Steuerung und Regelung einer Hausklimatisierung

Selim Jaeschke, Raphael Biermann

Matrikel-Nr. von Raphael B.
7205191

Matrikel-Nr. von Selim J.
7205879

Prüfer:
Friedrich Haase,
Michael Janke

Inhaltsverzeichnis

Einleitung

Dieses Dokument ist ein Bericht über das Projekt Steuerung und Regelung einer Hausklimatisierung, das im Rahmen des IT-Projekts als eine Problemstellung angeboten wird. Thematisiert wird die Steuerung von Haushaltsgeräten aus dem Bereich Klima und Heizen im Fokus der Optimierung der klimatischen Bedingungen in einem Verbraucherhaushalt. Heizung, Klimaanlage und Wärmetauscher werden in Bezug auf ihre Funktionsweise untersucht und in einer Micro-Controller Simulation geregelt. Es wird untersucht inwiefern die verschiedenen Komponenten jeweils allein und miteinander arbeiten können, um die optimalen Bedingungen einzuhalten.

Im Fokus des Projekts steht die Programmierung einiger Regler für die Einflussgrößen der Simulation in Anbetracht von realen Klimageräten in einem Haushalt, sowie deren logische Vernetzung.

Weiterhin werden mögliche Erweiterungen vorgestellt und es wird diskutiert, was noch ergänzt werden könnte.

Problemstellung

Es ist eine Arduino Nano Schaltung aus zwei Microcontrollern gegeben. Diese sind über den I^2C Bus vernetzt.

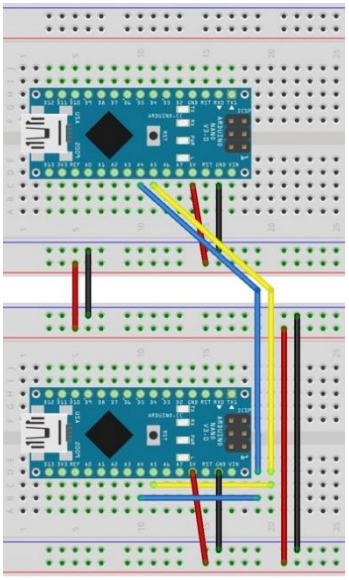

Abbildung 1: Master-Slave Schaltung der Arduinos.

Auf dem in Abbildung 1 oben dargestellten Arduino Nano läuft eine Simulation eines Hauses, das nur leicht isoliert ist [1], was für die Simulation bedeutet, dass Temperaturschwankungen außerhalb des Hauses schnell auch das Klima innerhalb beeinträchtigen.
Die Regelung wird in dem Arduino unten umgesetzt. Gegeben ist die Grundstruktur des Programmes, das später alle Einflussgrößen steuern wird.

In dem Haus gibt es einige Regelgrößen, die geregelt werden sollen, um das Klima trotz äußerer Einflüsse auf einem Soll-Wert zu halten.
Die wichtigste Regelgröße ist dabei die Temperatur. Weiterhin gibt es noch die Luftfeuchtigkeit und CO2 Konzentration.

Um die Temperatur beeinflussen zu können, bietet die Simulation eine Heizung, sowie eine Kühlung an. Weiterhin gibt es einen Wärmetauscher, der für Luftaustausch und Wärmetransfer genutzt werden kann. Dieser kann zudem verwendet werden, um die CO2 Konzentration und Luftfeuchtigkeit anzupassen.

Zur Regelung dieser Geräte in der Simulation gibt es einige Parameter, die später diskutiert werden.

Die Regelung wird komplett in C++ programmiert, da dies die Standardprogrammiersprache der Arduino IDE, der offiziellen Programmierumgebung der Arduino Micro-Controller Familie, ist. Das Arduino Projekt ist Open-Source und die Micro-Controller können auch von Drittanbietern erworben werden. Hier werden jedoch Micro-Controller des offiziellen Herstellers Arduino SA verwendet.

Simulation

Auf dem Arduino Nano oben in Abbildung 1 läuft eine Simulation, die als Black-Box, also als ein geschlossenes System, dessen Inhalt unbekannt ist, angenommen wird.
Während die Abläufe innerhalb der Simulation großenteils unbekannt sind, ermöglicht eine serielle Ausgabe folgender Größen über den I^2C Bus einen Einblick.
Weiterhin können beide Arduinos über den seriellen Port über USB verbunden werden, um gezielt Werte auszulesen und zu visualisieren.

Die Ausgabe von Regelgrößen wie dem aktuellen Tastverhältnis der Heizung ist für die Regelung nicht relevant, kann jedoch für die Anzeige auf einem Userinterface genutzt werden.

Tabelle 1: Simulationsparameter und ihre Bedeutung.

Kommando	Bedeutung für die Regelung
T?	Zeit in Sekunden, wie lange die Simulation läuft. Kann für Timings genutzt werden.
I?	Die Temperatur im Haus. Die Heizung (H) kann genutzt werden, um die Temperatur anzuheben. Eine Umluftkühlung (F) kann die Temperatur senken.
O?	Temperatur außerhalb des Hauses. Kann mithilfe des Wärmetauschers genutzt werden, wenn ein Temperaturaustausch erfolgen soll.
H?	Gibt das aktuelle Tastverhältnis der Heizung in % wieder. Diese Größe wird von der Regelung gesteuert.
E?	Gibt die aktuelle Luftaustauschmenge des Wärmetauschers in % wieder. Dies wird von der Regelung gesteuert.
e?	Gibt die aktuelle Wärmetransfermenge des Wärmetauschers in % wieder. Dies wird von der Regelung gesteuert.
i?	Die aktuelle Luftfeuchtigkeit im Haus. Sollte in einem Bereich von 40%-60% liegen. Eine zu hohe Luftfeuchtigkeit kann durch Kondensation Schimmel verursachen. Eine zu niedrige Luftfeuchtigkeit ist zwar für das Haus gut, kann jedoch gesundheitliche Probleme wie beispielsweise Nasenbluten verursachen. Es gibt zwar noch keinen Luftbefeuchter und Luftentfeuchter in der Simulation, allerdings kann über den Wärmetauscher mit aktivem Wärmetransfer und Luftdurchsatz die Luft ausgetauscht werden, ohne dass die Temperatur im Haus verändert wird.
o?	Die aktuelle Luftfeuchtigkeit außerhalb des Hauses.
C?	Die Kohlenstoffdioxidkonzentration. Ein Wert von <1000 ppm gilt als optimal.
k?	Der summierte Energieverbrauch aller Klimageräte. Kann angezeigt werden.
S?	Temperatursollwert in der Simulation. Dieser Wert wird von der Simulation nicht berücksichtigt. In der Umsetzung wird dieser Wert lokal definiert, um den Bus zu entlasten.
w=x	Einstellen von Winter oder Sommer. w=1 ist Winter und w=0 ist Sommer.
W?	Abfragen von Warnungen oder Fehlern
f	Die festgehaltenen Einstellungen können hier abgerufen werden
F?	Gibt das aktuelle Tastverhältnis der Klimaanlage in % wieder. Diese Größe wird von der Regelung gesteuert.
P?	Gibt die Personenanzahl im Haus an
p=x	Steht für den Party Modus (1), wodurch die Aktivität der Personen und damit Temperatur, CO2 und Luftfeuchtigkeit im Haus erhöht wird.
R	Der Reset Knopf der Simulation. Dadurch wird die Simulation zurückgesetzt.
V=x	Verbose Einstellung. Damit wird vor den Werten, die von der Plant kommen, deren Kürzel angezeigt.
H=n	Tastverhältnis der Heizung von 0-100% einstellen. Dies kann manuell über die Eingabeaufforderung oder später automatisiert über den Regler erfolgen.

E=n	Wärmetauscher Durchsatz von 0-100% einstellen. In der Grundfunktion ist dies wie ein Lüfter zu sehen. So kann auch Frischluft ausgetauscht werden.
e=n	Wärmetauscher Wärme-Transfer von 0-100% einstellen. Der Wärmetransfer ist wichtig, wenn Wärme getauscht werden soll. Dies ist nützlich, wenn die Temperatur im Haus gehalten werden soll.
F=x	Kühlleistung der Umluftkühlung von 0-100% einstellen. Die kühlende Alternative der Heizung in der Simulation. Ansteuerung mit invertiertem Regler möglich.
P=n	Anzahl der Personen von 0-50 einstellen. Nach [2] erhöhen sich dadurch Luftfeuchtigkeit, CO2 Konzentration und Wärme.
f=x	Festhalten der Simulationswerte. Dadurch bleibt die Temperatur außerhalb konstant und es können Fehler auf den Regler zurückgeführt werden.

Eine Simulation mit realem Zeitverlauf würde einen sehr langen Beobachtungszeitraum erfordern. Daher läuft die Simulation in einem Zeitraffer (1s entspricht 5 min simulierte Zeit).

Heizung

Eine simple Heizung ist die Wärmequelle der Simulation und kann vom Controller mit einem Tastverhältnis von 0-100% angesteuert werden. Dabei ist das Tastverhältnis der prozentuale Anteil der Einschaltzeit der Heizung gegenüber der gesamten Einschaltperiode.
Wird die Heizung angesteuert, so reagiert die Simulation nicht direkt mit einer Temperaturänderung auf den neuen Heizwert. Wie in der Realität dauert es eine Weile, bis das System auf die Änderung reagiert, was sich bei der Regelung durch eine Verlängerung der Einschwingzeit zu bemerken macht.

Wärmetauscher

In dem Haus ist ein Wärmetauscher in einem Frischluftkanal installiert, der sich über Wärmetauscher Durchsatz in vol%/min, also die Fördermenge an Luft und Wärmetransfer jeweils von 0-100% einstellen lässt.
Ein Wärmetauscher ist in seiner Grundfunktion ein Apparat, in dem die Wärmeenergie eines strömenden Mediums auf ein anderes übertragen wird (vgl. [2]). Dieser Volumenstrom kann beispielsweise Wasser in einem Kesselkreislauf mit Puffer sein, wo ein Wassertank aufgewärmt wird und die Heizkreisläufe des Hauses durch den Tank geführt werden. Dabei wird lediglich die gespeicherte Wärme des Puffers auf die Rohre übertragen, während sich die Flüssigkeiten nicht mischen.
In diesem Beispiel wird auch ein Wärmetauscher nach dem Prinzip der indirekten Wärmeübertragung verwendet, der die Wärmeenergie von zwei entgegengesetzten Luftströmen tauscht. Diese sogenannte Wärmerückgewinnung ermöglicht einen Frischluftaustausch ohne einen Verlust von Energie in Form von verlorener Wärme oder sogar Kälte durch das Lüften [3], wenn der Wärmetransfer idealerweise bei 100% liegt.
Bei ausgeschalteten Wärmetransfer und eingeschaltetem Durchsatz verhält sich der Wärmetauscher wie ein geöffnetes Fenster und ein Luftaustausch ohne Energierückgewinnung ist möglich. Dieses Prinzip wird beispielsweise bei neuen Wärmetauschertrocknern verwendet, damit die warme Luft innerhalb des Gerätes bleibt und der Energieverbrauch sinkt.

Umluftkühlung

In dem Haus ist ein einfacher Umluftkühler installiert, der von 0-100% angesteuert werden kann. Auffällig ist, dass die Simulation sehr sensitiv auf die Umluftkühlung reagiert. Für die Implementierung bedeutet das, dass mögliche Koeffizienten kleiner gewählt werden müssen, damit Schwingungen vermieden werden.

Der Mensch

In der Simulation kann auch die Anzahl an Menschen in dem Haus eingestellt werden. Dabei ist zu beachten, dass ein Mensch auch eine Wärmequelle ist.
Ein Mensch strahlt je nach Aktivität etwa 100-300 Watt Wärme aus (vgl. [4]). Bei körperlicher Betätigung ist letzter Wert zu betrachten, da dieser ziemlich hoch ist.
Bei einer Anzahl von 10 Personen im Haus sind dies bereits 3000 Watt, was der Leistung einer stärkeren Elektroheizung entspricht. Weiterhin muss auf die CO2 Konzentration geachtet werden, da ein Mensch etwa 20l CO2 pro Stunde ausatmet (vgl. [4]). Ein Frischluftaustausch ist also unabdingbar.

Kommunikation über I^2C

Der I^2C Bus ist ein serieller Datenbus, der ursprünglich von Philips zur Kommunikation von Sensoren und Aktoren über kurze Distanzen in der Unterhaltungselektronik entwickelt wurde. Der Bus besteht aus einer Datenleitung (SDA) und einer Taktleitung (SCL), die für die Taktung der seriellen Datenübertragung bestimmt ist. Der Arduino Nano verwendet den analogen Port A4 für die Datenleitung und Port A5 für die Taktleitung.

Über den Bus kommunizieren alle angeschlossenen Treiber nach dem Client-Server Prinzip (damals Master-Slave). In diesem Fall ist der Server der Controller und alle anderen angeschlossenen Geräte sind die sogenannten Clients. Jeder Teilnehmer bekommt eine individuelle 7 Bit breite Adresse. Insgesamt sind 112 Geräte anschließbar, da 16 Adressen reserviert sind. Die Standardbusfrequenz beträgt 100kHz, diese ist auch in diesem Fall beibehalten. Die Kommunikation zwischen Controller und Plant wurde auf 100kHz festgelegt und das später hinzugefügte Display hat standardmäßig die gleiche Frequenz.
Über den Bus wird in 8-Bit Sequenzen kommuniziert.

Das Standard LCD 16x2 ist ohne weitere Module nicht I^2C kompatibel und muss über einen 8 Bit Vektor angesteuert werden. Mithilfe des Moduls LCM1602 kann diese Hürde jedoch überwunden werden, da dieses Modul das LCD I^2C fähig macht und mithilfe der Library *LiquidCrystal_I2C* in der Arduino IDE einfach angesteuert werden kann. Standardmäßig hat dieses Modul die Client-Adresse *0x27*, die hier aufgrund keiner Adresskonflikte beibehalten wurde. Adressänderungen könnten über das Verlöten der Kontakte A0 bis A2 an dem Modul vorgenommen werden.

Mithilfe der Library können einfach Wörter der Datentypen *char, byte, int, long* oder *string* angezeigt werden. Strings werden hier genutzt, um die Anzeigeelemente zu betiteln. Um nicht alle Nachkommastellen einer Zahl anzuzeigen und Platz zu sparen, können double-Variablen über die explizite Typkonvertierung mit *(int)* in Integer konvertiert werden, um nur die ganzzahlige Komponente anzuzeigen.

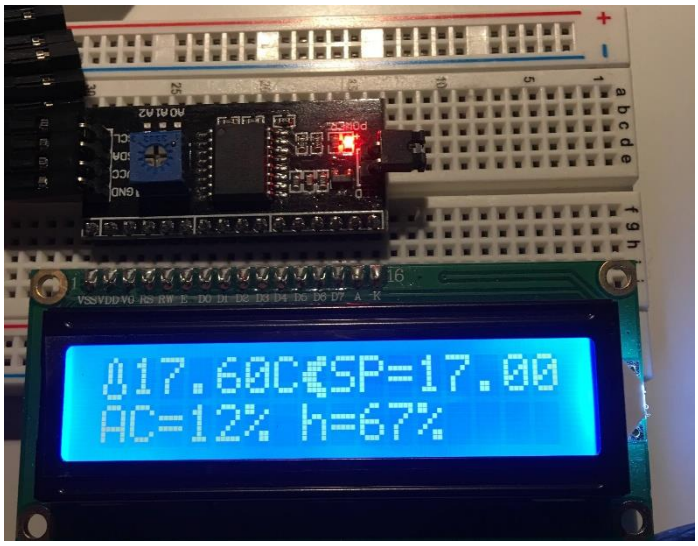

Abbildung 2: LCD-Anzeige mit Ist-Temperatur, Soll-Temperatur, Leistung der aktuell aktiven Einheit und Luftfeuchtigkeit.

Weiterhin hilfreich ist die Möglichkeit, benutzerdefinierte Charaktere anzuzeigen. Kleine Icons geben dem Nutzer Auskunft und es wird wenig Platz verbraucht. In **Fehler! V erweisquelle konnte nicht gefunden werden.** ist beispielsweise gerade der Nachtzeitplan aktiv und die Solltemperatur wird automatisch runtergeregelt. Es wird ein Mond angezeigt, um dies zu signalisieren. Bei Tageszeit wird eine Sonne angezeigt. Natürlich ist der Platz für Icons auf 5x8 Pixel begrenzt und es ist nur sinnvoll, eindeutig verständliche Icons zu zeigen. Eine Heizung oder Kühlung so darzustellen, dass es verständlich ist, ist auf der begrenzten Fläche schwierig. Es ist zu diskutieren, ob Symbole trotzdem sinnvoll sein könnten und wie weit eine Bedienungsanleitung Abhilfe schaffen würde.

Regelung

Die Logikfunktion

Um die Temperatur im Haus regeln zu können, bedarf es einer Logikfunktion *logic()*, die für jeden Fall, ausgehend den Werten innerhalb und außerhalb des Hauses, entscheidet, ob jeweils die Heizung, die Umluftkühlung oder der Wärmetauscher eingeschaltet werden soll.

Es gibt verschiedene Möglichkeiten, diese Logik umzusetzen.

Eine Möglichkeit ist die Orientierung an dem Jahreszeitenzustand *w* für Winter und Sommer, um entweder in einen Heizmodus oder Kühlmodus zu wechseln.
Dies wird von einigen Herstellern umgesetzt, in dem das jeweilige Datum im digitalen Thermostat hinterlegt wird. So wird eine Heizperiode für die kalten Monate definiert, in der die Heizung aktiv ist, während im Sommer die Heizung komplett ausgeschaltet wird.
Dies beweist sich jedoch nicht als optimal, da auch im Sommer die Temperatur in kalten Nächten stark fallen kann. Je nach Isolierung kann auch innerhalb des Hauses die Temperatur unterhalb der Wohlfühltemperatur fallen.
Am Anfang des Entwicklungszeitraumes dieses Projekts wurde die Steuerung genau auf diese Art umgesetzt. Da jedoch der Nutzer volle Kontrolle über das Gerät haben soll und auch beispielsweise im Sommer nachts die Möglichkeit haben soll, die Heizung einzuschalten, wurde eine Alternative gesucht.

Eine weitere Möglichkeit ist je nach Innentemperatur zu entscheiden, ob in den Heiz- oder Kühlmodus gewechselt werden soll.
Damit jedoch nicht die ganze Zeit zwischen Heizung und Kühlung hin und her geschaltet wird, wenn beispielsweise die Temperatur um den Sollwert schwingt, braucht es einen Puffer.

Der Puffer wird hier als Nebenbedingung konjunktiv verknüpft und wird als Zähler realisiert.

Fällt die Temperatur unter den Sollwert, so wird der Zähler inkrementiert und ist stets positiv. Dadurch wird die Heizung eingeschaltet.
Sobald die Temperatur über den Sollwert steigt, wird der Zähler dekrementiert. So ist die Bedingung *dIndoorTemperature > soll* zwar erfüllt und es steht nahe, die Kühlung einzuschalten. Der Pufferzähler ist jedoch positiv und lässt dies erst nicht zu. So hat das System genug Zeit sich einzuschwingen und die Umluftkühlung wird nur eingeschaltet, wenn es wirklich notwendig ist.

Temperaturregelung

Die simulierten Umwelteinflüsse außerhalb des Hauses bewirken eine sinusartige Temperaturkurve, die die Temperatur innerhalb des Hauses aufgrund der leichten Isolierung beeinträchtigt.

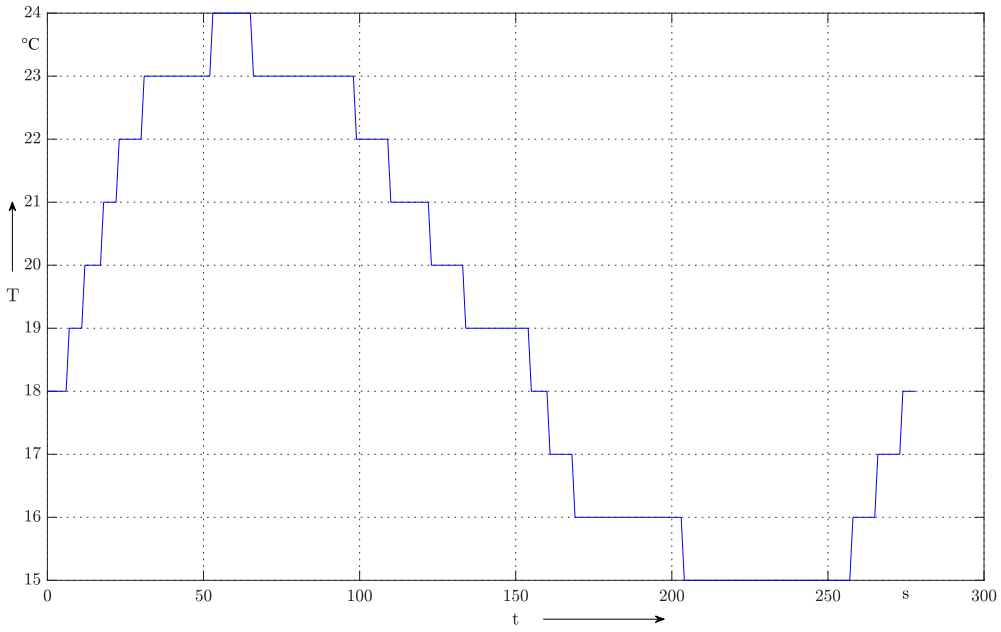

Diagramm 1: Temperaturverlauf außerhalb des Hauses während des Sommers.

Diagramm 1 zeigt beispielhaft, wie der Temperaturverlauf außerhalb des Hauses aussehen kann. Dabei zeigt sich ein deutliches Temperaturspektrum, beispielswiese zwischen Nacht und Tag, das hier von 15° bis 24° reicht. Ziel ist es, die Temperatur auf dem voreingestellten Sollwert zu halten. Dazu gibt es zwei Möglichkeiten, Heizen und Kühlen, dessen Regelung und die damit verbundene Problematik in den nächsten Abschnitten erläutert wird.

Heizen

In diesem Abschnitt wird diskutiert, wie eine Grundregelung aufgebaut werden kann. Da die Regelungen für die Umluftkühlung und weitere Elemente hierauf aufbauen, wird dieser Abschnitt besonders genau beschrieben.

Der erste Anfang ist die Automatisierung der Heizung durch das Definieren des Heizwertes, der über I^2C an die Simulation übergeben wird.

Dafür wird die vordefinierte Funktion *CreateNextSteadyCommand(char szCommand[])* verwendet. Die Funktion generiert alle 100ms einen String, der *szCommand* definiert, der an die Simulation gesendet wird. So werden nacheinander die Werte für beispielsweise die Innenraumtemperatur, die Luftfeuchtigkeit, die CO2 Konzentration und so weiter abgefragt und basierend darauf müssen die Regelparameter reagieren. Die Liste mit Abfragen und neuen Anfragen kann stetig durch das Hinzufügen von neuen Cases in der Switch-Anweisung erweitert werden.
Für die Heizung ist der Wert der Innenraumtemperatur relevant und muss gespeichert werden. Danach kann aus dem Temperaturwert Wert für das Tastverhältnis der Heizung generiert werden.
In dieser Realisierung wird für die Innenraumtemperatur die Variable *dIndoorTemperature* verwendet. Das Taktverhältnis könnte beispielsweise immer, wenn der Heizwert unter den Sollwert fällt, auf 100% gestellt werden. Dies würde jedoch ein starkes Schwingen der Innenraumtemperatur verursachen, da die Innentemperatur sehr sensitiv gegenüber der Wärmequelle ist und gleichzeitig eine Änderung erst nach einer kurzen Latenz sichtbar wird.
Eine andere Möglichkeit sind lineare und quadratische Regler, die je nach Temperatur ein unterschiedliches Tastverhältnis berechnen.
Nach Implementation eines vergleichbaren Reglers zeigte sich folgender Temperaturverlauf (grün).

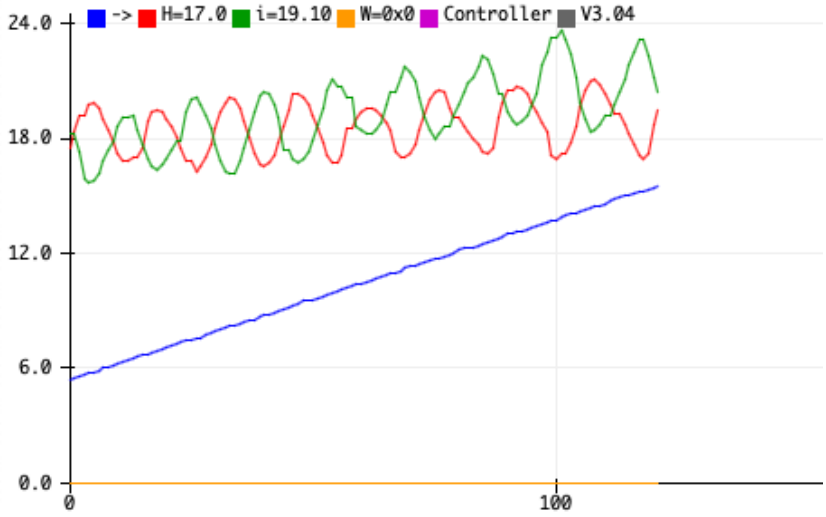

Abbildung 3: Screenshot der Innenraumtemperaturkurve

Es zeigt sich deutlich, dass die Temperatur zwar um einen Wert schwingt, jedoch keine Dämpfung dieser Schwingung auf längere Zeit zu erwarten ist.

Daher wird ein anderer Regler benötigt, der nicht so stark schwingt: Der **PI-Regler**.

Der PI-Regler ist eine Erweiterung der linearen Regelung. Er besteht aus zwei Teilen, dem P-Glied und dem I-Glied.
Das P-Glied ist der proportionale Teil der Regelung. Es besteht aus dem jeweiligen absoluten Fehler der Temperatur und einem Koeffizienten:

$$P = p * \Delta T = p * (T_{soll} - T_{real})$$

Formel 1: P-Glied

Der Koeffizient *p* muss entsprechend gewählt werden. Hier wurde dieser nach dem Trial & Error Prinzip ermittelt.

Das I-Glied ist der Integrale Anteil des Reglers. Durch die Summierung aller Fehler über die Laufzeit des Reglers wird dieser Anteil immer größer und korrigiert den proportionalen Anteil des Reglers, wenn dieser sich nicht weiter ändert.

$$I = I + i * \Delta T$$

Formel 2: I-Glied.

Der Koeffizient des I-Glieds muss sehr klein gewählt werden ($i \ll 1$), weil die Logikfunktion die Regelung alle 100ms aufruft und das I-Glied dementsprechend schnell wächst. Die Latenz der Regelung ist jedoch groß und der Soll-Wert wird schnell übertroffen.
Folgendes Diagramm zeigt den aktiven PI-Regler.

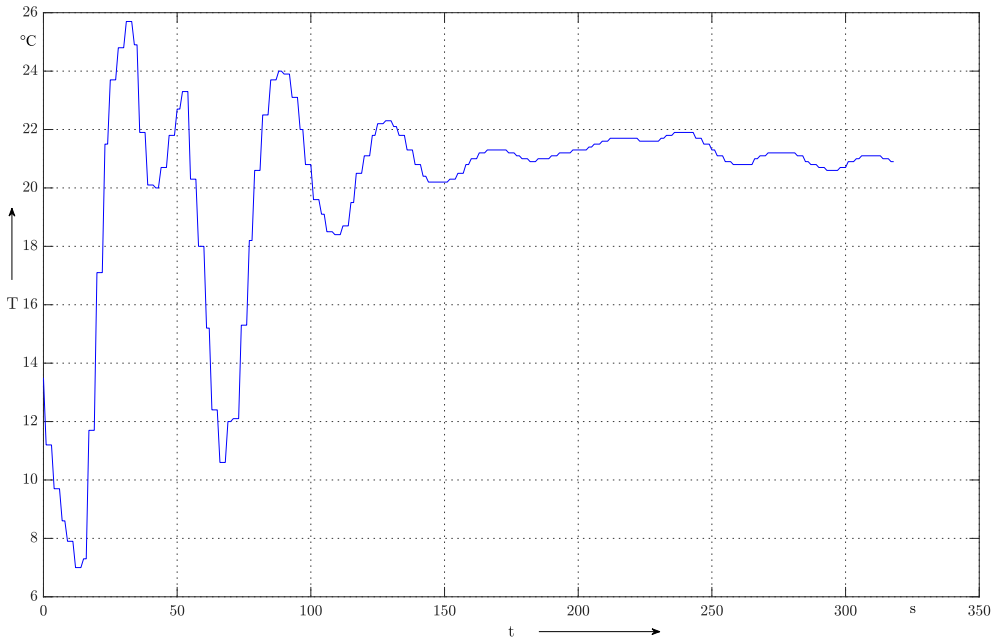

Diagramm 2: Verlauf der Temperatur im Haus während des Wintermodus bei laufender Heizung mit Sollwert T=21°C.

In Diagramm 2 ist der Temperaturverlauf zu sehen, nachdem im Winter die Heizung, mit dem Ziel die Temperatur im Haus auf 21°C zu regeln, eingeschaltet wurde. Zu Anfang benötigt die Heizung Zeit, um sich auf die gewünschte Soll-Temperatur einzustellen. Ein Problem ist, dass die Regelung aktiv gegen Temperaturänderungen regeln muss, die hier aufgrund des Zeitraffers sehr schnell auftreten. Die Einschwingdauer für einen äußerst präzisen Wert nahe des Sollwerts nimmt genauso wie in der Realität viel Zeit in Anspruch. Umso länger die Heizung aktiv ist, desto präziser wird die Temperatur, wobei sich die tatsächliche Temperatur im Haus etwa in einem Toleranzrahmen von +/- 1°C einschwingt.

Der Wärmetauscher kann hier als Unterstützung der Heizung genutzt werden, wenn die Außentemperatur größer beziehungsweise gleich dem Sollwert ist.
Dafür muss der Wärmetransfer jedoch ausgeschaltet werden. Der PI-Regler kann genau so wie hier auf den Durchsatz des Wärmetauschers angewendet werden.

Kühlen

Der PI-Regler aus dem vorherigen Abschnitt kann invertiert auch auf die Umluftkühlung angewendet werden. Da der Fehler ΔT bei steigender Temperatur negativer wird, reicht es, ein negatives Vorzeichen hinzuzufügen. Der Regler funktioniert mit gleichen Koeffizienten ähnlich gut.
Im Sommer gibt es die Problematik kalter Nächte, die die Regelung beeinträchtigen. Im Winter muss der PI-Regler lediglich gegen Kälte anheizen. Im Sommer kann jedoch die Temperatur in der Nacht fallen und die Heizung muss eingeschaltet werden. Dadurch entstehen neue Einschwingprozesse. Diese Problematik ist in Diagramm 3 dargestellt. Hier ist der Mittelwert in etwa 21°C, was der Soll-Temperatur entspricht. Die Einschwingvorgänge nach dem Umschalten von Winter auf Sommer liegen alle in einem Bereich von $\pm 2°C$.

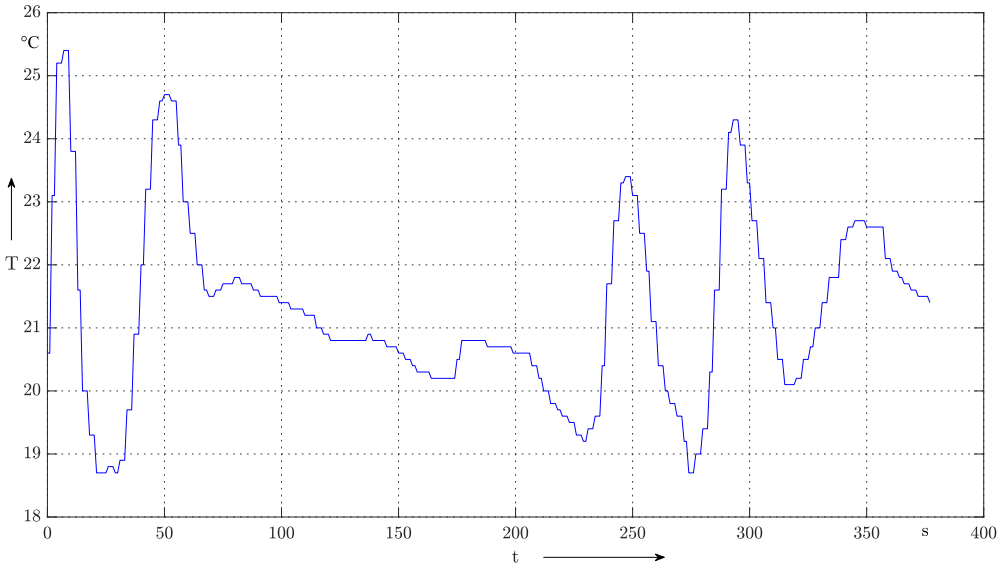

Diagramm 3: Verlauf der Innentemperatur bei laufender Kühlung mit Sollwert T=21°C.

Wenn die Außentemperatur kleiner als der Soll-Wert ist, kann durch PI geregeltes Lüften mittels des Wärmetauschers die Kühlung unterstützt werden. Dafür muss der Wärmetransfer ausgeschaltet werden.

Zeitplan

Wie in der Realität ist die Soll-Temperatur abhängig von den persönlichen Vorlieben der Bewohner des Hauses.
Generell ist die Wohlfühltemperatur tagsüber höher als nachts und beträgt je nach Aktivität 20-22°C. Nachts sollte eine Temperatur von 16°-18° im Haus eingestellt sein, damit optimale Schlafbedingungen gegeben sind.
Daher bietet es sich an, diese Wohlfühltemperaturen schon vorher zu implementieren, damit diese ohne weitere Einstellungen vornehmen zu müssen immer den richtigen Wert hat. So könnte beispielsweise die Temperatur von 22 Uhr nachts bis 6 Uhr morgens auf 17°C und die restliche Zeit tagsüber auf 20° eingestellt werden.
Damit der Benutzer trotzdem Änderungen vornehmen kann, muss jedoch gespeichert werden, ob es manuelle Einstellungen am Tag gab, damit diese nicht vom Programm überschrieben werden.

Natürlich finden durch das aktive Regeln der Temperatur neue Einschwingvorgänge statt, wenn die Abstände zu groß sind. Das ist jedoch normal und findet aus eigener Erfahrung auch bei vielen Thermostaten so statt.

Hardware-Erweiterungen

Wie oben erwähnt, wurde ein Display an den Controller angeschlossen, um die aktuelle Temperatur und andere Werte auszulesen.
Um mit der Simulation zu interagieren, wurden zwei Push-Buttons angeschlossen.
Die Push-Buttons sind über die digitalen Eingänge D2 und D3 an den Controller Arduino angeschlossen.

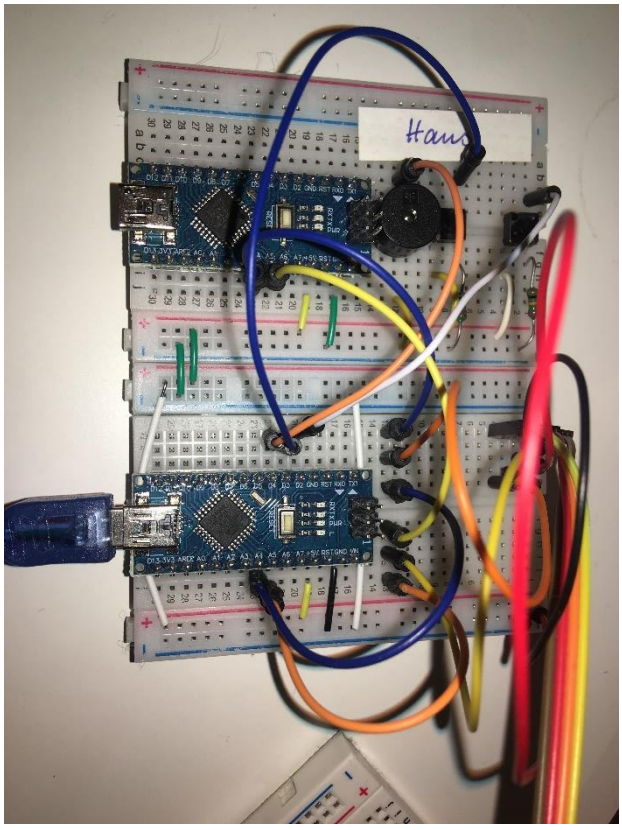

Abbildung 4: Foto der Schaltung.

Wenn die Schalter offen sind, sind die digitalen Ports je über einen 10k Pull-Down Widerstand an der Masse angeschlossen, damit auf den Ports 0V und damit eine logische 0 anliegt.
Beim Drücken der Schalter werden die Ports mit der 5V Spannungsversorgung des Arduinos verbunden, wodurch der Controller eine logische 1 liest.

Im Controller kann der Zustand der digitalen Ports in einer Variablen gespeichert werden. Die Push-Buttons sind standardmäßig nicht entprellt. Weiterhin ist die Push-Button Variable die ganze Zeit *high,* wenn der Button gedrückt wird. Dies kann jedoch einfach mit den Timing-Funktionen des Arduinos gelöst werden, da keine höheren Ansprüche an die Buttons gestellt werden. So wird jede Sekunde der Zustand der Ports gelesen.

Weiterhin wurde ein Lautsprecher installiert, der über Warnmeldungen akustisch informiert. Dies gestaltete sich als sinnvoll, da normalerweise kein Endnutzer ständig auf sein Thermostat schaut. Da hohe CO2 Konzentrationen von mehr als 1000ppm für Unwohl sorgen und eine signifikante Steigerung darüber hinaus sogar ungesund ist, wurde dafür ein Alarm installiert. Auch die anderen Warnmeldungen für zu niedrige und zu hohe Temperatur werden akustisch ausgegeben.
Der Lautsprecher ist ein passiver Piezo Lautsprecher, der über die Digitalen IO Ports angeschlossen werden kann.

Folgende in Fritzing visualisierte Schaltung zeigt den Gesamtaufbau:

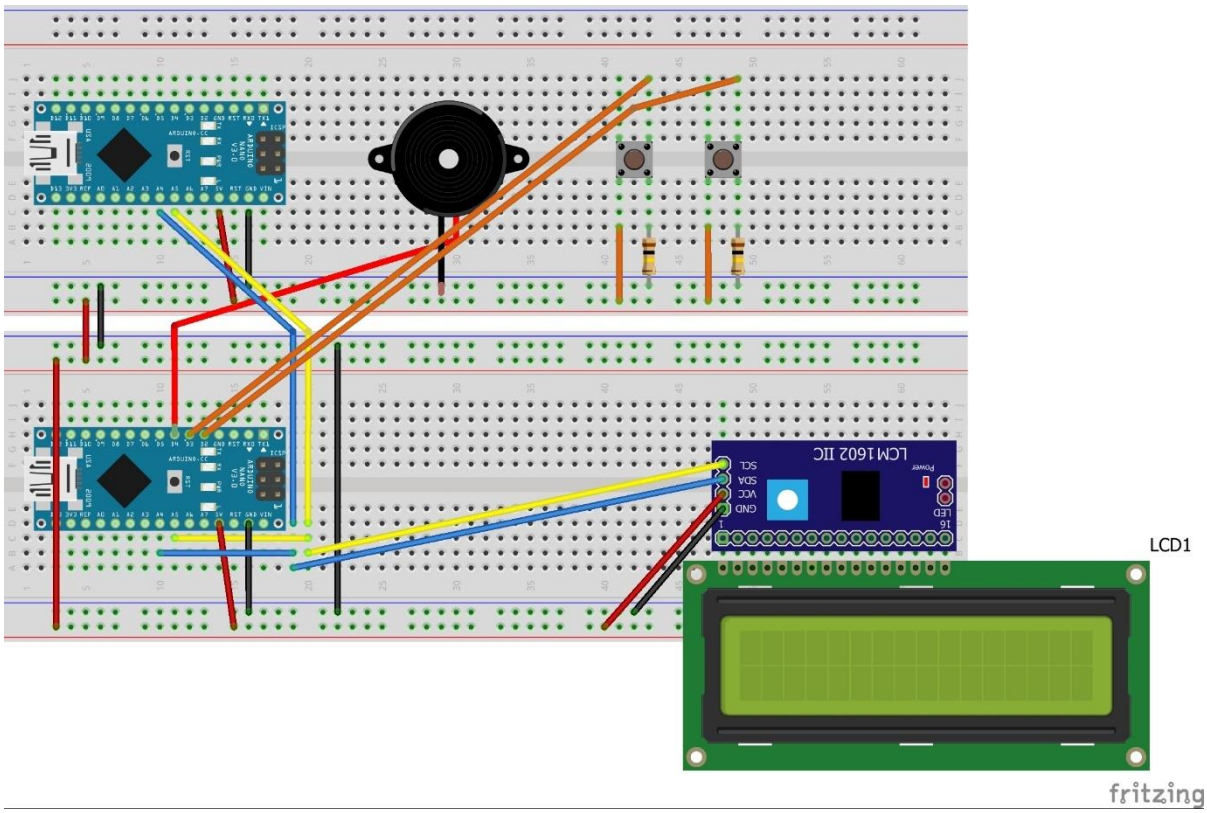

Abbildung 5: Schaltung mit Addons in Fritzing.

Ausblick

Mit weiterer Hardware können noch weitere Funktionen realisiert werden. Interessant wäre die Einbindung der Schaltung ins Netzwerk.

Für diesen Zweck können die ESP8266 Microcontroller verwendet werden, die auch Arduino kompatibel sind und mit der Arduino IDE in C++ programmiert werden können. Aufgrund des geringen Preises eignen sie sich für viele weitere Projekte im Smart-Home Bereich wie Alarmanlagen, Wetterstationen und sogar Klimaautomatisierung.
Da die Simulation teilweise sehr realistisch ist, könnten die hier programmierten Regler voraussichtlich einfach auf andere reale Projekte übertragen werden.

Ein Raspberry Pi mit Home-Assistant könnte als Server dienen, um alle Sensoren und Aktoren im Haus zu verwalten. Praktischerweise können so auch viele andere Geräte wie beispielsweise die Google Home Reihe für Spracherkennung eingebunden werden.

Zusammenfassung

Zusammenfassend lässt sich sagen, dass die im Projekt erarbeiteten Regler und Logiken für die Hausklimatisierung gut funktionieren, was sich an dem Zeitverhalten der Temperatur im Haus zeigt. Die Temperaturwerte befinden sich nach dem Einschwingen großenteils in dem angepeilten Toleranzbereich von $\pm 1°C$. Gleichzeitig zeigen sich Schwächen der Simulation durch den Zeitraffer und dementsprechende Einschwingvorgänge. In der Realität können

öfter neue Temperaturwerte gemessen werden, wodurch schneller auf Veränderungen reagiert werden kann. Es ist weiterhin zu diesem Aspekt zu analysieren, ob ein weiteres differentielles Glied im Regler dort vorteilhaft wäre.
Aus eigener Erfahrung haben Thermostate im Haushalt auch Probleme mit schwingendem Verhalten. Dies könnte unteranderem daran liegen, dass der Regler auf die Bedingungen angepasst werden muss. Es kann auch sein, dass nur Hersteller von Thermostaten höherer Preisklassen PI(D) Regler verbauen.
Weiterhin zeigen sich viele Möglichkeiten zur Erweiterung des Systems und Chancen für andere Projekte.

Literaturverzeichnis

[1] F. Haase, „Hausklimatisierung,“ FH Dortmund, Dortmund, 2020.

[2] P. D.-I. E. Specht, „Der Mensch als wärmetechnisches System,“ Institut für Strömungstechnik und Thermodynamik, Magdeburg, 17.05.2005.

[3] H. Titze, Elemente des Apparatebaues, Berlin: Springer, 1967.

[4] Wikipedia, „https://de.wikipedia.org/wiki/W%C3%A4rmer%C3%BCckgewinnung,“ 19 Oktober 2020. [Online]. Available: https://de.wikipedia.org/wiki/W%C3%A4rmer%C3%BCckgewinnung. [Zugriff am 30 01 2021].

[5] P. J. Plate, „Der I2C-Bus,“ 12 September 2017. [Online]. Available: http://www.netzmafia.de/skripten/hardware/Arduino/Programmierung/wire_library.html. [Zugriff am 31 Januar 2021].