Projekt Software Engineering Temperatureinstellung und Steuerung eines Backofens

Lukas Kraft

28. Mai 2025

Einleitung

Dieses Dokument beschreibt die funktionalen und sicherheitsrelevanten Anforderungen an die Steuerung und Temperatureinstellung eines elektrischen Haushaltsbackofens. Ziel ist es, eine benutzerfreundliche, sichere und zuverlässige Ofensteuerung zu realisieren.

1 Annahmen

Für die nachfolgenden Anforderungen gelten die folgenden Systemannahmen, auf denen das Design, die Funktion und die Umsetzung der Steuerung basieren:

- Das System verfügt über ein Display, das folgende Informationen anzeigen kann:
  - Aktuelle Temperatur im Garraum
  - Symbol für den Vorheizstatus
  - Aktuell ausgewählte Heizart
  - Uhrzeit bzw. Restlaufzeit eines Timers
  - Warnung bei automatischer Abschaltung des Heizvorgangs
- Zur Bedienung stehen zwei physische Drehregler zur Verfügung:
  - Ein Regler zur Auswahl der Betriebsart (Heizfunktion)
  - Ein Regler zur Einstellung der Temperatur
- Der Backofen verfügt über drei separat ansteuerbare Heizaggregate:
  - Oberer Heizkörper („Rohrhitzeaggregat“) mit einer Leistung von 3000 W
  - Unterer Heizkörper mit einer Leistung von 1500 W
  - Ringheizkörper an der Rückseite mit einer Leistung von 2000 W
- Ein Ventilator befindet sich an der Rückseite des Garraums und ist separat schaltbar (Ein/Aus).
- Ein Türkontaktsensor ist verbaut, der den Zustand der Backofentür (offen/geschlossen) erkennt.
- Ein Thermometer zur Messung der aktuellen Backofentemperatur ist vorhanden.
- Alle Heizaggregate sind ausschließlich binär schaltbar (Ein/Aus).
- Der Ventilator ist ebenfalls ausschließlich binär schaltbar (Ein/Aus).

2 Requirements Engineering

2.1 Funktionale Anforderungen

R-1.1 Das System muss es dem Benutzer ermöglichen, die Temperatur über einen Drehregler einzustellen.

R-1.2 Das System muss Temperatureinstellungen in 1 °C-Schritten zulassen.

R-1.3 Das System muss Temperatureinstellungen im Bereich von 50 °C bis 300 °C unterstützen.

R-1.4 Das System muss dem Benutzer erlauben, eine Betriebsart über einen separaten Drehregler auszuwählen.

R-1.5 Folgende Betriebsmodi müssen wählbar sein:

- Ober-/Unterhitze
- Oberhitze
- Unterhitze
- Grillfunktion
- Umluft
- Heißluft

R-1.6 Das System muss die Heizaggregate entsprechend der gewählten Betriebsart automatisch aktivieren:

- Ober-/Unterhitze: Oberer und unterer Heizkörper
- Oberhitze: Oberer Heizkörper
- Unterhitze: Unterer Heizkörper
- Grillfunktion: Oberer Heizkörper
- Umluft: Oberer und unterer Heizkörper + Ventilator
- Heißluft: Ringheizkörper hinten + Ventilator

R-1.7 Das System muss die Heizaggregate einzeln ansteuern können.

R-1.8 Das System muss die Temperaturregelung mit einer Abtastrate von 1 Hz durchführen.

R-1.9 Das System muss alle Heizaggregate deaktivieren, wenn die aktuelle Temperatur gleich oder größer der Solltemperatur ist.

R-1.10 Das System muss die gemäß Betriebsart definierten Heizaggregate aktivieren, wenn die aktuelle Temperatur unterhalb der Solltemperatur liegt.

R-1.11 Im Grillmodus muss das System die eingestellte Temperatur intern in vier Grillstufen umwandeln:

- Stufe 1: bis einschließlich 240 °C
- Stufe 2: bis einschließlich 260 °C
- Stufe 3: bis einschließlich 280 °C

- Stufe 4: bis einschließlich 300 °C

R-1.12 Das System muss dem Benutzer folgende Informationen über ein Display anzeigen:

- Aktuelle Temperatur im Garraum
- Vorheizstatus (z. B. Symbol, wenn Solltemperatur erreicht)
- Restzeit eines eingestellten Timers
- Warnung bei automatischer Abschaltung

R-1.13 Das System muss die Heizfunktion automatisch deaktivieren, wenn ein Timer abgelaufen ist.

2.2 Sicherheitsanforderungen

R-2.1 Das System muss den Heizbetrieb automatisch abschalten, wenn die Temperatur 320 °C überschreitet.

R-2.2 Das System darf den Heizbetrieb nicht aktivieren, wenn die Backofentür geöffnet ist.

R-2.3 Das System muss eine Fehlfunktion eines Heizaggregats erkennen, wenn:

- die Temperatur innerhalb von 10 Sekunden (10 Abtastzyklen) um weniger als 1 °C steigt und
- gleichzeitig die Ist-Temperatur mehr als 10 % unter der Solltemperatur liegt.

In diesem Fall muss eine Warnung ausgegeben oder der Fehler im Systemprotokoll erfasst werden.

2.3 Nicht-funktionale Anforderungen

R-3.1 Das System muss in der Lage sein, die Solltemperatur von 200 °C innerhalb von maximal 10 Minuten zu erreichen.

R-3.2 Das Display muss Statusänderungen (z. B. Temperatur, Timer) innerhalb von 1 Sekunde anzeigen.

R-3.3 Das System muss alle sicherheitsrelevanten Zustandsänderungen und Fehlerereignisse mit Zeitstempel in einer Logdatei protokollieren.

3 Software Architektur

Im Folgenden werden alle Module des Systems beschrieben und deren jeweilige Funktionalität erläutert. Die strukturellen Abhängigkeiten zwischen den Modulen sind in Abbildung 1 visuell dargestellt.

Das generelle Ziel ist, die verschiedenen Aufgaben in einzelne Module zu unterteilen um alles klar und verständlich zu verteilen. Außerdem werden alle gegebenen Hardwarekomponenten wie Sensoren oder Heizkörper in eigenen Klassen und Modulen gekapselt. Ebenso wie die Informationen die zwischen Modulen geschickt werden. Diese Klassen werden in den Global Models definiert und überall im Code hergenommen. Ebenso wird das Logging in ein eigenes Modul ausgelagert um dies aus dem Produktivcode herauszulassen.

Main

Das `Main`-Modul bildet den Einstiegspunkt des Programms. Es initialisiert das Gesamtsystem und erstellt eine Instanz des zentralen Steuerungsmoduls `OvenRunner`.

OvenRunner

Der `OvenRunner` ist die zentrale Steuereinheit der Anwendung. Er übernimmt die Koordination aller funktionalen Module innerhalb einer kontinuierlich laufenden Hauptschleife. Zu seinen Aufgaben gehören:

- Aufruf und Steuerung aller relevanten Module (z. B. `InputHandlerModule`, `ModeControlModule`, `OutputHandlerModule`)
- Initialisierung und Start des `SafetyModule` in einem separaten Thread
- Start eines Timers nach Empfang eines entsprechenden Signals vom `InputHandlerModule`

Der genaue Ablauf dieser Kontrollschleife ist in Abbildung 2 dargestellt.

SafetyModule

Das `SafetyModule` führt kontinuierlich Sicherheitsüberprüfungen durch (z. B. Temperaturüberschreitung, geöffnete Tür, Ausfall eines Heizaggregats). Es wird in einem eigenen Thread betrieben und läuft parallel zu den übrigen Modulen.

InputHandlerModule

Dieses Modul erfasst und verarbeitet sämtliche Benutzereingaben. Es liest die Drehregler für Temperatur und Betriebsmodus aus und erkennt, ob ein Timerwert eingestellt wurde. Die daraus resultierenden Informationen werden an den `OvenRunner` zurückgemeldet.

ModeControlModule

Das `ModeControlModule` interpretiert die übergebenen Eingabewerte und leitet basierend darauf die Steuerung der Heizaggregate ein. Die konkrete Aktivierung erfolgt über das `ThermalControlModule`.

ThermalControlModule

Im `ThermalControlModule` ist die Logik zur Steuerung aller Heizaggregate und des Ventilators implementiert. Es bildet die Schnittstelle zur Hardware und übernimmt das gezielte Aktivieren oder Deaktivieren der Komponenten gemäß dem gewählten Betriebsmodus.

OutputHandlerModule

Das `OutputHandlerModule` ist für die Darstellung aller relevanten Systeminformationen auf dem Display verantwortlich. Dazu zählen insbesondere:

- Aktuelle Temperatur
- Eingestellter Betriebsmodus
- Status des Timers
- Sicherheits- und Warnhinweise (z. B. automatische Abschaltung)

SensorModule

Das `SensorModule` stellt die Anbindung und Abfrage der physischen Sensoren des Backofens bereit. Es enthält unter anderem:

- einen `TemperatureSensor`, der die aktuelle Temperatur im Garraum liefert
- einen `DoorSensor`, der überprüft, ob die Backofentür geöffnet oder geschlossen ist

GlobalModels

Das `GlobalModels`-Modul kapselt zentrale Datenstrukturen, die zur systemweiten Kommunikation und Zustandshaltung verwendet werden. Dazu gehören unter anderem:

- `InputValues`: Enthält Temperatur, Modus und Timerinformationen aus der Benutzereingabe
- `CookingMode`: Definiert alle verfügbaren Betriebsmodi des Backofens (z. B. Grill, Umluft, Ober-/Unterhitze)
- `OvenState`: Hält den aktuellen Zustand des Backofens (z. B. `Idle`, `Preheating`, `Active`)

LoggingModule

Das `LoggingModule` verwaltet die zentrale Protokollierung aller sicherheitsrelevanten Ereignisse und Zustandsänderungen. Es wird von sämtlichen Modulen aufgerufen und speichert alle Einträge mit einem Zeitstempel. Üblicherweise ist dieses Modul als Singleton implementiert, um globalen Zugriff zu gewährleisten.

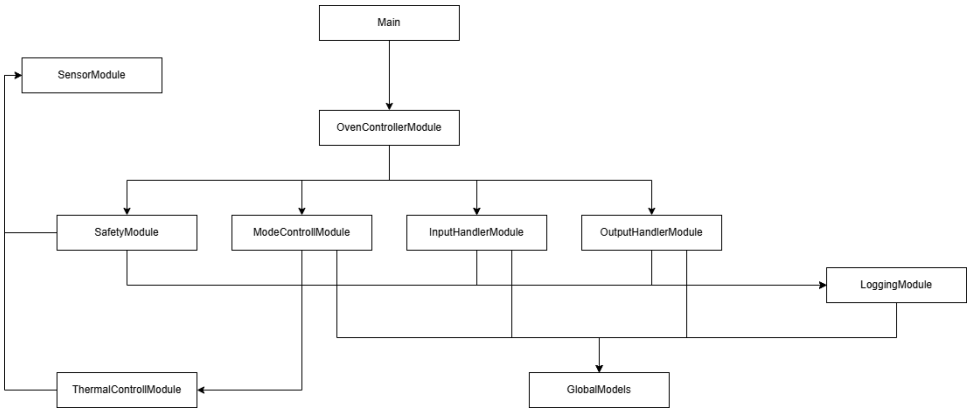

Abbildung 1: Modulübersicht des Systems

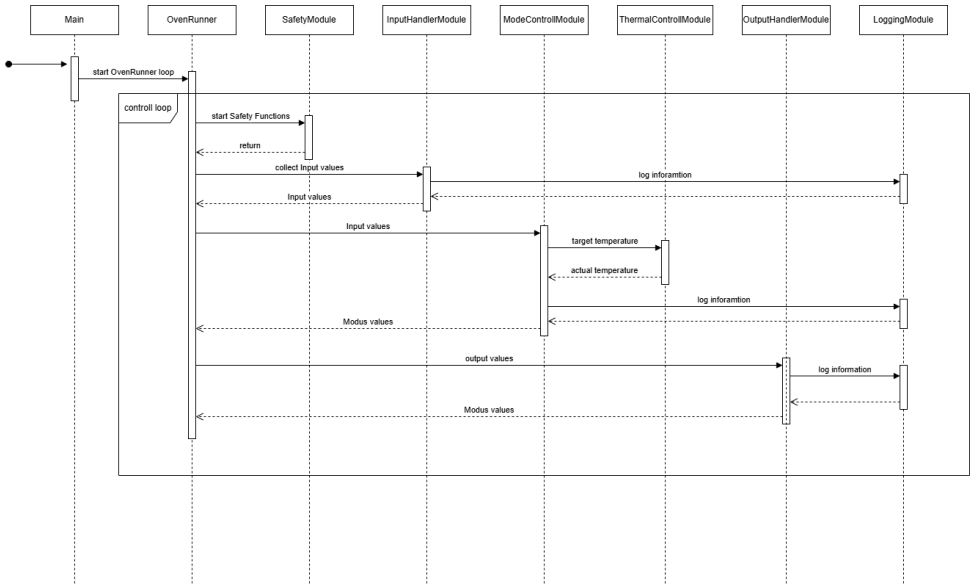

Abbildung 2: Kontrollschleife des OvenRunners

4 Software Design

Im Folgenden werden alle zentralen Module des Backofensystems beschrieben. Jedes Modul erfüllt eine klar definierte Aufgabe und folgt etablierten Software-Designprinzipien wie beispiel den *Strategy Pattern*, *State Pattern* oder der Trennung von Sensorik, Steuerung und Darstellung.

GlobalModels

Das Modul `GlobalModels` stellt die grundlegenden Datenmodelle und Enums bereit, die als gemeinsame Kommunikationsstruktur zwischen den Modulen dienen.

Das zentrale Eingabeobjekt ist `InputValues`, welches die drei wesentlichen Benutzereingaben enthält: die gewünschte Temperatur (als `double`), den gewählten Modus des Backens (als `CookingMode`-Enum) sowie die Timerdauer (als `TimeSpan`).

Analog dazu enthält die Klasse `OutputValues` alle Informationen, die dem Benutzer angezeigt werden sollen – also die aktuelle Temperatur, eine Markierung, ob die Zieltemperatur erreicht ist, der aktuelle Timerstatus sowie ein Flag, ob eine Warnung angezeigt werden soll.

Die Enumeration `CookingMode` beschreibt die verschiedenen verfügbaren Heizarten des Ofens (z. B. Grill, Umluft, Ober-/Unterhitze).

Der `OvenState`-Enum definiert die internen Zustände der Gerätesteuerung, wie z. B. `Idle`, `Preheating`, `Active`, `Error` und `Off`.

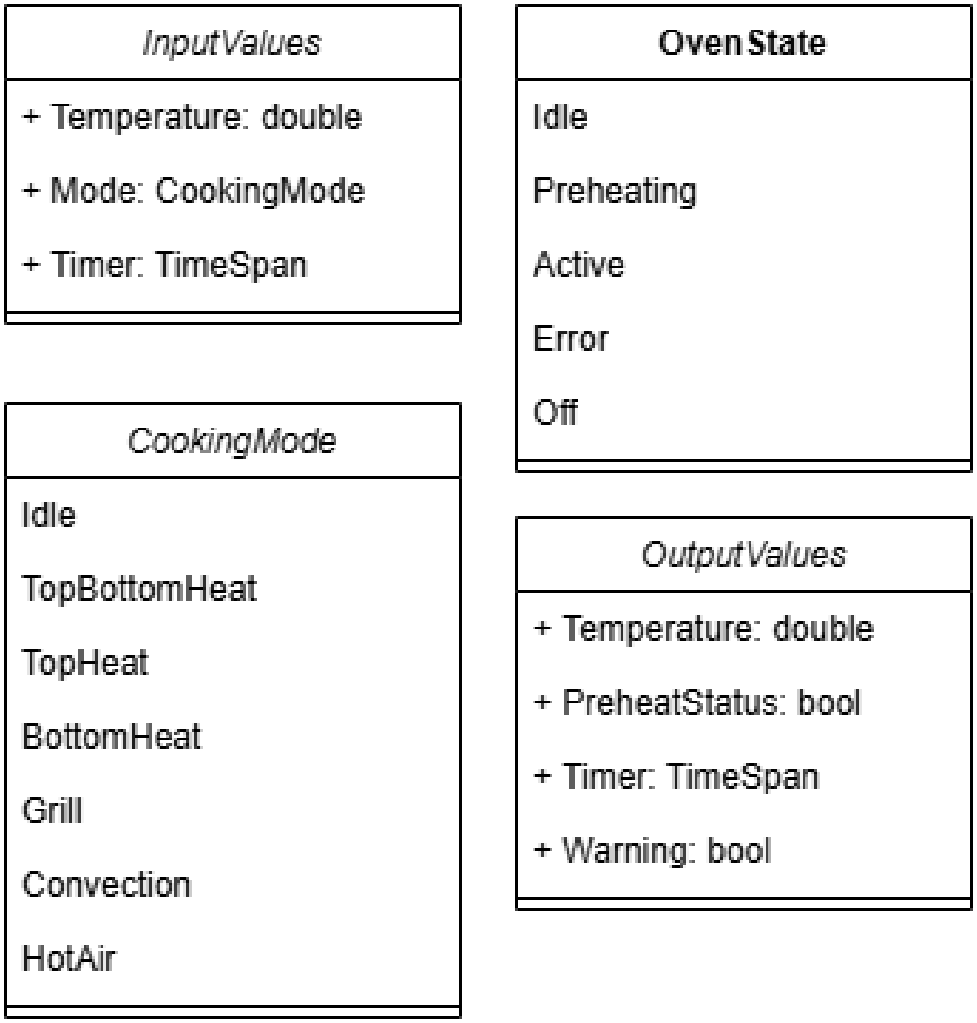

Abbildung 3: Klassendiagram der Global Models

Main Modul

Das `Main`-Modul stellt den Einstiegspunkt des Systems dar. Es enthält die Main-Methode, welche beim Programmstart automatisch ausgeführt wird.

Im Rahmen dieser Methode wird die zentrale Komponente des Backofensystems, der `OvenController`, initialisiert und anschließend die `Run()`-Methode aufgerufen, um den Prozess zu starten.

Das `Main`-Modul hat somit die alleinige Aufgabe, das System einmalig korrekt zu starten.

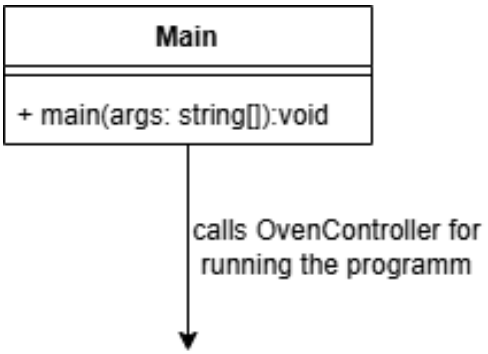

Abbildung 4: Klassendiagram der Mainklasse

OvenControllerModule

Das `OvenControllerModule` ist das zentrale Steuerungsmodul des Systems. Es steuert den Ablauf über ein State Pattern und enthält eine Referenz auf den aktuellen Zustand des Ofens.

In regelmäßigen Zyklen ruft der Controller die Methode `Run()` auf, in der zunächst die Eingaben abgefragt und anschließend an das `ModeControlModule` weitergegeben werden. Der Controller ist auch für die Aktualisierung des Displays verantwortlich, indem er das `OutputHandlerModule` mit neuen Daten versorgt.

Zudem werden in dieser Schleife der `Temperatursensor` aktualisiert, um das Verhalten eines physikalischen Sensors zu simulieren.

Die Zustände sind wie folgt umgesetzt:

- `IdleState`: Der Ofen ist inaktiv, alle Heizungen sind abgeschaltet.
- `PreHeatingState`: Der Ofen heizt auf die Zieltemperatur auf.
- `ActiveState`: Der Ofen hält die Zieltemperatur konstant.
- `ErrorState`: Eine Sicherheitsverletzung wurde festgestellt, alle Systeme werden abgeschaltet. Aus diesem Zustand kann nicht zurückgekehrt werden.
- `OffState`: Der Ofen wurde manuell oder nach Ablauf des Timers deaktiviert.

Der Controller kann durch das `SafetyModule` asynchron in den `ErrorState` versetzt werden, um auf kritische Ereignisse sofort reagieren zu können.

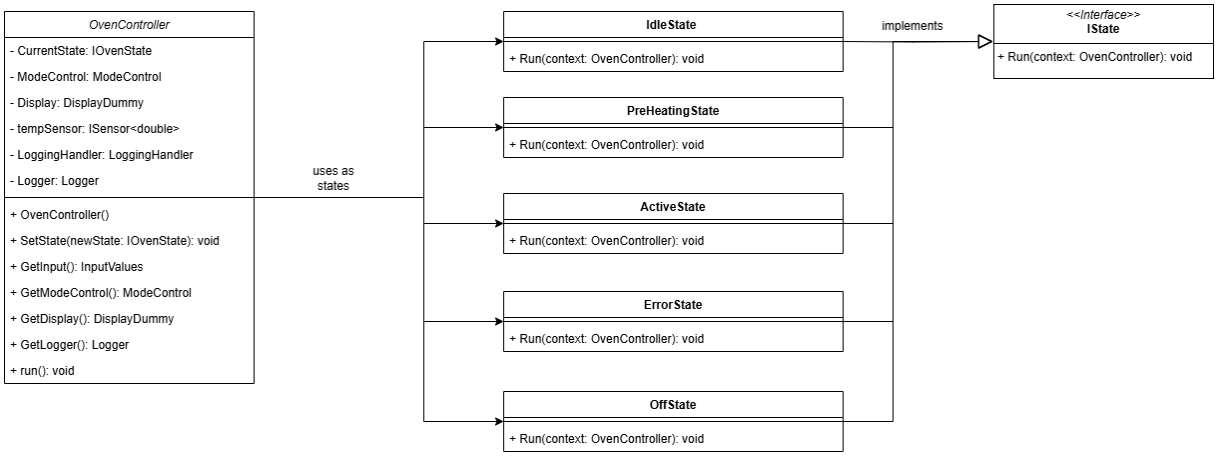

Abbildung 5: Klassendiagram des OvenController Moduls

SafetyModule

Das `SafetyModule` ist strikt vom restlichen System entkoppelt und läuft in einem eigenen Thread. Es überwacht kontinuierlich den Zustand von Sensoren, um sicherheitsrelevante Fehler frühzeitig zu erkennen.

Die Klasse `SafetyHandler` verwaltet eine Liste von Sicherheitsregeln und ruft deren `Check()`-Methoden in regelmäßigen Intervallen auf. Wird eine Regel verletzt, so wird der `OvenController` sofort benachrichtigt und in den `ErrorState` oder `OffState` versetzt.

Folgende Regeln sind implementiert:

- `OverheatRule`: Schlägt Alarm, wenn die Temperatur einen kritischen Maximalwert überschreitet. Führt zum Wechsel in den `ErrorState`.
- `DoorOpenRule`: Erkennt eine geöffnete Tür während des Heizbetriebs. Führt zum Wechsel in den `OffState`.
- `HeaterFailureRule`: Erkennt, ob bei aktiver Heizung die Temperatur über längere Zeit nicht steigt. Führt ebenfalls zum Wechsel in den `ErrorState`.

Durch die Ausführung in einem eigenen Thread ist das Modul vollständig von der Hauptlogik entkoppelt und gewährleistet somit eine hohe Reaktionssicherheit. Selbst wenn es im restlichen System zu Verzögerungen oder Fehlern kommt, laufen die sicherheitsrelevanten Prüfungen unabhängig und kontinuierlich weiter.

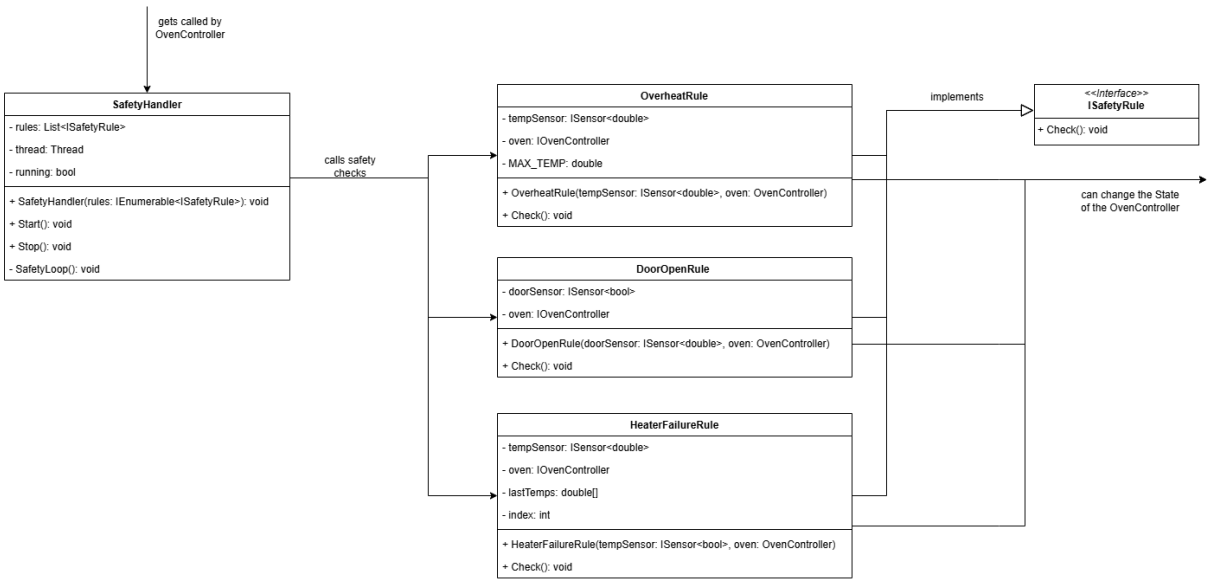

Abbildung 6: Klassendiagram des Safety Moduls

InputHandlerModule

Das `InputHandlerModule` dient der Erfassung physischer Benutzereingaben. Es verarbeitet keine automatischen Sensordaten, sondern ausschließlich manuelle Interaktionen.

Es liest die Drehwinkel zweier Regler – einen für die Temperatur und einen für den Modus – sowie den eingestellten Timerwert. Die Drehregler implementieren ein generisches Interface `IRotaryController<T>`, um unterschiedliche Rückgabetypen (z. B. `int` oder `Enum`) zu ermöglichen.

Die Klasse `InputHandler` aggregiert alle Eingaben in einem `InputValues`-Objekt, das so eine kompakte und erweiterbare Übergabestruktur bereitstellt. Der Zugriff erfolgt über einen `InputHandlerProxy`, der gleichzeitig für das Logging der Eingaben verantwortlich ist.

Das Proxy-Pattern wurde gewählt, um die Protokollierungslogik vom Produktivcode zu trennen.

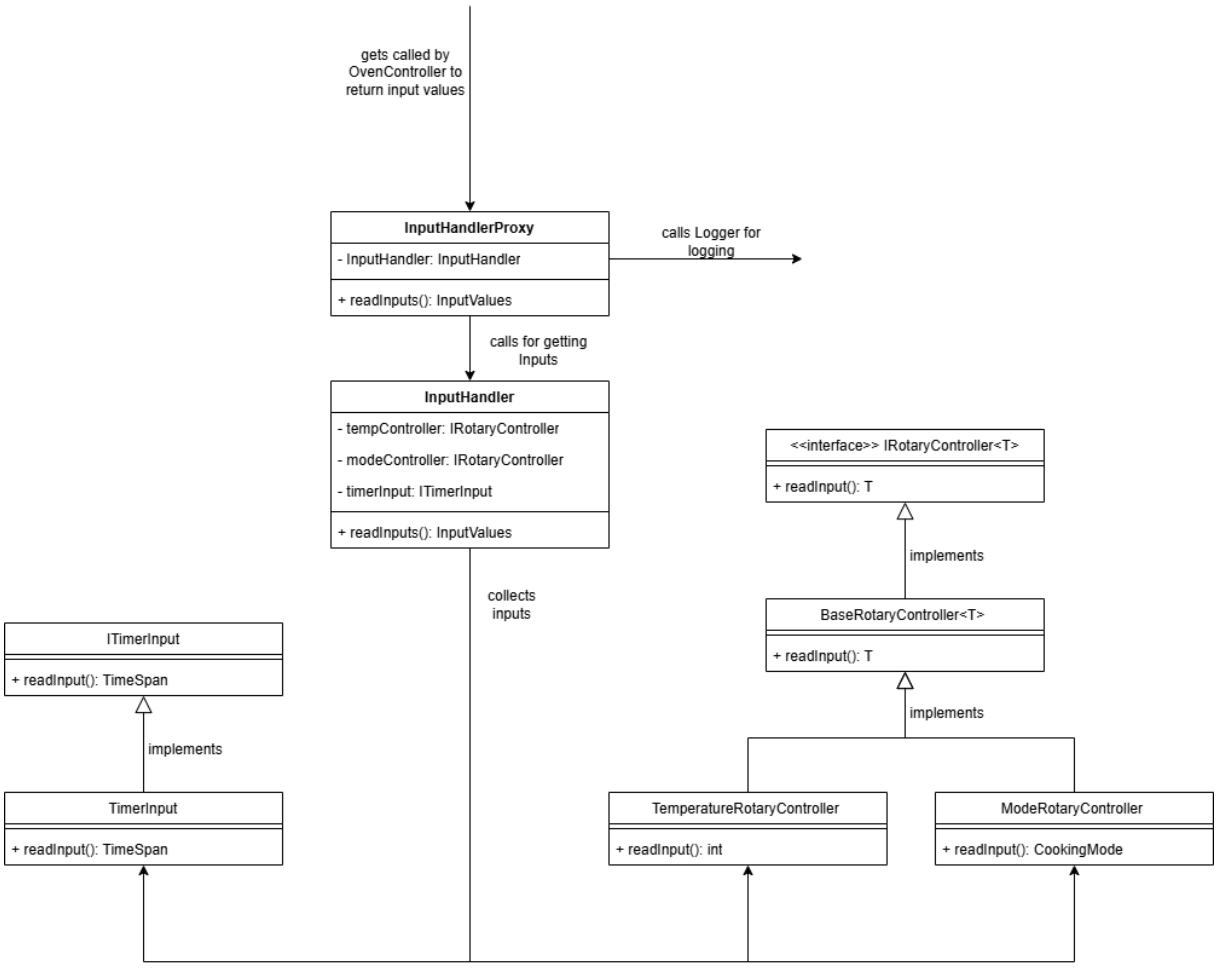

Abbildung 7: Klassendiagram des InputHandler Moduls

ModeControlModule

Das `ModeControlModule` realisiert das Strategy Pattern, um die Heizlogik modular und austauschbar zu gestalten. Die zentrale Klasse `ModeControl` hält eine Referenz auf die aktuell aktive `IModeStrategy` und kann diese zur Laufzeit austauschen, basierend auf dem übergebenen `CookingMode`.

Die Strategie wird anschließend vom `OvenController` verwendet, um die notwendigen Heizkomponenten zu aktivieren.

Folgende Strategien sind verfügbar:

- `IdleMode`: keine Heizkomponenten aktiv (Standardzustand)
- `TopBottomHeatMode`: obere und untere Heizkörper aktiv
- `TopHeatMode`: nur oberer Heizkörper aktiv
- `BottomHeatMode`: nur unterer Heizkörper aktiv
- `GrillMode`: oberer Heizkörper aktiv
- `ConvectionMode`: Ober- und Unterhitze plus Ventilator
- `HotAirMode`: hinterer Heizkörper und Ventilator aktiv

Wie beim Input-Modul erfolgt der Zugriff über einen `ModeControlProxy`, um Logging-Funktionalität zu kapseln.

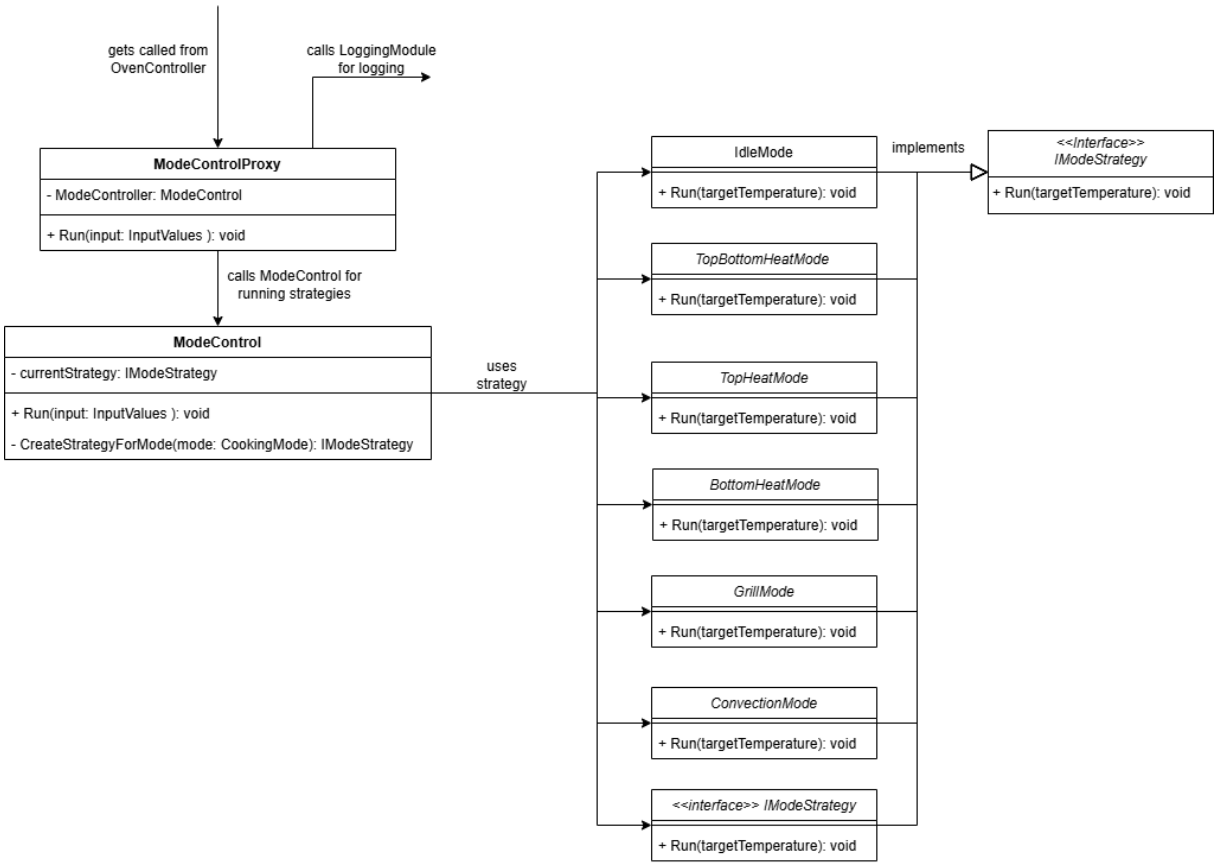

Abbildung 8: Klassendiagram des ModeControl Moduls

ThermalControlModule

Das `ThermalControlModule` bildet die unterste Schicht zur Ansteuerung der Heizkörper und Lüfter. Jede Komponente ist als Singleton implementiert, um zu garantieren, dass es von jedem Aktor nur genau eine Instanz im System gibt.

Die Klassen implementieren die Interfaces `IThermalController` (für Ansteuerung) sowie `ITemperatureSource` (für Temperaturmessung). Die Heiz- und Lüftereinheiten werden ausschließlich von Strategien im `ModeControlModule` aktiviert.

Die verfügbaren Komponenten sind:

- `TopHeater`
- `BottomHeater`
- `RearHeater`
- `RearFan`

Da die Ansteuerlogik vollständig ausgelagert ist, bleibt das Modul selbst zustandslos und wartungsarm.

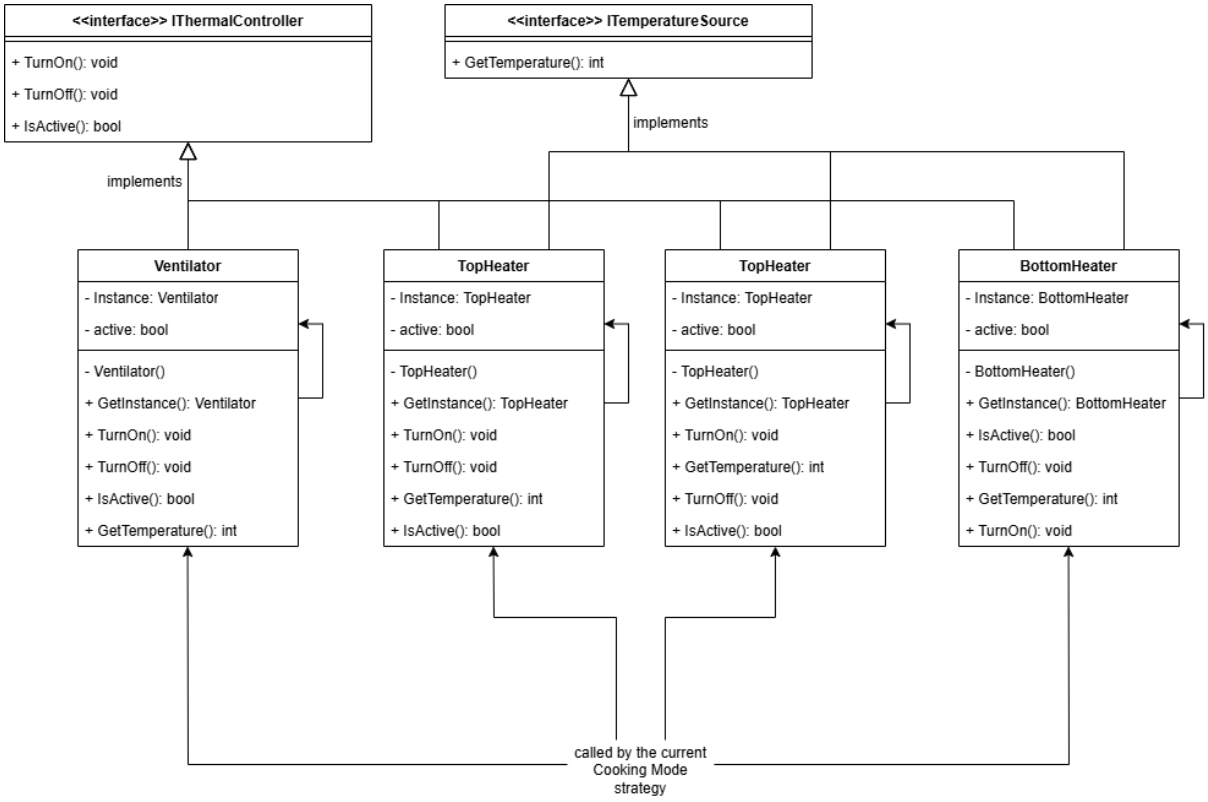

Abbildung 9: Klassendiagram des ThermalControl Moduls

OutputHandlerModule

Das `OutputHandlerModule` simuliert die grafische Anzeige des Backofens. Die Klasse `DisplayDummy` erhält regelmäßig ein `OutputValues`-Objekt mit allen anzuzeigenden Informationen.

Dazu gehören die aktuelle Temperatur, der Vorheizstatus, die verbleibende Garzeit sowie sicherheitsrelevante Hinweise. Das Modul stellt so die Schnittstelle zu einem realen Display dar und ermöglicht eine klare Trennung zwischen Steuerlogik und Präsentation.

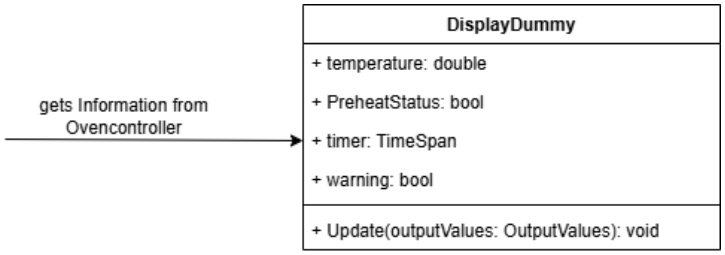

Abbildung 10: Klassendiagram des OutputHandler Moduls

SensorModule

Im `SensorModule` sind alle Sensoren abstrahiert und zentral definiert. Jeder Sensor implementiert das generische Interface `ISensor<T>` zur typisierten Abfrage.

Der `TemperatureSensor` berechnet die durchschnittliche Temperatur aus den Heizaggregaten, während der `DoorSensor` meldet, ob die Tür geöffnet ist. Beide Sensoren können von mehreren Modulen (z. B. Safety und OvenController) gleichzeitig verwendet werden.

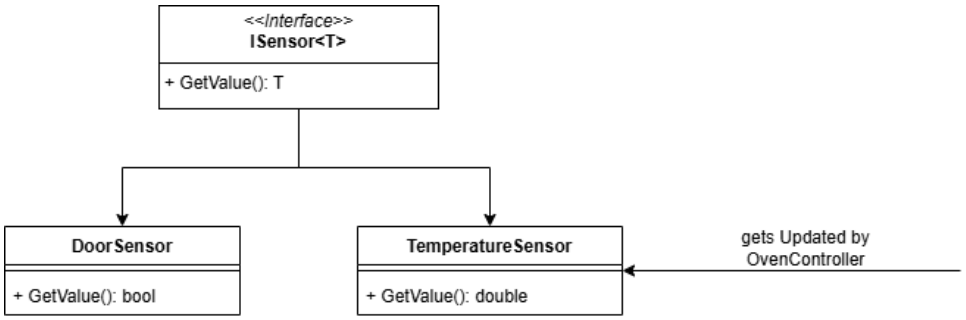

Abbildung 11: Klassendiagram des Sensor Moduls

LoggingModule

Das `LoggingModule` verwaltet die zentrale Protokollierung aller Systemereignisse. Über das Singleton `LoggingHandler` lassen sich für jedes Subsystem individuelle Logger abrufen.

Diese protokollieren Ereignisse mit Zeitstempel, etwa Eingaben, Zustandswechsel oder Warnungen. Durch die zentrale Verwaltung ist eine konsistente und nachvollziehbare Dokumentation aller Vorgänge gewährleistet.

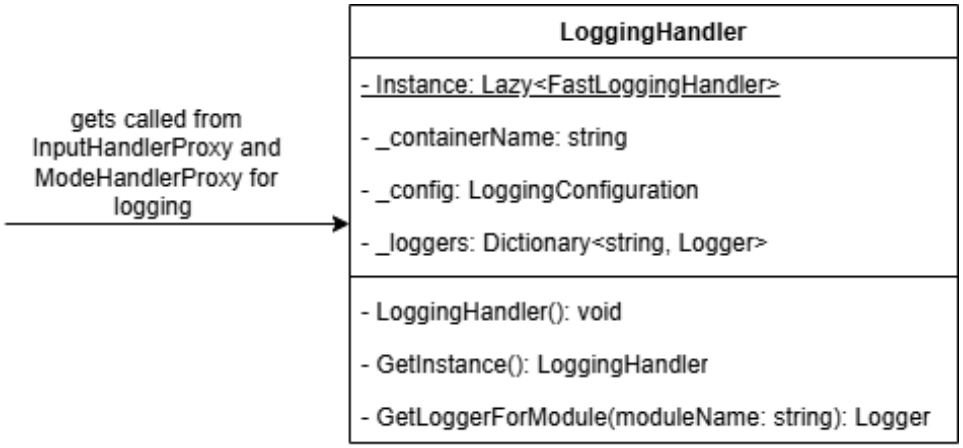

Abbildung 12: Klassendiagram des LoggingHandlers

5 Geplante Implementierungsiterationen

Die Implementierung des Backofensystems soll in mehreren aufeinander aufbauenden Iterationen erfolgen. Jede Iteration verfolgt dabei konkrete Entwicklungsziele, sodass schrittweise eine vollständige und modular aufgebaute Steuerungssoftware entsteht. Im Folgenden ist der geplante Ablauf detailliert beschrieben.

Iteration 1: Grundstruktur und Kernfunktionalität

In der ersten Iteration wird die grundlegende Systemarchitektur geschaffen. Das `Main`-Modul wird vollständig implementiert, um den Einstiegspunkt und die Initialisierung des Gesamtsystems zu realisieren. Ebenso werden im `GlobalModels`-Modul die zentralen Datenklassen `InputValues` und `OutputValues` erstellt.

Das `OvenControllerModule` wird vorerst nur mit dem `ActiveState` ausgestattet, um eine einfache Temperatursteuerung zu ermöglichen. Die Module zur Eingabe- und Ausgabeerfassung (`InputHandlerModule` und `OutputHandlerModule`) werden vollständig umgesetzt, sodass die Benutzerschnittstelle initial funktionsfähig ist.

Im `ModeControlModule` wird eine erste Standardstrategie realisiert, die alle verfügbaren Heizaggregate gleichzeitig nutzt. Das `ThermalControlModule` mit sämtlichen Aktoren (Heizkörper und Ventilator) sowie das `SensorModule` zur Temperatur- und Türerfassung sollen bereits vollständig implementiert werden.

Iteration 2: Logging und Strukturierung

In der zweiten Iteration wird das `LoggingModule` hinzugefügt, um sicherheitsrelevante und benutzerbezogene Ereignisse zentral zu protokollieren. Zusätzlich sollen Proxy-Klassen für das `InputHandlerModule` und das `ModeControlModule` eingeführt werden, um eine saubere Trennung zwischen Produktivcode und Logging umzusetzen. So wird eine erweiterbare, wartbare Architektur geschaffen.

Iteration 3: Erweiterung der Zustände und Heizmodi

In der dritten Iteration wird das `OvenControllerModule` um alle noch fehlenden Zustände ergänzt. Dazu zählen `IdleState`, `PreHeatingState`, `ErrorState` und `OffState`, welche gemäß dem `State Pattern` implementiert werden. Parallel dazu werden im `ModeControlModule` sämtliche Heizstrategien umgesetzt.

Iteration 4: Sicherheitsmodul

Die finale Iteration wird sich der Umsetzung des `SafetyModule` widmen. Dieses soll als eigenständiger Thread agieren und kontinuierlich alle sicherheitsrelevanten Bedingungen überwachen. Die Implementierung umfasst die drei Sicherheitsregeln `OverheatRule`, `DoorOpenRule` und `HeaterFailureRule`. Damit wird ein zuverlässiger Schutz des Gesamtsystems sichergestellt.

6 Iteration 1 – Grundfunktionen

6.1 Zeitraum

29.05.2025 – 30.05.2025

6.2 Ziele der Iteration

Ziel dieser Iteration war die Implementierung einer grundlegenden Temperaturregelung. Die Temperatur sollte über einen simulierten Drehregler einstellbar sein. Entsprechend dem gewählten Wert sollten alle relevanten Heizaggregate sowie der Ventilator automatisch aktiviert werden. Das System sollte die Zieltemperatur erreichen und anschließend stabil halten. Komplexe Zustandsverwaltung oder alternative Steuerstrategien wurden in dieser Phase noch nicht berücksichtigt.

Folgende Anforderungen wurden in dieser Iteration umgesetzt:

- **R-6.2**: Temperatureinstellung über Drehregler
- **R-6.2**: Temperatureinstellung in 1 °C-Schritten
- **R-6.2**: Temperaturbereich von 50 °C bis 300 °C
- **R-6.2**: Einzelansteuerung der Heizaggregate
- **R-6.2**: Temperaturregelung mit 1 Hz Abtastrate
- **R-6.2**: Abschalten der Heizaggregate bei erreichter Zieltemperatur
- **R-6.2**: Aktivierung der Heizaggregate unterhalb der Zieltemperatur
- **R-6.2**: Anzeige von Temperatur, Vorheizstatus, Timer und Warnungen auf dem Display
- **R-6.2**: Erreichen von 200 °C innerhalb von 10 Minuten

6.3 Requirements und Tests

Requirement	Beschreibung	Klasse/Methode	Testfäl
R-6.2	Temperatureinstellung über einen Drehregler	TemperatureRotaryController.ReadInput()	TC-3-1
R-6.2	Temperatureinstellungen in 1 °C-Schritten	TemperatureRotaryController.ReadInput()	TC-3-1
R-6.2	Temperaturbereich von 50 °C bis 300 °C	TemperatureRotaryController.ReadInput()	TC-3-1
R-6.2	Einzelansteuerung der Heizaggregate	IModeStrategy.Run(int targetTemperature)	TC-5-1
R-6.2	Temperaturregelung mit 1 Hz Abtastrate	OvenController.Run()	TC-0-1
R-6.2	Abschalten der Heizaggregate bei Zieltemperatur	IModeStrategy.Run(int targetTemperature)	TC-4-2,
R-6.2	Aktivierung der Heizaggregate unterhalb der Zieltemperatur	IModeStrategy.Run(int targetTemperature)	TC-4-2,
R-6.2	Anzeige von Temperatur, Status, Timer und Warnungen	DisplayDummy.Update(OutputValues)	TC-6-1
R-6.2	Erreichen von 200 °C in 10 Minuten	OvenController.Run()	TC-0-2

6.4 Abweichungen vom Feinentwurf

- `BaseRotaryController`: Methode `GetModuloAngle()` hinzugefügt, um das Durchdrehen zu ermöglichen
- `BaseRotaryController`: neue Eigenschaft `Angle`, um Vererbung zu erleichtern
- `TemperatureRotaryController`: neue Parameter `min` und `max` zur Begrenzung der Temperatur
- `Main` wurde in `Program` umbenannt (C#-Konvention)
- `ModeControl` wurde in `ModeController` umbenannt
- `Modes` enthalten jetzt eine Liste von `ThermalController` zur gezielten Steuerung
- `DisplayDummy`: neue Methode zur Anzeige von Werten auf der Konsole
- `OvenController`: neue Referenz auf `InputHandler`
- `OvenController`: Methode `GetTemperature()` zur Abfrage des Sensors
- `OvenController`: neue Methode `Loop()` für zyklisches `Run()`
- `DoorSensor`: neue Variable für Türstatus und Methode `SetDoorState()`
- `TemperatureSensor`: neue Variable für Temperaturwert und Methode `UpdateTemperature()`
- `FanController`: Methode `GetTemperature()` entfernt

Quellen

Die folgenden Onlinequellen wurden für die Erstellung der funktionalen und technischen Anforderungen herangezogen. Alle Links wurden zuletzt am **28.05.2025** auf ihren Inhalt überprüft (Zugriffsdatum).

- https://www.hea.de/fachwissen/herde-backoefen/elektroherde-aufbau-und-funktion
- https://www.schoener-wohnen.de/service/umluft-oder-heissluft-beim-backofen-13488 html
- https://de.wikipedia.org/wiki/Backofen#Haushaltsback%C3%B6fen

Nutzung von Künstlicher Intelligenz als Unterstützung

Requirements Engineering

Für die Umsetzung dieses Projektes wurde ChatGPT4o als Unterstützung verwendet. Die folgenden Punkte zeigen, wo in welcher Form KI zum Einzatz kam.

- Beim Requirements Engineering wurde es zur Formulierung und Validierung eigener Ideen verwendet.
- Beim der Software-Architektur wurde es zur Formulierung, Benennung von Modulen und Validierung eigener Ideen verwendet.
- Beim dem Software Design wurde es zur Formulierung, Benennung von Modulen und Validierung eigener Ideen verwendet.