HAW HAMBURG

Bachelorarbeit

Thorben Strübing

Entwurf, Aufbau und Test einer adaptiven Heizsteuerung für ein multifunktionales Leichtbaupaneel

Fakultät Technik und Informatik
Department Fahrzeugtechnik und Flugzeugbau

Faculty of Engineering and Computer Science
Department of Automotive and Aeronautical Engineering

Thorben Strübing

Entwurf, Aufbau und Test einer adaptiven Heizsteuerung für ein multifunktionales Leichtbaupaneel

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Mechatronik
am Department Fahrzeugtechnik und Flugzeugbau
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Erstprüfer: M. Sc. Maximilian Schutzeichel
Zweitprüfer: Prof. Dr.-Ing. habil. Thomas Kletschkowski

Abgabedatum: 17.April.2020

Zusammenfassung

Thorben Strübing

Thema der Bachelorarbeit

Entwurf, Aufbau und Test einer adaptiven Heizsteuerung für ein multifunktionales Leichtbaupaneel

Stichworte

Adaptive Temperaturregelung, Oberflächentemperaturaufnahme, De-Icing

Kurzzusammenfassung

De-Icing Systeme dienen in der Luftfahrt als eines der wichtigsten Sicherheitssysteme. Sie ermöglichen eine Enteisung an strömungsführenden Flugzeugstrukturen, welches für optimale Flugeigenschaften essenziell ist. Das am weitesten verbreitete De-Icing System in kommerziellen Turbofan-Flugzeugen ist das Zapfluftsystem. Dieses System trägt durch seine Beschaffenheit permanent zur Gesamtmasse des Flugzeugs bei, insbesondere auch dann, wenn keine Enteisung nötig ist. Es sorgt demnach doppelt für einen höheren Treibstoffverbrauch, indem es die Systemmasse vergrößert und durch die Zapfluftentnahme die Triebwerkseffizienz reduziert. Die zusätzliche Nutzung lasttragender Strukturelemente zur Abgabe Joulescher Wärme könnte das Zapfluftsystem ablösen. Zur Untersuchung des Potentials soll ein System entwickelt werden, welches die Oberfläche eines bereitgestellten multifunktionalen Leichtbaupaneel durch eine adaptiv gestaltete Heizrate auf eine einstellbare Temperatur erhitzt und diese anschließend hält. Das System soll dabei durch eine geeignete Sensoranordnung die Oberflächentemperaturverteilung diskret aufzeichnen und anschließend die relevanten Prozessdaten für eine externe Auswertung abspeichern. Die Daten sollen dabei helfen, Kenntnislücken bezüglich der Wärmeausbreitung an der Oberfläche und deren Regulierbarkeit zu schließen. Zudem sollen Erkenntnisse über das Systemverhalten herausgearbeitet werden. Nach Abschluss einer domänenspezifischen Entwicklung der Unterkomponenten und der Systemintegration wird das System verifiziert und das Ergebnis der Entwicklung anhand eines Anforderungsabgleiches bewertet. Abschließend werden anhand einer fallspezifischen Energiebilanz die Möglichkeiten zu einer zeitlichen und energetischen Optimierung des Prozesses vorgestellt.

Thorben Strübing

Title of the paper

Design, construction and test of an adaptive heating control for a multifunctional lightweight panel

Keywords

Adaptive temperature control, surface temperature recording, De-icing

Abstract

De-icing systems serve as one of the most important security systems in commercial aircraft. They enable ice removal on aerodynamic aircraft structures, which is essential for optimal flight characteristics. The bleed air system is the most common de-icing system in commercial turbofan aircraft. Due to its presence, this system adds permanently to the total mass of the aircraft, especially when de-icing is not necessary. It therefore leads to higher fuel consumption, by increasing the aircraft mass and reducing the engine efficiency through the extraction of bleed air. Using load-bearing structural elements to additionally dissipate joule heat could replace the whole bleed air system. To investigate the potential, a system is to be developed, which heats the surface of a provided multifunctional lightweight panel by an adaptively designed heating rate to an adjustable temperature and then maintains this temperature. The system should record the surface temperature distribution discretely using a suitable sensor arrangement and then save relevant process data for external evaluation. The data is intended to help close gaps in the knowledge about heat propagation at the surface and its controllability. In addition, knowledge about system behavior should be elaborated. After completion of a domain-specific development of the sub-components and the system integration, the system is verified and the result of the development is evaluated on the basis of a requirement comparison. Finally, a case-specific energy balance is used to present the possibilities for optimising the process in terms of time and energy.

Inhaltsverzeichnis

Formelzeichen

h_{sp2}	Schichtdicke des Versuchspaneels aus Glasfaserfilament $[\mathrm{mm}]$
m_{sp2}	Massenbelegung einer Glasfaserfilament-Schicht $[\frac{\mathrm{g}}{\mathrm{m}^2}]$
$\hat{I}_P$	Spitzenwert des Heizstroms, der das Paneel durchfließt $[\mathrm{A}]$
$\hat{I}_{P_{real}}$	Realer Spitzenwert des Heizstrom, der das Paneel durchfließt $[\mathrm{A}]$
$\hat{U}_P$	Spitzenwert der am Paneel angelegten Spannung $[\mathrm{V}]$
$\hat{U}_{PMW_{Arduino}}$	Spannungspegel des Arduino-PWM Signals während der $\tilde{T}_{on}$-Phase $[\mathrm{V}]$
ρ_f	Spezifischer elektrischer Widerstand einer Rovingfaser $[\Omega \cdot \mathrm{cm}]$
$\tilde{T}_{off_{real}}$	Reale Zeitdauer eines Low-Pegels bei einem PWM-Signal einer Periode $[\mathrm{s}]$
$\tilde{T}_{off}$	Zeitdauer eines Low-Pegels bei einem PWM-Signal einer Periode $[\mathrm{s}]$
$\tilde{T}_{on_{real}}$	Reale Zeitdauer eines High-Pegels bei einem PWM-Signal einer Periode $[\mathrm{s}]$
$\tilde{T}_{on}$	Zeitdauer eines High-Pegels bei einem PWM-Signal einer Periode $[\mathrm{s}]$
$\tilde{T}_{PWM}$	Periodendauer des PWM-Signals $[\mathrm{s}]$
A_f	Querschnittsfläche einer Rovingsfaser $[\mathrm{mm}^2]$
d_f	Durchmesser einer Rovingfaser $[\mathrm{mm}]$
E_i	Aufgebrachte elektrische Energie je Heizabschnitten $[\mathrm{J}]$

f_{Ab} Nötige Abtastrate für eine Strommessung$[\frac{1}{\mathrm{s}}]$

f_{PWM} Frequenz des PWM-Signals $[\frac{1}{\mathrm{s}}]$

f_T Abtastrate der Temperaturmessung$[\frac{1}{\mathrm{s}}]$

h_{P1} Höhe des Prepreg-Versuchspaneels $[\mathrm{mm}]$

h_{P2} Höhe des Glasfaser-Versuchspaneels $[\mathrm{mm}]$

h_{sp1} Schichtdicke des Versuchspaneels aus Prepreg $[\mathrm{mm}]$

I_P Mittlerer Heizstrom, der das Paneel durchfließt $[\mathrm{A}]$

I_D Stromfluss durch den MOSFET $[\mathrm{A}]$

I_{F1-F4} Strom, der die Heizfäden 1-4 durchfließt $[\mathrm{A}]$

I_{Mess} Strommesswert des Stromsensors $[\mathrm{A}]$

$I_{P_{real}}$ Realer mittlerer Heizstrom, der das Paneel durchfließt $[\mathrm{A}]$

l_F Länge des verwendeten Rovings $[\mathrm{mm}]$

l_p Kantenlänge des quadratischen Versuchspaneels $[\mathrm{mm}]$

l_{BF} Breite der verwendeten Heizfaser $[\mathrm{mm}]$

l_{Z1} Breite des ersten Schichtzuschnittes $[\mathrm{mm}]$

l_{Z2} Breite des zweiten Schichtzuschnittes $[\mathrm{mm}]$

m_{sp1} Massenbelegung einer Prepreg-Schicht $[\frac{\mathrm{g}}{\mathrm{m}^2}]$

n_f Anzahl der Kohlefasern im Filamentgarn $[-]$

p Tastgrad $[-]$

P_{1-2} Potentialmesspunkte in einer Messbrücke $[-]$

R_f Widerstandswert einer Rovingfaser $[\Omega]$

R_T Widerstandswert eines temperaturveränderlichen Messwiderstandes $[\Omega]$

R_{2-4} Widerstandswerte konstanter Widerstände in einer Messbrücke $[\Omega]$

$R_{DS_{on}}$ Drain-zu-Source Widerstand des MOSFET im geschalteten Zustand $[\Omega]$

R_{F1-F4} Widerstände der Heizfasern $[\Omega]$

$R_{F_{theoretisch}}$ Theoretischer Widerstandswert eines Rovings $[\Omega]$

$R_{P_{theoretisch}}$ Theoretischer Widerstandswert des Versuchspaneels$[\Omega]$

R_P Widerstandswert des Versuchspaneels$[\Omega]$

T_0 Anfangstemperatur des Paneels $[°\mathrm{C}]$

T_{an} Korrigierte Temperatur der Paneeloberfläche $[°\mathrm{C}]$

T_{cam} Gemessene Temperatur der Wärmebildkamera $[°\mathrm{C}]$

$T_{F_{max}}$ Maximale zulässige Temperatur der Rovings während des Heizprozesses $[°\mathrm{C}]$

T_{Hys} Untere Grenze der Schalthysterese $[°\mathrm{C}]$

T_{in} Innenraumtemperatur des Temperaturprüfschrankes $[°\mathrm{C}]$

$T_{M_{1-3},max}$ Maximaler korrigierter Temperaturwert der Messarrays 1-3 $[°\mathrm{C}]$

$T_{M_{1-3},real,max}$ Maximaler realer Temperaturwert der Messarrays 1-3 $[°\mathrm{C}]$

$T_{M_{1-3}}$ Korrigierte Temperaturmessung der Messarrays 1-3 $[°\mathrm{C}]$

$T_{M_{rel,M1-3}}$ Normierte Temperaturmessung der Messarrays 1-3 $[°\mathrm{C}]$

T_{max_F} Maximale Temperaturbelastbarkeit des Rovings $[°C]$

$T_{max_{P1}}$ Maximale Temperaturbelastbarkeit des Prepreg-Paneels $[°C]$

$T_{max_{P2}}$ Maximale Temperaturbelastbarkeit des Glasfaser-Paneels $[°C]$

T_{sens} Gemessene Temperatur eines Temperatursensors $[°C]$

T_{Ziel} Einstellbare Zieltemperatur der Paneeloberfläche $[°C]$

U_P Gleichwert der am Versuchspaneel anliegenden Spannung $[V]$

U_{GS} Spannung zwischen Gate und Source $[V]$

U_{Mess} Potentialunterschied zwischen den Messpunkten P_1 und P_2 $[V]$

U_{out} Signalspannung des Stromsensors $[V]$

U_{R2} Abfallende Spannungen über dem Widerstand R_2 in einer Messbrücke $[V]$

U_{RT} Abfallende Spannungen über dem Widerstand R_T in einer Messbrücke $[V]$

U_{th} Schwellspannung (engl. threshold voltage) eines MOSFET $[V]$

Tabellenverzeichnis

Abbildungsverzeichnis

1. Motivation und Problemstellung

„Im Jahr 2018 reisten 1 100 Millionen Passagiere in der Europänischen Union (EU) mit dem Flugzeug. Dies ist ein Anstieg um 6% gegenüber dem Jahr 2017 und um 43 % gegenüber 2010. In diesem Zeitraum ist der Fluggastverkehr in der EU stetig gestiegen.“
[7]

In einem Zeitalter, in dem Fliegen trotz hohem Ausstoß von klimaschädlichen Abgasen immer populärer wird, gilt es Flugzeuge effizienter auszulegen, dabei aber auch die Sicherheit der Passagiere zu jedem Zeitpunkt zu gewährleisten

Flugzeuge sind sowohl am Boden als auch während des Fluges atmosphärischen Bedingungen ausgesetzt, die Eisbildung auf Tragflächen und anderen Oberflächen der Flugzeugstruktur, wie Höhen- und Seitenleitwerken, begünstigen. Werden diese Eisschichten nicht entfernt, können sie das Flugzeug übermäßig belasten und die Tragflächenkonfiguration verändern. Im schlimmsten Fall führt dies zum Absturz des Flugzeuges [8]. Die Vereisung verändert die Form und die Oberflächeneigenschaften von Flugzeugkomponenten. Das hat zur Folge, dass der Luftstrom, der über die vereisten Komponenten fließt, veränderte aerodynamischen Kräfte und Momente erzeugt. Die typischen Auswirkungen der Vereisung sind ein verringerter Anstellwinkel und ein verringerter maximaler Auftrieb. Außerdem wird der Luftwiderstand erhöht, was wiederum zu einem erheblich höheren Kraftstoffverbrauch führt. Zusätzlich kann die Vereisung auch die Wirksamkeit der Steueroberfläche, die Gelenkmomente und die Dämpfung beeinflussen. So führen diese Effekte zu einer veränderten Stabilität und Kontrolle des Flugzeugs [9]. Aus diesem Grund stehen Enteisungssysteme zur Verfügung (engl. De-Icing systems).

Das am weitesten verbreitete De-Icing System in kommerziellen Turbofan-Flugzeugen ist das Zapfluftsystem. Bei diesem System wird komprimierte und heiße Luft aus

dem Kompressorabschnitt des Triebwerkes abgezapft und durch ein Rohr in die Vorderkantenkammer (38) des Tragflügels oder in einen anderen zu enteisenden Teil der Flugzeugstruktur geleitet. Von da aus strömt es gegen die Innenfläche der Struktur und wärmt diese auf (siehe Abbildung 1.1).

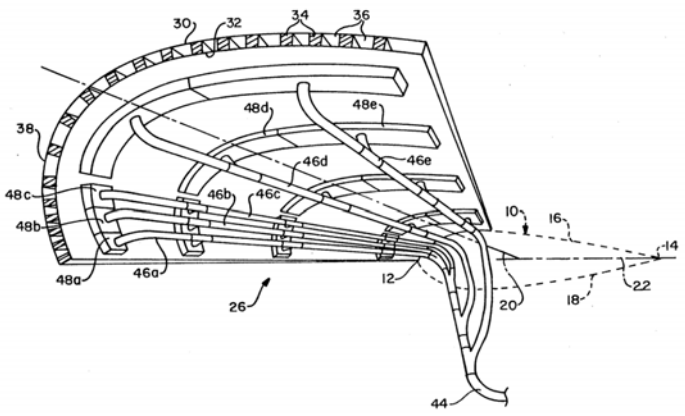


Abbildung 1.1.: Querschnitt einer Tragflächenstruktur mit De-Icing System [2]

Bei diesem Enteisungssystem für Zapfluft handelt es sich im Allgemeinen um ein System mit den Betriebszuständen «Ein» und «Aus» ohne jegliche Einstellungsmöglichkeit. Das Auslegungskriterium des Systems ist dabei ein maximaler Vereisungsschutz beim Idle Descent. Dieser Begriff beschreibt einen Sinkflug mit minimalem Treibstoffverbrauch, der erreicht wird, indem die Triebwerke auf Leerlauf reduziert werden. Da die Zapflufttemperatur und der Zapfluftdruck bei dieser Leistungseinstellung sehr niedrig sind, muss der Massenstrom groß sein. Daher ist das System bei weniger anspruchsvollen atmosphärischen Bedingungen überdimensioniert und verschwenderisch. Die komprimierte Luft wird nach dem Aufheizen der Innenseite der Flügelvorderkante aus dem Flügel in die Umgebung abgeleitet. Der Wirkungsgrad liegt dabei unter 50% [2].

Ein weiterer und folgenschwerer Nachteil des Zapfluftsystems ist jedoch das zusätzliche Gewicht. Sollte während eines Fluges keine Enteisung notwendig sein, ist das System trotzdem an Bord und erhöht so permanent die Gesamtmasse des Flugzeugs. Die Folge ist ein zusätzlicher Kraftstoffverbrauch.

Der Entwurf von multifunktionalen, faserverstärkten Kunststoffbauteilen, die Aufgaben der Lastübertragung und der elektrischen Wärmeerzeugung im Bereich des De-Icing übernehmen können, ist Gegenstand aktueller Forschung [10]. Es wird die Idee verfolgt, Kohlefasern in faserverstärkten Strukturen zusätzlich als Heizfasern zu verwenden. Die Verwendung dieses alternativen Verfahrens an einer Flügelvorderkante ersetzt das bestehende Zapfluftsystem und kann so zu einer Gewichtsreduktion der Flugzeugmasse führen. Durch Anschluss an eine externe Stromquelle entsteht Joulsche Wärme, die das gesamte Bauteil aufheizt und somit potentiell die entsprechende Strukturoberfläche von Eis befreit. Mithilfe einer steuerbaren Stromquelle ist dieser Prozess außerdem hinsichtlich Heizrate und Zieltemperatur adaptiv regelbar.

Zu diesem Zweck müssen jedoch vorerst Kenntnislücken gefüllt werden. Zum einen muss das Heizverhalten bezüglich der Wärmeausbreitung an der Oberfläche der Struktur untersucht werden. Dies kann Aufschluss über die notwendige Dichte der Heizfasern geben. Zum anderen muss die Regulierbarkeit der Oberflächentemperatur analysiert werden.

Ziel dieser Bachelorarbeit ist es, eine adaptive Heizregelung zu entwickeln, aufzubauen und zu testen, mit der das Gesamtsystemverhalten hinsichtlich eines Heizvorganges experimentell untersucht werden kann.

Zur Verifikation kann auf mehrere Versuchspaneele zurückgegriffen werden. Diese stehen in zwei verschiedenen Bauarten zur Verfügung: Ein Versuchspaneel aus kohlefaserverstärktem Kunststoff und ein weiteres aus glasfaserverstärktem Kunststoff. Beide sind mit gesondert eingebrachten Kohlefaserrovings ausgestattet. So können Mess- und Steuerungsverfahren sowie das Heizverhalten des jeweiligen Versuchspaneels experimentell getestet und verifiziert werden.

Im folgenden Kapitel wird zunächst ein Überblick über die Methodik und die theoretischen Funktionalitäten der verwendeten Komponenten gegeben. In weiteren Kapiteln wird ferner der Aufbau des entwickelten Systems beschrieben. Dafür wird der Ansatz verfolgt, das physikalische System "Paneel"durch eine Aktorik anzuregen, hier Bestromung der Rovings, und die Reaktion des Paneels durch eine geeignete Sensorik zu erfassen. Anhand der Daten der Sensorik können durch einen Controller Anpassungen in der Aktorik vorgenommen werden.

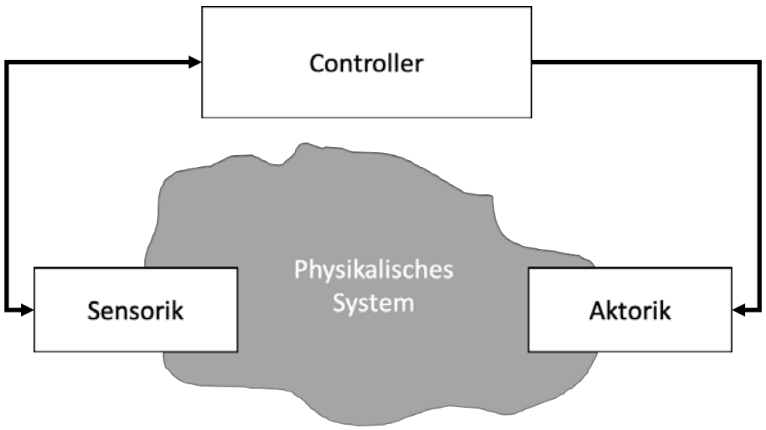


Abbildung 1.2.: Generelles Vorgehen für die Untersuchung eines physikalischen Systems

Nach Abschluss des Systementwurfs wird das System getestet und anschließend verifiziert. In einer abschließenden Zusammenfassung werden die Ergebnisse diskutiert und ein Ausblick darüber gegeben, mit welchen Schritten die Ergebnisse dieser Arbeit weiterentwickelt werden könnte.

2. Verwendete Hardware, Kommunikation und Methoden

Das folgende Kaptitel gibt einen Überblick über die verwendete Hardware, Kommunikationsschnittstellen und Methoden. Dabei wird ein besonderes Augenmerk darauf gelegt, theoretische Hintergrundinformationen und Kenndaten zu verschiedenen angewandten Technologien zu geben, die bei der Beschreibung des Systementwurfs aufgegriffen werden.

2.1. Systemneuentwicklung nach VDI 2206

Als Richtlinie zur Neuentwicklung des Systems dient das V-Modell nach VDI 2206 für mechatronische Systeme [3]. Das V-Modell regelt detailliert den Ablauf eines Neuentwicklungsprozesses (siehe Abbildung 2.1). Dabei lässt sich das Vorgehen nach dem V-Modell sowohl bei umfangreichen Projekten als auch bei kleineren Aufgabenstellungen anwenden. Durch eine systematische Durchführung der vorgegeben Schritte werden Projekte besser plan- und nachvollziehbar. Ferner spiegelt sich dies in der Qualität des Endproduktes wider.

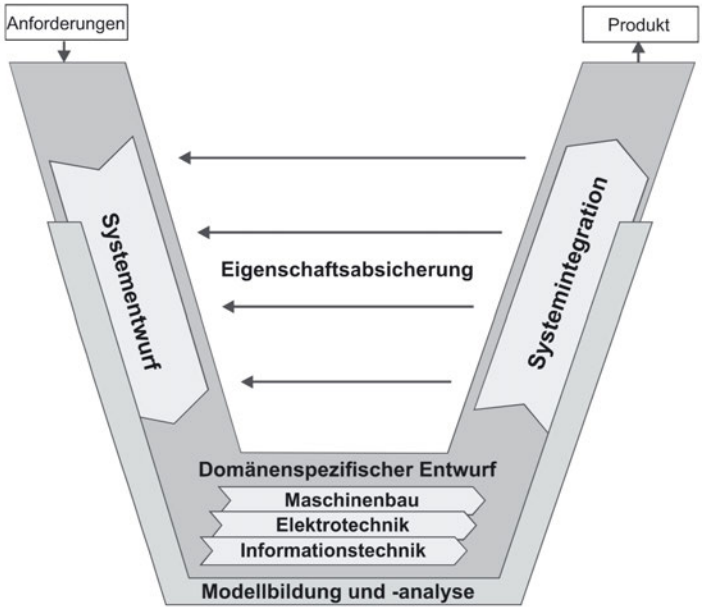


Abbildung 2.1.: V-Modell nach VDI 2206 [3]

Am Anfang steht die Systemanforderungsanalyse. Alle Anforderungen an das System und den Prozess werden herausgearbeitet und fixiert. Die Sammlung der Anforderungen dient als Lastenheft und legt die Funktionen des Systems fest. Außerdem stellt das Lastenheft zugleich den Maßstab dar, nach dem das spätere Produkt zu bewerten ist. Nach Festlegung der Systemanforderungen gilt es, einen Systementwurf zu erstellen, der diese Anforderungen erfüllt. Für den Entwurfsprozess wird das System in kleinere Einzelsysteme unterteilt und anschließend in domänenspezifische Entwicklungsprozesse überführt. Zum Schluss erfolgt die Systemintegration, bei der das System an den definierten Schnittstellen zusammengefügt wird. Abschließen wird das System getestet und das Erfüllen der Anforderungen anhand des Lastenheftes überprüft.

2.2. Anforderungen an den Systemaufbau

An die Heizregelung und die damit verbundene Aufnahme von Temperaturen auf der Paneeloberfläche sind bestimmte Anforderungen gestellt. Die Anforderungen sind dabei gezielt auf die Charakterisierung und Messung des Heizverhaltens des Paneels abgestimmt. Zum Zweck der Abstufung in Wichtigkeiten wird jede Anforderung in die Kategorie hart

oder weich gruppiert. Von einer harten Anforderung spricht man, wenn sie zwingend für die Systemfunktion erforderlich ist. Eine weiche Anforderung hingegen drückt eine Optionatlität aus. Wird sie nicht umgesetzt, stellt dies keinen Nachteil in der Funktionalität des System dar. Die Anforderungen werden zunächst in Tabellenform aufgeführt und anschließend ausführlich beschrieben.

Nummer	Bezeichnung	Quantifizierung	Klassifizierung
1	Erhitzen auf Zieltemperatur	-	Hart
2	Adaptive Heizrate für schnelle Erhitzung	-	Hart
3	Maximale Rovingtemperatur	$T_{F_{max}} = 100\,^{\circ}\mathrm{C}$	Hart
4	Benutzereingaben aufnehmen	-	Hart
5	Flächenabdeckende Temperaturmessung	-	Hart
6	Speichern der Messgrößen	-	Hart
7	Oberflächentemperaturaufnahme an drei Positionen	-	Hart
8	Modularer Aufbau	-	Weich
9	Toleranz der Temperaturmessung	$\pm 2\,\%$	Hart

Tabelle 2.1.: Anforderungen an den Systemaufbau

1. Das Paneel soll auf eine eingestellte Zieltemperatur erhitzt und anschließend auf dieser Temperatur gehalten werden

2. Die Heizrate soll dabei adaptiv ausgelegt sein, sodass die Zieltemperatur möglichst schnell erreicht wird

3. Um den Systemtest am Paneel möglichst realistisch abzubilden, wird eine maximale Kohelfaserrovingtemperatur von $T_{F_{max}} = 100\,^{\circ}\mathrm{C}$ vorgesehen. Dies entspricht der typischen Temperaturobergrenze der Betriebstemperatur der Rovings in Flugzeugen, die sich von $215\,\mathrm{K} - 373\,\mathrm{K}$ erstreckt [11].

4. Das Regelungsprogramm soll nach dem Einschalten Benutzereingaben aufnehmen.

Dabei soll eine Einstellung der Abtastrate und der Zieltemperatur möglich sein. Falsche Eingaben müssen zuverlässig abgefangen werden

5. Zur Charakterisierung des Paneels soll das Temperaturfeld auf seiner Oberfläche an diskreten Punkten aufgezeichnet werden. Es ist daher nötig, sich bei der Temperaturaufnahme nicht nur auf den Bereich unmittelbar über einem Kohlefaserroving zu beschränken, sondern zusätzlich und gerade in den Bereichen zwischen Kohelfaserrovings zu messen

6. Die gemessenen Temperaturen der Oberfläche, die Temperaturen der Rovings und ein Strom-Zeit-Verlauf sollen zum Zweck der Auswertung gespeichert werden

7. Um beurteilen zu können, ob die Messwerte aussagekräftig sind, soll die Temperaturmessung und Aufnahme in gleicher Konfiguration an drei Positionen längs der Rovings erfolgen

8. Das Messsystem soll modular aufgebaut und um weitere Temperatursensoren erweiterbar sein. So kann im Nachhinein die vom Messaufbau erfasste Fläche vergrößert werden

9. Für spätere Auswertungen ist die Qualität der Temperaturmessergebnisse von besonderer Bedeutung. Das Toleranzband der Messkette, vom Sensor bis zum Auslesen am Mikrocontroller, soll bei maximal $\pm 2\,\%$ der maximal im Systemaufbau zu erwartenden Temperatur liegen. Einen Nachweis zur Einhaltung dieser Toleranz ist außerdem zu erbringen

Der Systementwurf basiert auf den beschriebenen Anforderungen. Bevor der Systementwurf vorgestellt wird, werden nachfolgend die Komponenten zur Systembildung beschrieben.

2.3. Geplanter Systementwurf

Für eine übersichtliche Darstellung der Verknüpfungen zwischen den Sensoren, dem Mirko-Controller, den Aktoren und der Umwelt wird eine Funktionsstruktur mit unterschiedlichen Flussgrößen erstellt. Dabei werden grundsätzlich drei Arten von Flüssen unterschieden [12].

- Stoffflüsse: Beispiele für Stoffe, die zwischen Einheiten mechatronischer Systeme fließen, sind feste Körper, Prüfgegenstände, Gase oder Flüssigkeiten.
- Energieflüsse: Unter Energie ist in diesem Zusammenhang elektrische oder thermische Energie zu verstehen, jedoch können auch Größen wie Kraft oder Druck gemeint sein.
- Informationsflüsse: Informationen, die zwischen Einheiten eines mechatronischen Systems ausgetauscht werden, sind zum Beispiel Messgrößen, Steuerimpulse oder Daten.

Zunächst wird das Gesamtsystem als Kasten oder als Blackbox dargestellt (siehe Abbildung 2.2). Das bedeutet, dass detaillierte Funktionen vorerst ausgeblendet und lediglich die Eingangsflüsse und die Ausgangsflüsse betrachtet werden. Das Gesamtsystem wird mit einem Eingangsfluss von Energie, in Form von elektrischer Energie, und mit einem Informationsfluss, den Prozessparametern vom Bediener, gespeist. Das System wandelt diese Flüsse in die zwei Ausgangsflüsse Joulesche Wärme im Paneel und Prozessdaten, die gespeichert werden, um.

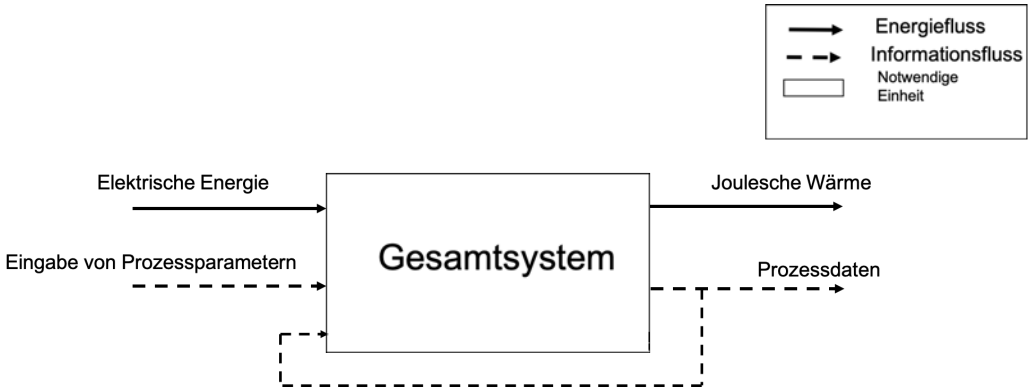


Abbildung 2.2.: Funktionsstruktur des Gesamtsystems

Im späteren Entwurfsprozess in Kapitel 3 wird die Blackbox in seine Unterbaugruppen aufgeschlüsselt und die Flüsse zwischen ihnen dargestellt.

Zu Erfüllung der Anforderungen wird ein vom Mikro-Controller generiertes PWM-Signal mit Hilfe einer MOSFET-Schaltung verstärkt. Das so erzeugte Leistungs-PWM Signal erzeugt einen Strom, der durch eine geeignete Schaltung durch Heizrovings geleitet wird, wodurch Joulesche Wärme entsteht und die Oberfläche erhitzt wird. Die Oberflächentemperatur des Paneels, die Rovingtemperatur und der Heizstrom werden zeitaktuell mit einer vorgegebenen

Rate abgetastet und gespeichert. Anhand der Auswertung der Temperaturen wird der Strom und damit indirekt die Temperatur durch die Anpassung des Tastgrades des PWM-Signals geregelt.

Auf Basis dieses Grundkonzeptes werden nachfolgend mögliche Systemkomponenten vorgestellt und verglichen, um eine Grundlage für den Entwurfsprozess zu bilden.

2.4. Aufbau und Versuchspaneele

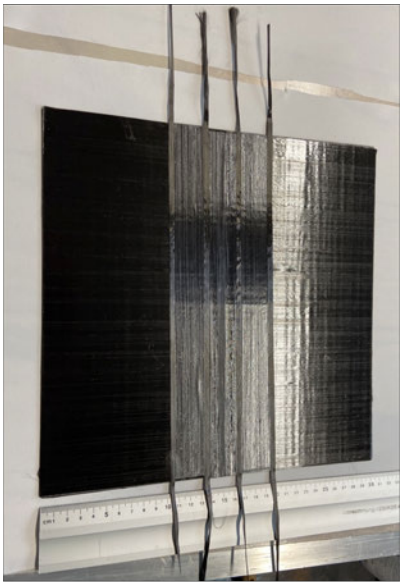
Es werden zwei Paneele aus unterschiedlichen Faserverbundmaterialien bereitgestellt. Bei der ersten der beiden Bauweisen der bereitgestellten Leichtbaupaneele handelt es sich um ein geschichtetes Laminat aus vorimprägnierten Kohlefasermatten (englisch: Prepreg). Verwendet wird CM-Preg T-C-230/600 CP004 39 der Firma c-m-p. Wichtige Materialkennwerte können Tabelle 2.2 entnommen werden.

Kennwert	Abkürzung	Wert
Schichtdicke	h_{sp1}	$0{,}25\,\mathrm{mm}$
Massenbelegung	m_{sp1}	$230\,\frac{\mathrm{g}}{\mathrm{m}^2}$
Maximale Temperaturbelastbarkeit	$T_{max_{P1}}$	$372\,^{\circ}\mathrm{C}$ [13]

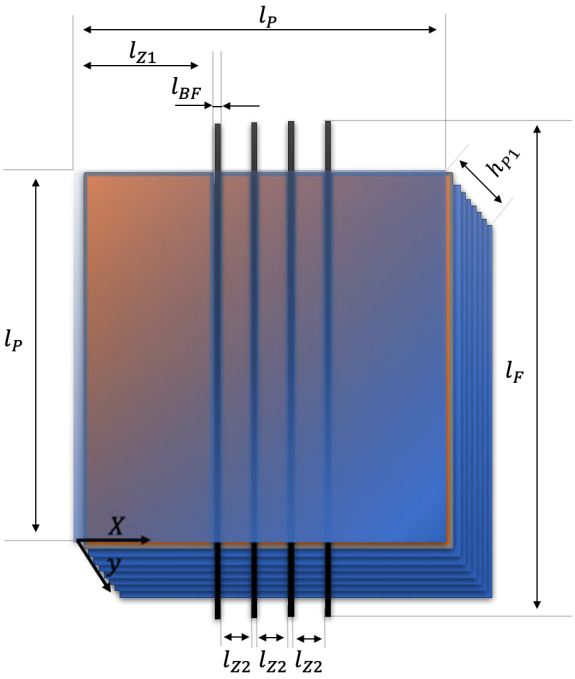
Tabelle 2.2.: Wesentliche Materialkennwerte CM-Preg T-C-230/600 CP004 39 [Quelle: Datenblatt im Anhang A2.1.]

Das Paneel ist in neun Schichten aufgebaut, die symmetrisch nach einem $[90°,0°]_s$ -Vorgehen geschichtet werden. Dabei werden die Fasern der ersten Schicht in $90°$-Richtung ausgerichtet. Die Fasern der zweiten Prepreg-Schicht werden orthogonal zu denen der ersten Schicht ausgerichtet. Sie zeigen demnach in $0°$-Richtung. Dieses Schema wird bis inklusive der siebten Schicht fortgeführt. Anschließend folgt eine achte modifizierte Laminatschicht. Sie beinhaltet vier Kohelfaserrovings, die später zur Beheizung dienen. Zwischen den Rovings wird jeweils ein Abschnitt des Prepegmaterials aufgebracht, um die achte Schicht zu komplettieren. Sie beginnt entlang der x-Achse (vergleiche Abbildung 2.3b) mit einem Prepreg-Zuschnitt mit der Breite l_{Z1}, gefolgt von dem ersten von vier Rovings mit der Länge l_F. Es folgen im Wechsel ein Prepreg-Zuschnitt der Breite l_{Z2} und einem Rovings. Nach dem vierten Rovings wird die achte Laminatschicht mit einem weiteren Prepreg-Zuschnitt mit der

Breite l_{Z1} vollendet. Abbildung 2.3a zeigt das Laminat nach Aufbringen der modifizierten Schicht.



(a) Unausgehärtetes Paneel mit aufgebrachter achter Schicht



(b) Schematischer Paneelaufbau

Abbildung 2.3.: Das Versuchspaneel

Die Maße aus Abbildung 2.3b und damit auch die der Paneelkomponenten können der folgenden Tabelle 2.3 entnommen werden:

Komponente	Abkürzung	Maß in $[\mathrm{mm}]$
Kantenlänge Paneel	l_P	300
Länge der Rovings	l_F	400
Breite der Rovings	l_{BF}	5
Breite des ersten Zuschnittes	l_{Z1}	102,5
Breite des ersten Zuschnittes	l_{Z2}	25
Höhe des Prepreg-Paneels	h_{P1}	2,25

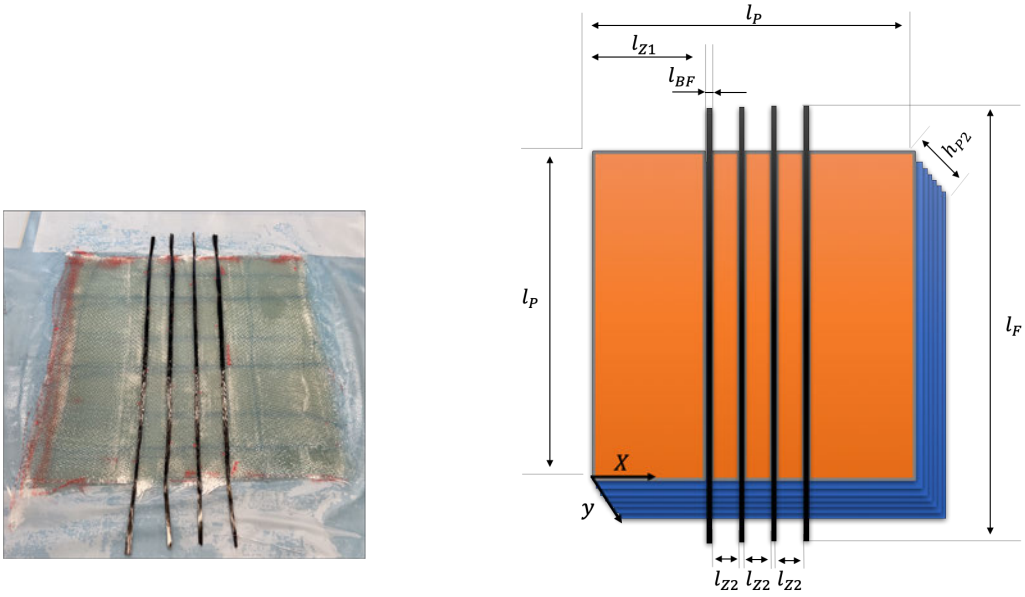
Tabelle 2.3.: Maße der Paneelbestandteile mit spezifischer Prepreg-Paneel Höhe h_{P1}

Bei der zweiten Bauweise wird das Prepreg durch Glasfaserzuschnitte ersetzt. Die Kennwerte der Glasfasermatten kann der Tabelle 2.4 entommen werden.

Kennwert	Abkürzung	Wert
Schichtdicke	h_{sp2}	$0{,}26\,\mathrm{mm}$
Massenbelegung	m_{sp2}	$296\,\frac{\mathrm{g}}{\mathrm{m}^2}$
Maximale Temperaturbelastbarkeit	$T_{max_{P2}}$	$600\,^{\circ}\mathrm{C}$

Tabelle 2.4.: Wesentliche Materialkennwerte des Glasfilamentgewebes der Firma interglas [Quelle: Datenblatt im Anhang A2.2.]

Der Schichtaufbau des Glasfaserpaneels wird im Vergleich zum Prepreg-Paneel geringfügig abgeändert. Es besteht anstatt der neun Schichten lediglich aus sieben Glasfaserschichten. Die siebte und oberste Schicht ist um die Rovings erweitert. Sie liegen somit direkt an der Oberfläche des Paneels, eine Deckschicht ist nicht vorhanden. Die Maße des Paneels und der Zuschnitte bleiben bis auf die Höhe nach Tabelle 2.3 gleich. Die Höhe des Glasfaser-Paneels beträgt $h_{P2} = 2\,\mathrm{mm}$. Die folgenden Abbildungen 2.4a und 2.4b zeigen den Aufbau des Glasfaser-Paneels.



(a) Unausgehärtetes Glasfaser-Paneel

(b) Schematischer Paneelaufbau

Abbildung 2.4.: Das Glasfaser-Versuchspaneel

Die ersten sechs qudratischen Glasfasermatten werden in der selber Hauptrichtung geschichtet und mit Epoxydharz verbunden. Bei der siebten und letzten Schicht werden die Zuschnitte und Kohelfaserrovings wie bei der achten Schicht des Prepreg-Paneel angeordnet. Nach dem Aushärten des Harzes ist das Paneel fertig.

Bei den beschriebenen Kohlefaserrovings handelt es sich um das Filamentgarn IMS65 der Firma Toho Tenax, , welches mit einer Schlichte vom Typ E23 versehen ist. Der Roving setzt sich aus einer Anzahl von n_f Fasern zusammen. Diese und andere Kennwerte des Garns können der Tabelle 2.5 entnommen werden.

Beschreibung	Abkürzung	Wert
Anzahl der Fasern im Roving	n_f	24.000
Faserdurchmesser	d_f	$5\,\mu\mathrm{m}$
Spezifischer elektrischer Widerstand einer Faser	ρ_f	$1{,}45\,\Omega\mathrm{m}$
Maximale Temperaturbelastbarkeit	T_{max_F}	$> 700\,^{\circ}\mathrm{C}$ [14]

Tabelle 2.5.: Wesentliche Materialkennwerte Tenax IMS65 [Quelle: Datenblatt im Anhang A2.3.]

Mit diesen Materialkennwerten ist es möglich, den elektrischen Widerstand einer einzelnen Faser zu berechnen. Dazu benötigt man sowohl den Querschnitt einer Faser a_f, der sich aus dem Durchmesser d_f einer Faser berechnen lässt, als auch den spezifischen elektrischen Widerstand einer Faser ρ_f. Zusammen mit der Länge einer Faser l_f, die gleich der Länge eines Rovings ist, $l_F = l_f$, und die man Tabelle 2.3 entnimmt, kann die Gleichung 2.2 angewandt werden [15].

$$R_f = \frac{\rho_f \cdot l_f}{a_f}. \tag{2.1}$$

Will man nun den Gesamtwiderstand des Kohlefaserrovings errechnen, muss man beachten, dass dieser aus einer Anzahl von n_f Fasern besteht. Ein Roving lässt sich also mit einer Widerstandsanordnung von n_f parallel geschalteten Widerständen mit dem Widerstandswert R_f modellieren, zu sehen in Abbildung 2.5.

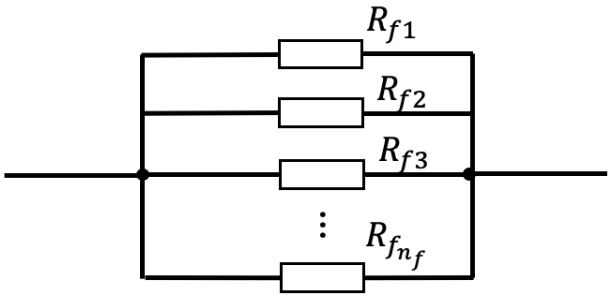


Abbildung 2.5.: Schaltungsmodell der Fasern innerhalb eines Rovings

Anschließend kann der Gesamtwiderstand berechnet werden [15]:

$$\frac{1}{R_F} = \sum_{k=1}^{n_f} \frac{1}{R_{f_k}} \tag{2.2}$$

Die Joulescher Wärme wird durch den die Kohelfaserrovings durchfließenden Strom I_{F1},I_{F2},I_{F3} und I_{F4} erzeugt. Die Kohelfaserrovings sind dabei als Widerstände mit Widerstandswerten R_{F1},R_{F2},R_{F3} und R_{F4} zu betrachten. Diese sind zueinander parallel geschaltet, somit liegt über jedem Roving die gleiche angelegte Versorgungsspannung $\hat{\mathrm{u}}_\mathrm{P}$ an [15].

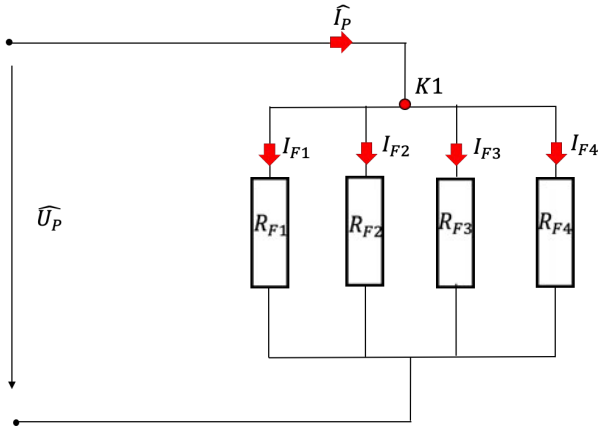


Abbildung 2.6.: Schaltplan des Paneels mit auftretenden Spannungen und Strömen

Die Ströme I_{F1},I_{F2},I_{F3} und I_{F4} lassen sich nach dem Ohmschen Gesetz [15] berechnen. Dafür werden die bekannten Größen der Widerstandswerte und der Versorgungsspannung

genutzt:

$$\hat{I}_{Fi} = \frac{U}{R_{Fi}}. \tag{2.3}$$

Anschließend kann anhand des 1. Kirchhoff'schen Gesetzes, der Knotenregel [15], der Gesamtstrom, der durch das Paneel fließt, berechnet werden. Dafür wird der Knoten K1 (siehe Abbildung 2.6) vor der Parallelschaltung der Widerstände der Rovings betrachtet. Knotenpunkte sind Punkte, an denen sich der Stromkreis verzweigt. In diesen Knotenpunkten muss die Summe der zufließenden Ströme gleich der Summe der abfließenden Ströme sein. Somit gilt :

$$K1 : \hat{I}_P = \sum_{k=1}^{4} I_{Fk}. \tag{2.4}$$

2.5. Verwendete Recheneinheit

Als Grundlage der zu entwickelnden Regelung dient ein ARDUINO MEGA 2560. Der auf einem ATmega2560-Prozessor aufbauende Mikrocontroller verfügt über 54 digitale Ein- und Ausgangspins, von denen 15 für PWM-Ausgangssignale genutzt werden können. Das PWM-Signal (siehe Abschnitt 2.7) hat dabei eine Auflösung von 2 Byte. Zusätzlich weist der ARDUINO MEGA 2560 16 analoge Eingänge mit einer 10 Bit Auflösung auf. Für die serielle Kommunikation stehen eine SPI Schnittelle (siehe Abschnitt 2.10) und vier Serialschnittstellen zur Verfügung, die 1-Wire (siehe Abschnitt 2.9) kompatibel sind [16]. Der Controller benötigt eine Versorgungsspannung von $9\,\mathrm{V}$. So ist laut Datenblatt ein Pegel von $5\,\mathrm{V}$ an den Ausgangspins gewährleistet. Der $5\,\mathrm{V}$-Ausgangspin ist für $\approx 400\,\mathrm{mA}$ an USB-Versorgung und für $\approx 900\,\mathrm{mA}$ bei Verwendung eines externen Netzteils geeignet. Als Programmieroberfläche dient die Arduino Software der aktuellsten Version 1.8.10. Der Arduino wird sowohl für die Aufnahme und Auswertung der Temperaturen und die Aufnahme des Heizstromes, als auch für die Spannungsregelung verwendet. Zunächst wird auf die Strommessung des Heizstromes eingegangen.

2.6. Messung des Heizstromes

Die Aufnahme des Heizstromes erfolgt durch analoge Messung durch einen ACS712-Chip. Der Strommesser besteht aus linearen Hall-Sensor. Der zu messende Strom fließt dabei durch einen Kupferwiderstand von $1{,}2\,\mathrm{m\Omega}$ innerhalb des Sensors. Das so durch den Stromfluss erzeugte Magnetfeld wird durch einen Hall-Sensor in eine zum Strom proportionale messbare Ausgangsspannung mit einer Schrittweite von $100\,\frac{\mathrm{mV}}{\mathrm{A}}$, gewandelt. Bei einem Strom von $0\,\mathrm{A}$ liegt die Ausgangsspannung bei einem Offset von $2{,}5\,\mathrm{V}$. Der Messfehler der Strommessung liegt bei $1{,}5\%$ [17].

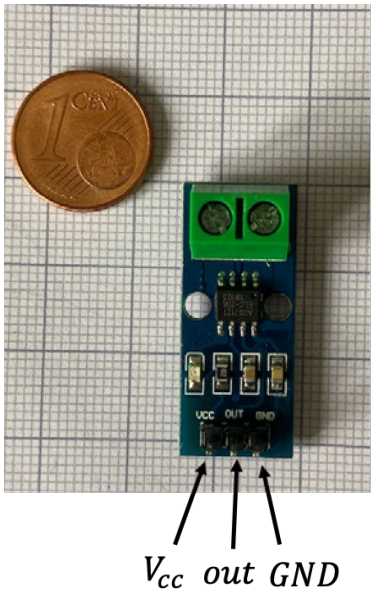


Abbildung 2.7.: Der Stromsensor ACS712 im Größenvergleich

2.7. Theorie zur Pulsweitenmodulation

Für die Spannungsregulierung der über die Kohelfaserrovings abfallende Spannung U_P findet ein pulsweitenmoduliertes Signal (abgekürzt PWM-Signal) Verwendung. Bei einem PWM-Signal wird bei fester Frequenz f das Verhältnis zwischen Einschaltzeit und Periodendauer variiert. Dabei spricht man bei dem Verhältnis von Einschaltzeit $\tilde{T}_{on}$ zu der Periodendauer $\tilde{T} = \frac{1}{f}$ von dem Tastgrad p, der sich anhand der Gleichung 2.5 berechnen lässt [18]:

$$p = \frac{\tilde{T}_{on}}{\tilde{T}}. \tag{2.5}$$

Abbildung 2.8 zeigt ein beispielhaftes PWM-Signal mit eingetragenen spezifischen Parametern. Das Signal $u(t)$ beginnt zum Zeitpunkt $t = 0\,\mathrm{s}$ bei seinem Spitzenwert von $\hat{u}$.

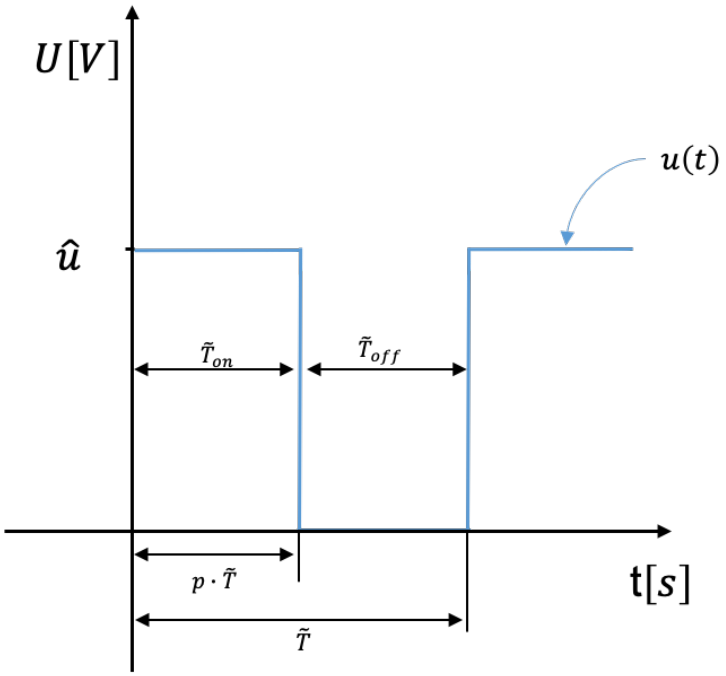


Abbildung 2.8.: PWM-Signal, dargestellt mit signalspezifischen Parametern

Nach Ablauf der Zeitdauer $\tilde{T}_{on} = p \cdot T$ fällt $u(t)$ auf einen Wert von $0\,\mathrm{V}$. Der Pegel wird bis zum Ablauf der Dauer $\tilde{T}_{off}$ beibehalten. Anschließend springt das Signal wieder zurück auf seinen Maximalwert von $\hat{u}$ und eine neue Periode beginnt. Der Tastgrades hat dabei einen gültigen Wertebereich von $0 \leq p \leq 1$. Das Signal hat die Form einer Mischspannung. Eine Mischspannung besteht immer aus einem Gleich- und einem Wechselanteil, wobei der Gleichanteil den Pegel darstellt, um den der Wechselanteil oszilliert [19]. Durch diese Eigenschaft lässt sich auch mit einer Mischspannung an Gleichstromverbrauchern eine Leistung umsetzen. Die Spannung, die über jenen Verbraucher abfällt, lässt sich bestimmen, indem man den Gleichwert des PWM-Signals errechnet. Bestimmt man den Gleichwert in Abhängigkeit des Tastgrades p, so ergibt sich nach [18]:

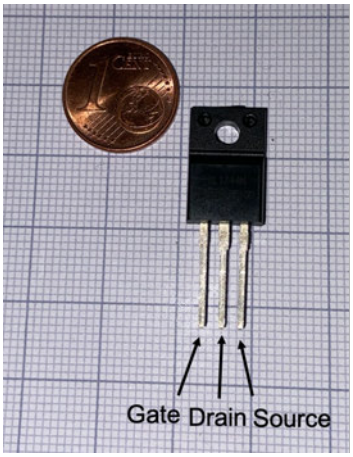
$$\bar{u} = \frac{1}{\tilde{T}} \int_0^{\tilde{T}} u(t)dt = \frac{1}{\tilde{T}} \cdot \left(\int_0^{p \cdot \tilde{T}} \hat{u}dt + \int_{p \cdot \tilde{T}}^{\tilde{T}} 0\mathrm{V}dt \right) = \hat{u} \cdot p \tag{2.6}$$

Wäre kein Gleichanteil vorhanden, würde der Gleichwert $\bar{u} = 0\,\mathrm{V}$. In der Theorie lassen sich auf diese Weise linear Gleichwerte von $0 \leq \bar{u} \leq hatu$ einstellen. In der Systementwicklung kann diese Art des Signals verwendet werden, um die am Paneel anliegende Spannung, die den Heizstrom fließen lässt, zu regeln. Dafür wird jedoch ein Leistungs-PWM-Signal benötigt, welches die Heizaufgabe erfüllen kann. In diesem Fall wird auf Halbleitertechnologien zurückgegriffen.

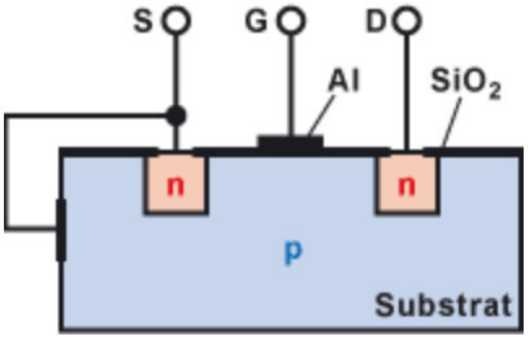
2.8. Verwendeter MOSFET und Halbleitertheorie

Für die Generierung eines Leistungs-PWM Signals, das genug Leistung aufbringt, um das Paneel zu erhitzen, kommt eine MOSFET-Schaltung zum Einsatz. Diese soll die Aufgabe übernehmen, das Steuer-PWM Signal des Arduinos zu verstärken. Auf ähnliche Wiese wurde bei einer Forschung zur Entwicklung eines Flügel-Temperaturkontrollsystem auf der Basis von Kohlenstoff-Nanomaterialien zum Anti-Icing und De-Icing unbemannter Luftfahrzeuge im Flug vorgegangen [20].

MOSFET steht für Metall-Oxid-Halbleiter-Feldeffekttransistor. Bei diesem aktiven Bauteil handelt es sich um einen steuerbaren Widerstand mit drei Anschlüssen: Source, Drain und Gate.



(a) MOSFET IRFZ44N im Größenvergleich



(b) Substratschichten innerhalb eines MOSFET [5]

Abbildung 2.9.: Der verwendete MOSFET

Trotz verschiedener Typen von MOSFETs wird hier nur auf den Anreicherungstyp eingegangen. Dem Anreicherungstyp-Feldeffekttransistor wird ein p-dotierter Halbleitersiliziumkristall zu Grunde gelegt, das sogenannte Substrat. In das Substrat werden zwei n-dotierte Inseln eingebracht. Anschließend wird der Kristall mit einer isolierenden Siliziumdioxid-Schicht abgedeckt. Das Aufdampfen einer Gate-Elektrode vollendet den Aufbau [5] (Abbildung 2.9b).

Bei einer n-Dotierung wird ein 5-wertiges Dotierelement zwischen die Siliziumatome des Halbleiterkristalls eingebracht, z.B. Phosphor. Dieses Dotierelement besitzt ein Außenelektron mehr als die Siliziumatome. Vier Außenelektronen des Dotierelementes können sich mit je einem Siliciumatom verbinden, das fünfte Elektron ist frei beweglich. Man bezeichnet diese Dotierelemente deshalb als Elektronendonator. Die freien Elektronen dienen nun als Ladungsträger. Die Dotierelemente sind durch Abgabe der Ladungsträger fest im Gitter eingebaut, es bewegen sich nur die negativen freien Elektronen. Neben vielen freien Elektronen sind jedoch auch einige Löcher vorhanden [4].

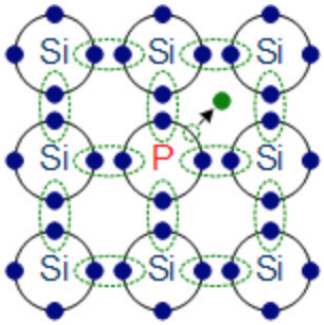


Abbildung 2.10.: N-dotierter Siliziumkristall [4]

Bei einer p-Dotierung passiert genau das Gegenteil. Hier wird ein 3-wertiges Dotierelement zwischen die Siliziumatome des Halbleiterkristalls eingebracht, z.B. Bor. Dieses besitzen ein Außenelektron zu wenig, um vier Siliziumatome binden zu können. So entsteht auf der äußersten Schicht ein Loch, das ein freies Elektron binden kann. Diese positiven Löcher dienen als Ladungsträger. Man nennt diese Dotieratome Elektronenakzeptor. Neben vielen Löchern sind jedoch auch einige freie Elektronen vorhanden [4].

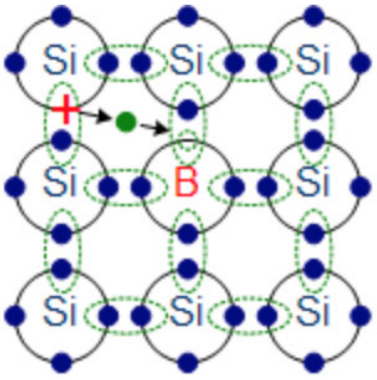


Abbildung 2.11.: P-dotierter Siliziumkristall [4]

Im Normalzustand ist ein Stromfluss von Source nach Drain nicht möglich. An einem Übergang von n-dotiertem und p-dotiertem Material gleichen sich die Landungsträger aus. Die freien Elektronen des n-dotierten Substrates füllen die Löcher des p-dotiertem Substrates auf. So entsteht eine isolierende Zone, die keinen Stromfluss zulässt, weil keine freien Ladungsträger vorhanden sind. Der MOSFET ist also selbstsperrend. Um eine Leitfähigkeit zwischen Drain und Source herzustellen, wird eine positive Spannung U_{GS} zwischen Gate

und Source angelegt. Dadurch entsteht in dem p-dotiertem Substrat ein elektrisches Feld. Die wenigen freien Elektronen werden vom positiven Gate angezogen und sammeln sich unter der Isolationsschicht, verdeutlicht durch die roten Pfeile in Abbildung 2.12. Die positiv geladenen Löcher werden aus diesem Bereich verdrängt. Ab einer bestimmten Schwellspannung U_{th} (engl. threshold voltage) ist die Verdrängung der positiven Löcher so groß, dass sich ein n-leitender Kanal bildet, der Source und Drain verbindet [5]. Diese Schwellenspannung U_{th} liegt bei dem verwendeten MOSFET IRFZ44N laut Datenblatt bei $2\,\mathrm{V} - 4\,\mathrm{V}$ [21].

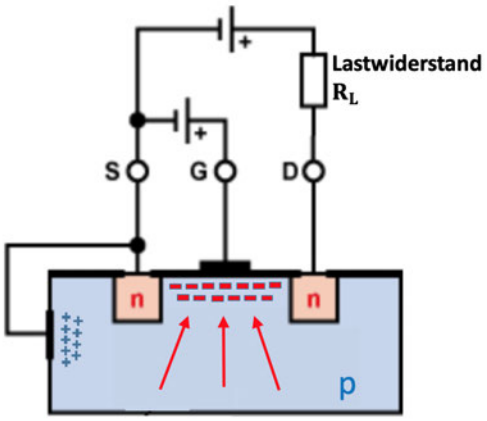


Abbildung 2.12.: Ausgebildeter n-Kanal durch Anschluss einer Gate-Source-Spannung U_{GS} [5]

Die Leitfähigkeit dieses gebildeten Leitungskanals lässt sich durch die Gatespannung U_{GS} steuern. Beispielweise ist bei einer Spannung von $U_{GS} = 5\,\mathrm{V}$ ein Stromfluss von bis zu $I_D = 22\,\mathrm{A}$ möglich [21]. Wird die positive Gatespannung ab der Schwellspannung zusätzlich erhöht, führt dies zu einer Steigerung der Feldstärke, was wiederum die Anreicherung des Kanals mit Elektronen fördert. Der Kanal wird leitfähiger,der Widerstand sinkt. Die Verringerung der positiven Gatespannung sorgt hingegen dafür, dass der Kanal weniger leitfähig wird. Zur Steuerung wird nur die Gatespannung U_{GS} benötigt. Die Steuerung des Stromes I_D durch den MOSFET erfolgt leistungslos [5]. Aufgrund dieser Tatsache wird die Ansteuerung für die Berechnung der auftretenden Verlustleistungen vernachlässigt.

Ist der MOSFET durchgeschaltet, so hat er je nach Gatespannung U_{GS} einen bestimmten Widerstand $R_{DS_{on}}$, an dem Leistung umgesetzt wird. Bei dem verwendeten MOSFET IRFZ44N (siehe Abbildung 2.9a) liegt der Widerstand laut Datenblatt bei einer angelegten

Spannung von $U_{GS} = 10\,\mathrm{V}$ bei $R_{DS_{on}} = 17\,\mathrm{m\Omega}$ [21]. Die umgesetzte Leistung in Watt $[\mathrm{W}]$ lässt sich durch den physikalischen Zusammenhang von Strom und Spannung [15] nach der folgenden Formel berechnen:

$$P = U \cdot I. \tag{2.7}$$

Kombiniert man diesen Zusammenhang mit dem Ohmschen Gesetz (siehe Gleichung 2.3), so kann man errechnen, welche Leistung am n-Kanal des MOSFET umgesetzt wird:

$$P = I_D^2 \cdot R_{DS_{on}} \tag{2.8}$$

Der MOSFET dient somit als Schalter und überträgt die Form des Steuer-PWM-Signals auf die Ausgangsspannung des Netzteiles mit dem Spitzenwert $\hat{u}$. Diese Ausgangsspannung kann später genutzt werden, um am Paneel Leistung in Form von Joulescher Wärme umzusetzen. Diese Temperatur gilt es zu überwachen und zu messen. Dafür ist eine geeignete Art der Temperaturmessung herauszuarbeiten.

2.9. Digitale Temperaturmessung und Übertragung via Datenbus

Für die Überwachung und Messung der Temperatur auf dem Paneel kommen sowohl digitale als auch analoge Temperaturmessung in Frage. Bei einer analogen Messung mit beispielsweise einem PT100-Messwiderstand ist zum Auswerten der Temperatur eine Messbrücke nötig. Bei der Wheatstoneschen Messbrücke handelt es sich um eine Anordnung von Widerständen, wie in Abbildung 2.13 dargestellt.

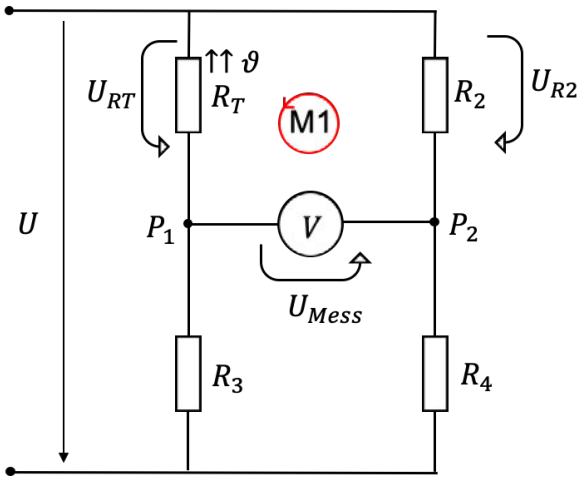


Abbildung 2.13.: Prinzipieller Aufbau einer Messbrücke für eine analoge Temperaturmessung

Über die in Reihe geschalteten Widerstände wird eine konstante Spannung U angelegt. Der Messwiderstand R_T verändert dabei abhängig von der ausgesetzten Temperatur seinen Widerstandswert. Wird beispielsweise ein PT100-Messwiderstand verwendet, beträgt der Widerstand des Sensors $100\,\Omega$ bei einer Temperatur von $0\,°\mathrm{C}$. Der Widerstand steigt linear mit der Temperatur [22]. Die anderen drei Widerstände werden so gewählt, dass die Messbrücke bei einer Temperatur von $0\,°\mathrm{C}$ ausgeglichen ist. Das bedeutet, dass alle Widerstände den gleichen Wert aufweisen und das Potential zwischen den beiden Messpunkten P_1 und P_2 somit $0\,\mathrm{V}$ beträgt. Mit steigender oder sinkender Temperatur erhöht oder verringert sich nun der Widerstandswert, was dazu führt, dass die Spannungen U_{RT} und U_{R2}, die über die Widerständen R_T und R_2 abfallen, nicht mehr gleich sind. Dadurch entsteht nach dem 2. Kirchhoff'sche Gesetz [15] ein messbares Potential zwischen den Punkten P_1 und P_2. Dieses Potential kann mit einem analogen Eingang eines Arduinos ausgewertet und so die Temperatur bestimmt werden. Da der verwendete Arduino jedoch nur 16 analoge Eingänge aufweist, was die Anzahl der verwendeten Sensoren begrenzt, und der Hardwareaufwand mit einer Messbrücke pro Messwiderstand ausgesprochen hoch ist, wird auf die digitale Temperaturmessung zurückgegriffen. Dafür kommt das digitale Thermometer DS18B20 von Maxim Integrated zu Einsatz, welches Messdaten über einen Datenbus übermittelt. Der Hardwareaufwand sinkt dadurch erheblich.

Das vom Werk aus kalibrierte digitale Thermometer DS18B20 bietet Temperaturmessungen in einem Messbereich von $-55\,°\mathrm{C}$ bis $125\,°\mathrm{C}$ mit einem geringen Messfehler. Dieser

Messfehler beträgt in dem Messbereich von $-10\,°\mathrm{C}$ bis $85°\,\mathrm{C}$ $\pm 0{,}5\,°\mathrm{C}$ [1]. Der Sensor benötigt eine Versorgungsspannung von $3\,\mathrm{V} - 5{,}5\,\mathrm{V}$. Im standby Zustand, in dem der Sensor weder die Temperatur misst noch kommuniziert, fließt ein Strom von $750\,\mathrm{nA}$.

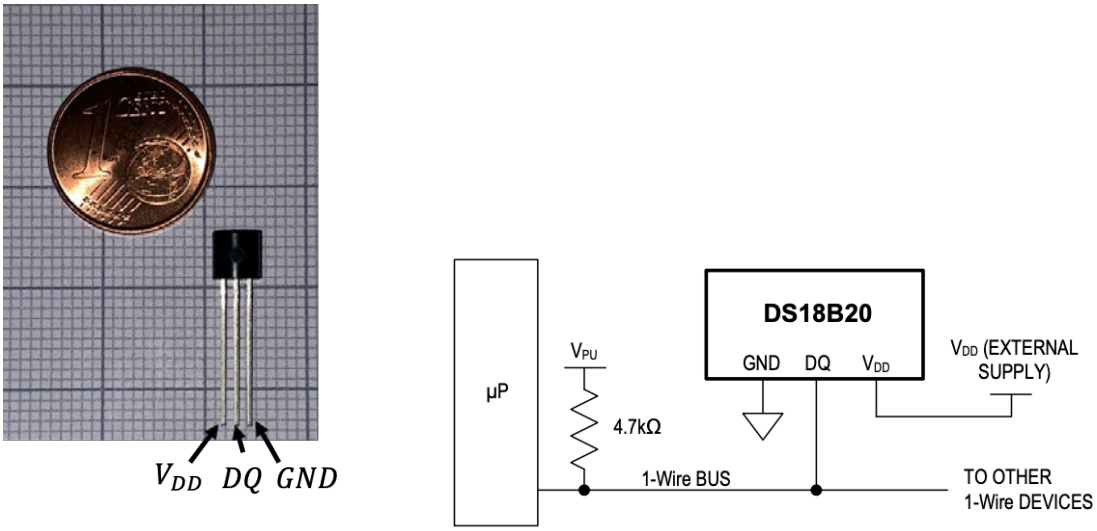
Der DS18B20 misst die Temperatur analog und wandelt diese intern in einen digitalen Wert um. Für diese Analog-Digital-Wandlung stellt der Sensor eine Auswahlmöglichkeit der Auflösung zur Verfügung. Diese kann vom Benutzer auf 9, 10, 11 oder 12 Bit eingestellt werden, was Inkrementen von $0{,}5°\,\mathrm{C}$, $0{,}25°\,\mathrm{C}$, $0{,}125°\,\mathrm{C}$ bzw. $0{,}0625°\,\mathrm{C}$ entspricht. Die Standardauflösung beim Einschalten ist 12-Bit. Die Erhöhung der Auflösung hat jedoch den Nachteil, dass die Zeit der Temperaturabfrage steigt. Dies ist Tabelle 2.6 zu entnehmen. Die Stromaufnahme während einer Temperaturumwandlung liegt typischerweise bei $1\,\mathrm{mA}$. Für den Systementwurf ergibt sich daher, dass der Arduino mit einem externen Netzteil versorgt werden muss, um die gleichzeitige Temperaturabfrage aller angeschlossenen Sensoren zu gewährleisten.

Auflösung in Bit	Temperaturumwandlungszeit in ms
9	93,75
10	187,5
11	375
12	750

Tabelle 2.6.: Temperaturumwandlungszeit in Abhängigkeit der Auflösungseinstellung [1]

Der DS18B20 kommuniziert über einen Datenbus, den 1-Wire-Bus. Hier wird nur eine Datenleitung für die Kommunikation mit einem zentralen Mikroprozessor benötigt. Das verringert den Hardwareaufwand und erhöht so die Übersichtlichkeit des Systems. Jeder DS18B20 verfügt über einen eindeutigen 64-Bit-Seriencode, wodurch eine gezielte Kommunikation mit einem ausgewählten Sensor möglich ist. Ein Mikroprozessor reicht somit zur Steuerung zahlreicher DS18B20 aus. Die Kommunikation und das Übertragen von Daten erfolgt über eine definierte Abfolge von Pegelflanken auf der Datenleitung. Dafür ist es nötig, die Datenleitung mit einem Pullup-Widerstand auf einen High-Pegel zu ziehen [1]. Die Funktion des Pullup-Widerstands kann Abschnitt 3.2.1 entnommen werden.

Der Anschluss eines DS18B20 kann dem Schaltplan in Abbildung 2.14b entnommen werden.



(a) Sensor DS18B20 im Größenvergleich

(b) Schaltplan für des Anschluss von DS18B20 Sensoren an den Datenbus [1]

Abbildung 2.14.: Temperatursensor DS18B20

Die Kommunikation über den 1-Wire Bus erfolgt nach dem Master-Slave Zugriffverfahren [6]. Das bedeutet, ein Master, in diesem Fall der Arduino, sendet über die Datenleitung ein bestimmtes Protokoll mit dem Aufruf zur Kommunikation mit einem ausgewählten Sensor, dem Slave. Erst nach Erkennen seines Aufrufes antwortet der adressierte Slave mit den geforderten Daten auf dem Bus. Um mehrere Slaves hinsichtlich ihres Anschlusses an den Bus zu organisieren, gibt es verschiedene definierte Schaltungsstrukturen von Master und Slave, wovon drei ausgewählte beschrieben werden.

1. Lineare Topologie - Der 1-Wire-Bus reicht vom Master bis zum entferntesten Slave. Andere Slaves werden mit kleineren Abzweigungen oder direkt an den 1-Wire-Bus angeschlossen [6].

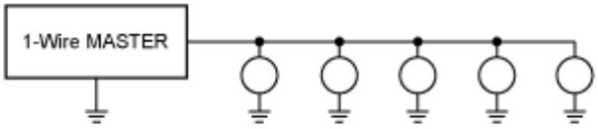


Abbildung 2.15.: Lineare Topologie [6]

2. Sterntopologie - Der 1-Wire-Bus ist direkt am Master oder in der Nähe aufgeteilt. Er erstreckt sich über mehrere Zweige unterschiedlicher Länge. Entlang dieser Zweige oder an den Enden befinden sich die Slaves [6].

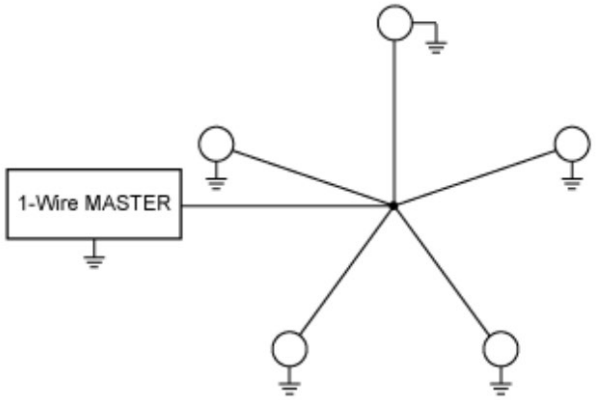


Abbildung 2.16.: Sterntopologie [6]

3. Geschaltetes Netzwerk - Hier werden zwei Topologien miteinander vermischt. Um ein Netzwerk, wenn es in seiner Komplexität steigt, nicht zu überlasten, wird es in Abschnitte unterteilt, die einzeln elektronisch eingeschaltet werden. Dabei ähnelt das Netzwerk physisch der Sterntopologie, elektrisch jedoch einer linearen Topologie. Durch gesteuertes Betätigen der Schalter ist immer nur ein Zweig aktiv [6].

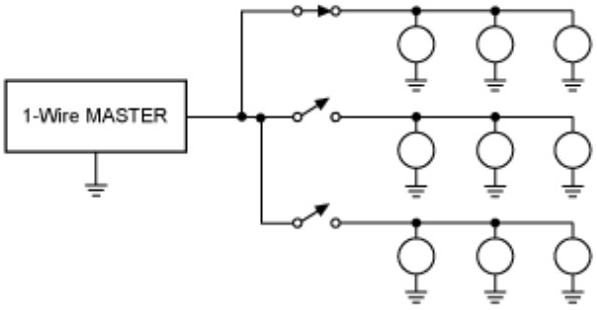


Abbildung 2.17.: Geschaltetes Netzwerk [6]

Das geschaltete Netzwerk und die Sterntopologie bringen den Vorteil mit sich, dass sie eine hohe Ausfallsicherheit haben. Sollte die Datenleitung in einem der Sternausläufer

unterbrochen werden, so können die restlichen Sternausläufer weiterhin mit dem Master kommunizieren. Dies ist bei der Linientopologie nicht der Fall. Wird die Datenleitung hier gekappt, so werden ab der defekten Stelle alle folgenden Sensoren vom Master nicht erreicht. Ein weiterer Vorteil der geschalteten Netzwerk und der Sterntopologie ist die Modularität, die sich durch den Sternpunkt direkt am Master ergibt. Hier können simpel und während des Betriebes weitere Sternausläufer hinzugefügt werden.

Die definierten Protokolle und genauere Informationen über die Kommunikation von Master und Slave können dem Datenblatt im Anhang entnommen werden (QUELLE im Anhangxxxx). Um auswertungsrelevante Daten speichern und wieder abrufen zu können, muss eine Art der Datenspeicherung dem System hinzufügt werden.

2.10. Datenspeichereinheit

Da der Arduino über keine Möglichkeit verfügt, Daten zu speichern, muss ein alternatives Speichermedium in das System integriert werden. Generell kann dafür jedes Gerät mit der Kompatibilität, mit einem Arduino zu kommunizieren, mit verfügbarem Speicherplatz verwendet werden, z.B. ein Raspberry Pi. Um die Komplexität des Systemaufbaus jedoch gering zu halten, wird ein Micro SD Card Memory Modul verwendet. Der Speicherplatz wird dabei durch eine integrierte Micro SD Karte zur Verfügung gestellt.

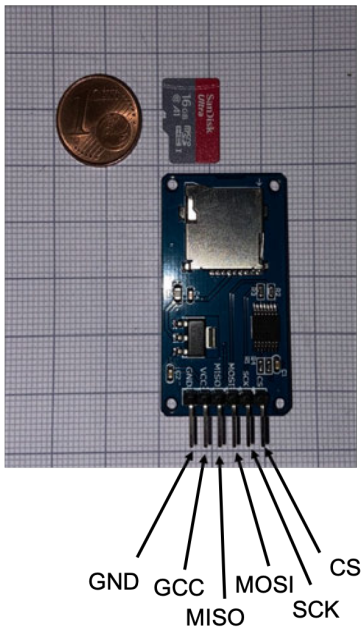


Abbildung 2.18.: Micro SD Card Memory Modul

Die Kommunikation zwischen Arduino und dem Modul findet über eine SPI-Schnittstelle statt. Serial Peripheral Interface (SPI) ist ein synchrones serielles Datenprotokoll, das von Mikrocontrollern für die schnelle Kommunikation mit einem oder mehreren Peripheriegeräten über kurze Entfernungen verwendet wird. Es kann auch für die Kommunikation zwischen zwei Mikrocontrollern verwendet werden [23]. Bei einer SPI-Verbindung gibt es einen Master, der normalerweise ein Mikrocontroller ist. Die zu steuernden verbundenen Peripheriegeräte sind die Slaves. Es gibt drei Leitungen, über die der Master mit allen Slaves verbunden ist:

1. MISO (Master In Slave Out) - Die Datenleitung des Slaves zum Senden von Daten an den Master
2. MOSI (Master Out Slave In) - Die Datenleitung des Masters zum Senden von Daten an die Peripheriegeräte
3. SCK (Serial Clock) - Eine vom Master generierte Taktung , die die Datenübertragungen synchronisiert

Weiterhin ist für jedes Slave-Peripheriegerät eine zusätzliche spezifische Leitung mit Verbindung zum Master vorgesehen, die als CS-Leitung (Chip Select) bezeichnet wird . So kann der Master ausgewählte Geräte zur Kommunikation freischalten. Eine ausführlichere

Beschreibung der SPI-Kommunikation kann [23] entnommen werden. Die gegebenen Informationen sind bei der Schnittellendefinition zu beachten.

Nachdem alle Komponenten und Methoden erarbeitet sind, gilt es im folgenden Kapitel den einen Systementwurf vorzunehmen. Dabei werden die Komponenten in Domänen aufgeteilt und es werden Schnittstellen definiert. Anhand der Aufteilung wird zunächst eine domänenspezifische Entwicklung vorgenommen, bevor das System schließlich an den Schnittellen zusammengeführt wird. Abschließend werden die Funktionen mit dem Lastenheft verglichen.

3. Domänenspezifische Entwicklung und Systemintegration

3.1. Vorbereitende Maßnahmen für die domänenspezifische Entwicklung

In diesem Kapitel wird das System anhand der zusammengestellten Anforderungen und nach Vorgaben des V-Modells entwickelt. Zuerst wird dafür eine detaillierte Funktionsstruktur erstellt. Diese schlüsselt das Gesamtsystem in Unterbaugruppen auf und zeigt die Verknüpfungen zwischen den Baugruppen anhand der Flüsse von Systemgrößen. Anschließend wird die Zugehörigkeit jeder Komponente zu einer Domäne herausgearbeitet. Die Einteilung findet in folgende Domänen statt:

- Elektrotechnik: Unter diesem Punkt werden die elektrischen Komponenten verbunden und zusammengefügt. Es werden Schaltpläne erstellt und die Funktion von Unterbaugruppen verifiziert.
- Informatik: Hier wird die Herangehensweise und die letztendliche programmierte Funktion der Regelung beschrieben. Dabei wird zur Visualisierung des Programmes insbesondere auf Flussdiagramme zurückgegriffen.
- Mechanik:Der Entwurf einer Sensorhaltung für die Temperatursensoren ist Gegenstand der Domäne Mechanik.

Darüber hinaus werden die Schnittstellen zwischen den Baugruppen unterschiedlicher Domänen definiert. Dabei gilt in diesem Zusammenhang eine Schnittstelle als die Definition, wie ein Fluss zwischen Domänen zu Stande kommt. Jeder Fluss von Informationen oder

Energie zwischen domänenspezifischen Baugruppen (siehe Abbildung 3.1) wird dafür betrachtet und es wird festgelegt, in welcher Art der Fluss stattfindet. So werden zusätzlich Funktionalitäten deutlich, über die Unterbaugruppen verfügen müssen, um diese Flüsse aufzunehmen und sie gegebenenfalls zu interpretieren. Dies verlässlich festzulegen ist essenziell für eine domänenspezifische Entwicklung und eine anschließende reibungslose Systemintegration, bei der die Arbeiten der Domänen zusammengeführt werden.

Die folgende Grafik zeigt die Funktionsstruktur des Systemes. Sie schlüsselt die Blackbox aus Abschnitt 2.3 in definierte Subkomponenten auf und zeigt die auftretenden Systemflüsse der Komponenten auf.

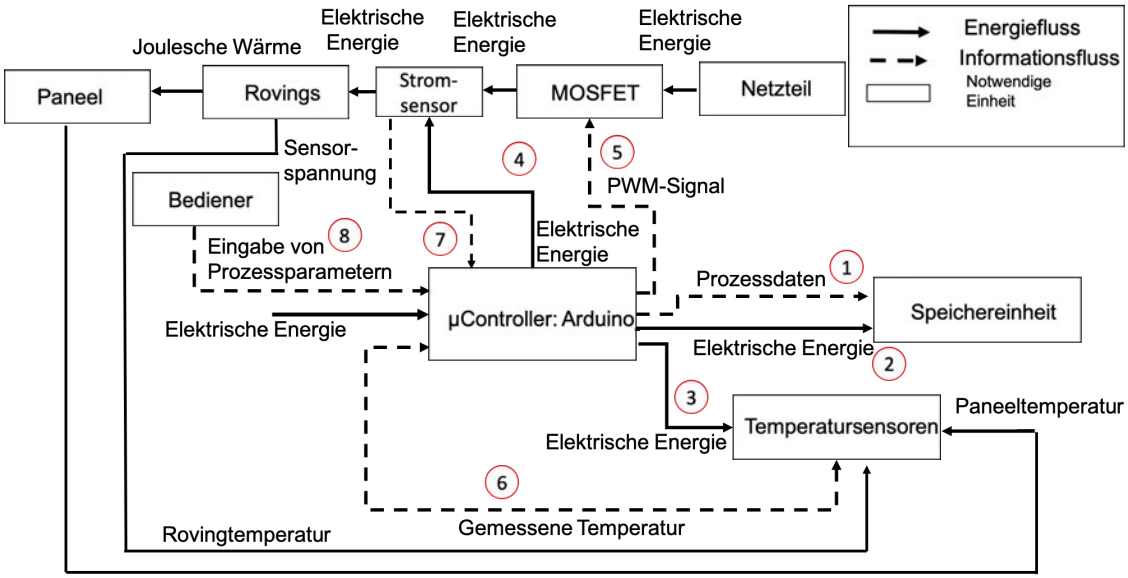


Abbildung 3.1.: Funktionsstruktur des Systemaufbaus

3.2. Schnittstellendefinition

Die Schnittstellen werden wie beschrieben nach der Zuordnung der Baugruppen zu Domänen definiert. Findet zwischen Baugruppen unterschiedlicher Domänen ein Fluss jeglicher Art statt, so muss definiert werden, über welchen welchen Kanal der Fluss zu Stande kommt. Jene Domänenzuordnung ist der Tabelle 3.1 zu entnehmen.

Baugruppe	Elektrotechnik	Informationstechnik	Mechanik
Controller: Arduino		X	
Speichereinheit	X		
Temperatursensoren	X		
Netzteil	X		
MOSFET	X		
Roving	X		
Stromsensor	X		
Paneel	X		
Sensorhalterung			X

Tabelle 3.1.: Domänenzuordnung der Baugruppen

Analysiert man anschließend, welche Baugruppen unterschiedlicher Domänen durch Flüsse verbunden sind, so ergibt sich eine Anzahl von acht zu definierenden Schnittstellen. Diese sind in Abbildung 3.1 nummeriert eingezeichnet.

1. Informationsfluss μController $\longleftrightarrow$ Speichereinheit : Die SPI-Schnittstelle wird verwendet, um zu speichernde Daten von dem μController an das Speichermodul zu senden oder abzufragen. Sowohl Speichermodul als auch μController verfügen über diese Funktionalität.
   - Ardunino: Pin 50 $\longleftarrow$ Speichereinheit : $MISO$
   - Ardunino: Pin 51 $\longrightarrow$ Speichereinheit : $MOSI$
   - Ardunino: Pin 52 $\longrightarrow$ Speichereinheit : SCK
   - Ardunino: Pin 53 $\longrightarrow$ Speichereinheit : CS
2. Energiefluss μController $\longrightarrow$ Speichereinheit : Das Speichermodul wird von dem Arduino mit einer Versorgungsspannung versehen. Über diese Schnittstelle wird der Fluss von elektrischer Energie hergestellt.
   - Ardunino: $5\,\mathrm{V}$-Pin $\longrightarrow$ Speichereinheit: V_{cc}

3. Energiefluss μController $\longrightarrow$ Temperatursensoren : Die Temperatursensoren werden von dem Arduino mit einer Versorgungsspannung versehen. Über diese Schnittstelle wird der Fluss von elektrischer Energie hergestellt.

   - Ardunino: $5\,\mathrm{V}$-Pin $\longrightarrow$ Temperatursensoren: V_{cc}

4. Energiefluss μController $\longrightarrow$ Stromsensor : Der Stromsensor wird von dem Arduino mit einer Versorgungsspannung versehen. Über diese Schnittstelle wird der Fluss von elektrischer Energie hergestellt.

   - Ardunino: $5\,\mathrm{V}$-Pin $\longrightarrow$ Stromsensor: V_{cc}

5. Informationsfluss μController $\longrightarrow$ MOSFET : Der Informationsfluss zwischen Arduino und MOSFET enthält ein zu verstärkendes PWM-Signal. Der Mosfet schaltet in den $\tilde{T}_{on}$- Phasen und lässt für jene Zeit die angeschlossene Spannung des Netzteils in das Paneel fließen.

   - Ardunino: Pin 9 $\longrightarrow$ MOSFET : $Gate$

6. Informationsfluss Temperatursensoren $\longleftrightarrow$ μController : Um einen Informationsfluss zwischen den Temperatursensoren und dem Ardunio herzustellen, wird die Kommunikation via bidirektionalen 1-Wire Bus festgelegt. Sowohl der Arduino als auch die Temperatursensoren DS18B20 sind in der Lage, über diesen Bus zu kommunizieren.

   - Arduino: Pin 2 $\longleftrightarrow$ DS18B20: DQ (Datenleitung)

7. Informationsfluss Stromsensor $\longrightarrow$ μController : Um einen Informationsfluss zwischen dem Stromsensor und dem Ardunio herzustellen, wird der Fluss über einen analogen Pin des Arduinos aufgenommen.

   - Arduino: Analog Pin 1 $\longrightarrow$ Stromsensor: out (Datenleitung)

8. Informationsfluss Bediener $\longrightarrow$ μController : Die Eingaben des Benutzers werden aus Text-Dateien, gespeichert auf dem SD-Kartenmodul, über eine SPI- Schnittstelle ausgelesen. So muss der Benutzer keine Änderungen direkt im Programmcode vornehmen.

Nach der Festlegung des Aufbaues und der Schnittstellen wird nun in die domänenspezifische Entwicklung übergegangen. Zunächst wird sich der Domäne der Mechanik gewidmet. Diese beschäftigt sich mit der Anordnung der Temperatursensoren auf dem Paneel und der Entwicklung einer Sensorhalterung, die die Sensoren in gewollter Anordnung hält. Zudem wird eine Methode entwickelt, die Temperatursensoren auf den Rovings zu befestigen, um so deren Temperatur aufzunehmen.

3.3. Mechanik

Bei der Aufnahme der Oberflächentemperatur sollen Kenntnisse darüber erlangt werden, ob sich die durch die Rovings erzeugte Wärme homogen verteilt. Dafür gilt es, die Messarreys mit Temperatursensoren geeignet anzuordnen und sie dabei zu fixieren. Die ausgewählte Anordnung ist in Abbildung 3.4 dargestellt.

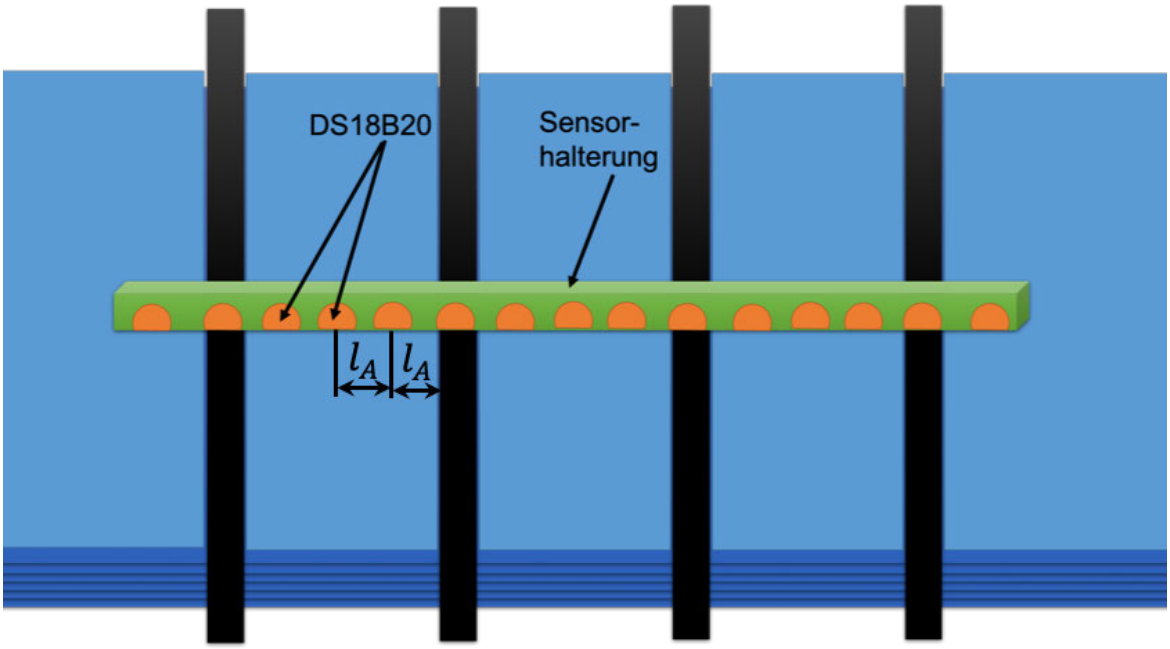


Abbildung 3.2.: Anordnung der Temperatursensoren auf dem Paneel

Über jedem Roving befindet sich ein Sensor des Messarreys. Um die Ausbreitung der Oberflächentemperatur zwischen den Rovings zu erfassen, wird der Bereich mit der Breite $25\,\mathrm{mm}$ (siehe Abschnitt 2.3) mit jeweils drei Temperatursensoren diskret abgetastet. Dabei

beträgt sowohl der Abstand zwischen Roving und Temperatursensor also auch der Abstand der drei Sensoren zueinander $l_A = 6{,}25\,\mathrm{mm}$.

Für die Fertigung einen solcher Halterung wird eine Holzlatte mittig mit Bohrungen nach definierten Abständen versehen. Der Kerndurchmesser einer Bohrung entspricht dabei mit $4{,}5\,\mathrm{mm}$ dem Durchmesser des Sensorkörpers des DS18B20. Anschließend wird das Holzstück der Länge nach halbiert. Abschließend werden die Sensoren mit einer geringen Menge Heißkleber in der Sensorhalterung fixiert. Das Ergebnis ist in Abbildung 3.3 dargestellt.

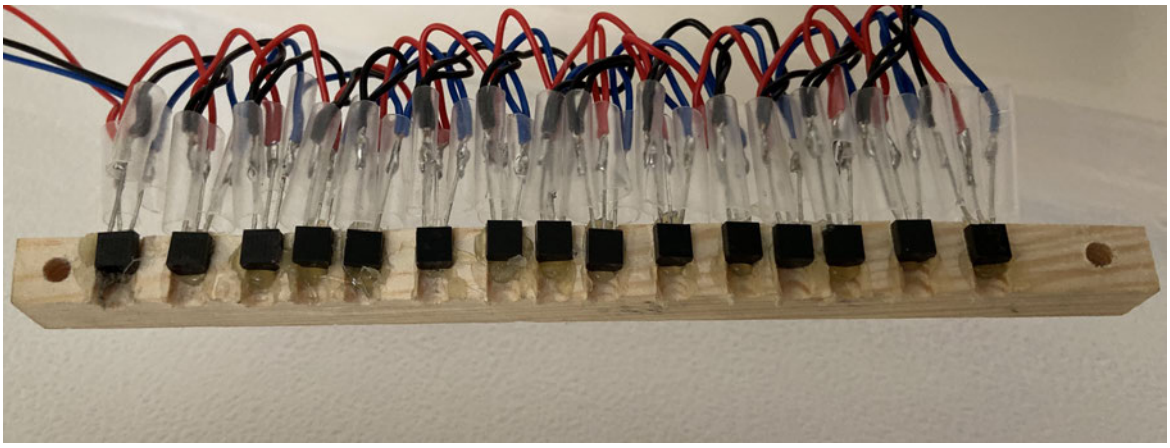


Abbildung 3.3.: Sensorhalterung eines Messarreys mit eingefügten Temperatursensoren

Für die Messung der Rovingtemperatur wird eine Klemmvorrichtung gefertig. Diese fixiert die Temperatursensoren auf je einem Kohelfaserroving.

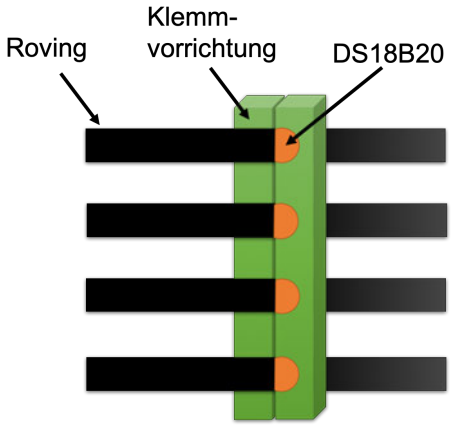


Abbildung 3.4.: Klemmvorrichtung für die Messung der Rovingtemperatur

Für alle Sensorhalterungen wird auf den Werkstoff Holz zurückgegriffen, da das verwendete Eichenholz eine vergleichsweise niedrige Wärmeleitfähigkeit von $0{,}166\,\frac{\mathrm{W}}{\mathrm{mK}}$ [24] aufweist, wohingegen Eisen einen über 400 mal höheren Wert von $72\,\frac{\mathrm{W}}{\mathrm{mK}}$ [25] besitzt. Diese Stoffeigenschaft gibt eine Aussage darüber, wie gut ein Material Wärme leitet. Je niedriger der Wert der Wärmeleitfähigkeit ist, desto besser ist die Wärmedämmung. So soll ein Wärmefluss zwischen Sensor und Sensorhalterung weitestgehend unterbunden werden, um eine Beeinflussung des Temperaturmesswertes zu vermeiden. Nach Abschluss der mechanischen Domäne wird zur Entwicklung der Elektrotechnik übergegangen.

3.4. Elektrotechnik

Abschnitt 3.4 befasst sich mit dem Aufbau der elektrotechnischen Komponenten, definiert durch Tabelle 3.1. Es werden Schaltpläne aufgezeigt und Kernfunktionen von Baugruppen überprüft.

3.4.1. Verstärkung des PWM-Signals und Anschluss des Versuchspaneels

Die Joulesche Wärme im Paneel wird durch den Gleichanteil eines verstärkten PWM-Signals erzeugt. Dabei wird der Gleichanteil im Folgenden als U_P bezeichnet, der Spitzenwert des PWM-Signals als $\hat{U}_P$. Zur Verstärkung des Pegels eines vom Arduino gesendeten PWM-Signals wird eine MOSFET-Schaltung verwendet. Die eingehenden Parameter $\tilde{T}_{on}$ und $\tilde{T}_{off}$ sollen dabei nicht verändert, sondern nur weitergegeben werden. So bleibt der Tastgrad bei dem zu verstärkenden und dem verstärkten Signal gleich. Während einer $\tilde{T}_{on}$-Phase schaltet der MOSFET und über dem Paneel liegt die Spannung des externen Netzteils $\hat{U}_P$ an. Bei einer anschließenden $\tilde{T}_{off}$-Phase hingegen soll der MOSFET sperren und es soll keine Spannung über dem Paneel abfallen. Um dies zu erfüllen, wird eine Schaltung nach Abbildung 3.5 aufgebaut.

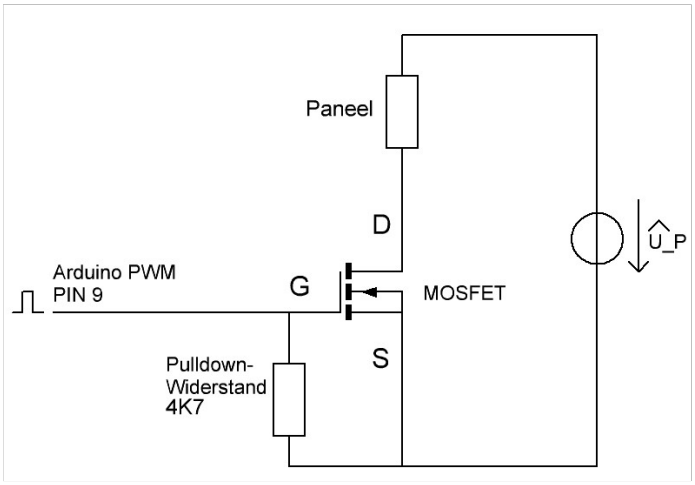
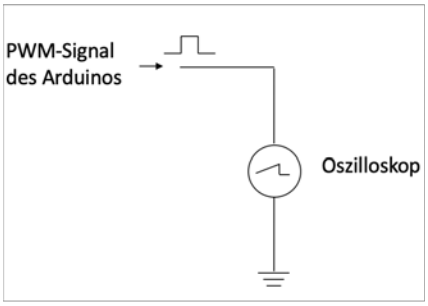
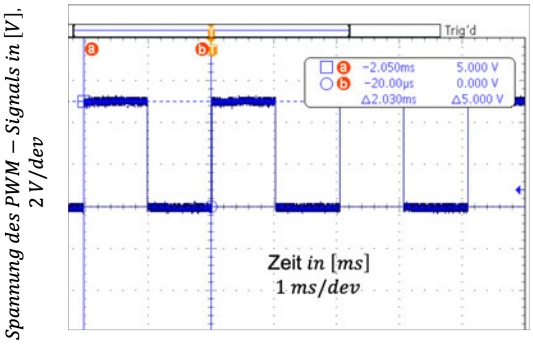


Abbildung 3.5.: Schaltplan der MOSFET-Schaltung

Die eine Seite des Paneels wird an den positiven Anschluss des Netzteils mit der Spannung $\hat{U}_P$ angeschlossen. Die andere Paneelseite wird mit dem Drain-Anschlusses des MOSFETs verbunden. Der Source-Anschluss des MOSFETs wird mit der Masse des Netzteiles verbunden und schließt damit den Stromkreis. Die Ansteuerung des MOSFETs an dem Gate-Anschluss erfolgt über den PWM-fähigen Pin 9 des Arduinos. Während der $\tilde{T}_{on}$-Phase des am Gate des MOSFETs anliegenden Signals wird im MOSFET ein elektrisches Feld aufgebaut, das zur Ausbildung eines leitfähigen Kanals innerhalb des MOSFETS führt - er wird leitend. Dafür muss zunächst jedoch überprüft werden, ob die Spannung des PWM-Signals des Arduinos $\hat{U}_{PWM_{Arduino}}$ größer ist als die maximale Schwellspannung $U_{th} = 4\,\mathrm{V}$ des verwendeten MOSFETs IRFZ44N. Dafür wird ein PWM-Signal aufgezeichnet (siehe Abbildung 3.6b).



(a) Schematischer Versuchsaufbau zur Aufnahme eines Arduino PWM-Signals mit einem Oszilloskop



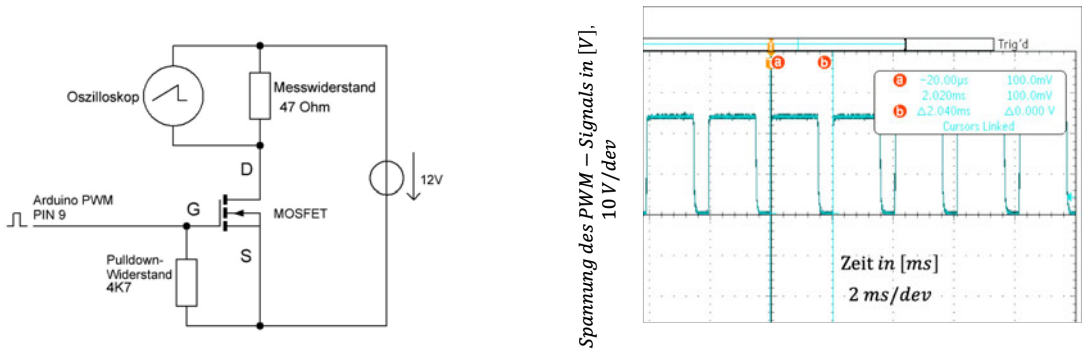
(b) Aufgezeichnetes PWM-Signal mit dem Tastgrad $p = 0{,}5$

Abbildung 3.6.: Aufnahme eines vom Arduino erzeugtem PWM-Signal

Zunächst wird die Frequenz des PWM-Signals anhand der vom Oszilloskop ermittelten Periodendauer von $\tilde{T}_{PWM} = 2{,}03\,\mathrm{ms}$ berechnet:

$$f_{PWM} = \frac{1}{\tilde{T}_{PWM}} = 429\,\mathrm{Hz} \tag{3.1}$$

Die anschließende Auswertung der Messung ergibt, dass sich während der $\tilde{T}_{on}$-Phase eine stabile Spannung von $\hat{U}_{PWM_{Arduino}} = 5\mathrm{V}$ einstellt. Die Schwellspannung U_{th} liegt hingegen bei $2\mathrm{V} - 4\mathrm{V}$. Damit ist sichergestellt, dass der MOSFET in der $\tilde{T}_{on}$-Phase schaltet. Damit der MOSFET in der $\tilde{T}_{on}$-Phase wieder sperrt, muss es ihm möglich sein, das elektrische Feld abzubauen. Dafür wird ein Pulldown Widerstand verbaut. Dieser sorgt während der $\tilde{T}_{off}$-Phase, in der keine Spannung U_{GS} anliegt, für ein Massenpotential am Gate-Anschluss. Das gespeicherte elektrische Feld kann so abgebaut werden und der MOSFET sperrt. Ist dieser nicht verbaut, bleibt das elektrische Feld, ähnlich wie bei einem Kondesator, im MOSFET bestehen und er bleibt durchgehend leitfähig. Um die Funktion der Schaltung zu überprüfen, wird diese an einen Roving eines Glasfaserpaneels angeschlossen. Dabei wird eine Versorgungsspannung von $\hat{U}_P = 12\mathrm{V}$ verwendet (siehe Abbildung 3.7a). Die über dem Roving abfallende Spannung wird von einem Oszilloskop aufgezeichnet (siehe Abbildung 3.7b). Der Gate Anschluss wird mit einem vom Arduino generierten PWM-Signal, Tastgrad $p = 0{,}75$, gespeist.



(a) Schematischer Versuchsaufbau zur Überprüfung der MOSFET-Verstärkerschaltung

(b) Das über einem Roving anliegende Leistungs-PWM-Signal mit dem Tastgrad $p = 0{,}75$

Abbildung 3.7.: Überprüfung der MOSFET-Verstärkerschaltung

Die Messung bestätigt die generelle Funktion der MOSFET-Verstärkerschaltung. Das verstärkte Signal weist, wie erwartet, die gleiche Periodendauer $\tilde{T}_{PWM}$ auf. Die Messung des Oszilloskopes ergibt einen Gleichanteil von $U_P = 8{,}74\,\mathrm{V}$ bei einem erwarteten Wert von $U_P = 9\,\mathrm{V}$. Anschließend werden zusätzliche Messpunkte aufgenommen, um die Linearität des Gleichwertes U_P im Verhältnis zum Tastgrad zu überprüfen. Das Ergebnis ist in dem folgenden Graphen dargestellt:

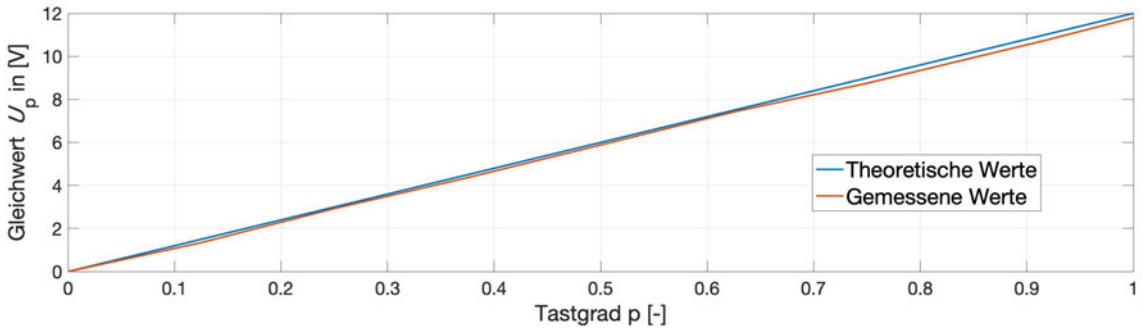


Abbildung 3.8.: Verhältnis von U_P zum Tastgrad

Die Linearität des Gleichanteils U_P im Verhältnis zum Tastgrad lässt sich deutlich erkennen. Es zeichnen sich maximale Abweichungen von $0{,}3\,\mathrm{V}$ ab. Erklärt werden können diese Abweichungen durch den nicht abrupten Abfall auf den Low-Pegeln zwischen der $\tilde{T}_{on}$- und der $\tilde{T}_{off}$-Phase. Es ist eine geringfügige Steigung zu erkennen (siehe Abbildung 3.7b).

Während dieser Zeit wird das elektrische Feld im MOSFET abgebaut, bis der Widerstand des leitfähigen Kanals zu hoch ist einen Stromfluss zu leiten. Erst dann ist der MOSFET wieder sperrend und der Low-Pegel von $U_P = 0\,\mathrm{V}$ stellt sich ein. Für die folgenden Berechnungen mit Verwendung des Gleichwertes U_P wird, aufgrund der geringen Abweichungen, mit dem theoretischen Gleichwert gerechnet.

Die erwarteten Widerstandswerte der einzelnen Rovings werden nachfolgend berechnet. Die Widerstandswerte dienen der Berechnung der erwarteten Ströme I_{F1-4}. Unter der Annahme, dass alle Fasern der Rovings die gleiche Länge besitzen, jede der n_f Fasern vorhanden ist und die gleiche Geometrie aufweist, kann der Widerstand eines Kohlefaserrovings und somit auch der Gesamtwiderstand der Anordnung bestimmt werden. Zuerst wird der Widerstand einer Faser berechnet:

$$R_f = \frac{\rho_f \cdot l_f}{A_f} = \frac{1{,}45 \cdot 10^{-3}\,\Omega\mathrm{cm} \cdot 40\,\mathrm{cm}}{\frac{\pi \cdot (5 \cdot 10^{-4}\,\mathrm{cm})^2}{4}} = 295{,}39\,\mathrm{k}\Omega \tag{3.2}$$

Der Widerstand eines Roving lässt sich unter der Betrachtung von $n_f = 24000$ parallel geschalteten Fasern bestimmen. So ergibt sich der Widerstand eines Rovings zu:

$$R_{F_{theoretisch}} = \left(\sum_{k=1}^{n_f} \frac{1}{R_{f_k}} \right)^{-1} = 12{,}308\,\Omega \tag{3.3}$$

Zusätzlich wird der Gesamtwiderstand der vier parallel geschalteten Rovings des Paneels berechnet:

$$R_{P_{theoretisch}} = \left(\sum_{k=1}^{4} \frac{1}{R_{F_k}} \right)^{-1} = 3{,}077\,\Omega \tag{3.4}$$

Anschließend werden die theoretischen Werte mit den real gemessenen Werten eines Glasfaserpaneels verglichen. Dafür wird der Widerstand von jedem der vier Rovings gemessen. Für die Messung werden die Rovingenden jeweils fest in eine Abgreifklemme eingespannt, um die Verbindung aller Fasern best möglich zu gewährleisten. Tabelle 3.2 zeigt die Messergebnisse auf.

Nummer des Rovings R_{F_i}	Widerstand des Rovings in $[\Omega]$	Theoretische erwarteter Wert in $[\Omega]$	Abweichung in $\%$
1	14,4	12,308	16,99
2	15	12,308	21,87
3	14,0	12,308	13,75
4	13,8	12,308	12,12

Tabelle 3.2.: Vergleich der realen Widerstandswerte der Kohlefaserrovings mit den theoretischen Werten; Messwerte wurden bei Raumtemperatur aufgezeichnet.

Die Messungen zeigen Abweichungen von $12{,}12\,\%$ bis hin zu $21{,}87\,\%$ auf. Diese lassen sich zum einen dadurch erklären, dass es bei der Messung vermutlich nicht möglich war, alle 24.000 Fasern der Rovings für eine perfekte Parallelschaltung zu verbinden. So würde der Widerstand steigen. Zum anderen können Fertigungsungenauigkeiten ein weiterer Punkt für die Abweichungen sein. Der theoretische ermittelte Widerstandswert beruht, wie erläutert, auf der Annahme eines perfekt gefertigten Rovings. In der Praxis weist dieser eventuelle Fehler, wie unterbrochene Fasern oder Abweichungen bezüglich der Anzahl und Geometrie der Fasern, auf. Bei der Messung ist aufgefallen, dass der Widerstand der Rovings mit steigender Temperatur fällt. So ist bei einer Temperatur von $100\,^{\circ}\mathrm{C}$, die mit einer Wärmebildkamera aufgenommen wurde, ein Abfall des Widerstandes um etwa $10\,\%$ beobachtet worden. Für die weitere Entwicklung des Systemes wird dieser Einfluss ignoriert, da dieser Effekt für alle Rovings gleichermaßen aufgefallen ist. Wäre dieser Effekt lediglich bei einzelnen Rovings beobachtet worden, müsste der Einfluss beachtet werden.

Mit den gemessenen Widerständen lässt sich nun sowohl der Gesamtwiderstand des Paneels R_P errechnen, als auch der erwartete Heizstrom $\hat{I}_P$. Der Heizstrom wird unter der Verwendung der Knotenregel durch die Addition der Ströme I_{F1-F4} bestimmt. Dafür wird lediglich die $\tilde{T}_{on}$-Phase des Leistungs-PWM-Signal betrachtet, in der die Spannung $\hat{U}_P$ über dem Paneel anliegt und der Heizstrom $\hat{I}_P$ fließt. In der $\tilde{T}_{off}$-Phase fließt hingegen kein Strom durch das Paneel. Die Ströme I_{F1-F4} werden nach dem Ohmschen Gesetzt anhand der gemessenen Widerstandswerte der Rovings berechnet. Dafür wird zunächst der

Gesamtwiderstand des Paneels berechnet:

$$R_P = \left(\sum_{k=1}^{4} \frac{1}{R_{F_k}} \right)^{-1} = 3{,}571\,\Omega \tag{3.5}$$

Anschließend kann der Heizstrom $\hat{I}_P$ berechnet werden:

$$\hat{I}_P = \frac{\hat{U}_P}{R_P} = \frac{12\,\mathrm{V}}{3{,}571\,\Omega} = 3{,}36\,\mathrm{A} \tag{3.6}$$

Bei einem MOSFET handelt es sich um einen steuerbaren Widerstand. Durchfließt diesen Widerstand nun der Heizstrom $\hat{I}_P$ muss beachtet werden, dass an seinem Kanalwiderstand $R_{DS_{on}}$ unvermeidlich eine Leistung umgesetzt wird. Es muss überprüft werden, ob diese auftretende Verlustleistung eine Relevanz für das weitere Vorgehen hat. Für die Berechnung wird nach [21] auf den Widerstandswert von $R_{DS_{on}} = 17\,\mathrm{m}\Omega$ zurückgegriffen. Es werden zunächst die beiden erwarteten Leistungen am MOSFET und am Paneel errechnet:

$$\hat{P}_{MOSFET} = \hat{I}_P^2 \cdot R_{DS_{on}} = 5{,}71\,\mathrm{mW} \tag{3.7}$$

$$\hat{P_P} = \hat{I}_P^2 \cdot R_P = 40{,}3151\,\mathrm{W} \tag{3.8}$$

Die Berechnungen zeigen, dass die umgesetzte Leistung am MOSFET während der $\tilde{T}_{on}$-Phase lediglich $0{,}014\%$ der Leistung ausmacht, die am Paneel umgesetzt wird. Im weiteren Vorgehen wird diese Verlustleistung daher vernachlässigt.

Um den Heizstrom zu messen, muss der Stromsensor in den Systemaufbau integriert werden. Der folgende Unterpunkt umfasst den Anschluss des ACS712-Moduls .

3.4.2. Integration des Stromsensors

Das ACS712-Modul interpretiert das Magnetfeld das entsteht, wenn der Heizstrom $\hat{I}_P$ den $R_S = 1{,}2\,\mathrm{m}\Omega$ Messwiderstand durchfließt. Dabei wird die messbare Signalspannung U_{out} nach dem proportionalen Zusammenhang: $U_{out} \sim \hat{I}_P$ ausgegeben. Für den Anschluss

bedeutet dies, dass der Heizstrom aus dem Netzteil zuerst den Sensor durchfließt und anschließend das Paneel.

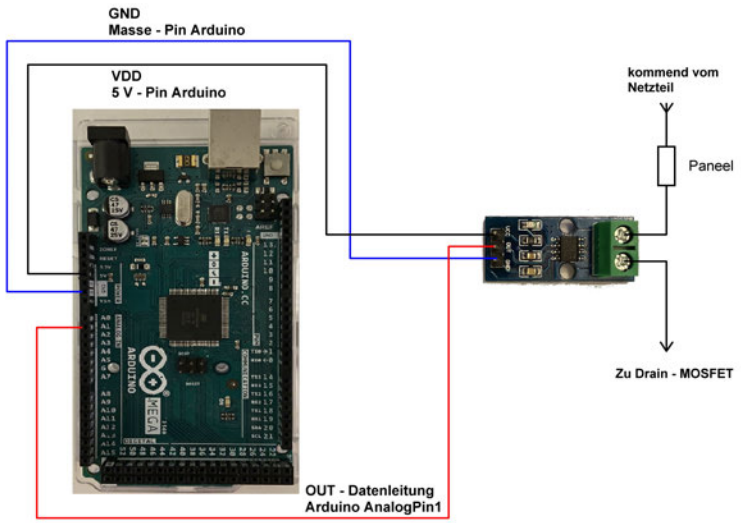


Abbildung 3.9.: Der Stromsensor, integriert in den Systemaufbau

Aufgrund des Flusses vom Heizstromes durch den Messwiderstand, wird hier ebenfalls eine Leistung umgesetzt. Diese wird wird in Gleichung 3.9 quantifiziert.

$$P_S = \hat{I}_P^2 \cdot R_S = 1{,}35\,\mathrm{mW} \tag{3.9}$$

Die Berechnungen zeigen, dass die umgesetzte Leistung am Stromsensor während der $\tilde{T}_{on}$-Phase lediglich $0{,}003\%$ der Leistung ausmacht, die am Paneel umgesetzt wird (siehe Gleichung 3.8). Im weiteren Vorgehen wird diese Verlustleistung daher ebenfalls vernachlässigt. Die Verifikation des Stromsensors wird aufgrund der hohen Abhängigkeit zur Programmierung in Unterabschnitt 3.5.4 behandelt. Hier wird der Messablauf und die analoge Messaufnahme näher beleuchtet. Der Fluss des Stromes $\hat{I}_P$ hat eine Erwärmung der Kohlefaserrovings zur Folge. Die diskrete Abtastung der Temperatur wird mithilfe von Temperatursensoren umgesetzt. Der Anschluss und die Anordnung der Sensoren ist Teil des folgenden Abschnittes.

3.4.3. Anordnung und Anschluss der Temperatursensoren

Für die Temperaturmessung sind digitale Temperatursensoren (DS18B20) vorgesehen. Die Paneeltemperatur wird durch drei Messarreys zu je 15 Sensoren aufgenommen. Zur Vorbereitung werden drei Arreys nach Abbildung 3.10 aufgebaut. Jeder Einzelne der 15 Masse-, Versorgungs- und Datenpins wird miteinander verbunden. Dabei ist eine klare Benennung der Sensoren für das weitere Vorgehen wichtig. Jeder Sensor erhält einen Präfix T mit einer angehängten Nummerierung $1-3$. Dies kennzeichnet die Zugehörigkeit des Sensors zu einem Messarrey. Anschließend folgt eine Nummerierung von $1-15$. Dabei ist der erste Sensor T_{X1} immer derjenige, der direkt mit dem Knotenpunkt des 1-Wire Bus verbunden ist. Die verwendete Laufvariable X ist dabei Platzhalter für die Arreynummerierung.

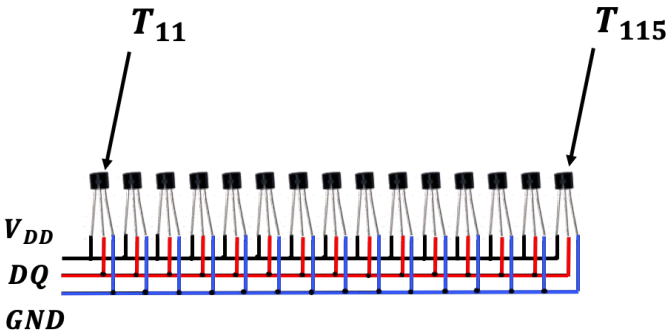


Abbildung 3.10.: Aufbau des ersten Messarreys

Für die Messung der Temperatur der Rovings wird ein viertes Messarrey mit vier Temperatursensoren aufgebaut, $T_{R_{1-4}}$. Der Aufbau erfolgt nach Vorbild der Abbildung 3.10.

Die Datenleitung ist vorbereitend durch einen Pullup Widerstand mit einem Wert von $4{,}7\,\mathrm{k\Omega}$ mit der Versorgungsspannung verbunden. Dieser sorgt für einen benötigten High-Pegel auf der Datenleitung. Während der erfolgenden Kommunikation wird die Datenleitung zur Übertragung von Informationen, in bestimmten Abständen durch Sensoren oder den Arduino auf einen Low-Pegel gezogen. Dies führt zu einem Potentialunterschied zwischen Datenleitung und der über den Pullup Widerstand verbundenen Versorgungsspannung. Um den unvermeidlichen Stromfluss hier nach dem Ohmschen Gesetz zu minimieren, wird der Wert des Pullup Widerstandes mit $4{,}7\,\mathrm{k\Omega}$ gewählt.

Anschließend folgt der Anschluss der vier Arreys an den 1-Wire Bus über die festgelegten Schnittstellen an Pin 2 des Arduinos nach Abbildung 3.11. Jedes Arrey wird durch den Arduino mit einer Spannung von $5\,\mathrm{V}$ versorgt und zudem durch einen Neutralleiter mit der Masse des Arduinos verbunden.

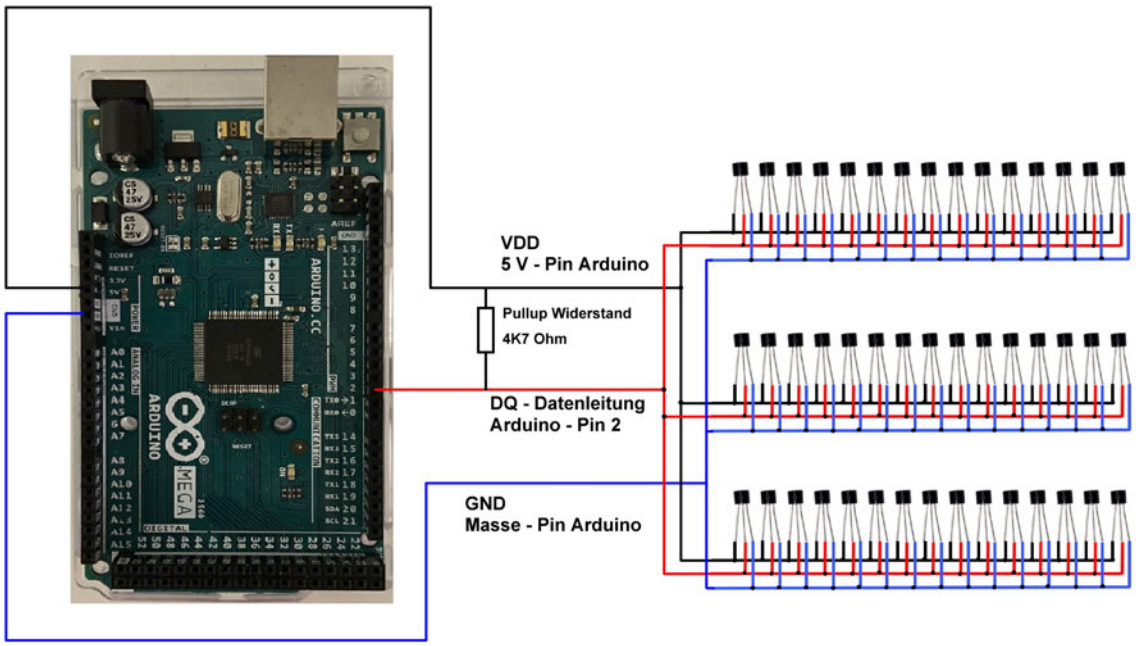


Abbildung 3.11.: Funktionsstruktur des Systemaufbaus

Die Sensorarreys sind mit ihrer Datenleitung am Master, dem Arduino, an einem Knotenpunkt verbunden. Daher liegt bei diesem Aufbau eine Sterntopologie vor. Diese hat den Vorteil, dass sie eine modularen Aufbau der Arreys zulässt: Jedes einzelne Messarrey kann von diesem Knotenpunkt entfernt und so vom Bus genommen werden. Genauso können Messarreys leicht hinzugefügt werden, indem sie mit dem Sternpunkt verbunden werden.

Abschließend wird die Messung der Temperatursensoren verifiziert und überprüft. Dafür wird ein Messarrey in eine gefertigte Sensorhalterung eingefügt, wie sie in Abschnitt 3.3 beschrieben wird.

Ziel ist es den Sensor in einem kontrolliertem Umfeld zu erhitzen und die gemessene Temperatur mit der Soll-Temperatur zu vergleichen. Der Prozess des Auslesens wird im Unterabschnitt 3.5.4 beschrieben. Zeigt die Auswertung des Messarreys Ergebnisse, die sich innerhalb des in den Anforderungen festgelegten Toleranzbereiches befindet, gilt die

Messkette anschließend als verifiziert. Für das Vorgehen wird ein Temperaturprüfschrank VT7011 von Vötsch verwendet. Dieser zeichnet sich durch eine Temperaturgenauigkeit von $\pm 1\,\mathrm{K}$ aus [26]. Zuerst wird dem Wärmeschrank eine Zieltemperatur von $40\,^{\circ}\mathrm{C}$ vorgegeben. Zum Zeitpunkt der Messung zeigt der Wärmeschrank eine Innenraumtemperatur von $T_{in} = 40{,}8\,^{\circ}\mathrm{C}$ an. Die Temperaturmessung des Messarreys liefert dabei Messwerte im Intervall $T_{sens} = [40{,}38\,^{\circ}\mathrm{C}; 40{,}5\,^{\circ}\mathrm{C}]$. Zusätzlich wird eine weitere Temperatur überprüft. Der Wärmeschrank wird auf $80\,^{\circ}\mathrm{C}$ aufgeheizt. Nachdem sich die Temperatur im Inneren den Ofens auf den eingestellten Wert eingeschwungen hat, wird erneut die Temperatur des Arreys ausgelesen. Die Messungen des Arreys ergeben Temperaturen von $T_{sens} = [78{,}58\,^{\circ}\mathrm{C}; 79{,}06\,^{\circ}\mathrm{C}]$. Auch diese Messwerte liegen innerhalb der gegebenen Toleranz. Das Vorgehen bestätigt also die generelle Messfunktion der Sensoren innerhalb der geforderten Toleranz.

Um im späteren Verlauf Vergleichsmessungen von den Temperaturen der Paneeloberfläche vornehmen zu können, wird im Zuge der Verwendung des Temperaturprüfschrank ebenfalls die Messung einer Fluke TIs75 - Wärmebildkamera überprüft. Es soll festgestellt werden, ob die Kamera die abgestrahlte Oberflächenwärme des Paneels unter Annahme der gleichen vorgegebenen Messtoleranz der Temperatursensoren erfassen kann. Dafür wird ein Glasfaser-Paneel in dem Temperaturprüfschrank platziert und es werden die selben beiden Kammertemperaturen, zunächst $40\,^{\circ}\mathrm{C}$, anschließend $80\,^{\circ}\mathrm{C}$, angefahren. Nachdem sich die Temperatur jeweils eingeschwungen haben und die Zieltemperatur durch den Temperaturprüfschrank angezeigt werden, wird das Paneel aus der Heizkammer entfernt und es wird mit der Wärmebildkamera zügig eine Aufnahme der Oberfläche gespeichert. Die Messungen der maximalen Oberflächentemperatur mittig des Paneels ergeben laut Kamera Werte von $T_{cam} = 40{,}6\,^{\circ}\mathrm{C}$ und $T_{cam} = 79{,}9\,^{\circ}\mathrm{C}$. Zusätzlich verifiziert diese Messung die Wärmebildkamera als geeignete Methode für Referenzmessungen der Paneeloberfläche. Die Referenzmessungen können zur Verifikation der Sensor-Messungen auf der Paneeloberfläche verwendet werden. Dies ist Gegenstand der nächsten Überprüfung. Nachdem die generelle Funktion der Sensoren in einer kontrollierten Temperaturumgebung verifiziert wurde, wird nun die Messgenauigkeit der Sensoren bezüglich der von den Rovings abgestrahlten Wärme untersucht. Diese wird nur auf einem Teil des Sensorkörpers abgestrahlt, der etwa $40\,\%$ der Gesamtoberfläche des Sensors entspricht. Die Wärmebildkamera wird dabei zur Aufnahme einer Vergleichsmessung verwendet. Für die Messung wird ein Temperatursensor in eine Sensorhalterung eingefügt und über einem Kohelfaserroving

platziert. Die Sensorhalterung wird dabei so fixiert, dass die Sensorkörperfläche möglichst ohne Luftspalt auf der Paneeloberfläche aufliegt. Die Wärmebildkamera zeichnet gleichzeitig die Wärmeentwicklung der Oberflächentemperatur über dem beheizten Roving auf. Anschließend wird die Verstärkerschaltung mit einer Reihe von Steuer-PWM-Signalen mit unterschiedlichem Tastgrad gespeist, um so den Gleichanteil U_P anzupassen. Während der Messungen ist eine deutliche Messträgheit der Temperatursensoren zu erkennen. Aus diesem Grund wird nach der Neueinstellung eines jeden Tastgrades eine Wartedauer von $5\,\mathrm{min}$ bis zum Ablesen der Sensortemperatur vorgesehen. Anhand der Messreihe werden $T_{sens}(p)$ und $T_{Cam}(p)$ in Abbildung 3.12 graphisch gegenübergestellt. Dabei wird für die die Folgende Dokumentation eine Abhängigkeit der Temperatur von der indirekten Messgröße des Tastgrades p hergestellt. Aufgrund des verifizieren proportionalen Zusammenhangs von $U_P \sim p$ nach Abbildung 3.8 und konstanten Widerstandswert der Rovings gilt zudem $I_P \sim p$. Eine Darstellung der Tempertur nach dem Zusammenhang $T(I_P(p))$ ist dem Anhang zu entnehmen (Anhang A.3.). Die Ergebnisse wurden mit einer Netzteilspannung von $\hat{U}_P = 13{,}86\,\mathrm{V}$ erzeugt. Diese Quellspannung wird ebenfalls, zum Erhalt der Aussagekraft, im späteren Versuchsaufbau verwendet.

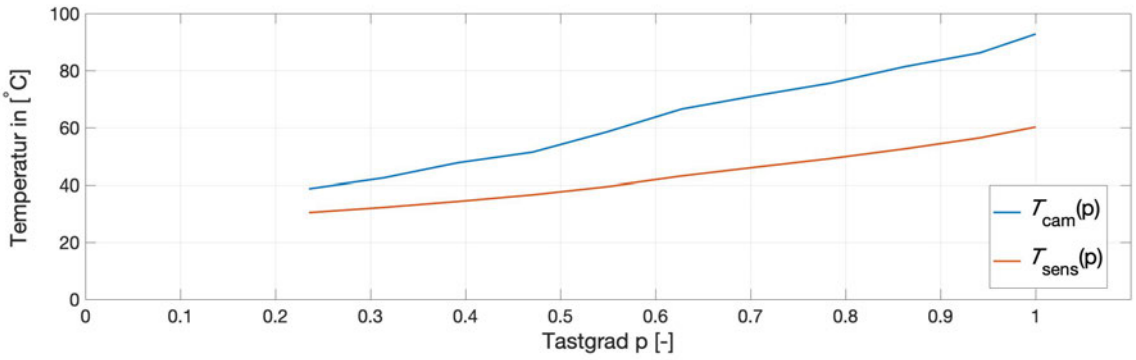


Abbildung 3.12.: Gegenüberstellung der Temperaturmessungen von Sensor und Wärmebildkamera

Die Ergebnisse der beiden Messmethoden zeigen eine signifikante Messabweichung des DS18B20 Temperatursensors im Vergleich zur verifizierten Messung mit der Wärmebildkamera. Im unteren Temperaturbereich, Tastgrad von $p = 0{,}23$, wird eine Abweichung von $\Delta\mathrm{T} = 8{,}4\,^{\circ}\mathrm{C}$ festgestellt. Im oberen Temperaturbereich, Tastgrad $p = 1$, sogar eine Abweichung von $\Delta\mathrm{T} = 32{,}61\,^{\circ}\mathrm{C}$.

Die Abweichungen werden auf zwei plausible Ursachen zurückgeführt: Zum einen ist

es denkbar, dass der erzeugte Wärmestrom zwischen Roving und Sensor nicht ausreicht, um die Wärmekapazität des Sensors vollständig zu füllen, während Wärme über die Oberfläche des Sensors zeitgleich mittels Konvektion wieder abgegeben wird. Zum anderen ist die Kontaktfläche zwischen Sensor und Paneeloberfläche nicht ideal glatt, sondern ist fertigungstechnisch von Unebenheiten geprägt. Dies führt zu einer zusätzlichen Verschlechterung der Wärmeleitung zwischen Paneel und Sensor. Die genaue Ursache der Abweichung lässt sich anhand der Messungen nicht bestimmen.

Als nächstes wird die Messung der Rovingtemperatur selber betrachtet. Hier ergibt sich bei einem Tastgrad von $p = 0{,}75$ und einer Haltezeit von ebenfalls $5\,\mathrm{min}$ eine durch die Wärmbildkamera festgestellte Rovingtemperatur von $T_{cam} = 101{,}8\,^{\circ}\mathrm{C}$. Ein Temperatursensor, der in die gefertigte Klemmvorrichtung eingefügt wurde (siehe Abschnitt 3.3), liefert eine gemessene Temperatur von $T_{sens} = 41{,}94\,^{\circ}\mathrm{C}$. Die Messaufgabe muss aufgrund der großen Abweichung zwischen den Messergebnissen als nicht erfüllt erkannt werden. Um die Funktion der Regelung dennoch nachzuweisen, wird im Unterabschnitt 3.5.5 eine geeignete Methode diskutiert.

3.4.4. Anschluss des Speichermoduls

Das Speichermodul und der Arduino verwenden das SPI-Protokoll für die Übertragung von Informationen. Für die Übertragung selber sind vier Verbindungen zum Arduino nötig, siehe Abschnitt 3.2. Zudem ist eine $5\,\mathrm{V}$ Spannungsversorgung notwendig, die der Arduino liefert. Der Anschluss ist in Abbildung 3.13 dargestellt.

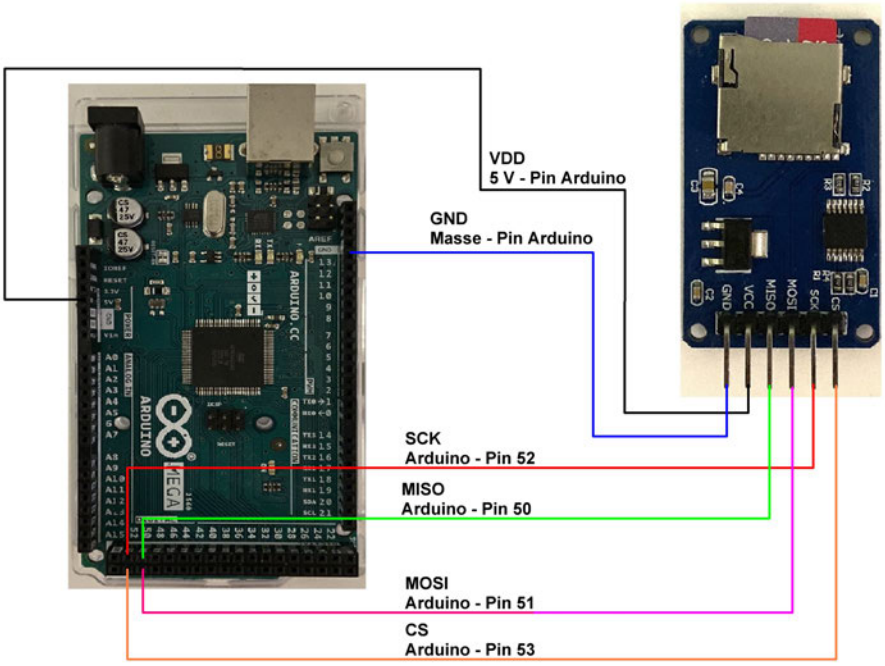


Abbildung 3.13.: Anschluss des Speichermoduls an die SPI-Schnittelle

Nach dem Aufzeigen des Aufbaus und Anschlusses aller Komponenten der Domäne Elektrotechnik, wird nun in die Domäne der Informatik übergegangen.

3.5. Informatik

3.5.1. Aufteilung des Programmes in Abschnitte

Für die Erfüllung der Aufgabenstellung wird ein Programm entwickelt, welches die Anforderungen erfüllt und dabei die definierten Schnittstellen beachtet. Der Ablauf erfolgt nach zyklischer Abarbeitung von Abschnitten, welche im Flussdiagramm in Abbildung 3.14 dargestellt sind.

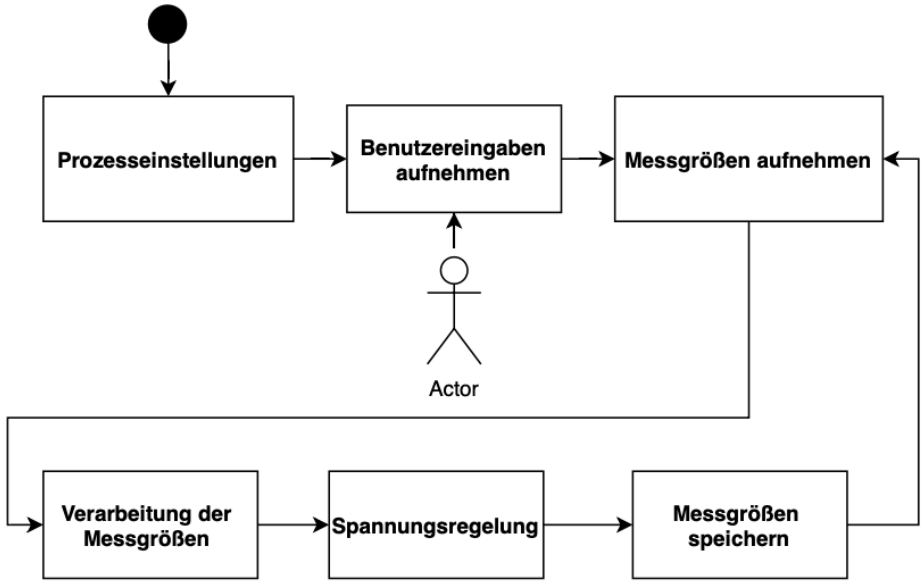


Abbildung 3.14.: Schrittweiser Ablauf des Regelungsprogrammes, dargestellt im Flussdiagramm

Nach der Prozesseinstellung und des Einlesens der Prozessparametern des Benutzers beginnt das Programm mit der Aufnahme der Messgrößen. In der anschließenden Verarbeitung werden die aufgenommenen Werte durch Algorithmen verarbeitet. Die Ergebnisse der Verarbeitung werden dem nächsten Schritt, der Spannungsregelung, zur Verfügung gestellt. Zum Schluss werden die gesammelten Messdaten an die Datenspeichereinheit übertragen und gespeichert. Der Ablauf wiederholt sich nach eingestellter Abtastfrequenz. Ziel der Regelung ist es, das Paneels durch eine adaptiv gestaltete Heizrate schnellstmöglich auf eine Zieltemperatur zu bringen und diese anschließend zu halten. Die folgenden Punkte gehen dabei auf die Abschnitte des Programmes ein. Unter zu Hilfenahme von Flussdiagrammen werden die Abläufe innerhalb der Schritte visualisiert.

3.5.2. Prozesseinstellungen

Für die Ausführung des Regelungsprogrammes müssen Voreinstellungen erfolgen. Außerdem Bibliotheken eingebunden werden, welche Funktionen enthalten, die die Programmierung der Regelung erleichtern.

- $OneWire.h$: Um generell mit Devices am 1-Wire Bus zu kommunizieren, muss diese Bibliothek eingebunden werden. Sie stellt Funktionen für den Zugriff auf 1-Wire Devices zur Verfügung, erzeugt Nachrichten nach dem definiertem 1-Wire Protokoll und übernimmt ebenfalls das Decodieren von empfangenen Nachrichten. Es wird dafür eine Instanz $oneWire$ der Klasse $OneWire$ erzeugt. Die Instanz übergibt die Information, welcher Pin des Arduinos als Datenpin genutzt wird.

- $DallasTemperature.h$: Diese Bibliothek ist in der Lage, die mit den Methoden der $OneWire.h$-Bibliothek erhaltenen Datenbits zu interpretieren und so die Auswertung von Nachrichten zu erleichtern. Dafür wird ebenfalls zuerst eine Instanz $sensors$ der Klasse $DallasTemperature$ erstellt. Als Übergabeinformation wird mit dem Referenzoperator $\&$ eine Referenz auf die Instanz $oneWire$ übergeben. Die Instanz $sensors$ kann so auf die Funktionen der Instanz $oneWire$ zugreifen, die erhaltenen Informationen verarbeiten und sie anschließend interpretiert ausgeben.

- $SPI.h$: Die Funktion der Bibliothek ist eine ähnliche wie jene von $OneWire.h$. Die Bibliothek codiert und decodiert Nachrichten der SPI-Schnittstelle. Die Erstellung einer Instanz ist hier nicht nötig. Die Funktionen der Klasse sind direkt verfügbar.

- $SD.h$: Speziell für die Kommunikation mit einem Speichermol über eine SPI-Schnistelle ist die Bibliothek $SD.h$ vorgesehen. Unter der internen Verwendung der Funktionen von $SPI.h$ wird direkt mit dem Speichermol kommuniziert. Die Erstellung einer Instanz ist hier ebenfalls nicht nötig, die Funktionen lassen sich mit dem Präfix $SD.$ aufrufen.

Mit $Serial.begin(9600)$ wird die serielle Kommunikation auf dem Arduino aktiviert. Dabei legt der Wert 9600 die Baudrate fest. So ist für das folgende Programm eine Übertragungsgeschwindigkeit von $9600 bits/sek$ definiert. Außerdem werden Pins definiert, die zum einen als Kommunikationsschnittstelle mit dem 1-Wire Bus dienen und zum anderen das PWM-Signal aussenden.

Der Fluss dieses Programmschrittes lässt sich der folgenden Grafik entnehmen.

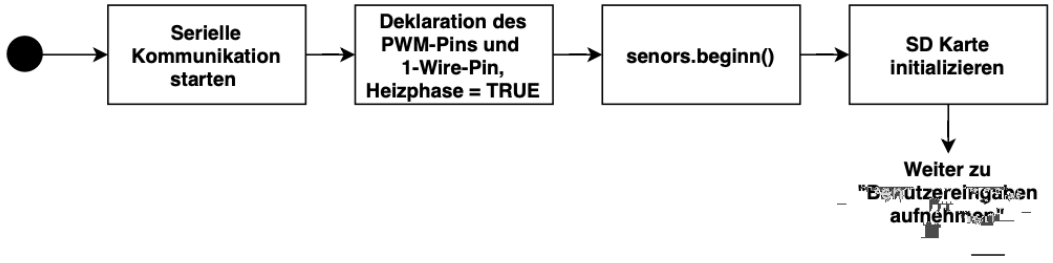


Abbildung 3.15.: Darstellung des Programmabschnittes der Prozesseinstellungen

Nachdem alle Vorbereitungen für den weiteren Programmverlauf vorgenommen sind, übernimmt das Programm die einstellbaren Prozessparameter des Benutzers.

3.5.3. Benutzereingaben aufnehmen

Der Bediener des Systemes soll vor dem Beginn der Regelung Prozessparameter einstellen können, um den Erwärmungsprozess und die Aufname der Temperatur zu gestalten. Dabei handelt es sich um die folgenden Parameter:

- Zieltemperatur der Paneeloberfläche T_{Ziel}: Die Zieltemperatur des Paneels gilt anschließend für das Regelungsprogramm als Grenztemperatur zwischen Heizphase und Haltephase: Wird die Zieltemperatur erstmals erreicht, wird die Heizphase beendet.
- Auflösung und Abtastrate : Die Auflösung der Temperatursensoren und die Abtastrate, mit der die Messarrays die Paneeltemperatur diskretisieren, hängen indirekt voneinander ab. Wie in Abschnitt 2.9 beschrieben, benötigen die Sensoren für die Umwandlung der anliegenden Temperatur in einen digitalen Wert, je nach gewollter Auflösung, unterschiedliche Zeiten. Zusätzlich muss Zeit für die Abarbeitung des Regelungsprogrammes vorgesehen werden. Aus diesem Grund werden zwei Einstellungsoptionen vorgesehen:
  1. Auflösung: 12Bit, Umwandlungszeit: $750\,\mathrm{ms} \rightarrow$ Abtastrate $f_T = 0{,}33\,\mathrm{Hz}$
  2. Auflösung: 11Bit, Umwandlungszeit: $375\,\mathrm{ms} \rightarrow$ Abtastrate $f_T = 0{,}75\,\mathrm{Hz}$

Die Parameter werden aus Text-Dateien ausgelesen, die sich auf der SD-Karte des Speichermoduls befinden und vom Benutzer vor dem Start gefüllt werden. Zunächst wird das SD-Karten Modul mit den hinterlegten Text-Dateien initialisiert. Dafür wird lediglich die Funktion $begin()$ der Instanz SD verwendet. Sie überprüft, ob die SD-Karte vorhanden und empfangsbereit ist. Diese Funktion liefert einen Rückgabewert vom Typ $bool$, dessen Wert das Ergebnis der Initialisierung ausgibt.

Auf der SD-Karte des Speichermoduls sind initial die beiden leeren Dateien $Abtastrate.txt$ und $Zieltemperatur.txt$ vorhanden. Das Auslesen einer Text Datei erfolgt nach den folgenden Muster: Zuerst wird mit dem Befehl $open()$ der Instanz SD eine gewünschte Datei geöffnet. Anschließend liest die Funktion $Serial.write(Textdatei.read())$ den Inhalt der geöffneten Datei aus. In diesem Schritt kann eine Zuordnung der ausgelesenen Daten zu einer Variablen stattfinden. Anschließend muss die Datei mit dem Befehl $close()$ geschlossen werden. Für das Einstellen einer Abtastrate mit zugehöriger Auflösung wird die Datei $Abtastrate.txt$ mit einer 11 oder 12, entsprechend der gewünschten Auflösung, gefüllt. Dabei legt die Abtastrate die Zeitdauer fest, nachdem die Messdaten erneut aufgenommen werden. Die gewünschte Zieltemperatur wird hingegen in die Datei $Zieltemperatur.txt$ eingefügt. Der Wertebereich liegt hier bei $T_{Ziel} = [25\,°\mathrm{C}; 85\,°\mathrm{C}]$. Dies entspricht dem Messbereich, in dem der Temperatursensor den geringsten Messfehler ausweist (siehe Abschnitt 2.9). Beide Parameter werden als Variablen gespeichert. Sollte einer der beiden ausgelesenen Parameter nicht im Wertebereich liegen, so wird die Regelung nicht gestartet. Befinden sich beide Parameter im gültigen Wertebereich, wird abschließend die gewünschte Auflösung der verwendeten Sensoren mit der Funktion $setResolution()$ eingestellt. Dafür wird jedem Sensor die vom Benutzer eingestellte Auflösung übergeben. Der Ablauf des Programmschrittes ist der folgenden Grafik zu entnehmen.

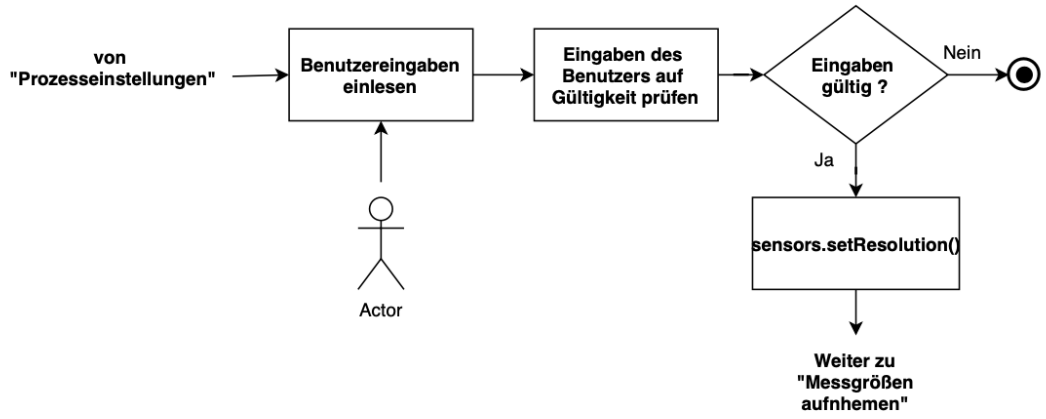


Abbildung 3.16.: Darstellung des Programmabschnittes der Aufnahme von Benutzereingaben

Nach Abschluss der Einstellungen ist das System bereit für das Aufnehmen der Messgrößen.

3.5.4. Aufnehmen der Messgrößen

Um Temperaturen gezielt von Sensoren am Bus auszulesen, muss vor Programmierungsbeginn der Regelung der 64-Bit-Seriencode eines jeden verwendeten Sensors ausgelesen werden. Dieser Seriencode ist die Adresse, mit der ein Sensor zur Kommunikation aufgefordert wird. Jeder Sensor wird dafür in eine simple Schaltung nach Abbildung 3.17 integriert.

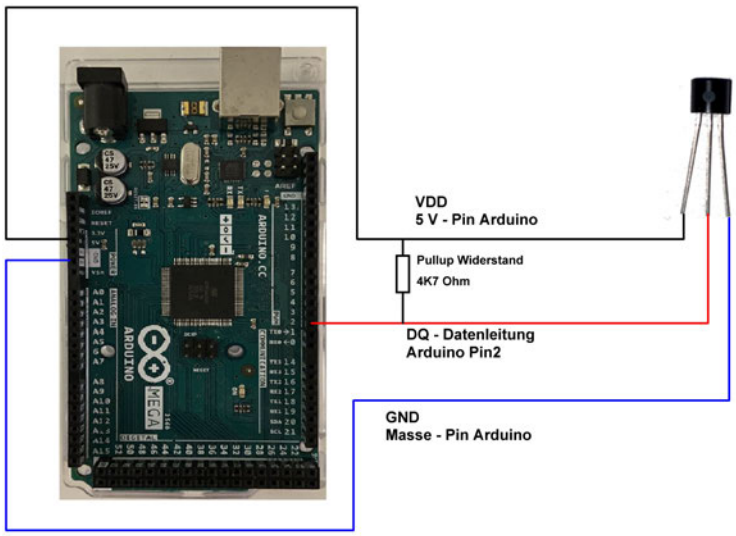


Abbildung 3.17.: Schaltungsaufbau zum Auslesen der Seriencodes der verwendeten DS18B20

Anschließend erfolgt die Ausführung der Funktion $adresseAusgeben$. Hier wird die aus der Funktion $search()$ der Instanz $oneWire$ erhaltene Adresse des an den 1-Wire angeschlossenen Sensors in hexadezimaler Form ausgegeben.

Als Nächstes werden die Device Adressen gespeichert. Die $DallasTemperature$ Bibliothek definiert dafür eine Variable vom Typ $DeviceAdress$, welches lediglich ein Byte-Arrey von acht Elementen enthalten kann. Dies entspricht genau der Länge einer Adresse. Es wird für jeden Sensor eine Variable vom Typ $DeviceAdress$ mit Inhalt der ausgelesenen Adresse, erstellt. Die Variable wird nach der vereinbarten Konvention (siehe Unterabschnitt 3.4.3) benannt. Das Ausleseprogramm wurde dafür konfiguriert, dass die Ausgabe bereits der Form eines Arreys entspricht. Demnach sind die hexadezimalen Zahlen mit Kommata getrennt.

Anschließend wird die Kommunikation mit den Sensoren am Datenbus durch den Befehl $sensors.beginn()$ gestartet. Mit dem Aufruf der Funktion $sensors.requestTemperatures()$ werden alle Sensoren am Bus dazu aufgefordert, den momentanen analogen Temperaturwert in einen digitalen umzuwandeln. Die Messarreys werden nacheinander ausgelesen, beginnend mit dem ersten. Für die Ausgabe der gemessenen Temperatur von ausgewählten Sensoren wird eine Funktion $printTemperature$ erstellt. Die Funktion benötigt als Übergabewert die Adresse vom Typ $DeviceAdress$ des auszulesenden Sensors. Also Rückgabewert liefert diese Funktion eine Temperatur in $^{\circ}C$. Gleichzeitig findet

innerhalb der Funktion eine Funktionsüberprüfung eines jeden ausgelesenen Sensors statt. Nicht funktionsfähige Sensoren können keine Flanken mehr auf dem Datenbus erzeugen. In diesem Fall wird ein Temperaturwert von $-127\,^{\circ}\mathrm{C}$ übertragen. Dies kann abgefangen werden und eine Fehlermeldung wird ausgegeben. Die Temperaturen der Messarreys werden nach der Abfrage in ein $float$-Arrey eingefügt. Für die folgende Auswertung wird zusätzlich ein Zeitstempel eines jeden Messauftrages gespeichert. Der Zeitstempel wird mit Hilfe der Funktion $millis()$ erzeugt. Diese liefert als Rückgabewert die vergangenen Sekunden seit Programmstart. Durch Differenzrechnung lässt sich so der zeitliche Abstand der Messungen bestimmen.

Bei der Strommessung wird auf analoge Messung zurückgegriffen. Der Arduino interpretiert durch die Funktion $analogRead()$ den an dem analogen Pin anliegende Signalspannung U_{out} des Stromsensors in einem Messbereich von $0\,\mathrm{V} - 5\,\mathrm{V}$ durch einen Analog/Digital-Wandler mit einer Auflösung von 10 Bit und gibt entsprechend einen Intergerwert zurück. Liegt beispielsweise eine Spannung von $5\,\mathrm{V}$ an, übergibt die Funktion den Maximalwert von 1023. Zu beachten ist, dass bei einer Signalspannung des Sensors von $2{,}5\,\mathrm{V}$ die gemessene Stromstärke $0\,\mathrm{A}$ beträgt. Anschließend kann in der Verarbeitung der Messgrößen durch Dreisatzrechnung anhand des Ergebnisses der Analog/Digital-Wandlung auf die Signalspannung des Stromsensors zurückgerechnet und damit der Heizstrom bestimmt werden.

$$I_{Mess} = (\frac{Analogwert}{1024} * 5000\,\mathrm{mV} - 2500\,\mathrm{mV}) \cdot \frac{1}{100\frac{mV}{A}} \tag{3.10}$$

Im Folgenden soll die Messung des Stromes verifiziert werden. Dafür wird der Strom durch einen Leistungswiderstand mit einem Widerstandswert von $12\,\Omega$ an einer $12\,\mathrm{V}$ Versorgungsspannung gemessen. So ist ein Stromfluss von $1\,\mathrm{A}$ gegeben. Durch ein Ausleseprogramm werden sowohl der analoge ausgelesene Wert, die zurückgerechnete Signalspannung des Sensors und der errechnete Stromwert am Sensor ausgegeben. Dabei werden die Messwerte mit einer Abtastrate von $0{,}5\,\mathrm{Hz}$ aufgenommen. Zunächst ist zu beachten, dass ein Stromwert von genau $1\,\mathrm{A}$ nicht aufgelöst werden kann. Die beiden nächsten auflösbaren Stromwerte sind $1{,}025\,\mathrm{A}$ und $0{,}977\,\mathrm{A}$. Es ist also zu erwarten, dass der gemessene Stromwert einen dieser beiden Werten annimmt. Die folgende Abbildung zeigt einen Auszug der Messdaten.

```
Sensorwert = 532        Sensorspannung in mV = 2597.656        Ampere = 0.977
Sensorwert = 531        Sensorspannung in mV = 2592.773        Ampere = 0.928
Sensorwert = 532        Sensorspannung in mV = 2597.656        Ampere = 0.977
Sensorwert = 532        Sensorspannung in mV = 2597.656        Ampere = 0.977
Sensorwert = 533        Sensorspannung in mV = 2602.539        Ampere = 1.025
Sensorwert = 534        Sensorspannung in mV = 2607.422        Ampere = 1.074
Sensorwert = 532        Sensorspannung in mV = 2597.656        Ampere = 0.977
Sensorwert = 532        Sensorspannung in mV = 2597.656        Ampere = 0.977
Sensorwert = 533        Sensorspannung in mV = 2602.539        Ampere = 1.025
```

Abbildung 3.18.: Auszug aus einer testweisen Strommessung

Die Messung bestätigt die aufgestellte Theorie. Es werden vermehrt die beiden Werte $1{,}025\,\mathrm{A}$ und $0{,}977\,\mathrm{A}$ gemessen. Jedoch treten auch einige größere Abweichungen auf. Diese lassen sich durch die 10-Bit Analog/Digital-Wandlung erklären. So wird beispielsweise für eine $0{,}977\,\mathrm{A}$-Messung und der dazu passende Analogwert von 532 ein Pegel von mindestens $2597{,}65\,\mathrm{mV}$ am messenden Analog-Pin nötig. Wird dieser auch nur minimal unterschritten, wird stattdessen als Ergebnis der Analog/Digital-Wandlung ein Anlogwert von 531 ausgegeben, welches einem Stromwert von $0{,}928\,\mathrm{A}$ entspricht. Spannungsschwankungen der Eingangsspannung oder andere Störeinflüsse, wie z.B. magnetische Felder von umstehenden Geräten, können diesen Messfehler begünstigen. Da der Hallsensor des Stromsensors das Magnetfeld des ihn durchfließenden Stromes auswertet, können umliegende Magnefeldstörquellen diese Messung verfälschen.

Anschließend wird der Stromsensor im realen Einsatz überprüft. Dafür wird eine von der Verstärkerschaltung erzeugte Spannung $U_P = 6\,\mathrm{V}$ bei $\hat{U}_P = 12\,\mathrm{V}$ an einem Kohelfaserroving mit $13{,}8\,\Omega$ angeschlossen, sodass ein Strom von $I_P = 0{,}43\,\mathrm{A}$ fließt. Der Stromsensor wird vor dem Roving in Reihe geschaltet und nimmt den Strom auf. Die anschließende Messung ergibt Strommesswerte von $I_P \approx 0\,\mathrm{A}$ oder $I_P \approx 0{,}87\,\mathrm{A}$. Dies lässt sich dadurch erklären, dass der Stromsensor Momentanwerte und keine Gleichwerte liefert. Um ein zuverlässiges Messergebnis des Verlaufes $I_P(t)$ zu erzeugen, müsste der Strom $\hat{I}_P(t)$ unter Berücksichtigung des Nyquist-Shannon Abtasttheorem [15] abgetastet werden. So ist es möglich eine exakte Rekonstruktion des zeitkontinuierlichen Stromverlaufes in einen diskreten umzuwandeln. Gemäß des Theorems bedarf es einer Abtastrate, die dem Doppelten der maximalen Frequenz des Signals entspricht. Für eine auftretende

PWM-Frequenz von $429\,\mathrm{Hz}$ nach Gleichung 3.1 müsste also mit einer Abtastfrequenz von $f_{ab} \geq 2 \cdot f_{PWM} \geq 858\,\mathrm{Hz}$ abgetastet werden. Wird diese kritische Abtastrate unterschritten, ist die Eindeutigkeit der Rekonstruktion nicht mehr gegeben. Anschließend müsste das diskrete Signal hinsichtlich der realen Parameter der Periodenzeit, $\tilde{T}_{PWM_{real}}$, der Zeitdauer des High-Pegels $\tilde{T}_{on_{real}}$ und des Highpegels des Stromes $\hat{I}_{P_{real}}$ untersucht werden. So könnte der reale Heizstrom $I_{P_{real}}$ errechnet werden:

$$I_{P_{real}} = \hat{I}_{P_{real}} \cdot \frac{\tilde{T}_{on_{real}}}{\tilde{T}_{PWM_{real}}}. \tag{3.11}$$

Aufgrund der späten Erkennung dieses Problems wird in Kapitel 4 der Heizstrom lediglich graphisch, anhand von Temperaturverläufen und Erkennung von Heizphasen, ermittelt. Der Stromsensor wird nicht in das System integriert.

Der Programmfluss ist in Abbildung 3.19 dargestellt.

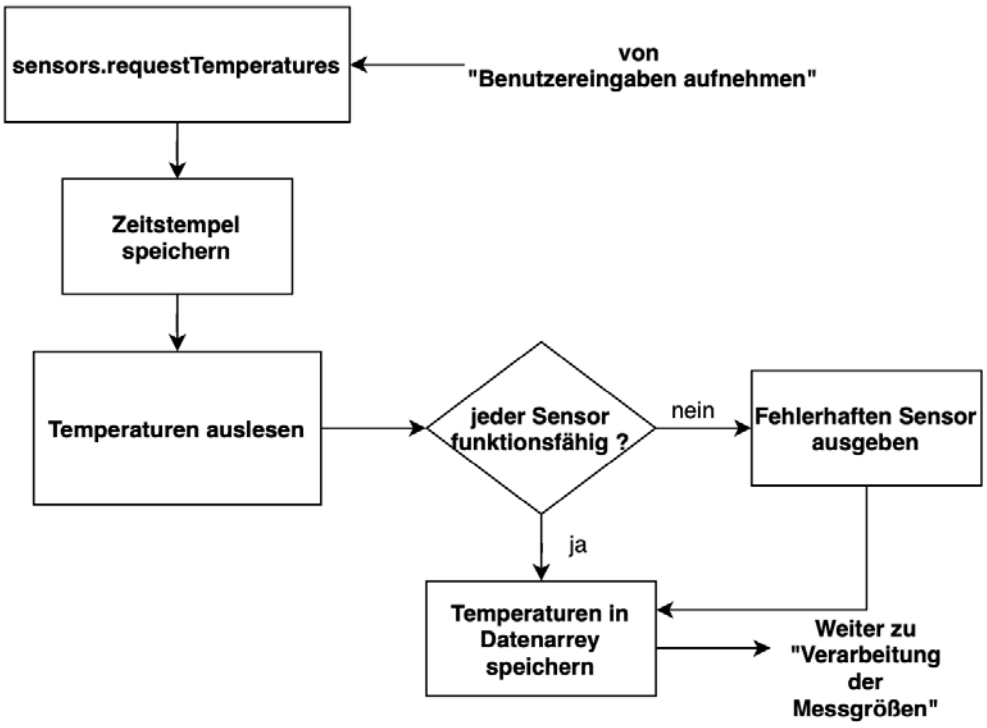


Abbildung 3.19.: Darstellung des Programmabschnittes der Aufnahme der Messgrößen

Nach Aufnahme der Temperaturdaten eines Messarreys werden diese für die anschließende Regelung verarbeitet.

3.5.5. Verarbeitung der Messgrößen

Die Datenarreys mit aufgenommenen Temperaturen werden für die Regelung verarbeitet. Dabei werden zwei Funktionen verwendet: $findHighest()$ und $TempAnpassen()$. Die Aufgabe von $findHighest()$ ist es, die höchste Temperatur der Paneeloberfläche zu finden. Anhand dieser werden Entscheidungen bei der Spannungsregelung getroffen. Die Funktion $TempAnpassen()$ ist mit der Korrektur des Messfehlers beauftragt. Dabei gilt es die gemessenen Temperaturen so anzupassen, dass die Funktion $T_{sens}(p)$ die Funktion $T_{cam}(p)$ annähert. Die Messkurven weisen einen linearen Verlauf auf. Daher wird für die Korrektur des Messfehlers ebenfalls ein linearer Ansatz verfolgt. Ziel ist es, eine Ausgleichsfunktion $T_{an} = f(T_{sens}(p))$ zu generieren, dessen Ergebnis als Offset zu den gemessenen Werten addiert wird. Dafür wird zunächst eine lineare Funktion anhand der Messabweichung bei $p = 0.2535$, welches dem ersten aufgenommenen Wert mit der kleinsten Messabweichung entspricht, und $p = 0{,}9412$, an dem sich die maximal einstellbare Panleeltemperatur von $85\,^{\circ}\mathrm{C}$ einstellt, aufgestellt. So ergibt sich die Korrekturformel zu:

$$T_{an} = T_{sens}(p) \cdot 0.828 - 16.881 + T_{sens}(p) \tag{3.12}$$

So ergibt sich die folgende angepasste Temperaturkurve:

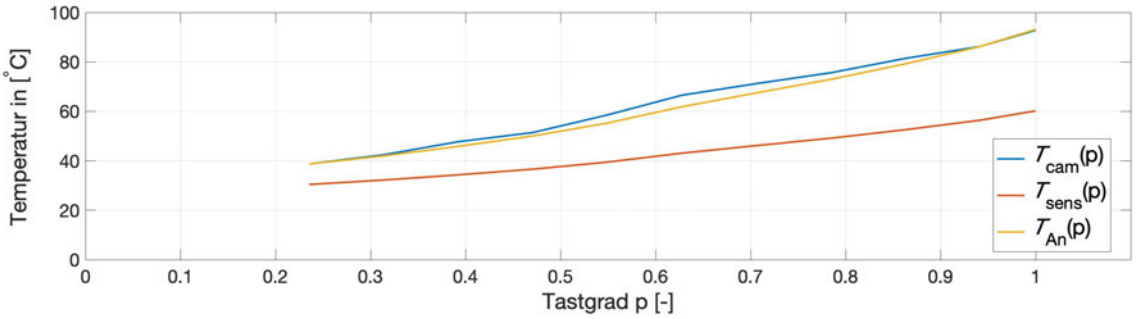


Abbildung 3.20.: Angepasste Temperatur T_{an} im Vergleich zu $T_{cam}(p)$ und $T_{sens}(p)$

Der Ablauf des Programmschrittes lässt sich in dem folgenden Flussdiagramm wiederfinden:

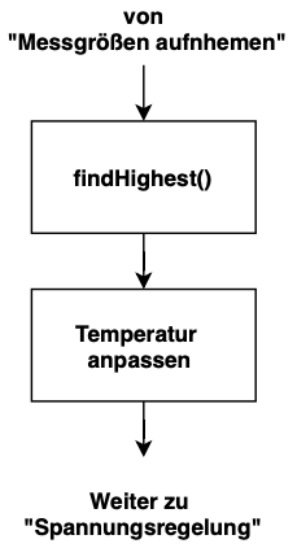


Abbildung 3.21.: Darstellung des Programmabschnittes der Verarbeitung der Messgrößen

Anhand der Ergebnisse der Verarbeitung der Messgrößen, wird die Regelung der Spannung durch die MOSFET-Verstärkerschaltung vorgenommen. Diese Ansteuerung wird in dem folgenden Programmschritt erläutert.

3.5.6. Adaptive Spannungsregelung per PWM-Signal

Das Paneel besteht aus Sicht der Temperaturregelung aus zwei wesentlichen Komponenten mit unterschiedlicher Dynamik: Zum einen die Rovings und zum anderen das sie umgebende Paneelmaterial. Werden die Rovings nun mit einem mittleren Heizstrom von $I_P = p \cdot \hat{I}_P$ beaufschlagt, erwärmen sich diese schneller als das Paneelmaterial. Während der Heizphase, in der das Paneel schnellstmöglich auf eine Zieltemperatur gebracht wird, ist die Heizrate der Rovings größer zu erwarten, also die der Paneeloberfläche. Da die Temperaturbelastbarkeit aller Paneelbestandteile eine weitaus höheren Wert aufweisen als der obere Grenzwert des Temperatursensormessbereiches, wird eine maximale Rovingtemperatur von $T_{F_{max}} =$

$100\,^{\circ}\mathrm{C}$ festgelegt. Wie in Kapitel 1 beschrieben entspricht dies der maximal zugelassenen Betriebstemperatur verbauter Rovings im System Flugzeug. Durch die Überwachung der Rovingtemperatur soll die Heizrate adaptiv an die vorherrschende Prozessdynamik angepasst werden, damit das Paneel schnellstmöglich seinen eingestellten Zielwert erreicht.

Der mittlere Heizstrom I_P wird indirekt über die Anpassung des Gleichanteil des Leistungs-PWM-Signals geregelt. Dabei weist das PWM-Signal eine Auflösung von 2 Byte auf. Demnach lässt sich die Spannung des Netzteiles von $\hat{U}_P$ in einem Bereich von $0\,\mathrm{V} - \hat{U}_P$ in 255 Schritten einteilen. Dies ergibt bei einer Spannung von $\hat{U}_P = 12\,\mathrm{V}$ eine Auflösung von:

$$\frac{12\,\mathrm{V}}{255\,Schritte} \approx 47{,}06\,\frac{\mathrm{mV}}{Schritt}$$

Durch den Befehl $analogWrite(PWM - Pin, *Value from\,0 - 255*)$ (siehe Abbildung 3.22) wird ein PWM-Signal erzeugt. Vorausgesetzt, es soll bei $\hat{I_P} = 12\,\mathrm{V}$ ein Gleichanteil von $U_P = 6\,\mathrm{V}$ erzeugt werden, benötigt man einen Tastgrad von $p = 0{,}5$. Der Befehl für ein solches PWM-Signal würde folglicherweise $analogWrite(PWM - Pin$,127) lauten.

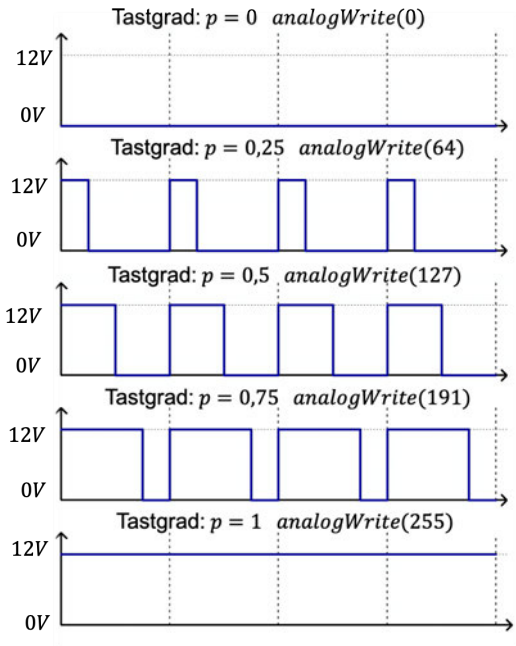


Abbildung 3.22.: Aufrufbefehle für Leistungs-PWM-Signale am Paneel

So fließt ein mittlerer Heizstrom von :

$$I_P = \frac{U_P}{R_P} = 1{,}68\,\mathrm{A} \tag{3.13}$$

Der Widerstand der Rovings kann dabei nach [11] mit steigender Temperatur als konstant angenommen werden.

Während der Heizphase soll das Paneel in möglichst kurzer Zeit auf seine Zieltemperatur erhitzt werden. Dies wird erreicht, indem das Paneel direkt mit dem maximalen Heizstrom von $\hat{I}_P$ durchflossen wird. Wird eine Rovingtemperatur von $T_{F_{max}} = 100\,^{\circ}\mathrm{C}$ erreicht, wird der Heizstrom auf ein absolutes Minimum von $I_P = 0\,\mathrm{A}$ reduziert. Erst nach Abfall der Temperatur auf $T_F = 99{,}5\,^{\circ}\mathrm{C}$ wird wieder der maximale Heizstrom durch die Rovings geleitet. So wird eine Schalthysterese von $0{,}5\,^{\circ}\mathrm{C}$ vorgesehen, die eine Verminderung der Ein- und Ausschaltvorgänge mit sich bringt. Aufgrund der festgestellten unzureichend genauen Messergebnisse, die bei der Messung der Rovingtemperatur auftreten, muss für die Vorgabe der schnellstmöglichen Erwärmung eine andere Methode entwickelt werden, die ohne Messung der Rovingtemperatur zu einem ähnlichen Ergebnis führt. Eine Korrektur der gemessenen Temperatur durch einen Algorithmus kommt dabei nicht in Frage, da der

Kontakt zwischen Sensor und Roving als zu fehlerbehaftet angesehen wird. Aus diesem Grund wird während der Heizphase anstatt eines variablem ein statischer Tastgrad umgesetzt. Dabei ist ein Tastgrad zu finden, der $I_{F_{max}} = 100\,^{\circ}\mathrm{C}$ mit sich führt. Dieses Verhalten ist bei einem eingestellten Tastgrad von $p = 0{,}75$ beobachten worden. So wird ersatzweise dieser Wert konstant für die Heizphase eingestellt. Aus diesem Grund wird dieser Wert auch für die Haltephase als maximaler Tastgrad verwendet.

Die Heizphase wird abgeschlossen, wenn die eingestellte Zieltemperatur auf dem Paneel durch die angepassten Sensorwerte erkannt wird. Anschließend wird in die Haltephase übergegangen. Um diese effizient zu gestalten, werden für das Halten der Zieltemperatur drei Temperaturbereiche definiert. Die Tastgrade wurden dabei nach den sich einstellenden Temperaturen nach der Funktion $T_{cam}(p)$ ausgewählt und mit einem Puffer versehen. Hier wird ebenfalls eine Schalthysterese von $0{,}5\,^{\circ}\mathrm{C}$ vorgesehen.

Temperaturbereich in $[^{\circ}\mathrm{C}]$	gewählter Tastgrad p	Erwartete Paneeltemperatur nach $T_{cam}(p)$ in $[^{\circ}\mathrm{C}]$
$25 \leq T_{Ziel} < 40$	0,3921	47,7
$40 \leq T_{Ziel} < 60$	0,6275	61,83
$60 \leq T_{Ziel} \leq 85$	0,75	71,2

Tabelle 3.3.: Temperaturbereiche für eine effizient gestaltete Haltephase

Die Anforderung, die maximale Rovingtemperatur $T_{F_{max}} = 100\,^{\circ}\mathrm{C}$ nicht zu überschreiten, bringt eine Einbuße bezüglich der vom Benutzer eingegebenen Prozessgröße mit sich. Die mit dem Tastgrad $0{,}75$ erreichte Rovingtemperatur erwärmt die Oberfläche des Paneels nach $T_{cam}(p)$, siehe Abbildung 3.12, selbst nach einer Wartezeit von $5\,\mathrm{min}$ nur auf $71{,}2^{\circ}\mathrm{C}$. Es ist zu erwarten, dass die Paneeltemperatur sich nach einer Zeitdauer von $t \gg 5\,\mathrm{min}$ der Führungsgröße $I_{F_{max}}$ annähert. Für den Systementwurf wird für den weiteren Verlauf die Einhaltung der Rovingtemperatur höher priorisiert, als das Erreichen der maximal einstellbaren Paneeltemperatur. Der Vorgang der Spannungsregelung ist dem folgenden Flussdiagramm in Abbildung 3.23 zu entnehmen.

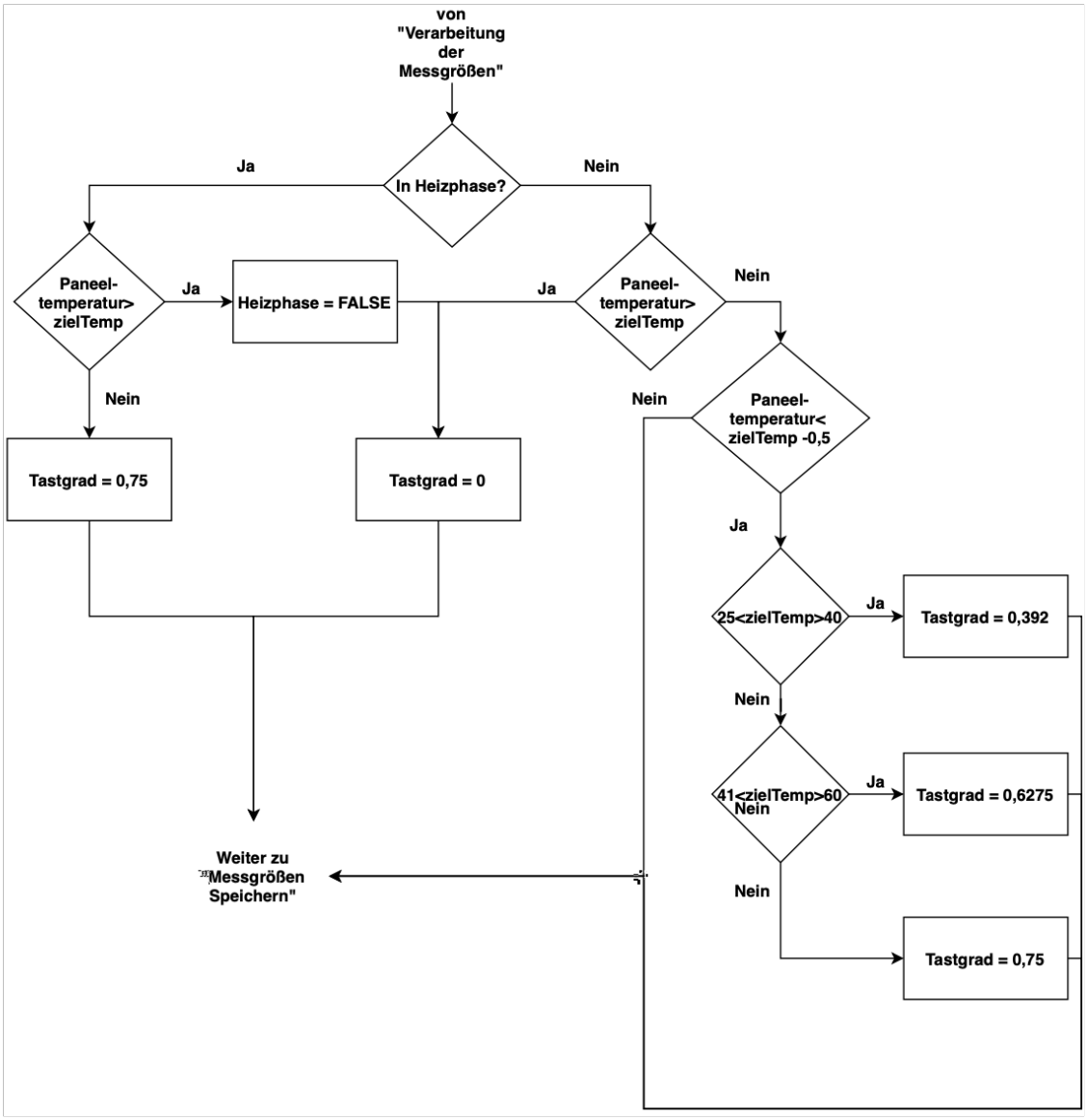


Abbildung 3.23.: Darstellung des Programmabschnittes der Spannungsregelung

Anschießend werden die aufgenommenen Messgrößen für die spätere Auswertung auf der SD-Karte des Speichermoduls abgelegt.

3.5.7. Messgrößen speichern

Der letzte Programmschritt umfasst die Speicherung der gesammelten Prozessdaten. Dieser wird jeweils nach dem Schritt der Spannungsregelung ausgeführt (siehe Abbildung 3.14). Dabei gilt es, die Daten der Messarreys mit je 15 gespeicherten Temperatur-$float$-Elementen, den Temperaturen, und den Zeitstempel auf der SD-Karte zu speichern. Um die Daten zu organisieren werden sie in bestimmter Reihenfolge in die Datei $\mathrm{Daten.txt}$ geschrieben:

- Daten.txt: Messarrey1[15], Messarrey2[15], Messarrey3[15], Zeitstempel

Für das Speichern der Daten wird zunächst die entsprechende Datei mit dem Befehl $SD.open()$ zum Beschreiben geöffnet. So kann mit der Funktion $print()$ die zu speichernden Daten übergeben werden. Als nächstes wird die fertig beschriebene Datei durch $close()$ geschlossen und der Schritt ist abgeschlossen. Anschließend wird wieder mit der Aufnahme von Messgrößen begonnen. Der Zyklus wiederholt sich dabei endlos, bis der Benutzer die Recheneinheit von seiner Versorgungsspannung trennt. Der Vorgang der Datenspeicherung ist dem folgenden Flussdiagramm zu entnehmen.

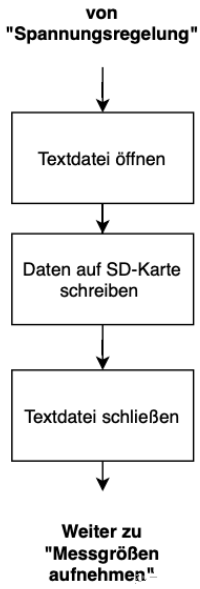


Abbildung 3.24.: Darstellung des Programmabschnittes der Speicherung von Messgrößen

Nach der Erläuterung der definierten Schritte und damit auch Fertigstellung der Regelung, ist die Entwicklung der Domäne Informationstechnik abgeschlossen. Nun wird zur Systemintegration übergegangen, in der die gefertigten Komponenten zu einem System zusammengefügt werden.

3.6. Systeminteragtion

Nach Abschluss der domänenspezifischen Entwicklungen, wird das System aus den Teilkomponenten zusammengefügt. Dabei findet als grundlegende Vorlage die Schnittstellendefinition aus Abschnitt 3.2 Verwendung. Aufgrund des in der vorangegangenen Phase der domänenspezifischen Entwicklung erfolgten Funktionsabgleiches der Komponenten ist bei der Zusammenführung Module, der Systemintegration, der Aufwand deutlich geringer. Teile des Systems wurden bereits getestet, ohne dass der Rest des Systems bereits umgesetzt war. So wurden auch geplante Komponenten und Messungen aus dem Systemaufbau entfernt, wie etwa die Strommessung und die Messung der Rovingtemperatur. Für den Wegfall wurden jedoch alternative Lösungen entwickelt.

Nach abgeschlossener Integration wird das System anhand eines Glasfaserpaneels getestet und die gespeicherten Messgrößen ausgewertet. Der Versuchsaufbau und die Versuchsbedingungen werden dargestellt. Anschließend werden die Ergebnisse diskutiert und ein Anforderungsabgleich erstellt. Das Ergebnis dieses Abgleiches ist ein Indikator dafür, wie das Ergebnis der Systementwicklung zu beurteilen ist.

4. Versuchsdurchführung und Regelungsverifikation

Dieses Kapitel umfasst die Verifikation des Regelungssystemes. Das System wird anhand eines bereitgestellten Glasfaserpaneel verifiziert. Dafür wird zunächst der Gesamtaufbau des Systemes, wie er aus der Systemintegration hervorgegangen ist, dargestellt. Nach der Erläuterung des Versuchsaufbaus und der Versuchsbedingungen werden die Ergebnisse eines Heizprozesses hinsichtlich der Regelungsfunktion analysiert. Der abschließende Anforderungsabgleich nach Tabelle 2.1 und die Beurteilung des Systementwurfes schließen das Kapitel ab.

4.1. Versuchsaufbau und Versuchsbedingungen

Für den Versuchsaufbau wird das System (siehe Kapitel 3) durch das Glasfaserpaneels ergänzt. Zunächst wird für die Oberfläche des Paneels ein Koordinatensystem definiert. Der Ursprung wird auf die Mitte der unteren Paneelkante zwischen den beiden inneren Rovings gelegt. So befinden sich die Sensoren $T18$, $T28$ und $T38$ auf der Y-Achse des Paneels. Die Abstände der Arrays von der X-Achse können der Prinzipskizze in Abbildung 4.1 entnommen werden. Die Positionen der Sensoren in X-Richtung sind hingegen durch Maße der Sensorhalterung aus Unterpunkt 3.5 vorgegeben. Das Paneel wird so an die Verstärkerschaltung angeschlossen, dass der Fluss des mittleren Heizstromes I_P entgegen der Y-Achse verläuft. Die Messarrays werden durch Verschraubungen fest auf der Oberfläche des Paneels fixiert und auf die Oberfläche gepresst.

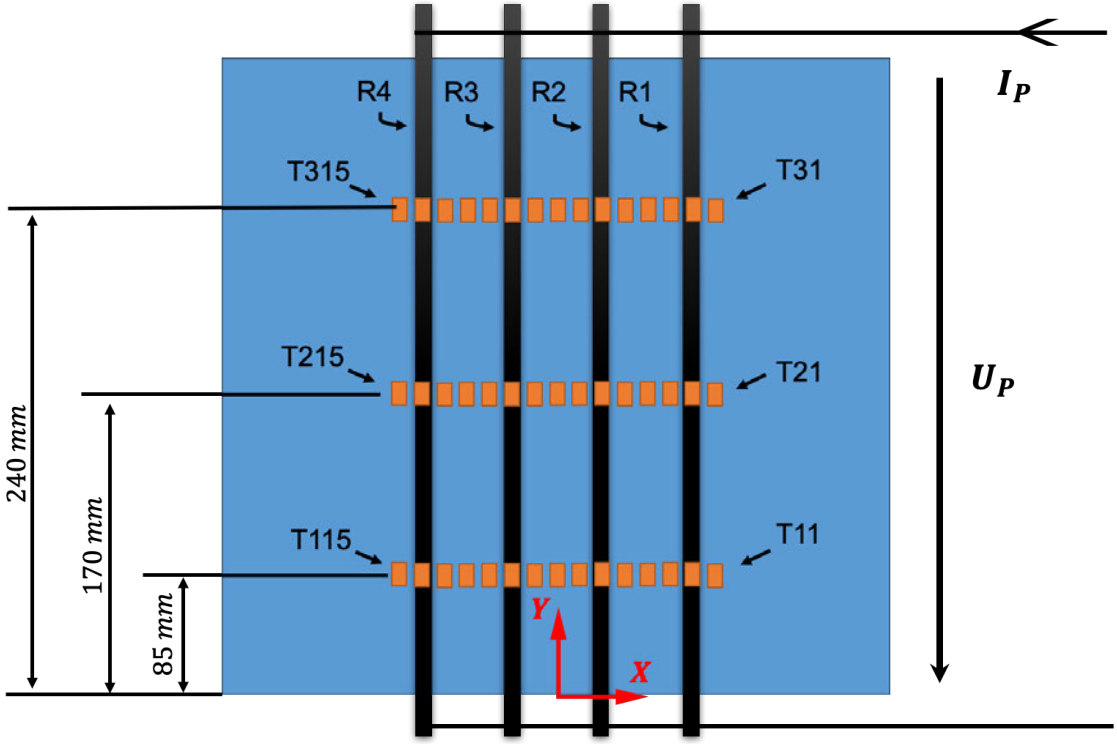


Abbildung 4.1.: Positionierung der Messarrays auf dem Paneel

Nach Anschluss der Kohlefaserrovings des Versuchspaneels an die Verstärkerschaltung ist der Gesamtsystemaufbau komplett und bereit für die Test- und Verifikationsphase. Dabei wird die Verbindung zwischen den Rovings und dem System mit Hilfe von Abgreifklemmen hergestellt. Das Modul mit der Verstärkerschaltung und dem Sternpunkt des 1-Wire Busses stellt beim Versuchsaufbau den Mittelpunkt des Systemes dar. Über diese Hauptplatine werden außerdem alle Sensoren und das Speichermodul an die nötigen $5\,\mathrm{V}$-Versorgungsspannung angeschlossen. Der Aufbau des Versuches erfolgt nach Abbildung 4.2.

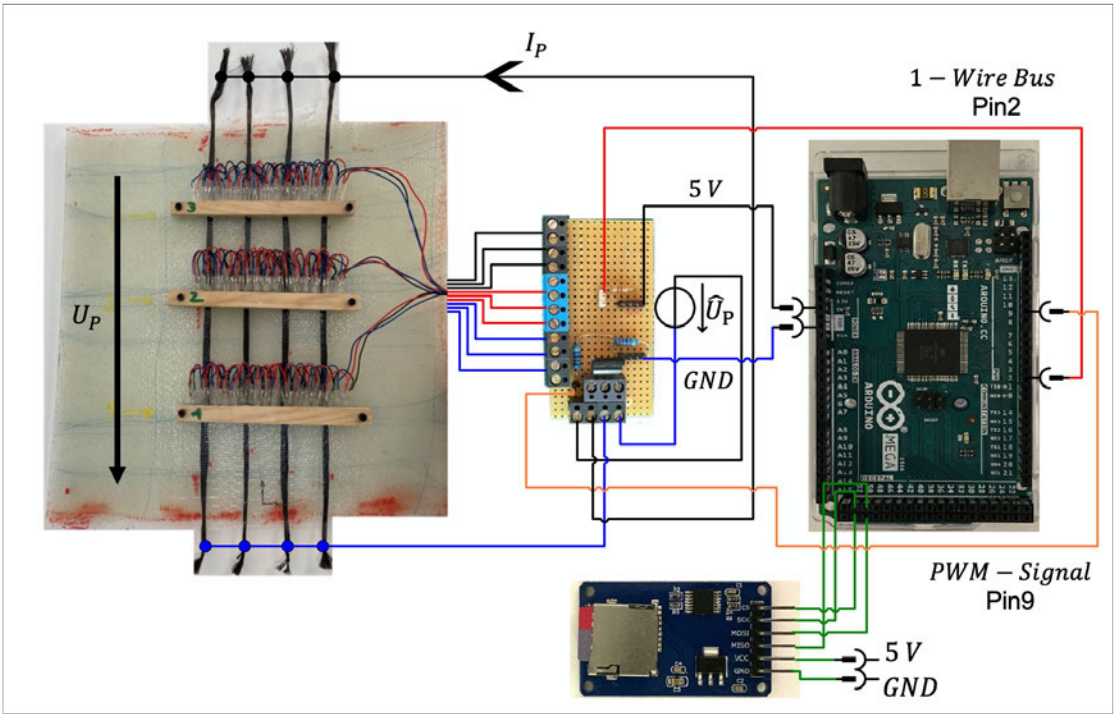


Abbildung 4.2.: Das Gesamtsystem mit allen verwendeten Komponenten

Nach dem Aufbau kann das System getestet werden. Für die Auswertung werden aufgezeichnete Daten eines Regelungsprozesses, der über einen Zeitraum von $21\,\mathrm{min}$ lief, ausgewertet. Dabei wurde eine Zieltemperatur von $50\,^{\circ}\mathrm{C}$, bei einer Sensorauflösung von $12Bit$ vorgegeben. So stellt sich nach Programmierung eine Abtastrate von $f_T = 0{,}33\,\mathrm{Hz}$ ein. Die Daten werden in dem folgenden Kapitel visualisiert und ausgewertet. Dabei steht die Verifikation des Systemes im Vordergrund. Es werden verschiedene Darstellungen der Oberflächentemperatur aufgezeigt und wichtige Indikatoren beschrieben.

4.2. Verifikation der Temperaturreglung

Zuerst wird in diesem Kapitel die korrekte Funktion der Regelung nachgewiesen. Alle getroffenen Aussagen stützen sich dabei auf die korrigierte Temperatur der Messarrays T_{M1}, T_{M2} und T_{M3}, da diese die reale Oberflächentemperatur annähern (siehe Unterabschnitt 3.5.5.). Die Regelung soll die Paneeloberfläche auf eine eingestellte Temperatur, hier

$T_{Ziel} = 50\,°\mathrm{C}$, erhitzen und diese Temperatur halten. Die Programmierung orientiert an der höchsten gemessenen Temperatur auf der Oberfläche. Es wird erwartet, dass die maximale Oberflächentemperatur nach erstmaligem Erreichen der Zieltemperatur und dem anschließenden Übergang in die Haltephase zunächst wieder auf einen Messwert von $T_{Hys} = 44{,}5\,°\mathrm{C}$ sinkt. Dies entspricht der programmierten Schalthysterese von $\Delta\mathrm{T} = 0{,}5\,°\mathrm{C}$. Anschließend wird die Temperatur wieder auf $50\,°\mathrm{C}$ ansteigen und so über die Dauer des folgenden Prozesses um den Temperaturwert von $49{,}75\,°\mathrm{C}$ oszillieren. Die Abbildung 4.3 zeigt die Messergebnisse der höchst gemessenen Temperaturen $T_{M1,max}$, $T_{M2,max}$ und $T_{M3,max}$ der Messarrays über die Prozesszeit auf. Dabei werden sowohl die real gemessenen Temperaturwerte $T_{M1,real,max}$, $T_{M2,real,max}$ und $T_{M3,real,max}$, als auch die durch die Korrekturformel generierten Temperaturen dargestellt.

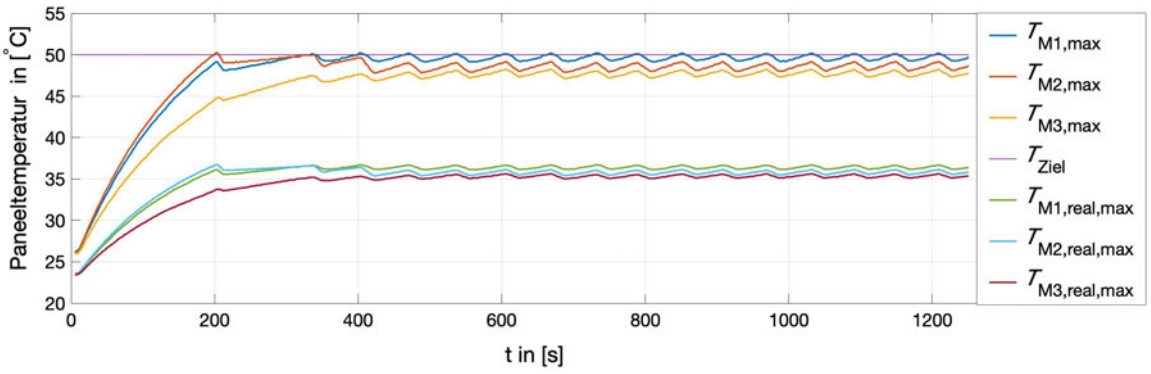


Abbildung 4.3.: Höchste Paneeltemperatur je Abtastpunkt im Laufe eines Regelungsprozesses

Nach Start des Prozesses befindet sich die Regelung in der Heizphase. Es ist unmittelbar eine Temperaturerhöhung zu erkennen. Dabei ist zu beobachten, dass das Messarray 2 über die gesamte Heizphase die höchste Temperatur erfasst und somit zunächst den Prozess führt. Messarray 1 misst über die Heizphase etwa $1\,°\mathrm{C}$ weniger als Messarray 2. Messarray 3 hingegen nähert sich nach Abschluss der Heizphase lediglich $T_{M3,max} = 45\,°\mathrm{C}$ an. Das Paneel erreicht, detektiert durch das zweite Messarray, nach etwa drei Minuten zum ersten mal die Zieltemperatur. Das Regelungsprogramm geht anschließend in die Haltephase über. Die Oberflächentemperatur unterschreitet die untere Grenze der Schalthysterese von $T_{Hys} = 49{,}5\,°\mathrm{C}$ um $\Delta\mathrm{T} = 0{,}46\,°\mathrm{C}$ und steigt anschließend wieder an. Erklärt werden kann diese Abweichung durch die bereits erkannte Messträgheit der Temperatursensoren, die eine Wärmeänderung verzögert wahrnehmen. Dem könnte durch eine Verkleinerung der

Schalthysterese entgegengewirkt werden. Ebenfalls ist es wahrscheinlich, dass das Paneel verzögert auf einen Anstieg des Heizstromes reagiert. Nach etwa zwei Minuten wird die Zieltemperatur erneut erreicht. Dabei detektiert ab diesem Zeitpunkt nicht mehr Messarray 2 die höchste Temperatur, sondern Messarray 1. Messarray 1 übernimmt somit von nun an die Führung des Prozesses. Nach zwei Haltephase-Schaltperioden, ab $t = 473{,}3\,\mathrm{s}$, scheint sich das System in einem eingeschwungenen Zustand zu befinden. Es lässt sich erkennen, dass die Temperatur ab diesem Zeitpunkt zunächst auf etwa $T_{M1} = 49{,}16\,^{\circ}\mathrm{C}$ abfällt. Dieser Wert unterschreitet den Erwartungswert T_{Hys} um $0{,}34\,^{\circ}\mathrm{C}$. Anschließend werden die Rovings wieder beheizt, bis die Zieltemperatur erneut erreicht wird. Dabei überschreitet die maximale korrigierte Temperatur die Zieltemperatur nur um maximal $0{,}19\,^{\circ}\mathrm{C}$. Das Überschwingen der gemessenen Temperatur lässt sich ebenfalls durch die Messträgheit der Temperatursensoren erklären. Die Funktion der entwickelten Regelung ist somit nachgewiesen.

Bei der Auswertung fällt zusätzlich auf, dass der maximale korrigierte Temperaturwert des ersten Messarrays im eingeschwungenen Zustand um etwa $1{,}03\,^{\circ}\mathrm{C}$ höher als der des zweiten Messarrays ist. Außerdem ist der maximale korrigierte Temperaturwert des zweiten Messarrays erneut etwa $1{,}16\,^{\circ}\mathrm{C}$ höher als der des dritten Messarrays. Es ist also ein Temperaturabfall der Oberflächentemperatur entgegen der $Y - Achse$ längs der eingebetteten Rovings zu erkennen. Dies entspricht einem Temperaturabfall zwischen Messarray 1 und 2 von $\Delta T = 0{,}012\,\frac{^{\circ}\mathrm{C}}{\mathrm{mm}}$, zwischen Messarray 2 und 3 hingegen $\Delta T = 0{,}017\,\frac{^{\circ}\mathrm{C}}{\mathrm{mm}}$. Die Temperatur scheint entlang der technischen Stromrichtung zuzunehmen, entlang der physikalischen Stromrichtung, der Strömungsrichtung der Elektronen in einem Leiter [27], welche entlang der $Y - Achse$ verläuft, hingegen abzunehmen.

Als nächstes wird die Entwicklung der aufgenommenen Oberflächentemperatur in Abbildung 4.4, Abbildung 4.5 und Abbildung 4.6 entlang der $X - Achse$ und über der Prozesszeit visualisiert. Es ist zu erwarten, dass die Bereiche über den Rovings durch höhere Temperaturen deutlicher hervorstechen als die Zwischenbereiche. Dabei ist durch die gemessenen Widerstandswerte nach Tabelle 3.2 eine bestimmte Reihenfolge bezüglich der Oberflächentemperatur zu erwarten. Den Roving mit dem niedrigsten Widerstandswert durchfließt nach Knotenregel der höchste Anteil des mittleren Heizstromes I_P. Es ist somit zu erwarten, dass der Roving R_{F_4} die höchste Temperatur annimmt, gefolgt von R_{F_3} und R_{F_1}. Der Roving R_{F_2} besitzt nach Messung den höchsten Widerstandswert und dürfte sich somit am geringsten erhitzen.

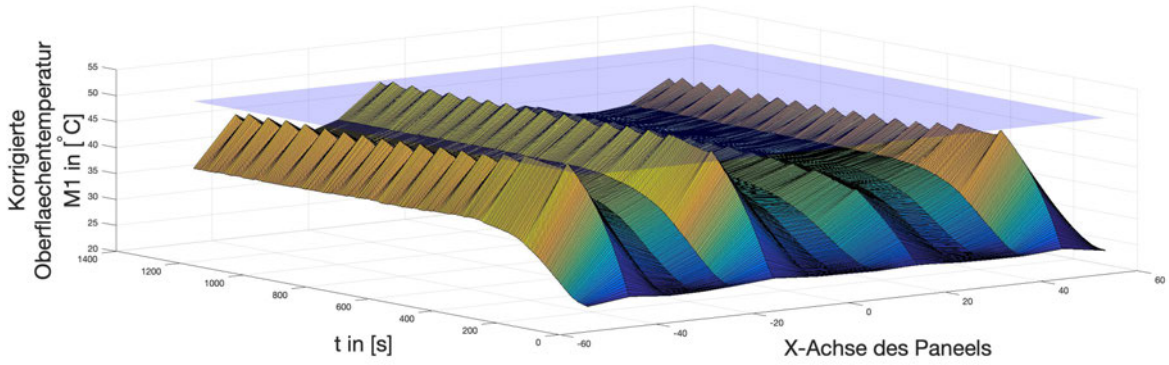


Abbildung 4.4.: Korrigierte Oberflächentemperatur im Laufe eines Regelungsprozesses mit eingezeichneter Zieltemperaturebene, T_{M1}

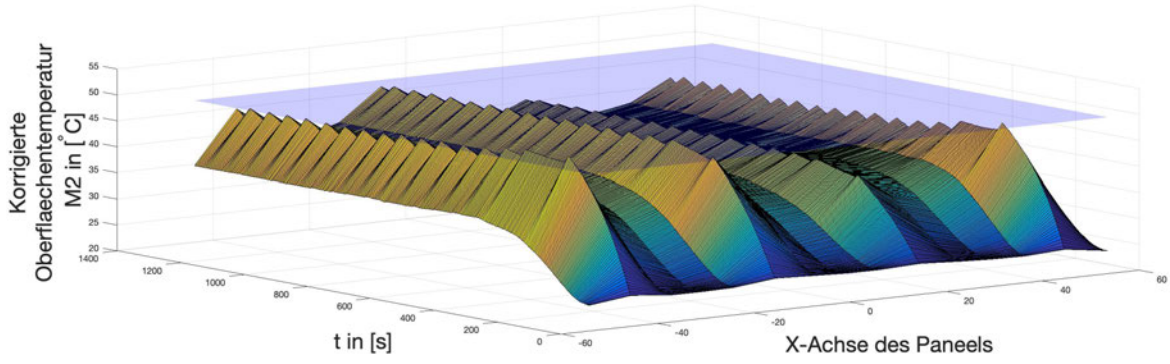


Abbildung 4.5.: Korrigierte Oberflächentemperatur im Laufe eines Regelungsprozesses mit eingezeichneter Zieltemperaturebene, T_{M2}

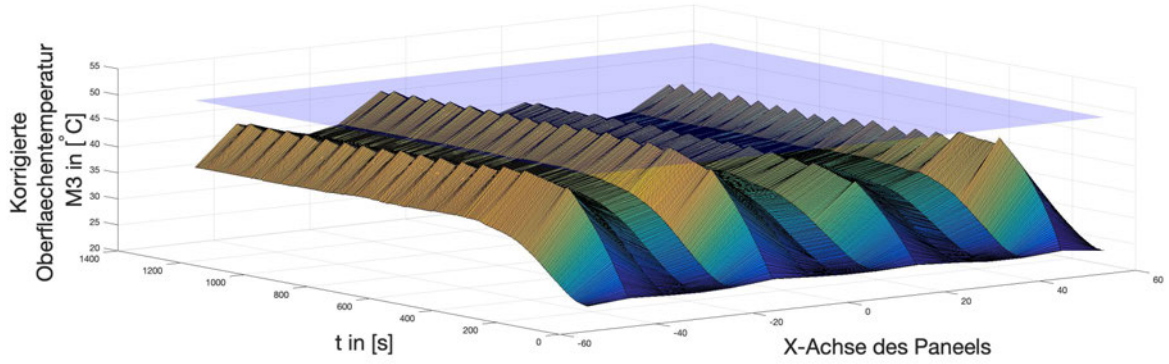


Abbildung 4.6.: Korrigierte Oberflächentemperatur im Laufe eines Regelungsprozesses mit eingezeichneter Zieltemperaturebene, T_{M3}

Die Auswertung der Messergebnisse ergibt eine andere Reihenfolge der vier Rovings bezüglich der Höhe der gemessenen Temperaturen. Es ist zu erkennen, dass Roving R_{F_3} im eingeschwungenen Zustand den höchsten Temperaturmesswert annimmt, nicht Roving R_{F_4}. Eine wahrscheinliche Ursache dafür ist, dass der Anschluss der Rovings zum Zeitpunkt des Versuches nicht reproduziert werden konnte und so andere, rückwirkend nicht quantifizierbare Widerstandswerte vorliegen. Die beobachtete wärmeabhängige Widerstandsverringerung könnte hier ebenfalls einen Einfluss haben. In den Grafiken ist der Übergang von Heiz- zur Haltephase deutlich zu erkennen, ebenso wie die programmierte Schalthysterese, die sich durch Zacken in Temperaturverläufen der Haltephase manifestiert (siehe Abbildung 4.4 - Abbildung 4.6). Für die Darstellungen der gemessenen Temperaturverteilung durch die Messarrays 1-3 werden die Messwerte der Oberflächentemperaturen entlang der $X - Achse$ zunächst auf den höchsten Messwert je Abtastzeitpunkt nach $T_{rel,Mi} = \frac{T_{Mi}}{T_{Mi,max}}$ normiert. So lässt sich der prozentuale Anteil des Temperaturunterschiedes und die relative Temperaturverteilung je Messzeitpunkt aufzeigen (siehe Abbildung 4.7 bis Abbildung 4.12). Dafür werden für jedes Array vier ausgewählte Zeitpunkte ausgewertet: Diese sind zunächst zwei Messpunkte während der Heizphase nach einer Prozesszeit von $101{,}2\,\mathrm{s}$ und $200{,}2\,\mathrm{s}$, ein weiterer nach $401{,}3\,\mathrm{s}$ und einen letzten Messpunkt im eingeschwungenen Zustand des Systemes nach $1001\,\mathrm{s}$. Diese Werte werden ebenfalls grafisch dargestellt. Die Ergebnisse werden dabei nur kurz umrissen.

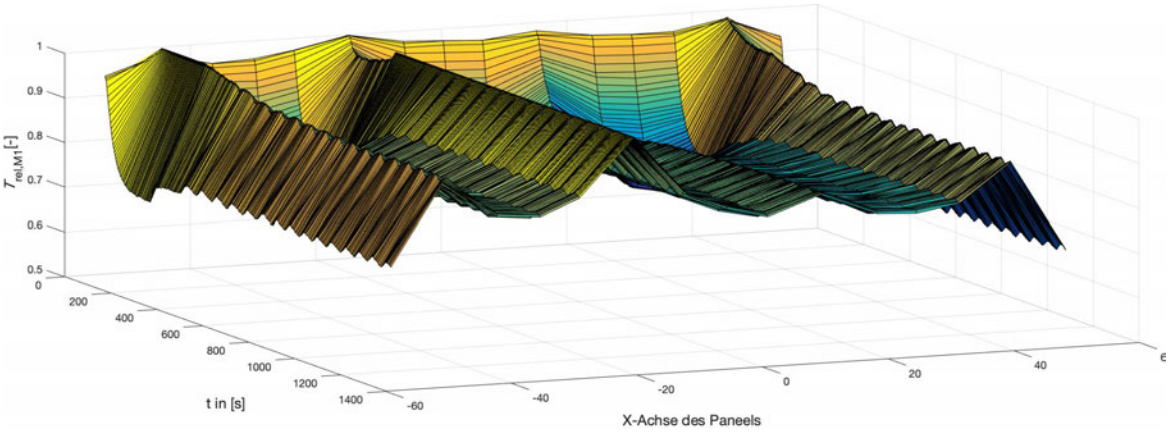


Abbildung 4.7.: Normierte Oberflächentemperatur im Laufe eines Regelungsprozesses, Messarray 1

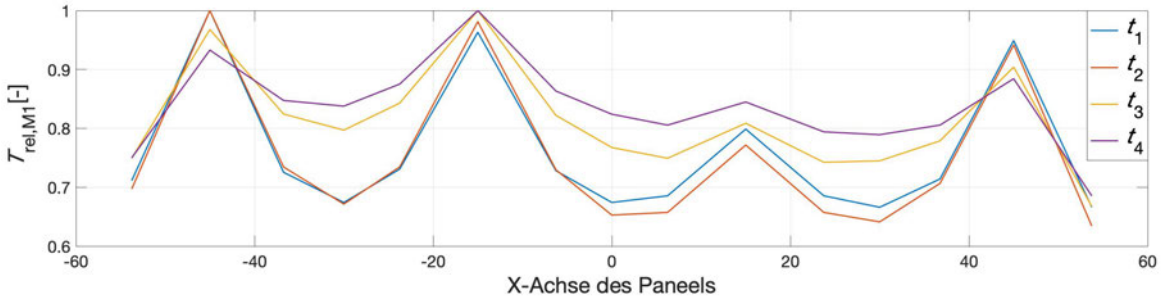


Abbildung 4.8.: Normierte Oberflächentemperatur im Laufe eines Regelungsprozesses zu ausgewählten Zeitpunkten, Messarray 1

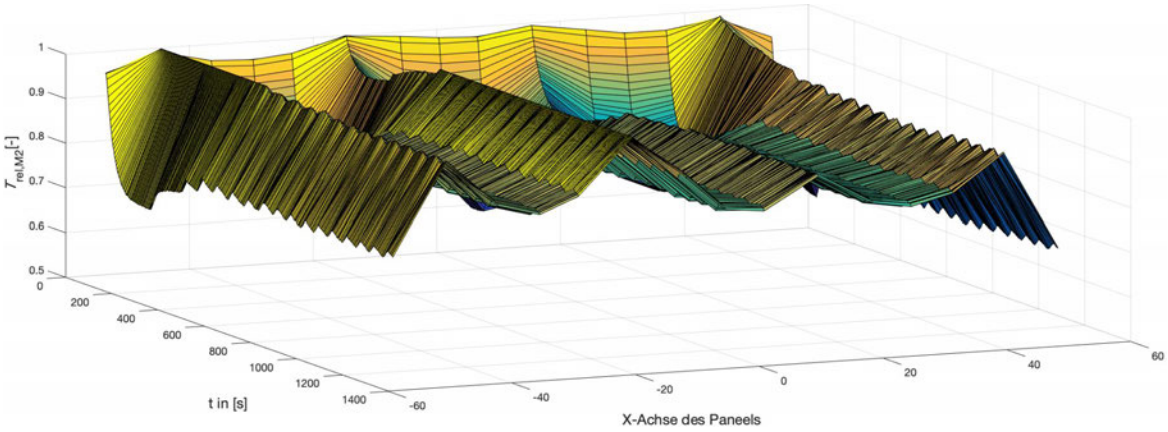


Abbildung 4.9.: Normierte Oberflächentemperatur im Laufe eines Regelungsprozesses, Messarray 2

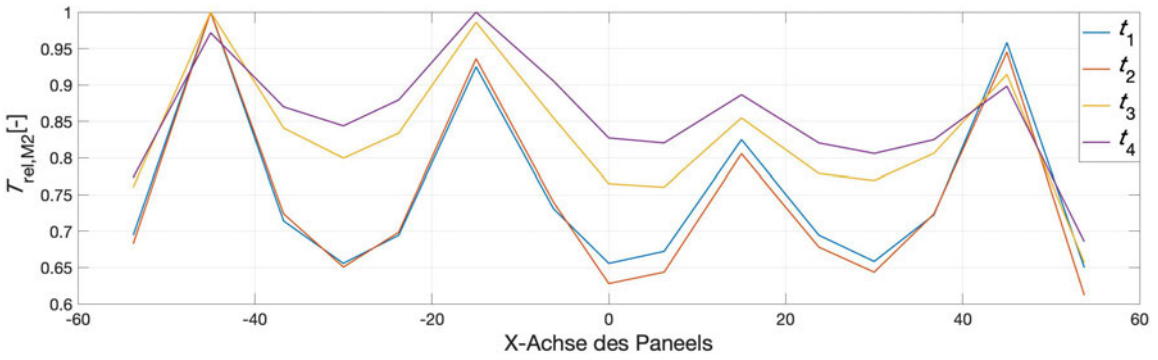


Abbildung 4.10.: Normierte Oberflächentemperatur im Laufe eines Regelungsprozesses zu ausgewählten Zeitpunkten, Messarray 2

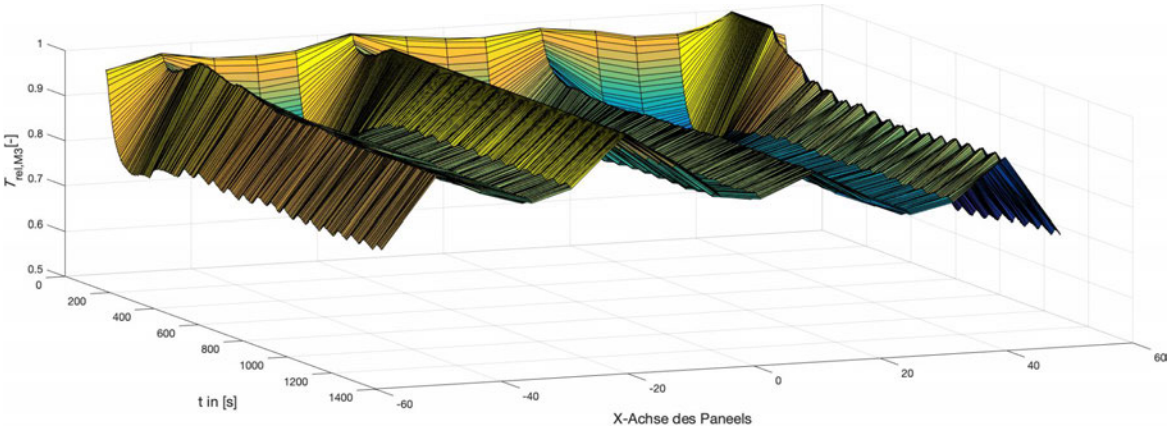


Abbildung 4.11.: Normierte Oberflächentemperatur im Laufe eines Regelungsprozesses, Messarray 3

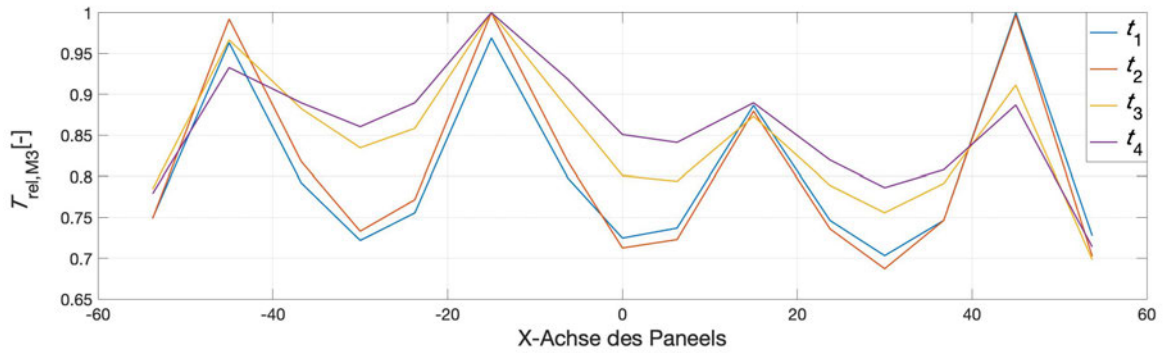


Abbildung 4.12.: Normierte Oberflächentemperatur im Laufe eines Regelungsprozesses zu ausgewählten Zeitpunkten, Messarray 3

Die Grafiken geben einen guten Überblick darüber, wie die Oberflächentemperatur des Paneels sich entlang der X-Achse verteilt. Die Abtastung der Oberflächentemperatur durch die ausgewählte Anzahl von Temperatursensoren entlang der X-Achse liefern aussagekräftige Messergebnisse, die Interpretationen zur Temperaturverteilung zulassen. Dabei ist eine Abtastung der Bereiche zwischen den Rovings durch drei Sensoren ausreichend. Interessant wäre eine Abtastung der Temperatur entlang der gesamten X-Achse des Paneels. So könnte anhand der Außenbereiche, die an die äußeren Rovings grenzen, untersucht werden, bis zu welchem Abstand von einem beheizten Roving eine Oberflächentemperaturerhöhung erfasst wird. Der Aufbau der Messarrays in einer Sterntopologie lässt diese Erweiterung zu. Auffällig ist, dass während der Heizphase Messarray 1 und Messarray 2 die höchste Temperatur über Roving R_{F_4} erkennen, Messarray 3 über R_{F_1}. Im eingeschwungenen Zustand jedoch detektieren alle Arrays die Temperatur über dem Roving R_{F_3} als die höchste Temperatur, die Temperatur über dem Roving R_{F_2} als die niedrigste. Die Messarrays zeigen hier relative Temperaturwerte in einem Intervall von $T_{rel} = [0{,}845; 0{,}89]$ im Vergleich zur Messung über dem Roving R_{F_3}. Aufgrund der widerstandsabhängigen Erwärmung und den so auftretenden Temperaturunterschieden wäre es denkbar, den Systemaufbau umzugestalten. Im aktuellen System wird T_{Ziel} in der Haltephase lediglich über R_{F_3} erreicht. Um diese Temperatur über jedem Roving zu erreichen, wäre es denkbar, von der aktuellen Parallelschaltung der Rovings abzuweichen und den Heizstrom für jeden Roving einzeln zu regeln. So könnte sowohl der prozentuale Temperaturunterschied verringert als auch die gleichmäßige Temperaturverteilung der Rovingzwischenräume hergestellt werden.

Anschließend wird für den aufgezeichneten Prozess eine fallspezifische Energiebilanz aufgestellt.

4.3. Aufstellen einer fallspezifischen Energiebilanz

Im Folgenden wird die durch das Paneel aufgenommene elektrische Energie auf Basis einer Messreihe bestimmt. Diese wird im System von den Rovings in Joulesche Wärme umgewandelt. Aufgrund der fehlenden Messung des realen Heizstromes wird unter den möglichen Varianten auf eine graphische Analyse zurückgegriffen. Anhand der gemessenen Temperatur werden Zeitbereiche Δt_i definiert, in denen der Heizstrom fließt. Das System startet in der Heizphase und verlässt diese nach Erreichen der eingestellten Zieltemperatur,

in diesem Fall $T_{Ziel} = 50\,°\mathrm{C}$. So kann grafisch der Zeitbereich der Heizphase bestimmt werden. Gleiches gilt für die anschließend folgende Haltephase. Hier lassen sich ebenfalls relevante Bereiche definieren, in denen ein Heizstrom fließt, sobald die untere Grenze der Schalthysterese von $T_{Hys} = 49{,}5\,°\mathrm{C}$ erkannt wird. Der Stromfluss stoppt zu dem Zeitpunkt, wenn die eingestellte Zieltemperatur erneut erreicht wird. Die Bereiche sind in Abbildung 4.13 markiert. Für die Auswertung werden die ersten acht Regelungsbereiche betrachtet.

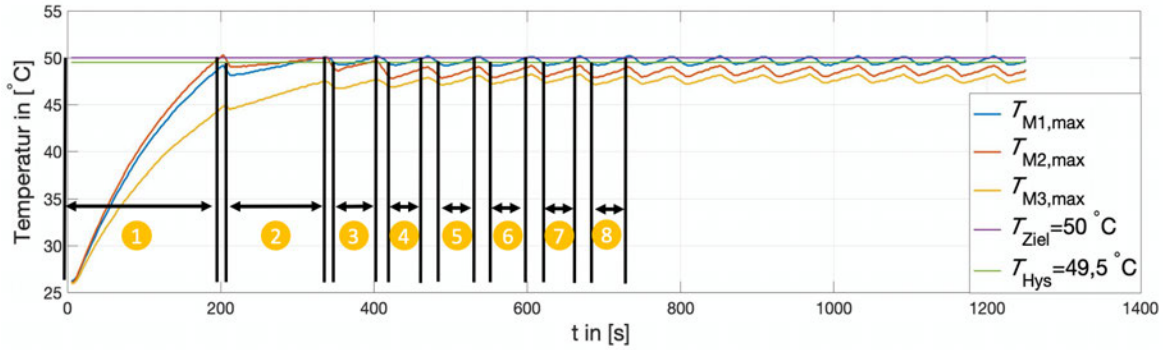


Abbildung 4.13.: Heizbereiche eines Regelungsprozesses

Für jeden der acht Heizbereiche ist dabei der Tastgrad p bekannt. Während der Heizphase wird ein konstanter Tastgrad von $p = 0{,}75$ vorgegeben. Während der Haltephase geht mit der vorgegeben Zieltemperatur $T_{Ziel} = 50\,°\mathrm{C}$ ein Tastgrad von $p = 0{,}6275$ einher. Anschließend lässt sich unter Verwendung des ermittelten Gesamtwiderstandes des Paneels R_P die aufgewendete elektrische Energie für jeden Bereich E_{1-8} berechnen. Dafür wird die Verwendung einer Quellspannung von $\hat{U}_P = 13{,}86\,\mathrm{V}$ beachtet (siehe Unterabschnitt 3.4.3.).

$$E_i = \frac{U_P^2}{R_P} \cdot \Delta t_i = \frac{(\hat{U}_P \cdot p)^2}{R_P} \cdot \Delta t_i \tag{4.1}$$

Bereich	Zeitdauer Δt_i in [s]	Phase	Elektrische Energie E_i in [Ws = J]
1	200,2	Heizphase	6057,9
2	129,1	Haltephase	2734,6
3	54	Haltephase	1143,8
4	48	Haltephase	1016,7
5	48	Haltephase	1016,7
6	52,1	Haltephase	1103,6
7	45	Haltephase	953,2
8	45	Haltephase	953,2

Tabelle 4.1.: Zeitdauer und Energieaufwand der gekennzeichneten Bereiche in Abbildung 4.13

Die Auswertung nach Tabelle 4.1 zeigt, dass dem Paneel eine Energie von $6057{,}9\,\mathrm{J}$ zugeführt werden muss, um die Oberfläche von seinem Startwert von $T_0 = 25{,}97\,^\circ\mathrm{C}$ um $\Delta T = 24{,}03\,^\circ\mathrm{C}$ zu erhitzen, sodass die Zieltemperatur $T_{Ziel} = 50\,^\circ\mathrm{C}$ erreicht wird. In der anschließenden ersten Haltephase mit einem verminderten Tastgrad werden $2734{,}6\,\mathrm{J}$ aufgebracht, um das Paneel um $\Delta T = 1{,}88\,^\circ\mathrm{C}$ auf die eingestellte Zieltemperatur zu erwärmen. Es fällt auf, dass der Energieverbrauch dieses Abschnittes etwa die Hälfte der verbrauchten Energie während der Haltephase beträgt. In den folgenden Halteabschnitten $3-8$ werden für je ein $\Delta T \approx 1\,^\circ\mathrm{C}$ Energie in einem Bereich von $E_i = [953{,}2\,\mathrm{J}; 1143{,}8\,\mathrm{J}]$ aufgebracht. Dabei ist zu beobachten, dass sich die Energie sich mit laufender Prozesszeit verringert. In dem Gesamtzeitraum der acht Bereiche von etwa $12\,\mathrm{min}$ ist so eine Energie von $E_{ges} = 14{,}98\,\mathrm{kJ}$ verbraucht worden. Dabei ist zu erkennen, dass die Zeitdauer Δt_i in der Haltephase mit zunehmender Prozesszeit t abnimmt. Dies ist ein Indikator dafür, dass die Oberfläche des Paneels mit der Zeit eine höhere Dynamik aufweist. Mit laufender Prozessdauer erwärmt sich das gesamte Paneel zunehmend, sodass der Wärmefluss von den Rovings in das Paneel abnimmt. Zur Effizienzsteigerung des Prozesses wäre ein optimierter Übergang von Heizphase in Haltephase denkbar. Außerdem könnte in der Haltephase ein höherer Tastgrad als im betrachteten System vorgegeben werden, um sowohl die Einschwingdauer als auch die Heizzeit Δt_i in Halteperioden zu verkürzen. Jedoch würde sich so durch den quadratischen Einfluss der Spannung U_P die elektrische Leistung stark erhöhen. Zudem besteht die Möglichkeit, durch eine Verminderung der Schalthysterese die Abkühlung der Rovings zu vermindern, sodass in einer Halteperiode eine niedrigere Temperaturdifferenz zur Zieltemperatur überwunden werden müsste. So

würden kürzere Heizzeiten Δt_i erzeugt werden, jedoch wäre ein Anstieg der Anzahl an Halteperioden die Folge. In weiteren Untersuchungen müsste durch Variation des Tastgrades und der Schalthysterese eine geeignete Kombination gefunden werden, die energetische und zeitliche Optimierung vereint. Diese Optimierung könnte durch die Integration einer adaptiven Energiereglung erfolgen. Abschließend wird das entwickelte System in einem Anforderungsabgleich beurteilt und bewertet.

4.4. Anforderungsabgleich

Der Anforderungsabgleich findet nach den gesammelten Anforderungen, dargestellt in Tabelle 2.1, statt. Es wird kritisch beurteilt, welche Anforderungen als erfüllt oder nicht erfüllt angesehen werden.

- Anforderung 1: Das Paneel wird wie gefordert auf eine Zieltemperatur erhitzt und anschließend auf dieser gehalten (siehe Abbildung 4.3.). Dabei ist jedoch zu betrachten, dass die Temperatur lediglich durch Korrekturmaßnahmen der Messwerte erkannt wurde. Dennoch ist festgestellt, dass das System die gewünschte Funktionalität aufweist. Somit gilt die Anforderung als erfüllt.

- Anforderung 2: Die Anforderung, die Heizrate während der Heizphase adaptiv zu gestalten, ist als nicht erfüllt zu werten. Es wurde lediglich eine alternative Methodik erarbeitet, den Tastgrad hart vorzugeben. Für eine adaptive Regelung in geplanter Konfiguration müsste die Rovingtemperatur zuverlässig aufgenommen werden.

- Anforderung 3: Die Anforderung, eine Rovingtemperatur von $T_{F_{max}}$, welche gleichzeitig als Information für die adaptive Heizregelung dienen sollte, nicht zu überschreiten, konnte nicht umgesetzt werden. Aufgrund der beobachteten fehlerhaften Temperaturmessung der Rovingtemperatur durch die Temperatursensoren konnte diese Messgröße nicht als Eingangswert für die Regelung genutzt werden. Daher ist die Anforderung als nicht erfüllt anzusehen.

- Anforderung 4: Das System liest nach seinem Start die gewünschten Prozessparameter des Benutzers ein und überprüft diese auf seine Gültigkeit. Sollten die Parameter

nicht der definierten Gültigkeit entsprechen, wird es dem Benutzer mitgeteilt und das Regelungsprogramm startet nicht. Die Anforderung ist somit erfüllt.

- Anforderung 5: Die diskrete Temperaturaufnahme erfolgt, vorgegeben durch das Design der Sensorhalterung, in den Bereichen direkt oberhalb der Rovings, also auch in den Zwischenbereichen. So lässt sich die Oberflächentemperaturverteilung durch insgesamt 15 diskrete Temperaturmessungen pro Array entlang der X-Achse beurteilen. Die Anforderung gilt als erfüllt.

- Anforderung 6: Das System soll die Fähigkeit aufweisen, Prozessdaten zu speichern. Dabei sind explizit Temperaturen und ein Strom/Zeit Verlauf zu nennen. Die Oberflächentemperaturen der drei Messarrays werden zusammen mit einem Zeitstempel erfolgreich gespeichert. Die Anforderung, einen Heizstrom/Zeit Verlauf zu speichern, ist aufgrund beschriebener Problematik bezüglich der Strommessung nicht gegeben. Eine Methode, wie diese Strommessung durchführt werden kann, wurde jedoch dargestellt.

- Anforderung 7: Aufgrund der Verwendung von drei Messarrays entlang der Y-Achse wird die Anforderung als erfüllt betrachtet.

- Anforderung 8: Durch die verwendete Sterntopologie weist der Datenbus eine ausreichende Voraussetzung bezüglich einer Modularität auf. Der Anschluss des Paneels und des Netzteiles an die Hauptplatine erfolgt durch Anschlussklemmen. So lässt sich das System für die Lagerung zerlegen. Der MOSFET ist ebenfalls nicht an der Platine verlötet, sondern durch Anschlussklemmen in das System integriert. So ist ein eventueller Austausch leicht möglich. Die Anforderung gilt als erfüllt.

- Anforderung 9: Die letzte Anforderung legt die zulässige Toleranz der Temperaturmessung der Sensoren fest. Für die Aufnahme der Oberflächentemperatur beträgt diese nach der $\pm 2\,\%$ Vorgabe der maximal zu messenden Temperatur von $85\,^{\circ}\mathrm{C}$ eine Toleranz von $\pm 1{,}7\,^{\circ}\mathrm{C}$. Diese Toleranz wurde im Zuge einer Verifikation durch Wärmezufuhr eines Temperaturprüfschrankes nachgewiesen. In der geplanten Messkonfiguration wird diese Toleranz gehalten. Die Anforderung ist nicht erfüllt.

Die Anforderungsanalyse zeigt, dass vier der neun definierten Anforderungen nicht erfüllt wurden. Nach der Vorgabe aus Abschnitt 2.2. handelt es sich dabei um harte Anforderungen. Das System muss in weiteren Schritten überarbeitet werden, um die zwei wesentlichen

Schwachstellen der Entwicklungsergebnisses zu beheben. Dabei handelt es sich zum einen um eine geeignetere Auswahl von Temperatursensoren, die die diskrete Abtastung der Oberflächentemperatur nach gestellten Anforderungen erfüllen. Zum anderen muss eine Implementierung der Strommessung vorgenommen werden. Die aufgebrachte elektrische Energie könnte so auf dem Arduino berechnet werden und in die Regelung einfließen. So könnte man zusätzlich zu einer adaptiven Heizrate eine adaptive Energieregelung implementieren, die das System in einen optimalen energetisch Zustand bringt. Das folgende und letzte Kapitel fasst das Ergebnis der Systementwicklung und wichtige Erkenntnisse des Systemverhaltens zusammen.

5. Resümee

5.1. Erlangte Erkenntnisse

In diesem Kapitel werden die wichtigsten Ergebnisse aus der vorliegenden Systementwicklung und abgeleitete Schlussfolgerungen über das Systemverhalten dargestellt. Als Ergebnis des Entwicklungsprozesses geht ein System hervor, bei dem eine zuverlässige Aufheizung des Paneels auf eine vom Benutzer festlegte Temperatur nachgewiesen werden konnte. Die Temperatursensoren zeigten in der Verifikationsphase eine deutliche Messdifferenz zwischen gemessener und realer Temperatur der Paneeloberfläche. Diese Feststellung wird auf den nicht idealen Kontakt zwischen Sensorik und Substrat zurückgeführt. Zudem wird angenommen, dass eine Verfälschung des Messergebnisses durch einen nicht ausreichenden Wärmestrom zwischen Roving und Sensor vorliegt. So wird die Wärmekapazität des Sensors nicht vollständig gefüllt. Auch ein konvektiver Wärmestrom an die Umgebung begünstigt diesen Messfehler. Jedoch konnte mithilfe einer linearen Anpassung eine zufriedenstellende Korrektur der Messwerte erreicht werden. Die zuverlässige Aufzeichnung der gemessenen Systemgrößen lässt im Anschluss einer Messung eine externe Verarbeitung zu. Anhand dieser Auswertungen konnten wichtige Beobachtungen bezüglich des Systemverhaltens gesammelt werden. Es lässt sich feststellten, dass die ausgewählte Konfiguration der Temperatursensoren in drei Messarrays und die Abtastung der Temperatur durch 15 Sensoren entlang der X-Achse des Paneels Messergebnisse liefert, anhand derer sich Aussagen über die Temperaturverteilung in unterschiedlichen Stadien der Reglung treffen lassen. Durch die Temperaturaufnahme entlang der Rovings ist eine Temperaturzunahme von der Oberfläche entlang der physikalischen Stromrichtung festgestellt worden, welche sich in einem Bereich von $\Delta T = [0{,}012\,\frac{^\circ\mathrm{C}}{\mathrm{mm}}; 0{,}017\,\frac{^\circ\mathrm{C}}{\mathrm{mm}}]$ befanden. Diese Abweichung konnte im Laufe dieser Arbeit nicht erklärt werden. Zudem wurde mit

steigender Temperatur eine Widerstandsabnahme der Rovings beobachtet. Diese liegt in der Temperaturspanne von Raumtemperatur bis $100\,^{\circ}\mathrm{C}$ bei etwa 10%.

Es ist ebenfalls erkannt worden, dass die Parallelschaltung der Rovings zu einer unterschiedlichen Erhitzung der Bereiche über und zwischen den Rovings geführt hat. Der Grund dafür wird auf die variierenden Wiederstandwerte der Rovings und die damit einhergehende unterschiedliche Erhitzung zurückgeführt. Aus diesem Grund wird die Empfehlung einer separaten Ansteuerung der Rovings empfohlen.

Zusätzlich zeigt die fallspezifische Energiebilanz auf, dass der höchste Energieverbrauch während der Heizphase auftritt. In der anschließend folgenden Haltephase sinkt der Energieverbrauch mit steigender Prozesszeit. Diese Abnahme wird auf den Effekt zurückgeführt, dass die Oberfläche des Paneels mit steigender Prozesszeit eine höhere Dynamik aufweist. Gleichzeitig wird dargelegt, dass mit einer Variation des Tatgrades und der Schalthysterese der Prozess zeitlich und energetisch optimiert werden kann. Es wird vorgeschlagen, dies von einer adaptiven Energieregelung in der Haltephase vornehmen zu lassen.

5.2. Ausblick

Basierend auf den vorliegenden Schlussfolgerungen werden nachfolgend mögliche Anknüpfungspunkte für weitere Arbeiten dargestellt. Dabei wird der Fokus auf das Optimierungspotential des Systementwurfs gelegt. Während der Entwicklung und der Auswertung des Systementwurfes sind charakteristische Erkenntnisse bezüglich des Verhaltens des Versuchspaneels erlangt worden, die im Rahmen dieser Arbeit nicht erklärt werden konnten. Zunächst ist der beobachtete Wärmeabfall der Rovings entlang des physikalischen Stromflusses zu nennen. Dieses Verhalten gilt es weiter zu untersuchen, gerade im Hinblick darauf, dass im Versuchspaneel Rovings mit der Länge $40\,\mathrm{cm}$ verwendet wurden und bereits hier signifikante Messunterschiede festgestellt wurden. Eingesetzt im Zielsystem Flugzeug mit Verwendung von größeren Rovinglängen gilt es dieser Charakteristik Beachtung zu schenken.

Darüber hinaus ist die beobachtete wärmeabhängige Widerstandsänderung zu untersuchen und zu betrachten, ob diese Änderung eventuell einer Linearität folgt. Es lässt sich

vermuten, dass Auffälligkeiten während der Heizphase und der ersten Haltephaseperiode auf diese Änderung zurückzuführen sind.Damit ist die Beobachtung gemeint, dass trotz der Feststellung der Temperaturabnahme entlang der Y-Achse Messarray 2 eine höhere Temperatur detektiert hat als Messaray 1, im eingeschwungenen Zustand jedoch Messarray 1 durchgehend etwa $1\,^{\circ}\mathrm{C}$ mehr gemessen hat. Hier müssten zur Klärung ebenfalls weiterführende Forschungen durchgeführt werden. Um die Erfüllung der vier verfehlten Anforderungen in weiteren aktualisierten Entwicklungen sicherzustellen, werden essenziell nötige Veränderung im Folgenden herausgearbeitet und alternative Methoden vorgestellt. Zur Erfüllung der Anforderung 6 müsste eine Strom-Auswerteroutine in das Regelungsprogramm nach Erläuterung im Unterabschnitt 3.5.4 implementiert werden. Anschließend könnte ein Heizstrom/Zeit Diagramm erstellt werden. Außerdem ist erkannt worden, dass sich das Verfehlen der Anforderungen 2,3 und 9 auf die unzureichend genaue Temperaturmessung der Oberfläche des Paneels und der Rovings selber zurückführen lässt. Da das Technologiegebiet der Temperaturmessung eine breite Auswahlmöglichkeit bietet, gilt es eine Messmethode zu finden, welche die genannten Problematiken des DS18B20 in umgesetzter Konfiguration vermeidet und somit den festgestellten Messfehler umgeht. Zunächst sollte bei einer Neuauswahl der Temperatursensorik der Fokus auf einer niedrigen Wärmekapazität des Sensorkörpers liegen. Dies ist gegeben, wenn der Sensor einen möglichst kleinen Messkörper aufweist [28]. So wird die vom Sensor gespeicherte Wärme verringert und die Messdynamik könnte deutlich erhöht werden. Beispielhaft wäre dabei der PTFC PT100-Messwiderstand der Firma TE Connectivity [29] zu nennen. Dieser weist neben einer hohen Messgenauigkeit von $\pm 0{,}43\,^{\circ}\mathrm{C}$ ebenfalls eine geringe Sensorfläche von $4{,}6\,\mathrm{mm}^2$ auf. Zum Vergleich : Die aufliegende Sensorfläche des DS18B20 liegt bei $16\,\mathrm{mm}^2$. Die Verwendung der analogen Temperaturmessung würde jedoch mit einem erhöhten Hardwareaufwand einhergehen, wie in Abschnitt 2.9 beschrieben. Es ist jedoch durchaus denkbar, dass die Verwendung der vorgeschlagenen Sensorik für eine nach Anforderung 9 ausreichend genaue Temperaturmessung sorgt. Eine weitere Alternative wäre dadurch gegeben, eine rein bildgebende Auswertung der Temperaturen durch Wärmebildmessung zu verwenden. Diese könnte auf ausgewählte relevante Bereiche des Paneels fokussiert werden und Messergebnisse an eine Regeleinheit weitergeben. Diese Lösungsvariante würde den geringsten Hardwareaufwand aufweisen.

Außerdem zeigt die Auswertung der fallspezifischen Energiebilanz ein deutliches Potential hinsichtlich der zeitlichen und energetischen Optimierung von Heizvorgängen. Durch die

Stellschrauben des Tastgrades und der Schalthysterese und einer zusätzlichen nachträglichen Integration einer adaptiven Heizrate kann der Prozess in weiteren Untersuchungen optimiert werden.

Zusammenfassend zeigt sich, dass die vorliegende Systementwicklung trotz Mängel bezüglich der Anforderungserfüllung Messergebnisse erzeugt, die eine Charakterisierung der Paneeloberfläche bezüglich der Oberflächentemperaturverteilung zulassen und so dabei helfen Kenntnislücken zu schließen. Für einen voll funktionsfähigen Zustand sind jedoch weiterführende Entwicklungsarbeiten nötig.

Literaturverzeichnis

[1] Maxim Integrated. Ds18b20 datasheet. Online, abgerufen am 02.02.2020: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.

[2] Peter K. C. Rudolph and Dezso Georgefalvy. Anti-icing system for aircraft, 1990. US Patent 5114100A.

[3] Verein Deutscher Ingenieure (VDI). *Entwicklungsmethodik für mechatronische Systeme (VDI 2206): Design methodology for mechatronic systems*. VDI, 2004.

[4] Halbleitertechnologie von A bis Z. Grundlagen: Dotieren: n- und p-Halbleiter. Online, abgerufen am 31.01.2020: www.halbleiter.org/grundlagen/dotieren.

[5] P. Schnabel. *Elektronik-Fibel: Elektronik-Grundlagen, Messtechnik, Bauelemente, Schaltungstechnik, Digitaltechnik*. Schnabel, 2007.

[6] Maxim Integrated. Guidelines for reliable long line 1-wire networks. Online, abgerufen am 31.01.2020: https://www.maximintegrated.com/en/design/technical-documents/tutorials/1/148.html.

[7] Natalia Petrovova. Rekordzahl von über 1,1 Milliarden beförderter Fluggäste im Jahr 2018. Eurostat Pressestelle, 2019. Online, abgerufen am 10.01.2020.

[8] The National Transportation Safety Board. Aircraft accident report, 1994.

[9] Thomas P. Ratvasky, Billy P. Barnhart, and Sam Lee. Current methods modeling and simulating icing effects on aircraft performance, stability, control. *Journal of Aircraft*, 47(1):201–211, 2010.

[10] Maximilian Schutzeichel and Peter Linde. Beheizbare Vorderkantenvorrichtung, Vorderkantenheizsystem und Luftfahrzeug damit. Patent : Aktenzeichen DE: 10 2018 004 814.5, 19.06.2018.

[11] MOH Schutzeichel, T Kletschkowski, P Linde, and LE Asp. Experimental characterization of multifunctional polymer electrolyte coated carbon fibres. *Functional Composites and Structures*, 1(2):025001, 2019.

[12] Karlheinz Roth. *Konstruieren mit Konstruktionskatalogen*, chapter Methodisches Ermitteln der Funktionen und der Funktionsstrukturen, pages 81–106. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[13] Bo Qi, Zhengkai Yuan, Shaorong Lu, Kuo Liu, Shanrong Li, Liping Yang, and Jinhong Yu. Mechanical and thermal properties of epoxy composites containing graphene oxide and liquid crystalline epoxy. *Fibers and Polymers*, 15:326–333, 02 2014.

[14] S. Leijonmarck, T. Carlson, G. Lindbergh, L.E. Asp, H. Maples, and A. Bismarck. Solid polymer electrolyte-coated carbon fibres for structural and novel micro batteries. *Composites Science and Technology*, 89:149 – 157, 2013.

[15] Wilfried Plaßmann and Detlef Schulz. *Handbuch Elektrotechnik: Grundlagen und Anwendungen für Elektrotechniker*. Springer Fachmedien Wiesbaden, 2013.

[16] Arduino. Atmega2560 - tech specs. Online, abgerufen am 19.02.2020: https://store.arduino.cc/arduino-mega-2560-rev3.

[17] Allegro. Fully integrated, hall effect-based linear current sensor with 2.1 kvrms voltage isolation and a low-resistance current conductor. Online, abgerufen am 29.02.2020: https://www.alldatasheet.com/datasheet-pdf/pdf/168326/ALLEGRO/ACS712.html.

[18] Mikrocontroller.net. Pulsweitenmodulation. Online, abgerufen am 23.01.2020: https://www.mikrocontroller.net/articles/Pulsweitenmodulation.

[19] Dieter Zastrow. *Elektronik - Lehr- und Arbeitsbuch*, chapter Gleichrichtung, pages 174–196. Vieweg+Teubner Verlag, Wiesbaden, 1988.

[20] Kim Sorensen, Andreas Helland, and Tor Johansen. Carbon nanomaterial-based wing temperature control system for in-flight anti-icing and de-icing of unmanned aerial vehicles. *IEEE Aerospace Conference Proceedings*, 2015, 06 2015.

[21] International Rectifier. Irfz44n hexfet® power mosfet. Online, abgerufen am 02.02.2020: https://pdf1.alldatasheet.com/datasheet-pdf/view/68619/IRF/IRFZ44N.html.

[22] Stephan Messlinger. Zur Temperaturmessung mit Platin-Widerstandsthermometern und prema 5017 dmm. Uni Bayreuth. Online, abgerufen am 27.01.2020: https://epub.uni-bayreuth.de/43/1/tmeas-dossier.pdf.

[23] Arduino.cc. Spi library. Online, abgerufen am 06.02.2020: https://www.arduino.cc/en/Reference/SPI.

[24] Peter Niemz. Untersuchungen zur Wärmeleitfähigkeit ausgewählter einheimischer und fremdländischer Holzarten. *Bauphysik*, 29(4):311–312, 2007.

[25] Nikolaus Hannoschöck. *Wärmeleitung und -transport - Grundlagen der Wärme- und Stoffübertragung*. Springer-Verlag, Berlin Heidelberg New York, 2018.

[26] Vötsch. Vötsch - temperature test chambers. online, abgerufen am 11.03.2020.

[27] Stefan Rinner. Elektrische Netzwerke. In *Physik für Wirtschaftsingenieure*, pages 179–191. Springer, 2018.

[28] Robert Bauer, Markus Gölles, Thomas Brunner, Nicolaos Dourdoumas, and Ingwald Obernberger. Was messen Temperatursensoren in einer Biomasse-Feuerung wirklich?(what is really measured by temperature sensors in a biomass furnace?). *at-Automatisierungstechnik*, 55(12/2007):600–607, 2007.

[29] SENSOR SOLUTIONS. Pt temperature sensor – ptf family. Online, abgerufen am 31.03.2020: https://docs.rs-online.com/c289/0900766b81554053.pdf.

A. Anhang

A.1. Bedienungsanleitung

Die folgende Beschreibung erläutert den Umgang mit dem entwickelten System. Für die Erzeugung von Messdaten und zum Start des Prozesses sind die folgenden zum Teil mitgelieferten Komponenten notwendig:

1. Computer mit Arduinosoftware Version 1.8.10
2. Arduino Mega mit einem USB-Verbindungskabel
3. Systemplatine mit Verstärkerschaltung und Sternpunktanschluss des Datenbusses
4. Drei Messarrays mit je 15 Tempertursensoren DS18B20, eingefügt in die Sensorhalterung
5. Speichereinheit mit Micro-SD Karte und Übertragungsadapter
6. Netzteil mit $\hat{U}_P = 13{,}86\,\mathrm{V}$

Zunächst wird das System nach dem Folgenden Schaltplan aufgebaut:

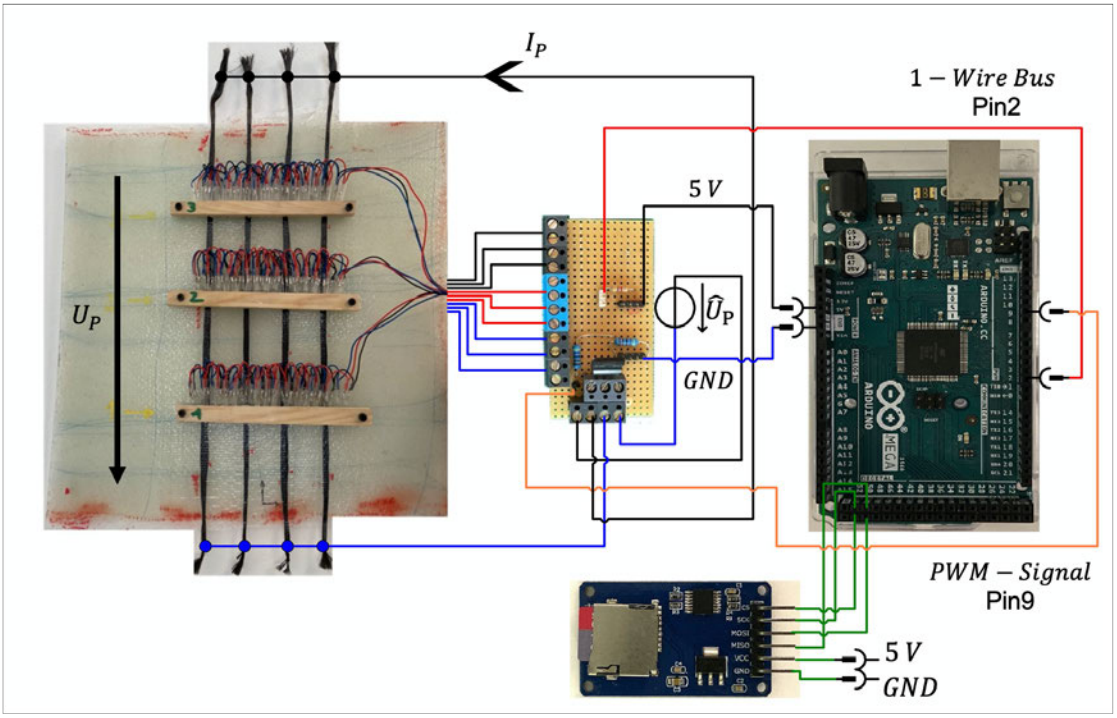


Abbildung A.1.: Gesamtschaltplan

Die Inbetriebnahme erfolgt nach den folgenden Ablauf:

- Nach dem Systemaufbau werden die Textdateien auf der SD-Karte mit den gewünschten Prozessparametern gefüllt. Der Arduino ist noch nicht an den PC oder an einer Versorgungsspannung angeschlossen. Für die Auflösung sind lediglich die Eingaben 11 oder 12 in die Datei gültig, für andere Eingaben geht das Programm in den Error-Mode und zeigt dem Benutzer einen Fehler an. Für die Zieltemperatur ist ein Wertebereich von $[25.00; 85.00]$ vorgegeben. Dabei müssen die Vor-und Nachkommazahl des Eintrages der gewählte Zieltemperatur in die txt-Datei durch einen Punkt getrennt werden. Das System akzeptiert zwei Nachkommastellen.
- Nun wird der Ardunino an den PC angeschlossen und das Regelungsprogrammmit der Arduino-Software geöffnet. Anschließend wird die SD-Karte in das Speichermodul eingefügt. Das Programm wird nun hochgeladen. Für die Überwachung des Prozesses muss der Serielle Monitor der Arduino Software geöffnet werden. Anschließend

bekommt der Benutzer über den Monitor die Rückmeldung, ob seine Eingaben gültig waren.

Abbildung A.2.: Anzeige nach gültigen Eingaben

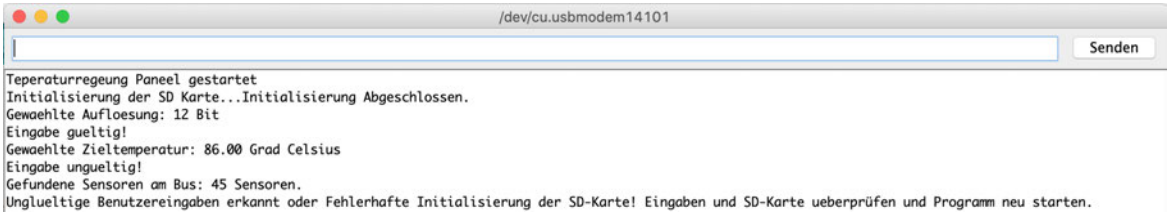


Abbildung A.3.: Anzeige nach ungültigen Eingaben

Nur nach gültigen Eingaben beginnt das System mit dem Beheizen des Paneels und der Aufzeichnung der Oberflächentemperatur.

- Über den Seriellen Monitor kann der Prozess überwacht werden. Das Programm zeigt nach einer Messung die Temperatur eines jeden Sensors an.

- Zur Beendigung des Prozesses reicht es, den Arduino und das Netzteil vom Strom zu trennen. Anschließend kann die SD-Karte aus dem Modul entfernt und mit Hilfe des Adapters mit dem Computer verbunden werden. Das Regelungsprogramm hat eine neue Datei erstellt, welche die Daten des Prozesses beinhalten: $DATEN.txt$. Mithilfe des auf der SD-Karte enthaltenen Matlab-Auswerteprogrammes, können die Daten direkt ausgewertet werden. Vor dem Start eines neuen Regelungsprozesses, muss die Datei $DATEN.txt$ von der SD-Karte gelöscht werden, da ansonsten die Messdaten des folgenden Prozesses an die bereits erzeugten angehängt werden.

A.2. Datenblätter

Technical Data Sheet

A.2.1. CM-Preg T-C-230/600 CP004 39

Description: CM-Preg T-C- 230/600 CP004 39

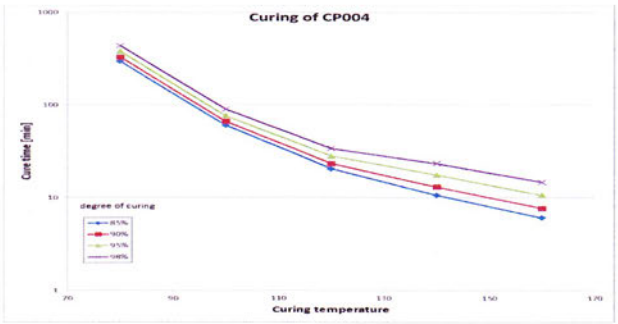
The CP004 Systems are thermoplastic modified Epoxy systems with outstanding toughening properties. The resin system is mainly for industrial applications and suitable to carry high loads

Benefits and Features:

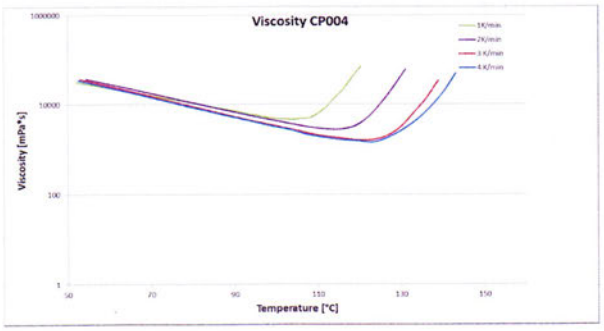
Modified flow characteristics
Good bonding properties
Toughend System
Suitable for a wide temperature range

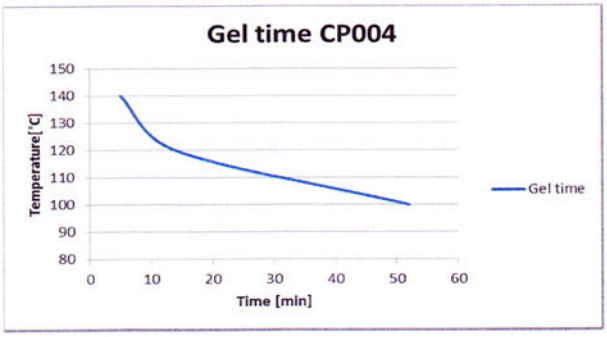
Matrix Properties:

		CP004
min. Viscosity	mPas	1600
T_g (1h/120°C)	°C	130
Color		nature
Specials		n.a.



For more information please contact
c-m-p GmbH Industrieparkstr 15 52525 Heinsberg Tel.: +49 (0) 2452 1572110
www.c-m-p-gmbh.de support@c-m-p-gmbh.de





Storage Life and Storage Conditions

60 Days	@	20 °C
12 month	@	-18 °C

Processing

Prepregs out of CP004 resin system are processable with all common techologies. The typical processing window is between 80°C and 165°C. The curing time varies from 15min up to 4hours depending on temperature

For more information please contact
c-m-p GmbH Industrieparkstr 15 52525 Heinsberg Tel.: +49 (0) 2452 1572110
www.c-m-p-gmbh.de support@c-m-p-gmbh.de

Technical Data Sheet

Prepreg Properties:

Textur:			Epoxy Carbon UD-Prepreg	
Fiber Areal weight:	g/m^2	DIN	29971	230
Resin Content:	%	DIN	2557 C	39
Prepreg Areal Weight:	g/m^2	DIN	2557 C	377
Width:	mm			600
Thickness :	mm			0,25

Laminate-Properties (Example - HT - Carbon fiber / 60 Vol%)

Artikel				CM-Preg F	CM-Preg T
Grade				Köper 2/2	UD
FAW	g/m^2			245	420
Tensile Strength 0°	MPa	DIN	ISO 527	1100	1900
E-Modulus 0°	GPa	DIN	ISO 527	70	135
Flexural Strength	MPa	DIN	ISO 14125	1050	1250
Flexural-Modulus 0°	GPa	DIN	ISO 14125	62	110
ILSF	MPa	DIN	EN 2563	70	58

additional mechanical properties available on request

Important

This is not a specification. All information is believed to be accurate in relation to the performance, storage and other characteristics of the product without acceptance of liability. Users are held to do their own tests to check the suitability of the product for its particular purpose

print date 30.03.2017

For more information please contact
c-m-p GmbH Industrieparkstr 15 52525 Heinsberg Tel.: +49 (0) 2452 1572110
www.c-m-p-gmbh.de support@c-m-p-gmbh.de

A.2.2. Glasfilamentgewebe

interglas® porcherindustries

porcherindustries Germany GmbH

GLASFILAMENTGEWEBE für die KUNSTSTOFFVERSTÄRKUNG

PRODUKTSPEZIFIKATION

				Prüfnorm
Artikel-Nummer	**07781**	**Alte ITG-Bez** 92626		MIL-Y-1140H
US-Style	7781		VIII B	AMS / MIL-C-9084
WLB Nr.	8.4568.60			DIN 65066
British Standard	BS 3396			BS 3396

Finish/Ausrüstung	FK144

	Einheit		Toleranz	Prüfnorm
Bindung		Atlas 1/7		DIN ISO 9354
Flächengewicht	g / m²	296,0	± 5%	DIN EN 12127
Garn	tex			DIN EN 12654
Kettgarn		EC6-68 tex		
Schußgarn		EC6-68 tex		
Fadendichte	Fd / cm			DIN EN 1049
Kette		22,0	± 5%	
Schuß		21,5	± 5%	
Temperaturbelastung 1)				
Dauerbelastung	℃	260		
kurzzeitige Belastung	℃	600		
Feuchtegehalt	%	< 0,2		DIN EN 3616
Finishgehalt	%	0,08 - 0,28		DIN ISO 1887
				DIN EN 60
Dicke (Richtwert trocken)	mm	0,35	± 5%	DIN ISO 4603/E
im Laminat	mm	0,26	± 5%	

Die o.g. Daten beschreiben technische Belange nach dem heutigen Stand unserer Kenntnisse. Sie stellen weder Qualitätsmerkmale dar noch entbinden sie von der Eigenverantwortlichkeit beim Umgang mit unseren Geweben. Änderungen sind vorbehalten.

Porcher Industries Germany GmbH, Benzstraße 14, D-89155 Erbach, Telefon +49 (0)7305 / 955-485, Fax +49 (0)7305 / 955-524

interglas® porcherindustries

porcherindustries Germany GmbH

GLASFILAMENTGEWEBE für die KUNSTSTOFFVERSTÄRKUNG

PRODUKTSPEZIFIKATION

	Einheit	Norm	P-D ITG	Toleranz	Prüfnorm
Qualitäts-Nummer		**07781**		**/ FK144**	
Reißfestigkeit					ASTM D 4029
Kette	N/cm			>	
Schuß	N/cm			>	
Zugfestigkeit					DIN EN 2747
Kette	MPa	335		>	
Schuß	MPa	320		>	
Zug-E-Modul					
Kette	GPa	19		>	
Schuß	GPa	18		>	
Druckfestigkeit					DIN 53454
Kette	MPa	375		>	DIN 65380
Schuß	MPa	360		>	DIN prEN 2580
Druck-E-Modul					
Kette	GPa			>	
Schuß	GPa			>	
Interlaminare Scherfestigkeit					DIN EN 2377
Kette	MPa	50		>	
Schuß	MPa	45		>	
Biegefestigkeit					DIN EN 2746
Kette	MPa	495		>	
Schuß	MPa	460		>	
Biege-E-Modul					
Kette	GPa	17		>	
Schuß	GPa	17		>	

Bemerkungen

1) Temperaturbelastung bezogen auf Trockengewebe

Porcher Industries Germany GmbH, Benzstraße 14, D-89155 Erbach, Telefon +49 (0)7305 / 955-485, Fax +49 (0)7305 / 955-524

A.2.3. Filamentgarn IMS65

Toho Tenax

TEIJIN

Delivery programme and characteristics for Tenax® IMS65 filament yarn

Brand name		**Tenax®**
Production site		**E**
Product designation		**IMS65 E23 24K 830tex**
Sizing properties		E23
Number of filaments		24.000
Nominal linear density [1)]	[tex]	830
Twist	[t/m]	0
Running length per kg	[m/kg]	1200

1) without sizing

Characteristics (typical values)

Filament diameter	[µm]	5
Density	[g/cm^3]	1.78
Tensile strength	[MPa]	6000
Tensile modulus	[GPa]	290
Elongation at break	[%]	1.9
Specific electrical resistance	[Ω cm]	1.45 x 10^{-3}

Sizing properties for fiber family IMS

IMS (Intermediate Modulus) is tailored to suit applications where strength and stiffness are of ultimate priority.

E23 = Type with ca. 1.3 % sizing based on epoxy resin

Please contact our sales team any time for choosing the right type. The stated numbers are typical values. For design purposes please request fiber specification.

Please note the application (aerospace or industry & sports) on your order.

Toho Tenax Europe GmbH Kasinostraße 19-21 42103 Wuppertal, Germany Tel.: +49 202 32 - 2339 Fax: +49 202 32 - 2360 www.tohotenax-eu.com sales@tohotenax-eu.com (01/2010)

A.2.4. Filamentgarn IMS65

Click here for production status of specific part numbers.

DS18B20

Programmable Resolution 1-Wire Digital Thermometer

General Description

The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. In addition, the DS18B20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

Each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20s to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18B20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment, or machinery, and process monitoring and control systems.

Applications

- Thermostatic Controls
- Industrial Systems
- Consumer Products
- Thermometers
- Thermally Sensitive Systems

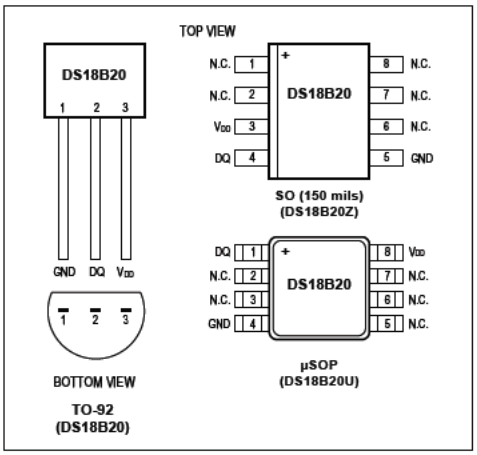
Ordering Information ***appears at end of data sheet.***

1-Wire is a registered trademark of Maxim Integrated Products, Inc.

Benefits and Features

- Unique 1-Wire® Interface Requires Only One Port Pin for Communication
- Reduce Component Count with Integrated Temperature Sensor and EEPROM
  - Measures Temperatures from -55°C to +125°C (-67°F to +257°F)
  - ±0.5°C Accuracy from -10°C to +85°C
  - Programmable Resolution from 9 Bits to 12 Bits
  - No External Components Required
- Parasitic Power Mode Requires Only 2 Pins for Operation (DQ and GND)
- Simplifies Distributed Temperature-Sensing Applications with Multidrop Capability
  - Each Device Has a Unique 64-Bit Serial Code Stored in On-Board ROM
- Flexible User-Definable Nonvolatile (NV) Alarm Settings with Alarm Search Command Identifies Devices with Temperatures Outside Programmed Limits
- Available in 8-Pin SO (150 mils), 8-Pin µSOP, and 3-Pin TO-92 Packages

Pin Configurations



19-7487; Rev 6; 7/19

maxim integrated™

Absolute Maximum Ratings

Voltage Range on Any Pin Relative to Ground ....-0.5V to +6.0V
Operating Temperature Range ..... -55°C to +125°C
Storage Temperature Range ..... -55°C to +125°C
Solder Temperature .....Refer to the IPC/JEDEC J-STD-020 Specification.

These are stress ratings only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

DC Electrical Characteristics

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Supply Voltage	V_{DD}	Local power (Note 1)		+3.0		+5.5	V
Pullup Supply Voltage	V_{PU}	Parasite power	(Notes 1, 2)	+3.0		+5.5	V
		Local power		+3.0		V_{DD}	
Thermometer Error	t_{ERR}	-10°C to +85°C	(Note 3)			±0.5	°C
		-30°C to +100°C				±1	
		-55°C to +125°C				±2	
Input Logic-Low	V_{IL}	(Notes 1, 4, 5)		-0.3		+0.8	V
Input Logic-High	V_{IH}	Local power	(Notes 1,6)	+2.2		The lower of 5.5 or V_{DD} + 0.3	V
		Parasite power		+3.0			
Sink Current	I_L	$V_{I/O}$ = 0.4V		4.0			mA
Standby Current	I_{DDS}	(Notes 7, 8)			750	1000	nA
Active Current	I_{DD}	V_{DD} = 5V (Note 9)			1	1.5	mA
DQ Input Current	I_{DQ}	(Note 10)			5		µA
Drift		(Note 11)			±0.2		°C

Note 1: All voltages are referenced to ground.
Note 2: The Pullup Supply Voltage specification assumes that the pullup device is ideal, and therefore the high level of the pullup is equal to V_{PU}. In order to meet the V_{IH} spec of the DS18B20, the actual supply rail for the strong pullup transistor must include margin for the voltage drop across the transistor when it is turned on; thus: $V_{PU\_ACTUAL} = V_{PU\_IDEAL} + V_{TRANSISTOR}$.
Note 3: See typical performance curve in Figure 1. Thermometer Error limits are 3-sigma values.
Note 4: Logic-low voltages are specified at a sink current of 4mA.
Note 5: To guarantee a presence pulse under low voltage parasite power conditions, V_{ILMAX} may have to be reduced to as low as 0.5V.
Note 6: Logic-high voltages are specified at a source current of 1mA.
Note 7: Standby current specified up to +70°C. Standby current typically is 3µA at +125°C.
Note 8: To minimize I_{DDS}, DQ should be within the following ranges: GND ≤ DQ ≤ GND + 0.3V or V_{DD} – 0.3V ≤ DQ ≤ V_{DD}.
Note 9: Active current refers to supply current during active temperature conversions or EEPROM writes.
Note 10: DQ line is high (“high-Z” state).
Note 11: Drift data is based on a 1000-hour stress test at +125°C with V_{DD} = 5.5V.

AC Electrical Characteristics–NV Memory

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
NV Write Cycle Time	t_{WR}			2	10	ms
EEPROM Writes	N_{EEWR}	-55°C to +55°C	50k			writes
EEPROM Data Retention	t_{EEDR}	-55°C to +55°C	10			years

AC Electrical Characteristics

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Temperature Conversion Time	t_{CONV}	9-bit resolution	(Note 12)			93.75	ms
		10-bit resolution				187.5	
		11-bit resolution				375	
		12-bit resolution				750	
Time to Strong Pullup On	t_{SPON}	Start convert T command issued				10	µs
Time Slot	t_{SLOT}	(Note 12)		60		120	µs
Recovery Time	t_{REC}	(Note 12)		1			µs
Write 0 Low Time	t_{LOW0}	(Note 12)		60		120	µs
Write 1 Low Time	t_{LOW1}	(Note 12)		1		15	µs
Read Data Valid	t_{RDV}	(Note 12)				15	µs
Reset Time High	t_{RSTH}	(Note 12)		480			µs
Reset Time Low	t_{RSTL}	(Notes 12, 13)		480			µs
Presence-Detect High	t_{PDHIGH}	(Note 12)		15		60	µs
Presence-Detect Low	t_{PDLOW}	(Note 12)		60		240	µs
Capacitance	$C_{IN/OUT}$					25	pF

Note 12: See the timing diagrams in Figure 2.
Note 13: Under parasite power, if t_{RSTL} > 960µs, a power-on reset can occur.

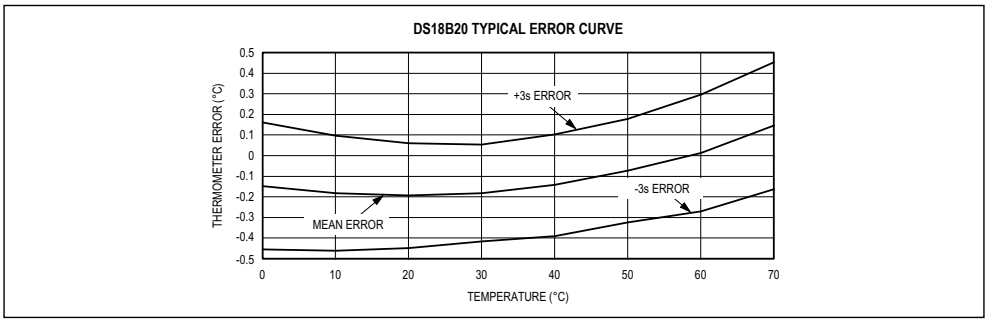


Figure 1. Typical Performance Curve

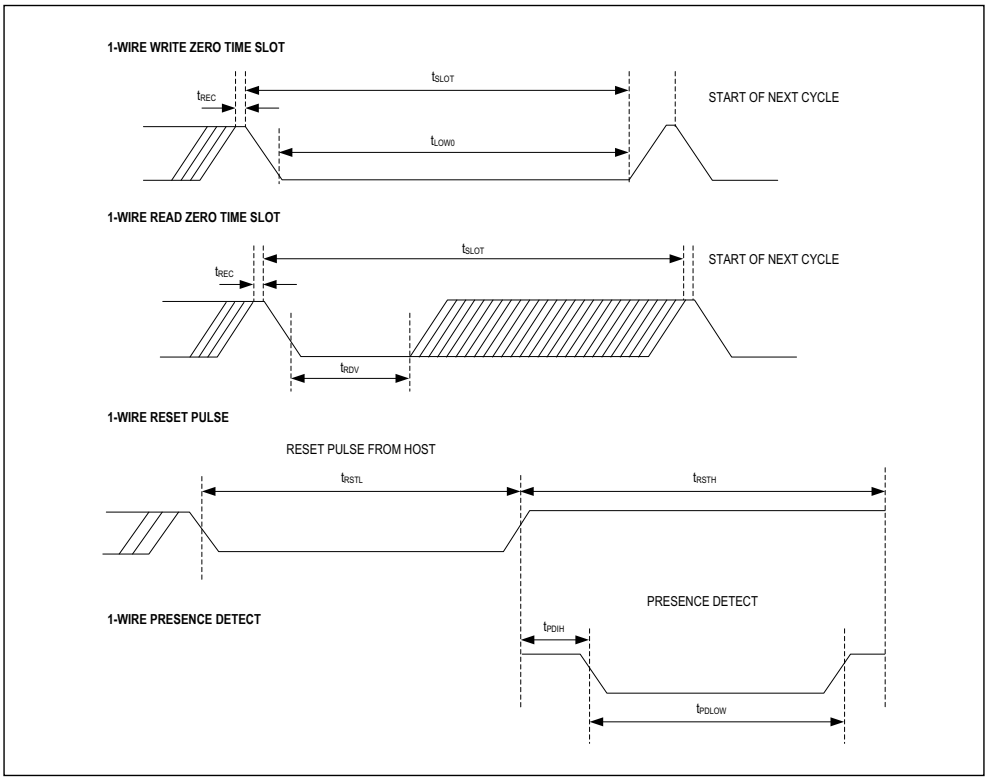


Figure 2. Timing Diagrams

Pin Description

PIN			NAME	FUNCTION
SO	µSOP	TO-92		
1, 2, 6, 7, 8	2, 3, 5, 6, 7	—	N.C.	No Connection
3	8	3	V_{DD}	Optional V_{DD}. V_{DD} must be grounded for operation in parasite power mode.
4	1	2	DQ	Data Input/Output. Open-drain 1-Wire interface pin. Also provides power to the device when used in parasite power mode (see the *Powering the DS18B20* section.)
5	4	1	GND	Ground

Overview

Figure 3 shows a block diagram of the DS18B20, and pin descriptions are given in the *Pin Description* table. The 64-bit ROM stores the device's unique serial code. The scratchpad memory contains the 2-byte temperature register that stores the digital output from the temperature sensor. In addition, the scratchpad provides access to the 1-byte upper and lower alarm trigger registers (T_H and T_L) and the 1-byte configuration register. The configuration register allows the user to set the resolution of the temperature-to-digital conversion to 9, 10, 11, or 12 bits. The T_H, T_L, and configuration registers are nonvolatile (EEPROM), so they will retain data when the device is powered down.

The DS18B20 uses Maxim's exclusive 1-Wire bus protocol that implements bus communication using one control signal. The control line requires a weak pullup resistor since all devices are linked to the bus via a 3-state or open-drain port (the DQ pin in the case of the DS18B20). In this bus system, the microprocessor (the master device) identifies and addresses devices on the bus using each device's unique 64-bit code. Because each device has a unique code, the number of devices that can be addressed on one bus is virtually unlimited. The 1-Wire bus protocol, including detailed explanations of the commands and "time slots," is covered in the *1-Wire Bus System* section.

Another feature of the DS18B20 is the ability to operate without an external power supply. Power is instead supplied through the 1-Wire pullup resistor through the DQ pin when the bus is high. The high bus signal also charges an internal capacitor (C_{PP}), which then supplies power to the device when the bus is low. This method of deriving power from the 1-Wire bus is referred to as "parasite power." As an alternative, the DS18B20 may also be powered by an external supply on V_{DD}.

Operation—Measuring Temperature

The core functionality of the DS18B20 is its direct-to-digital temperature sensor. The resolution of the temperature sensor is user-configurable to 9, 10, 11, or 12 bits, corresponding to increments of 0.5°C, 0.25°C, 0.125°C, and 0.0625°C, respectively. The default resolution at power-up is 12-bit. The DS18B20 powers up in a low-power idle state. To initiate a temperature measurement and A-to-D conversion, the master must issue a Convert T [44h] command. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18B20 returns to its idle state. If the DS18B20 is powered by an external supply, the master can issue "read time slots" (see the *1-Wire Bus System* section) after the Convert T command and the DS18B20 will respond by transmitting 0 while the temperature conversion is in progress and 1 when the conversion is done. If the DS18B20 is powered with parasite power, this notification technique cannot be used since the bus must be pulled high by a strong pullup during the entire temperature conversion. The bus requirements for parasite power are explained in detail in the *Powering the DS18B20* section.

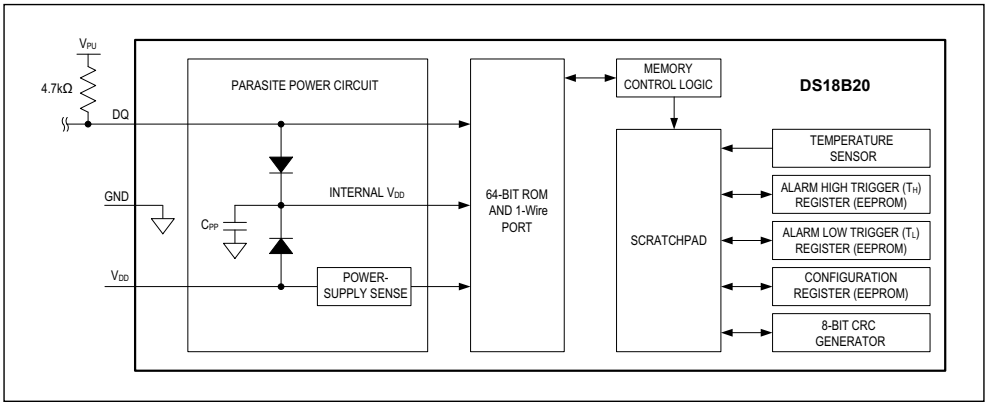


Figure 3. DS18B20 Block Diagram

The DS18B20 output temperature data is calibrated in degrees Celsius; for Fahrenheit applications, a lookup table or conversion routine must be used. The temperature data is stored as a 16-bit sign-extended two's complement number in the temperature register (see Figure 4). The sign bits (S) indicate if the temperature is positive or negative: for positive numbers S = 0 and for negative numbers S = 1. If the DS18B20 is configured for 12-bit resolution, all bits in the temperature register will contain valid data. For 11-bit resolution, bit 0 is undefined. For 10-bit resolution, bits 1 and 0 are undefined, and for 9-bit resolution bits 2, 1, and 0 are undefined. Table 1 gives examples of digital output data and the corresponding temperature reading for 12-bit resolution conversions.

Operation—Alarm Signaling

After the DS18B20 performs a temperature conversion, the temperature value is compared to the user-defined two's complement alarm trigger values stored in the 1-byte T_H and T_L registers (see Figure 5). The sign bit (S) indicates if the value is positive or negative: for positive numbers S = 0 and for negative numbers S = 1. The T_H and T_L registers are nonvolatile (EEPROM) so they will retain data when the device is powered down. T_H and T_L can be accessed through bytes 2 and 3 of the scratchpad as explained in the *Memory* section.

Only bits 11 through 4 of the temperature register are used in the T_H and T_L comparison since T_H and T_L are 8-bit registers. If the measured temperature is lower than

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	BIT 15	**BIT 14**	**BIT 13**	**BIT 12**	**BIT 11**	**BIT 10**	**BIT 9**	**BIT 8**
MS BYTE	S	S	S	S	S	2^6	2^5	2^4

S = SIGN

Figure 4. Temperature Register Format

Table 1. Temperature/Data Relationship

TEMPERATURE (°C)	DIGITAL OUTPUT (BINARY)	DIGITAL OUTPUT (HEX)
+125	0000 0111 1101 0000	07D0h
+85*	0000 0101 0101 0000	0550h
+25.0625	0000 0001 1001 0001	0191h
+10.125	0000 0000 1010 0010	00A2h
+0.5	0000 0000 0000 1000	0008h
0	0000 0000 0000 0000	0000h
-0.5	1111 1111 1111 1000	FFF8h
-10.125	1111 1111 0101 1110	FF5Eh
-25.0625	1111 1110 0110 1111	FE6Fh
-55	1111 1100 1001 0000	FC90h

**The power-on reset value of the temperature register is +85°C.*

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
S	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Figure 5. T_H and T_L Register Format

or equal to T_L or higher than or equal to T_H, an alarm condition exists and an alarm flag is set inside the DS18B20. This flag is updated after every temperature measurement; therefore, if the alarm condition goes away, the flag will be turned off after the next temperature conversion.

The master device can check the alarm flag status of all DS18B20s on the bus by issuing an Alarm Search [ECh] command. Any DS18B20s with a set alarm flag will respond to the command, so the master can determine exactly which DS18B20s have experienced an alarm condition. If an alarm condition exists and the T_H or T_L settings have changed, another temperature conversion should be done to validate the alarm condition.

Powering the DS18B20

The DS18B20 can be powered by an external supply on the V_{DD} pin, or it can operate in "parasite power" mode, which allows the DS18B20 to function without a local external supply. Parasite power is very useful for applications that require remote temperature sensing or that are very space constrained. Figure 3 shows the DS18B20's parasite-power control circuitry, which "steals" power from the 1-Wire bus via the DQ pin when the bus is high. The stolen charge powers the DS18B20 while the bus is high, and some of the charge is stored on the parasite power capacitor (C_{PP}) to provide power when the bus is low. When the DS18B20 is used in parasite power mode, the V_{DD} pin must be connected to ground.

In parasite power mode, the 1-Wire bus and CPP can provide sufficient current to the DS18B20 for most operations as long as the specified timing and voltage requirements are met (see the *DC Electrical Characteristics* and *AC Electrical Characteristics*). However, when the DS18B20 is performing temperature conversions or copying data from the scratchpad memory to EEPROM, the operating current can be as high as 1.5mA. This current can cause an unacceptable voltage drop across the weak 1-Wire pullup resistor and is more current than can be supplied by C_{PP}. To assure that the DS18B20 has sufficient supply current, it is necessary to provide a strong pullup on the 1-Wire bus whenever temperature conversions are taking place or data is being copied from the scratchpad to EEPROM. This can be accomplished by using a MOSFET to pull the bus directly to the rail as shown in Figure 6. The 1-Wire bus must be switched to the strong pullup within 10µs (max) after a Convert T [44h] or Copy Scratchpad [48h] command is issued, and the bus must be held high by the pullup for the duration of the conversion (t_{CONV}) or data transfer (t_{WR} = 10ms). No other activity can take place on the 1-Wire bus while the pullup is enabled.

The DS18B20 can also be powered by the conventional method of connecting an external power supply to the V_{DD} pin, as shown in Figure 7. The advantage of this method is that the MOSFET pullup is not required, and the 1-Wire bus is free to carry other traffic during the temperature conversion time.

The use of parasite power is not recommended for temperatures above +100°C since the DS18B20 may not be able to sustain communications due to the higher leakage currents that can exist at these temperatures. For applications in which such temperatures are likely, it is strongly recommended that the DS18B20 be powered by an external power supply.

In some situations the bus master may not know whether the DS18B20s on the bus are parasite powered or powered by external supplies. The master needs this information to determine if the strong bus pullup should be used during temperature conversions. To get this information, the master can issue a Skip ROM [CCh] command followed by a Read Power Supply [B4h] command followed by a "read time slot". During the read time slot, parasite powered DS18B20s will pull the bus low, and externally powered DS18B20s will let the bus remain high. If the bus is pulled low, the master knows that it must supply the strong pullup on the 1-Wire bus during temperature conversions.

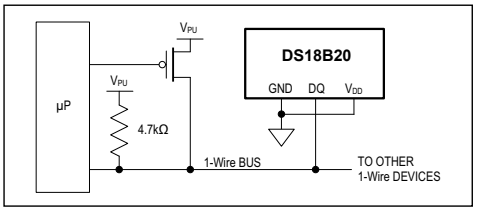


Figure 6. Supplying the Parasite-Powered DS18B20 During Temperature Conversions

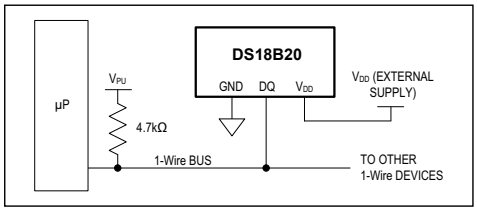


Figure 7. Powering the DS18B20 with an External Supply

64-BIT Lasered ROM code

Each DS18B20 contains a unique 64–bit code (see Figure 8) stored in ROM. The least significant 8 bits of the ROM code contain the DS18B20's 1-Wire family code: 28h. The next 48 bits contain a unique serial number. The most significant 8 bits contain a cyclic redundancy check (CRC) byte that is calculated from the first 56 bits of the ROM code. A detailed explanation of the CRC bits is provided in the *CRC Generation* section. The 64-bit ROM code and associated ROM function control logic allow the DS18B20 to operate as a 1-Wire device using the protocol detailed in the *1-Wire Bus System* section.

Memory

The DS18B20's memory is organized as shown in Figure 9. The memory consists of an SRAM scratchpad with nonvolatile EEPROM storage for the high and low alarm trigger registers (T_H and T_L) and configuration register. Note that if the DS18B20 alarm function is not used, the TH and TL registers can serve as general-purpose memory. All memory commands are described in detail in the *DS18B20 Function Commands* section.

Byte 0 and byte 1 of the scratchpad contain the LSB and the MSB of the temperature register, respectively. These bytes are read-only. Bytes 2 and 3 provide access to TH and TL registers. Byte 4 contains the configuration register data, which is explained in detail in the *Configuration Register* section. Bytes 5, 6, and 7 are reserved for internal use by the device and cannot be overwritten.

Byte 8 of the scratchpad is read-only and contains the CRC code for bytes 0 through 7 of the scratchpad. The DS18B20 generates this CRC using the method described in the *CRC Generation* section.

Data is written to bytes 2, 3, and 4 of the scratchpad using the Write Scratchpad [4Eh] command; the data must be transmitted to the DS18B20 starting with the least significant bit of byte 2. To verify data integrity, the scratchpad can be read (using the Read Scratchpad [BEh] command) after the data is written. When reading the scratchpad, data is transferred over the 1-Wire bus starting with the least significant bit of byte 0. To transfer the T_H, T_L and configuration data from the scratchpad to EEPROM, the master must issue the Copy Scratchpad [48h] command.

Data in the EEPROM registers is retained when the device is powered down; at power-up the EEPROM data is reloaded into the corresponding scratchpad locations. Data can also be reloaded from EEPROM to the scratchpad at any time using the Recall E^2 [B8h] command. The master can issue read time slots following the Recall E^2 command and the DS18B20 will indicate the status of the recall by transmitting 0 while the recall is in progress and 1 when the recall is done.

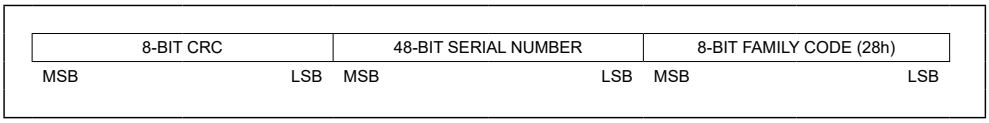


Figure 8. 64-Bit Lasered ROM Code

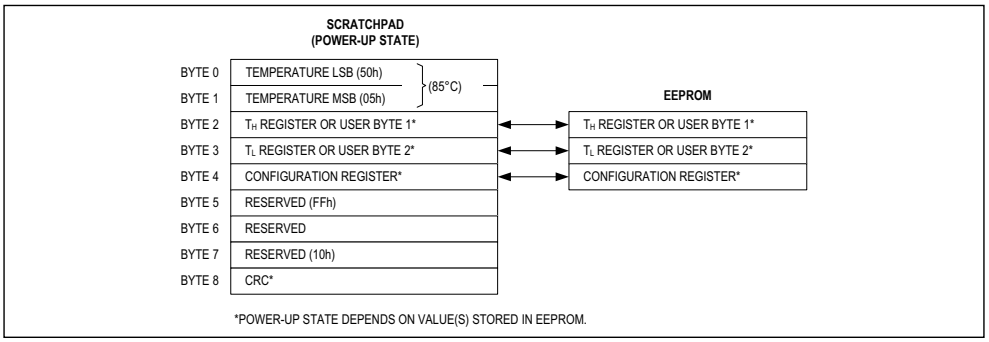


Figure 9. DS18B20 Memory Map

Configuration Register

Byte 4 of the scratchpad memory contains the configuration register, which is organized as illustrated in Figure 10. The user can set the conversion resolution of the DS18B20 using the R0 and R1 bits in this register as shown in Table 2. The power-up default of these bits is R0 = 1 and R1 = 1 (12-bit resolution). Note that there is a direct tradeoff between resolution and conversion time. Bit 7 and bits 0 to 4 in the configuration register are reserved for internal use by the device and cannot be overwritten.

CRC Generation

CRC bytes are provided as part of the DS18B20's 64-bit ROM code and in the 9th byte of the scratchpad memory. The ROM code CRC is calculated from the first 56 bits of the ROM code and is contained in the most significant byte of the ROM. The scratchpad CRC is calculated from the data stored in the scratchpad, and therefore it changes when the data in the scratchpad changes. The CRCs provide the bus master with a method of data validation when data is read from the DS18B20. To verify that data has been read correctly, the bus master must re-calculate the CRC from the received data and then compare this value to either the ROM code CRC (for ROM reads) or to the scratchpad CRC (for scratchpad reads). If the calculated CRC matches the read CRC, the data has been received error free. The comparison of CRC values and the decision to continue with an operation are determined entirely by the bus master. There is no circuitry inside the DS18B20 that prevents a command sequence from proceeding if the DS18B20 CRC (ROM or scratchpad) does not match the value generated by the bus master.

The equivalent polynomial function of the CRC (ROM or scratchpad) is:

$$CRC = X^8 + X^5 + X^4 + 1$$

The bus master can re-calculate the CRC and compare it to the CRC values from the DS18B20 using the polynomial generator shown in Figure 11. This circuit consists of a shift register and XOR gates, and the shift register bits are initialized to 0. Starting with the least significant bit of the ROM code or the least significant bit of byte 0 in the scratchpad, one bit at a time should shifted into the shift register. After shifting in the 56th bit from the ROM or the most significant bit of byte 7 from the scratchpad, the polynomial generator will contain the recalculated CRC. Next, the 8-bit ROM code or scratchpad CRC from the DS18B20 must be shifted into the circuit. At this point, if the re-calculated CRC was correct, the shift register will contain all 0s. Additional information about the Maxim 1-Wire cyclic redundancy check is available in *Application Note 27: Understanding and Using Cyclic Redundancy Checks with Maxim iButton Products*.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0	R1	R0	1	1	1	1	1

Figure 10. Configuration Register

Table 2. Thermometer Resolution Configuration

R1	R0	RESOLUTION (BITS)	MAX CONVERSION TIME	
0	0	9	93.75ms	(t_{CONV}/8)
0	1	10	187.5ms	(t_{CONV}/4)
1	0	11	375ms	(t_{CONV}/2)
1	1	12	750ms	(t_{CONV})

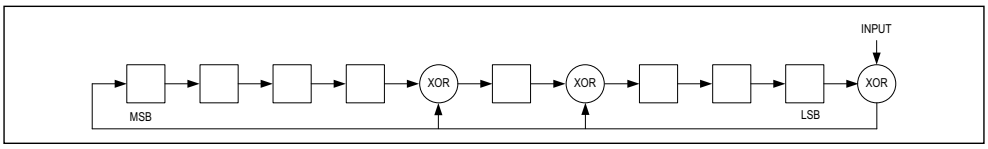


Figure 11. CRC Generator

1-Wire Bus System

The 1-Wire bus system uses a single bus master to control one or more slave devices. The DS18B20 is always a slave. When there is only one slave on the bus, the system is referred to as a “single-drop” system; the system is “multidrop” if there are multiple slaves on the bus.

All data and commands are transmitted least significant bit first over the 1-Wire bus.

The following discussion of the 1-Wire bus system is broken down into three topics: hardware configuration, transaction sequence, and 1-Wire signaling (signal types and timing).

Hardware Configuration

The 1-Wire bus has by definition only a single data line. Each device (master or slave) interfaces to the data line via an open-drain or 3-state port. This allows each device to “release” the data line when the device is not transmitting data so the bus is available for use by another device. The 1-Wire port of the DS18B20 (the DQ pin) is open drain with an internal circuit equivalent to that shown in Figure 12.

The 1-Wire bus requires an external pullup resistor of approximately 5kΩ; thus, the idle state for the 1-Wire bus is high. If for any reason a transaction needs to be suspended, the bus MUST be left in the idle state if the transaction is to resume. Infinite recovery time can occur between bits so long as the 1-Wire bus is in the inactive (high) state during the recovery period. If the bus is held low for more than 480µs, all components on the bus will be reset.

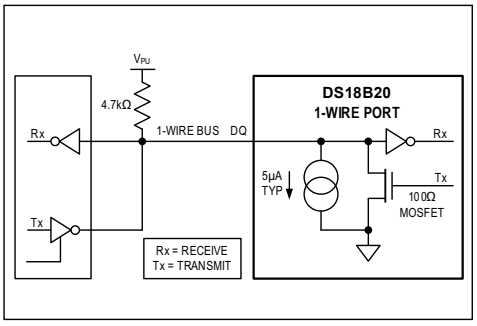


Figure 12. Hardware Configuration

Transaction Sequence

The transaction sequence for accessing the DS18B20 is as follows:

Step 1. Initialization

Step 2. ROM Command (followed by any required data exchange)

Step 3. DS18B20 Function Command (followed by any required data exchange)

It is very important to follow this sequence every time the DS18B20 is accessed, as the DS18B20 will not respond if any steps in the sequence are missing or out of order. Exceptions to this rule are the Search ROM [F0h] and Alarm Search [ECh] commands. After issuing either of these ROM commands, the master must return to Step 1 in the sequence.

Initialization

All transactions on the 1-Wire bus begin with an initialization sequence. The initialization sequence consists of a reset pulse transmitted by the bus master followed by presence pulse(s) transmitted by the slave(s). The presence pulse lets the bus master know that slave devices (such as the DS18B20) are on the bus and are ready to operate. Timing for the reset and presence pulses is detailed in the *1-Wire Signaling* section.

ROM Commands

After the bus master has detected a presence pulse, it can issue a ROM command. These commands operate on the unique 64-bit ROM codes of each slave device and allow the master to single out a specific device if many are present on the 1-Wire bus. These commands also allow the master to determine how many and what types of devices are present on the bus or if any device has experienced an alarm condition. There are five ROM commands, and each command is 8 bits long. The master device must issue an appropriate ROM command before issuing a DS18B20 function command. A flowchart for operation of the ROM commands is shown in Figure 13.

Search Rom [F0h]

When a system is initially powered up, the master must identify the ROM codes of all slave devices on the bus, which allows the master to determine the number of slaves and their device types. The master learns the ROM codes through a process of elimination that requires the master to perform a Search ROM cycle (i.e., Search ROM command followed by data exchange) as many times as necessary to identify all of the slave devices.

If there is only one slave on the bus, the simpler Read ROM [33h] command can be used in place of the Search ROM process. For a detailed explanation of the Search ROM procedure, refer to *Application Note 937: Book of iButton® Standards*. After every Search ROM cycle, the bus master must return to Step 1 (Initialization) in the transaction sequence.

Read Rom [33h]

This command can only be used when there is one slave on the bus. It allows the bus master to read the slave's 64-bit ROM code without using the Search ROM procedure. If this command is used when there is more than one slave present on the bus, a data collision will occur when all the slaves attempt to respond at the same time.

Match Rom [55H]

The match ROM command followed by a 64-bit ROM code sequence allows the bus master to address a specific slave device on a multidrop or single-drop bus. Only the slave that exactly matches the 64-bit ROM code sequence will respond to the function command issued by the master; all other slaves on the bus will wait for a reset pulse.

Skip Rom [CCh]

The master can use this command to address all devices on the bus simultaneously without sending out any ROM code information. For example, the master can make all DS18B20s on the bus perform simultaneous temperature conversions by issuing a Skip ROM command followed by a Convert T [44h] command.

Note that the Read Scratchpad [BEh] command can follow the Skip ROM command only if there is a single slave device on the bus. In this case, time is saved by allowing the master to read from the slave without sending the device's 64-bit ROM code. A Skip ROM command followed by a Read Scratchpad command will cause a data collision on the bus if there is more than one slave since multiple devices will attempt to transmit data simultaneously.

Alarm Search [ECh]

The operation of this command is identical to the operation of the Search ROM command except that only slaves with a set alarm flag will respond. This command allows the master device to determine if any DS18B20s experienced an alarm condition during the most recent temperature conversion. After every Alarm Search cycle (i.e., Alarm Search command followed by data exchange), the bus master must return to Step 1 (Initialization) in the transaction sequence. See the *Operation—Alarm Signaling* section for an explanation of alarm flag operation.

DS18B20 Function Commands

After the bus master has used a ROM command to address the DS18B20 with which it wishes to communicate, the master can issue one of the DS18B20 function commands. These commands allow the master to write to and read from the DS18B20's scratchpad memory, initiate temperature conversions and determine the power supply mode. The DS18B20 function commands, which are described below, are summarized in Table 3 and illustrated by the flowchart in Figure 14.

Convert T [44h]

This command initiates a single temperature conversion. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18B20 returns to its low-power idle state. If the device is being used in parasite power mode, within 10µs (max) after this command is issued the master must enable a strong pullup on the 1-Wire bus for the duration of the conversion (t_{CONV}) as described in the *Powering the DS18B20* section. If the DS18B20 is powered by an external supply, the master can issue read time slots after the Convert T command and the DS18B20 will respond by transmitting a 0 while the temperature conversion is in progress and a 1 when the conversion is done. In parasite power mode this notification technique cannot be used since the bus is pulled high by the strong pullup during the conversion.

Write Scratchpad [4Eh]

This command allows the master to write 3 bytes of data to the DS18B20's scratchpad. The first data byte is written into the T_H register (byte 2 of the scratchpad), the second byte is written into the T_L register (byte 3), and the third byte is written into the configuration register (byte 4). Data must be transmitted least significant bit first. All three bytes MUST be written before the master issues a reset, or the data may be corrupted.

Read Scratchpad [BEh]

This command allows the master to read the contents of the scratchpad. The data transfer starts with the least significant bit of byte 0 and continues through the scratchpad until the 9th byte (byte 8 – CRC) is read. The master may issue a reset to terminate reading at any time if only part of the scratchpad data is needed.

iButton is a registered trademark of Maxim Integrated Products, Inc.

Copy Scratchpad [48h]

This command copies the contents of the scratchpad T_H, T_L and configuration registers (bytes 2, 3 and 4) to EEPROM. If the device is being used in parasite power mode, within 10µs (max) after this command is issued the master must enable a strong pullup on the 1-Wire bus for at least 10ms as described in the *Powering the DS18B20* section.

Recall E^2 [B8h]

This command recalls the alarm trigger values (T_H and T_L) and configuration data from EEPROM and places the data in bytes 2, 3, and 4, respectively, in the scratchpad memory. The master device can issue read time slots following the Recall E^2 command and the DS18B20 will indicate the status of the recall by transmitting 0 while the recall is in progress and 1 when the recall is done. The recall operation happens automatically at power-up, so valid data is available in the scratchpad as soon as power is applied to the device.

Read Power Supply [B4h]

The master device issues this command followed by a read time slot to determine if any DS18B20s on the bus are using parasite power. During the read time slot, parasite powered DS18B20s will pull the bus low, and externally powered DS18B20s will let the bus remain high. See the *Powering the DS18B20* section for usage information for this command.

Table 3. DS18B20 Function Command Set

COMMAND	DESCRIPTION	PROTOCOL	1-Wire BUS ACTIVITY AFTER COMMAND IS ISSUED	NOTES
TEMPERATURE CONVERSION COMMANDS				
Convert T	Initiates temperature conversion.	44h	DS18B20 transmits conversion status to master (not applicable for parasite-powered DS18B20s).	1
MEMORY COMMANDS				
Read Scratchpad	Reads the entire scratchpad including the CRC byte.	BEh	DS18B20 transmits up to 9 data bytes to master.	2
Write Scratchpad	Writes data into scratchpad bytes 2, 3, and 4 (T_H, T_L, and configuration registers).	4Eh	Master transmits 3 data bytes to DS18B20.	3
Copy Scratchpad	Copies T_H, T_L, and configuration register data from the scratchpad to EEPROM.	48h	None	1
Recall E^2	Recalls T_H, T_L, and configuration register data from EEPROM to the scratchpad.	B8h	DS18B20 transmits recall status to master.	
Read Power Supply	Signals DS18B20 power supply mode to the master.	B4h	DS18B20 transmits supply status to master.	

Note 1: For parasite-powered DS18B20s, the master must enable a strong pullup on the 1-Wire bus during temperature conversions and copies from the scratchpad to EEPROM. No other bus activity may take place during this time.
Note 2: The master can interrupt the transmission of data at any time by issuing a reset.
Note 3: All three bytes must be written before a reset is issued.

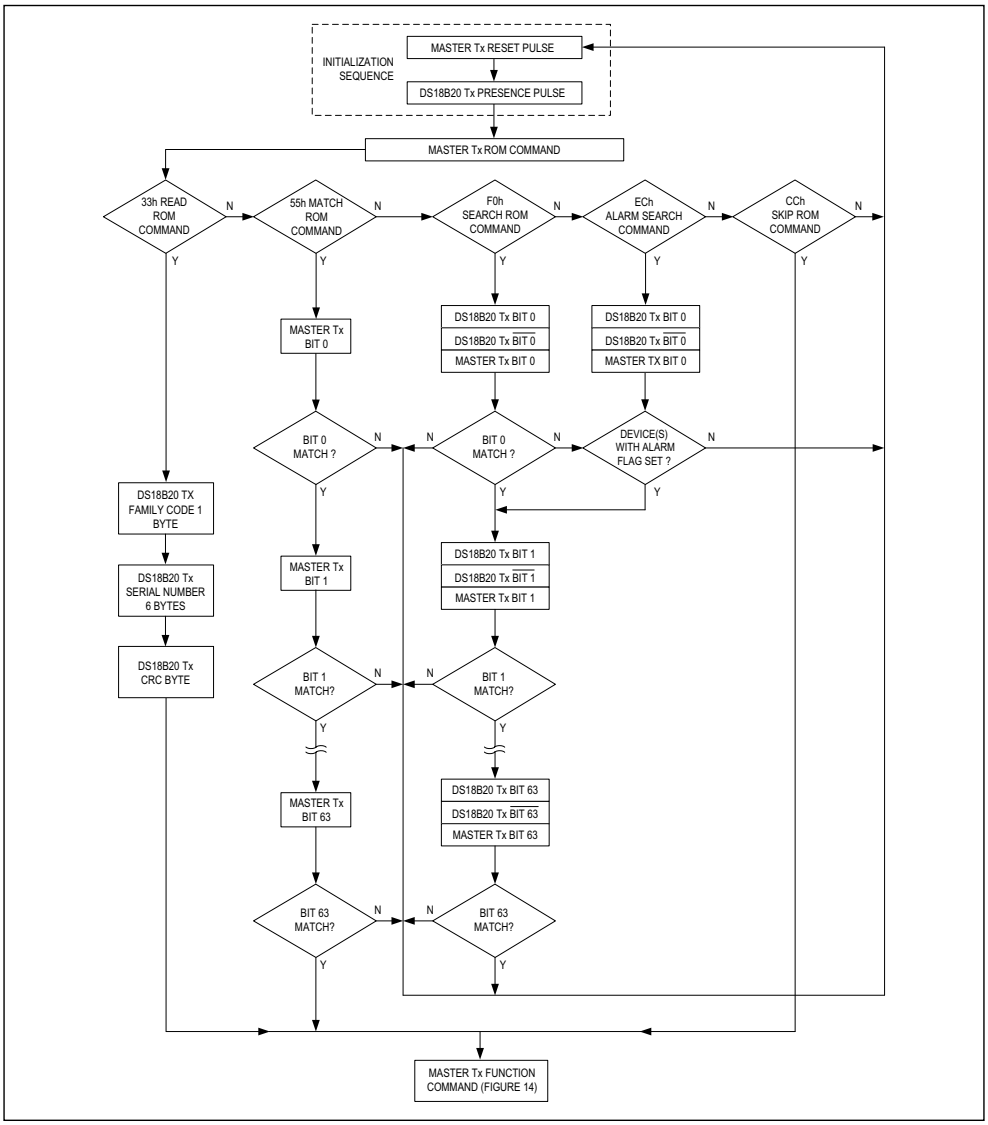


Figure 13. ROM Commands Flowchart

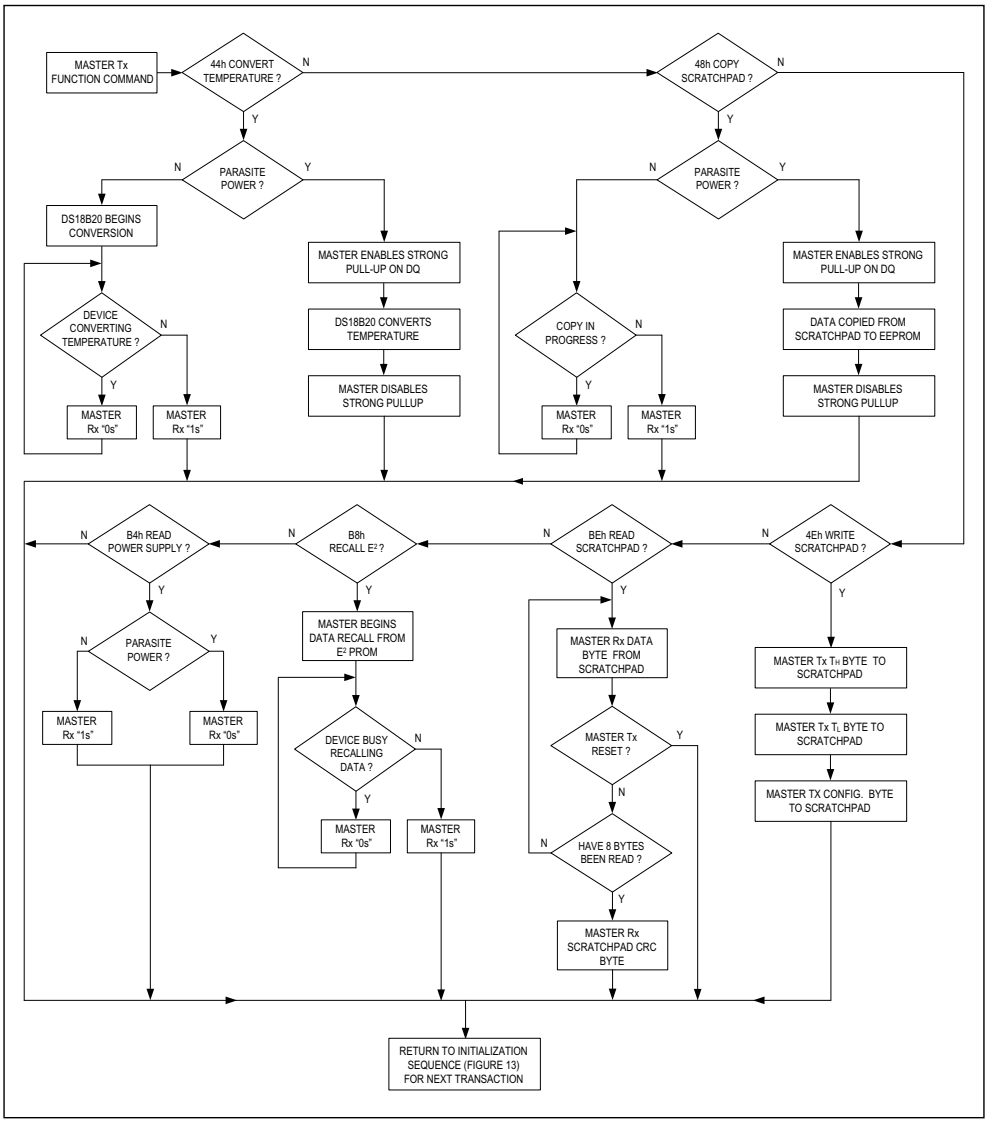


Figure 14. DS18B20 Function Commands Flowchart

1-Wire Signaling

The DS18B20 uses a strict 1-Wire communication protocol to ensure data integrity. Several signal types are defined by this protocol: reset pulse, presence pulse, write 0, write 1, read 0, and read 1. The bus master initiates all these signals, with the exception of the presence pulse.

Initialization Procedure—Reset And Presence Pulses

All communication with the DS18B20 begins with an initialization sequence that consists of a reset pulse from the master followed by a presence pulse from the DS18B20. This is illustrated in Figure 15. When the DS18B20 sends the presence pulse in response to the reset, it is indicating to the master that it is on the bus and ready to operate.

During the initialization sequence the bus master transmits (T_X) the reset pulse by pulling the 1-Wire bus low for a minimum of 480µs. The bus master then releases the bus and goes into receive mode (R_X). When the bus is released, the 5kΩ pullup resistor pulls the 1-Wire bus high. When the DS18B20 detects this rising edge, it waits 15µs to 60µs and then transmits a presence pulse by pulling the 1-Wire bus low for 60µs to 240µs.

Read/Write Time Slots

The bus master writes data to the DS18B20 during write time slots and reads data from the DS18B20 during read time slots. One bit of data is transmitted over the 1-Wire bus per time slot.

Write Time Slots

There are two types of write time slots: “Write 1” time slots and “Write 0” time slots. The bus master uses a Write 1 time slot to write a logic 1 to the DS18B20 and a Write 0 time slot to write a logic 0 to the DS18B20. All write time slots must be a minimum of 60µs in duration with a minimum of a 1µs recovery time between individual write slots. Both types of write time slots are initiated by the master pulling the 1-Wire bus low (see Figure 14).

To generate a Write 1 time slot, after pulling the 1-Wire bus low, the bus master must release the 1-Wire bus within 15µs. When the bus is released, the 5kΩ pullup resistor will pull the bus high. To generate a Write 0 time slot, after pulling the 1-Wire bus low, the bus master must continue to hold the bus low for the duration of the time slot (at least 60µs).

The DS18B20 samples the 1-Wire bus during a window that lasts from 15µs to 60µs after the master initiates the write time slot. If the bus is high during the sampling window, a 1 is written to the DS18B20. If the line is low, a 0 is written to the DS18B20.

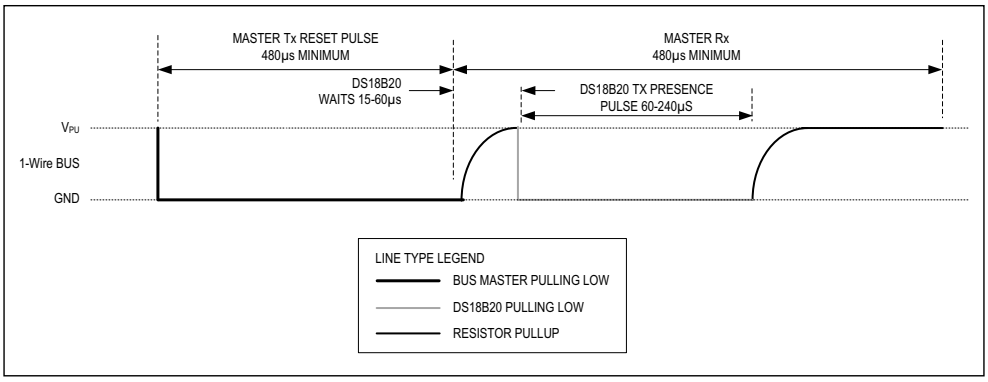


Figure 15. Initialization Timing

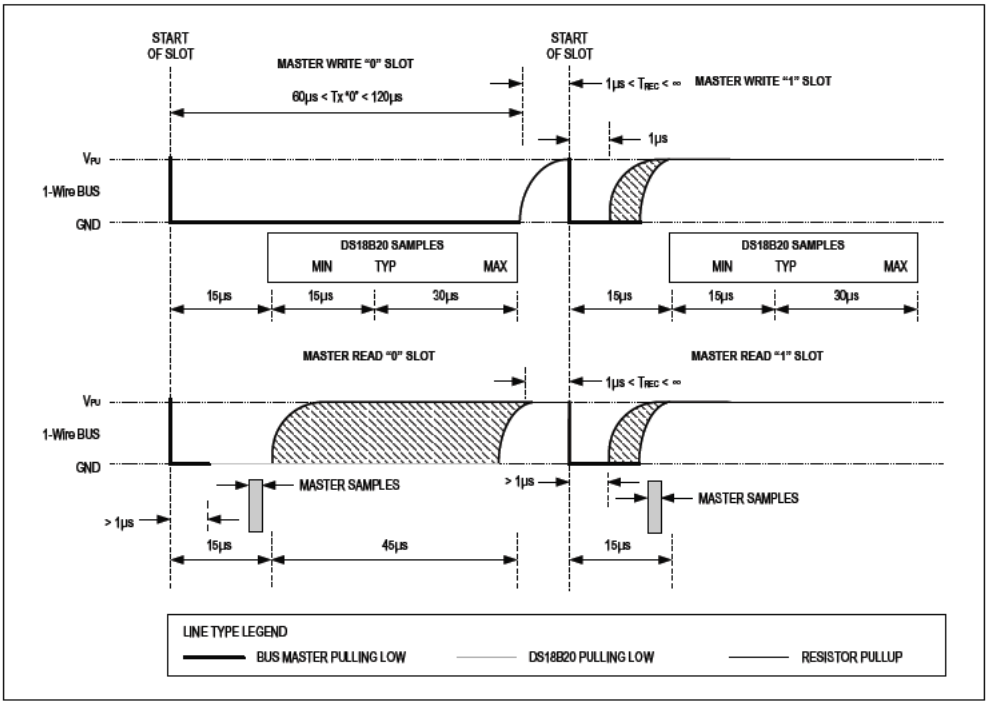


Figure 16. Read/Write Time Slot Timing Diagram

Read Time Slots

The DS18B20 can only transmit data to the master when the master issues read time slots. Therefore, the master must generate read time slots immediately after issuing a Read Scratchpad [BEh] or Read Power Supply [B4h] command, so that the DS18B20 can provide the requested data. In addition, the master can generate read time slots after issuing Convert T [44h] or Recall E^2 [B8h] commands to find out the status of the operation as explained in the *DS18B20 Function Commands* section.

All read time slots must be a minimum of 60µs in duration with a minimum of a 1µs recovery time between slots. A read time slot is initiated by the master device pulling the 1-Wire bus low for a minimum of 1µs and then releasing the bus (see Figure 16). After the master initiates the read time slot, the DS18B20 will begin transmitting a 1 or 0 on bus. The DS18B20 transmits a 1 by leaving the bus high and transmits a 0 by pulling the bus low. When transmitting a 0, the DS18B20 will release the bus by the end of the time slot, and the bus will be pulled back to its high idle state by the pullup resister. Output data from the DS18B20 is valid for 15µs after the falling edge that initiated the read time slot. Therefore, the master must release the bus and then sample the bus state within 15µs from the start of the slot.

Figure 17 illustrates that the sum of T_{INIT}, T_{RC}, and T_{SAMPLE} must be less than 15µs for a read time slot. Figure 18 shows that system timing margin is maximized by keeping T_{INIT} and T_{RC} as short as possible and by locating the master sample time during read time slots towards the end of the 15µs period.

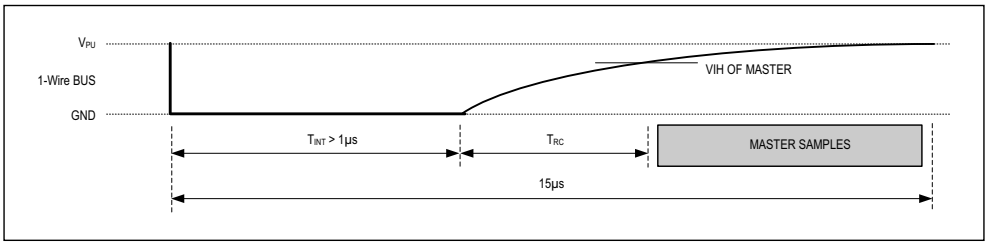


Figure 17. Detailed Master Read 1 Timing

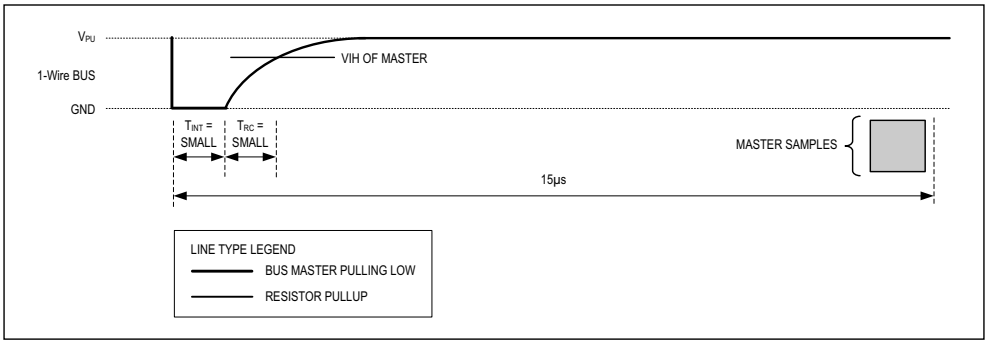


Figure 18. Recommended Master Read 1 Timing

Related Application Notes

The following application notes can be applied to the DS18B20 and are available at www.maximintegrated.com.

Application Note 27: Understanding and Using Cyclic Redundancy Checks with Maxim iButton Products

Application Note 122: Using Dallas' 1-Wire ICs in 1-Cell Li-Ion Battery Packs with Low-Side N-Channel Safety FETs Master

Application Note 126: 1-Wire Communication Through Software

Application Note 162: Interfacing the DS18x20/DS1822 1-Wire Temperature Sensor in a Microcontroller Environment

Application Note 208: Curve Fitting the Error of a Bandgap-Based Digital Temperature Sensor

Application Note 2420: 1-Wire Communication with a Microchip PICmicro Microcontroller

Application Note 3754: Single-Wire Serial Bus Carries Isolated Power and Data

Sample 1-Wire subroutines that can be used in conjunction with A*pplication Note 74: Reading and Writing iButtons via Serial Interfaces* can be downloaded from the Maxim website.

DS18B20 Operation Example 1

In this example there are multiple DS18B20s on the bus and they are using parasite power. The bus master initiates a temperature conversion in a specific DS18B20 and then reads its scratchpad and recalculates the CRC to verify the data.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20s respond with presence pulse.
Tx	55h	Master issues Match ROM command.
Tx	64-bit ROM code	Master sends DS18B20 ROM code.
Tx	44h	Master issues Convert T command.
Tx	DQ line held high by strong pullup	Master applies strong pullup to DQ for the duration of the conversion (t_{CONV}).
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20s respond with presence pulse.
Tx	55h	Master issues Match ROM command.
Tx	64-bit ROM code	Master sends DS18B20 ROM code.
Tx	BEh	Master issues Read Scratchpad command.
Rx	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.

DS18B20 Operation Example 2

In this example there is only one DS18B20 on the bus and it is using parasite power. The master writes to the TH, TL, and configuration registers in the DS18B20 scratchpad and then reads the scratchpad and recalculates the CRC to verify the data. The master then copies the scratchpad contents to EEPROM.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	4Eh	Master issues Write Scratchpad command.
Tx	3 data bytes	Master sends three data bytes to scratchpad (T_H, T_L, and config).
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	BEh	Master issues Read Scratchpad command.
Rx	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	48h	Master issues Copy Scratchpad command.
Tx	DQ line held high by strong pullup	Master applies strong pullup to DQ for at least 10ms while copy operation is in progress.

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE	TOP MARK
DS18B20	-55°C to +125°C	3 TO-92	18B20
DS18B20+	-55°C to +125°C	3 TO-92	18B20
DS18B20/T&R	-55°C to +125°C	3 TO-92 (2000 Piece)	18B20
DS18B20+T&R	-55°C to +125°C	3 TO-92 (2000 Piece)	18B20
DS18B20-SL/T&R	-55°C to +125°C	3 TO-92 (2000 Piece)*	18B20
DS18B20-SL+T&R	-55°C to +125°C	3 TO-92 (2000 Piece)*	18B20
DS18B20U	-55°C to +125°C	8 FSOP	18B20
DS18B20U+	-55°C to +125°C	8 FSOP	18B20
DS18B20U/T&R	-55°C to +125°C	8 FSOP (3000 Piece)	18B20
DS18B20U+T&R	-55°C to +125°C	8 FSOP (3000 Piece)	18B20
DS18B20Z	-55°C to +125°C	8 SO	DS18B20
DS18B20Z+	-55°C to +125°C	8 SO	DS18B20
DS18B20Z/T&R	-55°C to +125°C	8 SO (2500 Piece)	DS18B20
DS18B20Z+T&R	-55°C to +125°C	8 SO (2500 Piece)	DS18B20

+Denotes a lead-free package. A "+" will appear on the top mark of lead-free packages.

T&R = Tape and reel.

**TO-92 packages in tape and reel can be ordered with straight or formed leads. Choose "SL" for straight leads. Bulk TO-92 orders are straight leads only.*

Revision History

REVISION DATE	DESCRIPTION	PAGES CHANGED
3/1/07	In the Absolute Maximum Ratings section, removed the reflow oven temperature value of +220°C. Reference to JEDEC specification for reflow remains.	19
10/12/07	In the *Operation—Alarm Signaling* section, added “or equal to” in the description for a TH alarm condition	5
	In the *Memory* section, removed incorrect text describing memory.	7
	In the *Configuration Register* section, removed incorrect text describing configuration register.	8
4/22/08	In the *Ordering Information* table, added TO-92 straight-lead packages and included a note that the TO-92 package in tape and reel can be ordered with either formed or straight leads.	2
1/15	Updated *Benefits and Features* section	1
09/18	Updated *DC Electrical Characteristics* table	2
7/19	Updated Figure 12	10

A.3. $T(I_P(p))$

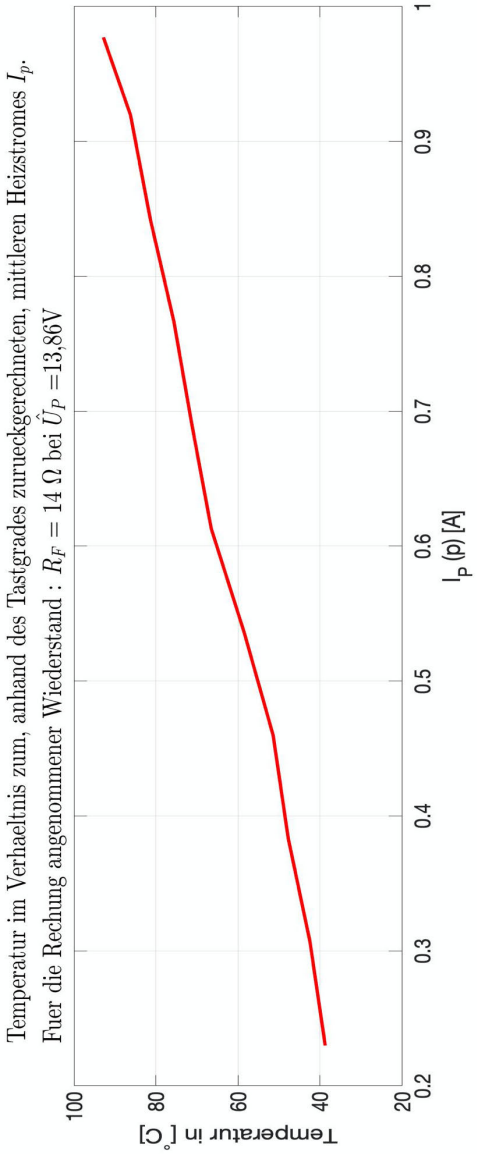


Abbildung A.4.: $T(I_P(p))$

A.4. Programmcode

```
1
2  #include <OneWire.h>
3  #include <DallasTemperature.h>
4  #include <SPI.h>
5  #include <SD.h>
6
7  const int chipSelect = 53;
8  #define ONE_WIRE_BUS 2 // Data wire is plugged into port 2
     on the Arduino
9  #define PWM 9          // PWM wire is plugged into port 9 on
     the Arduino
10 #define MaxTemp 100
11 float TemperaturM1[15] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0};
12 float TemperaturM2[15] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0};
13 float TemperaturM3[15] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0};
14 float TemperaturMR[4] = {0, 0, 0, 0};
15
16 bool Aufheizphase = 1;
17 bool Gultig;
18 int DelayTime = 0;
19 File myFile;
20 long Pause= 0;
21 int Case = 1;
22 char Zeichen;
23 char Temp[4] = {0, 0, 0, 0};
24 char Auflo [2] = {0, 0};
25 float ZielTemp = 30;
26 byte precision = 12;
```

```
27 long currentMillis = 0;
28
29 OneWire oneWire(ONE_WIRE_BUS);
30 DallasTemperature sensors(&oneWire);
31
32 // ---------------arrays to hold device addresses
     ----------------------------
33 DeviceAddress
34
35 T11 = {0x28, 0x2F, 0x33, 0x9E, 0x0B, 0x00, 0x00, 0x73},
36 //T11 = {0x28, 0x6A, 0xB7, 0x24, 0x0B, 0x00, 0x00, 0xB6},
37 T12 = {0x28, 0xF9, 0x15, 0x26, 0x0B, 0x00, 0x00, 0xEC},
38 T13 = {0x28, 0x3F, 0xC5, 0x25, 0x0B, 0x00, 0x00, 0x17},
39 T14 = {0x28, 0xF8, 0x45, 0x26, 0x0B, 0x00, 0x00, 0x4E},
40 T15 = {0x28, 0x1A, 0x5D, 0x26, 0x0B, 0x00, 0x00, 0x4B},
41 T16 = {0x28, 0x41, 0xE4, 0x25, 0x0B, 0x00, 0x00, 0xA9},
42 T17 = {0x28, 0x60, 0xBC, 0x24, 0x0B, 0x00, 0x00, 0x0C},
43 T18 = {0x28, 0xB0, 0xE8, 0x25, 0x0B, 0x00, 0x00, 0xCD},
44 T19 = {0x28, 0x5E, 0xDB, 0x25, 0x0B, 0x00, 0x00, 0x3D},
45 T110 = {0x28, 0x9C, 0x4C, 0x25, 0x0B, 0x00, 0x00, 0x2A},
46 T111 = {0x28, 0xA3, 0xF0, 0x25, 0x0B, 0x00, 0x00, 0x8D},
47 T112 = {0x28, 0x04, 0x47, 0x25, 0x0B, 0x00, 0x00, 0x4A},
48 T113 = {0x28, 0x80, 0xDC, 0x25, 0x0B, 0x00, 0x00, 0xBB},
49 T114 = {0x28, 0x57, 0x36, 0x25, 0x0B, 0x00, 0x00, 0x9D},
50 T115 = {0x28, 0x08, 0x02, 0x25, 0x0B, 0x00, 0x00, 0x0C},
51 T21 = {0x28, 0x39, 0xDD, 0x25, 0x0B, 0x00, 0x00, 0xE7},
52 T22 = {0x28, 0x31, 0x1C, 0x26, 0x0B, 0x00, 0x00, 0x21},
53 T23 = {0x28, 0x47, 0xB8, 0x25, 0x0B, 0x00, 0x00, 0xAF},
54 T24 = {0x28, 0x49, 0xD2, 0x24, 0x0B, 0x00, 0x00, 0x9F},
55 T25 = {0x28, 0x6D, 0xC8, 0x24, 0x0B, 0x00, 0x00, 0x34},
56 T26 = {0x28, 0x71, 0xE6, 0x24, 0x0B, 0x00, 0x00, 0x48},
57 T27 = {0x28, 0x1B, 0x2E, 0x25, 0x0B, 0x00, 0x00, 0xD7},
58 T28 = {0x28, 0xCF, 0xCB, 0x24, 0x0B, 0x00, 0x00, 0x48},
59 T29 = {0x28, 0xEC, 0xB2, 0x24, 0x0B, 0x00, 0x00, 0x39},
```

```
T210 = {0x28, 0x39, 0x36, 0x25, 0x0B, 0x00, 0x00, 0x4D},
T211 = {0x28, 0x92, 0x1B, 0x54, 0x0B, 0x00, 0x00, 0xDA},
T212 = {0x28, 0xF7, 0xA0, 0x25, 0x0B, 0x00, 0x00, 0xEA},
T213 = {0x28, 0xCE, 0x64, 0x25, 0x0B, 0x00, 0x00, 0xAC},
T214 = {0x28, 0xB2, 0xE8, 0x55, 0x0B, 0x00, 0x00, 0x0B},
T215 = {0x28, 0x1D, 0x2B, 0x25, 0x0B, 0x00, 0x00, 0xB7},
T31 = {0x28, 0x77, 0xB8, 0x24, 0x0B, 0x00, 0x00, 0xCD},
T32 = {0x28, 0xB4, 0xE8, 0x24, 0x0B, 0x00, 0x00, 0x9E},
T33 = {0x28, 0xA0, 0xB0, 0x24, 0x0B, 0x00, 0x00, 0xB2},
T34 = {0x28, 0x1F, 0xFE, 0x55, 0x0B, 0x00, 0x00, 0xFD},
T35 = {0x28, 0x58, 0xC8, 0x24, 0x0B, 0x00, 0x00, 0x32},
T36 = {0x28, 0xB7, 0xE7, 0x24, 0x0B, 0x00, 0x00, 0xA8},
T37 = {0x28, 0xED, 0x43, 0x25, 0x0B, 0x00, 0x00, 0xEA},
T38 = {0x28, 0x02, 0x46, 0x25, 0x0B, 0x00, 0x00, 0x35},
T39 = {0x28, 0x9D, 0x49, 0x26, 0x0B, 0x00, 0x00, 0x47},
T310 = {0x28, 0x12, 0xCF, 0x24, 0x0B, 0x00, 0x00, 0xD9},
T311 = {0x28, 0x41, 0xBF, 0x54, 0x0B, 0x00, 0x00, 0x6B},
T312 = {0x28, 0x60, 0x46, 0x25, 0x0B, 0x00, 0x00, 0x98},
T313 = {0x28, 0x11, 0x36, 0x25, 0x0B, 0x00, 0x00, 0x5A},
T314 = {0x28, 0xC9, 0xAF, 0x24, 0x0B, 0x00, 0x00, 0xF4},
T315 = {0x28, 0x68, 0x0C, 0x26, 0x0B, 0x00, 0x00, 0xE5},
TR1 = {0x28, 0x94, 0x09, 0x26, 0x0B, 0x00, 0x00, 0x38},
TR2 = {0x28, 0x9E, 0xAD, 0x25, 0x0B, 0x00, 0x00, 0x53},
TR3 = {0x28, 0x38, 0xA1, 0x25, 0x0B, 0x00, 0x00, 0x9C},
TR4 = {0x28, 0x67, 0x67, 0x26, 0x0B, 0x00, 0x00, 0xA0};
void setup(void) {


  pinMode(PWM, OUTPUT); // initialize PWM pin
  analogWrite(PWM, 0);
  delay(1000);
  Serial.begin(9600); //Start serial port
  Serial.println("Teperaturregeung␣Paneel␣gestartet");

```

```
94  //------------------initialize SD-Card
        ----------------------
95
96  Serial.print("Initialisierung der SD Karte...");
97
98  if (!SD.begin()) {
99    Serial.println("Initialisierung Fehlgeschlagen!");
100   Case=2;
101   return;
102 } else {
103   Serial.println("Initialisierung Abgeschlossen.");
104 }
105
106 // Aufloesung auslesen
107 myFile.close();
108 int i = 0;
109 myFile = SD.open("Auflo.TXT");
110 while (myFile.available()) {
111   Auflo[i] = myFile.read();
112   i++;
113 }
114 precision = atoi(Auflo);
115 Serial.print("Gewaehlte Aufloesung: ");Serial.print(
      precision);Serial.print(" Bit");Serial.print('\n');
116 //Serial.println("HAllo");
117 if (precision == 11 || precision == 12 ) {
118   Gultig = 1;
119   Serial.print("Eingabe gueltig!");
120   if (precision == 11) {
121     DelayTime = 350;
122     Pause = 1500;
123   } else {
124     DelayTime = 750;
125     Pause = 3000;
```

```
126     }
127   } else {
128     Gultig = 0;
129     Serial.print("Eingabe␣ungueltig!");
130     Case=2; // error case
131   }
132
133   myFile.close();
134
135   // Zieltemperatur auslesen
136   i = 0;
137   myFile = SD.open("ZielTemp.txt");
138   while (myFile.available()) {
139     Temp[i] = myFile.read();
140     i++;
141   }
142
143   ZielTemp = atof(Temp);
144   Serial.print('\n');Serial.print("Gewaehlte␣Zieltemperatur:
        ␣");Serial.print(ZielTemp);Serial.print("␣Grad␣Celsius"
        );Serial.print('\n');
145   if (25,00 <= ZielTemp <= 85,00) {
146     Gultig = 1;
147     Serial.print("Eingabe␣gueltig!");
148   } else {
149     Gultig = 0;
150         Serial.print("Eingabe␣ungueltig!");
151     Case =2; // error case
152   }
153   myFile.close();
154 //-----------------Start up the DallasTemperature library
      ---------------------------------
155   sensors.begin();
```

```
156  // ------------locate devices on the bus
        ------------------------------------------------
157  Serial.print('\n');Serial.print("Gefundene␣Sensoren␣am␣Bus
        :␣");
158  //Serial.print(sensors.getDeviceCount(), DEC);
159  Serial.print("45␣Sensoren.");
160  Serial.print('\n');
161  // ----------------------set the defined resolution per
        device---------------------------
162  sensors.setResolution(T11, precision);
163  sensors.setResolution(T12, precision);
164  sensors.setResolution(T13, precision);
165  sensors.setResolution(T14, precision);
166  sensors.setResolution(T15, precision);
167  sensors.setResolution(T16, precision);
168  sensors.setResolution(T17, precision);
169  sensors.setResolution(T18, precision);
170  sensors.setResolution(T19, precision);
171  sensors.setResolution(T110, precision);
172  sensors.setResolution(T111, precision);
173  sensors.setResolution(T112, precision);
174  sensors.setResolution(T113, precision);
175  sensors.setResolution(T114, precision);
176  sensors.setResolution(T115, precision);
177  sensors.setResolution(T21, precision);
178  sensors.setResolution(T22, precision);
179  sensors.setResolution(T23, precision);
180  sensors.setResolution(T24, precision);
181  sensors.setResolution(T25, precision);
182  sensors.setResolution(T26, precision);
183  sensors.setResolution(T27, precision);
184  sensors.setResolution(T28, precision);
185  sensors.setResolution(T29, precision);
186  sensors.setResolution(T210, precision);
```

```
187   sensors.setResolution(T211, precision);
188   sensors.setResolution(T212, precision);
189   sensors.setResolution(T213, precision);
190   sensors.setResolution(T214, precision);
191   sensors.setResolution(T215, precision);
192   sensors.setResolution(T31, precision);
193   sensors.setResolution(T32, precision);
194   sensors.setResolution(T33, precision);
195   sensors.setResolution(T34, precision);
196   sensors.setResolution(T35, precision);
197   sensors.setResolution(T36, precision);
198   sensors.setResolution(T37, precision);
199   sensors.setResolution(T38, precision);
200   sensors.setResolution(T39, precision);
201   sensors.setResolution(T310, precision);
202   sensors.setResolution(T311, precision);
203   sensors.setResolution(T312, precision);
204   sensors.setResolution(T313, precision);
205   sensors.setResolution(T314, precision);
206   sensors.setResolution(T315, precision);
207
208 }
209 // ------------------------function to get back the
      temperature of a device-------------------
210 float printTemperature(DeviceAddress deviceAddress) {
211   float tempC = sensors.getTempC(deviceAddress);
212   if (tempC == -127.00) {
213     Serial.print("Temperaturabfrage␣Fehlgeschlagen!␣␣");
        Serial.println();
214   }
215   return tempC;
216 }
217 // ---------------------function to find highest number of
      15 -------------------
```

```
218 float findHighest(float arrey[15]) {
219   float highest = 0;
220   for (int i = 0; i < 15; i++) {
221     if (arrey[i] > highest) {
222       highest = arrey[i];
223     }
224   }
225 
226   return highest;
227 }
228 // ---------------------function to find highest number of
      four -------------------
229 float findHighestOfFour(float arrey[4]) {
230   float highest;
231   if (arrey[0] > arrey[1] && arrey[0] > arrey[2] && arrey[0]
        > arrey[3]) {
232     highest = arrey[0];
233   } else if (arrey[1] > arrey[2] && arrey[1] > arrey[3]) {
234     highest = arrey[1];
235   } else if (arrey[2] > arrey[3]) {
236     highest = arrey[2];
237   } else {
238     highest = arrey[3];
239   }
240   return highest;
241 }
242 //--------------------- Funktion, um Messfehler
      auszugleichen------------
243 float TempAnpassen(float alteTemperatur) {
244   float angepassteTemp = alteTemperatur * 0.816 - 16.503 +
       alteTemperatur;
245   return angepassteTemp;
246 }
247 //------------------- Loop----------------
```

```
248 long Zeitstempel = -1;
249 void loop(void) {
250   long currentMillis = millis();
251
252   if (Zeitstempel < (currentMillis - Pause)) { //Wenn der
        Zeitstempel der letzten Ausfuehrung kleiner als der
        aktuelle Zeitstempel minus der Pause ist dann soll der
        Code ausgefuehrt werden. -> Feste Tastrate
253     Zeitstempel = millis();
254     switch (Case) {
255       case 1:
256         //Ueberschreiben des alten Zeitstempels mit dem
              neuen Wert.
257         Serial.print("Temperaturen werden abgefragt...Array
              1,2 & 3: "); sensors.requestTemperatures();delay(
              DelayTime);Serial.println("Fertig");
258
259         //--------------- print and safe the device
              temperature and the time of Measurement, Matrix
              1-------------------------------------------------------

260         for (int k = 0; k < 45; k++) {
261           if (k == 0) {
262             TemperaturM1[0] = printTemperature(T11); Serial.
                  print("Sensor 1.1 "); Serial.print(" ");
                  Serial.print(TemperaturM1[0]); Serial.print("
                  C"); Serial.println();
263           } else if (k == 1) {
264             TemperaturM1[1] = printTemperature(T12); Serial.
                  print("Sensor 1.2 "); Serial.print(" ");
                  Serial.print(TemperaturM1[1]); Serial.print("
                  C"); Serial.println();
265           } else if (k == 2) {
266             TemperaturM1[2] = printTemperature(T13); Serial.
```

```
                print("Sensor 1.3 "); Serial.print(" ");
                Serial.print(TemperaturM1[2]); Serial.print("
                C"); Serial.println();
267           } else if (k == 3) {
268             TemperaturM1[3] = printTemperature(T14); Serial.
                print("Sensor 1.4 "); Serial.print(" ");
                Serial.print(TemperaturM1[3]); Serial.print("
                C"); Serial.println();
269           } else if (k == 4) {
270             TemperaturM1[4] = printTemperature(T15); Serial.
                print("Sensor 1.5 "); Serial.print(" ");
                Serial.print(TemperaturM1[4]); Serial.print("
                C"); Serial.println();
271           } else if (k == 5) {
272             TemperaturM1[5] = printTemperature(T16); Serial.
                print("Sensor 1.6 "); Serial.print(" ");
                Serial.print(TemperaturM1[5]); Serial.print("
                C"); Serial.println();
273           } else if (k == 6) {
274             TemperaturM1[6] = printTemperature(T17); Serial.
                print("Sensor 1.7 "); Serial.print(" ");
                Serial.print(TemperaturM1[6]); Serial.print("
                C"); Serial.println();
275           } else if (k == 7) {
276             TemperaturM1[7] = printTemperature(T18); Serial.
                print("Sensor 1.8 "); Serial.print(" ");
                Serial.print(TemperaturM1[7]); Serial.print("
                C"); Serial.println();
277           } else if (k == 8) {
278             TemperaturM1[8] = printTemperature(T19); Serial.
                print("Sensor 1.9 "); Serial.print(" ");
                Serial.print(TemperaturM1[8]); Serial.print("
                C"); Serial.println();
279           } else if (k == 9) {
```

```
280             TemperaturM1[9] = printTemperature(T110); Serial
                    .print("Sensor␣1.10␣"); Serial.print("␣");
                    Serial.print(TemperaturM1[9]); Serial.print("
                    C"); Serial.println();
281           } else if (k == 10) {
282             TemperaturM1[10] = printTemperature(T111);
                    Serial.print("Sensor␣1.11␣"); Serial.print("␣
                    "); Serial.print(TemperaturM1[10]); Serial.
                    print("C"); Serial.println();
283           } else if (k == 11) {
284             TemperaturM1[11] = printTemperature(T112);
                    Serial.print("Sensor␣1.12␣"); Serial.print("␣
                    "); Serial.print(TemperaturM1[11]); Serial.
                    print("C"); Serial.println();
285           } else if (k == 12) {
286             TemperaturM1[12] = printTemperature(T113);
                    Serial.print("Sensor␣1.13␣"); Serial.print("␣
                    "); Serial.print(TemperaturM1[12]); Serial.
                    print("C"); Serial.println();
287           } else if (k == 13) {
288             TemperaturM1[13] = printTemperature(T114);
                    Serial.print("Sensor␣1.14␣"); Serial.print("␣
                    "); Serial.print(TemperaturM1[13]); Serial.
                    print("C"); Serial.println();
289           } else if (k == 14) {
290             TemperaturM1[14] = printTemperature(T115);
                    Serial.print("Sensor␣1.15␣"); Serial.print("␣
                    "); Serial.print(TemperaturM1[14]); Serial.
                    print("C"); Serial.println();
291           } else if (k == 15) {
292             TemperaturM2[0] = printTemperature(T21); Serial.
                    print("Sensor␣2.1␣"); Serial.print("␣");
                    Serial.print(TemperaturM2[0]); Serial.print("
                    C"); Serial.println();
```

```
293             } else if (k == 16) {
294               TemperaturM2[1] = printTemperature(T22); Serial.
                     print("Sensor 2.2 "); Serial.print(" ");
                     Serial.print(TemperaturM2[1]); Serial.print("
                     C"); Serial.println();
295             } else if (k == 17) {
296               TemperaturM2[2] = printTemperature(T23); Serial.
                     print("Sensor 2.3 "); Serial.print(" ");
                     Serial.print(TemperaturM2[2]); Serial.print("
                     C"); Serial.println();
297             } else if (k == 18) {
298               TemperaturM2[3] = printTemperature(T24); Serial.
                     print("Sensor 2.4 "); Serial.print(" ");
                     Serial.print(TemperaturM2[3]); Serial.print("
                     C"); Serial.println();
299             } else if (k == 19) {
300               TemperaturM2[4] = printTemperature(T25); Serial.
                     print("Sensor 2.5 "); Serial.print(" ");
                     Serial.print(TemperaturM2[4]); Serial.print("
                     C"); Serial.println();
301             } else if (k == 20) {
302               TemperaturM2[5] = printTemperature(T26); Serial.
                     print("Sensor 2.6 "); Serial.print(" ");
                     Serial.print(TemperaturM2[5]); Serial.print("
                     C"); Serial.println();
303             } else if (k == 21) {
304               TemperaturM2[6] = printTemperature(T27); Serial.
                     print("Sensor 2.7 "); Serial.print(" ");
                     Serial.print(TemperaturM2[6]); Serial.print("
                     C"); Serial.println();
305             } else if (k == 22) {
306               TemperaturM2[7] = printTemperature(T28); Serial.
                     print("Sensor 2.8 "); Serial.print(" ");
```

```
                Serial.print(TemperaturM2[7]); Serial.print("
                C"); Serial.println();
307           } else if (k == 23) {
308             TemperaturM2[8] = printTemperature(T29); Serial.
                print("Sensor 2.9 "); Serial.print(" ");
                Serial.print(TemperaturM2[8]); Serial.print("
                C"); Serial.println();
309           } else if (k == 24) {
310             TemperaturM2[9] = printTemperature(T210); Serial
                .print("Sensor 2.10 "); Serial.print(" ");
                Serial.print(TemperaturM2[9]); Serial.prinZ("
                C"); Serial.println();
311           } else if (k == 25) {
312             TemperaturM2[10] = printTemperature(T211);
                Serial.print("Sensor 2.11 "); Serial.print(" 
                "); Serial.print(TemperaturM2[10]); Serial.
                print("C"); Serial.println();
313           } else if (k == 26) {
314             TemperaturM2[11] = printTemperature(T212);
                Serial.print("Sensor 2.12 "); Serial.print(" 
                "); Serial.print(TemperaturM2[11]); Serial.
                print("C"); Serial.println();
315           } else if (k == 27) {
316             TemperaturM2[12] = printTemperature(T213);
                Serial.print("Sensor 2.13 "); Serial.print(" 
                "); Serial.print(TemperaturM2[12]); Serial.
                print("C"); Serial.println();
317           } else if (k == 28) {
318             TemperaturM2[13] = printTemperature(T214);
                Serial.print("Sensor 2.14 "); Serial.print(" 
                "); Serial.print(TemperaturM2[13]); Serial.
                print("C"); Serial.println();
319           } else if (k == 29) {
320             TemperaturM2[14] = printTemperature(T215);
```

```
                Serial.print("Sensor 2.15 "); Serial.print("
                "); Serial.print(TemperaturM2[14]); Serial.
                print("C"); Serial.println();
321           } else if (k == 30) {
322             TemperaturM3[0] = printTemperature(T31); Serial.
                print("Sensor 3.1 "); Serial.print(" ");
                Serial.print(TemperaturM3[0]); Serial.print("
                C"); Serial.println();
323           } else if (k == 31) {
324             TemperaturM3[1] = printTemperature(T32); Serial.
                print("Sensor 3.2 "); Serial.print(" ");
                Serial.print(TemperaturM3[1]); Serial.print("
                C"); Serial.println();
325           } else if (k == 32) {
326             TemperaturM3[2] = printTemperature(T33); Serial.
                print("Sensor 3.3 "); Serial.print(" ");
                Serial.print(TemperaturM3[2]); Serial.print("
                C"); Serial.println();
327           } else if (k == 33) {
328             TemperaturM3[3] = printTemperature(T34); Serial.
                print("Sensor 3.4 "); Serial.print(" ");
                Serial.print(TemperaturM3[3]); Serial.print("
                C"); Serial.println();
329           } else if (k == 34) {
330             TemperaturM3[4] = printTemperature(T35); Serial.
                print("Sensor 3.5 "); Serial.print(" ");
                Serial.print(TemperaturM3[4]); Serial.print("
                C"); Serial.println();
331           } else if (k == 35) {
332             TemperaturM3[5] = printTemperature(T36); Serial.
                print("Sensor 3.6 "); Serial.print(" ");
                Serial.print(TemperaturM3[5]); Serial.print("
                C"); Serial.println();
333           } else if (k == 36) {
```

```
334             TemperaturM3[6] = printTemperature(T37); Serial.
                    print("Sensor 3.7 "); Serial.print(" ");
                    Serial.print(TemperaturM3[6]); Serial.print("
                    C"); Serial.println();
335           } else if (k == 37) {
336             TemperaturM3[7] = printTemperature(T38); Serial.
                    print("Sensor 3.8 "); Serial.print(" ");
                    Serial.print(TemperaturM3[7]); Serial.print("
                    C"); Serial.println();
337           } else if (k == 38) {
338             TemperaturM3[8] = printTemperature(T39); Serial.
                    print("Sensor 3.9 "); Serial.print(" ");
                    Serial.print(TemperaturM3[8]); Serial.print("
                    C"); Serial.println();
339           } else if (k == 39) {
340             TemperaturM3[9] = printTemperature(T310); Serial
                    .print("Sensor 3.10 "); Serial.print(" ");
                    Serial.print(TemperaturM3[9]); Serial.print("
                    C"); Serial.println();
341           } else if (k == 40) {
342             TemperaturM3[10] = printTemperature(T311);
                    Serial.print("Sensor 3.11 "); Serial.print(" 
                    "); Serial.print(TemperaturM3[10]); Serial.
                    print("C"); Serial.println();
343           } else if (k == 41) {
344             TemperaturM3[11] = printTemperature(T312);
                    Serial.print("Sensor 3.12 "); Serial.print(" 
                    "); Serial.print(TemperaturM3[11]); Serial.
                    print("C"); Serial.println();
345           } else if (k == 42) {
346             TemperaturM3[12] = printTemperature(T313);
                    Serial.print("Sensor 3.13 "); Serial.print(" 
                    "); Serial.print(TemperaturM3[12]); Serial.
                    print("C"); Serial.println();
```

```
347           } else if (k == 43) {
348             TemperaturM3[13] = printTemperature(T314);
                  Serial.print("Sensor 3.14 "); Serial.print("
                  "); Serial.print(TemperaturM3[13]); Serial.
                  print("C"); Serial.println();
349           } else if (k == 44) {
350             TemperaturM3[14] = printTemperature(T315);
                  Serial.print("Sensor 3.15 "); Serial.print("
                  "); Serial.print(TemperaturM3[14]); Serial.
                  print("C"); Serial.println();
351           }
352         }
353         //----------------------------Spannung anpassen
              -------------------------------------
354         if (Aufheizphase) {         //Konstanter Tastgrad von
              p=0,75 in Heizphase
355           analogWrite(PWM, 191);
356           Serial.println("Heizphase");
357           Serial.println(TempAnpassen(findHighest(
                TemperaturM1)));
358         }
359         if (Aufheizphase & ((TempAnpassen(findHighest(
              TemperaturM1)) >= ZielTemp) || (TempAnpassen(
              findHighest(TemperaturM2)) >= ZielTemp) || (
              TempAnpassen(findHighest(TemperaturM3)) >=
              ZielTemp))) {            //Aufheizphase
              abgeschlossen
360           Aufheizphase = false;
361           Serial.println("Heizphase abgeschlossen");
362         }
363         if (!(Aufheizphase)& ((TempAnpassen(findHighest(
              TemperaturM1)) < (ZielTemp - 0.5)) && (
              TempAnpassen(findHighest(TemperaturM2)) < (
              ZielTemp - 0.5)) && (TempAnpassen(findHighest(
```

```
            TemperaturM3)) < (ZielTemp - 0.5)))) {
                        //Zieltemperatur auf Oberflaeche
            unterschritten
364         if (ZielTemp < 40) {
365           analogWrite(PWM, 80);
366           Serial.println("Haltephase-␣Bereich␣bis␣40C");
367           Serial.println(TempAnpassen(findHighest(
                TemperaturM1)));
368         }
369         if ((ZielTemp >= 40) && (ZielTemp < 60)) {
370           analogWrite(PWM, 127);
371           Serial.println("Haltephase-␣Bereich␣bis␣40C␣-␣60
                C␣");
372           Serial.println(TempAnpassen(findHighest(
                TemperaturM1)));
373         }
374         if ((ZielTemp >= 60)) {
375           analogWrite(PWM, 191);
376           Serial.println("Haltephase-␣Bereich␣ueber␣60C␣")
                ;
377           Serial.println(TempAnpassen(findHighest(
                TemperaturM1)));
378         }
379       }
380       if (((TempAnpassen(findHighest(TemperaturM1)) >=
            ZielTemp) || (TempAnpassen(findHighest(
            TemperaturM2)) >= ZielTemp) || (TempAnpassen(
            findHighest(TemperaturM3)) >= ZielTemp))) {
                        //Zieltemperatur auf Oberflaeche
            ueberschritten
381         analogWrite(PWM, 0);
382         Serial.println("Zieltemperatur␣auf␣Oberflaeche␣
              ueberschritten");
```

```
383           Serial.println(TempAnpassen(findHighest(
                  TemperaturM1)));
384         }
385
386
387         //---------------------------Daten Speichern
                 ----------------------
388         myFile = SD.open("Daten.txt", FILE_WRITE);
389         if (myFile) {                              // if the
                 file opened okay, write to it:
390           Serial.print("Writing␣to␣Daten.txt...");
391           for (int k = 0; k < 15; k++) {
392             myFile.print(TemperaturM1[k]); myFile.print("␣")
                    ;
393           }
394           for (int k = 0; k < 15; k++) {
395             myFile.print(TemperaturM2[k]); myFile.print("␣")
                    ;
396           }
397           for (int k = 0; k < 15; k++) {
398             myFile.print(TemperaturM3[k]); myFile.print("␣")
                    ;
399           }
400           myFile.print("␣");
401           myFile.print(Zeitstempel);
402           //          myFile.print(" ");
403           //          myFile.print(Ampere);
404           myFile.println();
405           // close the file:
406           myFile.close();
407           Serial.println("done.");
408         } else {
409           // if the file didn't open, print an error:
410           Serial.println("error␣opening␣Daten.txt");
```

```
411           }
412           Serial.println("Ende");
413           break;
414         case 2:
415           Serial.println("Ungluelтige␣Benutzereingaben␣erkannt
                ␣oder␣Fehlerhafte␣Initialisierung␣der␣SD-Karte!␣
                Eingaben␣und␣SD-Karte␣ueberpruefen␣und␣Programm␣
                neu␣starten.␣");
416           delay(1500);
417           break;
418       }
419     }
420 }
```

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: Thorben

Vorname: Strübing

dass ich die vorliegende Bachelorarbeit bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:
Entwurf, Aufbau und Test einer adaptiven Heizsteuerung für ein multifunktionales Leichtbaupaneel

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der -bitte auswählen- ist erfolgt durch:

Hamburg	17.04.2020	[redacted]
Ort	Datum	Unterschrift im Original