code 2.1 zeigt, wie mithilfe des `DS18B20`-Sensors genau einmal pro Sekunde die Temperatur gemessen und ausgegeben werden kann. In den Zeilen 1 bis 5 werden die beiden notwendigen Bibliotheken eingebunden und Initialisierungen vorgenommen. Insbesondere wird in Zeile 3 eine Konstante (`const`) mit dem Namen `sensorPin` deklariert. Diese bekommt den Wert 2 als Ganzzahl (`int` für integer) zugewiesen, da die Datenleitung des Temperatursensors an Pin Nr. 2 des Arduino angeschlossen wird. Anschließend werden in Zeile 7 eine Variable `t` für die Zeit, in Zeile 8 eine Konstante (`const`) mit dem Namen `dt` für die Größe des Zeitintervalls von einer Sekunde und in Zeile 9 eine weitere Variable `x` für die Temperatur als Regelgröße deklariert. Die Werte dieser drei Größen werden als Dezimalzahlen vom Typ `float` (floating point number) verarbeitet.
Zwischen den Zeilen 11 und 14 erstreckt sich die `setup()`-Funktion, welche in jedem Arduino-Programm enthalten ist und einmalig bei Programmstart ausgeführt wird. Sie nimmt keine Parameter entgegen und liefert keinen Rückgabewert (Typ `void`). In ihr wird in Zeile 12 die serielle Schnittstelle mit einer Baudrate von 9600 sowie in Zeile 13 die Temperaturübertragung über den Sensor aktiviert.
Zwischen den Zeilen 16 und 27 ist die Funktion `loop()` implementiert. Diese, ebenfalls in jedem Arduino-Programm enthaltene Funktion, stellt eine Endlosschleife dar, d. h. der in ihr implementierte Code

[1] Hiervon wird in dieser Arbeit allerdings kein Gebrauch gemacht.
[2] `https://www.pjrc.com/teensy/td_libs_OneWire.html` Abfragedatum: 28.02.2022
[3] `https://github.com/milesburton/Arduino-Temperature-Control-Library` Abfragedatum: 28.02.2022

wird nach einem Durchlauf fortlaufend wiederholt. Sie wird nach Ausführung der `setup()`-Funktion gestartet und liefert ebenfalls keinen Rückgabewert.
In den Zeilen 17 und 18 kommt die Funktion `millis()` zum Einsatz. Sie gibt die Anzahl an Millisekunden als ganze Zahl vom Typ `int` (integer) aus, welche seit dem Programmstart vergangen sind. Hierüber wird der Variablen `t` in Zeile 18 innerhalb eines Schleifendurchlaufs der aktuelle Zeitwert zugeordnet. Die Division durch den Faktor 1000,0 sorgt für die Umrechnung von Millisekunden in Sekunden[1].
In den Zeilen 20 und 21 wird noch die aktuelle Temperatur abgefragt[2] und in der Einheit °C der Variablen `x` zugewiesen. Das Einlesen eines Temperaturwertes nimmt bei der Standardauflösung von 12 bit mit dem `DS18B20`-Sensor etwa 750 ms in Anspruch. Das elektrische Signal der Rückführgröße r wird durch die Funktionalität der `DallasTemperature`-Bibliothek automatisch in den Wert der Regelgröße x umgewandelt, weshalb hier und im weiteren Verlauf dieser Arbeit nicht mehr weiter zwischen x und r unterschieden wird[3].
Zur Übertragung und zur Ausgabe der eingelesenen Werte, z. B. auf dem seriellen Monitor (siehe Abb. 2.7), wird die Funktion `Serial.print()` (bzw. `Serial.println()` mit zusätzlichem Zeilenumbruch) verwendet. Zeile 24 dient dazu, zwischen den Ausgaben des Zeit- und Temperaturwertes einen zusätzlichen Tabulator zur Trennung und Formatierung auszugeben.
Besondere Bedeutung kommt der Zeile 17 zu. Hier wird eine Bedingung formuliert, welche überprüft, ob zwischen dem gerade vorliegenden Zeitpunkt und dem zuletzt bestimmten Zeitwert (jeweils in der Einheit s) mindestens der Wert für das in Zeile 8 festgelegte Zeitintervall vergangen ist. Nur dann wird der zwischen den Zeilen 17 und 26 durch die geschweiften Klammern eingeschlossene Code auch ausgeführt. Da für dessen Ausführung allerdings weniger als eine Sekunde erforderlich ist, wird auf diese Weise erreicht, dass exakt mit der Abtastrate von einmal pro Sekunde Werte übertragen/ausgegeben werden.
Code 2.1 dient als Grundlage für fast alle weiteren in dieser Arbeit verwendeten Programme. Dort taucht er in kaum veränderter Weise immer wieder auf und wird lediglich erweitert.

[1] Die explizite Angabe des Umrechnungsfaktors als Dezimalzahl erzwingt, dass das Ergebnis ebenfalls eine Dezimalzahl vom Typ `float` ergeben und somit nicht grob gerundet werden muss.
[2] Über den Index `0` wird der erste (und hier einzige) Sensor an der Datenleitung angesprochen.
[3] Es wird daher von $r = x$ und $e = w - r = w - x$ ausgegangen.

Code 2.1.: *zeitabhängige Temperaturmessung mit einem DS18B20-Sensor*

```
#include <OneWire.h>
#include <DallasTemperature.h>
const int sensorPin = 2;                // Sensor an Pin 2
OneWire oneWire(sensorPin);
DallasTemperature sensors(&oneWire);

float t = 0.0;                          // Zeit in s
const float dt = 1.0;                   // Zeitintervall in s
float x;                                // Temperatur in °C

void setup() {
  Serial.begin(9600);
  sensors.begin();
}

void loop() {
  if (millis()/1000.0 - t >= dt) {
    t = millis()/1000.0;                // die aktuelle Zeit in s wird bestimmt
    // die aktuelle Temperatur wird eingelessen
    sensors.requestTemperatures();
    x = sensors.getTempCByIndex(0);
    // die aktuellen Werte werden ausgegeben
    Serial.print(t, 1);                 // Zeit mit einer Nachkommastelle
    Serial.print('\t');                 // Tabulator
    Serial.println(x, 3);               // Temperatur mit drei Nachkommastellen und Zeilenumbruch
  }
}
```

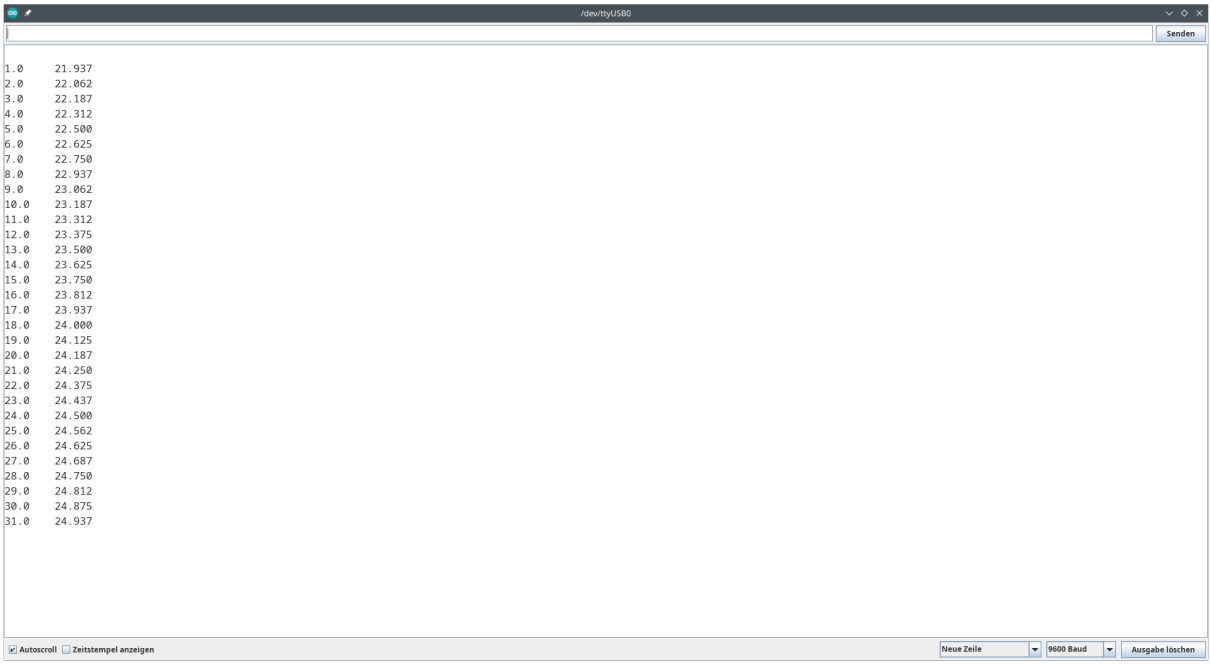


Abbildung 2.7: *zeitabhängige Temperaturaufnahme im Serial-Monitor*

2.4. Kenngrößen einer guten Regelung

Abbildung 2.8 zeigt einen typischen zeitlichen Verlauf einer Regelung. Nach Eintritt einer Störung verändert sich die Regelgröße x und verlässt zunächst das Toleranzband δx. Die Zeitspanne zwischen dem ersten Verlassen und dem Wiedereintritt in das Toleranzband nennt man Anregelzeit T_{an}. Bis die Regelgröße nach dem ersten Verlassen wieder dauerhaft im Toleranzband verbleibt, vergeht die Ausregelzeit T_{aus}. Findet die Regelgröße einen neuen stabilen Zustand, der nicht genau beim Sollwert liegt, so spricht man von einer bleibenden Regeldifferenz e_{b}.

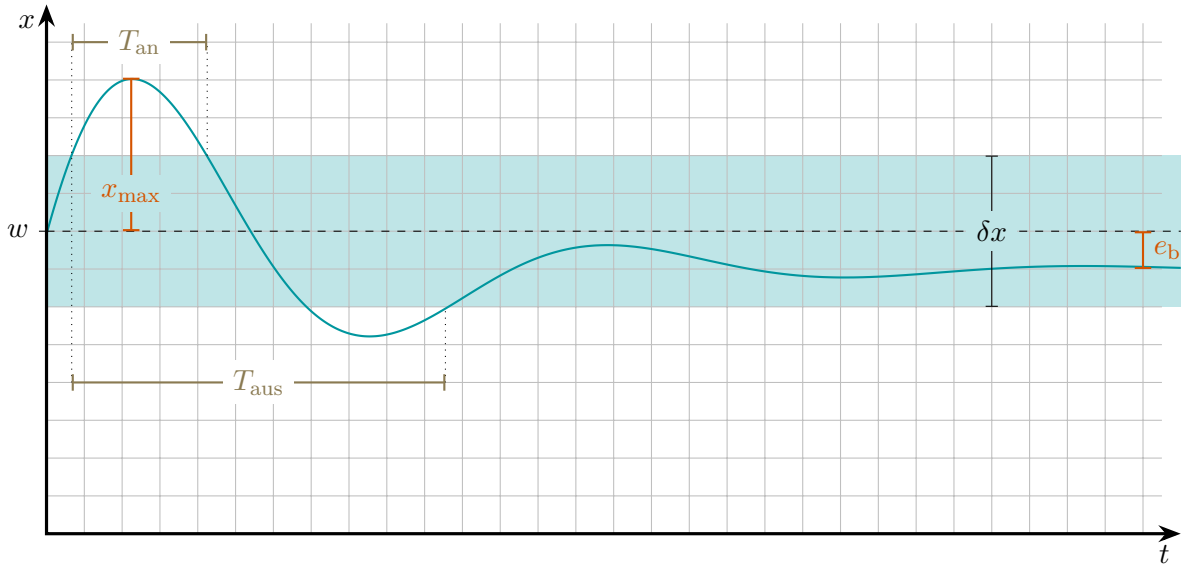


Abbildung 2.8: *Verlauf einer Regelung nach Eintritt einer Störung*

Kennzeichen einer guten Regelung sind demnach, dass nach Eintritt einer Störung der Istwert dem Sollwert wieder angleichen wird und zwar

- so **schnell** wie möglich, d. h. mit kurzer An- und Ausregelzeit,
- so **schwingungsfrei** wie möglich, u. a. mit einer geringen maximalen Überschwingweite x_{max} und
- so **genau** wie möglich, d. h. nach Möglichkeit ohne eine bleibende Regeldifferenz.

2.5. Zusammenfassung und Anregungen für den Unterricht

Bislang wurden Aufbau, Elemente und Anforderungen aber auch Limitierungen zum Temperaturregelkreis vorgestellt. Dabei wurde Fachbegriffen eine konkrete und zum Teil (be)greifbare Bedeutung zugewiesen. Eine mögliche Lernsituation in einem handlungsorientierten (Technik-)Unterricht könnte allgemein als

Design und Aufbau eines (Temperatur-)Regelkreises

formuliert werden. Dabei muss es sich natürlich nicht zwingend um einen Temperaturregelkreis handeln. In einer ersten Phase sollten Informationen zum Aufbau gesammelt und konkrete Bauteile ausgewählt werden. Neben den notwendigen elektrotechnischen Grundlagen gilt es hierbei, sich auch schon mit der Verwendung der Mikrocontroller-Plattform auseinanderzusetzen. Konkret durchgeführte Tests und die Bewertung der Eignung eines oder mehrerer Temperatursensoren stellen einen ersten Schritt zur Vervollständigung des Regelkreises dar. Im nun folgenden Abschnitt 3 werden mit der Stelleinrichtung und der Regelstrecke weitere Bestandteile des Regelkreises konkretisiert, sodass diese in die hier angerissene Lernsituation integriert werden können.

2.6. Ideen für Aufgabenstellungen

- Ordnen Sie die Größen und Fachbegriffe aus Abbildung 2.1b dem Regelkreismodell aus Abbildung D.1 zu.
- Der Füllstand an Wasser in einem Behälter mit einem Zu- und einem Ablauf soll konstant gehalten werden. Entwickeln Sie einen entsprechenden Regelkreis und wählen Sie geeignete Elemente/Bauteile aus.
- Prüfen Sie, welche Auflösung in °C sich mit einem `DS18B20`-Sensor bei Temperaturmessungen erreichen lässt, wenn dieser im Standard-12 bit-Modus arbeitet.
- Prüfen Sie, wie lange die Aufnahme eines Temperaturwertes mit einem `DS18B20`-Sensor mindestens dauert. Hierzu kann z. B. in Code 2.1 der Wert für das Zeitintervall gezielt verändert werden.
- Nehmen Sie einen Pt100-Temperatursensor mithilfe eines `MAX38165`-Controller-Boards in Betrieb und vergleichen Sie dessen Eignung hinsichtlich Auflösung und Zeitdauer einer Messwertaufnahme mit einem `DS18B20`-Sensor.

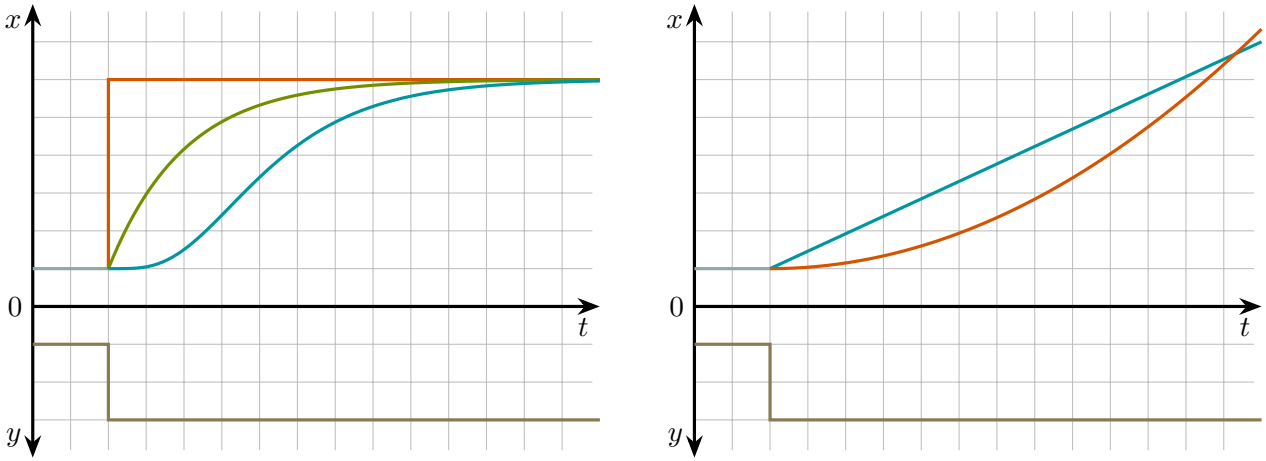
3. Aufbau und Untersuchung der Regelstrecke

3.1. Zeitverhalten von Regelstrecken

Die Regelstrecke ist der Teil des Regelkreises, innerhalb dessen die Regelgröße konstant gehalten werden soll. Hierbei spielt das Zeitverhalten der Regelgröße bei Veränderungen der Stellgröße hinsichtlich der Regelbarkeit und der Eignung verschiedener Reglertypen eine Rolle. Zur Beurteilung des Zeitverhaltens kann man das Sprungantwortverfahren verwenden. Hierbei wird der zeitliche Verlauf der Regelgröße x nach einer sprunghaften Veränderung der Stellgröße y untersucht. Man unterscheidet dabei u. a.:

Regelstrecken mit Ausgleich (auch als proportionale Regelstrecken bezeichnet), bei denen sich für die Regelgröße x wieder ein neuer, stabiler Gleichgewichtszustand einstellen wird (siehe Abb. 3.1a), sowie

Regelstrecken ohne Ausgleich (auch als integrale Regelstrecken bezeichnet), bei denen sich für die Regelgröße x kein neuer, stabiler Gleichgewichtszustand einstellen wird (siehe Abb. 3.1b).



(a) *Regelstrecken mit Ausgleich* (b) *Regelstrecken ohne Ausgleich*

Abbildung 3.1: *Sprungantworten von Regelstrecken*

Ein typisches Beispiel für eine Regelstrecke ohne Ausgleich ist der Füllstand einer Flüssigkeit in einem Behälter (vgl. Abb. 3.2). Fließt genauso viel Flüssigkeit in den Behälter wie auch wieder hinausläuft, so liegt ein Gleichgewichtszustand mit konstant bleibendem Füllstand vor. Nach sprunghafter Erhöhung des Zuflusses steigt der Füllstand allerdings so lange an, bis der Behälter voll ist und ggf. überläuft. Innerhalb des Messbereichs stellt sich hierbei kein neuer stabiler Gleichgewichtszustand ein. Je nach Behälterform kann der Füllstand dabei linear oder in Form einer Kurve ansteigen.

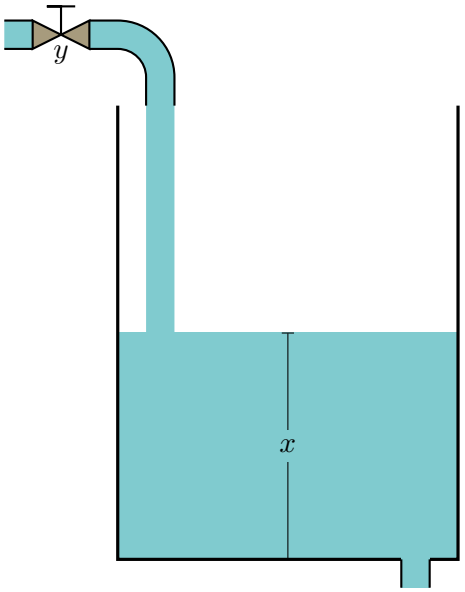


Abbildung 3.2: *Füllstandregelstrecke mit Zu- und Ablauf*

Temperaturregelstrecken sind Regelstrecken mit Ausgleich. Als Beispiel kann die Raumtemperatur betrachtet werden. Wird dem Raum konstant Wärme zugeführt, stellt sich nach einer gewissen Zeit ein Gleichgewicht zwischen der zugeführten Wärme und den Wärmeverlusten nach außen ein. Wird die Heizleistung sprunghaft erhöht, steigt die Temperatur zunächst an. Da die Wärmeverluste aber mit steigendem Temperaturunterschied zur Umgebung ebenfalls größer werden ($\dot{Q} \sim \Delta x$), findet sich (nach einer gewissen Zeit) wieder ein neuer stabiler Gleichgewichtszustand.
Bei Regelstrecken mit Ausgleich (Abb. 3.1a) unterscheidet man solche nullter Ordnung, erster Ordnung und höherer Ordnung, wobei die Ordnung die Anzahl der Energie- und Massenspeicher angibt, welche zu Verzögerungen führen[1]. Man nennt diese daher auch Verzögerungsglieder. Bei zwei oder mehr solcher Glieder spricht man nur noch von einer Regelstrecke höherer Ordnung, da es nicht praktikabel ist, jedes einzelne Verzögerungsglied detailliert zu erfassen bzw. zu bewerten. Hierzu zählen insbesondere Temperaturregelstrecken. In der Praxis reicht es meist aus, das dynamische Verhalten auf zwei charakteristische Zeiten zurückzuführen, über die Aussagen zur Regelbarkeit getroffen werden können. Es handelt sich hierbei um die sogenannte Verzugszeit T_e und Ausgleichszeit T_b. Abbildung 3.3 zeigt, wie diese anhand des Verlaufs der Sprungantwort mithilfe einer Wendetangente bestimmt werden können.

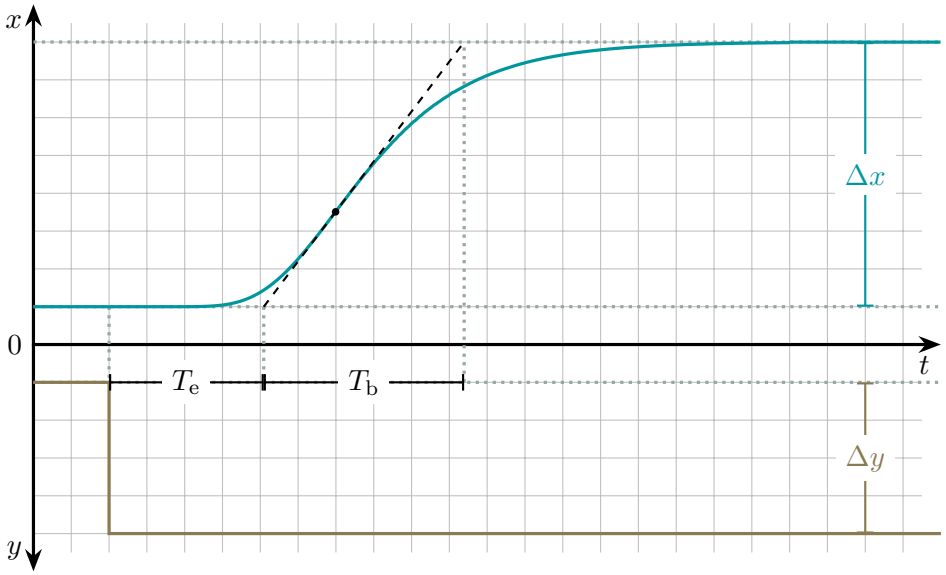


Abbildung 3.3: *Sprungantwort einer Regelstrecke höherer Ordnung*

Die Verzugs- und Ausgleichszeit können herangezogen werden, um Aussagen über die Regelbarkeit einer Regelstrecke höherer Ordnung zu treffen. Hierbei wirkt sich eine kurze Ausgleichszeit im Verhältnis zu einer langen Verzugszeit negativ aus. In Tabelle 3.1 sind hierzu grobe Richtwerte angegeben.

Tabelle 3.1: *Richtwerte zur Beurteilung der Regelbarkeit einer Regelstrecke höherer Ordnung*

Verhältnis $T_\mathrm{b}/T_\mathrm{e}$	*Beurteilung der Regelbarkeit*
$0 < T_\mathrm{b}/T_\mathrm{e} \leq 3$	schwer regelbar
$3 < T_\mathrm{b}/T_\mathrm{e} \leq 10$	noch regelbar
$T_\mathrm{b}/T_\mathrm{e} > 10$	gut regelbar

Eine weitere Kenngröße einer proportionalen Regelstrecke mit Ausgleich ist der Übertragungsbeiwert K_s, welcher nach Gleichung 3.1 berechnet wird. Dabei werden die Stellgrößenänderung Δy und die Regelgrößenänderung Δx meist in Prozent angegeben, wobei man sich bei letzterer auf die Größe

[1] In Anhang E werden konkrete Beispiele aus der Elektrotechnik gezeigt.

des Messumfangs beziehen kann, was man als Normierung bezeichnet.

$$K_\mathrm{s} = \frac{\Delta x}{\Delta y} \tag{3.1}$$

3.2. Designentscheidungen

Die in Abschnitt 2.2.1 beschriebenen Limitierungen der Regeleinrichtung müssen beim konkreten Design der Temperaturregelstrecke berücksichtigt werden. So sollten Verzugs- und Ausgleichszeit deutlich über der möglichen Abtastrate liegen. Dies wird dadurch erreicht, dass mehrere Energie- bzw. Wärmespeicher mit genügend großer Wärmekapazität verbaut werden, die zwischen dem Heizdraht (Stellglied) und dem Temperatursensor (Messglied) liegen.
Andererseits wäre es natürlich auch wünschenswert, dass die Verzugs- und Ausgleichszeit nicht zu groß werden, damit die Aufnahme von Sprungantworten und Regelverläufen nicht zu viel Zeit (im Unterricht) in Anspruch nehmen wird und man Auswirkungen von Veränderungen in Einstellungen innerhalb der Regeleinrichtung und an der Regelstrecke, wozu auch Störgrößen gehören, schnell analysieren kann. Hier muss gegebenenfalls ein Kompromiss gefunden werden. Weiterhin sollte (zunächst) ein Verhältnis aus Ausgleichs- und Verzugszeit erreicht werden, welches eine gute Regelbarkeit verspricht (vgl. Tabelle 3.1).
In Abschnitt 2.2 wurde der verwendete Regelkreis mit samt der Bauteile, welche die Regelstrecke ausmachen, schon kurz vorgestellt. Das Design aus Heizwiderstand, Aluminiumfolie, Schutzmantel des Temperatursensors, ... ist dabei variabel und kann auf unterschiedliche Weise angepasst werden. Durch größere oder kleinere Mengen an Aluminiumfolie lässt sich die Wärmekapazität dieses Energiespeichers anpassen. Zudem kann der Abstand zwischen Temperatursensor und Heizwiderstand mithilfe der Aluminiumfolie als Abstandshalter variiert werden. Größere oder kleinere Mengen an umwickeltem Klebeband stellen ebenfalls einen weiteren Energiespeicher dar und können ggf. auch eingesetzt werden, um eine begrenzt wärmeisolierende Wirkung zu entfalten. Weitere Energiespeicher, z. B. in Form von Knetmasse, können leicht in direkten Kontakt mit der Regelstrecke gebracht werden.
Einen wesentlichen Einfluss auf das zeitliche Verhalten der Temperaturregelung hat auch die Heizleistung. Aus den auf dem Widerstand angegebenen Werten lassen sich die Maximalwerte für Spannung und Strom berechnen, welche von dem Netzteil bereitgestellt werden müssen:

$$U_\mathrm{max} = \sqrt{P_\mathrm{max} \cdot R} = \sqrt{10\,\mathrm{W} \cdot 22\,\Omega} = 14{,}8\,\mathrm{V}$$
$$I_\mathrm{max} = \sqrt{\frac{P_\mathrm{max}}{R}} = \frac{10\,\mathrm{W}}{22\,\Omega} = 0{,}67\,\mathrm{A}$$

Will man die Regelstrecke mit weniger Heizleistung bzw. mit einer geringeren Spannung und einem niedrigeren Strom betreiben, so ist dies auch möglich, was folgende Beispielrechnung zeigt:

$$P(9\,\mathrm{V}) = \frac{(9\,\mathrm{V})^2}{22\,\Omega} = 3{,}68\,\mathrm{W}$$
$$I(9\,\mathrm{V}) = \frac{U}{R} = \frac{9\,\mathrm{V}}{22\,\Omega} = 0{,}41\,\mathrm{A}$$

In dieser Arbeit wird durchgehend eine Spannung von $U_\mathrm{max} = 9\,\mathrm{V}$ verwendet. Um das zeitliche Verhalten der Temperaturregelstrecke zu testen und zu prüfen, ob das Regelstreckendesign die Anforderungen erfüllt, wird nun in Abschnitt 3.3 das Sprungantwortverfahren angewendet.

3.3. Umsetzung des Sprungantwortverfahrens

Zur Umsetzung des Sprungantwortverfahrens wird Code 2.1 nun um die Möglichkeit der Übertragung einer Stellgröße an den Transistor erweitert. Dabei soll der Wert der Stellgröße nach Druck auf einen Taster sprunghaft verändert werden können.
Da mit einem PWM-Signal zur Übertragung der Stellgröße gearbeitet wird, muss der Signalpin (Gatepin) des Transistor-Boards mit einem PWM-fähigen Pin des Arduino-Boards verbunden werden. Beim Arduino Nano ist dies u. a. der Pin Nr. 5. In Zeile 6 von Code 3.1 wird hierzu eine Konstante mit dem Namen `gatePin` deklariert und dieser der Wert 5 als Ganzzahl (`int`) zugewiesen. Über den Namen wird innerhalb der `setup()`-Funktion in Zeile 24 der so angesprochene Arduino-Pin als Ausgang (`OUTPUT`) festgelegt.
In Zeile 12 wird eine Variable `y` für den Wert der Stellgröße in Prozent deklariert. Ihr wird zunächst der Ausgangswert 0 zugewiesen. Im laufenden Programm soll der Stellgröße nach Druck auf den Taster durch einen ausgelösten Interrupt ein neuer Wert zugewiesen werden. Variablen, welche auf diese Weise verändert werden, müssen zusätzlich als `volatile` deklariert werden.
Der Taster wird zusammen mit einem 10 kΩ-pulldown-Widerstand wie in Abbildung 3.4 gezeigt mit dem Ground-Pin (Gnd) und dem 5 V-Festspannungsausgang sowie mit Pin Nr. 3 des Arduino-Boards verbunden. In Zeile 7 wird eine Konstante mit dem Namen `buttonPin` deklariert. Zum Ansprechen von Pin Nr. 3 wird ihr der entsprechende Wert zugewiesen. In Zeile 25 wird innerhalb der `setupt()`-Funktion dieser Arduino-Pin dann noch als Eingang (`Input`) festgelegt.

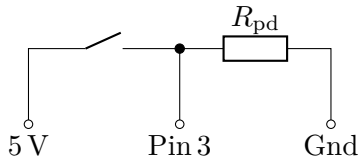


Abbildung 3.4: *Taster mit pulldown-Widerstand*

Beim Arduino Nano ist Pin Nr. 3 interruptfähig. Dies wird in Zeile 26 ebenfalls im Rahmen der `setup()`-Funktion aktiviert. Bei ansteigendem Pegel (`RISING`) von 0 V (`LOW`) auf 5 V (`HIGH`) an Pin Nr. 3 wird die `loop()`-Funktion unterbrochen und die zwischen den Zeilen 17 und 19 implementierte Funktion `buttonFunc()` ausgeführt, innerhalb welcher der Stellgröße `y` ein neuer, fester Wert zugewiesen wird. In Zeile 36 wird in der Endlosschleife der `loop()`-Funktion der Wert der Stellgröße von Prozent in den Wertebereich eines 8 bit-Signals von 0 bis 255 umgerechnet, sodass über die `analogWrite()`-Funktion der entsprechende Wert als PWM-Signal an den mit dem Transistor verbundenen Ausgang des Arduino-Boards übertragen wird.
Neben den Werten der Zeitvariablen `t` und der Regelgrößenvariablen `x` wird in Zeile 42 nun auch noch der Wert der Stellgrößenvariablen `y` mithilfe der Anweisung `Serial.println(y)` über die serielle Schnittstelle übertragen.

Code 3.1.: *Umsetzung des Sprungantwortverfahrens*

```
#include <OneWire.h>
#include <DallasTemperature.h>
const int sensorPin = 2;              // Sensor an Pin 2
OneWire oneWire(sensorPin);
DallasTemperature sensors(&oneWire);
const int gatePin = 5;                // Transistor an Pin 5
const int buttonPin = 3;              // Taster an Pin 3

const float dt = 1.0;                 // Zeitintervall in s
float t = 0.0;                        // aktuelle Zeit in s
float x = 0.0;                        // aktuelle Temperatur in °C
volatile int y = 0;                   // Stellgroesse in Prozent

/* Die im folgenden definierte Funktion wird ausgefuehrt,
 * solbald durch den Taster ein Interrupt ausgeloest wird.
 */
void buttonFunc() {
  y = 50;
}

void setup() {
  Serial.begin(9600);
  sensors.begin();
  pinMode(gatePin, OUTPUT);
  pinMode(buttonPin, INPUT);
  attachInterrupt(digitalPinToInterrupt(buttonPin), buttonFunc, RISING);
}

void loop() {
  if (millis() / 1000.0 - t >= dt) {
    t = millis() / 1000.0;

    sensors.requestTemperatures();
    x = sensors.getTempCByIndex(0);
    // Die Stellgroesse wird zur Uebertragung in ein 8bit-Wertebereich umgerechnet
    analogWrite(gatePin, 255 * y / 100);

    Serial.print(t, 1);
    Serial.print('\t');
    Serial.print(x, 3);
    Serial.print('\t');
    Serial.println(y);
  }
}
```

3.3.1. Beispieldaten und Interpretation

Abbildung 3.5 zeigt die Aufnahme von Sprungantworten der Temperaturregelstrecke bei unterschiedlich großen Stellgrößensprüngen. Diese wurden jeweils zum Zeitpunkt $t = 1\,\text{min}$ durch Betätigung des Tasters ausgelöst. Bei einer der Kurven wurde etwas Knetmasse als zusätzlicher Energiespeicher in Kontakt mit der Regelstrecke gebracht.
Man beobachtet, dass die Temperatur jeweils ausgehend von ca. 18,1 °C gegen einen neuen stabilen Wert strebt. Dieser liegt bei einem Stellgrößensprung von 25 % bei ca. 37,1 °C und bei einem Sprung von $y = 50\,\%$ bei ca. 16,1 °C. Das Verhältnis der Stellgrößenänderungen stimmt mit dem Verhältnis der Regelgrößenänderung im Rahmen der signifikanten Ziffern hervorragend überein:

$$\frac{\Delta y_1}{\Delta y_2} = \frac{50\,\%}{25\,\%} = 2 \qquad \text{vergleiche} \qquad \frac{\Delta x_1}{\Delta x_2} = \frac{56{,}1\,°\text{C} - 18{,}1\,°\text{C}}{37{,}1\,°\text{C} - 18{,}1\,°\text{C}} = \frac{38{,}0\,°\text{C}}{19{,}0\,°\text{C}} = 2{,}00$$

Legt man einen Messumfang von 0 °C bis 100 °C zugrunde, so berechnet sich der Übertragungsbeiwert dieser Temperaturregelstrecke wie folgt, wobei im Zähler der Brüche die Regelgrößenänderung relativ zum Messumfang jeweils in Prozent umgerechnet wird:

$$K_\text{s} = \frac{100\,\% \cdot 19{,}0\,°\text{C}/100\,°\text{C}}{25\,\%} = 0{,}76 \quad \text{bzw.} \quad K_\text{s} = \frac{100\,\% \cdot 38{,}0\,°\text{C}/100\,°\text{C}}{50\,\%} = 0{,}76$$

Weiterhin ist in Abb. 3.5 ein Wendepunkt in den Kurvenverläufen zu erkennen. Insgesamt lässt sich schließen, dass es sich um eine proportionale Regelstrecke höherer Ordnung handelt.
Zudem ist klar zu erkennen, dass durch etwas Knetmasse als zusätzlicher Energiespeicher bei gleicher Stellgrößenänderung ein wesentlich kleinerer Temperatursprung auftritt, was neben der erhöhten Wärmekapazität auch über eine vergrößerte Austauschfläche mit der Umgebung zu erklären ist.

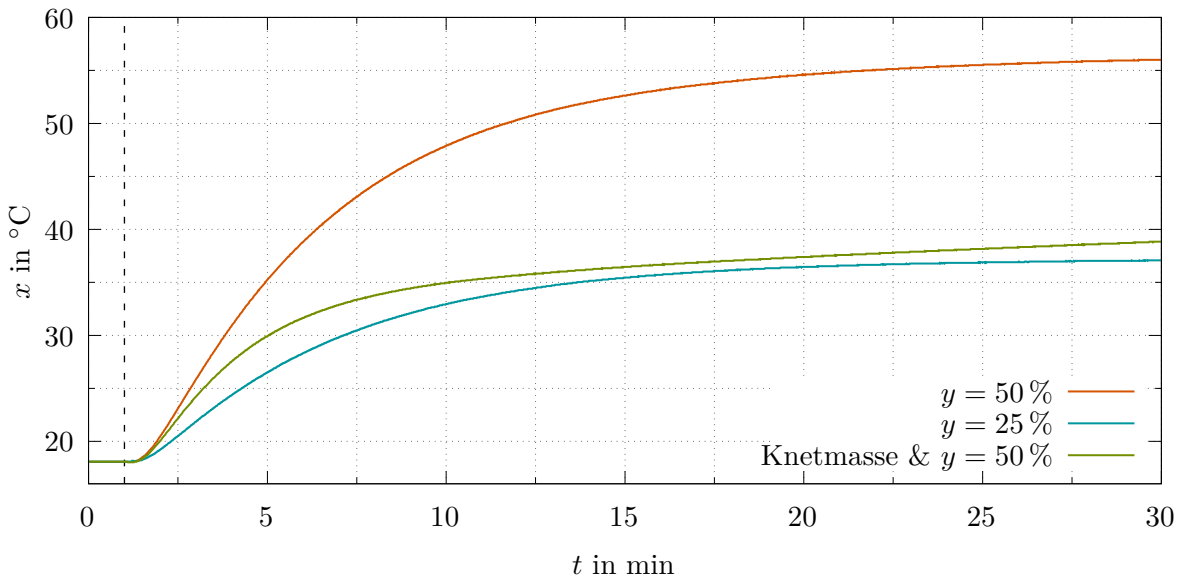


Abbildung 3.5: *Sprungantworten bei verschiedenen Stellgrößenänderungen zum Zeitpunkt* $t = 60\,\text{s}$

Das dynamische Verhalten einer proportionalen Regelstrecke höherer Ordnung lässt sich durch die Verzugszeit T_e und die Ausgleichszeit T_b charakterisieren. Diese beiden Zeiten können anhand der Sprungantwort mit Hilfe einer Wendetangente grafisch bestimmt werden.
Zur Ermittelung des Wendepunkts sucht man den Zeitpunkt im Temperaturverlauf, zu dem die Steigung s am größten ist. Diese kann man aus dem Differenzenquotienten aufeinanderfolgender

Datenpunkte näherungsweise bestimmen.

$$s = \frac{\Delta x}{\Delta t} = \frac{x_k - x_{k-1}}{t_k - t_{k-1}} \tag{3.2}$$

Um der begrenzten Genauigkeit durch die zeitlich Auflösung sowie jener des Temperatursensors etwas Rechnung zu tragen, empfiehlt es sich, die Steigung zu einem Zeitpunkt immer aus mehreren umgebenden Punkten zu bestimmen. Hierzu kann z. B. in einem Tabellenkalkulationsprogramm die Funktion `=Steigung(...)` verwendet werden, was in Tabelle 3.2 gezeigt wird.

Tabelle 3.2: *Bestimmung der Steigung mit einer Tabellenkalkulation*

	A	B	C
1	t in s	x in °C	s in °C/s
2	1,0	22,21	—
3	2,0	22,21	—
4	3,0	22,27	`=Steigung(B2:B6;A2:A6)`
5	4,0	22,27	`=Steigung(B3:B7;A3:A7)`
6	5,0	22,27	`=Steigung(B4:B8;A4:A8)`
7	6,0	22,33	*usw.*
8	7,0	22,33	*usw.*

Die Verzugs- und Ausgleichszeit werden nun zur Sprungantwort mit $y = 50\,\%$ (ohne Knetmasse) exemplarisch bestimmt. In einer Umgebung von $t \pm 6\,\mathrm{s}$ findet sich die größte Steigung zum Zeitpunkt $t_{\mathrm{wp}} = 160\,\mathrm{s}$ mit einem Wert von $s_{\mathrm{wp}} = 9{,}375 \cdot 10^{-2}\,\frac{^\circ\mathrm{C}}{\mathrm{s}}$. Die gemessene Temperatur beträgt zu diesem Zeitpunkt $x_{\mathrm{wp}} = 24{,}0\,^\circ\mathrm{C}$, sodass die Wendetangente durch die Gleichung

$$x(t) = s_{\mathrm{wp}} \cdot (t - t_{\mathrm{wp}}) + x_{\mathrm{wp}} = 9{,}375 \cdot 10^{-2}\,\frac{^\circ\mathrm{C}}{\mathrm{s}} \cdot (t - 160\,\mathrm{s}) + 24{,}0\,^\circ\mathrm{C}$$

beschrieben werden kann. Vor dem Stellgrößensprung lag die Temperatur konstant bei 18,1 °C. Aus dem Temperaturverlauf wird der neue stabile Temperaturwert bei ca. 56,1 °C abgeschätzt.

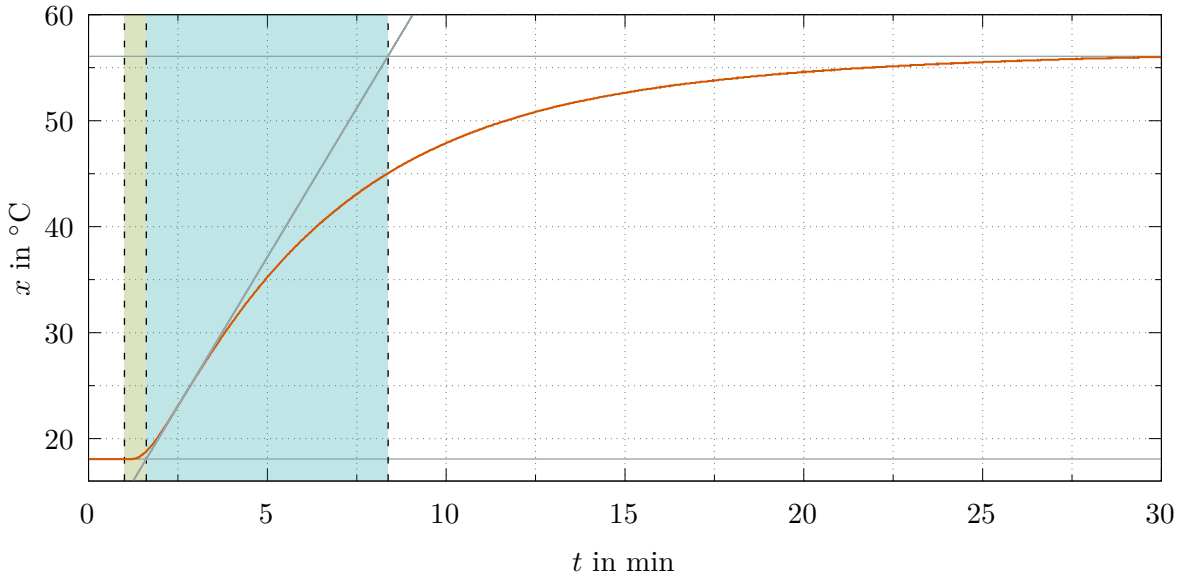


Abbildung 3.6: *Sprungantwort, Wendetangente sowie Verzugs- und Ausgleichszeit*

Hiermit berechnet sich eine Verzugszeit von $T_e \approx 36{,}7\,\mathrm{s}$ und eine Ausgleichszeit von $T_b \approx 402\,\mathrm{s}$. Das Verhältnis von Ausgleichs- zu Verzugszeit beträgt damit

$$\frac{T_b}{T_e} = \frac{402\,\mathrm{s}}{36{,}7\,\mathrm{s}} \approx 11 \quad ,$$

womit laut Tabelle 3.1 eine gute Regelbarkeit der Regelstrecke zu erwarten ist.

3.4. Zusammenfassung und Anregungen für den Unterricht

Es wurde gezeigt, wie ein Transistor als Stellgerät verwendet und in den Regelkreis integriert werden kann. Zusammen mit der Temperaturmessung ist damit auch ein Test des Regelstreckenverhaltens möglich, sodass für die in Abschnitt 2.5 angerissene Lernsituation nun alle technischen Mittel zur Vervollständigung des Regelkreises prinzipiell zur Verfügung stehen und in den folgenden Abschnitten 4 sowie 5 der Fokus auf verschiedene Regelalgorithmen gelegt werden kann.
Unzählige Variationsmöglichkeiten an der Regelstrecke bieten an dieser Stelle allerdings noch enormes Potential für vielfältige experimentelle Aufgaben zur

Charakterisierung des Zeitverhaltens einer (Temperatur-)Regelstrecke.

Exemplarisch kann hier neben dem Hinzufügen von Energiespeichern auch die Untersuchung der Auswirkung einer zusätzlichen Isolierung oder einer Veränderung der Sensorposition genannt werden. Auch andere Regelstreckendesigns wären denkbar. Ein mit Wasser gefülltes Gefäß, in welches ein kleines Heizelement eintaucht, könnte ein Modell eines Reaktors einer Anlage aus der chemischen Industrie darstellen. Hier könnten neben Veränderungen des Füllstands und des Einflusses einer Isolierung auch die Auswirkung einer Rührvorrichtung auf den Temperaturverlauf von Interesse sein.

3.5. Ideen für Aufgabenstellungen

- Nehmen Sie für die Temperaturregelstrecke Sprungantworten mit verschieden großen Stellgrößensprüngen auf. Vergleichen Sie die Verläufe und beurteilen Sie, um welche Art von Regelstrecke es sich handelt.
- Bestimmen Sie den Übertragungsbeiwert K_s der Temperaturregelstrecke.
- Berechnen Sie mithilfe des Übertragungsbeiwertes der Regelstrecke die notwendige Stellgröße, um ausgehend von einer Umgebungstemperatur von 20 °C eine stabile Temperatur von 48 °C zu erreichen.
- Prüfen Sie mithilfe des Übertragungsbeiwertes, ob ausgehend von einer Umgebungstemperatur von 20 °C eine Temperatur von 120 °C mit dem Regelstreckenmodell erreicht werden kann.

  Sollten Sie zu dem Ergebnis kommen, dass dies nicht möglich ist, entwickeln Sie Maßnahmen, um die gewünschte Temperatur erreichen zu können.
- Bestimmen Sie die Verzugs- und Ausgleichszeit der Temperaturregelstrecke über die Konstruktion einer Wendetangente z. B. mithilfe eines Tabellenkalkulationsprogramm.
- Prüfen Sie, ob die Lage/Position des Temperatursensors einen Einfluss auf die Verzugs- und Ausgleichszeit hat.

- Erweitern Sie die Temperaturregelstrecke durch einen weiteren Energiespeicher[1].

  Schätzen Sie die Auswirkung auf die Sprungantwort der Regelstrecke ab und kontrollieren Sie Ihre Erwartung durch eine entsprechende Datenaufnahme und Auswertung.

- Untersuchen Sie, wie sich ein zusätzlicher Energiespeicher auf das Verhältnis von Ausgleichs- zu Verzugszeit auswirkt und beurteilen Sie die Konsequenz hinsichtlich der Regelbarkeit.

- Ermitteln Sie den Zusammenhang zwischen dem 8 bit-PWM-Stellsignal und der Heizleistung. Messen Sie dazu z. B. mithilfe von Multimetern die zeitlichen Effektivwerte des Stroms durch und des Spannungsabfalls über dem Heizwiderstand bei verschiedenen Stellwerten. Stellen Sie die Ergebnisse grafisch dar und beurteilen Sie, ob man von einer Proportionalität ausgehen kann.

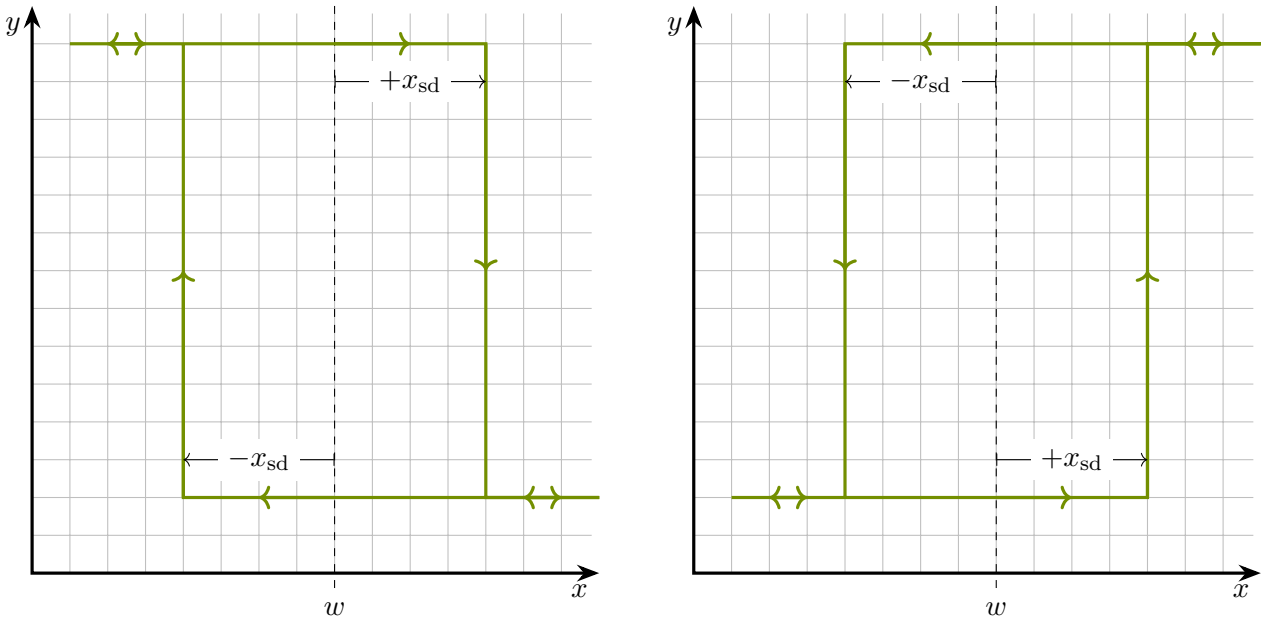
[1] z. B. Knetmasse, einen passiven Kühlkörper, ...

4. Umsetzung von unstetigen Regelungen

Unstetige Regler geben nur eine begrenzte Anzahl an Stellwerten aus. Als erstes Beispiel kann hier der Zweipunktregler genannt werden, welcher, wie es der Name schon andeutet, genau zwei Stellwerte liefert. Bei einer Temperaturregelung kann es sich beispielsweise um die Werte $y = 0\,\%$ und $y = 100\,\%$ handeln, d. h. das Heizelement kann entweder aus- oder eingeschaltet werden. Im eingeschalteten Zustand arbeitet es dann mit einer konstanten Heizleistung. Der Steller muss somit lediglich eine Schaltfunktion übernehmen, sodass man im Temperaturregelkreis auch ein einfaches Relais anstelle eines Transistors verwenden könnte.
Ein weiteres Beispiel für einen unstetigen Regler ist der Dreipunktregler. Dieser liefert drei mögliche Schaltzustände. Bei einer Temperaturregelung könnte es sich dabei um die Zustände „heizen mit voller Leistung“ ($y = 100\,\%$), „heizen mit verminderter Leistung“ (z. B. $y = 50\,\%$) und den ausgeschalteten Zustand ($y = 0\,\%$) handeln. Mit einem geeigneten Steller und Stellglied[1] wären auch die Zustände „heizen“ ($y = +100\,\%$), „kühlen“ ($y = -100\,\%$) und „ausschalten“ ($y = 0\,\%$) denkbar.
Mehrpunktregler können auch dann verwendet werden, wenn eine entsprechende Anzahl an Stellgliedern durch individuelle Schalter angesprochen werden können. So kann beispielsweise eine Dreipunktregelung mit zwei (gleichen) Heizelementen auch durch die Zustände „beide Heizelementen eingeschaltet“, „nur ein Heizelement eingeschaltet“ und „kein Heizelement eingeschaltet“ realisiert werden.

4.1. Zweipunktregelung mit Hysterese

Schaltet der Steller bei einer schnellen Regelstrecke unterhalb des Sollwertes w an und bei Überschreitung ab, so muss die Stellgröße sehr oft ihren Wert ändern und der Steller sowie das Stellglied können durch das häufige Schalten stark belastet werden. Um die Schalthäufigkeit eines unstetigen Reglers zu reduzieren, arbeitet man meist mit einer Schaltdifferenz $\pm x_{\mathrm{sd}}$, welche man auch als Hysterese bezeichnet[2].



(a) *Ausschalten oberhalb und Einschalten unterhalb des Sollwerts*

(b) *Einschalten oberhalb und Ausschalten unterhalb des Sollwerts*

Abbildung 4.1: *Hysteresekurven von Zweipunktreglern*

[1] z. B. einem Peltier-Element, vgl. Anhang C.1
[2] Die Schaltpunkte müssen nicht zwingend symmetrisch um den Sollwert liegen.

Die Hysteresekurve aus Abbildung 4.1a zeigt, wie der Stellwert bei Überschreitung der Grenze $w + x_{\mathrm{sd}}$ vom oberen, maximalen Wert $y_{\max}$ auf den unteren, minimalen Wert $y_{\min}$ springt. Für eine Temperaturregelung würde dies bedeuten, dass das Heizelement erst etwas über der Solltemperatur abgeschaltet wird. Bei Unterschreitung der Grenze $w - x_{\mathrm{sd}}$ wird wieder der obere Stellwert angenommen, d. h. das Heizelement wird etwas unterhalb der Solltemperatur wieder eingeschaltet.
Will man eine Temperatur auf einen Wert unterhalb der Umgebungstemperatur regeln, so benötigt man ein Kühlelement anstelle eines Heizelements. Abbildung 4.1b zeigt eine Hysteresekurve für eine entsprechende Zweipunktregelung. Hier wird das Kühlelement eingeschaltet, sobald die Temperatur den Wert $w + x_{\mathrm{sd}}$ überschreitet. Bei Unterschreitung des Wertes $w - x_{\mathrm{sd}}$ wird es hingegen wieder ausgeschaltet.

4.1.1. Implementierung einer Zweipunkttemperaturregelung mit Hysterese

Code 3.1 wird nun zu einer Zweipunkttemperaturregelung angepasst und erweitert[1]. In Zeile 12 wird eine Konstante `w` für den Sollwert festgelegt. Dieser wird der Wert 50,0 °C zugewiesen. Zudem wird in Zeile 13 eine weitere Konstante `xSD` für die Schaltdifferenz deklariert mit einer Wertzuweisung von 0,25 °C. Um zwischen zwei Stellwerten schalten zu können, müssen konstante Werte für die minimale Stellgröße `yMin` und die maximale Stellgröße `yMax` in den Zeilen 15 und 16 festgelegt werden. Diese können dann im laufenden Programm der variablen Stellgröße `y` zugewiesen werden, welche in Zeile 17 deklariert wird. Ihr wird ein Anfangswert von `yMax` zugewiesen, sodass das Programm zunächst mit dem Heizzustand beginnt.
Als Steller wird weiterhin das Transistorboard (und kein Relais) verwendet (Codezeilen 6, 22 und 35), sodass durch Veränderung der Werte für `yMax` Zweipunktregelungen mit unterschiedlichen Heizleistungen untersucht werden können.
Innerhalb der `loop()`-Funktion wird in jedem Schleifendurchlauf in den Zeilen 32 und 33 geprüft, ob ein Wechsel des Stellgrößenwertes erfolgen muss. Hierzu werden Vergleichsoperatoren verwendet, die prüfen, ob zwei Werte identisch/gleich (`==`) sind bzw. ob ein Wert größer (`>`) oder kleiner (`<`) als ein anderer ist.
Zeile 32 liest sich dabei wie folgt: Besitzt die Stellgröße `y` aktuell den Wert der Konstanten `yMax` und liegt der momentane Wert der Regelgröße `x` über dem Ergebnis der Rechnung `w + xSD`, dann wird der Stellgröße der Wert der Konstanten `yMin` neu zugewiesen, d. h. das Heizelement wird ausgeschaltet.
Zeile 33 bedeutet analog dazu: Besitzt die Stellgröße `y` aktuell den Wert der Konstanten `yMin` und liegt der momentane Wert der Regelgröße `x` unter dem Ergebnis der Rechnung `w - xSD`, dann wird der Stellgröße der Wert der Konstanten `yMax` neu zugewiesen, d. h. das Heizelement wird eingeschaltet.
Die beiden Formulierungen spiegeln den Hystereseverlauf aus Abbildung 4.1a wider. In allen anderen Fällen wird nichts weiter unternommen, was bedeutet, dass innerhalb des aktuellen Schleifendurchlaufs die Stellgröße ihren momentanen Wert beibehält, bis in einem späteren Schleifendurchlauf nach ausreichender Temperaturveränderung die entsprechende Bedingung für einen Stellgrößenwechsel erfüllt wird.

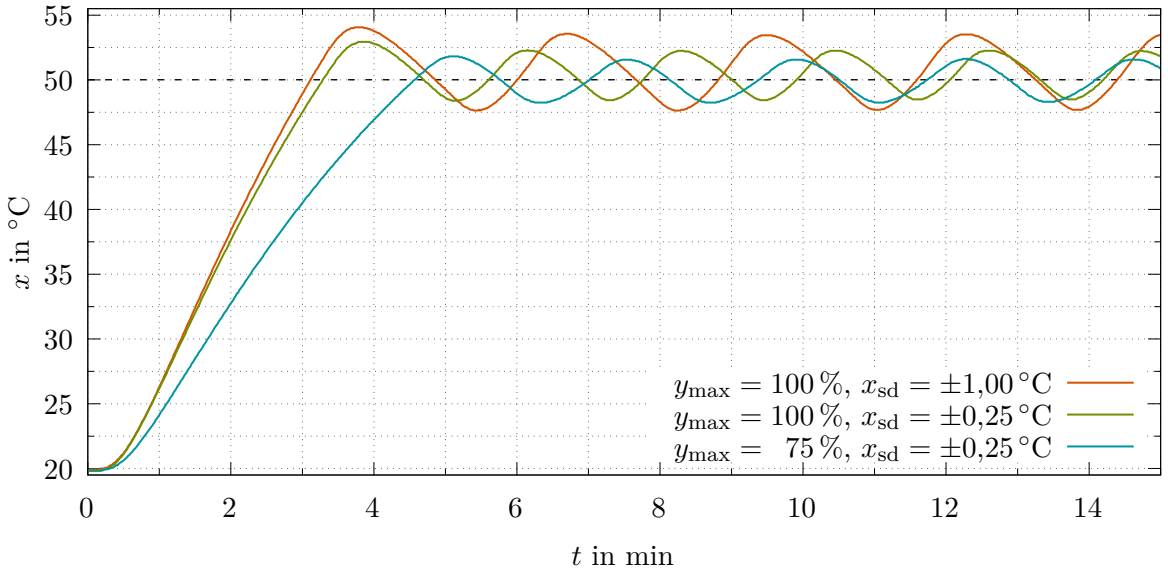
[1] Da der Taster hierbei keine Verwendung findet, können alle entsprechenden Zeilen, darunter auch die Interruptfunktion, entfernt werden.

Code 4.1.: *Zweipunktregelung mit Hysterese*

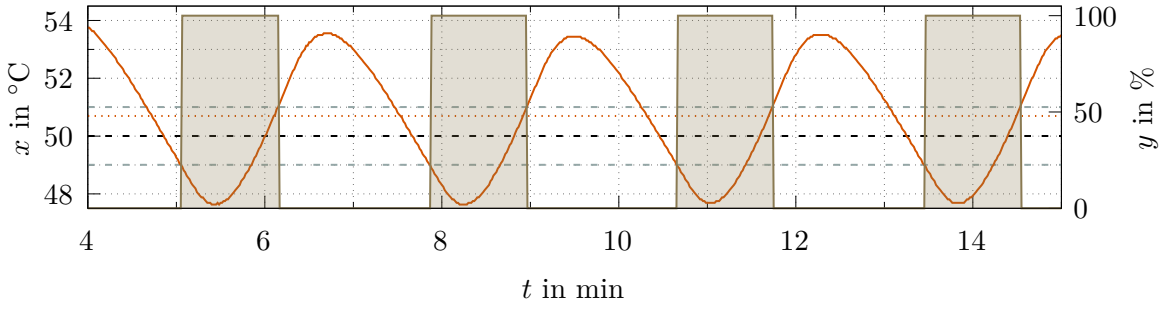
```
#include <OneWire.h>
#include <DallasTemperature.h>
const int sensorPin = 2;
OneWire oneWire(sensorPin);
DallasTemperature sensors(&oneWire);
const int gatePin = 5;  // Pin zum Ansteuern des Transistor-Gates

float t = 0.0;          // aktuelle Zeit in s
const float dt = 1.0;   // Zeitintervall in s

float x = 0.0;          // aktuelle Temperatur in °C
const float w = 50.0;   // Sollwert in °C
const float xSD = 0.25; // Schaltweite in °C

const int yMin = 0;     // minimale Stellgroesse in Prozent
const int yMax = 100;   // maximale Stellgroesse in Prozent
int y = yMax;           // Stellgroesse in Prozent

void setup() {
  Serial.begin(9600);
  sensors.begin();
  pinMode(gatePin, OUTPUT);
}

void loop() {
  if (millis() / 1000.0 - t >= dt) {
    t = millis() / 1000.0;

    sensors.requestTemperatures();
    x = sensors.getTempCByIndex(0);

    if (y == yMax and x > w + xSD) {y = yMin;}      // Bedingung an der oberen Schaltschwelle
    else if (y == yMin and x < w - xSD) {y = yMax;} // Bedingung an der unteren Schaltschwelle

    analogWrite(gatePin, 255 * y / 100);

    Serial.print(t, 1);
    Serial.print('\t');
    Serial.print(x, 3);
    Serial.print('\t');
    Serial.println(y);
  }
}
```

4.1.2. Beispieldaten und Interpretation

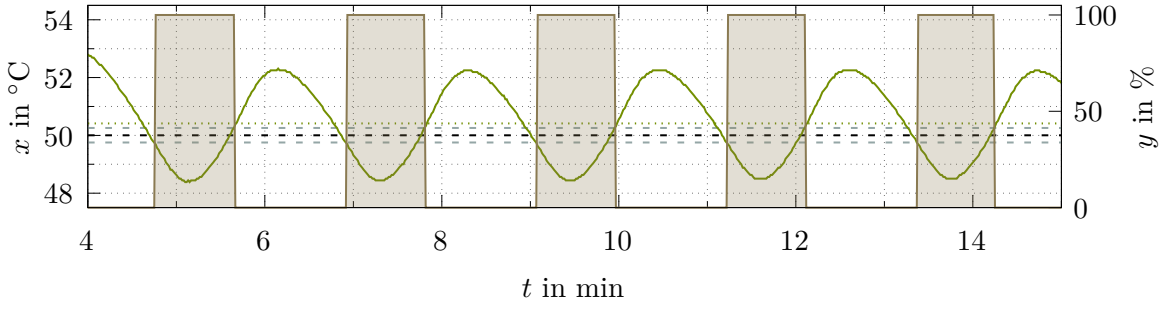
Mithilfe von Code 4.1 werden nun Verläufe von Zweipunkttemperaturregelungen mit einem Sollwert von $w = 50{,}0\,°\mathrm{C}$ mit unterschiedlichen Werten für die Schaltdifferenz x_{sd} und die maximale Stellgröße $y_{\max}$ bzw. Heizleistung aufgenommen. Die Ergebnisse sind in Abbildung 4.2 gegenübergestellt. In allen Fällen ist zu erkennen, dass es ausgehend von einer Raumtemperatur von ca. 20 °C nach einer gewissen Aufheizzeit zu einer Oszillation der Temperatur um den Sollwert kommt. Dabei dauert es bei dem kleineren Wert für die maximale Stellgröße und der damit zusammenhängend geringeren Heizleistung erwartungsgemäß länger, bis der Sollwert zum ersten Mal erreicht wird.



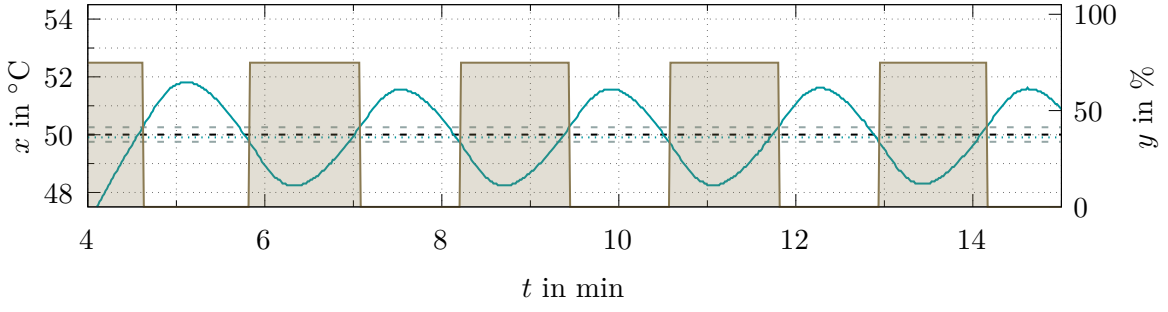
(a) *verschiedene Einstellungen im direkten Vergleich*



(b) *Ausschnitt zu:* $y_{\max} = 100\,\%$, $x_{\mathrm{sd}} = \pm 1{,}00\,^{\circ}\mathrm{C}$



(c) *Ausschnitt zu:* $y_{\max} = 100\,\%$, $x_{\mathrm{sd}} = \pm 0{,}25\,^{\circ}\mathrm{C}$



(d) *Ausschnitt zu:* $y_{\max} = 75\,\%$, $x_{\mathrm{sd}} = \pm 0{,}25\,^{\circ}\mathrm{C}$

Abbildung 4.2: *Temperaturverläufe bei Zweipunktregelungen*

Die Ausschnitte in den Abbildungen 4.2b, 4.2c und 4.2d ermöglichen einen genaueren Blick auf das oszillierende Verhalten. Die gestrichelten, horizontalen Linien geben die Lage des Sollwertes und der

Schaltgrenzen an. Neben dem Temperaturverlauf ist zusätzlich der Stellgrößenverlauf eingetragen, wobei die eingefärbten Flächen andeuten, dass in diesen Zeiträumen geheizt wird. Der Flächeninhalt stellt dabei ein Maß für die zugeführte Wärmemenge dar.
In allen drei Verläufen kann man feststellen, dass nach Erreichen der oberen Schaltschwelle ($w + x_{\mathrm{sd}}$) die Stellgröße auf den Wert 0 % fällt und somit das Heizelement ausgeschaltet wird. Aufgrund der Verzugszeit innerhalb der Regelstrecke steigt die Temperatur trotzdem noch über einen gewissen Zeitraum etwas weiter an. Analog dazu verhält es sich bei Unterschreitung der unteren Schaltschwelle ($w - x_{\mathrm{sd}}$). Hier sinkt die Temperatur noch weiter und es vergeht ebenfalls etwas Zeit, bis die Temperatur nach Anschalten des Heizelements wieder ansteigt. Die in der Farbe des Temperaturverlaufs eingetragen gepunkteten, horizontalen Linien stellen jeweils den zeitlichen Temperaturmittelwert dar. Eine genaue Betrachtung zeigt, dass diese Mittelwerte nicht exakt beim Sollwert liegen.
Abbildung 4.2b zeigt, dass für den bei der größten Schaltdifferenz und bei der größten Heizleistung aufgenommenen Verlauf auch die höchste Schwankungsbreite der Temperatur als Regelgröße auftritt. Im direkten Vergleich zum Fall aus Abb. 4.2c, bei dem mit dem gleichen Stellwert aber einer verminderten Schaltdifferenz gearbeitet wurde, ist die Schalthäufigkeit geringer, wobei die Heizzeit innerhalb eines Schaltzyklus allerdings etwas länger ausfällt.
Die geringsten Temperaturschwankungen sind in Abb. 4.2d zu beobachten. Grund hierfür ist neben der kleinen Schaltdifferenz auch der verminderte Wert für die Stellgröße im eingeschalteten Zustand auf 75 % der maximalen Heizleistung. Auch der Temperaturmittelwert liegt bei dieser Messreihe am nächsten am Sollwert.

genauere Analyse der Stellgrößenverläufe: Die Stellgrößenverläufe erinnern an die Illustration zur Pulsweitenmodulation (vgl. Abb. 2.4). In Analogie zum Tastgrad kann über das Verhältnis aus Heizzeit t_{h} zur Periodendauer eines Schaltzyklus T_{z} ein mittlerer Wert für die Stellgröße $\overline{y}$ berechnet werden:

$$\overline{y} = y_{\max} \cdot \frac{t_{\mathrm{h}}}{T_{\mathrm{z}}} \tag{4.1}$$

In den Tabellen 4.1, 4.2 und 4.3 sind die genauen Schaltzeitpunkte aufgelistet, welche den Datensätzen zu den drei Temperaturverläufen entnommen worden sind. Zu den angegebenen Zeiten wird dabei in den zugeordneten Zustand gewechselt. Entsprechend Gleichung 4.1 werden daraus Mittelwerte für die Stellgröße berechnet, wobei jeweils über drei aufeinanderfolgende Schaltzyklen gemittelt wird. Damit kann kleineren Störungen, welche während der Messzeit aufgetreten sein können, etwas Rechnung getragen werden.

Tabelle 4.1: *Schaltzeiten zu Abb. 4.2b*

Schaltzeit in s	304	370	473	538	640	705	808
Schaltzustand	an	aus	an	aus	an	aus	an

$$\overline{y} = 100\,\% \cdot \left(\frac{370\,\mathrm{s} - 304\,\mathrm{s}}{472\,\mathrm{s} - 304\,\mathrm{s}} + \frac{538\,\mathrm{s} - 473\,\mathrm{s}}{639\,\mathrm{s} - 473\,\mathrm{s}} + \frac{705\,\mathrm{s} - 640\,\mathrm{s}}{807\,\mathrm{s} - 640\,\mathrm{s}} \right) \div 3 = 39{,}1\,\%$$

Tabelle 4.2: *Schaltzeiten zu Abb. 4.2c*

Schaltzeit in s	286	340	416	469	545	598	674
Schaltzustand	an	aus	an	aus	an	aus	an

$$\overline{y} = 100\,\% \cdot \left(\frac{340\,\mathrm{s} - 286\,\mathrm{s}}{415\,\mathrm{s} - 286\,\mathrm{s}} + \frac{469\,\mathrm{s} - 416\,\mathrm{s}}{544\,\mathrm{s} - 416\,\mathrm{s}} + \frac{598\,\mathrm{s} - 545\,\mathrm{s}}{673\,\mathrm{s} - 545\,\mathrm{s}} \right) \div 3 = 41{,}6\,\%$$

Tabelle 4.3: *Schaltzeiten zu Abb. 4.2d*

Schaltzeit in s	350	425	493	567	635	709	777
Schaltzustand	an	aus	an	aus	an	aus	an

$$\overline{y} = \quad 75\,\% \cdot \left(\frac{425\,\mathrm{s} - 350\,\mathrm{s}}{492\,\mathrm{s} - 350\,\mathrm{s}} + \frac{567\,\mathrm{s} - 493\,\mathrm{s}}{634\,\mathrm{s} - 493\,\mathrm{s}} + \frac{709\,\mathrm{s} - 635\,\mathrm{s}}{776\,\mathrm{s} - 635\,\mathrm{s}} \right) \div 3 = 39{,}4\,\%$$

Die über die Rechnungen erhaltenen Ergebnisse können mit der Stellgrößenänderung Δy verglichen werden, welche nach Gleichung 3.1 mithilfe des in Abschnitt 3.3.1 ermittelten Übertragungsbeiwertes von $K_\mathrm{s} = 0{,}76$ der Temperaturregelstrecke bestimmt werden kann. Zur Umrechnung der Regelgrößenänderung Δx, welche sich aus dem Temperaturunterschied zwischen dem Sollwert von 50 °C und der Raumtemperatur von 20 °C ergibt, in Prozent wird hierbei wieder ein Messumfang von 100 °C zugrunde gelegt:

$$\Delta y = \frac{\Delta x}{K_\mathrm{s}} = \frac{100\,\% \cdot (50\,^\circ\mathrm{C} - 20\,^\circ\mathrm{C})/100\,^\circ\mathrm{C}}{0{,}76} = 39{,}5\,\%$$

Zwei der drei ermittelten mittleren Stellgrößen zeigen eine gute Übereinstimmung mit dem über den Übertragungsbeiwert berechneten Ergebnis. Lediglich das Ergebnis aus der zweiten Messreihe mit den Parametern $y_\mathrm{max} = 100\,\%$ und $x_\mathrm{sd} = \pm 0{,}25\,^\circ\mathrm{C}$ liegt mit $\overline{y} = 41{,}6\,\%$ etwas deutlicher über dem Wert von $\Delta y = 39{,}5\,\%$. Eine mögliche Erklärung für diese Abweichung könnte in einer (größeren) Störung zu suchen sein, welche eventuell während der Datenaufnahme aufgetreten ist. Beispielsweise könnte ein kurzzeitig geöffnetes Fenster zu erhöhten Wärmeverlusten an die Umgebung geführt haben, was durch eine erhöhte Heizleistung kompensiert werden muss.

4.2. Dreipunktregelung mit Hysterese

Es soll nun eine Dreipunkttemperaturregelung mit zwei Hystereseschleifen und einer Totzone realisiert werden. Abbildung 4.3 zeigt einen möglichen Verlauf der Stellwerte y in Abhängigkeit von der Temperatur als Regelgröße x. Neben dem eigentlichen Sollwert w wird ein weiterer Regelgrößenwert für eine Hilfsstufe h festgelegt. Jeder Hystereseschleife wird eine eigene Schaltdifferenz ($x_{\mathrm{sd},h}$ und $x_{\mathrm{sd},w}$) zugeordnet[1].

Die Schaltzustände und Schaltpunkte sollten für das in dieser Arbeit verwendete Regelstreckenmodell für die Dreipunkttemperaturregelung so gewählt werden, dass ausgehend von Raumtemperatur u. a. ein schnelles Aufheizen in Richtung Sollwert gewährleistet wird. Mit reduzierter Stellgröße soll dann

[1] Auch hier müssen die Schaltpunkte nicht zwingend symmetrisch angeordnet werden.

in der Umgebung um den Sollwert eine präzisere Regelung erreicht werden. Die Totzone soll ein zu starkes Überschwingen der Temperatur über den Sollwert hinaus verhindern. Die optimale Größe dieser Zone sowie die optimale Lage der Schaltpunkte werden u. a. durch die Verzugs- und Ausgleichszeit der Regelstrecke beeinflusst und können experimentell untersucht werden.

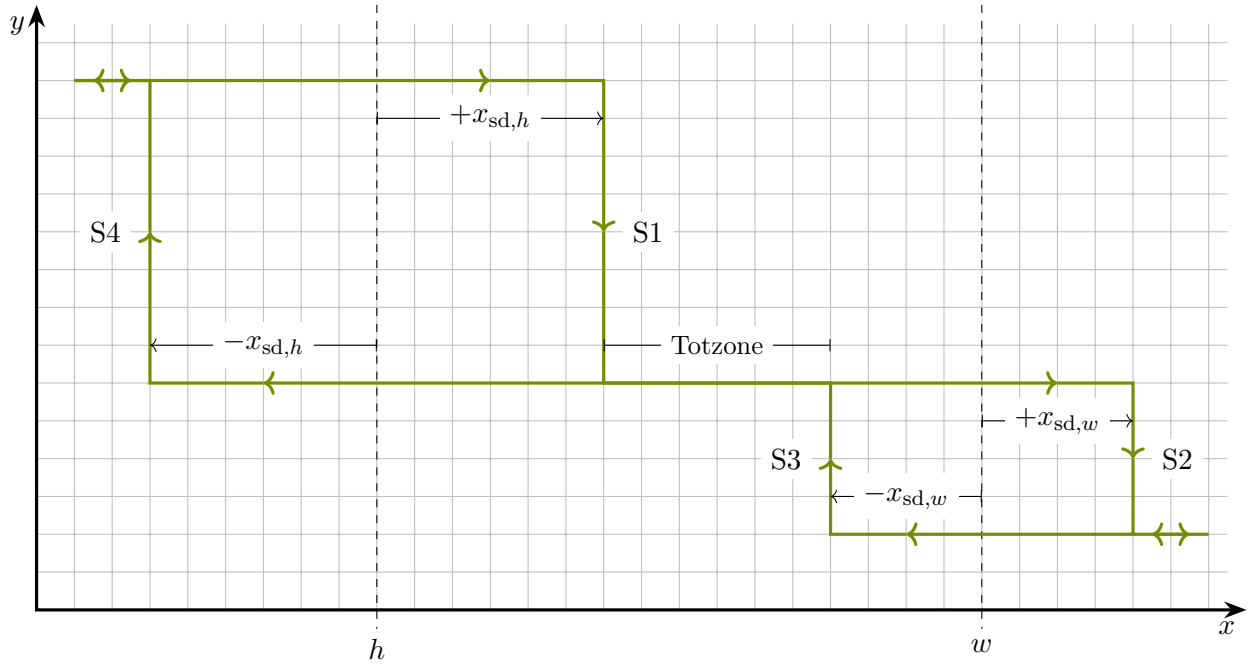


Abbildung 4.3: *Beispiel einer Schaltkurve eines Dreipunktreglers mit zwei Hystereseschleifen und einer Totzone*

4.2.1. Implementierung einer Dreipunkttemperaturregelung

Code 4.1 kann mit verhältnismäßig wenig Aufwand zu einer Dreipunktregelung (Code 4.2) erweitert bzw. angepasst werden. Neben einer Konstanten für den Sollwert `w` wird in Zeile 13 eine weitere Konstante `h` für die Lage der Hilfsstufe festgelegt und dieser ein Wert zugewiesen. Zudem werden in den Zeilen 14 und 15 Konstanten `xSDw` und `xSDh` für die Schaltdifferenzen der beiden Hystereseschleifen mit entsprechenden Wertzuweisungen festgelegt. In den Zeilen 17, 18 und 19 werden Konstanten mit Wertzuweisungen für die drei Schaltzustände (`yMin`, `yMid` und `yMax`) deklariert. Die Variable `y` wird darauf folgend in Zeile 20 deklariert. Sie erhält den Wert der Konstanten `yMid` als Startwert.
Aus Abbildung 4.3 können vier Schaltbedingungen S1 bis S4 abgeleitet werden. Diese sind innerhalb der `loop()`-Funktion in den Zeilen 35 bis 38 implementiert und lesen sich wie folgt:
[S1] Besitzt die Stellgröße `y` aktuell den Wert der Konstanten `yMax` und liegt der momentane Wert der Regelgröße `x` über dem Ergebnis der Rechnung `h + xSDh`, dann wird der Stellgröße der Wert der Konstanten `yMid` neu zugewiesen.
[S2] Besitzt die Stellgröße `y` aktuell den Wert der Konstanten `yMid` und liegt der momentane Wert der Regelgröße `x` über dem Ergebnis der Rechnung `w + xSDw`, dann wird der Stellgröße der Wert der Konstanten `yMin` neu zugewiesen.
[S3] Besitzt die Stellgröße `y` aktuell den Wert der Konstanten `yMin` und liegt der momentane Wert der Regelgröße `x` unter dem Ergebnis der Rechnung `w - xSDw`, dann wird der Stellgröße der Wert der Konstanten `yMid` neu zugewiesen.
[S4] Besitzt die Stellgröße `y` aktuell den Wert der Konstanten `yMid` und liegt der momentane Wert der Regelgröße `x` unter dem Ergebnis der Rechnung `h - xSDh`, dann wird der Stellgröße der Wert der Konstanten `yMax` neu zugewiesen.

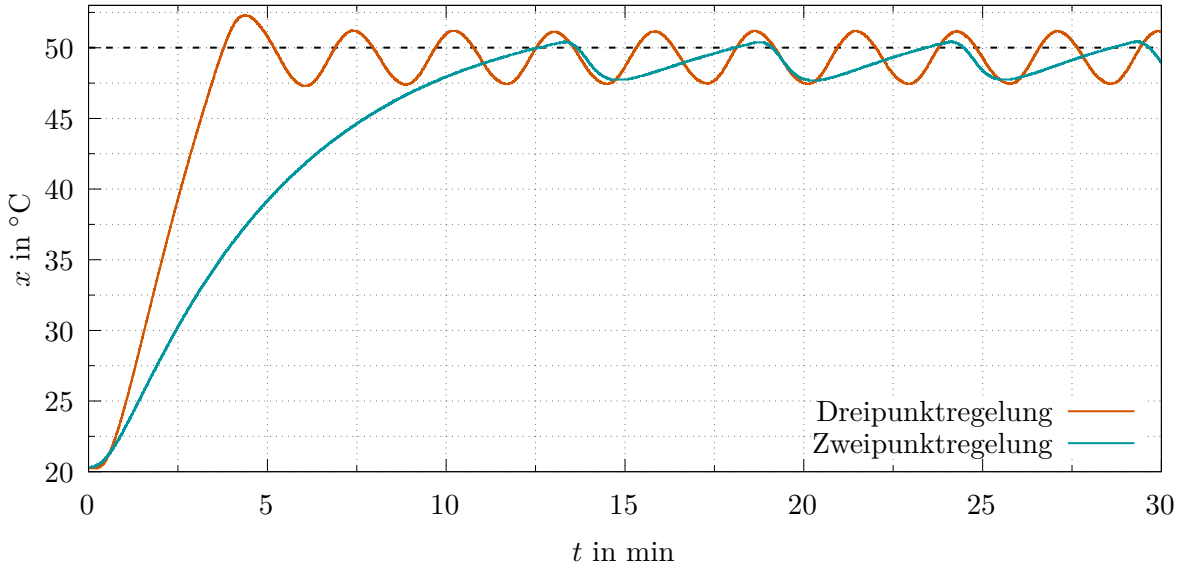
Wie auch schon bei der Implementierung der Zweipunkttemperaturregelung behält in allen anderen Fällen die Stellgröße ihren aktuellen Wert bei, bis sich in einem der folgenden Schleifendurchläufe die Temperatur soweit verändert hat, dass eine der vier Bedingungen erfüllt wird.

Code 4.2.: *Dreipunktregelung mit zwei Hystereseschleifen*

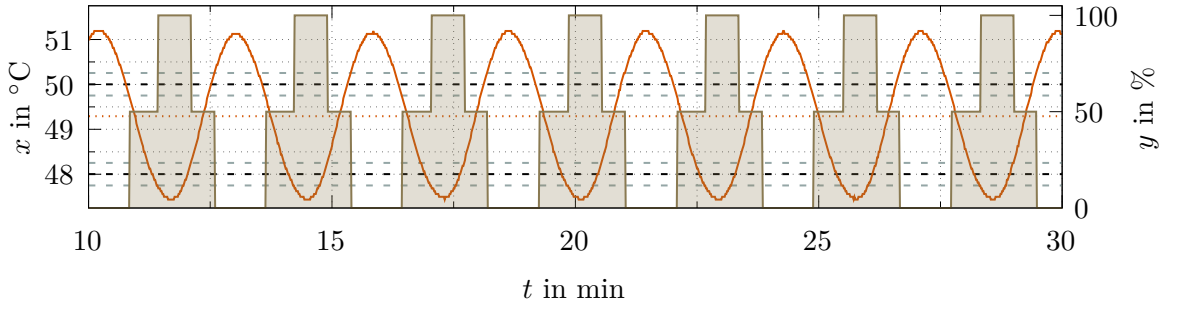
```
#include <OneWire.h>
#include <DallasTemperature.h>
const int sensorPin = 2;
OneWire oneWire(sensorPin);
DallasTemperature sensors(&oneWire);
const int gatePin = 5;      // Pin zum Ansteuern des Transistor-Gates

float t = 0.0;              // aktuelle Zeit in s
const float dt = 1.0;       // Zeitintervall in s

float x = 0.0;              // aktuelle Temperatur in °C
const float w = 50.0;       // Sollwert in °C
const float h = 48.0;       // Hilfsstufe in °C
const float xSDw = 0.25;    // Schaltweite um den Sollwert in °C
const float xSDh = 0.25;    // Schaltweite um die Hilfsstufe in °C

const int yMin = 0;         // minimale Stellgroesse in Prozent
const int yMid = 50;        // mittlere Stellgroesse in Prozent
const int yMax = 100;       // maximale Stellgroesse in Prozent
int y = yMid;               // Stellgroesse in Prozent

void setup() {
  Serial.begin(9600);
  sensors.begin();
  pinMode(gatePin, OUTPUT);
}

void loop() {
  if (millis() / 1000.0 - t >= dt) {
    t = millis() / 1000.0;
    sensors.requestTemperatures();
    x = sensors.getTempCByIndex(0);

    if (y == yMax and x > h + xSDh) {y = yMid;}       // Schaltbedingung S1
    else if (y == yMid and x > w + xSDw) {y = yMin;}  // Schaltbedingung S2
    else if (y == yMin and x < w - xSDw) {y = yMid;}  // Schaltbedingung S3
    else if (y == yMid and x < h - xSDh) {y = yMax;}  // Schaltbedingung S4

    analogWrite(gatePin, 255 * y / 100.0);

    Serial.print(t, 1);
    Serial.print('\t');
    Serial.print(x, 3);
    Serial.print('\t');
    Serial.println(y);
  }
}
```

4.2.2. Beispieldaten und Interpretation

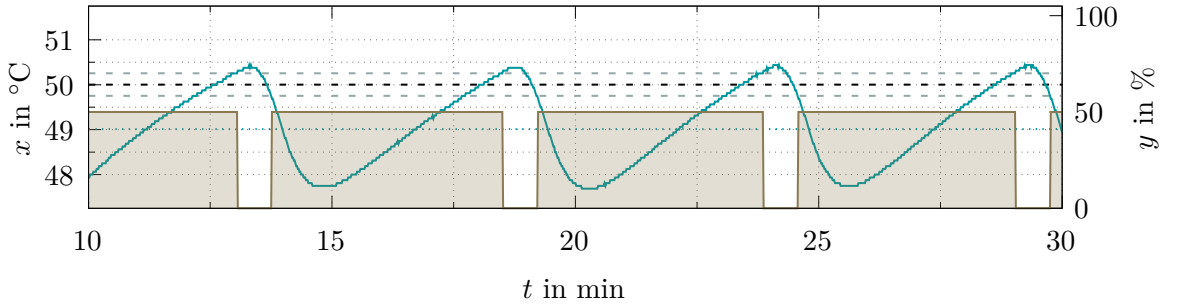
Abbildung 4.4 zeigt den Vergleich einer Dreipunktregelung mit den Schaltzuständen $y_{\min} = 0\,\%$, $y_{\text{mid}} = 50\,\%$ und $y_{\max} = 100\,\%$ sowie den Parametern $w = 50\,°\text{C}$, $h = 48\,°\text{C}$ und $x_{\text{sd},w} = x_{\text{sd},h} = 0{,}25\,°\text{C}$ mit einer Zweipunkttemperaturregelung, bei welcher die Einstellungen $y_{\min} = 0\,\%$, $y_{\max} = 50\,\%$, $w = 50\,°\text{C}$ und $x_{\text{sd}} = 0{,}25\,°\text{C}$ vorgenommen worden sind. Somit sind die Hystereseschleifen um den Sollwert identisch parametrisiert und die Dreipunktregelung unterscheidet sich nur durch die Hilfsstufe. Es ist klar zu erkennen, dass bei der Zweipunkttemperaturregelung durch die verminderte Stellgröße und damit auch geringere Heizleistung der Aufheizvorgang ausgehend von Raumtemperatur wesentlich länger dauert. Erst nach ca. 12,5 min erreicht die Temperatur das erste Mal den Sollwert. Der Verlauf ist bis dahin mit jenem der Sprungantwort aus Abbildung 3.5 vergleichbar, welcher ebenfalls mit $y = 50\,\%$ aufgenommen worden ist.



(a) *Temperaturverläufe im direkten Vergleich*



(b) *Verlauf von Temperatur und Stellgröße bei der Dreipunktregelung*



(c) *Verlauf von Temperatur und Stellgröße bei der Zweipunktregelung*

Abbildung 4.4: *Vergleich einer Dreipunktregelung mit einer Zweipunktregelung*

Ein weiterer Unterschied besteht in der Periodendauer des oszillierenden Temperaturverlaufs, nachdem die Solltemperatur zum ersten Mal erreicht worden ist. Hier zeigt die Dreipunktregelung bei den gewählten Parametern einen deutlich kleineren Zeitwert. Die Ausschnitte in den Abbildungen 4.4b und 4.4c ermöglichen dazu einen detaillierteren Vergleich. Der Sollwert und die Schaltgrenzen sowie die Lage der Hilfsstufe bei der Dreipunktregelung sind durch gestrichelte horizontale Linien eingetragen. Die in der Farbe des Temperaturverlaufs gezeichnete, gepunktete Linie stellt den Mittelwert für die Temperatur nach dem Aufheizvorgang dar, welcher jeweils über die Messwerte aus drei aufeinanderfolgenden Schaltzyklen berechnet worden ist.
Der Verlauf der Zweipunktregelung zeigt eine verhältnismäßig kleine Überschwingweite über den Sollwert hinaus. Nach Abschaltung des Heizelements fällt die Temperatur hingegen stärker ab um dann nach Anschalten des Heizelements wieder langsam zu steigen. Dieses Verhalten sowie die lange Periodendauer lässt sich damit erklären, dass der Stellwert von 50 % nach der Sprungantwort aus Abb. 3.5 bei dauerhaftem Heizen maximal auf einen Temperaturwert von ca. 55 °C hinauslaufen könnte und die Wärmeverluste in diesem Temperaturbereich dafür sorgen, dass die Steigung der Temperaturkurve schon stark abgeflacht ist. Der Mittelwert der Temperatur liegt im eingeschwungenen Zustand knapp über 49,0 °C und damit deutlich unterhalb des Sollwerts.
Bei der Dreipunkt- ist im Vergleich zur Zweipunkttemperaturregelung die Schalthäufigkeit deutlich und die Überschwingweite über den Sollwert hinaus etwas größer. Die mittlere Temperatur liegt im eingeschwungenen Zustand bei ca. 49,3 °C und damit etwas näher am Sollwert.
Durch Anpassung der Hilfsstufe, der Schaltdifferenzen und der Stellwerte könnte sicher noch eine Optimierung des Regelverhaltens erzielt werden. In Anhang C.1 wird am Beispiel eines Peltier-Elements eine weitere unstetige Regelung mit drei verschiedenen Schaltzuständen gezeigt.

Darstellung des gemessenen Hystereseverlaufs: Aus den aufgezeichneten Werten lässt sich auch noch der Hystereseverlauf rekonstruieren. Dies ist am Beispiel der Dreipunktregelung in Abbildung 4.5 gezeigt. Im Vergleich zum idealen, theoretischen Verlauf aus Abb. 4.3 kann man erkennen, dass die Schaltflanken nicht senkrecht steigen bzw. fallen. Die Ursache hierfür ist in der begrenzten Auflösung des Temperatursensors und der damit verbundenen schrittweisen Aufzeichnung zu suchen. Innerhalb der Abtastrate von einer Sekunde kann sich der Temperaturwert schon um einen oder sogar mehrere Temperaturschritte verändert haben.

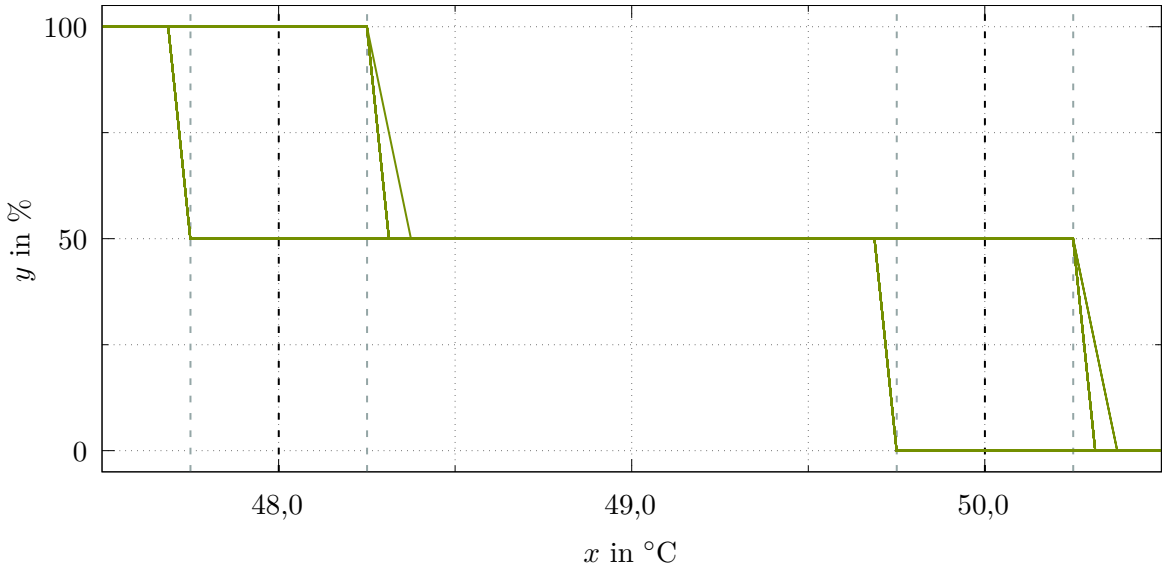


Abbildung 4.5: *gemessener Verlauf der Hystereseschleifen einer Dreipunktregelung*

4.3. Zusammenfassung und Anregungen für den Unterricht

Unstetige Regelungen lassen sich durch die Formulierung von Schaltbedingungen, welche sich aus Hystereseverläufen ableiten lassen, mit verhältnismäßig wenig Aufwand implementieren und testen. Da hierzu noch keine komplexeren (mathematischen) Modelle verstanden werden müssen, bieten sie sich als Erstes zur Umsetzung von Regelalgorithmen an. Dennoch liefert der hierüber geschlossene Temperaturregelkreis vielfältige Möglichkeiten zum Experimentieren. Eine entsprechende Lernsituation könnte durch die

Implementierung und Untersuchung der Eignung von unstetigen Temperaturregelungen

umgesetzt werden. Dabei können durch die Lehrkraft Anforderungen an die Genauigkeit und ggf. Geschwindigkeit des Regelvorgangs gestellt werden, welche von den Lernenden erreicht werden sollen. Es wäre auch denkbar, dass hierbei einzelne Kleingruppen bei der Findung von optimalen Regelparametern in einen Wettstreit treten. Eine im Vorfeld stattgefundene qualitative oder auch quantitative Untersuchung des zeitlichen Regelstreckenverhaltens mithilfe des Sprungantwortverfahrens kann hierzu eventuell auch noch nützliche Anhaltspunkte liefern.
Das Erkennen der Grenzen unstetiger Regelungen kann dann Motivation zur Auseinandersetzung mit komplexeren, stetigen Regelungen sein, um noch bessere Ergebnisse zu erzielen und die im Regelkreis steckenden Möglichkeiten optimal auszunutzen. Hierauf wird nun im folgenden Abschnitt 5 eingegangen.

4.4. Ideen für Aufgabenstellungen

- Untersuchen und erklären Sie den Zusammenhand zwischen Schaltdifferenz, Schwankungsbreite und Schalthäufigkeit bei Zweipunktregelungen.
- Bestimmen Sie die mittlere Stellgröße bei Zweipunkttemperaturregelungen mit gleichem Sollwert aber unterschiedlichen Schaltdifferenzen, nachdem der Aufheizvorgang beendet ist und sich ein schwingender Temperaturverlauf eingestellt hat.
- Untersuchen Sie, ob die Schaltdifferenz die mittlere Stellgröße bei unterschiedlich hohen Sollwerten beeinflusst.
- Eine Lüfter soll als Stellgerät dienen, um mithilfe von unstetigen Regelungen die Temperatur an einem sich erwärmenden Bauteil (z. B. einem Heizwiderstand) zu kontrollieren. Entwickeln Sie Hysteresekurven für entsprechende Zweipunkt- und Dreipunkttemperaturregelungen und leiten Sie aus diesen Codezeilen zur Umsetzung der Schaltbedingungen ab.
- Berechnen Sie für eine Dreipunkttemperaturregelung (nach Beendigung des Aufheizvorgangs) die mittlere Stellgröße im schwingenden Zustand.
- Optimieren Sie den Verlauf einer Dreipunkttemperaturregelung hinsichtlich eines Kompromisses aus Schwankungsbreite, Schalthäufigkeit und Geschwindigkeit des Aufheizvorgangs, indem Sie systematisch die Parameter der Schaltdifferenzen, der Lage der Hilfsstufe und der Totzone variieren.

5. Umsetzung von stetigen PID-Regelungen

Stetige Regler haben den Vorteil, dass sie innerhalb des Stellbereichs von 0 % bis 100 % jeden beliebigen Stellwert ausgeben können. Hierdurch kann eine genauere und schnellere Regelung erfolgen, sofern diese Größe clever bestimmt und über einen geeigneten Steller auf das Stellglied übertragen wird. Dabei kann es geschickt sein, nicht nur über die gegenwärtige Regeldifferenz sondern auch anhand des vergangenen Verlaufs und des zukünftigen Trends der Regeldifferenz den Stellwert zu bilden.
In Abschnitt 5.1 wird dies am Beispiel von Reglern mit proportionalen, integralen und differentialen Anteilen (kurz PID-Regelung) mathematisch diskutiert. Anschließend erfolgt in Abschnitt 5.2 eine anschauliche Betrachtung anhand eines mechanischen Modells und in Abschnitt 5.3 eine zunächst allgemeine Implementierung in Form eines Arduino-Codes, bevor in Abschnitt 5.4 eine Anpassung zur Temperaturregelung und in Abschnitt 5.5 eine Umsetzung mit Messungen und Auswertungen im Temperaturregelkreis erfolgt.

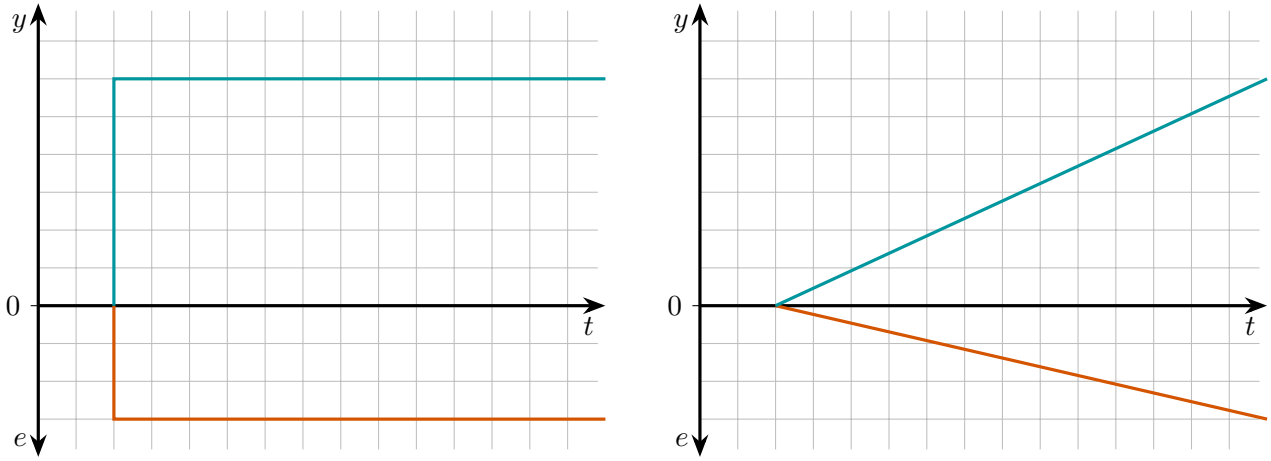
5.1. Mathematische Beschreibung der einzelnen Anteile

5.1.1. Proportionalanteil und P-Regelung

Eine erste Idee zur Realisierung eines stetigen Reglers besteht darin, dass die Stellgröße y umso größer sein sollte, je höher die Regeldifferenz e ist. Bei einem Proportionalregler (P-Regler) wird die Stellgröße y_p nach Gleichung 5.1 berechnet. Die Proportionalitätskonstante K_p wird dabei als Proportionalbeiwert bezeichnet. Gibt man die Stellgröße und die Regeldifferenz in Prozent an, so ist diese Konstante dimensionslos. Zur Umrechnung der Regeldifferenz in Prozent kann man sich z. B. auf die Größe des Messumfangs beziehen. Man spricht dabei von einer Normierung.

$$y_\mathrm{p}(t) = K_\mathrm{p} \cdot e(t) \tag{5.1}$$

Die Diagramme in Abbildung 5.1 zeigen das Zeitverhalten eines Reglers mit reinem P-Verhalten. Dabei ist in Abb. 5.1a die Sprungantwort dargestellt, welche das statische Verhalten bei konstanter Regeldifferenz beschreibt. Abb. 5.1b zeigt die Anstiegsantwort. Damit wird das dynamische Verhalten bei gleichmäßig steigender Regeldifferenz visualisiert. Beide Diagramme zeigen, dass der P-Regler sofort auf das Auftreten einer Regeldifferenz mit einer proportionalen Stellgröße reagiert.



(a) *Sprungantwort, statisches Verhalten* (b) *Anstiegsantwort, dynamisches Verhalten*

Abbildung 5.1: *Zeitverhalten eines P-Reglers*

Eine weitere Kenngröße einer P-Regelung ist der Proportionalbereich X_p. Dieser gibt den Bereich der Regelgröße x an, in welchem die Stellgröße y_p proportional zur Regeldifferenz e ist. Er hängt

über den vollen Stellbereich $Y = 100\,\%$ mit dem Proportionalbeiwert K_p zusammen. Außerhalb des Proportionalitätsbereichs kommt es zu Übersteuerung, d. h. die Stellgröße befindet sich an einer der Grenzen $y = 0\,\%$ oder $y = 100\,\%$ und kann sich dann nicht mehr weiter ändern.

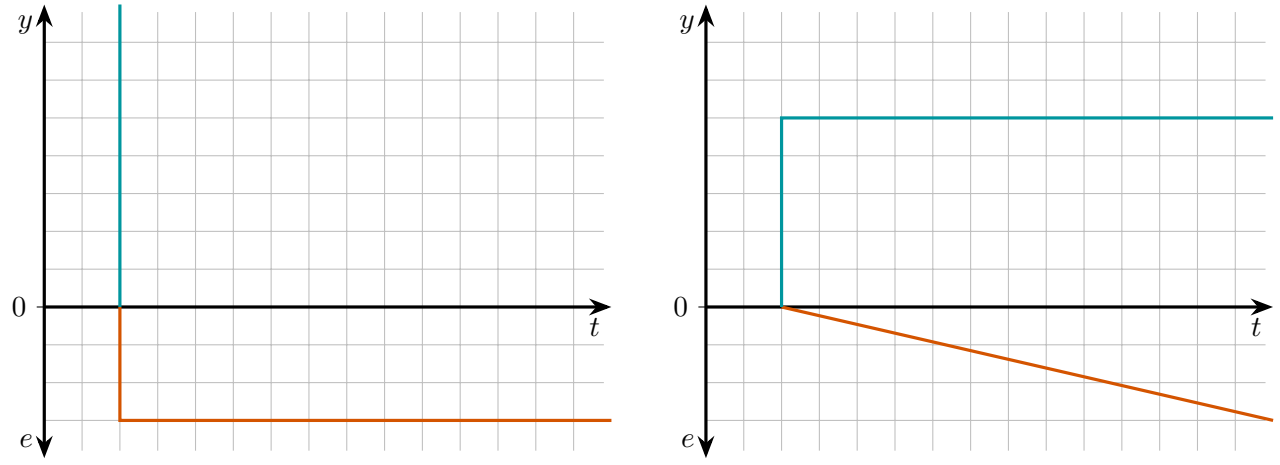
$$K_\mathrm{p} = \frac{Y}{X_\mathrm{p}} \tag{5.2}$$

5.1.2. Differentialanteil

Der Differentialanteil produziert eine Stellgröße, welche proportional zur Änderungsgeschwindigkeit $\mathrm{d}e/\mathrm{d}t = \dot{e}$ der Regeldifferenz ist. Diese wird nach Gleichung 5.3 berechnet. Da die Regeldifferenz nach $e = w - r$ bzw. $e = w - x$ bestimmt wird, bedeutet dies, dass die Stellgröße umso größer ist, je schneller die Regelgröße x auf den Sollwert w zu oder von ihm wegläuft. Die Proportionalitätskonstante K_d wird als Differenzierbeiwert bezeichnet. Gibt man die Stellgröße und die Regeldifferenz (im Rahmen einer Normierung) in Prozent an, so wird der Differenzierbeiwert in einer Zeiteinheit (z. B. in s) bestimmt.

$$y_\mathrm{d}(t) = K_\mathrm{d} \cdot \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{5.3}$$

Das Zeitverhalten des D-Anteils kann in den Diagrammen aus Abbildung 5.2 betrachtet werden. Die Sprungantwort in Abb. 5.2a zeigt, dass zum Zeitpunkt des Eintretens des Sprungs der Regeldifferenz die Stellgröße aufgrund der unendlichen Steigung ($\mathrm{d}e/\mathrm{d}t = \infty$) sofort in die Übersteuerung ($y_\mathrm{d} = 100\,\%$) springt, um danach bei konstant bleibender Regeldifferenz ($\mathrm{d}e/\mathrm{d}t = 0$) gleich wieder auf null zu sinken. Man spricht hier von einem Nadelimpuls. Die Anstiegsantwort in Abb. 5.2b beschreibt das dynamische Verhalten des D-Anteils. Bei gleichmäßig ansteigender Regeldifferenz ($\mathrm{d}e/\mathrm{d}t = \text{konstant}$) nimmt die Stellgröße einen konstanten Wert an. Durch die Reaktion auf die Veränderung der Regeldifferenz $e = w - x$ bewertet der D-Anteil die zukünftige Entwicklung der Regelgröße x.



(a) *Sprungantwort* (b) *Anstiegsantwort*

Abbildung 5.2: *Zeitverhalten des D-Anteils*

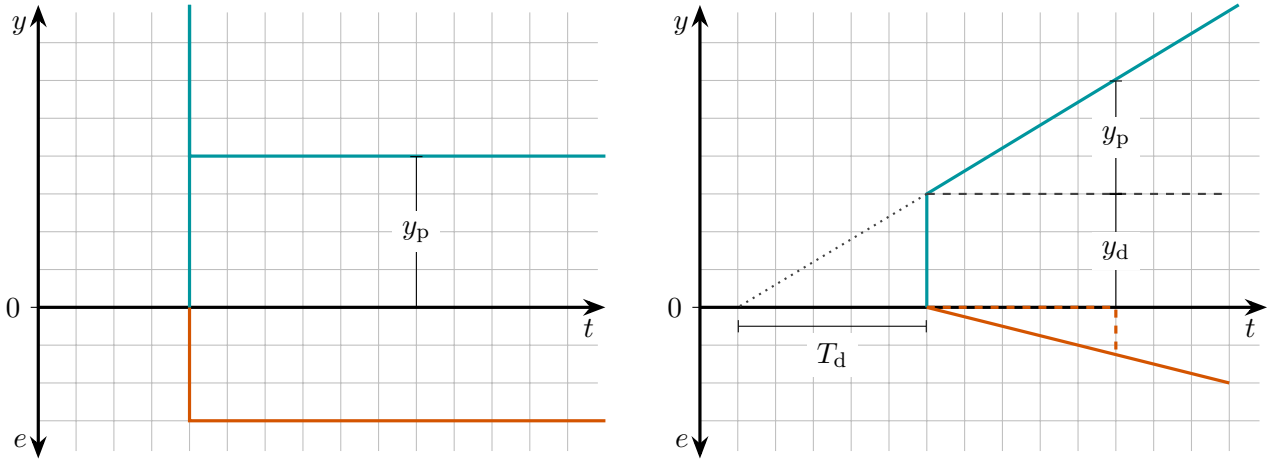
Da der D-Anteil nur auf Änderungen der Regeldifferenz reagiert, ist eine reine D-Regelung nicht sinnvoll bzw. möglich. Bei konstanter Regeldifferenz ($\mathrm{d}e/\mathrm{d}t = 0$) wird keine Stellgröße ausgegeben ($y_\mathrm{d} = 0$), welche der Regeldifferenz entgegenwirken und somit eine Angleichung an den Sollwert herbeiführen könnte.

5.1.3. PD-Regelung

Kombiniert man den P- mit dem D-Anteil, indem man beide addiert, so erhält man eine PD-Regelung, welche nach Gleichung 5.4 beschrieben werden kann.

$$y(t) = y_\mathrm{p} + y_\mathrm{d} = K_\mathrm{p} \cdot e(t) + K_\mathrm{d} \cdot \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{5.4}$$

Das Zeitverhalten eines PD-Reglers ist in den Diagrammen aus Abbildung 5.3 gezeigt. In der Sprungantwort (Abb. 5.3a) erkennt man die D-Wirkung nur am auftretenden Nadelimpuls. Die Anstiegsantwort (Abb. 5.3b) für die Stellgröße besteht aus einer konstanten Stufe durch den D-Anteil y_d, welche mit einem gleichmäßigen Anstieg durch den P-Anteil y_p überlagert ist.



(a) *Sprungantwort* (b) *Anstiegsantwort*

Abbildung 5.3: *Zeitverhalten eines PD-Reglers*

Anstelle des Differenzierbeiwertes K_d kann auch die Vorhaltzeit T_d zur Parametrisierung einer PD-Regelung herangezogen werden. Diese gibt die Zeit an, welche der P-Anteil in der Anstiegsantwort benötigt, um dieselbe Stellgrößenänderung wie der D-Anteil hervorzurufen. Geht man von einer konstant steigenden Regeldifferenz der Form $e(t) = m \cdot t$ aus, so folgt:

$$\begin{aligned} y_\mathrm{p}(T_\mathrm{d}) &= y_\mathrm{d} \\ K_\mathrm{p} \cdot e(T_\mathrm{d}) &= K_\mathrm{d} \cdot \frac{\mathrm{d}e(t)}{\mathrm{d}t} \\ K_\mathrm{p} \cdot m \cdot T_\mathrm{d} &= K_\mathrm{d} \cdot m \\ T_\mathrm{d} &= \frac{K_\mathrm{d}}{K_\mathrm{p}} \end{aligned} \tag{5.5}$$

Gleichung 5.5 zeigt, dass die Vorhaltzeit T_d durch das Verhältnis von K_d zu K_p bestimmt wird. Damit ergibt sich durch Einsetzen in Gleichung 5.4 und Umformung Gleichung 5.6 für die Zeitabhängigkeit der Stellgröße einer PD-Regelung.

$$\begin{aligned} y(t) &= K_\mathrm{p} \cdot e(t) + K_\mathrm{d} \cdot \frac{\mathrm{d}e(t)}{\mathrm{d}t} \\ &= K_\mathrm{p} \cdot \left(e(t) + \frac{K_\mathrm{d}}{K_\mathrm{p}} \cdot \frac{\mathrm{d}e(t)}{\mathrm{d}t} \right) \\ &= K_\mathrm{p} \cdot \left(e(t) + T_\mathrm{d} \cdot \frac{\mathrm{d}e(t)}{\mathrm{d}t} \right) \end{aligned} \tag{5.6}$$

Nutzt man T_d anstelle von K_d zur Parametrisierung einer PD-Regelung, so kann man die D-Wirkung vergrößern, indem man T_d erhöht. Eine Veränderung von K_p wirkt sich bei konstanter Vorhaltzeit allerdings gleichermaßen auf die P- und die D-Wirkung aus. Will man beispielsweise den P-Anteil verdoppeln und den D-Anteil unverändert belassen, so muss man K_p verdoppeln und T_d halbieren.

5.1.4. Integralanteil und I-Regelung

Der Integralanteil sorgt nach Gleichung 5.7 für eine zur Regeldifferenz e proportionale Änderungsgeschwindigkeit $\frac{\mathrm{d}y_\mathrm{i}}{\mathrm{d}t}$ der Stellgröße. Die Proportionalitätskonstante K_i wird als Integrierbeiwert bezeichnet. Gibt man die Stellgröße und die Regeldifferenz in Prozent an, so wird der Integrierbeiwert in einer inversen Zeiteinheit (z. B. in $\frac{1}{\mathrm{s}}$) bestimmt. Durch Umformung erhält man Gleichung 5.8, mit welcher sich die Stellgrößenänderung $\mathrm{d}y_\mathrm{i}$ innerhalb eines (infinitesimal) kleinen Zeitintervalls $\mathrm{d}t$ berechnen lässt. Bildet man von dieser Gleichung das Zeitintegral über das Zeitintervall $[0, t]$, so ergibt sich über Gleichung 5.9 die Stellgröße, welche sich bis zum Zeitpunkt t gebildet hat. Diese Gleichung erklärt auch den Namen Integralanteil bzw. Integralregelung. Da der Integralanteil zur Bildung der Stellgröße $y_\mathrm{i}(t)$ zu einem bestimmten Zeitpunkt τ den gesamten Verlauf der Regeldifferenz e im integrierten Zeitintervall $[0, t]$ berücksichtigt, kann man sagen, dass der I-Anteil die Vergangenheit der Regeldifferenz bewertet.

$$\frac{\mathrm{d}y_\mathrm{i}(t)}{\mathrm{d}t} = K_\mathrm{i} \cdot e(t) \tag{5.7}$$

$$\mathrm{d}y_\mathrm{i}(t) = K_\mathrm{i} \cdot e(t) \cdot \mathrm{d}t \tag{5.8}$$

$$y_\mathrm{i}(t) = K_\mathrm{i} \int_0^t e(\tau)\mathrm{d}\tau \tag{5.9}$$

Abbildung 5.4 zeigt die Sprungantwort eines I-Reglers. Dabei kann die Änderungsgeschwindigkeit der Stellgröße über ein Steigungsdreieck bestimmt werden. Man erkennt, dass sich die Stellgröße allmählich aufbaut. Da hier das Zeitintegral über die Regeldifferenz eine Rechteckfläche darstellt, erhält man nach doppelter Integrationszeit auch die doppelte Integralfläche und damit auch den doppelten Wert für die Stellgröße[1]. Eine Anstiegsantwort wird für einen I-Regler und im Folgenden auch für alle weiteren Regler mit I-Anteil nicht betrachtet. Sie würde in Form einer Parabel verlaufen. Bei einem Test eines I-Reglers mit dem Anstiegsantwortverfahren könnte man in einem entsprechenden Diagramm eine Abweichung von einer perfekten Parabelform nur schwer per Augenmaß erkennen, während die Güte des linearen Verlaufs in der Sprungantwort leicht zu beurteilen ist.

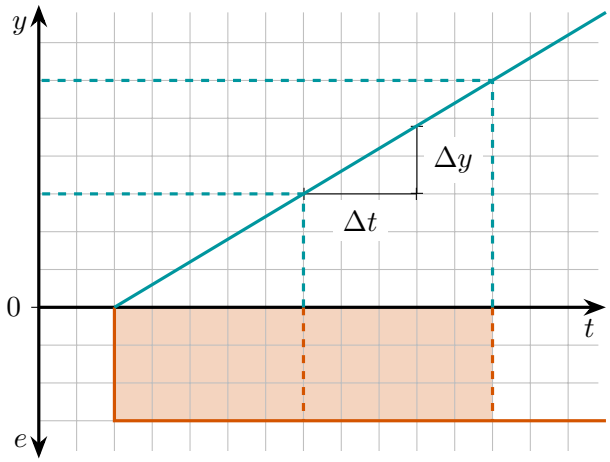


Abbildung 5.4: *Sprungantwort des I-Anteils*

[1] Eine Erklärung über die Bildung von Flächeninhalten findet sich auch in einer Beispielauswertung in Abbildung 5.14b in Abschnitt 5.5.4 auf Seite 50.

Ein weiterer Kennwert eines reinen I-Reglers ist die Integrierzeit T_I. Diese gibt die Zeitspanne an, welche die Stellgröße in der Sprungantwort des I-Regler benötigt, um denselben Wert wie der Eingangssprung der Regeldifferenz anzunehmen. Bei einem Sprung der Regeldifferenz von 100 % ist dies die Zeit, welche zum Durchlaufen des vollen Stellbereichs von 100 % benötigt wird. Aus Gleichung 5.8 folgt:

$$100\,\% = K_\mathrm{i} \cdot 100\,\% \cdot T_\mathrm{I} \qquad \Rightarrow \qquad T_\mathrm{I} = \frac{1}{K_\mathrm{i}} \tag{5.10}$$

Gleichung 5.10 zeigt, dass eine Vergrößerung des Integrierbeiwertes K_i dazu führt, dass die Integrierzeit T_I kleiner wird. Der Stellbereich wird dadurch schneller durchlaufen.

5.1.5. PI-Regelung

Kombiniert man den P- mit dem I-Anteil, indem man wiederum beide addiert, so erhält man eine PI-Regelung, welche nach Gleichung 5.11 beschrieben werden kann.

$$y(t) = y_\mathrm{p} + y_\mathrm{i} = K_\mathrm{p} \cdot e(t) + K_\mathrm{i} \int_0^t e(\tau)\mathrm{d}\tau \tag{5.11}$$

Abbildung 5.5 zeigt die Sprungantwort eines PI-Reglers.

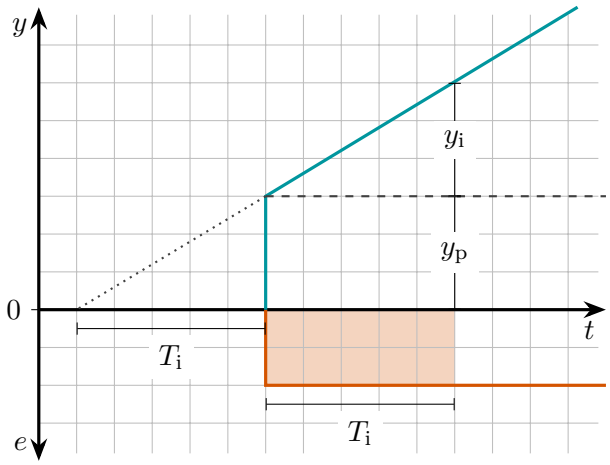


Abbildung 5.5: *Sprungantwort eines PI-Reglers*

Neben dem Intergrierbeiwert K_i kann eine PI-Regelung auch durch die Nachstellzeit T_i parametrisiert werden. Diese gibt die Zeit an, welche der I-Anteil in der Sprungantwort benötigt, um dieselbe Stellgröße y_i wie der P-Anteil hervorzurufen. Bei einer konstanten Regeldifferenz $e(t) = e$ berechnet sich das Integral im I-Anteil als Rechteckfläche. Es folgt somit Gleichung 5.12, in welcher der Zusammenhang zwischen Nachstellzeit, Proportionalbeiwert und Integrierbeiwert angegeben ist.

$$\begin{aligned} y_\mathrm{p} &= y_\mathrm{i}(T_\mathrm{i}) \\ K_\mathrm{p} \cdot e &= K_\mathrm{i} \cdot e \cdot T_\mathrm{i} \\ T_\mathrm{i} &= \frac{K_\mathrm{p}}{K_\mathrm{i}} \end{aligned} \tag{5.12}$$

Setzt man Gleichung 5.12 in Gleichung 5.11 ein, so erhält man nach Umformungen Gleichung 5.13, welche die Zeitabhängigkeit der Stellgröße einer PI-Regelung beschreibt. Um den I-Anteil zu erhöhen, muss die Nachstellzeit T_i verkleinert werden. Eine Veränderung von K_p wirkt sich bei konstanter Nachstellzeit gleichermaßen auf die P- und die I-Wirkung aus. Will man beispielsweise den P-Anteil verdoppeln und den I-Anteil unverändert belassen, so muss man sowohl K_p als auch T_i verdoppeln.

Um einen PI-Regler zu einem P-Regler neu zu konfigurieren, müsste man $T_\mathrm{i} = \infty$ einstellen.

$$\begin{aligned} y(t) &= K_\mathrm{p} \cdot e(t) + K_\mathrm{i} \int_0^t e(\tau)\mathrm{d}\tau \\ &= K_\mathrm{p} \cdot \left(e(t) + \frac{K_\mathrm{i}}{K_\mathrm{p}} \int_0^t e(\tau)\mathrm{d}\tau \right) \\ &= K_\mathrm{p} \cdot \left(e(t) + \frac{1}{T_\mathrm{i}} \int_0^t e(\tau)\mathrm{d}\tau \right) \end{aligned} \tag{5.13}$$

5.1.6. PID-Regelung

Natürlich kann man auch P-, I- und D-Anteil durch Addition kombinieren. Dadurch erhält man eine PID-Regelung, welche durch Gleichung 5.15 über die drei entsprechenden Beiwerte oder alternativ durch Gleichung 5.16 mithilfe des Proportionalbeiwertes, der Nachstellzeit und der Vorhaltzeit beschrieben werden kann.

$$y(t) = y_\mathrm{p} + y_\mathrm{i} + y_\mathrm{d} \tag{5.14}$$

$$= K_\mathrm{p} \cdot e(t) + K_\mathrm{i} \int_0^t e(\tau)\mathrm{d}\tau + K_\mathrm{d} \cdot \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{5.15}$$

$$= K_\mathrm{p} \cdot \left(e(t) + \frac{1}{T_\mathrm{i}} \int_0^t e(\tau)\mathrm{d}\tau + T_\mathrm{d} \cdot \frac{\mathrm{d}e(t)}{\mathrm{d}t} \right) \tag{5.16}$$

In Abbildung 5.6 ist die Sprungantwort einer PID-Regelung dargestellt.

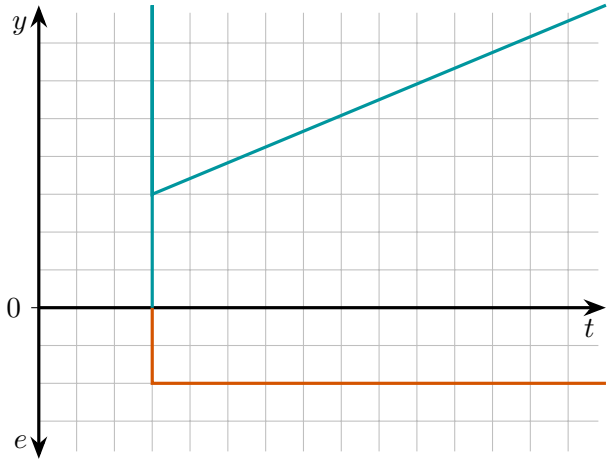


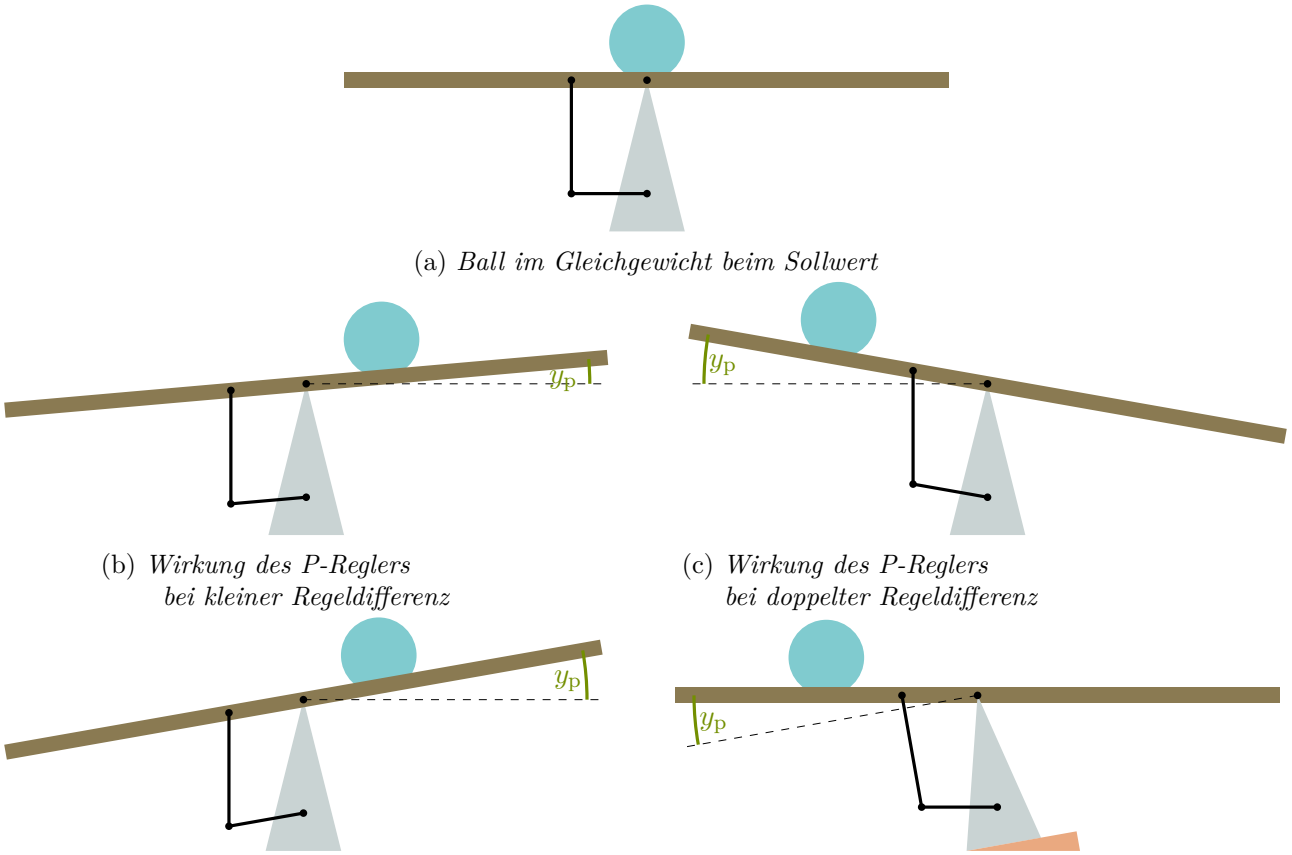
Abbildung 5.6: *Sprungantwort eines PID-Reglers*

5.2. Anschauliche Betrachtung einer PID-Regelung am Beispiel Ball-Balance

Die Abbildungen 5.7 ff. zeigen einen Aufbau zur Positionsregelung bzw. zum Balancieren eines Balls auf einer verstellbaren Wippe. Die Ballposition kann z. B. über einen Abstandssensor erfasst und die Wippe über ein Hebelsystem durch einen Servomotor als Stellgerät gekippt werden. Das Modell eignet sich besonders gut, um auf anschauliche Weise die Wirkung und Bedeutung der drei einzelnen Regelanteile mit ihren Vor- und Nachteilen direkt beobachten zu können. Eine reale Umsetzung wird in Anhang D demonstriert.

5.2.1. P-Wirkung

Nach Gleichung 5.1 gibt der P-Anteil bei Auftreten einer Regeldifferenz e unmittelbar eine zu dieser proportionale Stellgröße y_p aus.



(a) *Ball im Gleichgewicht beim Sollwert*

(b) *Wirkung des P-Reglers bei kleiner Regeldifferenz*

(c) *Wirkung des P-Reglers bei doppelter Regeldifferenz*

(d) *Wirkung des P-Reglers bei Verdopplung von K_p*

(e) *bleibende Regeldifferenz bei konstanter Störung*

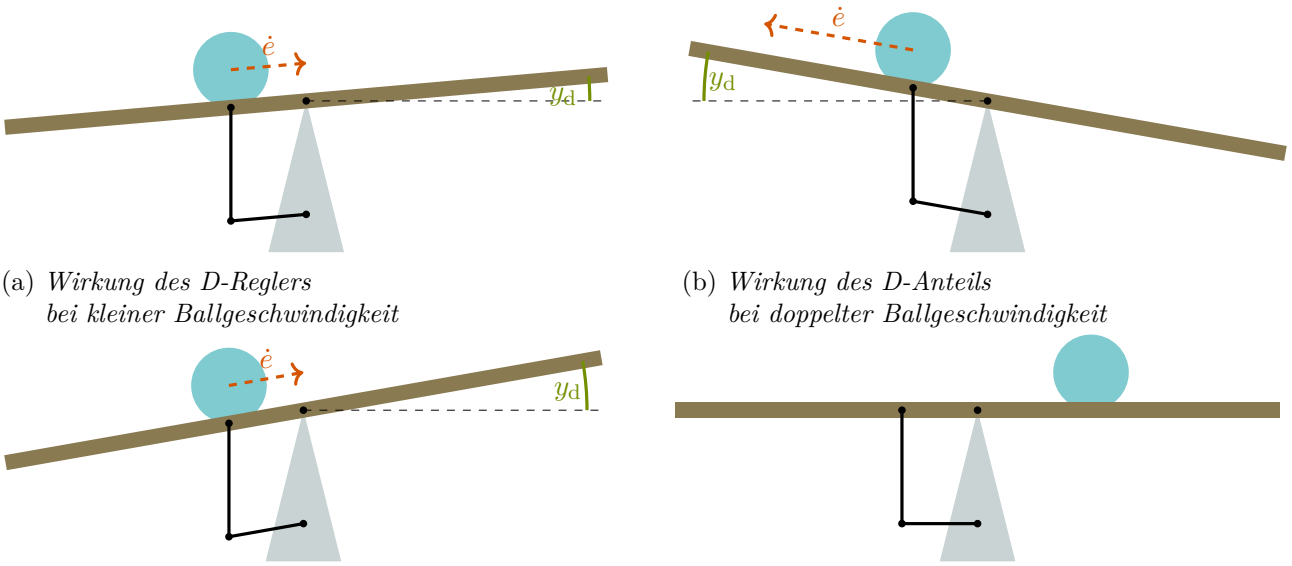
Abbildung 5.7: *P-Wirkung zur Positionsregelung eines Balls auf einer Wippe*

In Abbildung 5.7a befindet sich der Ball in der Mitte beim Sollwert. Die Regeldifferenz ist null. Damit gibt der P-Anteil keine Stellgröße aus und die Wippe befindet sich in der Horizontalen. Die Abbildungen 5.7b und 5.7c zeigen, wie der P-Anteil bei Auftreten einer Regeldifferenz eine Verstellung der Wippe über das Hebelsystem herbeiführt, deren Richtung vom Vorzeichen der Regeldifferenz abhängt. Bei Verdopplung der Regeldifferenz verdoppelt sich auch die Stellgröße. Eine Verdopplung von K_p bewirkt, dass bei gleicher Regeldifferenz der doppelte Wert für die Stellgröße ausgegeben wird (vgl. Abb.5.7b und 5.7d).
Abbildung 5.7e zeigt einen Nachteil einer reinen P-Regelung. In der dargestellten Situation wirkt eine dauerhafte, konstante Störung auf die Regelstrecke ein. Um den Ball zu stabilisieren, muss die Wippe wieder in die Horizontale gebracht werden. Nach Gleichung 5.1 wird die hierzu notwendige Stellgröße

vom P-Anteil aber nur ausgegeben, wenn auch eine Regeldifferenz vorhanden ist. Der P-Anteil kann somit die konstante Störung (alleine) nicht auf den Sollwert ausregeln. Man spricht in diesem Fall von einer bleibenden Regeldifferenz.
Erhöht man K_p, so ist nur noch eine kleinere Regeldifferenz zur Produktion der notwendigen Stellgröße notwendig. Die bleibende Regeldifferenz wird somit kleiner. Allerdings können zu große K_p-Werte zu (unkontrollierbaren) Schwingungen der Regelgröße x führen[1], was im gezeigten Beispiel auf die Trägheit des Balls und zu geringe dämpfende Reibung innerhalb der Regelstrecke zurückgeführt werden kann.

5.2.2. D-Wirkung

Nach Gleichung 5.3 gibt der D-Anteil eine Stellgröße y_d aus, die proportional zur Änderungsgeschwindigkeit der Regeldifferenz $\mathrm{d}e/\mathrm{d}t = \dot{e}$ und damit auch zur Geschwindigkeit des Balls ist.



(a) *Wirkung des D-Reglers bei kleiner Ballgeschwindigkeit*

(b) *Wirkung des D-Anteils bei doppelter Ballgeschwindigkeit*

(c) *Wirkung des D-Reglers bei Verdopplung von K_d*

(d) *keine D-Wirkung bei stillstehendem Ball $\dot{e} = 0$*

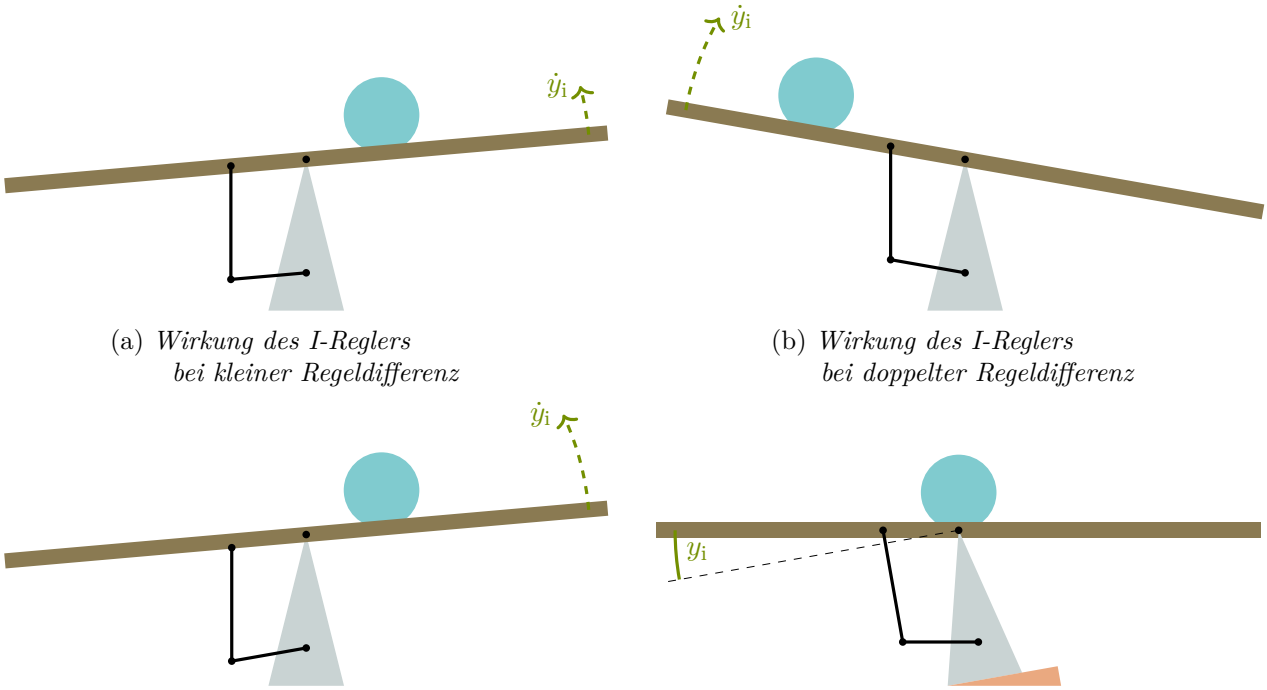
Abbildung 5.8: *D-Wirkung zur Positionsregelung eines Balls auf einer Wippe*

Die Abbildungen 5.8a und 5.8b zeigen, wie der D-Anteil auf die Ballgeschwindigkeit mit einer Verkippung der Wippe reagiert, die zu einer Abbremsung führt, d. h. die Kipprichtung hängt vom Vorzeichen der Regeldifferenzänderung und somit von der Bewegungsrichtung des Balls ab. Eine Verdopplung der Geschwindigkeit führt dabei zum doppelten Kippwinkel. Aufgrund seiner Trägheit neigt der Ball zum Überschwingen über den Sollwert. Durch den D-Anteil kann dem entgegengewirkt und das Überschwingen gedämpft werden. Eine Verdopplung von K_d bewirkt, dass bei gleicher Änderung der Regeldifferenz der doppelte Wert für die Stellgröße ausgegeben wird (vgl. Abb.5.8a und 5.8c).
Abbildung 5.8d zeigt einen Nachteil für den Fall, in welchem der D-Anteil alleine verwendet wird. Bei konstanter Regeldifferenz ($\dot{e} = 0$) wird keine Stellgröße ausgegeben. Die Wippe bleibt in der Horizontalen und somit entsteht keine Wirkung, die den Ball zurück zum Sollwert führen könnte.

[1] vgl. später Abb. 5.11a auf Seite 47

5.2.3. I-Wirkung

Nach Gleichung 5.7 wirkt der I-Anteil in der Art, dass er die Stellgröße kontinuierlich mit einer zur Regeldifferenz e proportionalen Geschwindigkeit $\mathrm{d}y_\mathrm{i}/\mathrm{d}t = \dot{y}_\mathrm{i}$ verändert.



(a) *Wirkung des I-Reglers bei kleiner Regeldifferenz*

(b) *Wirkung des I-Reglers bei doppelter Regeldifferenz*

(c) *Wirkung des I-Reglers bei Verdopplung von* K_i

(d) *bei konstanter Störung*

Abbildung 5.9: *I-Wirkung zur Positionsregelung eines Balls auf einer Wippe*

Die Abbildungen 5.9a und 5.9b zeigen, dass die Wippe bei Auftreten einer Regeldifferenz kontinuierlich verdreht wird, wobei die Drehrichtung vom Vorzeichen der Regeldifferenz abhängt. Eine Verdopplung der Regeldifferenz führt dabei auch zu einer Verdopplung der Drehgeschwindigkeit. Ebenso bewirkt eine Verdopplung von K_i, dass bei gleicher Regeldifferenz die doppelte Drehgeschwindigkeit hervorgerufen wird (vgl. Abb.5.9a und 5.9c).
Abbildung 5.9d zeigt, dass bei einer konstanten Störung die Wippe solange gedreht wird, bis der Ball wieder beim Sollwert angekommen ist und die Regeldifferenz damit null geworden ist. Die Stellgröße hat dabei einen neuen stabilen Wert angenommen, der die Wippe in die Horizontale zurück gebracht hat. Der Vorteil des I-Anteils ist demnach, dass eine konstante Störung vollständig ausgeregelt wird und es somit keine bleibende Regeldifferenz gibt.
Bis der I-Anteil die neue konstante Stellgröße gefunden hat, muss der Ball (wiederum aufgrund seiner Trägheit) im Allgemeinen einige Male hin und her schwingen, sodass der Regelvorgang sehr lange dauern kann, insbesondere wenn kaum Reibung entlang der Regelstrecke auftritt.

5.2.4. Fazit

Jeder Regelanteil hat seine besondere Wirkung/Bedeutung. Eine optimale Regelwirkung erzielt man aber nur dann, wenn die individuellen Vorteile der einzelnen Anteile kombiniert und dadurch gleichzeitig deren Nachteile eliminiert werden. Die Kunst besteht darin, die Gewichtung der drei Anteile untereinander so zu wählen, dass (bezogen auf die vorliegende Regelstrecke) die an einen optimalen Regelvorgang gestellten Anforderungen (vgl. Abschnitt 2.4) erfüllt werden.

5.3. Implementierung des PID-Algorithmus und Test mit dem Sprungantwortverfahren

Die Gleichungen 5.1, 5.3, 5.8 und 5.14 werden nun verwendet, um eine PID-Regelung zu implementieren. Dabei wird bewusst auf die drei Parameter K_p, K_i sowie K_d und nicht auf die Nachstellzeit T_i und die Vorhaltzeit T_d zurückgegriffen[1]. Dadurch bleibt eine unabhängige Gewichtung der jeweiligen Anteile möglich. Insbesondere kann durch Setzen von $K_\mathrm{p} = 0$ der P-Anteil zu Test- und Lehrzwecken auch komplett ausgeschaltet werden, sodass z. B. auch eine reine I-Regelung erprobt werden kann.
Aufgrund der Limitierungen der verwendeten Regeleinrichtung (vgl. Abschnitt 2.2.1) muss bei der Implementierung mit Näherungen gearbeitet werden. Insbesondere kommt hier die zeitliche Auflösung durch eine begrenzte Abtastrate zum Tragen. So kann die Ableitung nach der Zeit der Regeldifferenz nicht exakt bestimmt, sondern nur durch einen Differenzenquotienten nach Gleichung 5.17 aus den Regeldifferenzen zweier aufeinanderfolgender Zeitpunkte angenähert werden. Auch das Zeitintegral über die Regeldifferenz kann nur nach Gleichung 5.18 approximiert werden, indem man (kleine) Rechteckflächen aufsummiert.

$$\frac{\mathrm{d}e}{\mathrm{d}t} \approx \frac{\Delta e}{\Delta t} = \frac{e_k - e_{k-1}}{t_k - t_{k-1}} \tag{5.17}$$

$$\int e(\tau)\mathrm{d}\tau \approx \sum_k e_k \cdot \Delta t = \sum_k e_k \cdot (t_k - t_{k-1}) \tag{5.18}$$

Code 5.1+5.2 zeigt die Umsetzung des PID-Algorithmus. Um die erwartungsgemäße Funktion testen zu können, wird dabei das Sprungantwortverfahren implementiert. Da dabei der Regelkreis noch nicht geschlossen ist, d. h. insbesondere eine Übertragung der Stellgröße an den Steller und eine Aufnahme von aktuellen Temperaturwerten nicht erfolgen muss, werden diese Funktionalitäten noch nicht mit eingebaut und es wird sich zunächst auf die Stellgrößenbestimmung konzentriert. Nach erfolgreichem Test des Algorithmus wird dann in Abschnitt 5.4 eine Anpassung/Erweiterung zur Temperaturregelung vorgenommen und der Regelkreis geschlossen.
Um die Sprungantwort auslösen zu können, wird nochmal ein Taster verbaut und eine Interruptfunktion verwendet (vgl. Code 3.1). Der Taster wird wieder an Pin Nr. 3 angeschlossen. Dazu wird in Zeile 1 eine entsprechende Konstante deklariert. Der `buttonPin` wird innerhalb der `setup()`-Funktion in Zeile 33 als Eingang festgelegt und in Zeile 34 wird die Interruptfähigkeit aktiviert. Wechselt der Pegel am `buttonPin` bei Betätigung von `LOW` zu `HIGH` (`RISING`), so wird der Programmfluss unterbrochen und einmalig die Funktion `buttonFunc()` (Zeile 27 bis 29) ausgeführt. Diese weist dem Sollwert über die Variable `w` einen neuen Wert zu. Der Anfangswert des Sollwertes wird zuvor schon in Zeile 10 festgelegt. Da die entsprechende Variable `w` innerhalb der Interruptfunktion verändert wird, muss diese wieder als `volatile` deklariert werden.
Für die Regelgröße wird in Zeile 12 eine Variable `x` festgelegt, welcher zunächst der Wert des Sollwertes `w` zugewiesen wird. Darauf folgend wird in Zeile 13 eine Variable für die Regeldifferenz `e` deklariert, welche zunächst das Ergebnis der Rechnung `w - x` als Wert erhält. Da Regelgröße und Sollwert allerdings zu Beginn noch gleich sind, hat die Regeldifferenz damit den Anfangswert null.
Um den Differenzenquotienten und das Integral näherungsweise nach den Gleichungen 5.17 und 5.18 bestimmten zu können, werden in den Zeilen 5 und 14 die Variablen `tMem` und `eMem` deklariert, welche dazu dienen werden, den letzten Zeit- bzw. Regeldifferenzwert zu merken. Zudem wird in Zeile 11 eine Konstante `R` für die Größe des Messumfangs angelegt, sodass später die Regeldifferenz relativ hierzu in Prozent umgerechnet werden kann. Für das in dieser Arbeit verwendete Temperaturregelmodell wird hier und in allen folgenden Beispielen von einem Messumfang von 100 °C ausgegangen.

[1] Durch geringe Veränderungen am Code ist es natürlich möglich, den PID-Algorithmus auf die Verwendung der Nachstellzeit und der Vorhaltzeit anzupassen.

In den Zeilen 16 bis 18 werden Konstanten für die drei Beiwerte deklariert und diesen jeweils Zahlenwerte zugewiesen. Zur Bildung der Stellgröße aus den einzelnen Regelungsanteilen werden zwischen den Zeilen 20 und 25 mehrere Variablen mit Anfangswerten von null und eine Konstante angelegt. Den Variablen `yp`, `yi` und `yd` werden die momentanen Stellwerte der P-, I- bzw. D-Anteile in jedem Schleifendurchlauf zugeordnet. Die Variable `dyi` dient dazu, den Wert der Stellgrößenänderung des I-Anteils zwischen zwei aufeinanderfolgenden Zeitpunkten aufzunehmen. Eine weitere Konstante `yo` wird angelegt, um allgemein die Möglichkeit zu haben, einen Arbeitspunkt über einen Offsetwert einzustellen. Dieser wird zunächst noch auf null gesetzt und spielt hier bei der Aufnahme der Sprungantwort deshalb noch keine Rolle. Die Relevanz eines konstanten Offsets wird erst in Abschnitt 5.5.2 demonstriert. Letztendlich wird noch die Variable `y` benötigt, um hierin den Gesamtwert der Stellgröße zusammenfassen zu können.

Code 5.1.: *PID-Algorithmus und Sprungantwort - Teil 1*

```
1  const int buttonPin = 3;                  //Taster an Pin 3
2
3  const float dt = 0.1;                     // Zeitintervall in s
4  float t = 0.0;                            // aktuelle Zeit
5  float tMem = t;                           // vorherige Zeit
6
7  /* der Sollwert wird nach Tasterdruck ueber eine Interuptfunktion veraendert
8   * weshalb die Variable als volatile deklariert werden muss
9   */
10 volatile int w = 50;                       // Sollwert
11 const int R = 100;                        // Messumfang
12 float x = w;                              // aktueller Istwert
13 float e = w - x;                          // aktuelle Regeldifferenz in Prozent
14 float eMem = 0.0;                        // vorherige Regeldifferenz in Prozent
15
16 const float Kp = 2.0;                     // Proportionalbeiwert
17 const float Ki = 0.05;                    // Integralbeiwert
18 const float Kd = 10;                      // Differentialbeiwert
19
20 float yp = 0.0;                           // Stellwert P-Anteil in Prozent
21 float dyi = 0.0;                          // Stellwertaenderung I-Antiel in Prozent
22 float yi = 0.0;                          // Stellwert I-Anteil in Prozent
23 float yd = 0.0;                          // Stellwert D-Anteil in Prozent
24 const float yo = 0.0;                    // Stellwert-Offset in Prozent
25 float y = 0.0;                           // Stellwert gesamt in Prozent
26
27 void buttonFunc() {                      // Interuptfunktion
28   w = 70;
29 }
30
31 void setup() {
32   Serial.begin(9600);
33   pinMode(buttonPin, INPUT);
34   attachInterrupt(digitalPinToInterrupt(buttonPin), buttonFunc, RISING);
35 }
```

Innerhalb der `setup()`-Funktion wird zunächst in Zeile 38 der vorliegende Wert der Zeitvariablen `t` zum Merken in die Variable `tMem` geschrieben, bevor er in Zeile 39 mit dem neuen aktuellen Wert überschrieben wird. Ebenso wird in Zeile 40 der Wert der Regeldifferenz `e` in die Merkvariable `eMem` übertragen, um dann in Zeile 41 ebenfalls mit einem neuen Wert überschrieben zu werden. Dabei

wird `e` als Differenz aus Sollwert `w` und Istwert `x` relativ zum Messumfang `R` bestimmt und über den Faktor 100,0 in Prozent umgerechnet.

Code 5.2.: *PID-Algorithmus und Sprungantwort - Teil 2*

```
36 void loop() {
37   if (millis() / 1000.0 - t >= dt) {
38     tMem = t;                                  // vorherige Zeit merken
39     t = millis() / 1000.0;                     // aktuelle Zeit in s bestimmen
40     eMem = e;                                  // vorherige Regeldifferenz merken
41     e = 100.0 * (w - x) / R;                   // aktuelle Regeldifferenz berechnen in Prozent
42
43     yp = Kp * e;                               // P-Anteil des Stellwerts berechnen
44     yd = Kd * (e - eMem) / (t - tMem);         // D-Anteil des Stellwerts berechnen
45     dyi = Ki * e * (t - tMem);                 // I-Anteilaenderung des Stellwerts berechnen
46     yi += dyi;                                 // I-Anteilaenderung aufsummieren
47     // I-Anteil begrenzen
48     if (yi + yo < 0) {yi = -yo;}
49     else if (yi + yo > 100) {yi = 100 - yo;}
50
51     y = yp + yi + yd + yo;                     // Stellwertanteile aufsummieren
52     // Stellwertbegrenzung
53     if (y > 100) {y = 100;}
54     else if (y < 0) {y = 0;}
55     // die aktuellen Werte werden ausgegeben
56     Serial.print(t, 1);
57     Serial.print('\t');
58     Serial.print(y, 4);
59     Serial.print('\t');
60     Serial.println(e, 4);
61   }
62 }
```

In Zeile 43 wird der Stellwert des P-Anteils entsprechend Gleichung 5.1 berechnet und der Variablen `yp` zugewiesen. Der D-Anteil, welcher in Zeile 44 der Variablen `yd` zugeordnet wird, ergibt sich in Anlehnung an Gleichung 5.3 mithilfe eines Differenzenquotienten entsprechend Gleichung 5.17. Zeile 45 dient dazu, zu berechnen, um welchen Wert `dyi` sich der I-Anteil seid dem vorangegangenen Schleifendurchlauf innerhalb des Zeitintervalls `dt = t - tMem` verändert hat (vgl. Gleichung 5.8). Diese Änderung wird in Zeile 46 zum bereits bestehenden Wert der Variablen `yi` hinzu addiert, was der Annäherung der Integration über eine Summenbildung entspricht. Hierzu wird der Operator `+=` verwendet.
Es ist wichtig, den Wert des I-Anteils zu begrenzen, damit das Integral der Regeldifferenz sich nicht weiter verändert, sobald die Grenzen des Stellbereichs (Übersteuerung) erreicht werden. Ansonsten könnte der Wert der Variablen `yi` im Übersteuerungszeitraum z. B. weit über den Wert 100 % hinaus steigen und es würde nach einem Vorzeichenwechsel der Regeldifferenz lange dauern, bis er wieder unter 100 % fällt. Dadurch könnte die Übersteuerungszeit deutlich verlängert werden. Man spricht hier vom Windup-Effekt. Die Zeilen 48 und 49 dienen entsprechend dazu, den Wert der Variablen `yi` und damit den I-Anteil zu begrenzen. Sollte eine der formulierten Grenzbedingungen eintreffen, wird `yi` auf einen Wert festgesetzt, welcher zusammen mit dem Offset den oberen bzw. unteren Grenzwert des Stellbereichs ergibt. In Zeile 51 werden schließlich alle Anteile (inklusive eines möglichen Offsets) in der Variablen `y` zusammengefasst[1]. Auch hiernach muss nochmal geprüft werden, ob es zu einer

[1] Man bedenke nochmals, dass die Regeldifferenz auch negative Werte annehmen kann, sodass der Wert der Stellgröße im Laufe der Zeit steigen aber auch fallen kann.

Übersteuerung kommt, sodass eine entsprechende Begrenzung des Gesamtstellwerts `y` vorgenommen werden muss. Dazu dienen die in den Zeilen 53 und 54 formulierten Anweisungen.
Letztendlich werden in den Zeilen 56 bis 60 in jedem Schleifendurchlauf die Werte der Variablen `t`, `y` und `e` über die serielle Schnittstelle ausgegeben, sodass diese z. B. für eine grafische Darstellung der Sprungantwort verwendet werden können.

Beurteilung der aufgenommenen Sprungantwort: Abbildung 5.10 zeigt eine aufgenommene Sprungantwort zum implementierten PID-Algorithmus, wobei die Werte $K_\mathrm{p} = 2$, $K_\mathrm{i} = 0{,}05\,\frac{1}{\mathrm{s}}$ und $K_\mathrm{d} = 10\,\mathrm{s}$ verwendet worden sind.

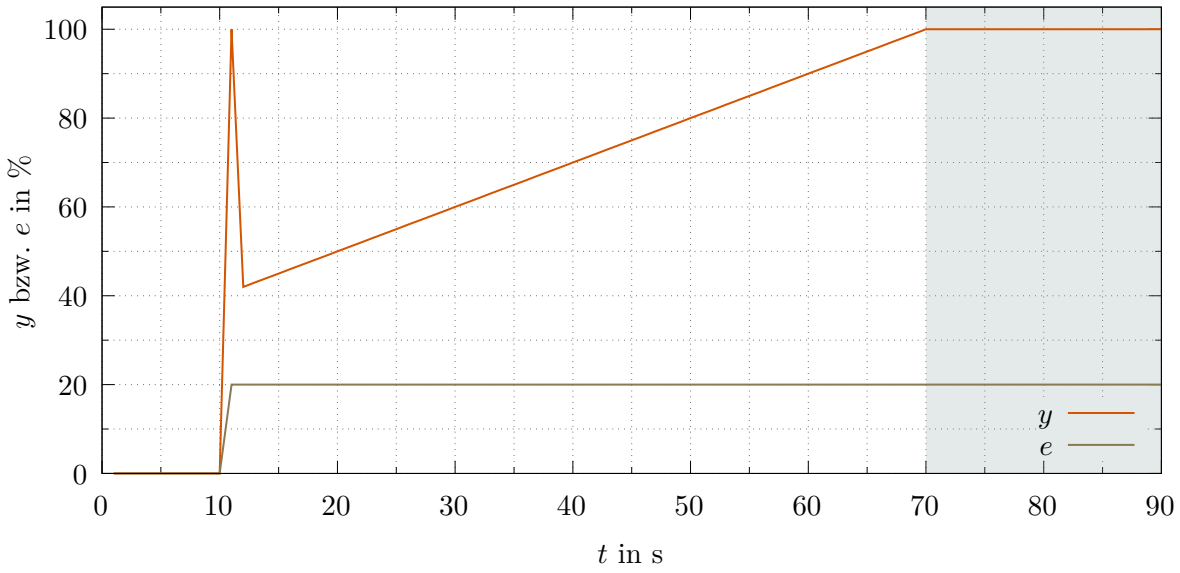


Abbildung 5.10: *Sprungantwort des implementierten PID-Algorithmus*

Zum Zeitpunkt $t = 10\,\mathrm{s}$ wurde durch Druck auf den Taster die Interruptfunktion ausgelöst und der Sollwert w von 50 auf 70 erhöht. Beim angenommenen Messumfang von 100 und einem Istwert von $x = 50$ entspricht dies einer Regeldifferenz von $e = 20\,\%$. Anhand des Diagramms ist zu erkennen, dass dieser Wert ab dem Zeitpunkt $t = 11\,\mathrm{s}$ gleichbleibend vorliegt. Die begrenzte Abtastrate von $\mathrm{dt} = 1\,\mathrm{s}$ führt zu Abweichungen vom idealen Verhalten des PID-Reglers. Der Sprung der Regeldifferenz ist nicht unendlich groß. Trotzdem zeigt die Rechnung

$$y_\mathrm{d} = K_\mathrm{d} \cdot \frac{\Delta e}{\Delta t} = 10\,\mathrm{s} \cdot \frac{20\,\%}{1\,\mathrm{s}} = 200\,\% > 100\,\% \quad ,$$

dass die Stellgröße zum Zeitpunkt $t = 11\,\mathrm{s}$ in die Übersteuerung springt. Danach ist die Regeldifferenzänderung $\dot{e} = 0$, sodass $y_\mathrm{d} = 0$ wird und ab dem Zeitpunkt $t = 12\,\mathrm{s}$ die Stellgröße zurück auf den Wert $y = y_\mathrm{p} + y_\mathrm{i}$ fällt. Zum Zeitpunkt $t = 12\,\mathrm{s}$ liegt die Auslösung der Regeldifferenz zwei Sekunden zurück und man erhält dort folgenden Stellwert:

$$y = y_\mathrm{p} + y_\mathrm{i} = K_\mathrm{p} \cdot e + K_\mathrm{i} \cdot e \cdot \Delta t = 2 \cdot 20\,\% + 0{,}05\,\frac{1}{\mathrm{s}} \cdot 20\,\% \cdot 2\,\mathrm{s} = 40\,\% + 2\,\% = 42\,\%$$

Danach steigt die Stellgröße aufgrund der fortlaufenden Integration mit einer konstanten Rate von

$$\frac{\mathrm{d}y_\mathrm{i}}{\mathrm{d}t} = K_\mathrm{i} \cdot e = 0{,}05\,\frac{1}{\mathrm{s}} \cdot 20\,\% = 1\,\frac{\%}{\mathrm{s}} \quad ,$$

bis ab dem Zeitpunkt $t = 70\,\mathrm{s}$ wieder eine Übersteuerung bei $y = 100\,\%$ erreicht wird.

5.4. Umsetzung des PID-Algorithmus auf Temperaturregelungen

Code 5.1+5.2 lässt sich mit den bisher in früheren Codebeispielen verwendeten Elementen leicht zu einer Temperaturregelung in einem geschlossenen Regelkreis anpassen. In Code 5.3+5.4 werden zunächst alle Zeilen entfernt, die im Zusammenhang mit der Auslösung der Sprungantwort über den Taster stehen. Dazu werden wieder alle notwendigen Bibliotheken und Codestücke zur Verwendung des `DS18B20`-Temperatursensors eingefügt. Der Regelkreis wird geschlossen, indem wieder alle relevanten Zeilen zur Übertragung der Stellgröße an den Transistor eingefügt werden. Am Ende der `loop()`-Funktion werden diesmal in jedem Schleifendurchlauf die aktuellen Werte der Variablen für die Zeit `t`, die Regelgröße `x` und die Stellgröße `y` über die serielle Schnittstelle ausgegeben.

Code 5.3.: *PID-Temperaturregelung - Teil 1*

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 2    // Sensor an Pin 2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
const int gatePin = 5;

const float dt = 1.0;     // Zeitintervall in s
float t = 0.0;            // aktuelle Zeit
float tMem = t;           // vorherige Zeit

float w = 50.0;           // Sollwert in °C
const float R = 100.0;    // Messumfang in °C
float x = 0.0;            // aktueller Istwert in °C
float e = 0.0;            // aktuelle Regeldifferenz in Prozent
float eMem = 0.0;         // vorherige Regeldifferenz in Prozent

const float Kp = 24.0;    // Proportionalbeiwert
const float Ki = 0.2;     // Integralbeiwert
const float Kd = 600;     // Differentialbeiwert

float yp = 0.0;           // Stellwert P-Anteil in Prozent
float dyi = 0.0;          // Stellwertaenderung I-Anteil in Prozent
float yi = 0.0;           // Stellwert I-Anteil in Prozent
float yd = 0.0;           // Stellwert D-Anteil in Prozent
const float yo = 0.0;     // Stellwert-Offset in Prozent
float y = 0.0;            // Stellwert gesamt in Prozent

void setup() {
  Serial.begin(9600);
  sensors.begin();
  pinMode(gatePin, OUTPUT);
}
```

Code 5.4.: *PID-Temperaturregelung - Teil 2*

```
34 void loop() {
35   if (millis() / 1000.0 >= t + dt) {
36     tMem = t;                                    // vorherige Zeit merken
37     t = millis() / 1000.0;                       // aktuelle Zeit in s bestimmen
38
39     sensors.requestTemperatures();
40     x = sensors.getTempCByIndex(0);              // aktuellen Istwert einlesen
41
42     eMem = e;                                    // vorherige Regeldifferenz merken
43     e = 100.0 * (w - x) / R;                     // aktuelle Regeldifferenz berechnen in Prozent
44
45     yp = Kp * e;                                 // P-Anteil des Stellwerts berechnen
46     yd = Kd * (e - eMem) / (t - tMem);           // D-Anteil des Stellwerts berechnen
47     dyi = Ki * e * (t - tMem);                   // I-Anteilaenderung des Stellwerts berechnen
48     yi += dyi;                                   // I-Anteilaenderung aufsummieren
49     // I-Anteil begrenzen
50     if (yi + yo < 0) {yi = -yo;}
51     else if (yi + yo > 100) {yi = 100 - yo;}
52
53     y = yp + yi + yd + yo;                       // Stellwertanteile aufsummieren
54     // Stellwertbegrenzung
55     if (y > 100) {y = 100;}
56     else if (y < 0) {y = 0;}
57     // Uebertragung des Stellwertes zum Transistor
58     analogWrite(gate, 255 * y / 100.0);
59     // die aktuellen Werte werden ausgegeben
60     Serial.print(t, 1);
61     Serial.print('\t');
62     Serial.print(x, 3);
63     Serial.print('\t');
64     Serial.println(y, 4);
65   }
66 }
```

Code 5.3+5.4 wird nun im folgenden Abschnitt 5.5 verwendet, um die einzelnen Regelungsarten zu testen und gegenüberzustellen. Dazu werden die Werte der entsprechenden Beiwerte über die Variablen `Kp`, `Kd` und `Ki` gezielt angepasst. Insbesondere kann ein Regelanteil ausgeschaltet werden, indem der zugehörige Beiwert auf den Wert null gesetzt wird.

5.5. Beispieldaten und Interpretation

Mithilfe von Code 5.3+5.4 werden nun Temperaturregelverläufe mit unterschiedlichen Werten für die drei Beiwerte K_p, K_i und K_d aufgenommen. Nach Bedarf werden Beiwerte auch auf null gesetzt, um systematisch die Wirkung einzelner Anteile untersuchen und dann die unterschiedlichen Regelungsarten gegenüberstellen zu können. Als Sollwert wird dabei immer ein Wert von $w = 50{,}0\,^\circ\mathrm{C}$ angesetzt. Zudem wird von einem Messumfang von $100\,^\circ\mathrm{C}$ ausgegangen.

5.5.1. P-Temperaturregelung

Abbildung 5.11a zeigt den Verlauf von P-Regelungen bei verschiedenen Proportionalbeiwerten K_p im direkten Vergleich[1]. Es ist zu erkennen, dass der Aufheizvorgang bis zum Zeitpunkt $t \approx 2\,\mathrm{min}$ annähernd gleich verläuft. Dies ist darauf zurückzuführen, dass aufgrund von Übersteuerung die Stellgröße hier ihren maximalen Wert von $100\,\%$ nicht überschreiten kann.
Die drei Regelverläufe zeigen, dass sich nach einer gewissen (Einschwing-)Zeit ein stabiler Temperaturwert unterhalb des Sollwerts einstellt. Wie bei dem Ball-Balance-Modell (Abb. 5.7) kommt es zu einer bleibenden Regelabweichung. Diese lässt sich hier wie folgt erklären. Da der Sollwert über der Umgebungstemperatur liegt, muss der Regelstrecke Wärme zugeführt werden. Um dabei eine stabile Temperatur zu erhalten, ist ein Gleichgewicht zwischen der zugeführten Wärmemenge und den Wärmeverlusten an die Umgebung erforderlich. Die notwendige, stabile Stellgröße wird vom P-Anteil nach Gleichung 5.1 aber nur ausgegeben, wenn noch eine Regeldifferenz besteht. Je größer der K_p-Wert gewählt wird, desto kleiner muss diese bleibende Regeldifferenz sein. Allerdings ist bei größer werdendem K_p-Wert auch zu beobachten, dass sich Schwingungen im Regelgrößenverlauf ausbilden, die auch über den Sollwert hinaus gehen können. Bei weiterer Steigerung von K_p werden diese Schwingungen aufgrund der Verzugszeit der Temperaturregelstrecke immer stärker. Zudem wird der Proportionalbereich kleiner, sodass es schon bei kleinen Regeldifferenzen zu Übersteuerung kommt und sich der P-Regler dem Verhalten eines Zweipunktreglers (vgl. Abb. 4.2) annähert.

genauere Betrachtung des Stellgrößenverlaufs: Bei $K_\mathrm{p} = 24$ ergibt die Rechnung

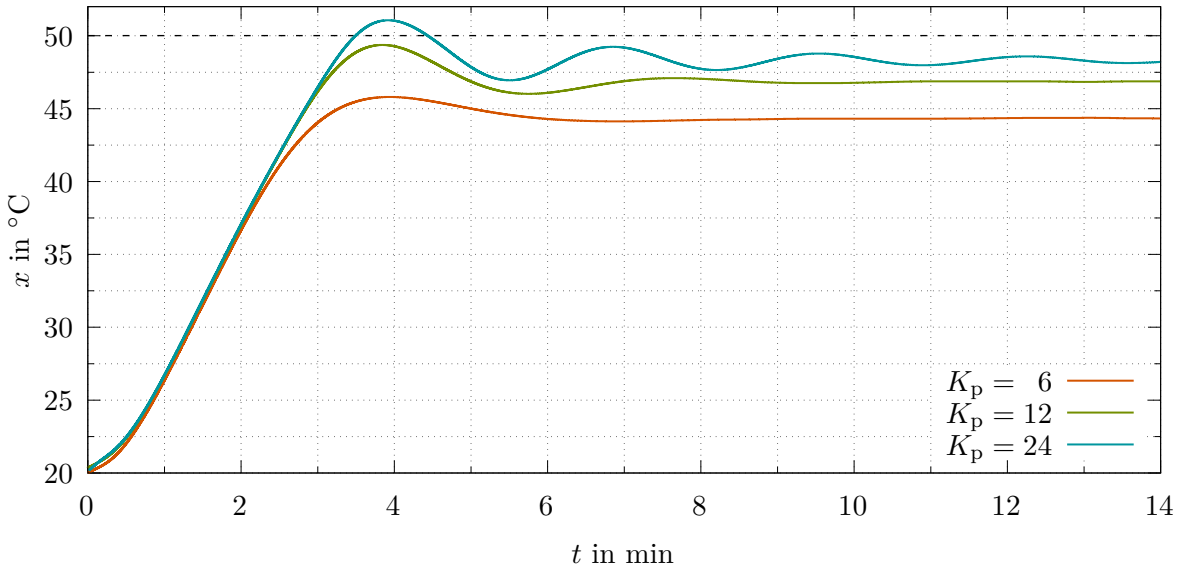
$$e = 100\,\%/24 = 4{,}167\,\% \quad ,$$

dass dieser Wert für die Regeldifferenz die maximale Stellgröße von $100\,\%$ erzeugt. Eine größere Regeldifferenz führt zu Übersteuerung. Da keine negativen Stellgrößen im verwendeten Regelkreis möglich sind, wird bei Temperaturwerten über dem Sollwert und somit bei negativen Regeldifferenzen die Stellgröße auf null gesetzt[2].
Beim Messumfang von $R = 100\,^\circ\mathrm{C}$ und mit dem Sollwert von $50\,^\circ\mathrm{C}$ wird eine Regeldifferenz von $4{,}167\,\%$ bei nachfolgend berechnetem Istwert erreicht:

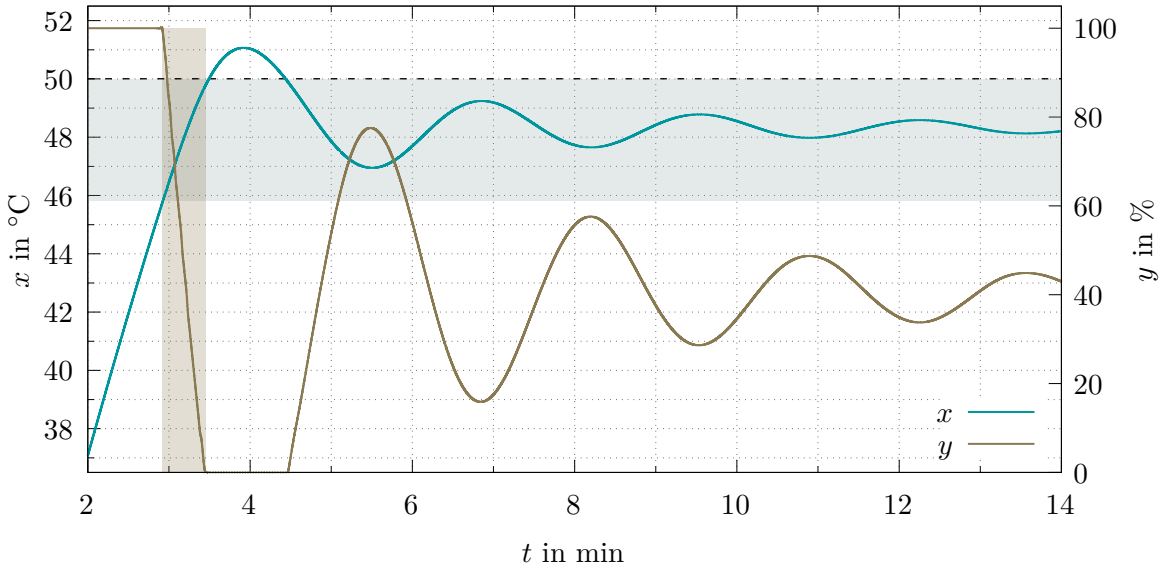
$$e = 100\,\% \cdot \frac{w - x}{R} \qquad \Rightarrow \qquad x = w - \frac{e \cdot R}{100\,\%} = 50\,^\circ\mathrm{C} - \frac{4{,}167\,\% \cdot 100\,^\circ\mathrm{C}}{100\,\%} = 45{,}833\,^\circ\mathrm{C}$$

Der in Abbildung 5.11b grau hervorgehobene Proportionalitätsbereich erstreckt sich demnach von $45{,}833\,^\circ\mathrm{C}$ bis zum Sollwert von $50\,^\circ\mathrm{C}$. Der braun gekennzeichnete Bereich zeigt, wie die Stellgröße ihren vollen Wertebereich durchläuft, wenn die Regelgröße den Proportionalitätsbereich durchquert.

[1] Zur verbesserten Darstellung wurden hier und im Folgenden die Messverläufe interpoliert. Dadurch sind Stufen, welche aus der limitierten Auflösung in der Temperaturmessung und der Stellwertübertragung resultieren, nicht mehr zu erkennen.
[2] sofern kein Offset für die Stellgröße verwendet wird (siehe Abschnitt 5.5.2)



(a) *verschiedene K_p-Werte im Vergleich*



(b) *Gegenüberstellung des Regelgrößen- und Stellgrößenverlaufs bei $K_\mathrm{p} = 24$*
Der Proportionalbereich X_p ist grau hervorgehoben.

Abbildung 5.11: *zeitlicher Verlauf von P-Regelungen*

5.5.2. Optimierung der P-Regelung durch Hinzufügen eines Offsets

Aus Abbildung 5.11b kann abgelesen werden, dass bei $K_\mathrm{p} = 24$ sich eine Stellgröße von ca. $y = 40\,\%$ im Gleichgewichtszustand einstellt. Es wird nun gezeigt, wie ein konstantes Offset für die Stellgröße bzw. ein Arbeitspunkt die P-Regelung positiv beeinflussen kann. Dabei wird zunächst zu Demonstrationszwecken bewusst ein Wert von $y_\mathrm{o} = 20\,\%$ und nicht von $40\,\%$ gewählt, damit im Gleichgewichtszustand der P-Anteil noch eine signifikante Wirkung entfalten muss. Der Gesamtwert der Stellgröße wird nach der Gleichung $y = y_\mathrm{p} + y_\mathrm{o}$ berechnet.
Es wird nun untersucht, wie sich der Proportionalbereich und die bleibende Regeldifferenz verändern. Durch das Offset reicht nun unterhalb des Sollwerts eine geringere Regeldifferenz aus, um eine Gesamtstellgröße von $100\,\%$ zu produzieren:

$$\frac{y_\mathrm{p}}{K_\mathrm{p}} = \frac{y - y_\mathrm{o}}{K_\mathrm{p}} = e \qquad \Rightarrow \qquad e = \frac{100\,\% - 20\,\%}{24} = 3{,}333\,\%$$

Diese wird bei folgender Temperatur erreicht:

$$x = 50\,°\mathrm{C} - \frac{3{,}333\,\% \cdot 100\,°\mathrm{C}}{100\,\%} = 46{,}522\,°\mathrm{C}$$

Oberhalb des Sollwertes muss der P-Anteil nun dafür sorgen, dass die durch das Offset hervorgerufene Stellgröße von 20 % abgebaut wird. Der Heizvorgang wird somit nicht beim Sollwert gestoppt sondern bis zum oberen Endwert des Proportionalitätsbereich mit kontinuierlich abnehmender Stellgröße und damit auch Heizleistung noch weiter geführt. Die entsprechende Temperatur lässt sich folgendermaßen berechnen:

$$e = -\frac{20\,\%}{24} = -0{,}833\,\% \qquad \Rightarrow \qquad x = 50\,°\mathrm{C} - \frac{-0{,}833\,\% \cdot 100\,°\mathrm{C}}{100\,\%} = 50{,}833\,°\mathrm{C}$$

In Abbildung 5.12 ist der berechnete Proportionalitätsbereich von 46,522 °C bis 50,833 °C grau hervorgehoben. Im Vergleich zu Abbildung 5.11b ist festzustellen, dass die bleibende Regeldifferenz nun geringer ausfällt, da durch das Offset der P-Anteil jetzt einen geringeren Wert für die Stellgröße produzieren muss. Allerdings ist durch die etwas höhere Temperatur und den damit auch steigenden Wärmeverlusten an die Umgebung[1] nun eine insgesamt leicht vergrößerte Stellgröße zur Erreichung eines stabilen Gleichgewichtszustand erforderlich.
Theoretisch wäre es denkbar, durch ein passendes Stellgrößenoffset die bleibende Regeldifferenz auf null zu reduzieren. Sollte sich aber die Umgebungstemperatur verändern, so ist wiederum eine andere Stellgröße erforderlich, um genau beim Sollwert den Gleichgewichtszustand zu erreichen. Die Aufgabe einer dynamischen Regelung ist es gerade, flexibel auf unerwünschte Störungen schnell und automatisiert reagieren zu können.

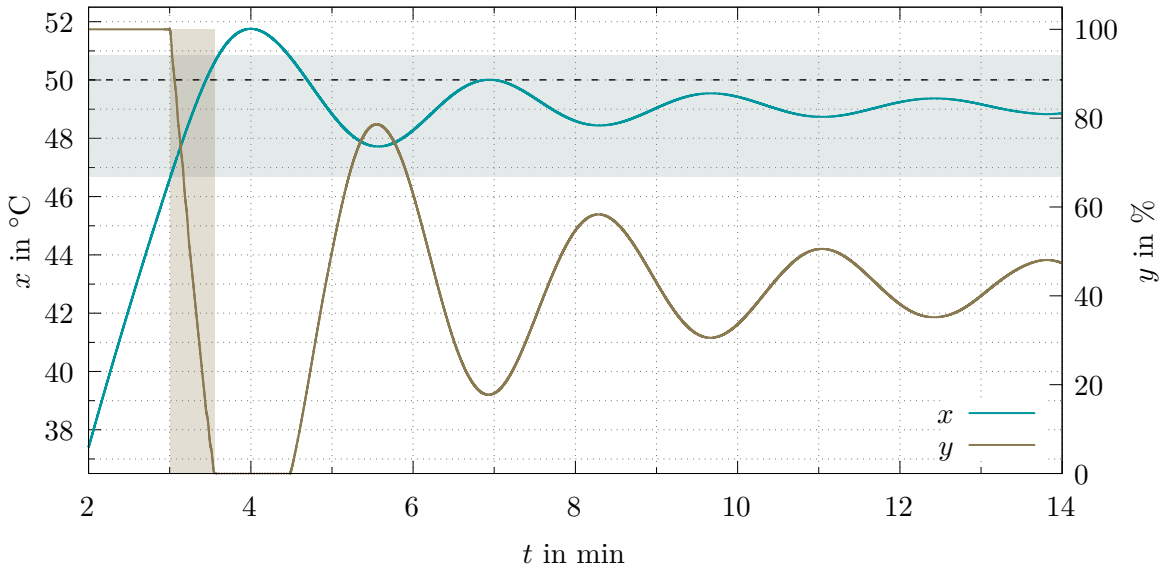


Abbildung 5.12: *Optimierung der P-Regelung durch Hinzufügen eines konstanten Offsets von* $y_\mathrm{o} = 20\,\%$. *Gegenüberstellung des Regelgrößen- und Stellgrößenverlaufs bei* $K_\mathrm{p} = 24$ *Der Proportionalbereich* X_p *ist grau hervorgehoben.*

5.5.3. PD-Temperaturregelung

Ein Kennzeichen einer guten Regelung ist es, den Istwert möglichst schwingungsfrei, insbesondere mit nur kleiner Überschwingweite an den Sollwert heranzuführen. Wie auch im Ball-Balance-Modell aus

[1] $\dot{Q} \sim \Delta x$

Abschnitt 5.2 (vgl. Abb. 5.8), kann ein D-Anteil verwendet werden, um Schwingungen im Temperaturverlauf zu dämpfen. Aus $e = w - x$ folgt aufgrund der Konstanz des Sollwertes $\mathrm{d}e/\mathrm{d}t = -\mathrm{d}x/\mathrm{d}t$. Steigt die Temperatur an, so wird die Stellgröße reduziert, wohingegen sie bei abfallender Temperatur wieder erhöht wird, insbesondere auch dann, wenn die Temperatur von höheren Werten wieder zurück in Richtung Sollwert sinkt[1]. Zu den Zeitpunkten, an denen sich die Temperatur am stärksten ändert und die Temperaturkurve ihre größte Steigung hat, ist die D-Wirkung am größten.

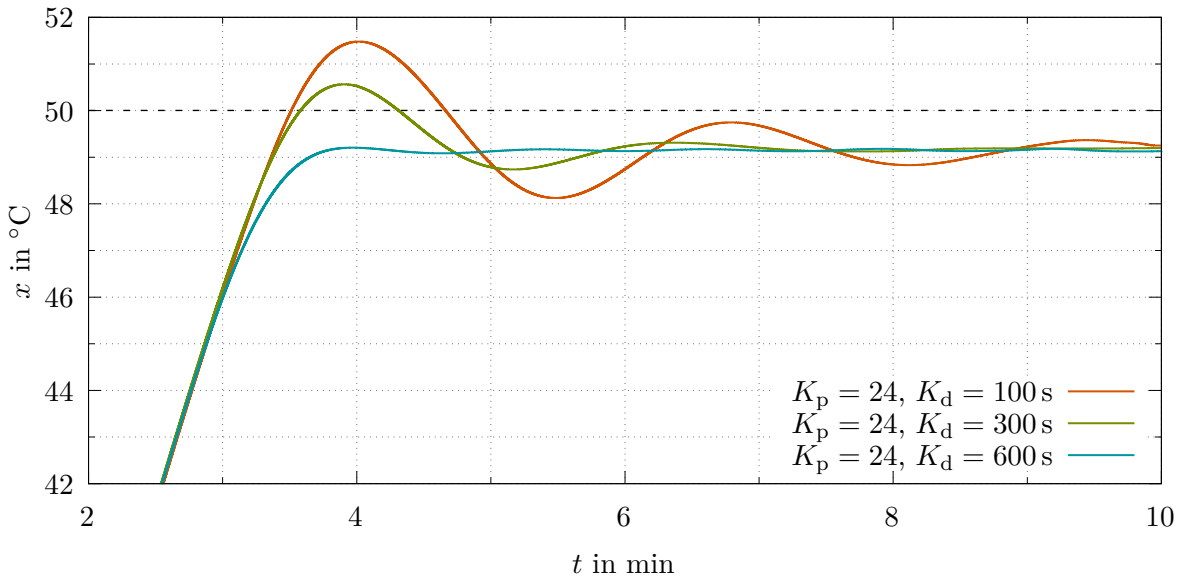


Abbildung 5.13: *zeitlicher Verlauf von PD-Regelungen für verschiedene K_d-Werte im Vergleich*

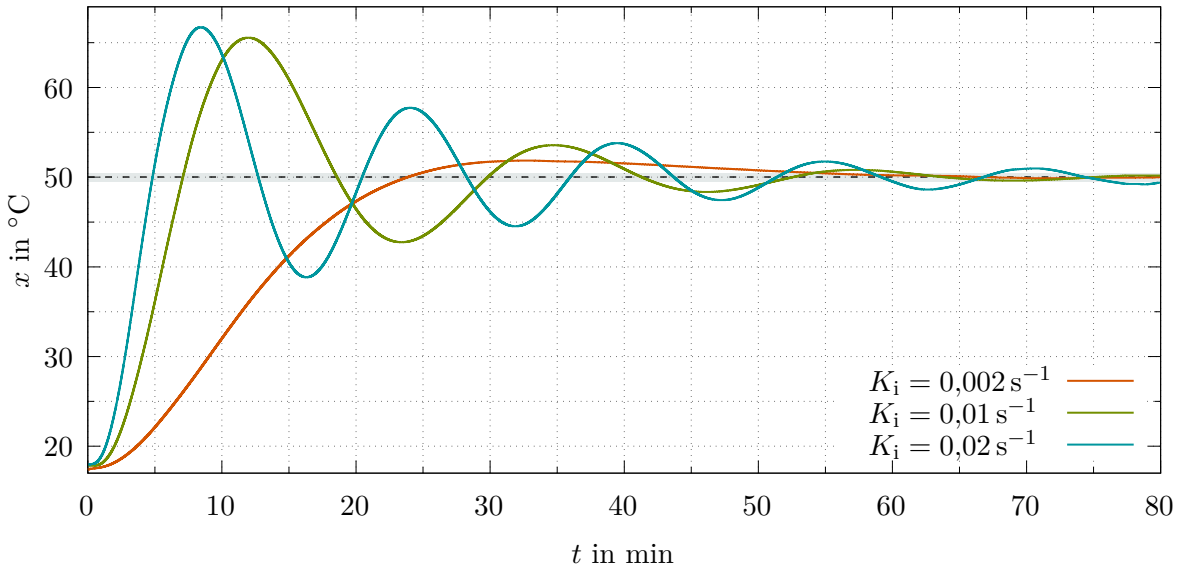
Abbildung 5.13 zeigt die Wirkung des D-Anteils für unterschiedlich große Differenzierbeiwerte K_d im Rahmen einer PD-Regelung. Dabei wurde ein konstanter Proportionalbeiwert von $K_\mathrm{p} = 24$ und ein Stellgrößenoffset von $y_\mathrm{o} = 20\,\%$ verwendet, damit ein direkter Vergleich mit dem Verlauf aus Abbildung 5.12 möglich ist.
Die immer stärker dämpfende Wirkung bei größer werdenden K_d-Werten ist deutlich zu erkennen. Allerdings hat der D-Anteil keinen Einfluss auf die bleibende Regeldifferenz, welche sich bei allen drei Verläufen beim gleichen Wert einpendelt. Um diese zu beseitigen, ist ein Integralanteil notwendig, dessen Wirkung auf die Temperaturregelstrecke ab dem nächsten Abschnitt 5.5.4 untersucht wird.

5.5.4. I-Temperaturregelung

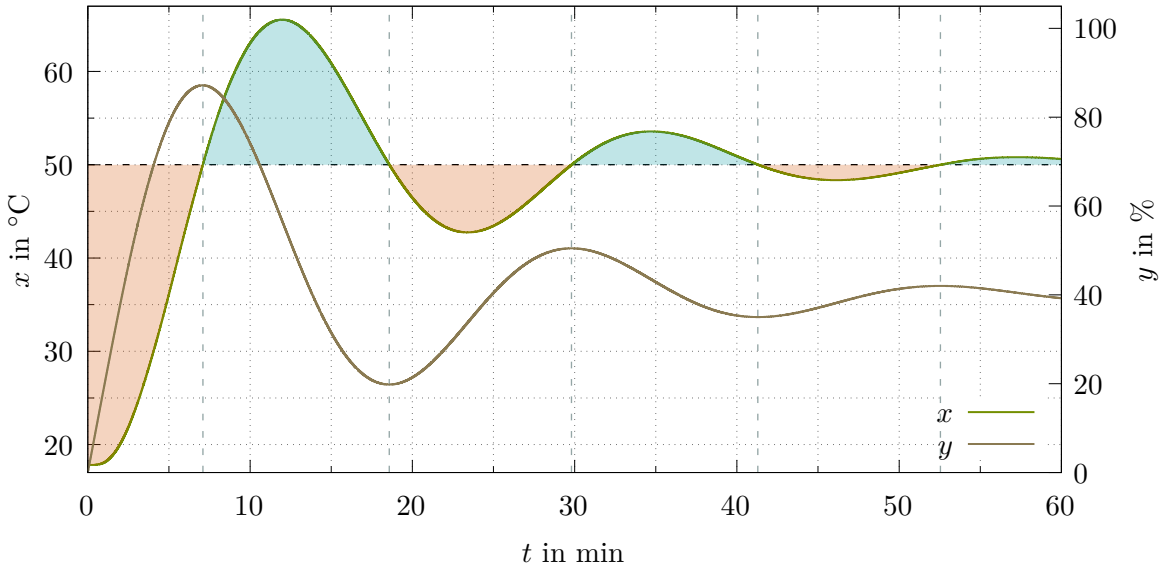
Abbildung 5.14a zeigt die zeitlichen Verläufe von drei reinen I-Regelungen mit unterschiedlichen Integralbeiwerten K_i im direkten Vergleich. Hierbei wurde kein Offset für die Stellgröße verwendet. Man erkennt, dass sich nach einer langen Einschwingzeit die Temperatur um den Sollwert einpendelt und es somit zu keiner bleibenden Regeldifferenz kommt. Zum späteren Vergleich mit anderen Regelungsarten wurde in grau ein Toleranzband von $50\,^\circ\mathrm{C} \pm 0{,}5\,^\circ\mathrm{C}$ in das Diagramm eingetragen. Je größer der K_i-Wert ist, desto früher erreicht die Regelgröße das erste Mal den Sollwert. Allerdings steigt dabei auch die Überschwingweite und die Schwingungsfrequenz.
In Abbildung 5.14b sind für den Regelverlauf mit $K_\mathrm{i} = 0{,}01\,\frac{1}{\mathrm{s}}$ die Bereiche mit einer positiven und einer negativen Regeldifferenz farblich unterschiedlich hervorgehoben. In Bereichen positiver Regeldifferenz steigt die Stellgröße proportional zum Wert der markierten Fläche, welche dem Zeitintegral der Regeldifferenz entspricht, an, während sie analog dazu in Bereichen negativer Regeldifferenz wieder

[1] Dieses Verhalten kann mit einer geschwindigkeitsabhängigen, dämpfenden Reibungskraft bei einer mechanischen Schwingung verglichen werden.

sinkt. Nachdem sich die Temperatur beim Sollwert eingependelt hat und die Regeldifferenz null geworden ist, ändert sich der Wert der Stellgröße nicht mehr[1]. Auch hier wird zur Erreichung des Gleichgewichtszustand eine Stellgröße von $y \approx 40\,\%$ benötigt.



(a) *verschiedene K_i-Werte im Vergleich*



(b) *Gegenüberstellung des Regelgrößen- und Stellgrößenverlaufs bei $K_\mathrm{i} = 0{,}01\,\frac{1}{\mathrm{s}}$*
Die hervorgehobenen Flächen stellen das Integral der Regeldifferenz dar, wobei die beiden Farben für die unterschiedlichen Vorzeichen (+ und −) stehen.

Abbildung 5.14: *zeitlicher Verlauf von I-Regelungen*

5.5.5. PI-Temperaturregelung

In Abschnitt 5.5.4 wurde gezeigt, dass eine Integralwirkung dafür sorgt, dass es zu keiner bleibenden Regeldifferenz kommt, es bis zur Ausregelung allerdings sehr lange dauern kann. Um mehr Dynamik in den Regelverlauf zu bringen, kann eine Kombination aus Proportional- und Integralwirkung verwendet werden.

[1] Es werden keine bzw. nur noch vernachlässigbar kleine Flächen zwischen dem Temperaturverlauf und der Sollwertlinie eingeschlossen!

Abbildung 5.15 zeigt den Verlauf von drei PI-Regelungen mit gleichen K_p- aber unterschiedlichen K_i-Werten im direkten Vergleich. Hierbei wurde auch wieder ein Stellgrößenoffset von $y_\mathrm{o} = 20\,\%$ verwendet. Zum Zeitpunkt $t = 0$ wurden die Regelungen ausgehend von einer Raumtemperatur von ca. 20 °C gestartet. Um die wesentlichen Details im Regelverlauf besser erkennen zu können, wird im Diagramm allerdings der Aufheizvorgang, während dessen alle drei Verläufe aufgrund des verhältnismäßig großen K_p-Wertes sich in der Übersteuerung bei $y = 100\,\%$ befanden und synchron verliefen, ausgeblendet und nur der Regelgrößenbereich zwischen 40 °C und 56 °C dargestellt.
Im Vergleich zur reinen I-Regelung aus Abbildung 5.14a ist zu erkennen, dass bei den beiden Verläufen mit den Parametern $K_\mathrm{i} = 0{,}05\,\frac{1}{\mathrm{s}}$ und $K_\mathrm{i} = 0{,}10\,\frac{1}{\mathrm{s}}$ das Einpendeln um den Sollwert und das Erreichen des Toleranzbandes nach deutlich kürzerer Zeit erfolgt. Obwohl die K_i-Werte deutlich größer gewählt worden sind als bei der reinen I-Regelung, fällt aufgrund des zusätzlichen K_p-Wertes die Überschwingweite wesentlich geringer aus, da der Proportionalanteil bei negativer Regeldifferenz sofort zu einer starken Reduzierung der Stellgröße führt.
Der Regelverlauf mit den Parametern $K_\mathrm{p} = 24$ und $K_\mathrm{i} = 0{,}20\,\frac{1}{\mathrm{s}}$ zeigt allerdings, dass der Integrierbeiwert nicht zu groß gewählt werden darf. Hier kommt es nach dem ersten größeren Überschwingen zu einem oszillierenden Verlauf mit nur noch schwach bis kaum gedämpfter Amplitude, welcher stark an das Verhalten einer unstetigen (Zweipunkt-)Regelung erinnert. Der Grund hierfür ist darin zu suchen, dass bei größer werdenden K_i-Werten kleinere Regeldifferenzen ausreichen, um schon nach kurzer Integrationszeit die Grenzen der Übersteuerung bei $y = 100\,\%$ bzw. $y = 0\,\%$ zu erreichen.

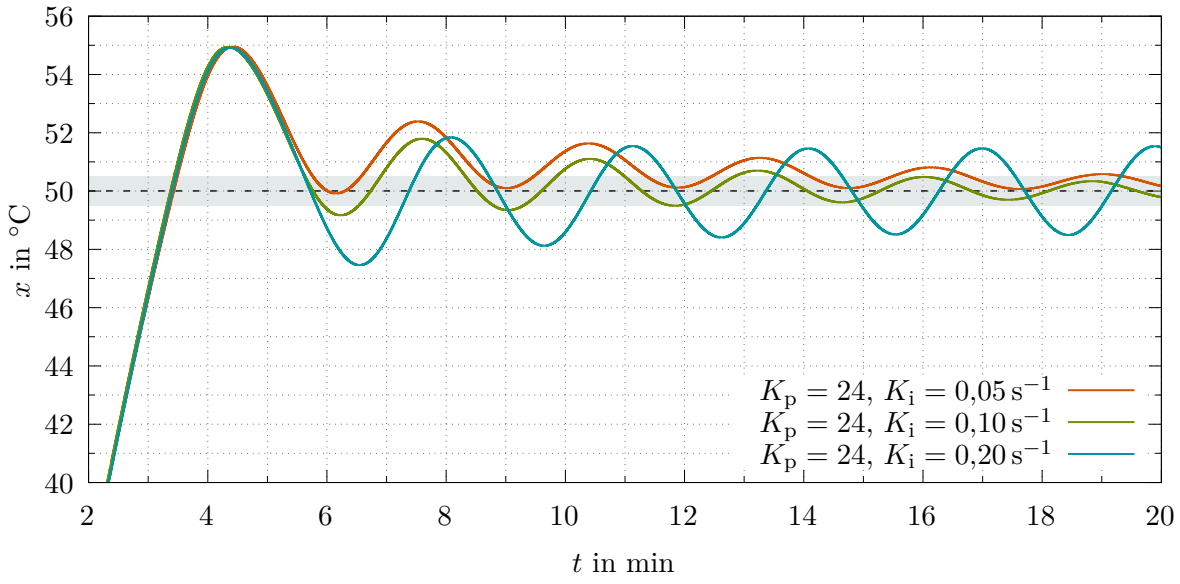


Abbildung 5.15: *zeitlicher Verlauf von PI-Regelungen für verschiedene K_i-Werte im Vergleich*

5.5.6. PID-Temperaturregelung

Zur Umsetzung einer PID-Temperaturregelung wird nun noch ein D-Anteil mit dem Ziel ergänzt, das Schwingungsverhalten der Regelgröße weiter zu dämpfen und das Toleranzband insgesamt schneller zu erreichen. Abbildung 5.16 zeigt hierzu exemplarisch drei aufgenommene Regelverläufe. Dabei wurde wiederum bei allen Messreihen ein Stellgrößenoffset von $y_\mathrm{o} = 20\,\%$ und ein konstanter Proportionalbeiwert von $K_\mathrm{p} = 24$ verwendet. Der Aufheizvorgang wurde auch hier ausgeblendet, da die Regelverläufe ausgehend von Raumtemperatur aufgrund der Übersteuerung zunächst wieder synchron verliefen.
Vergleicht man den Verlauf, welcher mit den Parametern $K_\mathrm{p} = 24$, $K_\mathrm{i} = 0{,}2\,\frac{1}{\mathrm{s}}$ und $K_\mathrm{d} = 600\,\mathrm{s}$ aufgenommen worden ist, mit jenem aus Abbildung 5.14a mit den Parametern $K_\mathrm{p} = 24$ und $K_\mathrm{i} = 0{,}2\,\frac{1}{\mathrm{s}}$, so kann man klar erkennen, dass der zusätzliche D-Anteil die Oszillationen stark dämpft. Damit fällt

die Überschwingweite deutlich geringer aus, die Regelgröße nähert sich fast asymptotisch von oben dem Sollwert[1] und erreicht nach ca. 6,5 min dauerhaft das Toleranzband.
Wird der Integrierbeiwert auf $K_\mathrm{i} = 0{,}5\,\frac{1}{\mathrm{s}}$ erhöht, so ergibt sich wieder eine stärkere Schwingung im Regelgrößenverlauf, welche aber ebenfalls durch den D-Anteil ausreichend schnell gedämpft wird, sodass das Toleranzband auch nach ca. 6,5 min dauerhaft eingehalten werden kann. Eine Erhöhung des Differenzierbeiwertes auf $K_\mathrm{d} = 900\,\mathrm{s}$ dämpft diese Oszillation weiter und die Regelgröße verläuft dadurch nach ca. 5,6 min vollständig im Toleranzband. Durch geschicktes Anpassen und Abstimmen der drei Parameter wäre sicher noch eine weitere Optimierung, z. B. hin zu einem aperiodischen Grenzverhalten, möglich.

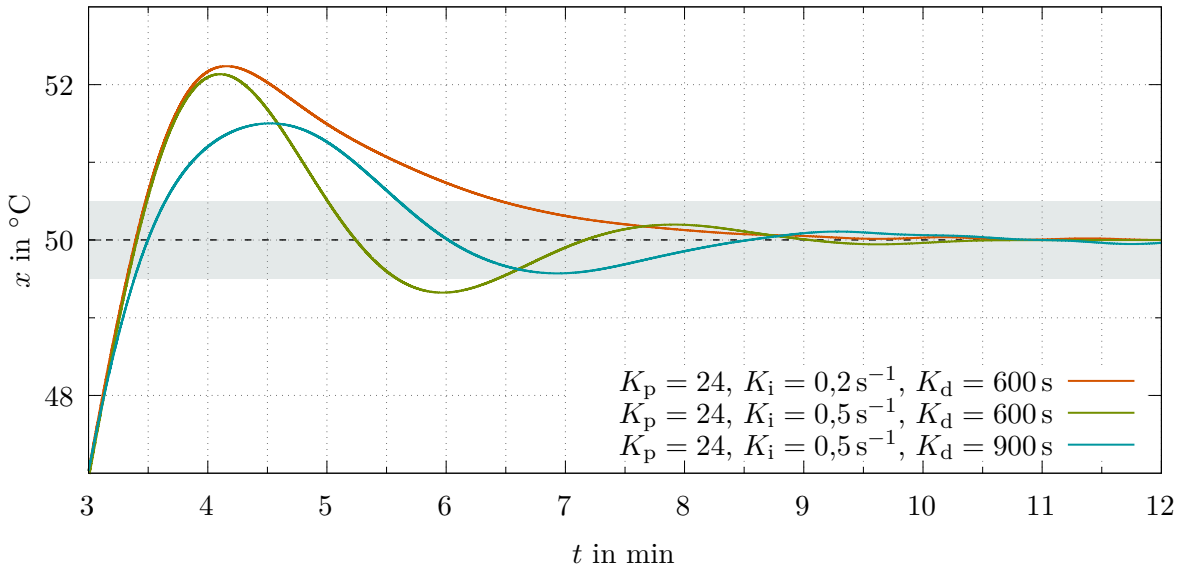


Abbildung 5.16: *zeitlicher Verlauf von PID-Regelungen bei unterschiedlichen Regelparametern*

5.6. Zusammenfassung und Anregungen für den Unterricht

Es wurde gezeigt, wie ein PID-Algorithmus unter Verwendung der Grundrechenarten auf einem Mikrocontroller umgesetzt werden kann, sofern eine Limitierung durch eine begrenzte Abtastrate tolerierbar ist. Dies ist bei langsamen Temperaturregelstrecken aber meist der Fall. Das mathematische Modell kann durch die Lernenden möglichst selbst umgesetzt werden und erfordert im Vergleich zu unstetigen Regelungen nur wenige Codezeilen mehr. Das Problem der Übersteuerung und der Stellwertbegrenzung muss dabei allerdings beachtet werden. Eine Lernsituation könnte dann durch die

realisiert werden. Neben der Programmierung und dem zielgerichteten Testen mit dem Sprungantwortverfahren kann hier wiederum die Optimierung der Regelung eine spannende Aufgabe sein. Eine ausführliche Informations- und Planungsphase, in welchen auch Analogien zu anderen Modellen betrachtet und Gleichgewichtszustände diskutiert werden, können zum Verständnis von positiven und negativen Eigenschaften der Regelanteile beitragen. Damit kann in der Ausführungsphase dann auch ein zielgerichtetes Herantasten an die optimalen Regelparameter erfolgen. Grafische Darstellungen und Gegenüberstellungen von Regel- und Stellgrößenverläufen ermöglichen zudem noch einen direkten Blick auf die Dynamik von Regelverläufen und können zur Kontrolle und Beurteilung dienen.

[1] Verbleibende minimale Oszillationen um den Sollwert herum könnten auch durch Störungen in Form von Schwankungen der Raumtemperatur und durch die begrenzte Abtastrate und Auflösung der Temperaturmessung bedingt sein.

5.7. Ideen für Aufgabenstellungen

- Beschreiben Sie, wie sich eine Verdopplung im Sprung der Regeldifferenz auf den Verlauf der Sprungantwort des PID-Reglers aus Abbildung 5.10 auswirkt.
- Stellen Sie grafische Darstellungen der Sprungantworten von PID-Reglern mit unterschiedlichen Werten für die Beiwerte K_p, K_i und K_d systematisch gegenüber[1].
- Passen Sie Code 5.1+5.2 so an, dass die Anstiegsantwort eines PI-Reglers zu Testzwecken aufgenommen werden kann[2].
- Ermitteln Sie den Verlauf der Anstiegsantworten von I-, PI- und PID-Reglern.
- Schreiben Sie Code 5.3+5.4 so um, dass anstelle der Beiwerte K_i und K_d die Variablen T_i und T_d für die Nachstellzeit und die Vorhaltzeit zur Parametrisierung herangezogen werden können.
- Berechnen Sie die Lage des Proportionalbereichs (obere und untere Grenztemperatur) einer P-Regelung unter Berücksichtigung der folgenden Rahmenbedingungen:

  Messumfang $R = 150\,^\circ\mathrm{C}$, Sollwert $w = 80\,^\circ\mathrm{C}$, Stellgrößenoffset $y_\mathrm{o} = 50\,\%$ und $K_\mathrm{p} = 15$
- Ermitteln Sie, wie sich die Lage des Proportionalbereichs einer P-Regelung verändert, wenn der Proportionalbeiwert K_p halbiert wird.
- Untersuchen Sie P-Regelungen mit verschiedenen K_p-Werten und beurteilen Sie das Auftreten einer bleibenden Regeldifferenz.
- Untersuchen Sie, inwiefern durch Einstellung eines Arbeitspunktes/Offset eine P-Regelung hinsichtlich der bleibenden Regeldifferenz optimiert werden kann.
- Untersuchen Sie den Einfluss des D-Anteils innerhalb von PD-Regelungen auf ein Schwingverhalten der Regelgröße.
- Untersuchen Sie für I- und PI-Regelungen, wie sich eine Vergrößerung des Inegrierbeiwertes K_i auf den Verlauf der Regelgröße auswirkt.
- Bei einem Sinterprozess muss eine Temperatur erreicht werden, welche ca. 20 % unterhalb der Schmelztemperatur des zu sinternden Materials liegt. Ein (zu großes) Überschwingen über den Sollwert hinaus darf dabei keinesfalls auftreten, während die Aufheizzeit bis zum Erreichen der Solltemperatur nur eine untergeordnete Rolle spielt.

  Erklären Sie, welche Art von Regelung sich für diese Aufgabe eignen könnte.
- Optimieren Sie eine PID-Regelung durch systematische Variation der drei Anteile und beurteilen Sie die Regelverläufe jeweils ausgehend von Umgebungstemperatur nach Bestimmung der An- und Ausregelzeit, der maximalen Überschwingweite und der bleibenden Regeldifferenz für unterschiedlich große Toleranzbänder.
- Der PID-Algorithmus soll so angepasst werden, dass der I-Anteil nur dann verändert wird, falls der P-Anteil alleine nicht schon für eine Übersteuerung sorgt. Passen Sie Code 5.3+5.4 entsprechend an und untersuchen Sie den Einfluss, insbesondere auf das Schwingungsverhalten der Regelgröße.

[1] Alternativ können auch die Parameter T_i und T_d herangezogen werden.

[2] *Es kann an geeigneter Stelle die Zeile* `w += dw;` *ergänzt werden, wobei innerhalb der Interruptfunktion nun der Wert der Variable* `dw` *verändert werden kann. Dieser sollte zunächst bei null liegen und durch den Interrupt auf einen Wert eingestellt werden, welcher festlegt, wie schnell sich der Sollwert und damit auch die Regeldifferenz in einem Zeitintervall verändert.*

A. Verwendung eines alternativen Temperatursensors

Die Temperaturabhängigkeit eines elektrischen Widerstands kann zur Konstruktion eines Temperatursensors ausgenutzt werden. Bei einem Pt100-Sensor handelt es sich um einen Platinwiderstand, welcher meist aus einer dünnen Leiterbahn so hergestellt wird, dass er bei einer Temperatur von 0 °C einen nominalen Widerstand von genau 100 Ω besitzt. Es werden für Temperaturmessungen auch Platinwiderstände mit anderen nominalen Widerständen, z. B. 1000 Ω bei 0 °C hergestellt. Entsprechend spricht man dann von einem Pt1000. Solche Platinwiderstände weisen einen positiven Temperaturkoeffizienten (PTC, engl. positive temperature coefficient) auf und zeigen eine in weiten Bereichen annähernd lineare Kennlinie. Zur praktischen Verwendung werden sie in Schutzmänteln mit vorkonfektionierten elektrischen Leitungen angeboten, wie es beispielsweise in Abbildung A.1a gezeigt ist.
Um die temperaturabhängige Widerstandsänderung genau messen zu können, wird ein PTC-Widerstand, wie beispielsweise der Pt100, oft in einer Wheatstoneschen-Messbrücke zusammen mit Referenzwiderständen verschaltet. Die Brückenspannung wird dann verstärkt und mit einem hochauflösenden Analog-Digital-Umsetzer (ADU, engl. ADC für analog to digital converter) in ein übertragbares Signal umgewandelt. Im Handel sind zur Verwendung mit einem Mikrocontroller passende Controller-Boards verfügbar. Abbildung A.1b zeigt ein solches Controller-Board, wie es von Adafruit Industries[1] angeboten wird. Der Pt100 kann hier direkt mit zwei Leitungen angeschlossen werden oder für genauere Messungen zur Kompensation der Leitungswiderstände auch in einer Drei- oder Vierleiterschaltung verbunden werden, wobei entsprechende Anpassungen auf dem Controller-Board durch Setzen von Lötpunkten notwendig werden, was in der zugehörigen Dokumentation detailliert beschrieben ist.

(a) *Pt100-Sensors im Schutzmantel mit Kabel* (b) *`MAX31865`-Controller-Board*

Abbildung A.1: *Verwendung eines Pt100-Temperatursensors mit einem `MAX3185`-Controller-Board*

In Code A.1 ist dargestellt, wie ein Pt100 zusammen mit einem `MAX31865`-Controller-Board und einem Arduino genutzt werden kann. Hierzu wird zunächst in Zeile 1 die Bibliothek `Adafruit_MAX31865.h` eingebunden. In den Zeilen 2 bis 5 werden Konstanten für die Pin-Nummern deklariert, über welche der Arduino mit dem Controller-Board verbunden wird. Anschließend kann in Zeile 6 dann ein `Adafruit_MAX31865`-Objekt mit dem Namen `sensor` erzeugt werden. In Zeile 7 wird eine Konstante `Rref` mit dem Wert des auf dem Controller-Board verbauten Referenzwiderstands und in Zeile 8 eine weitere Konstante `Rnominal` mit dem Nominalwert des verwendeten Platinwiderstands deklariert. Innerhalb der `setup()`-Funktion muss das zuvor deklarierte Objekt mit dem Namen `sensor` noch über die Methode `begin()` initialisiert werden. Dies geschieht hier in Zeile 16, wobei die Art der Verschaltung anzugeben ist. Im gezeigten Fall handelt es sich um eine Dreileiterschaltung, was über den Parameter `MAX31865_3WIRE` der Methode übergeben wird.

[1] `https://cdn-learn.adafruit.com/downloads/pdf/adafruit-max31865-rtd-pt100-amplifier.pdf` Abfragedatum: 22.07.2022

In der `loop()`-Funktion ist es nun möglich mithilfe der Methode `readRTD()` in Zeile 23 den aktuellen Sensorwert abzufragen. Über die Methode `temperature(Rnominal, Rref)`, welche die Werte des Nominal- und des Referenzwiderstands als Parameter benötigt, wird dann in Zeile 24 der Temperaturwert in °C berechnet und der Variablen `x` zugewiesen.

Code A.1.: *Pt100-Temperatursensor mit MAX38165 resistance to digital controller*

```
1  #include <Adafruit_MAX31865.h>
2  const int CS = 8;                 // Chip-Select-Pin
3  const int DI = 9;                 // Digital-Input-Pin
4  const int DO = 10;                // Digital-Output-Pin
5  const int CLK = 11;               // Clock-Pin
6  Adafruit_MAX31865 sensor = Adafruit_MAX31865(CS, DI, DO, CLK);
7  const float Rref = 430.0;         // Referenzwiderstand auf der Platine in Ohm
8  const float Rnominal = 100.0;     // Messwiderstand in Ohm
9
10 const float dt = 0.5;              // Zeitintervall in s
11 float t = 0.0;                     // aktuelle Zeit
12 float x = 0.0;                     // aktuelle Temperatur in °C
13
14 void setup() {
15   Serial.begin(9600);
16   sensor.begin(MAX31865_3WIRE);   // Initialisierung als Dreileiterschaltung
17 }
18
19 void loop() {
20   if (millis()/1000.0 - t >= dt) {
21     t = millis()/1000.0;
22
23     sensor.readRTD();
24     x = sensor.temperature(Rnominal, Rref);
25
26     Serial.print(t, 3);
27     Serial.print('\t');
28     Serial.println(x, 3);
29   }
30 }
```

Ausblick: Im Vergleich zu einem `DS18B20`-Sensor (Abschnitt 2.3) verspricht die Temperaturmessung über einen Pt100-Widerstand mithilfe eines `MAX31865`-Controller-Boards kürzere Messzeiten bei gleichzeitig höherer Auflösung. Der Einfluss auf die Güte insbesondere von stetigen Temperaturregelungen soll zukünftig in einer überarbeiten Version dieser Arbeit untersucht werden.

B. Nutzung einer vorgefertigten PID-Bibliothek

Code B.1.: *PID-Temperaturregelung unter Verwendung einer Arduino-Bibliothek*

```
#include <PID_v1.h>
#include <OneWire.h>
#include <DallasTemperature.h>
const int sensorPin = 2;
OneWire oneWire(sensorPin);
DallasTemperature sensors(&oneWire);
const int gatePin = 5;

const float Kp = 50;
const float Ki = 3;
const float Kd = 10;

float t = 0;
float w, x, y;

PID myPID(&x, &y, &w, Kp, Ki, Kd, DIRECT);

void setup() {
  Serial.begin(9600);
  sensors.begin();
  sensors.requestTemperatures();
  x = sensors.getTempCByIndex(0);
  myPID.SetMode(AUTOMATIC);
  myPID.SetOutputLimits(0, 100);
  pinMode(gatePin, OUTPUT);
}

void loop() {
  w = 50.0;
  t = millis() / 1000.0;
  sensors.requestTemperatures();
  x = sensors.getTempCByIndex(0);

  myPID.Compute();
  analogWrite(gatePin, 255 * y / 100.0);

  Serial.print(t, 1);
  Serial.print('\t');
  Serial.print(x, 3);
  Serial.print('\t');
  Serial.println(y, 4);
}
```

Code B.1 zeigt eine Möglichkeit, eine PID-Temperaturregelung unter Verwendung der Arduino PID Bibliothek von Brett Beauregard[1] zu implementieren. Ein Blick in deren Quellcode zeigt, dass keine Normierung vorgenommen wird, sodass in den Parametern K_p, K_i und K_d auch die Einheiten der Regel- und Stellgröße enthalten sein können. Daher ist bei Anwendung auf die im Hauptteil dieser Arbeit verwendeten Regelstrecke im Vergleich zu Code 5.3+5.4 auch von anderen Zahlenwerten zur Erzielung eines vergleichbaren Regelverhaltens auszugehen.

[1] `https://playground.arduino.cc/Code/PIDLibrary/` Abfragedatum: 20.02.2022

C. Alternative bzw. erweiterte Temperaturregler-Designs

C.1. Umsetzung einer Temperaturregelung mit einem Peltier-Element

Peltier-Elemente sind spezielle Halbleiterbauteile. Sie erzeugen eine Temperaturdifferenz zwischen ihrer Vorder- und Rückseite, wenn ein elektrischer Strom durch sie fließt. Der Temperaturunterschied vergrößert sich dabei mit wachsender Stromstärke. Sie können sowohl zum Heizen als auch zum Kühlen verwendet werden, wobei die Stromrichtung darüber entscheidet, welche der beiden Seiten warm und welche kalt wird. Oft findet man auch die Abkürzung TEC (engl. thermoelectric cooler) für Peltier-Elemente.
Abbildung C.1 zeigt ein Peltier-Element als Stellglied, welches zwischen zwei Kühlkörper aus Metall geklemmt ist. Diese bilden zu Testzwecken eine Temperaturregelstrecke, wobei noch ein `DS18B20`-Temperatursensor mit einem dieser Kühlkörper in Kontakt gebracht wird.

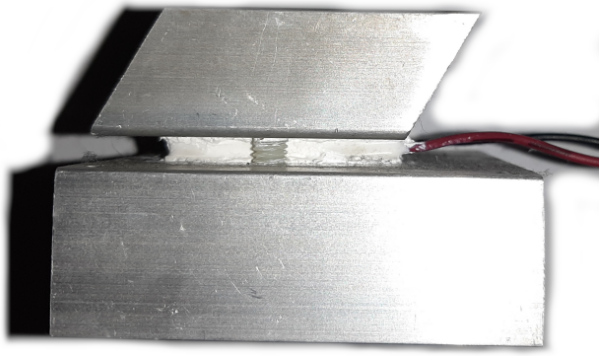
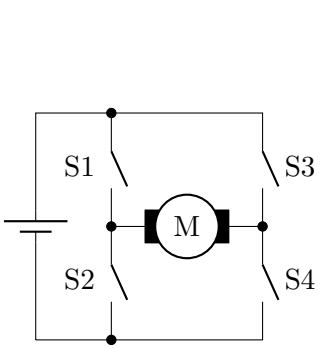
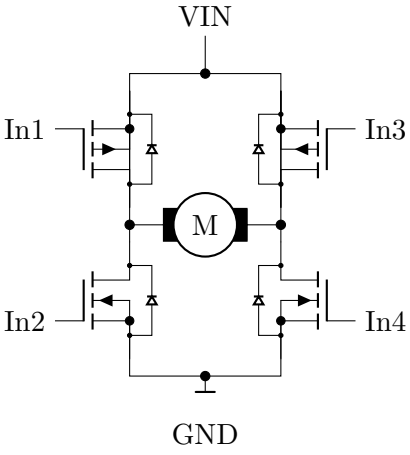
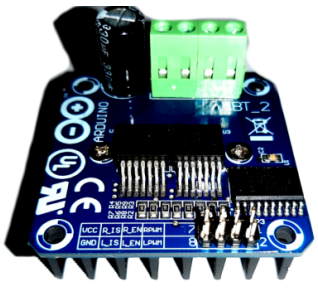


Abbildung C.1: *Peltier-Element zwischen zwei Kühlkörpern*

Um die Richtung des Stromflusses durch das Peltier-Element gezielt beeinflussen zu können, auch wenn nur eine einzelne Gleichstromquelle zur Verfügung steht, kann eine H-Brücke verwendet werden. H-Brücken werden oft auch zusammen mit Gleichstrommotoren verwendet. Das Prinzip ist zunächst in Abbildung C.2a dargestellt. Sind die Schalter S1 und S4 geschlossen und die Schalter S2 und S3 offen, so fließt der Strom von links nach rechts durch den Motor. Sind S2 sowie S3 geschlossen und S1 sowie S4 offen, ist es genau umgekehrt. Es ist zu beachten, dass S1 und S2 und/oder S3 und S4 nicht gleichzeitig geschlossen sein dürfen, da ansonsten die Stromquelle kurzgeschlossen wäre.







(a) *Realisierung mit Schaltern* (b) *Realisierung mit MOSFETs* (c) *Arduino-kompatible H-Brücke*

Abbildung C.2: *Aufbau einer H-Brücke*

Anstelle einfacher Schalter können auch MOSFETs in der H-Brücke verbaut werden (siehe Abb. C.2b). Die einzelnen Gate-Pins können dann z. B. von einem Arduino über ein PWM-Signal angesprochen werden, sodass nicht nur die Stromrichtung sondern auch die elektrische Leistung gezielt reguliert werden kann.
Im Handel ist eine Vielzahl an Arduino kompatiblen H-Brücken erhältlich, welche in der Regel für die Verwendung mit Gleichstrom- bzw. Schrittmotoren vorgesehen sind. Sie lassen sich aber auch mit Peltier-Elementen einsetzen, wenn man die maximal zulässigen Ströme und Spannungen beachtet. Abbildung C.2c zeigt das Beispiel einer H-Brücke mit der Modellnummer `BTS7960`. Wie diese mit einem Arduino für eine Dreipunktregelung verwendet werden kann, ist in Code C.1+C.2 implementiert. Aufgrund der Vielzahl an erhältlichen H-Brücken, für die zum Teil auch Programmbibliotheken zur Verfügung stehen, soll an dieser Stelle nicht detailliert auf die konkrete Verschaltung des hier verwendeten Modells eingegangen werden. Im Unterschied zur Dreipunktregelung aus Abschnitt 4.2.1 wird lediglich auf die in den Zeilen 18 bis 20 festgelegten Stellwerte mit verschiedenen Vorzeichen und die in den Zeilen 46 bis 53 implementierte, notwendige Fallunterscheidung aufmerksam gemacht. Die Codezeilen 40 bis 43 realisieren zudem die in Abbildung C.3 durch die Hystereseschleifen visualisierten Schaltbedingungen.

Code C.1.: *Dreipunktregelung mit einem Peltier-Element - Teil 1*

```
1  #include <OneWire.h>
2  #include <DallasTemperature.h>
3  #define ONE_WIRE_BUS 2    // Sensor an Pin 2
4  OneWire oneWire(ONE_WIRE_BUS);
5  DallasTemperature sensors(&oneWire);
6
7  const int reverseEnablePin = 8;
8  const int forwardEnablePin = 9;
9  const int reverseValuePin = 10;
10 const int forwardValuePin = 11;
11
12 const float dt = 1.0;     // Zeitintervall in s
13 float t = 0.0;            // aktuelle Zeit
14 float w = 50.0;           // Sollwert in °C
15 float x = w;              // aktueller Istwert in °C
16 const float dx = 0.25;    // Schaltdifferenz in °C
17
18 const int yMid = 0;
19 const int yMin = -100;
20 const int yMax = +100;
21 int y = yMid;
22
23 void setup() {
24   Serial.begin(9600);
25   sensors.begin();
26   pinMode(forwardEnablePin, OUTPUT);
27   pinMode(reverseEnablePin, OUTPUT);
28   pinMode(forwardValuePin, OUTPUT);
29   pinMode(reverseValuePin, OUTPUT);
30   digitalWrite(forwardEnablePin, HIGH);
31   digitalWrite(reverseEnablePin, HIGH);
32 }
```

Code C.2.: *Dreipunktregelung mit einem Peltier-Element - Teil 2*

```
33 void loop() {
34   if (millis()/1000.0 >= t + dt) {
35     t = millis()/1000.0;                              // aktuelle Zeit in s bestimmen
36
37     sensors.requestTemperatures();
38     x = sensors.getTempCByIndex(0);                   // aktuellen Istwert einlesen
39
40     if (y == yMax and x > w - dx) {y = yMid;}         // S1
41     else if (y == yMid and x > w + 2 * dx) {y = yMin;} // S2
42     else if (y == yMin and x < w + dx) {y = yMid;}    // S3
43     else if (y == yMid and x < w - 2 * dx) {y = yMax;} // S4
44
45     // Uebertragung der Stellgroesse zur H-Bruecke
46     if (y >= 0) {
47       analogWrite(reverseValuePin, 0);
48       analogWrite(forwardValuePin, 255 * y / 100.0);
49     }
50     else if (y < 0) {
51       analogWrite(forwardValuePin, 0);
52       analogWrite(reverseValuePin, - 255 * y / 100.0);
53     }
54
55     Serial.print(t, 1);
56     Serial.print('\t');
57     Serial.print(x, 3);
58     Serial.print('\t');
59     Serial.println(y);
60   }
61 }
```

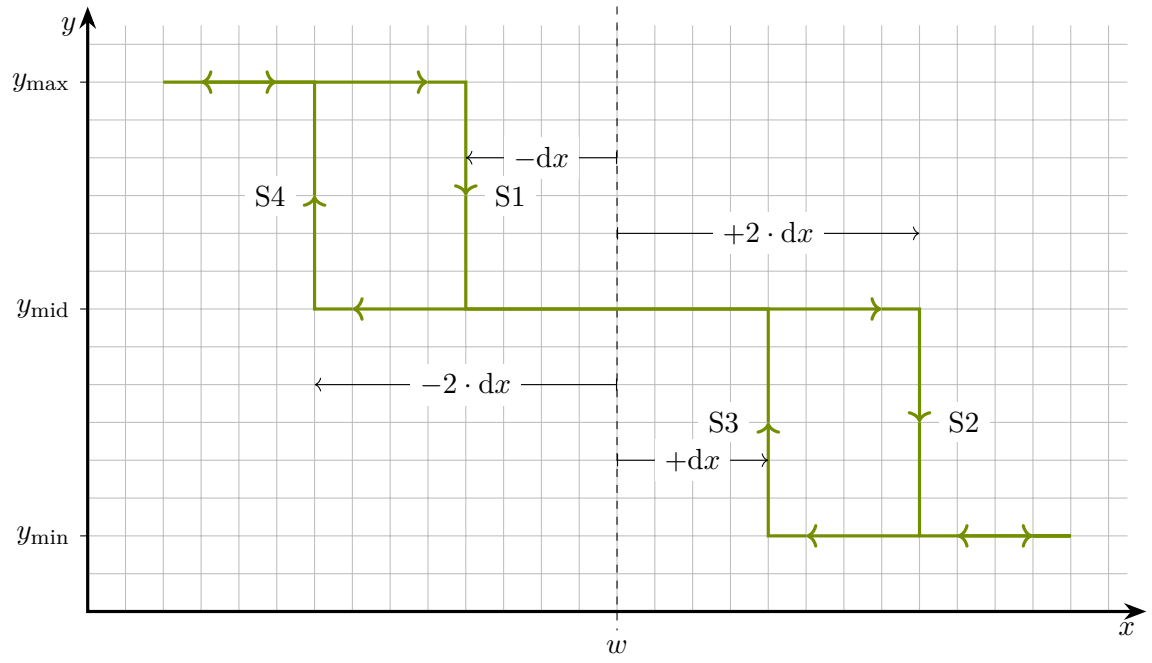


Abbildung C.3: *Hystereseschleifen zur Dreipunkttemperaturregelung mit einem Peltier-Element*

C.2. Design einer universell programmierbaren, eigenständigen Temperaturregler-Platine

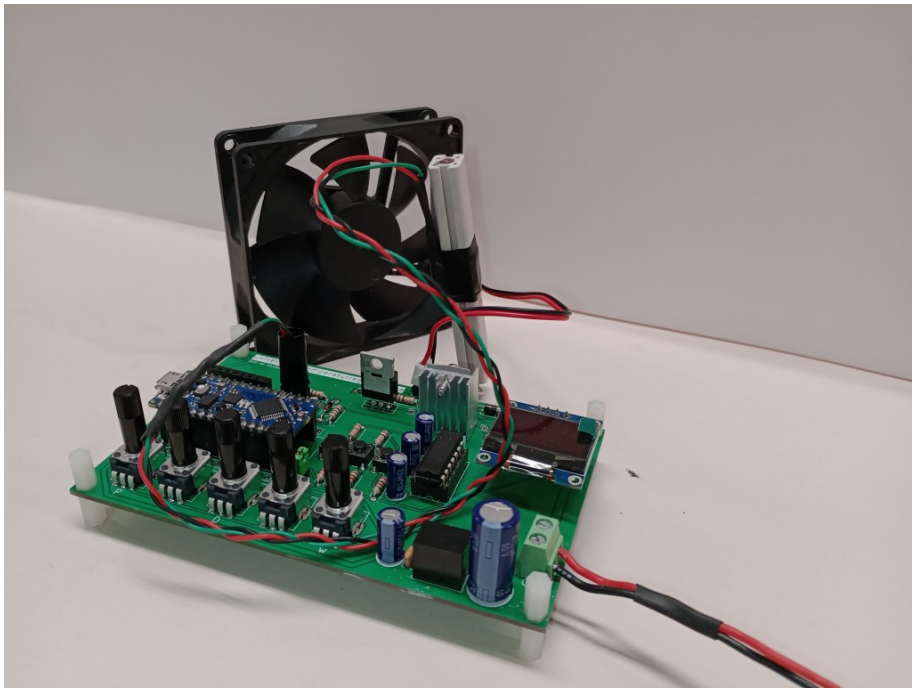


Abbildung C.4: *Temperaturregler-Platine mit angeschlossenem Heizwiderstand und Lüfter*

Abbildung C.4 zeigt eine Temperaturregler-Platine, welche passend zu den Beispielen aus dieser Arbeit entwickelt worden ist. Diese ist mit einem Arduino Nano Every bestückt, welcher relativ frei programmiert werden kann.
Die Versorgungsspannung für ein Heizelement[1] wird über Schraubklemmen angeschlossen. Diese ist zudem mit einem DC/DC-Wandler verbunden, welcher eine konstante Spannung von 9 V ausgibt und durch Setzen einer Steckbrücke auch den Arduino versorgen kann, sodass ein PC-unabhängiger Betrieb möglich ist. Außerdem liefert der DC/DC-Wandler noch ausreichend Leistung, um einen kleinen Lüfter über einen 9 V-Ausgang zu versorgen und mithilfe eines MOSFETs ansprechen zu können.
Es sind Anschlussmöglichkeiten für einen `DS18B20`-Temperatursensor und ein `MAX31865`-Controller-Board vorhanden, sodass verschiedene Messverfahren ausprobiert und gegenübergestellt werden können. Als Kontrollelemente stehen fünf Potentiometer, welche mit analogen Eingängen des Arduino verbunden sind, zur Verfügung. Zudem sind drei Taster verbaut, von denen zwei an Pins mit der Fähigkeit zu Interrupts angeschlossen sind. Ein über I^2C ansprechbares OLED-Display kann man so programmieren, dass gewünschte Informationen zu den Einstellungen des Reglers sowie zum Status des Regelbetriebs angezeigt werden.
Zudem können Buchsenleisten verlötet werden, welche den direkten Zugriff auf jeden Pin des Arduino-Boards ermöglichen, sodass individuelle Änderungen und Erweiterungen möglich bleiben. Die genaue Verschaltung der einzelnen Bestandteile kann Abbildung C.5 entnommen werden[2].

[1] im Bild ist beispielsweise ein Drahtwiderstand verschaltet
[2] Designpläne im KICAD-Format können beim Autor per E-Mail angefragt werden.

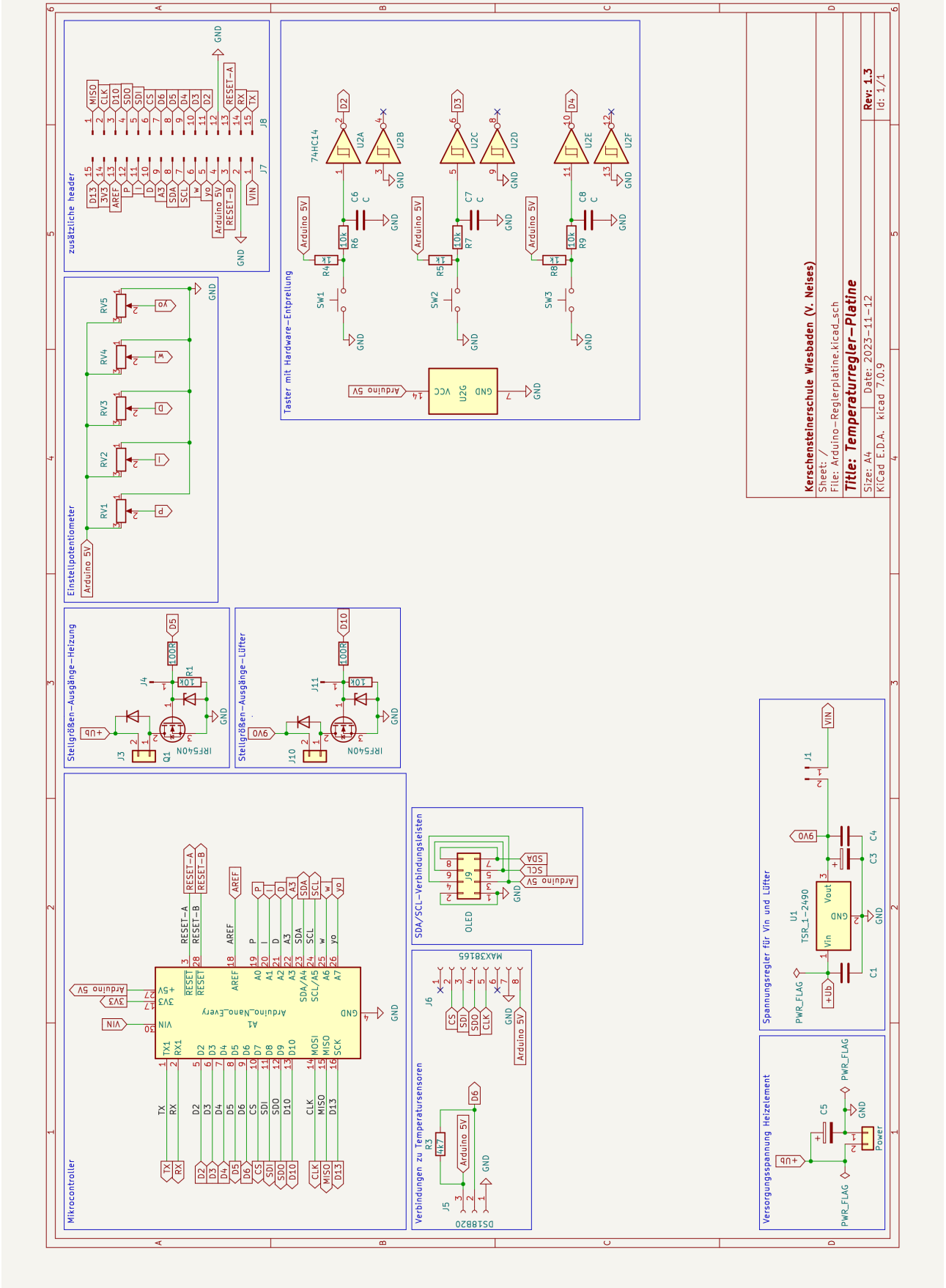


Abbildung C.5: *Schaltplan zur universellen Temperaturregler-Platine*

D. Übertragung auf andere Regelstrecken am Beispiel Ball-Balance

Der PID-Algorithmus lässt sich ohne große Änderungen auf andere Regelstrecken übertragen. Im Allgemeinen muss nur ein anderer Sensor ausgelesen und ein anderes Stellgerät angesprochen werden.

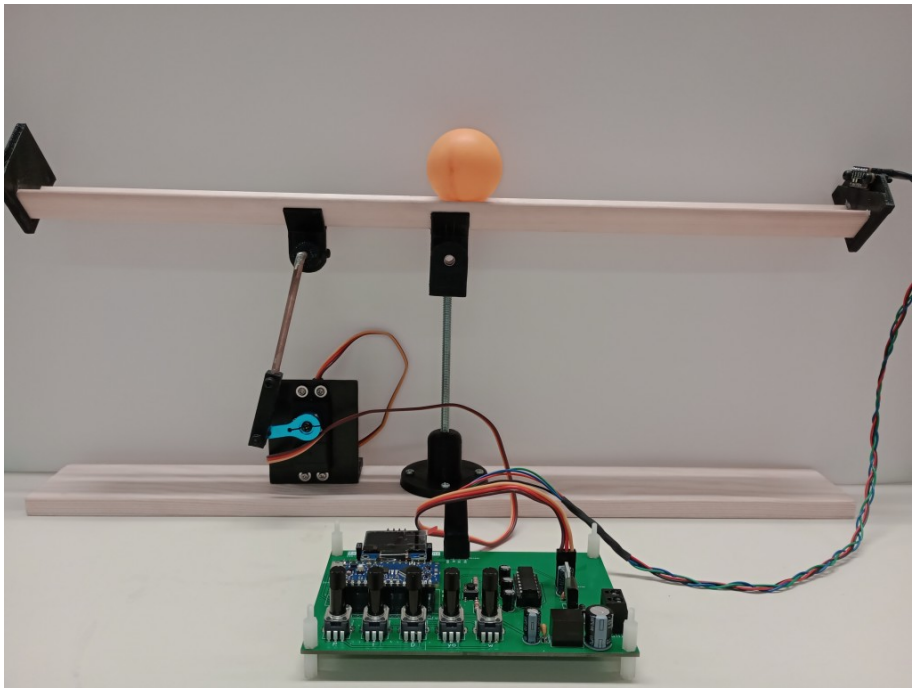


Abbildung D.1: *Positionsregelung/Balancieren eines Balls auf einer beweglichen Schiene*

Abbildung D.1 zeigt einen Aufbau zur Positionsregelung bzw. zum Balancieren eines Balls auf einer verstellbaren Schiene[1]. Als Stellgerät wird ein Servomotor verwendet. Zur Ansteuerung wird in Code D.1 in Zeile 1 die Bibliothek `Servo.h`[2] eingebunden und darüber in Zeile 2 ein Servo-Objekt mit dem Namen `myservo` erzeugt. Die Signalleitung des Servomotors wird an den PWM-fähigen Pin Nr. 9 angeschlossen, wozu in Zeile 3 eine entsprechende Konstante mit den Namen `servoPin` deklariert wird. Schließlich muss innerhalb der `setup()`-Funktion noch in Zeile 38 der `servoPin` dem Servo-Objekt `myservo` mithilfe der Methode `attach()` zugeordnet werden. Über die Methode `write()` kann dann das Stellsignal innerhalb der `loop()`-Funktion in Zeile 65 von Code D.2 zum Servomotor übertragen werden. Der verwendete Stellmotor deckt einen Stellbereich von 0° bis 180° in Schritten von 1° ab. Zur Verwendung der `write()`-Funktion muss die Stellgröße, welche zunächst in Prozent berechnet worden ist, in den entsprechenden Wertebereich umgerechnet werden.
Zur Positionsbestimmung des Balls entlang der Schiene wird der Sensor `VL53L1X` eingesetzt[3], welcher die Entfernung über die Laufzeit von nicht sichtbaren, infraroten Laserimpulsen (940 nm) bestimmt. Um den Sensor zusammen mit dem Arduino nutzen zu können, werden die Bibliotheken `Wire.h` und `VL53L1X.h`[4] in den Zeilen 5 und 6 von Code D.1 eingebunden. Die Sensordaten werden mithilfe des Übertragungsprotokolls I^2C (engl. inter-integrated circuit) an den Arduino gesendet. Dazu muss der Sensor an die Arduino-Pins SDA (engl. serial data) und SCL (engl. serial clock) als Signalleitungen

[1] Die zu sehende Platine wurde so hergestellt, dass ein Arduino samt Kontrollelementen integriert sind und ein Betrieb ohne angeschlossenen PC möglich ist. Entsprechende Designpläne im KICAD-Format können beim Autor per E-Mail angefragt werden.
[2] `https://www.arduino.cc/reference/en/libraries/servo/` Abfragedatum: 22.07.2022
[3] Erfahrungswerte des Autors haben gezeigt, dass der günstigere Sensor `VL53L0X` zur zuverlässigen Erfassung des kleinen Balls extrem genau ausgerichtet sein muss und öfter zu Ausreißern neigt, weshalb er eher nicht zu empfehlen ist.
[4] `https://github.com/pololu/vl53l1x-arduino` Abfragedatum: 22.07.2022

angeschlossen werden[1]. In Zeile 7 wird ein `VL53L1X`-Objekt mit dem Namen `sensor` erzeugt. Der Sensor kann in verschiedenen Messbereichen betrieben werden, was entsprechende Konfigurationen erfordert. Diese finden zusammen mit der Initialisierung in den Zeilen 30 bis 36 der `setup()`-Funktion statt. Für detaillierte Erklärungen sei an dieser Stelle auf die Dokumentation der Sensorbibliothek verwiesen. Zum Auslesen eines Sensorwertes wird in Zeile 45 von Code D.2 die Methode `read()` angewendet und der darüber erhaltene Wert der Variablen `x` für die Regelgröße zugewiesen.
Alle anderen Codebestandteile entsprechen der allgemeinen Implementierung des PID-Algorithmus, so wie sie auch für die Temperaturregelungen in Abschnitt 5.4 verwendet worden ist.

Code D.1.: *PID-Positionsregelung - Teil 1*

```
1  #include <Servo.h>
2  Servo myservo;
3  const int servoPin = 9;   // Servo-Pin
4
5  #include <Wire.h>
6  #include <VL53L1X.h>
7  VL53L1X sensor;
8
9  const float dt = 0.1;     // Zeitintervall in s
10 float t = 0.0;            // aktuelle Zeit in s
11 float tMem = 0.0;         // vorherige Zeit in s
12 int w = 230;              // Sollwert in mm
13 const int R = 100;        // Messumfang in mm
14 int x = w;                // aktueller Istwert in mm
15 float e = 0.0;            // aktuelle Regeldifferenz in Prozent
16 float eMem = 0.0;         // vorherige Regeldifferenz in Prozent
17
18 const float Kp = 1.2;     // Proportionalbeiwert
19 const float Ki = 0.8;     // Integralbeiwert
20 const float Kd = 1.0;     // Differentialbeiwert
21
22 float yo = 50.0;          // Stellwert-Offset beim Sollwert in Prozent
23 float yp = 0.0;           // Stellwert P-Anteil in Prozent
24 float yi = 0.0;           // Stellwert I-Anteil in Prozent
25 float dyi = 0.0;          // Aenderung I-Anteil in Prozent
26 float yd = 0.0;           // Stellwert D-Anteil in Prozent
27 float y = yo;             // Stellwert gesamt in Prozent
28
29 void setup() {
30   Wire.begin();
31   Wire.setClock(400000);
32   sensor.setTimeout(500);
33   sensor.init();
34   sensor.setDistanceMode(VL53L1X::Short);
35   sensor.setMeasurementTimingBudget(20000);
36   sensor.startContinuous(20);
37
38   myservo.attach(servoPin);
39 }
```

[1]Bei einem Arduino Uno sind diese als separate Pins zugänglich, wohingegen beim Arduino Nano die als analoge Eingänge verwendbaren Pins A4 und A5 auch für das I^2C-Protokoll genutzt werden können.

Code D.2.: *PID-Positionsregelung - Teil 2*

```
40 void loop() {
41   if (millis() / 1000.0 >= t + dt) {
42     t_prev = t;                                    // vorherige Zeit merken
43     t = millis() / 1000.0;                         // aktuelle Zeit in s bestimmen
44
45     x = sensor.read();                             // aktuellen Istwert in mm einlesen
46
47     eMem = e;                                      // vorherige Regeldifferenz merken
48     e = 100.0 * (w - x) / R;                       // aktuelle Regeldifferenz berechnen in Prozent
49
50     yp = Kp * e;                                   // P-Anteil des Stellwerts berechnen
51     yd = Kd * (e - eMem) / (t - tMem);             // D-Anteil des Stellwerts berechnen
52     dyi = Ki * e * (t - tMem);                     // I-Anteilaenderung des Stellwerts berechnen
53     yi += dyi;                                     // I-Anteilaenderung aufsummieren
54
55     // Begrenzung des I-Anteils
56     if (yo + yi < 0) {yi = -yo;}
57     else if (yo + yi > 100) {yi = yo;}
58
59     y = yo + yp + yi + yd;                         // Stellwertanteile aufsummieren
60
61     // Stellgroessenbegrenzung
62     if (y > 100) {y = 100;}
63     else if (y < 0) {y = 0;}
64
65     myservo.write(180.0 * y / 100);
66   }
67 }
```

E. Regelstrecken mit Ausgleich in der Elektrotechnik

Bei proportionalen Regelstrecken mit Ausgleich stellt sich (wie schon in Abschnitt 3.1 erwähnt) nach einer sprunghaften Veränderung der Stellgröße (als Eingangsgröße) wieder ein neuer stabiler Zustand für die Regelgröße (als Ausgängsgröße) ein. Je nach Anzahl an Energiespeichern bzw. Verzögerungsgliedern kann der zeitliche Regelgrößenverlauf dabei unterschiedlich aussehen.

E.1. Spannungsteiler als Beispiel einer proportionalen Regelstrecke nullter Ordnung

Abbildung E.1 zeigt die Schaltskizze eines einfachen Spannungsteilers aus zwei ohmschen Widerständen. Anhand der Spannungsteilergleichung E.1 kann man erkennen, dass sich die Ausgangsspannung U_a gleichzeitig mit und proportional zu der Eingangsspannung U_0 verändert, was dem Zeitverhalten einer proportionalen Regelstrecke nullter Ordnung ohne Verzögerungsglied entspricht.

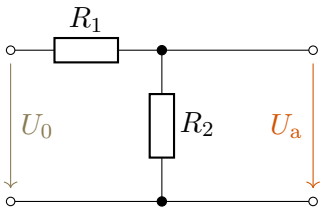
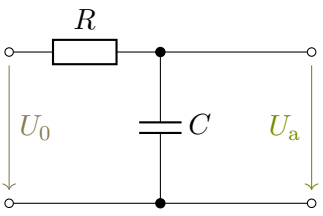


Abbildung E.1: *Schaltskizze Spannungsteiler*

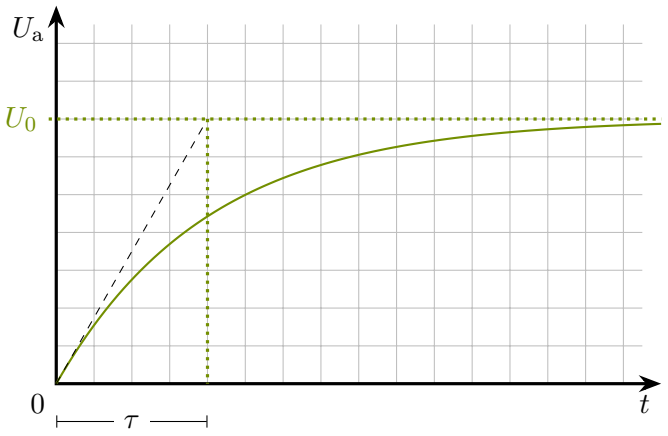
$$U_\mathrm{a} = U_0 \cdot \frac{R_2}{R_1 + R_2} \tag{E.1}$$

E.2. Ladekurve eines Kondensators (RC-Glied) als Beispiel einer proportionalen Regelstrecke erster Ordnung

Abbildung E.2a zeigt die Zusammenschaltung eines ohmschen Widerstands und eines Kondensators zu einem *RC*-Glied. Wird eine Eingangsspannung U_0 wie gezeigt angelegt, wird der Kondensator solange über einen Stromfluss durch den Widerstand geladen, bis die Ausgangsspannung U_a den Wert der Eingangsspannung angenommen hat[1] und keine weiteren Elektronen mehr in den Kondensator fließen. Der geladene Kondensator speichert in diesem Zustand dann eine Energiemenge $E = \frac{1}{2} \cdot C \cdot U_\mathrm{a}^2$.



(a) *Schaltskizze*



(b) *Ladekurve*

Abbildung E.2: *Spannung an einem RC-Glied*

Über grundlegende Gesetze der Elektrodynamik lassen sich die Gleichungen E.2 bis E.4 ableiten. Abbildung E.2b zeigt den zeitlichen Verlauf der Ladekurve sowie die grafische Bedeutung der Zeitkonstanten τ, welche maßgeblich durch die Größe des Widerstands und die Kapazität des Kondensators

[1] Dies dauert (zumindest) theoretisch unendlich lange.

beeinflusst wird. Je größer der Widerstand und je höher die Kapazität des Kondensators als Energiespeicher, desto größer wird die Zeitkonstante τ und desto flacher (bzw. verzögerter) verläuft der Anstieg der Ausgangsspannung U_a.

$$\text{Funktion:} \qquad U_\mathrm{a}(t) = U_0 \cdot \left(1 - \mathrm{e}^{-\frac{t}{\tau}}\right) \qquad \text{mit} \qquad \tau = R \cdot C \tag{E.2}$$

$$\text{erste Ableitung:} \qquad \dot{U}_\mathrm{a}(t) = \frac{\mathrm{d}U_\mathrm{a}}{\mathrm{d}t} = \frac{U_0}{\tau} \cdot \mathrm{e}^{-\frac{t}{\tau}} \tag{E.3}$$

$$\text{ausgewertet an der Stelle} \quad t = 0 \ : \qquad \dot{U}_\mathrm{a}(0) = \frac{U_0}{\tau} \cdot \mathrm{e}^{-\frac{0}{\tau}} = \frac{U_0}{\tau} \tag{E.4}$$

E.3. Hintereinander geschaltete RC-Glieder als Beispiele für proportionale Regelstrecken höherer Ordnung

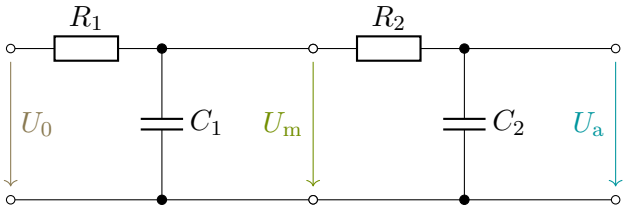


Abbildung E.3: *Hintereinander geschaltete RC-Glieder*

Schaltet man zwei *RC*-Glieder hintereinander, so stellt die Ausgangsspannung des ersten die Eingangsspannung des zweiten *RC*-Gliedes dar. Über die Gleichung zur Ladekurve eines einzelnen Kondensators erhält man durch entsprechende Kombination folgende Formel für den zeitlichen Verlauf der Ausgangsspannung.

$$U_\mathrm{a}(t) = U_0 \cdot \left(1 - \mathrm{e}^{-\frac{t}{\tau_1}}\right) \cdot \left(1 - \mathrm{e}^{-\frac{t}{\tau_2}}\right) = U_0 \cdot \prod_{i=1}^{2} \left(1 - \mathrm{e}^{-\frac{t}{\tau_i}}\right) \tag{E.5}$$

Geht man von gleichen Zeitkonstanten τ der einzelnen *RC*-Glieder aus und schaltet man eine beliebige Anzahl n zusammen, so ergibt sich:

$$U_\mathrm{a} = U_0 \cdot \left(1 - \mathrm{e}^{-\frac{t}{\tau}}\right)^n \tag{E.6}$$

In Abbildung E.4 wird Gleichung E.6 für die Werte $n = 1, 5, 10$ und 20 grafisch dargestellt, wobei der Verlauf mit $n = 1$ zur Regelstrecke erster Ordnung zum Vergleich dient. Für $n \geq 2$ ergibt sich der typische Kurvenverlauf der Sprungantwort einer proportionalen Regelstrecke höherer Ordnung, was sich insbesondere auch am Wendepunkt erkennen lässt. Je mehr *RC*-Glieder und damit auch Kondensatoren als Energiespeicher verbaut werden, desto größer wird die Verzugszeit und desto länger dauert es, bis sich die Ausgangsspannung (als Regelgröße) dem stabilen Endwert weitgehend angenähert hat.

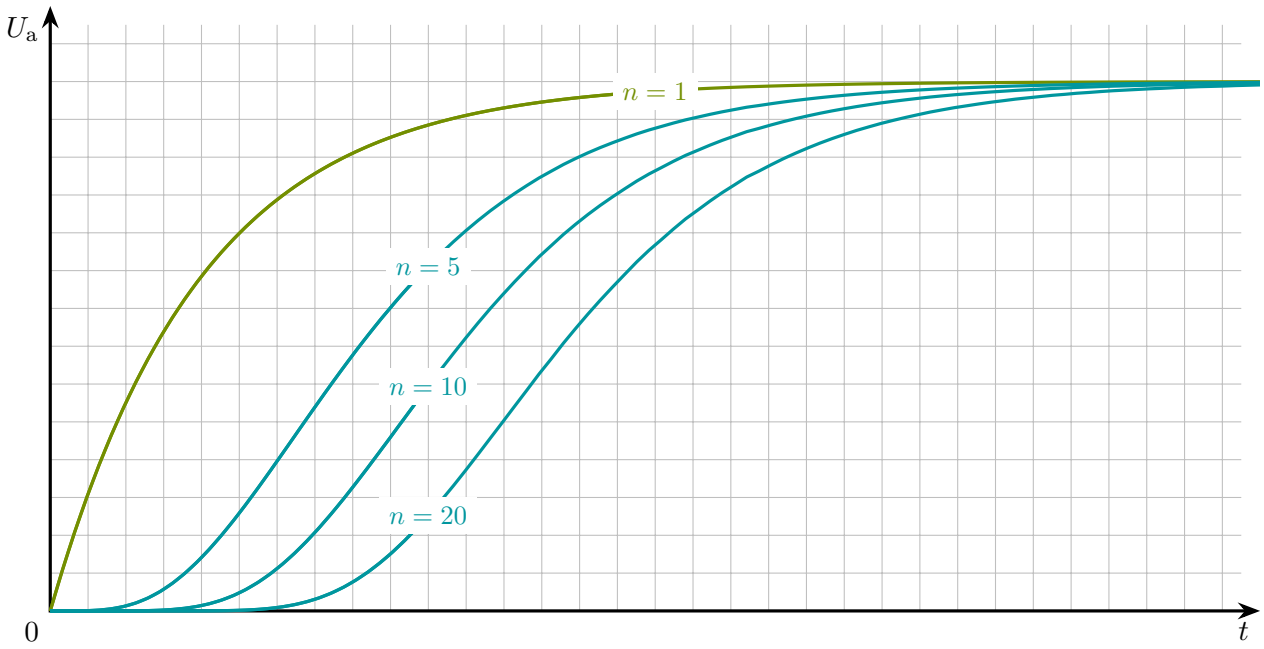


Abbildung E.4: *Kurvenschar zu Regelstrecken höherer Ordnung*

F. Realtime saving and plotting mit Python

Während einer Datenaufnahme zu einem Temperaturregelprozess kann es sehr hilfreich sein, die sich ergebenen Temperatur- und Stellgrößenverläufe live beobachten zu können. Dadurch lässt sich frühzeitig erkennen, ob eine vorgenommene Parametrisierung des Regelprozesses den gewünschten Effekt erzielt. Da die Fähigkeiten des „seriellen Plotters“ der Arduino-IDE recht eingeschränkt sind, wurde für diese Arbeit ein Python-Skript entwickelt, welches die vom Arduino über die serielle Schnittstelle übertragenen Daten direkt in einer csv-Datei abspeichert und gleichzeitig den Regel- und Stellgrößenverlauf in Echtzeit grafisch darstellt. Code F.1+F.2 zeigt den entsprechenden Quelltext, auf welchen an dieser Stelle nicht näher eingegangen wird. Das Wesentliche zur Nutzung kann der Kommentierung entnommen werden.

Code F.1.: *Python-Skript zum Speichern und zur grafischen Live-Darstellung der Daten - Teil 1*

```
1  import time, csv, serial
2  import multiprocessing
3
4  import matplotlib.pyplot as plt
5  import matplotlib.animation as animation
6
7  port = '/dev/ttyUSB0'   # diese String-Variable legt den verwendeten Port fest
8  baudrate = 9600         # diese Variable legt die verwendete Baudrate fest
9
10 ser = serial.Serial(port, baudrate)
11 ser.flushInput()
12
13 splitChar = '\t'         # Tabulator als Trennzeichen der seriellen Schnittstelle
14 delChar = '\t'           # Tabulator als Trennzeichen fuer die csv-Datei
15 fileName = 'data.csv'    # durch diese String-Variable wird der Dateiname festgelegt
16
17 def saving():
18     while True:
19         try:
20             ser_bytes = ser.readline().rstrip()
21             decoded_bytes = str(ser_bytes.decode('utf-8'))
22             dataArray = decoded_bytes.split(splitChar)
23
24             print(dataArray)
25             float(dataArray[0])
26
27             with open(fileName, 'a') as csvfile:
28                 writer = csv.writer(csvfile, delimiter = delChar)
29                 writer.writerow(dataArray)
30         except ValueError:
31             print('fehlerhafte Zeile übersprungen')
32             pass
33         except NameError:
34             print('falscher Port!?')
35             break
36         except KeyboardInterrupt:
37             print('Keyboard Interrupt')
38             break
39         except:
40             print('unerwarteter Abbruch')
41             break
```

Code F.2.: *Python-Skript zum Speichern und zur grafischen Live-Darstellung der Daten - Teil 2*

```
42 fig = plt.figure()
43 ax = fig.subplots(2)
44
45 i = 0
46 def animate(i):
47     t, x, y = [], [], []
48
49     with open(fileName, 'r') as csvfile:
50         reader = csv.reader(csvfile, delimiter = delChar)
51         for row in reader:
52             t.append(float(row[0]))
53             x.append(float(row[1]))
54             y.append(float(row[2]))
55
56     ax[0].clear()
57     ax[0].grid()
58     ax[0].set(xlabel = 'Zeit in s', ylabel = 'Temperatur in °C')
59     ax[0].plot(t, x, 'tab:blue')
60
61     ax[1].clear()
62     ax[1].grid()
63     ax[1].set(xlabel = 'Zeit in s', ylabel = 'Stellgröße in %')
64     ax[1].set_ylim(-5, 105)
65     ax[1].plot(t, y, 'tab:red')
66
67 p1 = multiprocessing.Process(target = saving)
68 p2 = multiprocessing.Process(target = animate, args = [i])
69
70 p1.start()
71 time.sleep(5)   # es werden zunaechst 5s lang Daten gesammelt bevor das Plotten startet
72 p2.start()
73
74 ani = animation.FuncAnimation(fig, animate, blit = True, interval = 1000)
75 plt.show()
76
77 p1.join()
78 p2.join()
```

Abbildung F.1 zeigt einen Screenshot einer Echtzeit-Datenaufnahme mit dem Python-Skript.

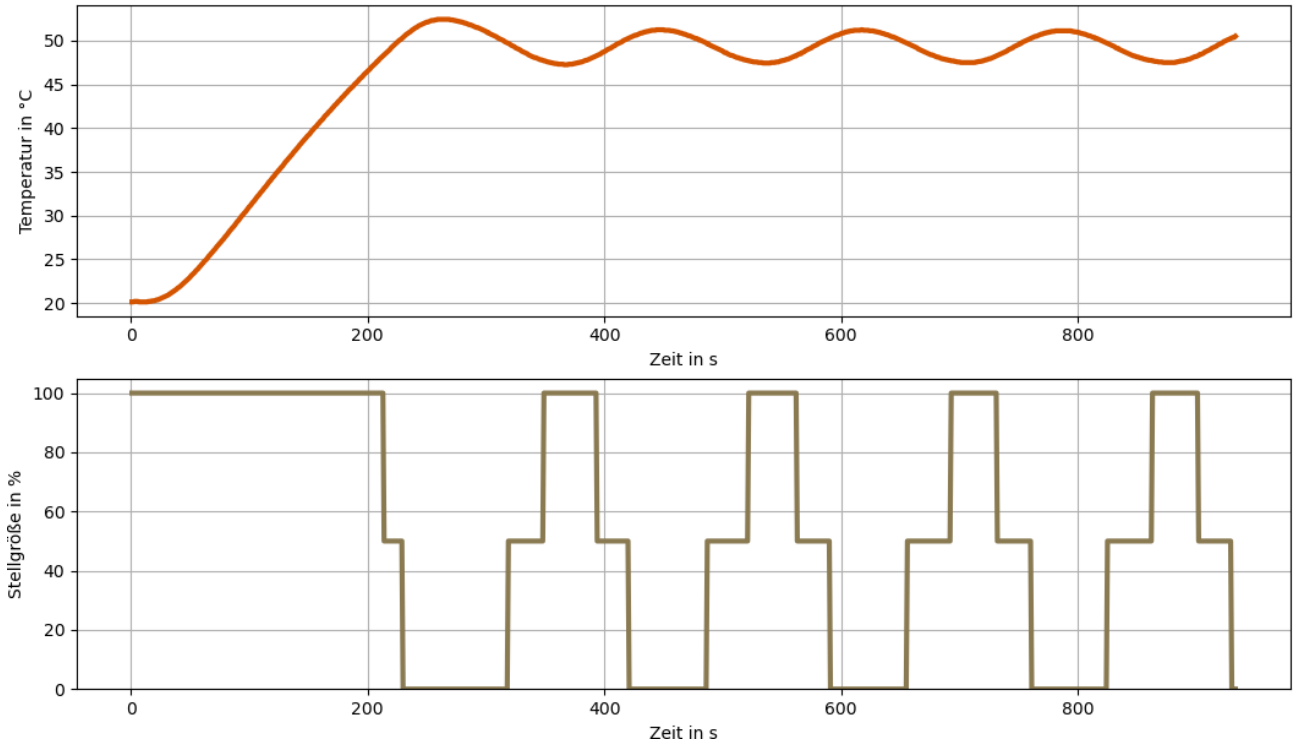


Abbildung F.1: *Screenshot einer realtime Datenaufnahme mit dem Python-Skript am Beispiel einer Dreipunkt-Temperaturregelung*

Abbildungsverzeichnis

Liste verwendeter Codes

Index