Projektbericht

Anwendungswerkstatt – „Strukturwandel“

Personalisierte Klimatisierung: Neue, energieeffiziente Gebäudeklimatisierung – personalisierte Klimatisierung am Arbeitsplatz

Gruppe 4

Angelini Evaluna - 389565

Hu Yuhan - 389240

Li Shutong - 389493

Wegewitz Stephan - 316021

Univ.-Prof. Dr.-Ing. habil. Christoph van Treeck

e3D Lehrstuhl für Energieeffizientes Bauen | RWTH AACHEN UNIVERSITY

Inhaltsverzeichnis

1. Einführung

1.1 Aufgabenstellung

Das Ziel unseres Projektes ist die Erstellung eines Klimatisierungssystems, das sich selbst regelt, um eine persönliche Klimatisierung im Open Space Office zu schaffen. Dahinter liegt die Möglichkeit das Wohlbefinden des Menschen am Arbeitsplatz zu erhöhen und dadurch auch den Energieverbrauch für die Heizung und Kühlung zu vermindern. Das Projekt wurde im Rahmen der Veranstaltung „Anwendungswerkstatt-Strukturwandel“ in einem Zeitraum von Zwei ein halb Monaten durchgeführt. Die verwendeten Geräte wurden entweder selbst erstanden oder durch die Technik des E3D zur Verfügung gestellt. Hierbei nochmal einen herzlichen Dank an die Techniker des E3D für ihre Unterstützung. Ziel war es ein eigenes personalisiertes Klimatisierungskonzept zu erarbeiten und dieses umzusetzen sowie zu testen. Die Aufgabe wurde vor dem Hintergrund der energieeffizienteren Gestaltung von Arbeitsplätzen im Büro gestellt, welcher einher geht mit der generellen Ausrichtung zur Energieeffizienz im Gebäudesektor im Deutschland, vor dem Hintergrund des Klimawandels.

1.2 Stand der Technik

Die Bauwirtschaft steht heute vor zwei großen Herausforderungen: die zunehmende Sorge Energie einzusparen und der wachsende Bedarf an Komfortverbesserungen (Veselý und Zeiler, 2014). In bisherigen Studien wurde großen Wert auf Heizungs-, Lüftungs- und Klimasysteme (HLK) und dem Komfort und der Energieeffizienzs gelegt (Holmes und Hacker, 2007; Yoshino et al., 2006). HLK benötigen einen sehr großen Anteil von über 60% der gesamten Gebäudeenergie (Pérez-Lombard et al., 2008). Gerade im Bereich der Nichtwohngebäude und im Gewerbe, Handel und Dienstleistungsbereich in Deutschland welcher rund 16% ausmacht, wird ein hoher Teil an Energie durch HLK verbraucht (Umweltbundesamt, 2018; Schloman et al., 2015). Der hohe Energieverbrauch herkömmlicher HLK-Systeme beruht im Wesentlichen auf der vereinheitlichen Regelung der Innenraumlufttemperatur (Hoyt et al., 2015).

Um HLK-Systeme noch spezifischer auf den Nutzer zu regeln, wurden in unterschiedlichen Studien personalisierte Konditionierungssysteme (PCS) entwickelt, wie z. B. personalisierte Lüftung, Aufbereitungssysteme für Arbeitsaufgaben und einen beheizten bzw. gekühlten Stuhl (Amai et al., 2007; Melikov, 2004; Veselý und Zeiler, 2014; Watanabe et al., 2009; Zhang et al., 2010d und 2015). Im Gegensatz zu den traditionellen HLK-Systemen, die den gesamten Raum so gestalten, dass ein einheitliches Raumklima entsteht, ist bei Personalsystemen nur ein relativ kleiner Raum um die Personen zur Beeinflussung vorgesehen.

Dadurch kann man den Arbeitsplatz vollständig personalisieren und für unter anderem biodynamische Beleuchtung, Temperatur, und Klima auf der Grundlage eines individuellen ergonomischen Ausweises und anhand persönlicher Vorlieben einstellen. PCS spart Energie, indem die Umgebungstemperatur weniger geregelt wird (Veselý und Zeiler, 2014). Daher kann in einer nicht einheitlichen Umgebung eine jährliche Energieeinsparung von 40–50% erzielt werden ausgestattet mit PCS-System.

Ein entscheidender Faktor bei dem Einsatz von personalisierter Heizsysteme ist die angewandte Regelungsstrategie, von welcher letztendlich auch die eingesparte Leistung abhängt. Veselý und Zeiler (Veselý und Zeiler, 2014) haben festgestellt, dass PCS überwiegend durch Benutzerinteraktionen gesteuert werden, z.B. der Benutzer steuert selbst

die Heiz- oder Kühlleistung. Eine jederzeit verfügbare Kontrolle über der thermischen Umgebung ist mit einem verbesserten thermischen Komfort verbunden (Boerstra et al., 2013). Allerdings kann auch eine Steuerung der PCS zu einer effizienteren Energieverwendung beitragen, die automatisiert ist und nur mit einer geringen Nutzerinteraktion auskommt. Dies hätte auch den Vorteil, dass es dem Anwender ermöglicht, sich besser auf anderen Aktivitäten zu konzentrieren.

2. Konzept und Aufbau

2.1 Konzept für die Regelung

Nach einer Literarturrecherche, wurde schnell ersichtlich, dass sich PCS's in mehrere Arten einteilen lassen können, insbesondere zwischen Kühl- und Heizsysteme. Es wurden bereits unterschiedliche Heizsysteme (Vesely et al., 2016; Zhang, 2015; Wilmer et al., 2014; Deng, 2016) und Kühlsysteme (Pasut, 2014; Vesely, 2014) getestet. Die Nutzung von Infrarotwärmeerzeugern hat sich in diesen Studien als effektiv erwiesen und sollte deswegen auch hier zum Einsatz kommen. Ebenso wurden bereits Studien mit Ventilation zur Kühlung erfolgreich durchgeführt. Es lag also auf der Hand für ein System, welches sowohl einen Kühlfall als auch einen Heizfall vorsieht, diese beiden Geräte in ein System zu integrieren.

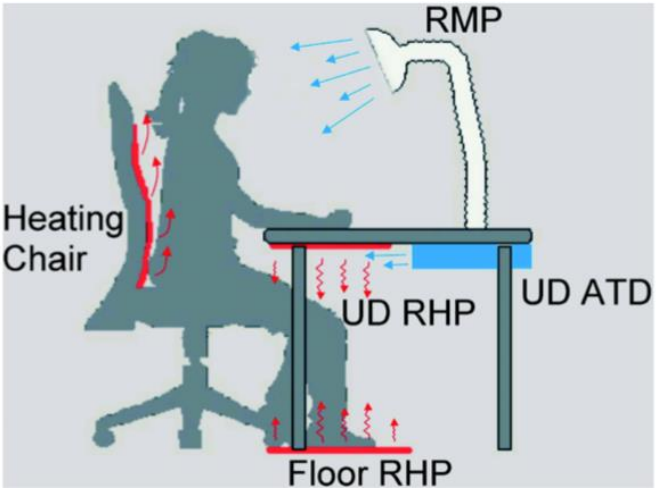


Abbildung 1 Skizze des verwendeten persönlichen Klimaregulierungssystems (Arens und Zhang, 2006)

In Anlehnung an das Kühlsystem aus (Arens und Zhang, 2006) wurde ein ähnlicher Versuchsaufbau vorgesehen, welcher in Abbildung 1 dargestellt wurde. Aufgrund der begrenzten Zeit und der begrenzten Mittel wurde ein verkleinerter Versuchsaufbau umgesetzt, der aber schon alle Elemente besitzt, die entscheidend sind. Namentlich sind das ein Widerstand statt einer Infrarotplatte und eine kleine Version eines Ventilators.

Mit dem Ziel eine Infrarotplatte und Ventilator anzusteuern, wurde der Regelkreis, der in Abbildung 2 zu sehen ist entworfen. Für einen geschlossenen Regelkreis müssen mehrere veränderliche Größe gegeben sein, die wir wie folgt zugeordnet haben

- Regelgröße --Temperatur der Haut (T_Haut)
- Stellgröße – Stärke der Heizung/Ventilators
- Sollwert – Temperatur die die Haut einhalten soll (T_Soll)

Daraufhin wurden zwei Regelkreise für Heizung und Kühlung entworfen, die ähnlich aber auch unabhängig voneinander laufen, siehe Abbildung 2 oder Anhang 1. Um ein gleichzeitiges aktiveren des Ventilators und der Infrarotplatte zu verhindern, wurde eine Fallunterscheidung vorangeschaltet, welche im ersten Entwurf Innenraumtemperatur und die Temperatur

außerhalb des Gebäudes miteinander vergleicht, um festzustellen, welches Gerät zur Temperierung nun genutzt wird. Sollte z.B. die Außentemperatur kälter sein als die Innentemperatur, sollte nur das Heizgerät zur Nutzung kommen.

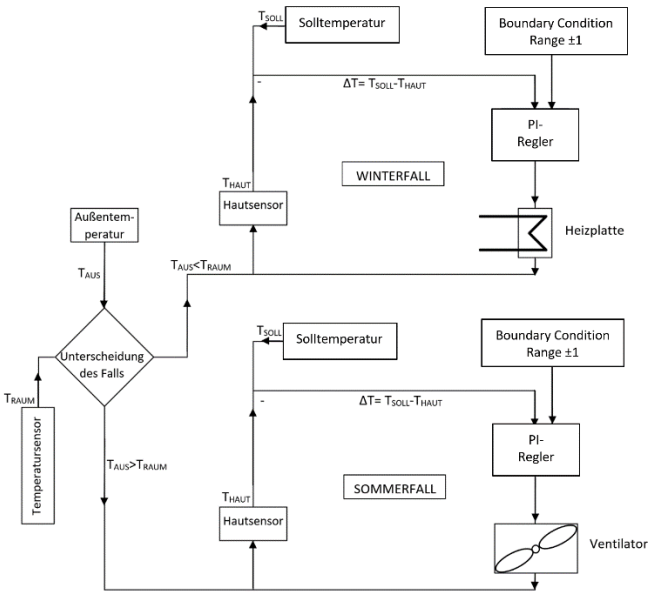


Abbildung 2 Erstes Konzept der Regelung (Eigene Darstellung)

Um den Aufwand zur Programmierung und Regelung zu reduzieren, haben wir einen zweiten Ansatz für einen verkleinerten Regelkreis, siehe Abbildung 3 und Anhang 2, entworfen, den wir jedoch nicht mehr in der vorgegebenen Zeit umsetzen konnten. Das tatsächliche Programm konnte bisher nur genutzt werden um entweder den zur Verfügung stehenden Ventilator oder den Heizstab einzeln zu regeln.

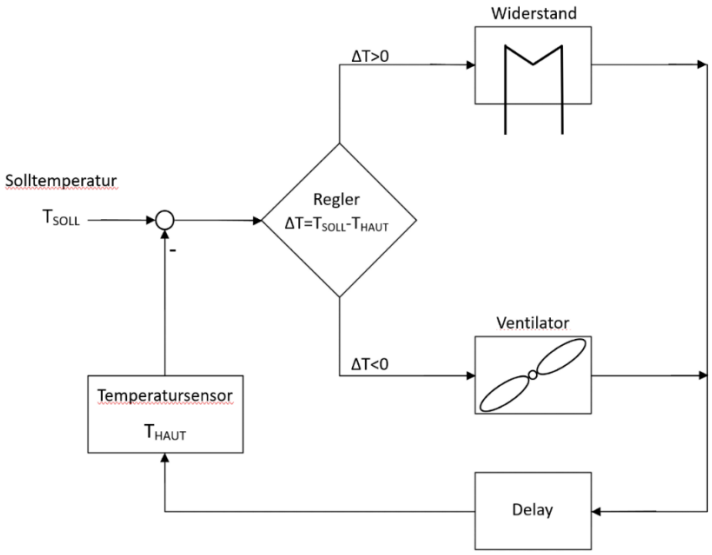


Abbildung 3 geplanter Regelkreis (Eigene Darstellung)

Die Regelung basiert auf das Erreichen der Soll-Temperatur, die für den einzelnen Probanden charakteristisch ist und in Normzustand der Versuchsperson gemessen wurde. Mehrere Hautsensoren werden auf Hände und Gesicht verteilt und platziert, welche später vom Widerstand und dem Ventilator erfasst werden. Diese sind die entscheidenden Teile des Körpers für die Bemessung, weil sie die Ersten sind, an denen sich theoretisch eine

Veränderung der Körpertemperatur zeigen wird. Durch die Kontaktsensoren ist es möglich die Hauttemperatur des Probanden zu messen, die mit der Solltemperatur verglichen werden soll.

Für den Vergleich stehen somit die zwei Temperaturen zur Verfügung, die in drei verschiedene Fällen kombiniert werden können.

Wenn T_{Soll}-T_{Haut}>0 ist, ist die Umgebungstemperatur für die Versuchsperson zu niedrig und die Heizplatte soll angeschaltet werden, um die Hauttemperatur zu erhöhen.
Im anderen Fall benötigt der Proband aufgrund einer negativen Temperaturdifferenz T_{Soll} -T_{Haut} < 0 eine Senkung der Umgebungstemperatur. In diesem Fall wird der Ventilator angetrieben. Die Strahlungstemperatur der Heizplatte und die Drehzahl des Ventilators sind abhängig von der Temperaturdifferenz. Wenn der Unterschied zwischen Soll- und Hauttemperatur zu groß ist, soll die Heizplatte eine stärkere Wärme ausgeben bzw. der Ventilator mit einer höheren Geschwindigkeit drehen. Ansonsten generiert die Heizplatte eine leichte Wärmemenge und der Ventilator dreht langsamer bei kleinen Temperaturunterschied.
Wenn T_{Soll} - T_{Haut} = 0 ist, ist der Zustand erreicht, in dem sich die Versuchsperson im Zustand der thermischen Behaglichkeit befindet und die Umgebungstemperatur nicht mehr angepasst werden muss erreicht.

Nach der Freischaltung oder Aktivierung eines des beiden Geräte soll das System anhalten, damit das Gerät die Zeit hat, durch die Regelung die Umgebungstemperatur abzutasten und die gewünschte Soll-Temperatur zu erreichen. Sobald dieses Zeitintervall abgelaufen ist, wird der Regelkreis neu gestartet und die Hauttemperatur durch den Temperatursensor wieder abgelesen.

2.2 Hauttemperatur

Ein Sensor soll die Temperatur der Haut als Regelgröße erfassen. Dies hat den Hintergrund, dass das thermische Empfinden von mehreren Faktoren abhängt, unter anderem auch der Aktivität des Menschen. Um diese Miteinzubeziehen gab es einen ersten Vorschlag, den Blutdruck als Messgröße zunehmen, um eine genaue Aussage über die Aktivitäten des Probanden zu ermöglichen, siehe auch Kapitel 5.1. Da dies aus technischen Möglichkeiten nicht möglich war, wurde als Alternative die Hauttemperatur als Regelgröße festgelegt.

Der Körper des Menschen unterscheidet sich in Körperschale und Körperkern, in Abbildung 4 entsprechen sie jeweils den hellen und dunklen Bereich. Der Körperkern besteht aus dem Inneren von Rumpf und Kopf. In diesem Bereich wird die Körpertemperatur annährend konstant gehalten und wird als Kerntemperatur bezeichnet. Die Körperschale umschreibt den Bereich der Extremitäten, der Haut und die darunter liegenden Schichten. Hier wird die Temperatur auch Schalentemperatur genannt und variiert stärker. Das Temperaturumfeld des unbekleideten menschlichen Körpers hängt von der Umgebunstemperatur ab. Wie in der darunter liegenden Abbildung 4 visualisiert, sind die Haut- oder Schalentemperatur auf Isothermen über einen ruhenden, unbekleideten Menschen bei 20°C und 35°C Lufttemperatur verteilt (Koralewski, 2006).

Die Hauttemperatur hat den Vorteil, dass sie in direkter Verbindung zur Körperkerntemperatur steht und einfacher zu Messen ist als die innere Körperkerntemperatur. Die Hauttemperatur wird durch den Wärmeaustausch auf der Oberfläche und durch Durchbltungsschwankungen der Haut und Veränderung der Temperatur des arteriellen Blutes beeinflusst.

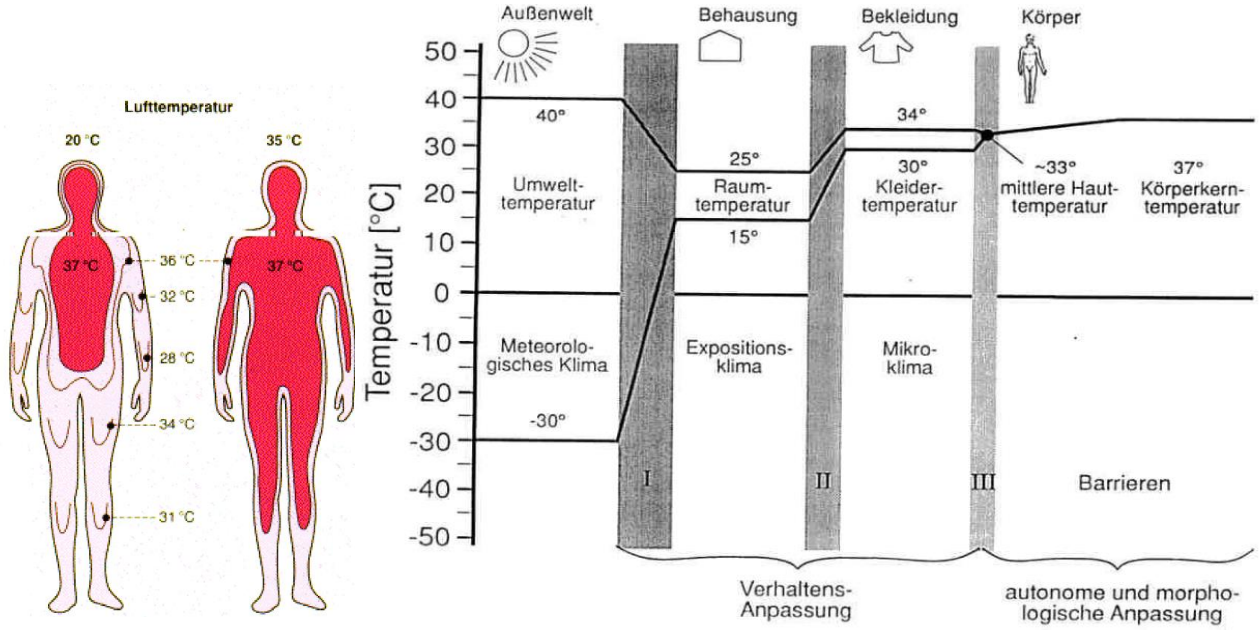


Abbildung 4 Verhalten der Körpertemperatur (Koralewski, 2006).

Grenzwerte für die lokale Hauttemperatur liegen bei maximal 43°C in heißem Umgebungsklima und in kalten Umgebungen beträgt die minimale lokale Hauttemperatur 15°C. Dies gilt insbesondere für die Extrempunkte: Gesicht, Finger und Zehnen (DIN EN ISO 9886).

Die Hauttemperatur ist also durch den direkten Kontakt zur Umgebung, durch die Heterogenität der Haut und durch Innere Vorgänge im Köper sehr heterogen (Zimmermann, 2000). Sie kann also nur zum indirekten Schluss auf die Körpertemperatur genutzt werden, spiegelt aber Aktivitätsgrad der Personen wieder. (Romanovsky, 2013).

Also müssten für den tatsächlichen Einsatz einer solchen Messung folgende Probleme gelöst werden:

- Heterogenität der Haut
- Bekleidung
- Position am Körper der Messung
- Die Rolle der Hauttemperatur in der Behaglichkeit

Da im Fokus der Projektarbeit die Umsetzung eine Regelung stand, soll an diese Stelle angenommen werden, dass nur die Umgebung um Hände und Stirn gemessen wird, um eine halbwegs homogene Messfläche zu erzeugen.

2.3 Verwendete Geräte

In diesem Entwurf haben wir hauptsächlich einen Arduino Uno R3, einen Temperatursensor, einen Ventilator, einen Wiederstand, und eine Steckplatine verwendet

Um die physischen Faktoren zu erfassen und zu kontrollieren, wurde ein Arduino-Board verwendet. Die Programmierbarkeit, Flexibilität und Skalierbarkeit waren die Gründe, die für die Auswahl dieses Mikrocontrollers ausschlaggebend waren (Faris und Mahmood, 2014).
Ein Arduino ist eine elektronische Prototyp-Plattform, bei der es sich um Open Source-Hardware und -Software handelt und flexibel und einfach zu bedienen ist. Die dazugehörige Arduino-Plattform besteht aus einem Arduino Board, der Arduino-Programmiersprache und dem Arduino Development Environment. Die Arduino-Programmiersprache ist eine Programmiersprache, die üblicherweise zur Erstellung von Software verwendet wird, die in

das Arduino-Board eingebettet ist. Die Arduino-Programmiersprache ähnelt der Programmiersprache C ++ (Odendahl, 2010).
Die Arduino-Entwicklungsumgebung ist eine Software, mit der Programme für Arduino geschrieben und kompiliert werden. Sie wird auch verwendet, um Programme, die kompiliert wurden, in den Arduino Board-Programmspeicher zu laden. In diesem Entwurf ist der Arduino Uno R3 ausgewählt worden, wie in Abbildung 5 gezeigt.

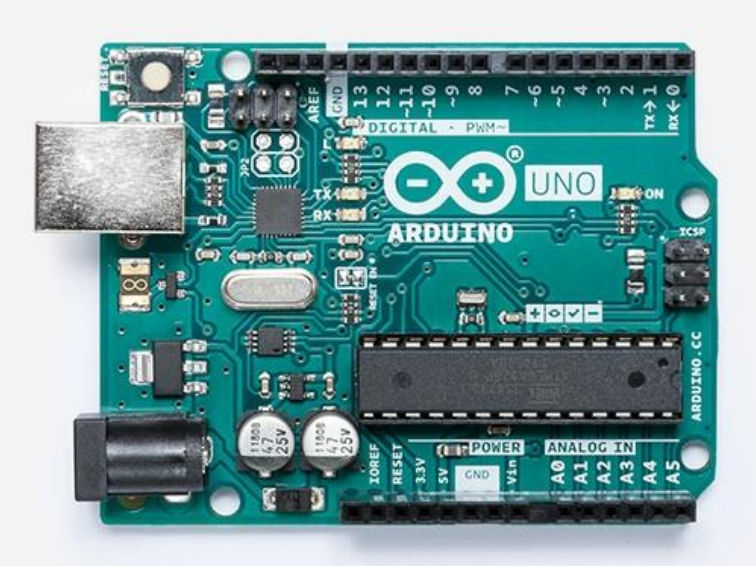


Abbildung 5 Arduino-Board UNO R3 (Quelle: https://store.arduino.cc/arduino-uno-rev3)

Für die Temperaturmessung wurde ein OneWire Kontaktsensor DS18B20 verwendet. Vorteil dieses Gerätes ist die kostengünstige Anschaffung, sowie die akkurate Messung und einfache Handhabung. Darüber hinaus kann der DS18B20 Strom direkt von der Datenleitung beziehen, sodass keine externe Stromversorgung erforderlich ist (Arduino Playground [1], 2019).

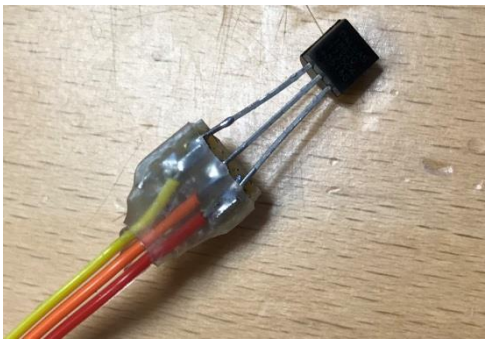


Abbildung 6 Benutzter DS18B20 (eigene Darstellung)

Für den Ventilator haben wir einen PBT-GF30-FR von FOXCONN genutzt. Der Ventilator kann einfach aus einem CPU Kühler entnommen werden. Der Ventilator wird mit 12V Strom versorgt und er ermöglicht, dass die Drehzahl des Ventilators per Arduino gesteuert wird.

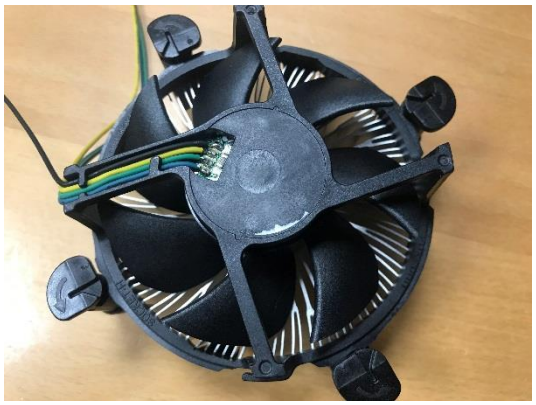


Abbildung 7 Genutzter Kühler (Eigene Darstellung)

Aufgrund von Ausrüstungsbeschränkungen haben wir einen kleinen Leistungswiederstand anstatt einer Infrarotplatte für die Heizung verwendet. Der 12R Widerstand ist kostengünstig und besitzt eine Leistung von 17W.

Abbildung 8 Genutzter 12R Widerstand (Eigene Darstellung)

Um die elektronischen Bauteile der Versuchsschaltung zu verbinden, haben wir eine lötfreie Steckplatine mit beidseitiger Doppelbusleiste verwendet, da der Aufbau schnell vonstattengeht und kein Löten erforderlich ist.

Zur Stromversorgung der elektronischen Komponenten gibt es eine Busleiste. Eine Busleiste enthält zwei Reihen zur Stromversorgung: Die für eine Versorgungsspannung vorgesehene Reihe ist rot markiert, während die Reihe für Masse bzw. GND blau ist.

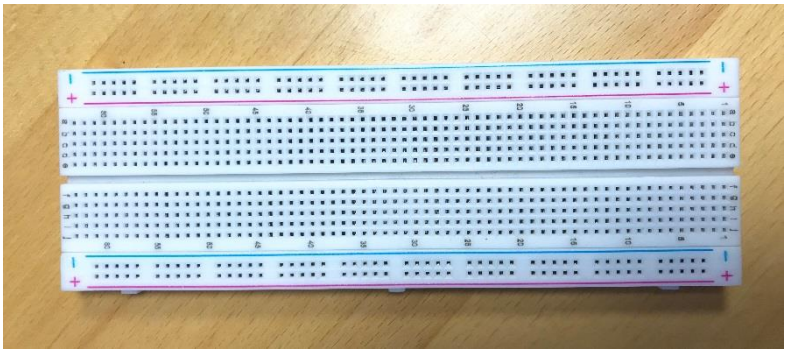


Abbildung 9 Genutzte Busleiste (Eigene Darstellung)

Die Fabrikate wurden, wie in Abbildung 10 und Anhang 4 dargestellt, zusammengeschlossen. Das Schaltbild ist dem Regelkreis, wie er in Kapitel 2 beschrieben wurde nachempfunden und verfügt hier über einen Kontakttemperatursensor, einen Ventilator und einen Widerstand, welche direkt durch den Arduino angesteuert werden können.

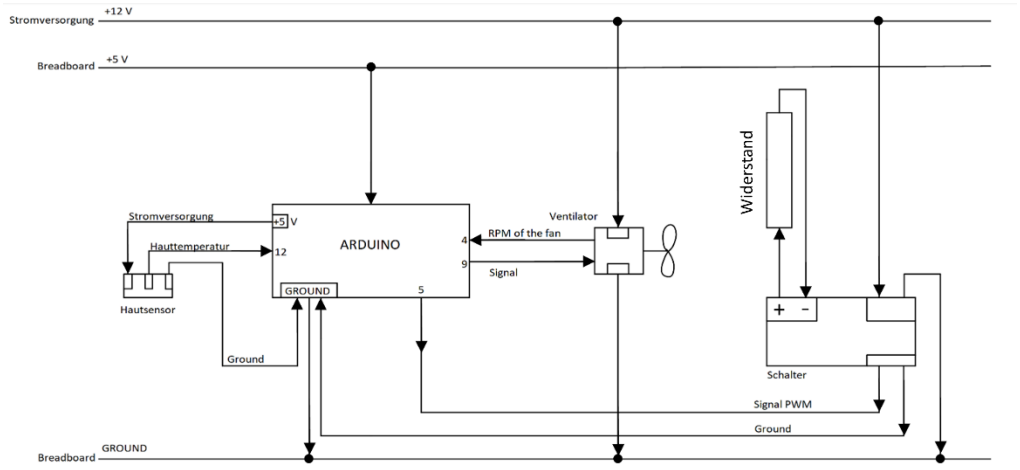


Abbildung 10 Arduino Schaltbild (Eigene Darstellung)

2.4 Programmierung des Arduino und weitere Software

Für die Umsetzung des Regelkreises wurde der Arduino mit Hilfe der Arduino IDE programmiert. Der Programmcode selbst ist im Anhang 7 und 8 zu finden. Für spezielle Anwendungen kann auf externe Bibliotheken zugegriffen werden. Unteranderem wurden verwendet:

- OneWire
- DallasTemperature
- PID_v1

OneWire und DallasTemperature wird genutzt, um den Kontaktsensor anzusprechen (Playground Arduino[1], 2019; Burton, 2019). Für eine schnelle Implementierung eines PID Reglers wurde auf die PID_v1 Bibliothek von Brett Beauregard zurückgegriffen. In der Library ist bereits ein PID-Regler programmiert, dessen Eingangs-, Ausgangsgröße und Koeffizienten durch den Nutzer freiwählbar sind (Playground Arduino[2], 2019).

Um eine Kommunikation mit dem Arduino zu ermöglichen, wurde außerdem ein Commandparser selbst programmiert. Dieser ermöglicht eine Veränderung der Parameter während des Rechenablaufs im Arduino. Eine Änderung der Solltemperatur ist durch Befehlsübergabe an den Arduino in Echtzeit möglich. Weitere Befehle können ebenfalls implementiert werden.

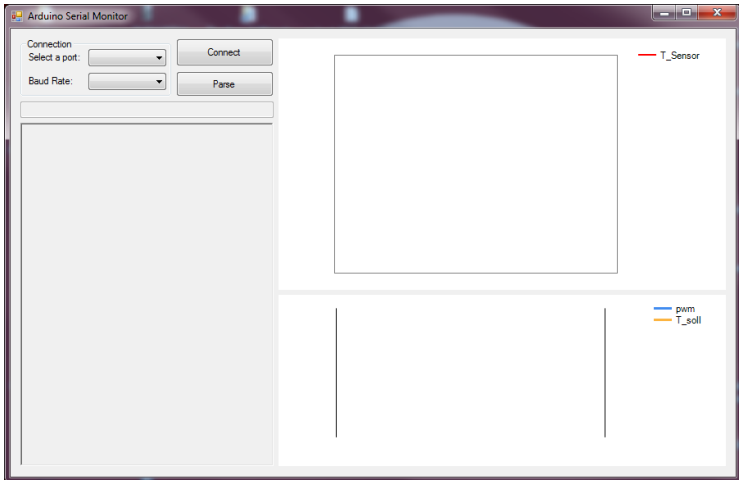


Abbildung 11 VBA Monitor mit Arduino Kompatibel (Eigene Darstellung)

Da die Arduino IDE eine Kommunikation entweder nur über den hauseigenen seriellen Monitor oder nur über Plotter zulässt, aber eine visuelle Auswertung z.B. während der Auswertung einer Sprungantwort notwendig ist und die gleichzeitige Speicherung der erzeugten Daten erwünscht ist, wurde noch ein eigener Monitor programmiert, der eine grafische Auswertung ermöglicht. Hierfür wurde auf die Programmiersprache VBA zurückgegriffen. Als Entwicklungsumgebung wurde hierfür Visual Studios 2017 (VS2017) genutzt. Diese bietet den Vorteil, dass hier eine direkte Anbindung an den Arduino und an die Arduino IDE in VS2017 direkt möglich ist, und die Programmierung in mehreren Sprachen wie C++, Python oder VBA durchführbar ist. Außerdem kann eine direkt ausführbare Datei im Format EXE aus dem fertigen Programmcode erstellt werden.

Der geschriebene Programmcode funktioniert angepasst an den Arduinocode in der Praxis, allerdings kommt es bei der Nutzung der Kommandoübergabe an den Arduino noch zu Fehlern. Die Kommandoübergabe funktioniert teilweise nur verzögert oder gar nicht. Generell braucht der Monitor noch eine weitere Ausarbeitung, welche aber in der kurzen Projektzeit nicht möglich war.

3. Parametrierung mit Einheitssprung

3.1 Versuch der Regelung vom Ventilator

Um Ventilator und Widerstand zu regeln, müssen Parameter des Reglers eingestellt werden. In Versuchen mit Einheitssprung wurden die Parameter separat ermittelt. Mithilfe der Software Arduino V1.8.8 wird die Drehzahl des Ventilators nach PID- Regler geregelt. Die Pulse Width Modulation (PWM), die die Drehzahl des Ventilators verändern kann, wird als die Stellgröße gestellt, damit die Temperatur vom Temperatursensor nach Einheitssprung bis zur Soll-Temperatur konstant gehalten werden kann. Je stärker das PWM Signal ist, also je näher es an die 255 herankommt, desto stärker dreht der Ventilator. Je näher das PWM gegen 0 geht, desto langsamer dreht der Ventilator. Der Ventilator kommt außerdem nicht zum Stillstand bei PWM = 0, sondern dreht mit geringer Drehzahl weiter.

Die Reaktion des Reglers auf den Einheitssprung wurde beobachtet, um die Stabilität des PI-Regler zu überprüfen. Der Einheitssprung kann durch den Widerstand als Wärmequelle kontrolliert erzeugt werden. Dazu muss der Parameter des proportionalen Anteiles K_P, der Parameter des integrierenden Anteiles K_I und der Parameter des differenziellen Anteiles K_P parametriet werden.

Wenn die Parameter K_P, K_I, K_D jeweils 50, 40 und 0,5 sind, verläuft die Temperatur vom Sensor DS18B20 und der PWM wie in Abbildung 12 und 13 zu sehen. Die Soll-Temperatur wurde in diesem Fall als 40℃ gestellt.

Die Temperatur wird am Anfang relativ schnell von 26℃ bis 40℃ geregelt. Danach ist die Temperatur 40℃. Wegen der Störung durch die Umgebung schwankt die Temperatur um 40℃. Die Störung der Umgebung wird allerdings schnell ausglichen. Deswegen ist der PID-Regler stabil.

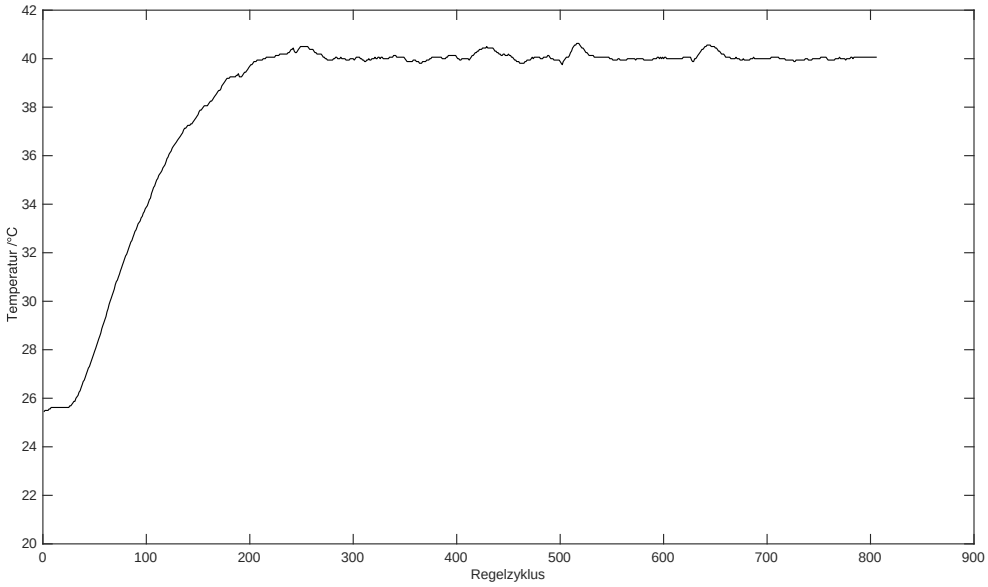


Abbildung 12 Das Temperaturverhalten nach Einheitssprung (Eigene Darstellung)

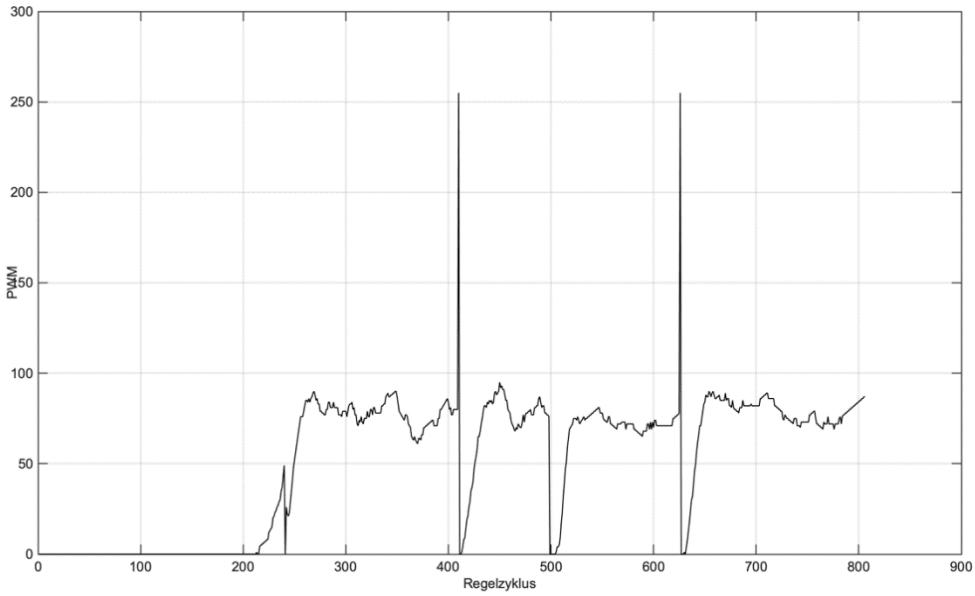


Abbildung 13 Das PWM-Verhalten nach Einheitssprung (Eigene Darstellung)

3.2 Versuch der Regelung vom Heizwiderstand

Ähnlich wie der Versuch des Ventilators lässt sich der Versuch der Regelung vom Heizwiderstand durchführen.

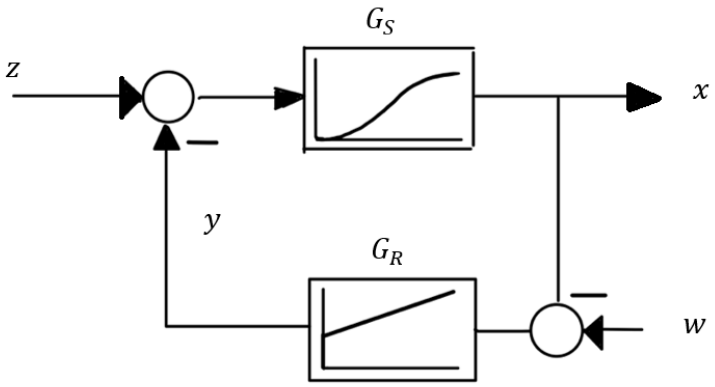


Abbildung 14 Regelkreis (Eigene Darstellung)

Der Regelkreis ist in Abbildung 14 zusehen. Die Temperatur des Heizwiderstandes soll trotz einwinkender Störung konstant gehalten werden. Die Regelstrecke soll sich durch Übertragungsfunktion

$$G_S = \frac{K_S}{(1 + sT)^3}$$

beschreiben lassen. Das heißt, es handelt sich um ein Verzögerungsglied dritter Ordnung (Abel, 2018). Der PI-Regler wurde so im Programm umgesetzt.

Wenn die Verstärkung K_P des proportionalen Anteiles und der Parameter K_I des integrierenden Anteiles jeweils 5 und 0,3 sind, verläuft die Temperatur vom Sensor DS18B20 und die PWM wie in der Abbildung 15 und 16. Die Soll-Temperatur wird in diesem Fall als 40℃ gestellt.

Die Temperatur wird von 29℃ auf 40℃ geregelt. Am Anfang stellt sich eine Überschwing wegen der Eigenschaft des integrierenden Anteiles des PI-Regler ein. Danach wird die Temperatur um 40℃ gehalten.

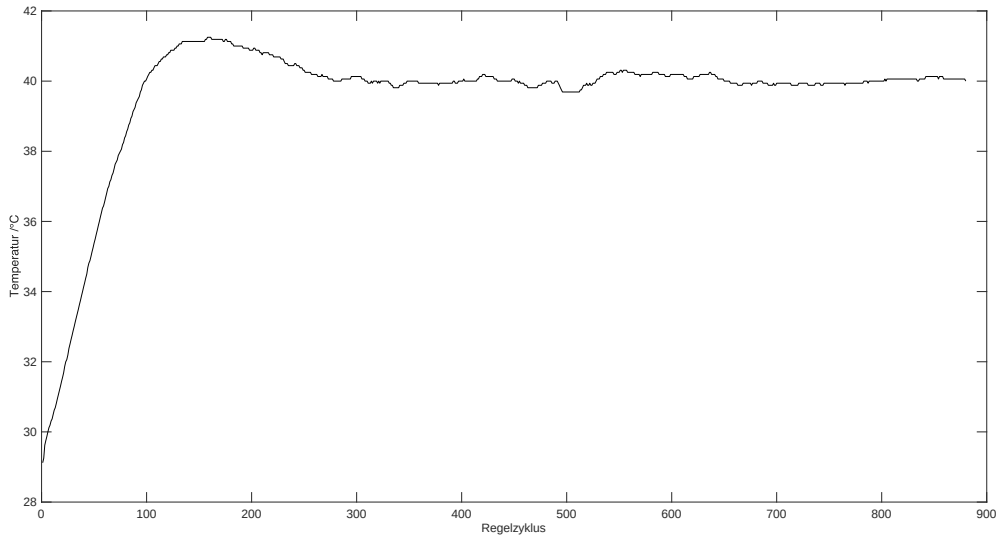


Abbildung 15 Das Temperaturverhalten nach Einheitssprung (Eigene Darstellung)

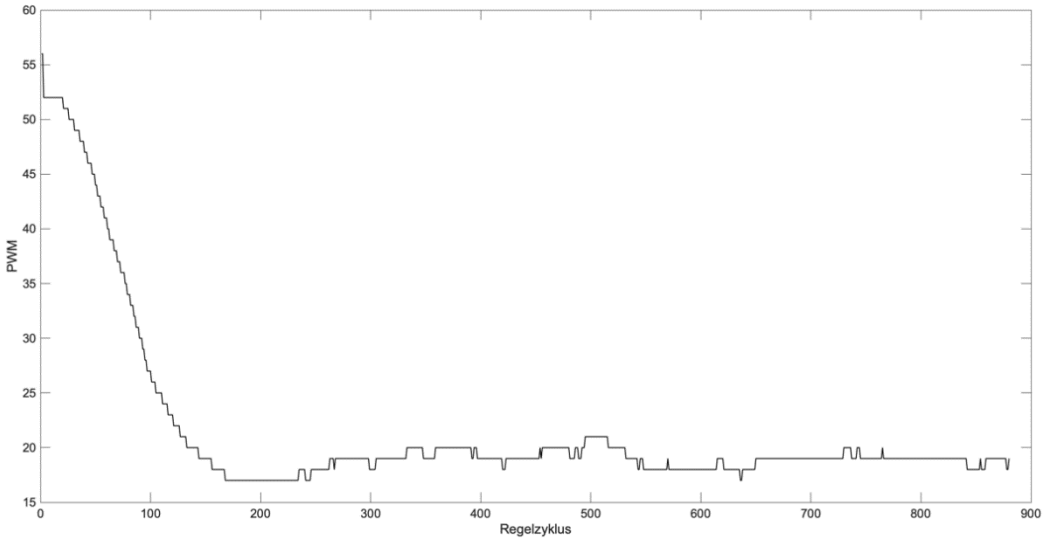


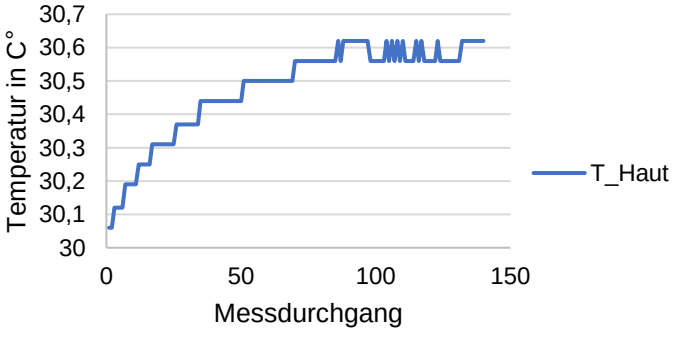
Abbildung 16 Das PWM-Verhalten nach Einheitssprung (Eigene Darstellung)

3.3 Versuch am Menschen

Um zu prüfen ob die Regelungen bei einem Menschen funktioniert, wurden zwei kurze Versuche durchgeführt. Es wurde jeweils die Regelung des Ventilators und die Regelung des Widerstandes am Menschen getestet. Die Hauttemperatur wurde hierfür am Handrücken durch den Kontaktsensor gemessen und die übrigen Geräte auf diese Fläche ausgerichtet.

Zu Beginn wurde die Hauttemperatur gemessen, bis ein stabiles Temperaturniveau erreicht wurde, um Störungen durch Körpertemperatur oder der Schwerfälligkeit des Sensors zu vermeiden, siehe auch Abbildung 17 und 19.

Danach wurde in der selben Ausgangsposition jeweils entweder Heizstab oder Ventilator eingesetzt. Der Ventilator wurde im Abstand von 14cm zur Handfläche aufgestellt. Wie in Abbildung 17 zu sehen, schafft es der Ventilator die Hauttemperatur an die Solltemperatur anzupassen. Er benötigte hierfür ca 5 Minuten, bis der Ventilator die Temperaturdifferenz zwischen Soll- und Hauttemperatur überwunden hatte und stabil beibehalten konnte, siehe Abbildung 18. Hier muss auch angemerkt werden, dass ein Nachteil des verwendeten Ventilators darin besteht, dass er nicht komplett zum Stillstand kommen kann, wenn er an einer Stromversorgung angeschlossen ist und somit eine kleine Restkühlung erzeugt, selbst wenn PWM bei 0 steht.



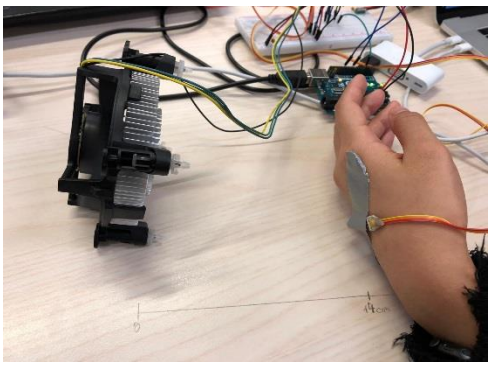


Abbildung 17 Vorläufige Hautemperaturmessung im Versuch 1 und Aufbau (Eigene Darstellung)

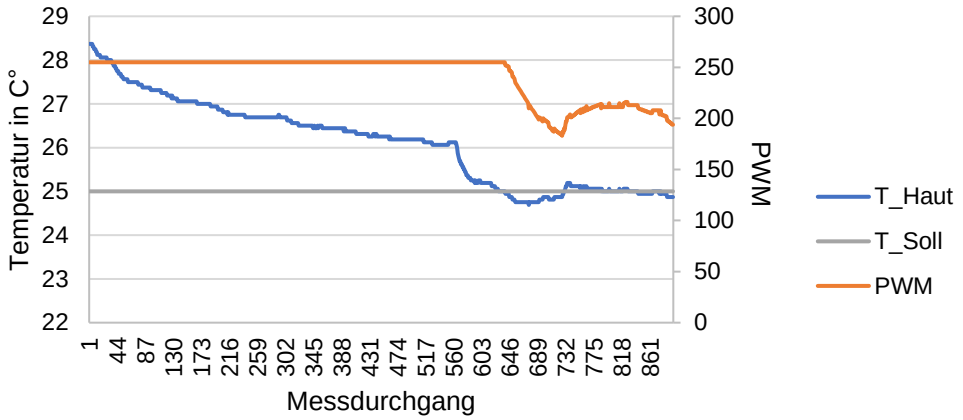


Abbildung 18 Messwerte Versuch 1 – Ventilator (Eigene Darstellung)

Analog zum Ventilator wurde auch der Versuch mit dem Heizstab durchgeführt. Auch hier galt, je größer das PWM ist, desto stärker die Wärmeabgabe durch den Heizstab. Hier zeigte sich die Regelung als sehr schwerfällig. Der Heizstab brauchte eine sehr lange Zeit, bis er die gewünschte Temperatur am Sensor erzeugen konnte, und er musste direkt an der Haut ohne nennenswerten Abstand zum Sensor gebracht werden, um einen Effekt zu erzeugen. Wie man in Abbildung 20 erkennt, schaffte es der Regler auch nicht in der Testzeit von 15 Minuten die Solltemperatur einzustellen. Welche Störgrößen hier noch miteinfließen war nicht ganz klar, jedoch könnte die Körperkerntemperatur ein Faktor hierfür sein, dass die Temperatur schließlich über der Solltemperatur stehen blieb.

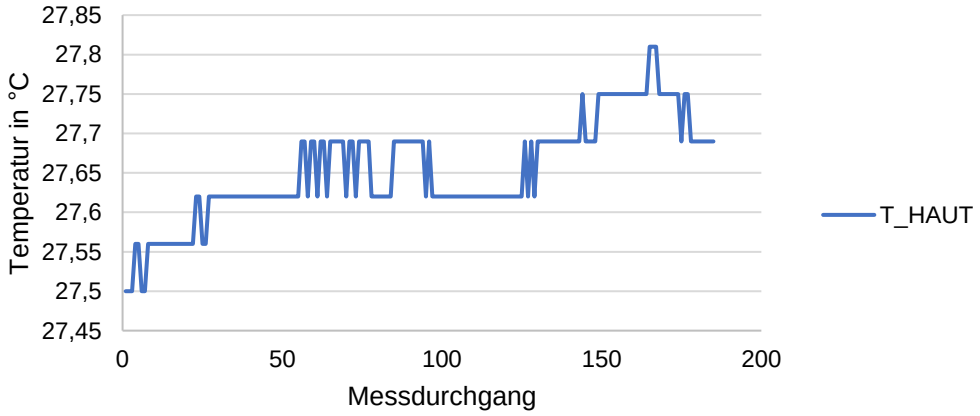


Abbildung 19 Vorläufige Hautemperaturmessung im Versuch 2 (Eigene Darstellung)

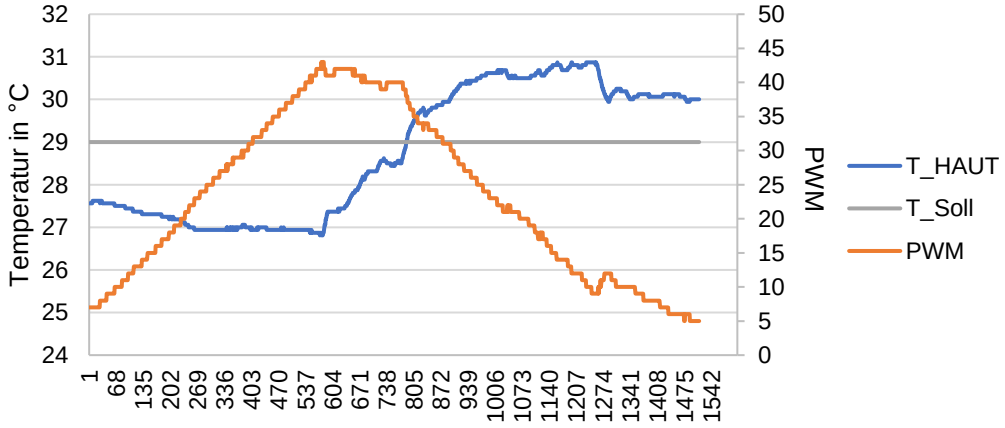


Abbildung 20 Messwerte Versuch – Heizstab (Eigene Darstellung)

4. Fazit

Es wurde die Aufgabe, eine Regelung für ein persönliches Klimasystem zu konzipieren und umzusetzen, bearbeitet. Dafür wurden zwei Regelungskonzepte vorgestellt, welche aber nicht mehr vollständig in der Bearbeitungszeit umgesetzt werden konnten. Als Regelungsgröße wurde hierfür die Hauttemperatur genutzt.
Mithilfe des Arduino Mikrocontrollers wurde daraufhin eine Regelung für einen Ventilator als auch für einen Heizstab in die Praxis umgesetzt. Beide Regelungen wurden parametriert in Versuchen durch Auswertung von Einheitssprüngen und dann bei einem Versuch am Menschen getestet. Hier stellte sich heraus, dass nur die Regelung des Ventilators stabil bleiben konnte, während die Regelung des Heizstabes die Solltemperatur nicht einstellen konnte.
Durch den einen erfolgreichen Versuch kann man darauf schließen, dass eine in sich geschlossene Regelung eines Klimasystems möglich ist, wenn diese auch nur für einen kleinen Bereich des Körpers geprüft wurde.
Im nächsten Kapitel sollen nun weiter Möglichkeiten einer Umsetzung besprochen werden, die nicht im Rahmen unserer Projektarbeit möglich waren.

5. Ausblick

5.1 Blutdruckmessung

Als Alternative Regelgröße könnte eine Blutdruckmessung dienen. Der Blutdruck ist der Druck, mit welchem das Blut aus dem Herz gepumpt wird. Der gemessene Druck besteht aus systolischen und diastolischen Druck. Der systolische Druck gibt an, mit welchem maximalen Druck das Herz Blut in die Arterien pumpt. Die darauffolgende Erschlaffung, die Diastole, ist die Zeit, mit der die Herzkammern wieder mit Blut gefüllt wird. Der Blutdruck sinkt ab und der niedrigste Druckwert wird als diastolischer Druck bezeichnet.

	Systolish (mmHg)	Diastolisch (mmHg)
Optimal	< 120	< 80
Normal	< 130	< 90
hochnormal	130 - 139	85 - 89

Tabelle 1 Blutdrucktabelle mit aktuell geltenden Richtwerte zur Beurteilung von Blutdruckwerten

Durch Untersuchungen wurde ein Zusammenhang zwischen Blutdruck und Wetter herausgefunden. In der kalten Jahreszeit ist der Blutdruck des Menschen höher als im Sommer. Bei unangenehmen Temperaturen reagiert der menschliche Körper auf zwei Arten von Thermoregulationen. Die vasomotorische Thermoregulation betrifft die peripheren Blutgefäße, die nahe der Haut liegen. Sie sind mit Ventilen ausgestattet, die durch Öffnen und Schließen den Blutfluss ermöglichen oder verhindern. Im Winter werden die peripheren Blutgefäße und Kapillaren verengt, um den Körper von Wärmeverlust zu schützen, damit weniger Blut fließen kann und die vorhandene Menge mit höherem Druck transportiert werden kann.
Der Prozess führt zu einer Abnahme der Oberflächentemperatur und des Wärmeaustausches nach außen. In heißen Umgebungen wird die gegenteilige Situation festgestellt. Die peripheren Blutgefäße werden gespannt, was eine Erhöhung der Blutversorgung bewirkt und der Blutdruck sinkt. Das führt zu einer Erhöhung der Hauttemperatur und des Wärmeaustausches nach außen.

Für den Fall, dass die vasomotorische Thermoregulation nicht ausreichend ist, greift die Verhaltensthermoregulation ein. Gegen die Kälte manifestiert es sich mit dem Schauder, der in der Aktivierung fast aller Muskelgruppen und in der Steigerung der Energieerzeugung im Körper besteht. Gegen die Hitze besteht das Schwitzen. Wenn auch die Verhaltensthermoregulation nicht ausreicht, kann man Hypothermie oder Hyperthermie haben. Das Subjekt befindet sich in eine Situation schwerer Unbehaglichkeit, wenn die Innentemperatur unter 35°C (Hypothermie) oder über 38°C (Hyperthermie) erreicht (Betta 2016; Bux, 2016).
Durch diesen Zusammenhang zwischen Umgebungstemperatur und Blutdruck könnte neben der Messung der Hauttemperatur ein genauerer Regler erreicht werden, der nur durch die dauerhafte Messung des Blutdruckes und der Hauttemperatur sich selbst regeln kann. Das klassische Blutdruckmessgerät ist im Abbildung 21 dargestellt. Seine Manschette um das Handgelenk wird mit Luft gefüllt und langsam entleert. Dann hört die Manschette mit einem Mikrofon auf die Strömungsgeräusche. Das Gerät ermittelt daraus einen Wert für die Systole (maximaler Druck, wenn das Herz pumpt) und für die Diastole (minimaler Druck, wenn das Herz sich entspannt). Ein solches Gerät gibt eine genaue Messung des Blutdruckes wieder, kann jedoch nicht permanent betrieben werden und geht mit Einschränkungen im Komfort durch seine Größe einher.

Abbildung 21 Blutdruckmessgerät (Quelle: https://www.real.de/product/317271550/)

Außerdem ist mittlerweile ein alternatives Blutdruckmessgerät als Uhr auf dem Markt. Es handelt sich hierbei um die Asus VivoWatch. Die Blutdruckmessung basiert auf eine Sensoren-Kombination mit Elektroden bzw. Elektrokardiogramm (EKG) und Rotlichtsensoren bzw. Photoplethysmographie (PPG) (Asus, 2019). Um den Blutdruck zu bestimmen, errechnet die Uhr die Zeit, die das Blut braucht, um vom Herzen zum Finger zu fließen und setzt dies ins Verhältnis zur Armlänge, die etwa die Hälfte der Körpergröße beträgt. Möglicherweise könnte mit einer solchen Technik eine komfortablere Lösung zur Messung des Blutdrucks erreicht werden.
Die Verwendung eines Blutdruckmessgeräts war nicht möglich, aufgrund von Zeitgründen und technischen Grenzen. Das Ablesen und die Verwendung der Blutdruckwerte stellt die größte Schwierigkeit dar, da uns keine Möglichkeit bekannt ist, Daten aus dem Blutdruckmessgerät zu erhalten und in ein Programm zu übertragen.
Eine andere Möglichkeit wäre, die Daten von Hand an das Programm zu übergeben, aber in diesem Fall wurde ein System mit zu großen Totzeiten und Aufwand durch den Nutzer entstehen.

5.2 Infrarotsensor

Eine weitere mögliche Verbesserung des Systems ist die Verwendung eines Infrarotsensors anstelle des Kontaktsensors zum Messen der Hauttemperatur. Dadurch könnte eine Zielperson erfasst werden, ohne in seinen Bewegungsabläufen und Komfort gestört zu

werden, da der Infrarotsensor auf Entfernung messen kann indem er die ausgehenden Energieströme aus dem Objekt misst.
In einer Studie nach Koralewski (2006) wurden bereits Vorteile und Nachteile zwischen Kontaktsensor und Infrarotsensor festgestellt, die wir in einem kleinen Versuch ebenfalls feststellen konnten. Kontaktthermometer entziehen dem Messobjekten an der Kontaktstelle Wärme und messen immer nur ihre eigene Fühlertemperatur. Das heißt aber im Umkehrschluss, dass der Messfühler mehr oder weniger, abhängig von seiner Masse, auch die Temperatur des zu messenden Objektes beeinflusst. Hinzu kommt, dass das Aufwärmen des Messfühlers eine bestimmte Zeit benötigt und nur verzögert auf Änderungen der Temperaturen reagiert. Das passiert nicht mit der Verwendung von Infrarot-Thermometer. Der Infrarotsensor ist besonders schnell für die Bestimmung der Oberflächentemperatur geeignet und benötigt keine Aufwärmzeit.

Ein Aufbau für einen vorläufigen Test ist im Anhang 5 zu finden. Für einen Test war ein Infrarotsensor des Typs Melexis Digital IR MLX90614 zur Verfügung gestellt worden.

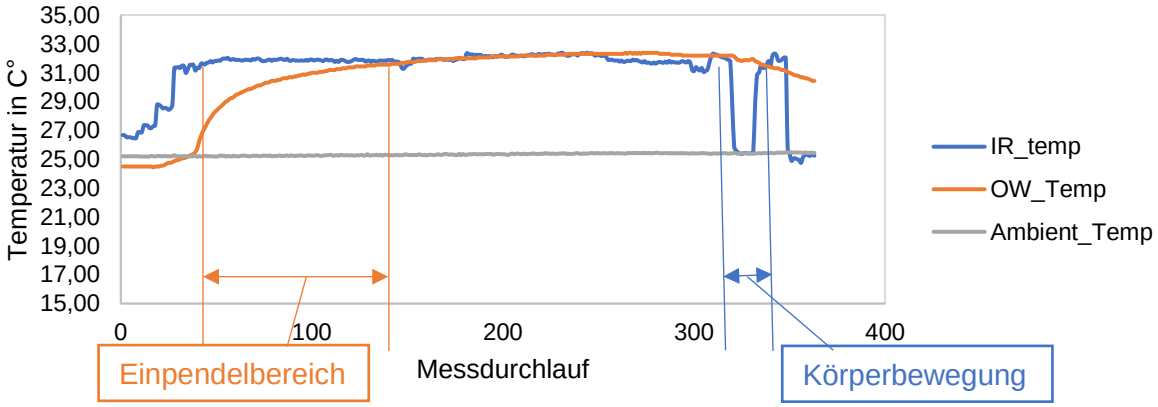


Abbildung 22 Vergleich OneWire Sensor und Infrarotsensor (Eigene Darstellung)

In einem Versuch wurde die Genauigkeit im Vergleich zum Kontaktsensor getestet, siehe Abbildung 22. Der Einpendelbereich zeigt das langsame Anpassen des Kontaktsensors an die Hauttemperatur, während der Infrarotsensor die Temperatur direkt erfasst. Durch die offene Blende des Sensors, ist dieser jedoch sehr empfindlich für Bewegungen.

Die Blende des Sensors ist um ca. 70° geöffnet und gibt somit eine mittlere Temperatur aus dem gesamten Sichtbereich ab. Damit müssen Objekte, deren Temperatur gemessen werden soll, sich nah am Sensor befinden. Eine Möglichkeit die Messung auf größere Entfernung zu ermöglichen bestände in der Beschränkung des Sichtbereiches oder Verkleinerung des Erfassungswinkels. Dies könnte entweder durch eine ideal schwarze Röhre oder durch ein Objektiv, wie es bei Infrarotkameras üblich ist, geschehen. Als Alternativmodel kann z.B. der GY-MLX90614-DCI IIC Langstrecken-Infrarot-Temperatursensor genutzt werden, bei welchem die ideale schwarze Röhre den Sichtbereich insoweit beschränkt, dass nur ein kleiner Bereich durch den Infrarotsensor erfasst wird.

5.3 Größerer Aufbau

Aus Zeitgründen haben wir nur das Lüftungssystem mit einem Kühlerventilator und die Heizung mit einem kleinen Widerstand entwickelt. Diese könnten durch größere Fabrikate und einer richtigen Infrarotplatte ersetzt werden.

Bei der Voruntersuchung der Literatur und Diskussionen wurde entschlossen, dass das die in Abbildung 1 dargestellte personalisierte Heizungssystem verwendet werden kann, um Kühlung und Heizung zu einer kompletten persönlichen Klimaanlage zu kombinieren.

Dieses System besteht aus einem beheizbaren Stuhl, einer beheizten Schreibtischmatte und einer beheizten Bodenmatte. Ein beheizter Stuhl hat eine relativ große Kontaktfläche zum Körper und kann daher durch Wärmeleitung mehr Wärme übertragen als andere Heizmethoden. Es hat sich auch in mehreren Studien als effektive personalisierte Heizmethode erwiesen (Melikov und Knudsen, 2007; Pasut et al., 2015; Zhang et al., 2007; Oi et al., 2011). Da Hände und Füße in der Regel die thermisch unbequemsten Körperteile unter kalten Bedingungen sind (Arens und Zhang, 2006), kann davon ausgegangen werden, dass die angewendete Erwärmung zu einem angenehmen Effekt der Wahrnehmung führt (de Dear, 2011).

In einer Büroumgebung ist es ebenfalls üblich, den Arbeitsplatz für kurze Zeit für eine Pause oder sonstigen Gründen zu verlassen. Deswegen gab es ebenfalls die Idee, die Anwesenheit durch einen Schallwellensensor zu bestimmen. Dieser hätte Bewegungen am Tisch aufgefangen und könnte, wenn eine Person zu weit vom Tisch entfernt ist, die Regelung deaktivieren. In der Kombination mit dem Infrarotsensor wäre somit ein komplett kontaktloses Messsystem entstanden.
Die Heizungs- und Lüftungsanlagen könnten durch Erstellen eines Webservers auf einem Tablet oder PC gesteuert werden. Es könnte auch eine App für Mobilgeräte erstellt werden. Mit drahtlosen Sensoren und Aktuator-Modulen wird die Überwachung und Steuerung der Heim- / Büroumgebung vollständig ausgeführt. Damit werden die unnötigen Geräte nach Bedarf ausgeschaltet. Ähnlich wie das von Sangal et al. entwickelte Hausautomationssystem (Sangal et al., 2016) würde ein solches vernetztes System dazu beitragen, den Energieverbrauch durch die kontinuierliche Überwachung und Steuerung der elektrischen Geräte zu reduzieren.

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

Abel, D: Umdruck zur Vorlesung Mess- und Regelungstechnik. 2018, 9. Auflage, Mainz Gmbh, Aachen

Amai, H., Tanabe, S., Akimoto, T., 2007. Thermal sensation and comfort with different task conditioning systems. Build. Environ. 42, 3955-3964.

Arens, E., Zhang, H., Huizenga, C., 2006a. Partial-and whole-body thermal sensation and comfort—part I: uniform environmental conditions. J. Therm. Biol. 31, 53-59.

Arens, E., Zhang, H., 2006. The skin's role in human thermoregulation and comfort, in: N. Pan, P. Gibson (Eds.), Therm. Moisture Transp. Fibrous Mater., Wood- head Publishing Ltd, 560-602.

Asus: ASUS VivoWatch BP (HC-A04), URL: https://www.asus.com/us/VivoWatch/ASUS-VivoWatch-BP-HC-A04/ [Abgerufen am 06.01.2019]

Betta, V.: Il benessere termoigrometrico negli ambienti moderati, URL: http://www.docente.unicas.it/useruploads/000356/files/dispense_benessere_termoigrometrico.pdf

Boerstra, A., Beuker, T., Loomans, M.G.L.C., Hensen, J.L.M., 2013. Impact of available and perceived control on comfort and health in European offices, Archit. Sci. Rev. 56, 30-41.

Burton, M, Newsome, T., Barros, T., Tillaart, R: Arduino-Temperature-Control-Library, README.md, URL: https://github.com/milesburton/Arduino-Temperature-Control-Library [Abgerufen am 06.01.2019]

Bux, K., Polte, C., 2016: Psychische Gesundheit in der Arbeitswelt - Klima

de Dear, R., 2011. Revisiting an old hypothesis of human thermal perception: alliesthesia, Build. Res. Inf. 39, 108-117.

Deng, Q., Wang, R., Li, Y., Miao , Y., Zhao, J., 2017: Human thermal sensation and comfort in a non-uniform environment with personalized heating, Science of The Total Environment 578 (2017): 242-248.

DIN EN ISO 9886 Ergonomie - Ermittlung der thermischen Beanspruchung durch physiologische Messungen (ISO 9886:2004); Deutsche Fassung EN ISO 9886:2004

Faris D.M.; M. B. Mahmood, 2014. Data Acquisition of Greenhouse Using Arduino, Journal of Babylon University/Pure and Applied Sciences/ No.7/ Vol.22, 1908-1916.

Holmes, M.J., Hacker, J.N., 2007. Climate change, thermal comfort and energy: meeting the design challenges of the 21st century. Energy Build. 39, 802-814.

Hoyt, T., Arens, E., Zhang, H., 2015. Extending air temperature setpoints: simulated energy savings and design considerations for new and retrofit buildings. Build. Environ. 88-96.

Melikov, A.K., 2004. Personalized ventilation. Indoor Air 14 (S7), 157-167.

Melikov, A.K., Knudsen, G.L., 2007. Human response to an individually controlled microenvironment, HVAC R Res. 13, 645-660.

Odendahl, M., Finn, M., Wenger, A., 2010: Arduino - Physical Computing für Bastler, Designer und Geeks, O'Reilly Germany

Oi, H., Yanagi, K., Tabata, K., Tochihara, Y., 2011. Effects of heated seat and foot heater on thermal comfort and heater energy consumption in vehicle, Ergonomics 54, 690-699.

Pasut, W., Zhang, H., Arens, E., Zhai, Y.,2015. Energy-efficient comfort with a heated/ cooled chair, Build. Environ. 84, 10-21.

Pérez-Lombard, L., Ortiz, J., Pout, C., 2008. A review on buildings energy consumption information. Energy Build. 40, 394-398.

Playground Arduino [1] : One Wire, URL: http://playground.arduino.cc/Learning/OneWire [Abgerufen am 06.01.2019]

Playground Arduino [2] : PID Library, URL: https://playground.arduino.cc/Code/PIDLibrary [Abgerufen am 06.01.2019]

Romanovsky, Andrej A: Skin temperature: its role in thermoregulation. *Acta physiologica*, 2014, 210. Jg., Nr. 3, S. 498-507.

Sangle, N., Sanap, S., Salunke, M., Patil,S.,2016. Smart Home System based on IoT, International Journal of Emerging Technology and Advanced Engineering, Volume 6, Issue 9, 168-170.

Schellen, L., van Marken Lichtenbelt, W.D., Loomans, M.G.L.C., Toftum, J., de Wit, M.H., 2010. Differences between young adults and elderly in thermal comfort, pro- ductivity, and thermal physiology in response to a moderate temperature drift and a steady-state condition, Indoor Air 20, 273-283.

Schlomann, B., Wohlfarth, K., Kleeberger, H, Hardi, L, Geiger, B., Pich, A., Gruber., E., Gerspacher, A., Holländer, E., Roser, A. , 2015: Energieverbrauch des Sektors Gewerbe, Handel, Dienstleistungen (GHD) in Deutschland für die Jahre 2011 bis 2013 Schlussbericht an das Bundesministerium für Wirtschaft und Energie (BMWi)

Shao, X., Li, X., 2015. Evaluating the potential of airflow patterns to maintain a non-uni- form indoor environment. Renew. Energy 73, 99-108.

Umweltbundesamt, 2018: Energieverbrauch nach Energieträgern, Sektoren und Anwendungen, URL: https://www.umweltbundesamt.de/daten/energie/energieverbrauch-nach-energietraegern-sektoren [Abgerufen am 06.01.2019]

Veselý, M., Zeiler, W., 2014. Personalized conditioning and its impact on thermal comfort and energy performance–a review. Renew. Sust. Energ. Rev. 34, 401-408.

Wang, D., Zhang, H., Arens, E., Huizenga, C., 2007. Observations of upper-extremity skin temperature and corresponding overall-body thermal sensations and comfort, Build. Environ. 42, 3933-3943.

Watanabe, S., Shimomura, T., Miyazaki, H., 2009. Thermal evaluation of a chair with fans as an individually controlled system. Build. Environ. 44, 1392-1398.

Zhang, H., Arens, E., Kim, D.E., Buchberger, E., Bauman, F., Huizenga, C., 2010d. Comfort, perceived air quality, and work performance in a low-power task–ambient conditioning system. Build. Environ. 45, 29-39.

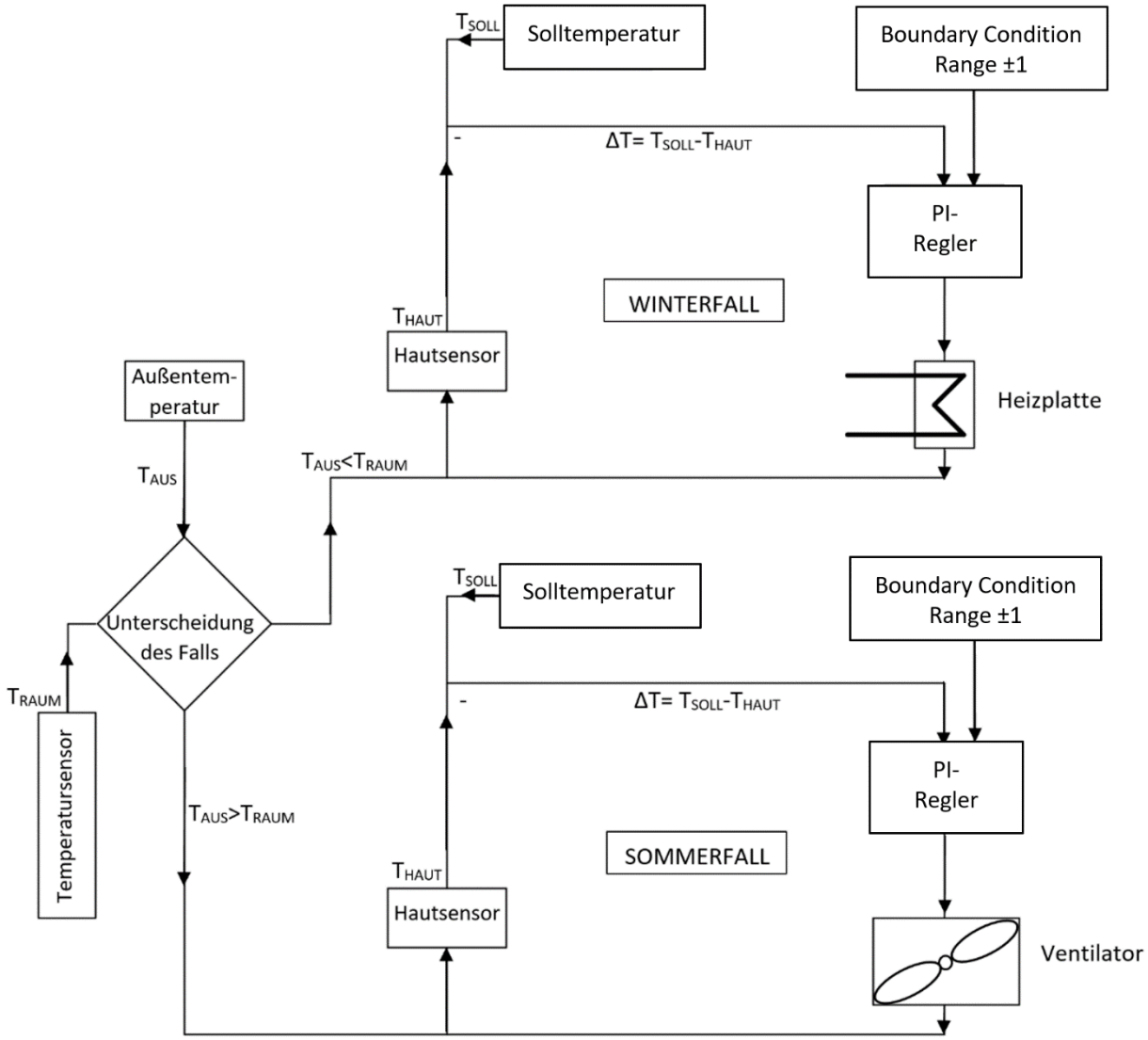
Zhang, H., Arens, E., Zhai, Y., 2015. A review of the corrective power of personal comfort systems in non-neutral ambient environments. Build. Environ. 91, 15-41.

Zhang, Y.F., Wyon, D.P., Fang, L., Melikov, A.K., 2007. The influence of heated or cooled seats on the acceptable ambient temperature range, Ergonomics 50, 586-600,

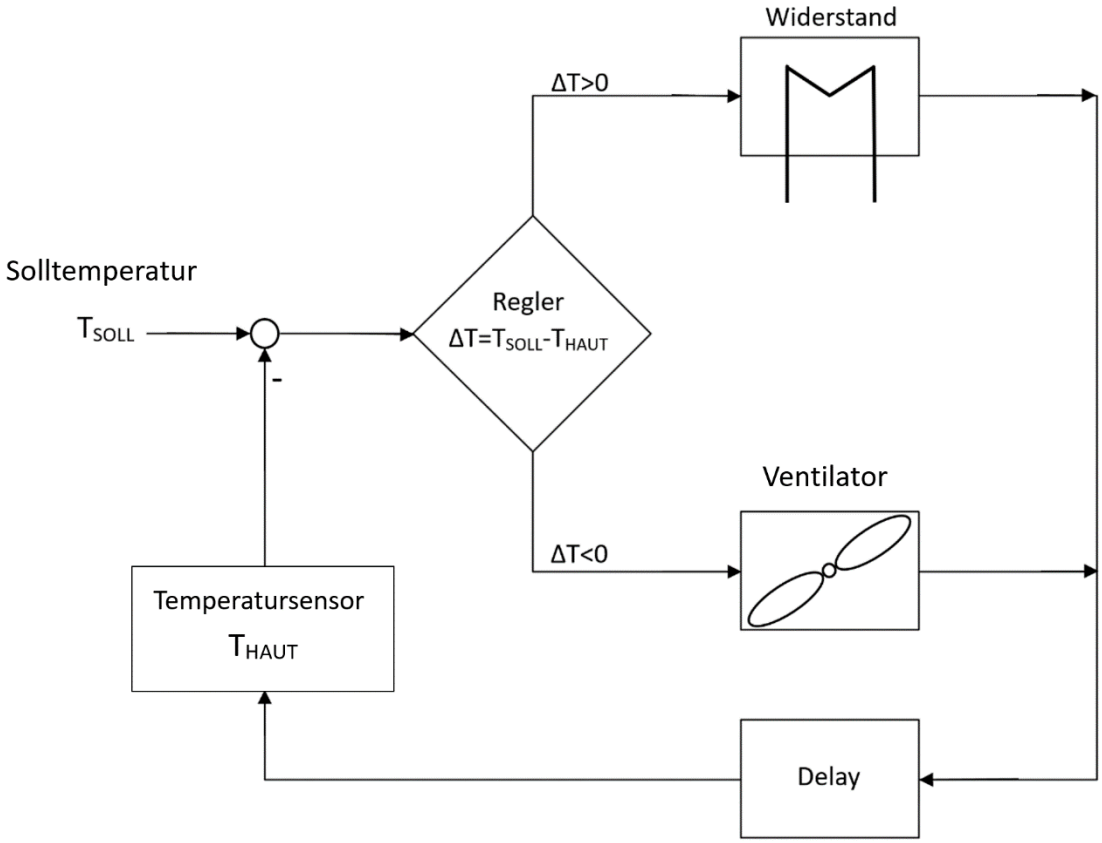
Zimmermann, M., 2000: Grundlagen physiologischer Regelungsprozesse des Menschen, in Schmidt, R. F., Thews, G., Lang, F., 2000: Physiologie des Menschen, 28. Auflage, Springer, Berlin Heidelberg

Anhang

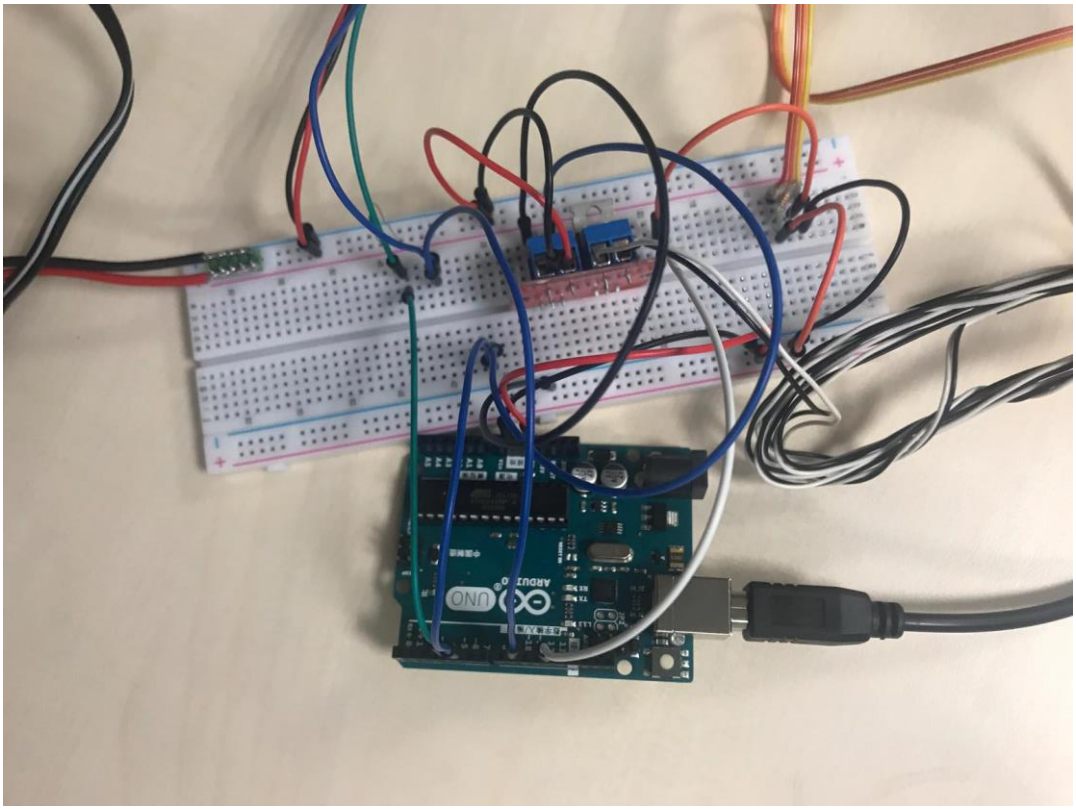
Anhang 1: Regelungskonzept 1



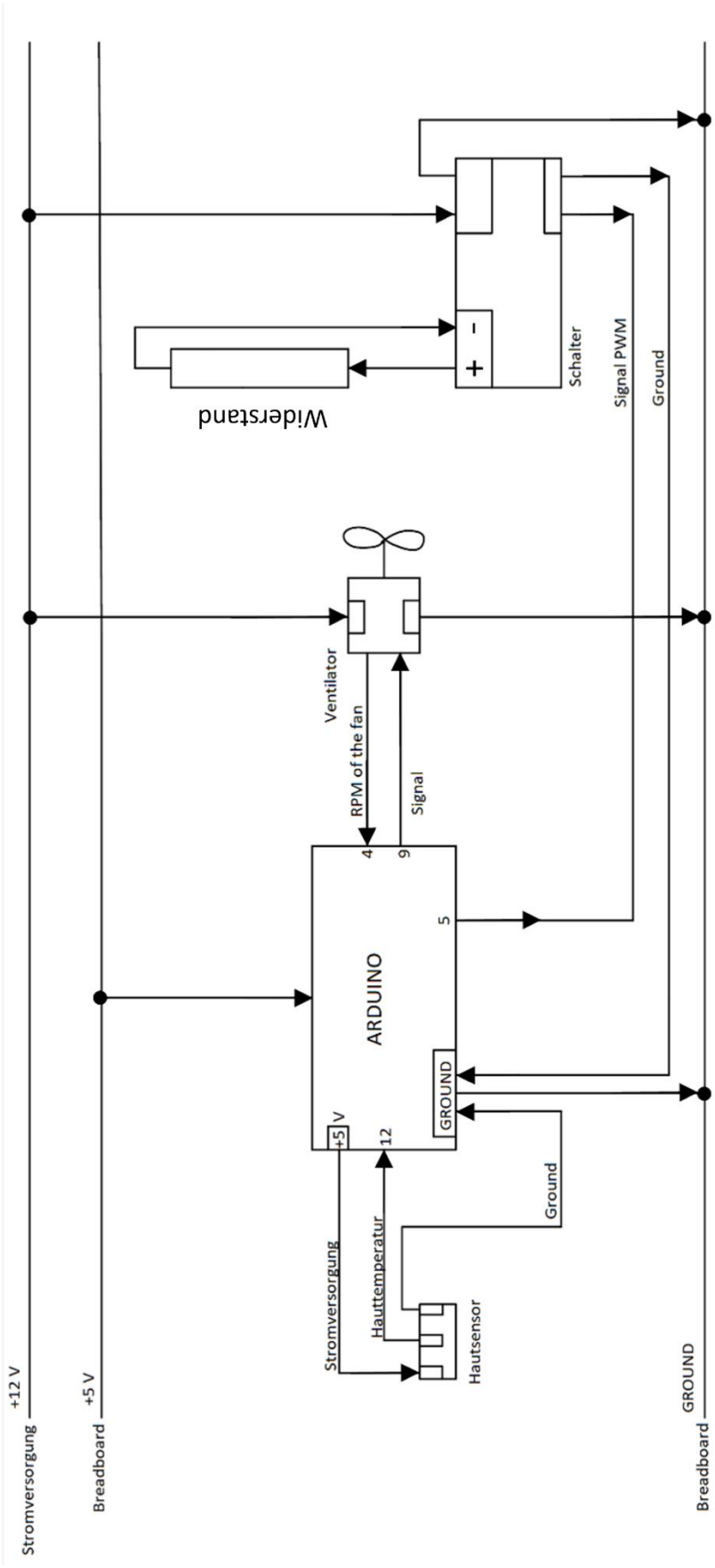
Anhang 2 : Regelungskonzept 2



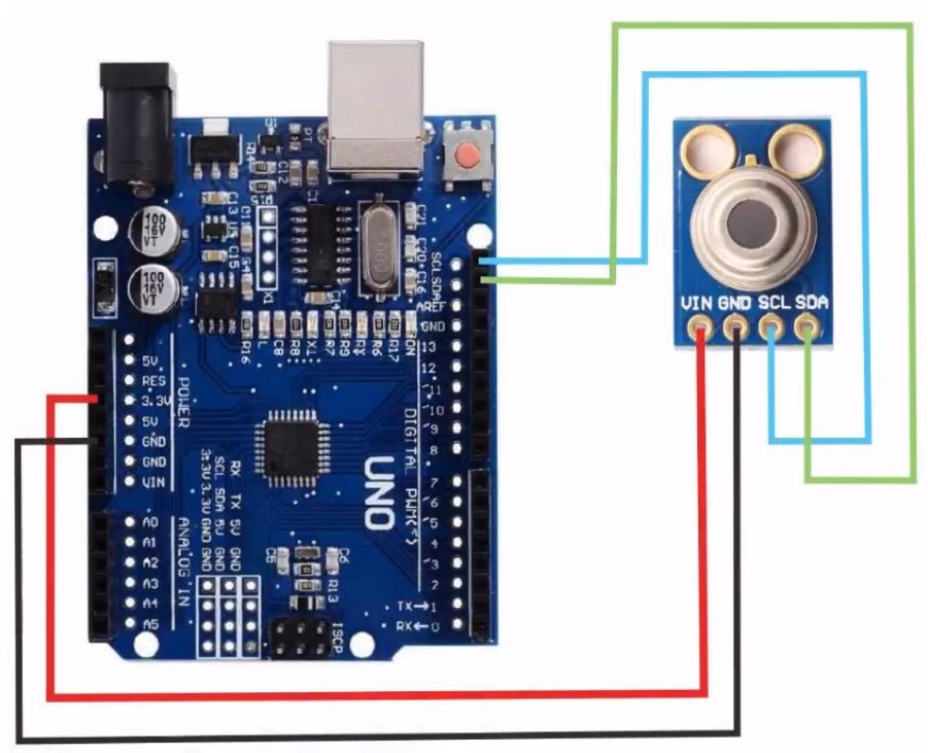
Anhang 3: Versuchsaufbau real



Anhang 4: Versuchsaufbau Schaltbild



Anhang 5: Versuchsaufbau Infrarotsensor



Anhang 6: Arduino Programmcode Ventilatorregelung

```
//Versuch der Regelung vom Ventilator
//PI-Regler

#include <OneWire.h>
#include <DallasTemperature.h>
#include "PID_v1.h"

#define ONE_WIRE_BUS 12  //the DT sensor Daten wird vom PIN 12 bekommen

#define pin 4  // Vom PIN4 wird die Drehzahl vom Ventilator gelesen
#define pwm 9 // Vom PIN9 wird die PWM Value vom Venzilator ausgegeben
#define pwm_heiz 5 // Vom PIN5 wird die PWM Value vom Heizwiderstand ausgegeben
double pwmValue = 100;  // PWM value für Ventilator, damit die Drehzahl vom Ventilator gereglt werden kann.
int pwmValue_heiz=70; // PWM value für Heizwiderstand

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

String command; //String für Parser
double t_input=40; //die Temoeratur vom Sensor
double t_offset=40;//Soll-Temperatur
double k_p=50;//die Verstärkung des proportionalen Anteil
double k_i=40;//der Parameter des integrierenden Anteil
double k_d=0.5; //der Parameter des differenzielen Anteil
PID myPID(&t_input, &pwmValue, &t_offset, k_p, k_i, k_d, DIRECT);//PID-Regler

int i=0;//Zähler des Zyklus

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);//Parametrien des Fenster
  pinMode(pin, INPUT_PULLUP);
  pinMode(pwm, OUTPUT);
  pinMode(pwm_heiz, OUTPUT);
  sensors.begin(); //Temperatur Sensor startet.

  myPID.SetMode(AUTOMATIC);//PI-Regler startet.
```

```
  analogWrite(pwm, pwmValue);
  delay(2000);
}

void loop() {
  // put your main code here, to run repeatedly:

  i++; //zählen die Zyklen
  parserCheck();//Parsersoll prüfen ob eine Commandline übergeben wurde
  sensors.requestTemperatures();//Temperatur vom Sensor abgefragt
  t_input=sensors.getTempCByIndex(0);//lesen die Temperatur des Sensor
  myPID.Compute();//regeln
  pwmValue=(int)pwmValue;// Formatkonvertierung vom double zur int
  if(pwmValue>255)  //PWM value ist zwischen 0 und 255
  {
    pwmValue=255;
    }
  if(pwmValue<0)
  {
    pwmValue=0;
    }
  analogWrite(pwm, 255-pwmValue);

  //Heizwiderstand wird eingeschaltet am 5th Zyklus. Hier funktioniert der Widerstand als eine Einheitssprung
  if(i==35)
  {
    analogWrite(pwm_heiz,pwmValue_heiz);
    }

//Ausgabe vom Daten
  Serial.print(255-pwmValue);
  Serial.print(" ");
  Serial.print(t_input);
  Serial.print(" ");
  Serial.println(t_offset);

}
```

```
//_____PARSER_____
_____

void parserCheck() {
  //Prüfen ob Serial vom USB aus übergeben wird
  //Jeder Charakter wird durchgegangen bis eine new line übergeben wird. Dann wird der Befehl überprüft und
ausgeführt
  while(Serial.available())
  {
     char c = Serial.read();

     if(c == '\n')
     //every command is separate from a line
     {
        //command wird überprüft, was für ein Befehl vorliegt
                    parseCommand(command);
        command = "";
     }
     else
     {
        //Zusammenfügen der Charakter
                    command += c;
     }
   }


}

void parseCommand(String com)
{
  //Erstellen von Arbeitsvariablen
            String part1;
   String part2;

//Teilen des Strings in 2 Teile
   part1 = com.substring(0, com.indexOf(" "));
   part2 = com.substring(com.indexOf(" ") + 1);

   //Überprüfen von part 1 und part 2
 if(part1.equalsIgnoreCase("setOffset"))
```

```
{
    int token2 = part2.toInt();

    t_offset = token2;
   //optionale Ausgabe
            // Serial.print("Solltemperatur geändert auf ");
    //Serial.print(t_offset);
    //Serial.println(" C°");
}
else if(part1.equalsIgnoreCase("setHeiz"))
{
  //Änderung PWHEIZ
  int token3 = part2.toInt();
  analogWrite(pwm_heiz, token3);

  }
  else if(part1.equalsIgnoreCase("ki"))
{
  //Änderung K_I Ändern
  int token4 = part2.toInt();
  k_i=token4;
  //Serial.print("k_i geändert auf ");
   // Serial.print(token4);
  }
  else if(part1.equalsIgnoreCase("kp"))
{
  //Änderung K_P
  int token5 = part2.toInt();
  k_p=token5;
  //Serial.print("k_p geändert auf ");
    //Serial.print(token5);
  }
else
{
    Serial.println("COMMAND NOT RECONGNIZED");
}

}
```

Anhang 7: Arduino Programmcode Heizungsregelung

```
//Versuch der Regelung vom Heizwiderstand
//PI-Regler

#include <OneWire.h>

#include <DallasTemperature.h>
#include "PID_v1.h"

#define ONE_WIRE_BUS 12  //the DT sensor Daten wird vom PIN 12 bekommen

#define pwm_heiz 5 // schalter wird mit PIN5 verbunden
double pwmValue = 100;  // PWM value für Ventilator, damit die Drehzahl vom Ventilator gereglt werden kann.
double pwmValue_heiz=0; // PWM value für Heizwiderstand

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

String command; //String für Parser
double t_input=40; //die Temoeratur vom Sensor
double t_offset=40;//Soll-Temperatur
double k_p=5;//die Verstärkung des proportionalen Anteil
double k_i=0.3;//die Verstärkung des integrierenden Anteil
double k_d=0; //Nur PI_Regler wird getestet,deswegen ist k_d=0
PID myPID(&t_input, &pwmValue_heiz, &t_offset, k_p, k_i, k_d, DIRECT);//PI-Regler

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);//Parametrien des Fenster
  pinMode(pwm_heiz, OUTPUT);//
  sensors.begin(); //Temperatur Sensor startet.

  myPID.SetMode(AUTOMATIC);//PI-Regler startet.

  delay(2000);
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  parserCheck();//Parsersoll prüfen ob eine Commandline übergeben wurde
  sensors.requestTemperatures();//Temperatur vom Sensor abgefragt
  t_input=sensors.getTempCByIndex(0);//lesen die Temperatur des Sensor
  myPID.Compute();//regeln
  pwmValue_heiz=(int)pwmValue_heiz;// Formatkonvertierung vom double zur int
  if(pwmValue_heiz>255) //PWM value ist zwischen 0 und 255
  {
    pwmValue_heiz=255;
    }
  if(pwmValue_heiz<0)
  {
    pwmValue_heiz=0;
    }
    analogWrite(pwm_heiz,pwmValue_heiz);

//Ausgabe vom Daten
  Serial.print(pwmValue_heiz);
  Serial.print(" ");
  Serial.print(t_input);
  Serial.print(" ");
  Serial.println(t_offset);

}

//_____PARSER_____
_____

void parserCheck() {
  //Prüfen ob Serial vom USB aus übergeben wird
  //Jeder Charakter wird durchgegangen bis eine new line übergeben wird. Dann wird der Befehl überprüft und
ausgeführt
  while(Serial.available())
  {
     char c = Serial.read();

     if(c == '\n')
     //every command is separate from a line
```

```
    {
        //command wird überprüft, was für ein Befehl vorliegt
                    parseCommand(command);
        command = "";
    }
    else
    {
        //Zusammenfügen der Charakter
                    command += c;
    }
  }


}

void parseCommand(String com)
{
  //Erstellen von Arbeitsvariablen
            String part1;
   String part2;

//Teilen des Strings in 2 Teile
   part1 = com.substring(0, com.indexOf(" "));
   part2 = com.substring(com.indexOf(" ") + 1);

   //Überprüfen von part 1 und part 2
 if(part1.equalsIgnoreCase("setOffset"))
 {
     int token2 = part2.toInt();

     t_offset = token2;
    //optionale Ausgabe
            // Serial.print("Solltemperatur geändert auf ");
     //Serial.print(t_offset);
     //Serial.println(" C°");
 }
 else if(part1.equalsIgnoreCase("setHeiz"))
 {
   //Änderung PWHEIZ
```

```
    int token3 = part2.toInt();
    analogWrite(pwm_heiz, token3);

    }
    else if(part1.equalsIgnoreCase("ki"))
  {
    //Änderung K_I Ändern
    int token4 = part2.toInt();
    k_i=token4;
    //Serial.print("k_i geändert auf ");
     // Serial.print(token4);
    }
    else if(part1.equalsIgnoreCase("kp"))
  {
    //Änderung K_P
    int token5 = part2.toInt();
    k_p=token5;
    //Serial.print("k_p geändert auf ");
      //Serial.print(token5);
    }
  else
  {
     Serial.println("COMMAND NOT RECONGNIZED");
  }

}
```

Anhang 7: Monitor Programmcode VBA-Form

```
Imports System.IO
Imports System.IO.Ports

Public Class Form1

    Dim value1 As Decimal
    Dim value2 As Decimal



    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load

        'SerialPort1.Close()
        'SerialPort1.PortName = "com5"
        'SerialPort1.BaudRate = "9600"
        'SerialPort1.DataBits = 8
        'SerialPort1.Parity = Parity.None
        'SerialPort1.StopBits = StopBits.One
        'SerialPort1.Handshake = Handshake.None
        'SerialPort1.Encoding = System.Text.Encoding.Default
        'SerialPort1.Open()
    End Sub
    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
        Dim s As String
        s = TextBox1.Text + "," + "," + "," + ","
        Dim somestring() As String
        ' Split string based on comma
        somestring = s.Split(New Char() {","c})
        TextBox2.Text = somestring(0)
        Try
            value1 = Convert.ToDecimal(TextBox2.Text)
        Catch ex As Exception
            TextBox1.Text = ""
        End Try

        TextBox3.Text = somestring(1)
        Try
```

```
            value2 = Convert.ToDecimal(TextBox3.Text)
        Catch ex As Exception
            TextBox1.Text = ""
        End Try
        TextBox4.Text = somestring(2)
        Chart1.Series("Humidity").Points.AddY(value2)
        Chart1.Series("Temperature").Points.AddY(value1)
        TextBox1.Text = ""

    End Sub

    Private Sub DataReceived(ByVal sender As Object, ByVal e As SerialDataReceivedEventArgs) Handles
SerialPort1.DataReceived
        Try
            Dim mydata As String = ""
            mydata = SerialPort1.ReadExisting()
            If TextBox1.InvokeRequired Then
                TextBox1.Invoke(DirectCast(Sub() TextBox1.Text &= mydata, MethodInvoker))
            Else
                TextBox1.Text &= mydata
            End If
        Catch ex As Exception
            MessageBox.Show(ex.Message)
        End Try
    End Sub
End Class
```

Anhang 8: Monitor Programmcode VBA-Designer

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()>

Partial Class Form1

    Inherits System.Windows.Forms.Form


    'Form overrides dispose to clean up the component list.

    <System.Diagnostics.DebuggerNonUserCode()>

    Protected Overrides Sub Dispose(ByVal disposing As Boolean)

        Try

            If disposing AndAlso components IsNot Nothing Then

                components.Dispose()

            End If

        Finally

            MyBase.Dispose(disposing)

        End Try

    End Sub


    'Required by the Windows Form Designer

    Private components As System.ComponentModel.IContainer


    'NOTE: The following procedure is required by the Windows Form Designer

    'It can be modified using the Windows Form Designer.

    'Do not modify it using the code editor.

    <System.Diagnostics.DebuggerStepThrough()>

    Private Sub InitializeComponent()

        Me.components = New System.ComponentModel.Container()

        Dim ChartArea1 As System.Windows.Forms.DataVisualization.Charting.ChartArea = New
System.Windows.Forms.DataVisualization.Charting.ChartArea()

        Dim Legend1 As System.Windows.Forms.DataVisualization.Charting.Legend = New
System.Windows.Forms.DataVisualization.Charting.Legend()

        Dim Series1 As System.Windows.Forms.DataVisualization.Charting.Series = New
System.Windows.Forms.DataVisualization.Charting.Series()

        Dim Series2 As System.Windows.Forms.DataVisualization.Charting.Series = New
System.Windows.Forms.DataVisualization.Charting.Series()

        Me.GroupBox1 = New System.Windows.Forms.GroupBox()

        Me.ComboBox2 = New System.Windows.Forms.ComboBox()

        Me.Label2 = New System.Windows.Forms.Label()

        Me.Label1 = New System.Windows.Forms.Label()

        Me.ComboBox1 = New System.Windows.Forms.ComboBox()

        Me.InputTextbox1 = New System.Windows.Forms.TextBox()

        Me.Button1 = New System.Windows.Forms.Button()

        Me.SerialPort1 = New System.IO.Ports.SerialPort(Me.components)

        Me.RichTextBox1 = New System.Windows.Forms.RichTextBox()
```

```
        Me.TextBox2 = New System.Windows.Forms.TextBox()
        Me.TextBox1 = New System.Windows.Forms.TextBox()
        Me.TextBox3 = New System.Windows.Forms.TextBox()
        Me.TextBox4 = New System.Windows.Forms.TextBox()
        Me.BackgroundWorker1 = New System.ComponentModel.BackgroundWorker()
        Me.BackgroundWorker2 = New System.ComponentModel.BackgroundWorker()
        Me.Chart1 = New System.Windows.Forms.DataVisualization.Charting.Chart()
        Me.Timer1 = New System.Windows.Forms.Timer(Me.components)
        Me.GroupBox1.SuspendLayout()
        CType(Me.Chart1, System.ComponentModel.ISupportInitialize).BeginInit()
        Me.SuspendLayout()
        '
        'GroupBox1
        '
        Me.GroupBox1.Anchor = CType(((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Left) _
            Or System.Windows.Forms.AnchorStyles.Right), System.Windows.Forms.AnchorStyles)
        Me.GroupBox1.Controls.Add(Me.ComboBox2)
        Me.GroupBox1.Controls.Add(Me.Label2)
        Me.GroupBox1.Controls.Add(Me.Label1)
        Me.GroupBox1.Controls.Add(Me.ComboBox1)
        Me.GroupBox1.Location = New System.Drawing.Point(12, 12)
        Me.GroupBox1.Name = "GroupBox1"
        Me.GroupBox1.Size = New System.Drawing.Size(215, 71)
        Me.GroupBox1.TabIndex = 0
        Me.GroupBox1.TabStop = False
        Me.GroupBox1.Text = "Connection"
        '
        'ComboBox2
        '
        Me.ComboBox2.Anchor = CType(((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Left) _
            Or System.Windows.Forms.AnchorStyles.Right), System.Windows.Forms.AnchorStyles)
        Me.ComboBox2.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList
        Me.ComboBox2.FormattingEnabled = True
        Me.ComboBox2.Items.AddRange(New Object() {"300", "1200", "2400", "4800", "9600", "19200", "38400", "57600",
"74880", "115200", "230400", "250000"})
        Me.ComboBox2.Location = New System.Drawing.Point(83, 42)
        Me.ComboBox2.Name = "ComboBox2"
        Me.ComboBox2.Size = New System.Drawing.Size(126, 21)
        Me.ComboBox2.TabIndex = 4
        '
        'Label2
```

```
        '
        Me.Label2.AutoSize = True
        Me.Label2.Location = New System.Drawing.Point(7, 45)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New System.Drawing.Size(61, 13)
        Me.Label2.TabIndex = 3
        Me.Label2.Text = "Baud Rate:"
        '
        'Label1
        '
        Me.Label1.AutoSize = True
        Me.Label1.Location = New System.Drawing.Point(7, 16)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(70, 13)
        Me.Label1.TabIndex = 1
        Me.Label1.Text = "Select a port:"
        '
        'ComboBox1
        '
        Me.ComboBox1.Anchor = CType(((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Left) _
            Or System.Windows.Forms.AnchorStyles.Right), System.Windows.Forms.AnchorStyles)
        Me.ComboBox1.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList
        Me.ComboBox1.FormattingEnabled = True
        Me.ComboBox1.Location = New System.Drawing.Point(83, 13)
        Me.ComboBox1.Name = "ComboBox1"
        Me.ComboBox1.Size = New System.Drawing.Size(126, 21)
        Me.ComboBox1.TabIndex = 0
        '
        'InputTextbox1
        '
        Me.InputTextbox1.Anchor = CType((((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Bottom) _
            Or System.Windows.Forms.AnchorStyles.Left) _
            Or System.Windows.Forms.AnchorStyles.Right), System.Windows.Forms.AnchorStyles)
        Me.InputTextbox1.Enabled = False
        Me.InputTextbox1.Location = New System.Drawing.Point(12, 89)
        Me.InputTextbox1.Name = "InputTextbox1"
        Me.InputTextbox1.Size = New System.Drawing.Size(215, 20)
        Me.InputTextbox1.TabIndex = 1
        '
        'Button1
```

```
        '
        Me.Button1.Anchor = CType((System.Windows.Forms.AnchorStyles.Top Or System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
        Me.Button1.Location = New System.Drawing.Point(242, 12)
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New System.Drawing.Size(75, 71)
        Me.Button1.TabIndex = 3
        Me.Button1.Text = "Connect"
        Me.Button1.UseVisualStyleBackColor = True
        '
        'SerialPort1
        '
        Me.SerialPort1.DiscardNull = True
        Me.SerialPort1.DtrEnable = True
        '
        'RichTextBox1
        '
        Me.RichTextBox1.Anchor = CType((((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Bottom) _
            Or System.Windows.Forms.AnchorStyles.Left) _
            Or System.Windows.Forms.AnchorStyles.Right), System.Windows.Forms.AnchorStyles)
        Me.RichTextBox1.Location = New System.Drawing.Point(12, 115)
        Me.RichTextBox1.Name = "RichTextBox1"
        Me.RichTextBox1.ReadOnly = True
        Me.RichTextBox1.Size = New System.Drawing.Size(215, 299)
        Me.RichTextBox1.TabIndex = 4
        Me.RichTextBox1.Text = ""
        '
        'TextBox2
        '
        Me.TextBox2.Location = New System.Drawing.Point(339, 131)
        Me.TextBox2.Name = "TextBox2"
        Me.TextBox2.Size = New System.Drawing.Size(100, 20)
        Me.TextBox2.TabIndex = 5
        '
        'TextBox1
        '
        Me.TextBox1.Location = New System.Drawing.Point(323, 2)
        Me.TextBox1.Multiline = True
        Me.TextBox1.Name = "TextBox1"
        Me.TextBox1.Size = New System.Drawing.Size(100, 123)
        Me.TextBox1.TabIndex = 6
```

```
'
'TextBox3
'
Me.TextBox3.Location = New System.Drawing.Point(339, 157)
Me.TextBox3.Name = "TextBox3"
Me.TextBox3.Size = New System.Drawing.Size(100, 20)
Me.TextBox3.TabIndex = 7
'
'TextBox4
'
Me.TextBox4.Location = New System.Drawing.Point(339, 183)
Me.TextBox4.Name = "TextBox4"
Me.TextBox4.Size = New System.Drawing.Size(100, 20)
Me.TextBox4.TabIndex = 8
'
'Chart1
'
ChartArea1.Name = "ChartArea1"
Me.Chart1.ChartAreas.Add(ChartArea1)
Legend1.Name = "Legend1"
Me.Chart1.Legends.Add(Legend1)
Me.Chart1.Location = New System.Drawing.Point(458, 57)
Me.Chart1.Name = "Chart1"
Series1.ChartArea = "ChartArea1"
Series1.ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line
Series1.Legend = "Legend1"
Series1.Name = "Humidity"
Series2.ChartArea = "ChartArea1"
Series2.ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line
Series2.Legend = "Legend1"
Series2.Name = "Temperature"
Me.Chart1.Series.Add(Series1)
Me.Chart1.Series.Add(Series2)
Me.Chart1.Size = New System.Drawing.Size(435, 317)
Me.Chart1.TabIndex = 9
Me.Chart1.Text = "Chart1"
'
'Timer1
'
Me.Timer1.Interval = 1
'
```

```
    'Form1
    '
    Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
    Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
    Me.ClientSize = New System.Drawing.Size(947, 427)
    Me.Controls.Add(Me.Chart1)
    Me.Controls.Add(Me.TextBox4)
    Me.Controls.Add(Me.TextBox3)
    Me.Controls.Add(Me.TextBox1)
    Me.Controls.Add(Me.TextBox2)
    Me.Controls.Add(Me.RichTextBox1)
    Me.Controls.Add(Me.Button1)
    Me.Controls.Add(Me.InputTextbox1)
    Me.Controls.Add(Me.GroupBox1)
    Me.MinimumSize = New System.Drawing.Size(300, 300)
    Me.Name = "Form1"
    Me.Text = "Arduino Serial Monitor"
    Me.GroupBox1.ResumeLayout(False)
    Me.GroupBox1.PerformLayout()
    CType(Me.Chart1, System.ComponentModel.ISupportInitialize).EndInit()
    Me.ResumeLayout(False)
    Me.PerformLayout()

End Sub

Friend WithEvents GroupBox1 As GroupBox
Friend WithEvents ComboBox1 As ComboBox
Friend WithEvents ComboBox2 As ComboBox
Friend WithEvents Label2 As Label
Friend WithEvents Label1 As Label
Friend WithEvents InputTextbox1 As TextBox
Friend WithEvents Button1 As Button
Friend WithEvents SerialPort1 As IO.Ports.SerialPort
Friend WithEvents RichTextBox1 As RichTextBox
Friend WithEvents TextBox2 As TextBox
Friend WithEvents TextBox1 As TextBox
Friend WithEvents TextBox3 As TextBox
Friend WithEvents TextBox4 As TextBox
Friend WithEvents BackgroundWorker1 As System.ComponentModel.BackgroundWorker
Friend WithEvents BackgroundWorker2 As System.ComponentModel.BackgroundWorker
Friend WithEvents Chart1 As DataVisualization.Charting.Chart
```

```
    Friend WithEvents Timer1 As Timer
End Class
```