Bode-Aided-Design (BAD) eines SIPART-Temperaturreglers über ein Web Interface

Simon Heß
Duale Hochschule Baden Württemberg
Stuttgart, Deutschland
et18075@lehre.dhbw-stuttgart.de

***Abstract*— The SIPART DR20 temperature control system has been created and tested by 2 previous works so far. In [1] the prototype was built and in [2] controlled via Bode Aided Design. In this work, the measurements of the previous works are validated, the usability of the system is improved by controlling it via a MATLAB app, and a concept is presented to operate the system remotely.**

***Zusammenfassung*— Die SIPART DR20 Temperatur-Regelungsanlage wurde bisher von 2 Vorgängerarbeiten erstellt und getestet. In [1] wurde der Prototyp aufgebaut und in [2] über Bode Aided Design geregelt. In dieser Arbeit werden die Messungen der Vorgängerarbeiten validiert, die Benutzerfreundlichkeit der Anlage durch die Ansteuerung über eine MATLAB-App verbessert und es wird ein Konzept vorgestellt, mit dem die Anlage ferngesteuert bedient werden kann.**

Keywords—Bode Aided Design, MATLAB, Regelungstechnik

I. Einführung

Im Zuge der voranschreitenden Digitalisierung und Automatisierung sind IOT Anwendungen immer mehr im Trend. Außerdem befinden wir uns derzeit in einer Periode, in der das flexible mobile Arbeiten in den Fokus gerückt wird. Remote Arbeiten ist immer mehr im Geschäftsalltag vorhanden und bald nicht mehr wegzudenken. Bis 2025 wird prognostiziert, dass 22% der amerikanisch arbeitenden Bevölkerung remote Arbeiten, was einen Anstieg um 87% im Vergleich zu der Zeit vor der Pandemie darstellt. [3] Deshalb wird in dieser Arbeit ein Konzept zur remote-Bedienung der Regelanlage erstellt.

Das Bode Aided Design nach Zacher ist eine neue Methode die Regelparamater einer Strecke zu bestimmen, ohne die Übertragungsfunktion zu kennen. Man benötigt ausschließlich das Bode Diagramm, für eine erste Näherung reicht sogar ein einzelner Messpunkt. Mit der derzeitigen Anlage ist die Aufnahme nicht möglich. Deshalb wird die Anlage um eine manuelle Ansteuerung erweitert. In diesem Zuge wird ein App geschrieben, welche die Benutzung benutzerfreundlicher macht.

Das Vorgehen der Arbeit ist dadurch folgendermaßen:

Zunächst wird die Strecke aufgenommen und mit den Ergebnissen der Vorgängerarbeiten verglichen. Im Anschluss wird diese Anlage dann mittels Bode Aided Design und konventionellen Regelmethoden geregelt. Im Anschluss wird eine MATLAB App geschrieben. Und es werden 2 Konzept zur remote-Bedienung der Anlage ausgearbeitet.

II. Stand der Technik

Diese Arbeit basiert auf der Regelanlange, die in der vorhergehenden Projektarbeit [1] aufgebaut und in Betrieb genommen wurde.

Die Temperatur einer Glühlampe wird durch die Ansteuerung eines Lüfters geregelt. Bei der Lampe handelt es sich um eine 12V Osram-Autolampe. Sie kann in zwei verschiedenen Leistungsstufen betrieben werden. Zum einen mit 55W, wenn nur das Abblendlicht angesteuert wird und zum anderen mit 115W, wenn Abblendlicht und Fernlicht angesteuert werden. Der Temperatursensor TMP36 misst die Temperatur die von der Lampe erzeugt wird und diese wird dann vom Arduino eingelesen. Die gemessene Temperatur wird daraufhin an den Regler übergeben und der bestimmte Regelwert vom Arduino eingelesen. Daraufhin steuert der Arduino den Lüfter an.

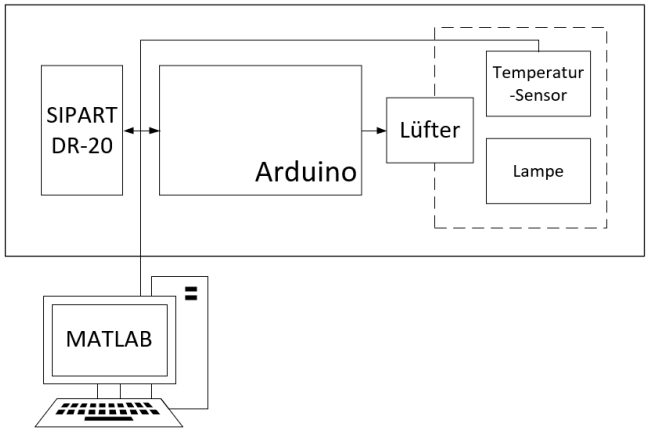

Abbildung 1. Aufbau der Regelanlage

III. Regelung der Anlage

A. Zeitprozentkennwert-Verfahren und Betragsoptimum

Zur Ermittlung der Strecke wird zunächst das Verhalten der Anlage nach einem Eingangssprung geprüft. Danach wird diese Sprungantwort nach dem Zeitprozentkennwert-Verfahren [4] analysiert und ausgewertet. Die ermittelte Strecke war:

$$G_{S1}(s) = \frac{-0{,}565}{1 + 71{,}9s}$$

Das Ergebnis der ersten Näherung weicht ziemlich von der gemessenen Strecke ab.

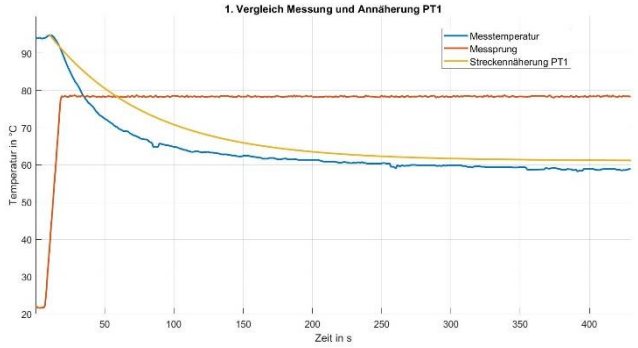

Abbildung 2. Erste Streckennäherung

Die Zeitkonstante der Strecke wird dahingegen angepasst. Dadurch ergibt sich für die Übertragungsfunktion:

$$G_{S2}(s) = \frac{-0{,}565}{1 + 40s}$$

In Abb. 3 ist zu sehen, dass die Näherung durch die Anpassung relativ exakt die Strecke trifft.

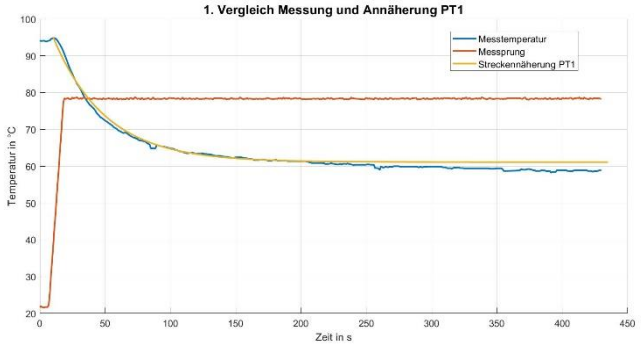

Abbildung 3. Zweite Streckennäherung

Die Parameter des PI-Reglers sind dann nach dem Betragsoptimum mit einer Ausregelzeit von $T_{aus} = 50s$:

$$G_R(s) = \frac{-5{,}522 \cdot (1 + s \cdot 40)}{s \cdot 40}$$

In Abb. 4 ist das Ergebnis der Simulation des gesamten Regelkreises zu sehen.

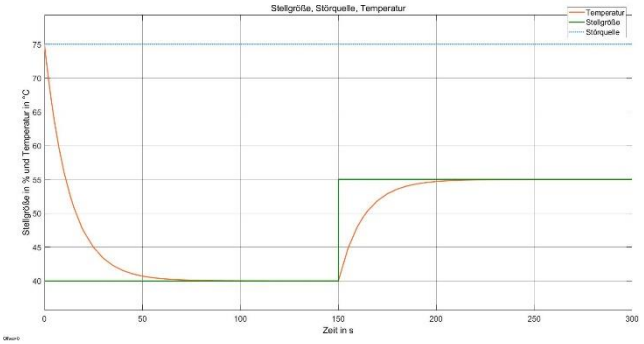

Abbildung 4. Regelkreis Simulation in MATLAB Simulink

Die Störquelle sind Konstante 75°C und soll die Glühbirne simulieren. Zu Beginn ist der Stellwert 40. Nach 150s springt dieser auf 55. Die Regelung funktioniert in der gewählten Ausregelzeit und ohne konstante Regelabweichung.

B. Bode Aided Design

Das Bode Diagramm der Strecke wird zunächst experimentell bestimmt. Dafür wird der Lüfter der Anlage (blau in Abb. 5) mit einem Sinus angesteuert. Aus der Differenz des Betrags und der Phasenverschiebung zur Temperatur (orange in Abb. 5) wird das Bode Diagramm erzeugt.

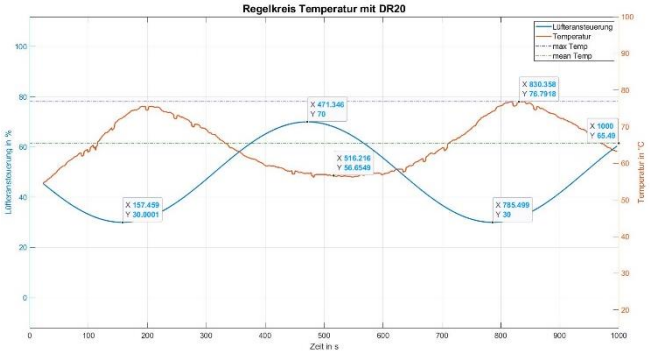

Abbildung 5. Regelkreis mit Sinus-Ansteuerung

Dabei gilt:

$$\Delta\varphi(\omega) = 180° - 360° \cdot \frac{t_2 - t_1}{T}$$

$$|G(\omega)| = 20 \cdot \log\left(\frac{x_0}{y_0}\right)$$

Das Verfahren wird mit mehreren Frequenzen wiederholt. Die ermittelten Werte werden in ein Diagramm eingetragen und nach dem Bode Aided Design [5] geregelt:

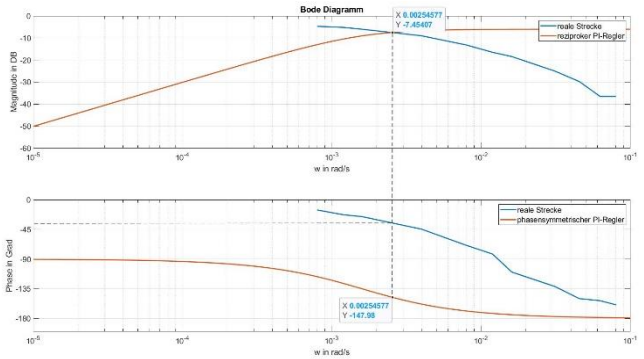

Abbildung 6. Bode Aided Design der ermittelten Strecke

Damit das System Stabil ist muss nach dem Stabilitätskriterium des Drei Bode Plots Verfahren der Phasengang der Strecke (bei der Frequenz des Schnittpunktes im Amplitudengang) über der des phasensymmetrischen Reglers liegen. Diese Bedingung ist erfüllt der Regelkreis ist stabil.

Zur Überprüfung wird das Ergebnis erneut simuliert. Die Übertragungsfunktionen sind diesmal:

$$G_S(s) = \frac{0{,}562}{(1 + 270{,}3 \cdot s)(1 + 27{,}03 \cdot s)}$$

$$G_R(s) = 2 \cdot \left(1 + \frac{1}{s \cdot 625}\right)$$

Daraus resultiert folgender Regelkreis:

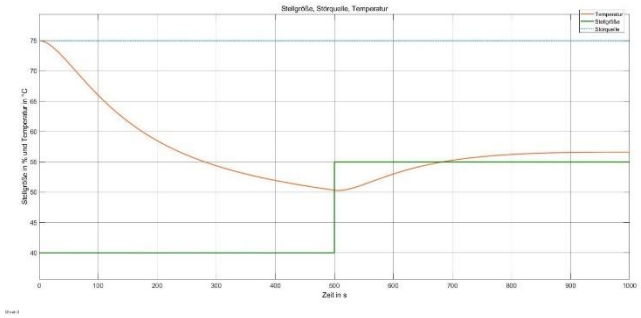

Abbildung 7. Regelkreis Simulation BAD in MATLAB Simulink

Beim Regler-Entwurf nach BAD wird keine Aussage über die Regelgeschwindigkeit getroffen. Der Fokus liegt auf stabilem Design, auch, wenn die Übertragungsfunktion der Strecke unbekannt ist. Daher ist die Regelung langsamer als nach dem Betragsoptimum.

IV. MATLAB App

In diesem Kapitel wird der Aufbau der entwickelten App beschrieben und die neuen Funktionen erklärt. Nachdem die App gestartet wird, ist der Bildschirm wie in Abb. 8 dargestellt zu sehen.

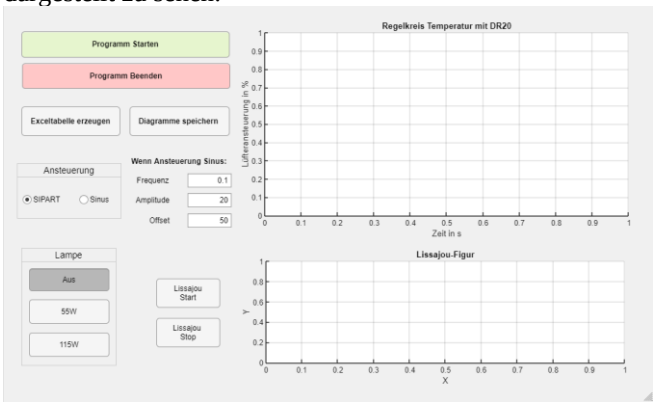

Abbildung 8. Aufbau MATLAB App

Die Funktionen der einzelnen Knöpfe werden in der folgenden beschrieben. Neben diesen Buttons gibt es noch eine Auswahl des Ansteuerungsmodus. Dieser kann entweder SIPART oder Sinus sein. Wenn Sinus ausgewählt ist kann die Frequenz, Amplitude und der Offset eingestellt werden. Wenn SIPART ausgewählt wird, wird der Lüfter über den SIPART-Regler angesteuert.

Die Lampe soll über den Buttondropdown entsprechend an und ausgeschalten werden. Diese Funktion steht noch nicht zur Verfügung.

Die Standardwerte die bei jeden „Programm Starten“ ausgewählt sind, sind SIPART Ansteuerung und Lissajous wird nicht ausgegeben.

Buttons	**MATLAB App Buttons und ihre Funktionen**
	Funktion
Programm Starten	Muss zu Beginn betätigt werden. Sobald gedrückt, beginnt die Anlage zu Regeln und die Lüfteransteuerung wird mit der Temperatur über der Zeit im oberen Graph eingetragen. Zum Neustarten des Graphs, muss die Taste erneut gedrückt werden.
Programm Beenden	Wenn betätigt stoppt die Regelung und die Ausgabe des Graphen. Falls die Lampe an ist, wird diese ausgeschalten
Exceltabelle erzeugen	Erzeugt eine Exceltabelle, mit den Werten des oberen Graphs
Diagramme speichern	Speichert die aktuellen Diagramme als MATLAB-Figuren.
Lissajou Start	Wenn betätigt, wird im unteren Diagramm ein Lissajous- Graph erzeugt.
Lissajou Stop	Wenn betätigt, wird die Ausgabe des Lissajous-Graphs beendet

Der grundlegende Programmablaufplan nach betätigen des „Programm Starten“ Buttons ist in Abb. 9 dargestellt.

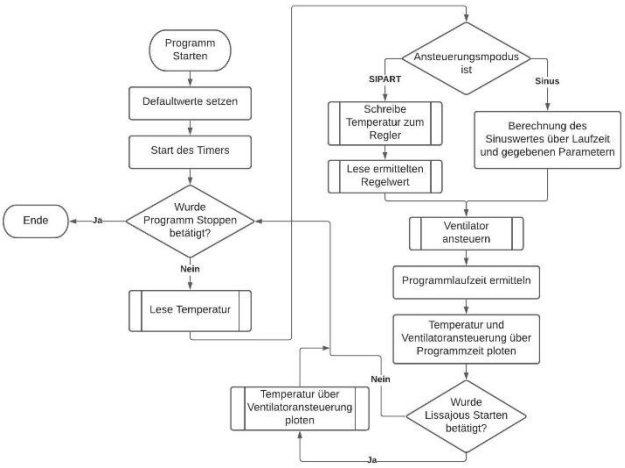

Abbildung 9. PAP „Programm Starten“ Button

Nachdem die Default-Werte gesetzt und der Timer zum ersten Mal gestartet wurde, beginnt die Schleife, die solange läuft, bis „Programm Stoppen“ betätigt wird. Zunächst wird die Temperatur über den Sensor eingelesen. Dann wird überprüft welcher Ansteuerungsmodus ausgewählt ist. Wenn SIPART ausgewählt ist, wird die Temperatur zum Regler geschrieben und der Regelwert des Reglers wieder eingelesen. Wenn Sinus ausgewählt ist, wird der Ansteuerungswert y nach folgender Formel berechnet:

$$y(t) = Amplitude \cdot \sin(Frequenz \cdot t) + Offset$$

Die Amplitude, Frequenz und der Offset sind die in der GUI eingestellten Werte.

Die Zeit t ist die Programmlaufzeit, also die Zeit die seit Start des Timers vergangen ist.

Die bestimmten Ansteuerungswerte werden zum Ventilator geschrieben. Daraufhin werden die jeweiligen Graphen erzeugt. Abb. 5 ist ein Beispiel für die Sinusansteuerung im Regelkreis Graph und Abb. 10 ist ein Beispiel für einen Lissajous-Graph.

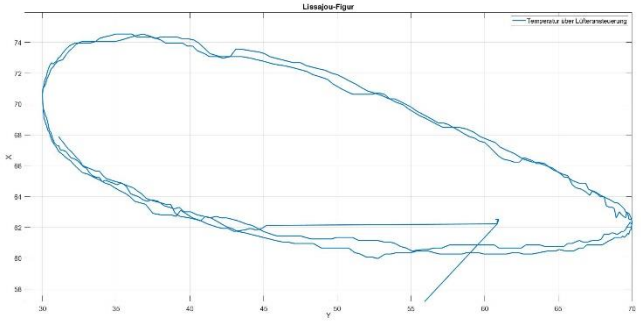

Abbildung 10. Beispiel Lissajous Graph

V. IOT KONZEPT

In diesem Kapitel werden 2 Konzepte vorgestellt, mit denen die Anlage remote bedient werden kann.
Als Server/Controller könnten Raspberry Pis verwendet werden. MATLAB bietet mit Matlab Web App Server eine Möglichkeit Matlab Apps als Web App zu benutzen und zu Teilen. Leider wird dadurch nur die Funktion gerteilt. Das heißt, dass jeder Nutzer die Anlage an seinem PC angeschlossen haben müsste. Es reicht nicht aus, die Anlage am Server angeschlossen zu haben. Daher gibt es derzeit noch keine Möglichkeit zum direkt remote bedienen der MATLAB App.

A. VNC

Die erste Methode ist simpel, aber nicht sehr elegant. Sie wird in Abb. 11 anhand einer erweiterten Darstellung von Abb. 1 erklärt.

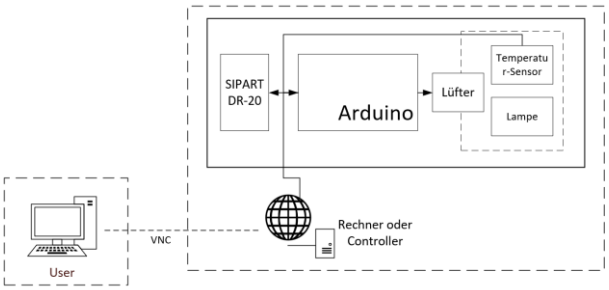

Abbildung 11. : Prinzipieller Aufbau VNC Konzept

Ein Rechner oder Controller auf dem die eine standalone-Version der MATLAB-App installiert ist wird über VNC (Virtual Network Computing) ferngesteuert.
Vorteile von diesem Konzept sind, dass die MATLAB App eins zu eins übernommen werden kann und keine Konfigurationen an der Steuerung der Regelanlage vorgenommen werden müssen. Ein Nachteil ist die unsaubere Benutzung über einen virtuellen Desktop.

B. MQTT

Das zweite Konzept ist aufwendiger in der Umsetzung. MQTT steht für „Message Queuing Telemetry Transport“ und ist ein offenes Nachrichtenprotokoll für Maschine zu Maschine Kommunikation.

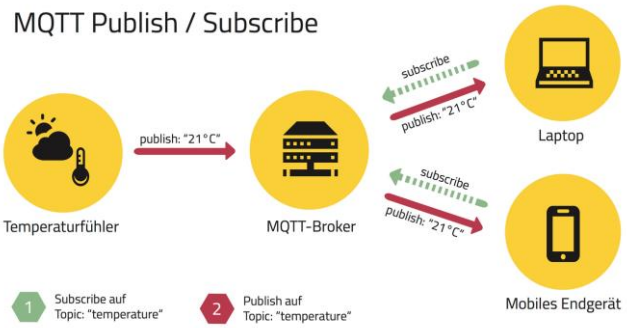

Abbildung 12. MQTT Publish/Subscribe-Architektur [6]

Es wird ein Topic erstellt, in Abb. 6.2-1 ist dieses zum Beispiel „temperatur“. Daraufhin gibt es einen Temperaturmesser, der diese Werte an den MQTT-Broker veröffentlicht. Beim MQTT-Broker handelt es sich um einen Server, der als Verteiler fungiert. Sobald die Temperatur Werte veröffentlicht werden, empfangen alle Benutzer die das Topic „temperatur“ abonniert haben diese Werte.
Auf unsere Regelanlage bezogen, können die Messwerte des Arduinos so über zum Beispiel einen Raspberry Pi übergeben werden und es können dem Arduino auch so über den Raspberry Pi Werte geschickt werden.
Vorteile sind, dass der Austausch der Daten sauber stattfindet und wirklich nur die relevanten Daten übertragen werden und nicht der komplette Bildschirm. Ein Nachteil ist aber, dass die Auswertung und Benutzung über die MATLAB App so nicht 1 zu 1 übernommen werden kann, sondern durch mehr Aufwand eingefügt werden muss. Ein möglicher Einstiegspunkt dabei wäre die Raspberry Pi Toolbox für MATLAB

VI. ERGEBNIS UND DISKUSSION

Das Ziel dieser Arbeit war es, die Strecke der gegebenen Temperaturregelanlage, unter Verwendung verschiedener Verfahren zu ermitteln und zu Regeln. Das Ergebnis wurde mit den Vorgängerarbeiten verglichen. Daraus ermittelten sich die Regelparameter über ein Standardverfahren und über das Bode Aided Design. Um Bode Aided Design verwenden zu können ist die experimentelle Bestimmung der Strecke notwendig. Dafür wurde eine Grafische Benutzeroberfläche entwickelt, die Funktionen der Vorgängerarbeiten beinhaltet, aber um diese Funktion erweitert wurde. Zum Abschluss sollte ein Konzept zur remote Steuerung der Anlage entwickelt und umgesetzt werden.
Die Ergebnisse der Strecken und Regler-Einstellung sind:

$$G_{S1}(s) = \frac{-0{,}565}{1 + 40s}$$

$$G_{R1}(s) = 5{,}522 \cdot \left(1 + \frac{1}{s \cdot 40}\right)$$

$$G_{S2}(s) = \frac{0{,}562}{(1 + 270{,}3 \cdot s)(1 + 27{,}03 \cdot s)}$$

$$G_{R2}(s) = \frac{37{,}5 \cdot (1 + s \cdot 270{,}3) \cdot (1 + s \cdot 27{,}03)}{s \cdot 270{,}3}$$

$G_{S1}(s)$ ist die nach ZPV ermittelte und angepasste Strecke. $G_{R1}(s)$ ist der PI-Regler der die Strecke $G_{S1}(s)$ nach Betragsoptimum regelt. $G_{S2}(s)$ ist die Näherung der Strecke die experimentell bestimmt wurde. Der durch BAD bestimmte Regler ist nicht aufgelistet, mit diesem ist der Regelkreis zwar stabil, aber die Regelung erfolgt sehr langsam. $G_{R2}(s)$ ist der PID-Regler, der die Strecke $G_{S2}(s)$ nach Betragsoptimum regelt.
Die Regler- und Strecken-Parameter von $G_{R1}(s)$ und $G_{s1}(s)$ ähneln denen der Vorgängerarbeiten. Es zeigt sich, dass die durch das ZPV ermittelten Streckennäherung die Strecke schlechter widerspiegeln als die des Wendetangenten-Verfahrens. Die Ergebnisse der experimentellen Streckenermittlung und dem daraus resultierendem PID-Regler wurden so vor dieser Arbeit noch nicht festgestellt.

Bisher wurde ausschließlich das Führungsverhalten betrachtet. Die Regler-Einstellung nach dem Störgrößenverhalten kann noch untersucht werden. Die Lampe ist die Störgröße des Regelkreises und damit die Komponente die variieren kann. Die Stellgröße ist bei der Temperaturregelung einer Lampe (nicht wie bei einer Klimaanlage) im Normalfall fest gewählt. (Bei uns ca. 50%)

Die MATLAB App wurde entwickelt und es wurde eine Sinusansteuerung implementiert, mit der das Bode Diagramm experimentell bestimmt werden kann.
Die Frequenz der erzeugten Sinus weicht von der eingestellten Frequenz ab. Sie ist in jedem Fall größer, aber nicht nach einem festen Faktor. Es könnte an der Rechenzeit liegen. Dieser Fehler kann überprüft und behoben werden.
Bisher ist es noch nicht möglich die Lampe über die App einzuschalten oder in der Helligkeit zu variieren. Außerdem können die Regler Parameter des SIPART nicht über die App eingestellt werden. Die App kann um diese Funktionen noch erweitert werden.
Das bestimmen des Betrags und der Phase für die experimentelle Bestimmung der Strecke muss bisher von Hand gemacht werden. Da kann ein Verfahren über die Lissajous-Figuren ausgearbeitet werden, das anhand der Formeln die Parameter bestimmt und diese dann in einen Array schreibt. Damit kann sogar vielleicht direkt das experimentell ermittelte Bode Diagramm ausgeben werden.

Die Umsetzung des IOT Konzepts hat aufgrund fehlender Zeit nicht funktioniert. Daher wurden 2 Konzepte erarbeitet, die weiterverfolgt werden können.
Alle erzeugten MATLAB-Graphen, Simulink Simulationen und MATLAB Skripte, sowie die App Datei werden an den Betreuer dieser Arbeit, Dr. Serge Zacher, übergeben.

REFERENCES

VII. REFERENCES

[1] Schöneweiß, D., Rüschhoff, H., Vesper, T., 2018. Inbetriebnahme und Laborversuch mit dem Regler SIPART DR20 von Siemens.

[2] Pfäffle, M. Bode Aided Design SIPART DR20 Kompaktregeler.

[3] FlexJobs Job Search Tips and Blog, 2020. Remote Work Statistics: Navigating the New Normal (2021) | FlexJobs. https://www.flexjobs.com/blog/post/remote-work-statistics/. Accessed 7 June 2021.

[4] Prof. Dr.-Ing. S. Zacher, 2018. Zeitprozentkennwert-Verfahren: Hinweise zur Streckenidentifikation und Reglereinstellung nach Schwarze / Latzel. Automation Letter Nr.24.

[5] Zacher, S., 2020. Drei-Bode-Plots-Verfahren für Regelungstechnik: Ein universelles Stabilittskriterium für stabile und instabile Regelstrecken. Springer Vieweg, Wiesbaden.

[6] Florian Raschbichler, 2017. MQTT – Leitfaden zum Protokoll für das Internet der Dinge | Informatik Aktuell. https://www.informatik-aktuell.de/betrieb/netzwerke/mqtt-leitfaden-zum-protokoll-fuer-das-internet-der-dinge.html. Accessed 10 June 2021.