Ereignisdiskrete Systeme

Praktikum Blatt 4 - Stateflow

Jan Kristel, Alexandra Moritz

Aufsicht von Frau Rembold

15. Juni 2023

Inhaltsverzeichnis

1 Was ist Stateflow ?

a) Was ist Stateflow?

Stateflow ist eine grafische Erweiterung von Simulink zur Modellierung und Simulation von

- Ereignisgesteuerten reaktiven Systemen mit endlich vielen Zuständen
- Zustandsautomaten (engl.: Finite State Machines)

Zustandsmaschinen sind ein Konzept aus der Informatik und der Systemtheorie, das verwendet wird, um das Verhalten eines Systems oder einer Komponente zu modellieren, das sich in verschiedenen Zuständen befinden kann. Es lassen sich Zustandsdiagramme erstellen, bestehend aus Zuständen, Übergängen zwischen diesen und Aktionen, die bei den Übergängen ausgeführt werden. Diese Diagramme beschreiben das Verhalten eines Systems.

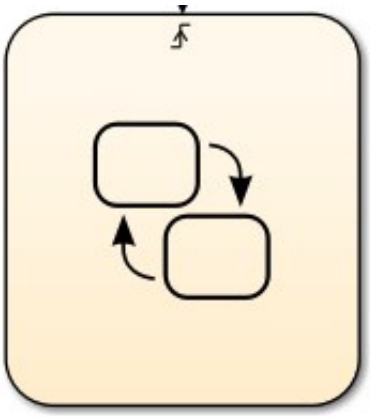
- Zustände repräsentieren, die verschiedenen Phasen oder Modi, in denen sich ein System befinden kann.
- Übergänge zeigen die Bedingungen an, unter denen das System von einem Zustand in einen anderen wechselt.
- Aktionen definieren die Operationen oder Reaktionen, die beim Zustandsübergang ausgeführt werden sollen.

In Kombination mit MATLAB und Simulink wird Stateflow verwendet, um das Verhalten von komplexen Systemen zu modellieren und zu simulieren. Dabei wird die Integration von Zustandsmaschinen in die allgemeine Systemmodellierung und -simulation ermöglicht und eine visuelle Darstellung des Systemverhaltens geboten. Entwickler bekommen eine effiziente Methode zur Modellierung und Implementierung von komplexen Systemverhalten.

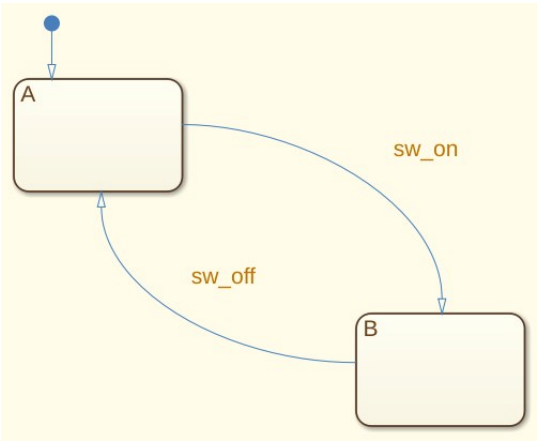
b) Was ist ein/e ...

b).1 Chart?

Zustandsautomaten werden als Zustandsübergangsdiagramme, s.g. **Charts** dargestellt.



(a) Darstellung eines Charts von "außen" aus dem Simulinkeditor



(b) Darstellung eines Chart von "innen".

Es besteht aus verschiedenen Zuständen, Übergängen, Aktionen und Ereignisse. Dabei kann ein Chart auch hierarchisch organisiert sein, wobei ein *übergeordneter Chart mehrere Untercharts enthalten kann.* Diese Hierarchie ermöglicht es, komplexe Systeme in kleinere

und besser handhabbare Teile zu zerlegen. In einem Chart können auch Aktionen definiert werden, die beim Eintritt oder Verlassen eines Zustands auszuführen sind. Beispielsweise Berechnungen, Signalmanipulationen oder das Auslösen von Ereignissen. Durch die Verwendung von Charts ist es möglich komplexe Systeme auf intuitive Weise zu modellieren und zu verstehen. Charts bieten eine visuelle Darstellung des Systemverhaltens und ermöglichen es Entwicklern, den Zustandsfluss und die Interaktionen zwischen den Zuständen effizient zu entwerfen und zu analysieren.

b).2 State?

Ein **State** repräsentiert einen bestimmten Zustand in dem sich ein System befindet. Dieser ist definiert durch eine eindeutige Kombination von Eigenschaften, die das Verhalten und die Merkmale des Systems zu einem bestimmten/dem aktuellekn Zeitpunkt beschreiben. Der State kann als eine Art Container betrachtet werden, der Aktionen, Übergänge und interne Variablen enthält (ähnlich dem Chart). Jeder State kann eine oder mehrere der folgenden Eigenschaften haben:

- **Aktionsblöcke** enthalten den auszuführenden Code/Aktionen, wenn der Zustand aktiv ist. Diese können *Berechnungen, Signalmanipulationen, Funktionsaufrufe oder andere spezifische Operationen* umfassen.

- **Übergänge** beschreiben den Wechsel von einem Zustand in einen anderen, unter bestimmten Bedingungen. Diese Bedingungen können auf Ereignissen, bool'schen Bedingungen oder zeitlichen Auslösern basieren.

- **Eingangsereignisse** werden von States empfangen, die von externen Quellen oder anderen Zuständen generiert werden. Diese Ereignisse können den Zustandsübergang oder das Verhalten im Zustand beeinflussen.

- **aktive Dauer** gibt an, wie lange der Zustand aktiv bleibt, bevor ein Übergang zu einem anderen Zustand ausgelöst wird.

b).3 History Junction?

Ein **History Junction** ermöglicht es, den Zustand eines übergeordneten States zu speichern und beim erneuten Eintritt in diesen an der Stelle fortzufahren, an der er zuvor verlassen wurde. Es stellt eine Möglichkeit der Wiederherstellung dar, anstatt immer wieder von Anfang an zu beginnen.

b).4 Default Transition?

Ein **Default Transition** ist eine Art Übergang in einem Zustandsautomaten, der ausgelöst wird, wenn keine anderen Übergänge die spezifischen Bedingungen erfüllen. Die Default Transition wird genutzt, um einen *Standardpfad* zu definieren, der genommen wird, wenn keine anderen Übergänge aktiviert werden können. Sie ist *optional und muss nicht in jedem Zustand definiert werden.* Wenn jedoch eine Default Transition in einem Zustand vorhanden ist und keine anderen Übergänge aktiviert werden, erfolgt der Zustandsübergang entsprechend dieser. Bei der Modellierung mit Stateflow ist es wichtig, die Verwendung von Default Transitions sorgfältig zu planen und zu überprüfen, um sicherzustellen, dass das Verhalten des Systems den Anforderungen und Spezifikationen entspricht.

b).5 Connective Junction?

Ein **Connective Junction** ermöglicht die Verbindung von mehreren ausgehenden Übergängen aus einem Zustand zu einem einzelnen Zielzustand. Normalerweise können in einem Zustand mehrere ausgehende Übergänge zu verschiedenen Zielzuständen definiert werden. Bei Verwendung einer Connective Junction können diese Übergänge an einem einzigen Verbindungspunkt zusammengeführt werden, bevor sie den Zielzustand erreichen.

b).6 Truth Table

Eine **Wahrheitstabelle** ist eine tabellarische Darstelliung von logischen Zuständen. Dabei erfüllt diese folgende Eigenschaften:

- Vermeidung von unübersichtlichen UND/ODER-Verknüpfungen
- wird wie grafische Funktionen verwendet
- enthält
  - Funktionsprototypen
  - Condition-Table
  - Decision-Table

D.h. jede Spalte D_i entspricht einer IF-Bedingung, jede Zeile ist eine UND-Verknüpfungen aller Bedingungen in einer Spalte. Die Aktion der ersten wahren Spalte (von links nach rechts) wird ausgeführt (und dann das Truth Table sofort verlassen)

b).7 Function ?

Eine **Funktion** repräsentiert einen blockartigen Container, der eine Gruppe von Aktionen oder Berechnungen enthält. Dieser wird verwendet, um komplexe Logik zu organisieren und *wiederverwendbaren Code* zu erstellen. Die Funktion bzw. der Block kann eine beliebige Anzahl von Ein- und Ausgabeparametern haben. Die enthaltenen Aktionen können ausgeführt, Variablen deklariert, Berechnungen durchgeführt, Bedingungen überprüft, Zustände geändert und andere Funktionen aufgerufen werden.
Funktionen können in einem Zustandsdiagramm oder in anderen Funktionen verwendet werden.

b).8 Embedded MATLAB Function ?

Bei **Embedded functions** handelt es sich um Funktionen, die *innerhalb einer anderen Funktion* definiert sind. Im Gegensatz zu separaten, eigenständigen Funktionen sind eingebettete Funktionen eng mit der umgebenden Funktion verbunden und haben *Zugriff auf deren Variablen und Parameter*. Ihre Verwendung bietet mehrere Vorteile:

- Modulare Programmierung
- Vermeidung von Namenskonflikten:
  Eingebettete Funktionen haben einen begrenzten Scope und können Variablennamen verwenden, die in der umgebenden Funktion bereits vorhanden sind, ohne Konflikte zu verursachen. Dadurch können lokale Variablen effektiv verwendet werden, ohne den globalen Namensraum zu beeinträchtigen.
- Datensicherheit:
  Die Trennung des Gültigkeitsbereichs der eingebetteten Funktionen und dem restliche Namensraum bezüglich der Variablen (lokal bzw. global) bietet eine gewisse Datensicherheit und verhindert unerwünschte Seiteneffekte oder Änderungen an den Variablen von außen.

b).9 Box ?

Zustände und Funktionen werden in rechteckigen Boxen dargestellt. Eine **Zustandsbox** repräsentiert einen State im Zustandsdiagramm, während eine **Funktionsbox** eine Funktion innerhalb diesem darstellt. Diese Boxen enthalten Logik, Aktionen und Transitionen, die den Zustandsautomaten steuern. Boxen sind grafische Einheiten zur visuellen Organisation eines Charts. Sie dienen dabei:

- der Zusammenfassung von beliebigen grafischen Elementen um Charts übersichtlicher zu machen.
- haben keinen Einfluss auf Funktionen des Modells
- können Einfluss nehmen auf die Aufführungsreihenfolge:
  - von oben nach unten
  - von links nach rechts (ähnlich wie bei parallelen Superstates)

c) Wie wird der grafische Editor gestartet?

- Öffnen eines Simulink-Modell: Lade das gewünschte Simulink-Modell, in dem du Stateflow verwenden möchtest, oder erstelle ein neues Modell.
- Öffnen des Stateflow-Editors: Im Reiter Modelling, sieht man eine Palette mit verschiedenen Simulink-Blöcken. Hier wählt man **Insert Chart**. Oder über den Libray Browser den Abschnitt Stateflow finden und den Chart-Block in das Modell ziehen.
- Doppelklick auf den Stateflow-/Chart-Block
- Danach wird der grafische Editor für Stateflow geöffnet. Hier können Zustandsdiagramme erstellt, Zustandsübergänge definiert und die Logik des Stateflow-Modells gestaltet werden.

d) Was ist der Model Explorer?

Es handelt sich um ein interaktives Fenster, das es ermöglicht, auf eine strukturierte und übersichtliche Weise auf verschiedene Elemente eines Simulink-Modells zuzugreifen und diese zu verwalten. Er bietet eine alternative Sicht auf das Modell und erleichtert die Navigation, Inspektion und Bearbeitung der Modellkomponenten. Einige Funktionen sind:

- Hierarchische Darstellung:
  Das Simulink-Modell wird in einer hierarchischen Struktur angezeigt. Es ermöglicht das Ausklappen/Zusammenklappen von Komponenten, um eine bessere Übersicht und Navigation zu ermöglichen.
- Elementinspektion:
  Durch Auswahl eines bestimmten Modell- oder Subsystemelements können detaillierte Informationen zu diesem Element angezeigt werden. Es bietet einen schnellen Zugriff auf die Parameter, Eigenschaften und Einstellungen des ausgewählten Elements.
- Verknüpfungen und Referenzen:
  Der Model Explorer zeigt die Verknüpfungen und Referenzen zwischen verschiedenen Modellkomponenten an. Dadurch können Abhängigkeiten und Beziehungen zwischen den Elementen visualisiert und verwaltet werden.
- Such- und Filterfunktionen:
  Mit diesen könne bestimmte Modellkomponenten basierend auf Namen, Typen oder anderen Kriterien gefunden werden können. Dies erleichtert die Suche in großen und komplexen Modellen.
- Modifikationen und Aktualisierungen:
  Es ist wird das Hinzufügen, Umbenennen, Löschen und Aktualisieren von Modellkomponenten ermöglicht. Dadurch können Modelländerungen schnell und einfach vorgenommen werden, ohne den grafischen Editor zu öffnen.

e) Wie werden einem Chart oder State Daten bzw. Events zugewiesen?

- Öffne den Stateflow-Editor:
  Öffne den Stateflow-Editor, indem du doppelt auf den entsprechenden Chart-Block oder State-Block im Simulink-Modell klickst.
- Definiere Eingänge:
  Im Stateflow-Editor kannst du Eingänge für den Chart oder State definieren. Klicke mit der rechten Maustaste auf den Chart-Block oder State-Block und wähle "Properties" oder "Eigenschaften" aus, um die Eigenschaften des Blocks anzuzeigen.
- Weise Eingänge zu:
  In den Eigenschaften des Charts oder States können Eingänge definieret werden. Klicke auf den Reiter "Inputs" oder

"Eingänge" und füge die gewünschten Eingänge hinzu. Hier könne sie benannt und ihren Datentyp festgelegt werden.

- Verwende Eingänge in Transitionen oder Aktionen:
  Sobald die Eingänge definiert sind, können sie in Transitionen oder Aktionen des Charts oder States verwendet werden. In einer Transition kann eine Bedingung, dargestellt durch **eckige Klammern**, festgelegt werden, die den Wert des Eingangs prüft, um den Zustandsübergang auszulösen. In Aktionen, dargestellt durch **geschweifte Klammern**, kann der Wert des Eingangs lesen und damit Berechnungen durchführen oder Zustandslogik steuern.

- Aktualisiere das Simulink-Modell:
  Nachdem Eingänge zugewiesen und verwendet wurden, speichert man den Stateflow-Editor und aktualisiertdas Simulink-Modells. Dadurch werden die Änderungen übernommen und das Modell ist bereit, Daten bzw. Events über die definierten Eingänge zu empfangen und zu verarbeiten.

- Durch die Zuweisung von Eingängen können Charts und States in Stateflow mit Daten bzw. Events interagieren und entsprechend reagieren. Es ermöglicht die flexible Steuerung des Zustandsverhaltens basierend auf externen Ereignissen oder Datenänderungen.

f) Wie werden im Chart Kommentare eingefügt ?

- Wähle das Tool "Comment" aus:
  Im Stateflow-Editor findet man in der Symbolleiste verschiedene Tools und Symbole. Klicke auf das "Comment"-Symbol, das oft wie ein Sprechblasensymbol aussieht. Alternativ kannst du die Tastenkombination "Ctrl+Alt+C" verwenden.

- das Kommentar-Symbol in den Chart ziehen:
  Nachdem das "Comment"-Tool ausgewählt ist, folgt dem Zeiger ein Kommentarsymbol. Mit Mausklick kann jetzt ein Fenster gezogen werden. Die Größe des Fenster gibt die Größe des Kommentars vor.

- den Kommentar schreiben:
  Sobald das Kommentar-Symbol platziert ist, wird ein Textfeld geöffnet, in dem der Kommentar eingegeben werden kann.

g) Was bedeutet in Stateflow "Exclusiv" und "Parallel" ?

g).1 Exclusiv

Exklusive Verbindung: Im Kontext von Zustandsübergängen bedeutet "exklusiv", dass nur ein einzelner Übergang zu einem bestimmten Zeitpunkt aktiv sein kann. Wenn mehrere Übergänge gleichzeitig aktiviert werden könnten, wird der Übergang mit der höchsten Priorität oder den spezifischsten Bedingungen ausgeführt.

Exklusiver Zustand: Ein exklusiver Zustand bezeichnet einen Zustand, der einen bestimmten Satz von Aktivitäten oder Aktionen ausführt, solange er aktiv ist. Andere Zustände im gleichen Zustandsdiagramm bleiben in diesem Fall deaktiviert.

g).2 Parallel

Parallele Aktivität: Im Kontext von Zuständen ermöglicht der parallele Verbindungstyp die gleichzeitige Aktivierung mehrerer Zustände. Dadurch können mehrere Aktivitäten oder Aktionen parallel ablaufen. Parallele Aktivitäten können entweder explizit als parallele Zustände im Zustandsdiagramm modelliert werden oder durch die Verwendung von Junctions oder inneren Transitionen erreicht werden.

Paralleler Zustand: Ein paralleler Zustand bezieht sich auf einen Zustand, der parallel zu anderen Zuständen aktiv sein kann. Dadurch können mehrere parallele Aktivitäten innerhalb des Modells stattfinden. Parallele Zustände können in Stateflow-Modellen verwendet werden, um komplexes Verhalten zu modellieren, das gleichzeitig oder unabhängig voneinander ablaufen kann.

Es ist wichtig zu beachten, dass die genaue Bedeutung von "exklusiv" und "parallel" stark vom Kontext innerhalb des Stateflow-Modells abhängt. Die Verwendung dieser Konzepte hängt von den spezifischen Anforderungen und der Modelllogik ab, die du in deinem Stateflow-Modell implementieren möchtest.

h) Wie wird ein Chart auf "Exclusiv" eingestellt ?

In Stateflow kann ein Chart nicht explizit auf den "Exklusiv"-Modus eingestellt werden. Der Exklusivitätsaspekt in Stateflow bezieht sich auf die Logik der Zustandsübergänge und Zustände im Chart. Wenn du sicherstellen möchtest, dass nur ein Übergang zu einem bestimmten Zeitpunkt aktiv ist, kannst du dies durch die richtige Definition der Bedingungen und Prioritäten der Zustandsübergänge erreichen. Die Priorität eines Übergangs kann durch die Reihenfolge der Übergänge im Zustandstisch oder durch die Verwendung von Guard-Bedingungen gesteuert werden. Der Übergang mit der höchsten Priorität oder der spezifischsten Bedingung wird aktiviert, während die anderen Übergänge inaktiv bleiben. Es ist wichDg, die Zustände und Übergänge in deinem Chart sorgfältig zu modellieren und sicherzustellen, dass die exklusive Logik deinen Anforderungen entspricht. Du kannst die Bedingungen und Prioritäten der Zustandsübergänge entsprechend anpassen, um sicherzustellen, dass nur der gewünschte Übergang zu einem bestimmten Zeitpunkt aktiv ist. Es gibt keine spezielle Einstellung oder Schalter, um den Chart selbst auf Ëxklusivßu setzen. Die Exklusivität wird durch die Definition der Zustandsübergänge und deren Bedingungen innerhalb des Charts erreicht.

i) Was beschreibt das Label einer Transition

- Bedingtes Label:
  Eine Transition kann ein bedingtes Label haben, das eine logische Bedingung enthält. Diese Bedingung wird evaluiert, und wenn sie wahr ist, wird der Übergang aktiviert. Ein Beispiel für ein bedingtes Label könnte $condition > 0$ sein, wobei *condition* eine Variable oder ein Ausdruck ist, der eine numerische Bedingung repräsentiert.

- Event Label:
  Eine Transition kann auch ein Event Label haben, das das Auftreten eines bestimmten Ereignisses auslöst. Das Ereignis kann entweder von externen Quellen wie Sensoren oder anderen Komponenten des Systems ausgelöst werden oder innerhalb des Stateflow-Modells generiert werden. Ein Event Label kann beispielsweise *event_name* sein, wobei *event_name* der Name des Ereignisses ist.

- Timer Label:
  Eine Transition kann auch ein Timer Label haben, das eine Zeitverzögerung angibt, bevor der Übergang aktiviert wird. Das Timer Label kann eine Zeitdauer in Sekunden oder eine Variable enthalten, die die Zeitverzögerung repräsentiert. Ein Timer Label könnte beispielsweise $timer > 5$ sein, wobei *timer* eine Variable ist, die die Zeit in Sekunden zählt, und der Übergang nach einer Verzögerung von mehr als $5 Sekunden$ aktiviert wird.

j) Wie starten Sie den Debugger ?

- Aktiviere den Debugging-Modus:
  Im Stateflow-Editor gibt es eine Symbolleiste mit verschiedenen Schaltflächen.

Suche nach dem "Debugging ModeSymbol, das oft wie ein kleines grünes Insekt aussieht. Klicke auf diese Schaltfläche, um den Debugging-Modus zu aktivieren. Alternativ funktioniert auch die Tastenkombination "Ctrl+D"verwenden.

- Breakpoints setzen:
  Um den Ablauf des Modells an bestimmten Punkten anzuhalten und den Debugger zu aktivieren, setze Breakpoints an den gewünschten Stellen. Klicke dazu mit der rechten Maustaste auf den Zustand oder die Transition, an der du einen Breakpoint setzen möchtest, und wähle SSet Breakpointöder "Breakpoint setzenäus dem Kontextmenü aus. Du kannst auch eine bestimmte Aktion in einer Aktionssprache oder einem MATLAB-Code mit einem Breakpoint markieren.

- Starte die Simualtion:
  Starte die Simulation deines Simulink-Modells. Dazu kannst du den Simulationsstart-Button in der Simulink-Oberfläche verwenden.

- Überwache den Debugging-Ablauf:
  Sobald die Simulation läuft und ein Breakpoint erreicht wird, stoppt die Ausführung an diesem Punkt, und der Debugger tritt in den Vordergrund. Du kannst Variablenwerte überprüfen, den Ablauf Schritt für Schritt verfolgen, den Zustand des Modells überwachen und Debugging-Aktinen durchführen, wie z.B. das Setzen von Haltepunkten, das Überwachen von Variablen oder das Verändern des Modellzustands.

2 Erste Schritte in Stateflow

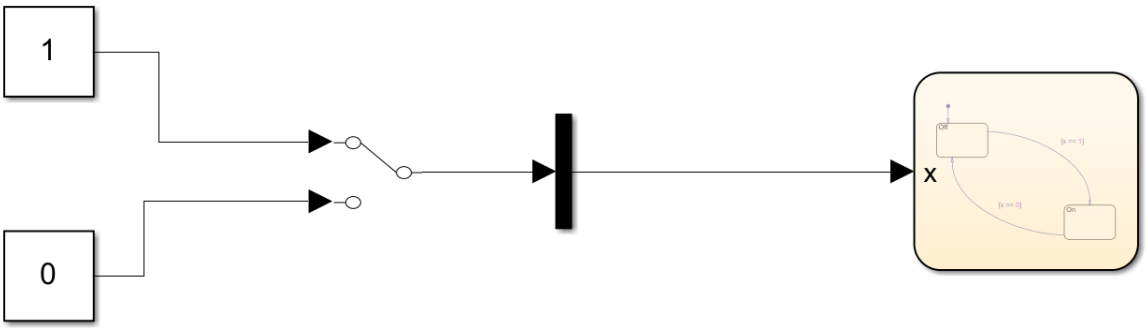


Abb. 2: Simulink Modell

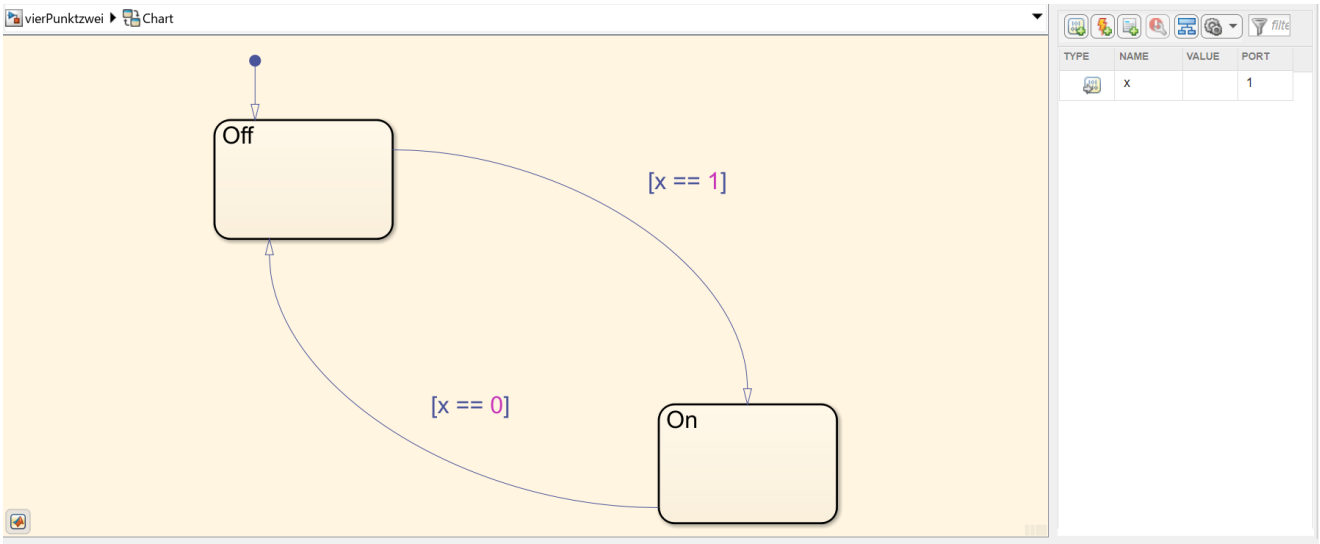


Abb. 3: Innenansicht des Stateflow Charts.

Rechts in Abb.4 ist die Tabelle mit den Variablen zu sehen. Da in diesem Chart nur eine, x, verwendet wird, sieht die Tabelle dementsprechend klein aus.

Type	Name	Value	Port
Inputdate	x	-	1

3 Weitere Schritte mit Stateflow

Für die beiden Teilaufgaben *a)* und *b)* wird das Simulink-Modell die Form aus Abb.5 beibehalten. Hier wird beschränken sich die Änderung auf das Stateflow-Chart selbst.

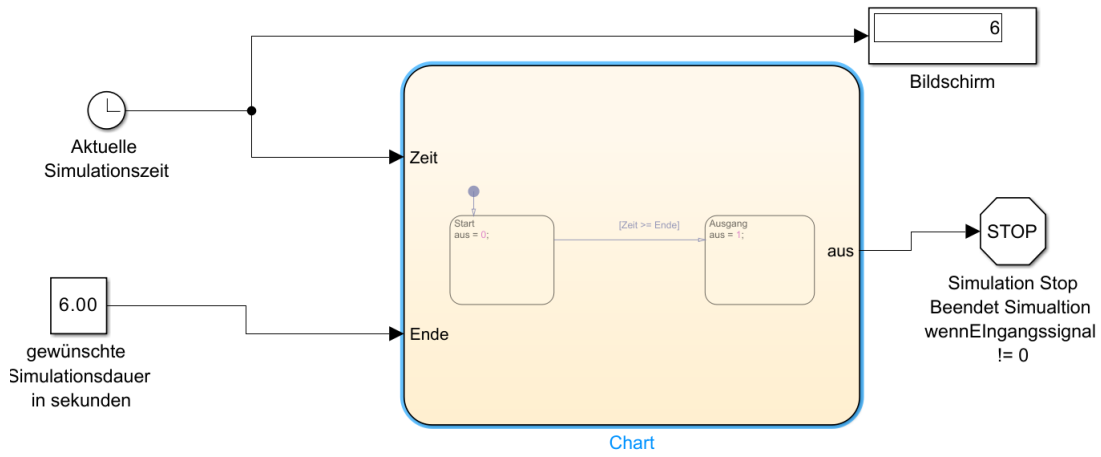


Abb. 4

a)

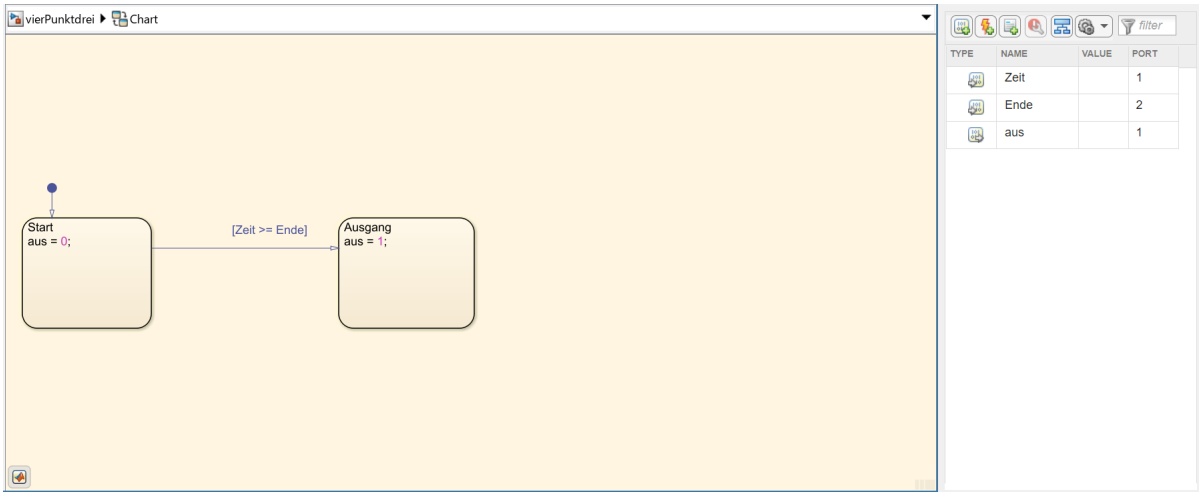


Abb. 5: Realisiert via States.

b)

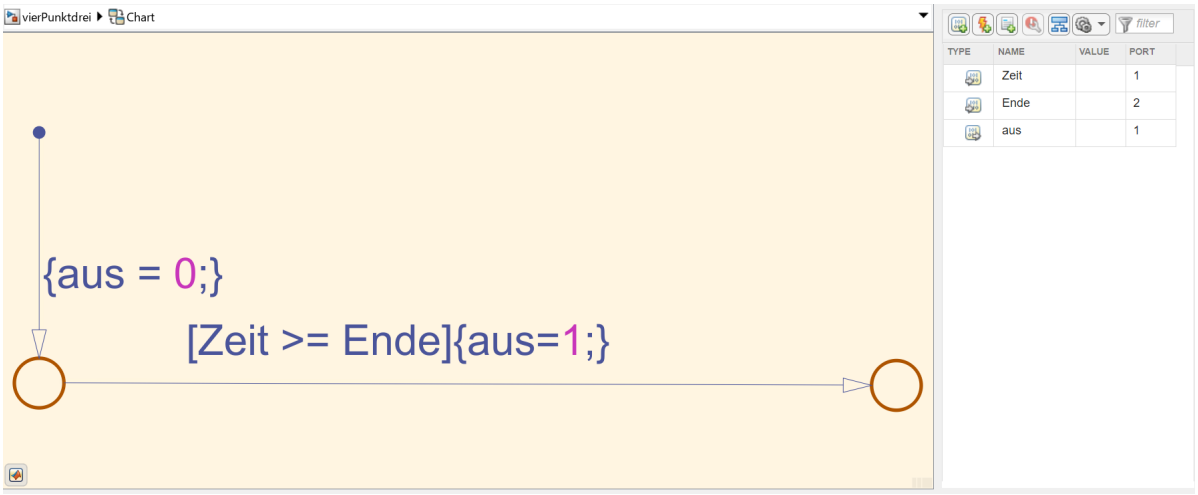


Abb. 6: Realisiert via Junctions.

c)

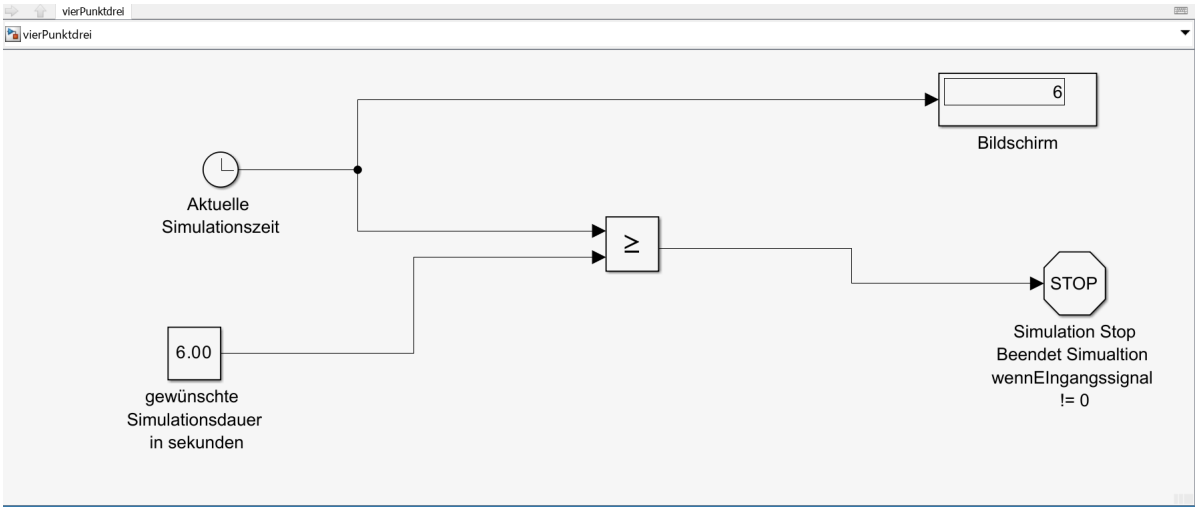


Abb. 7: Realisiert durch den Simulinkblock "Realtional Operator".

Durch die Realisierung mit dem Relational Operator der Simulink-Bibliothek fällt das Stateflow-Chart komplett raus.

d)

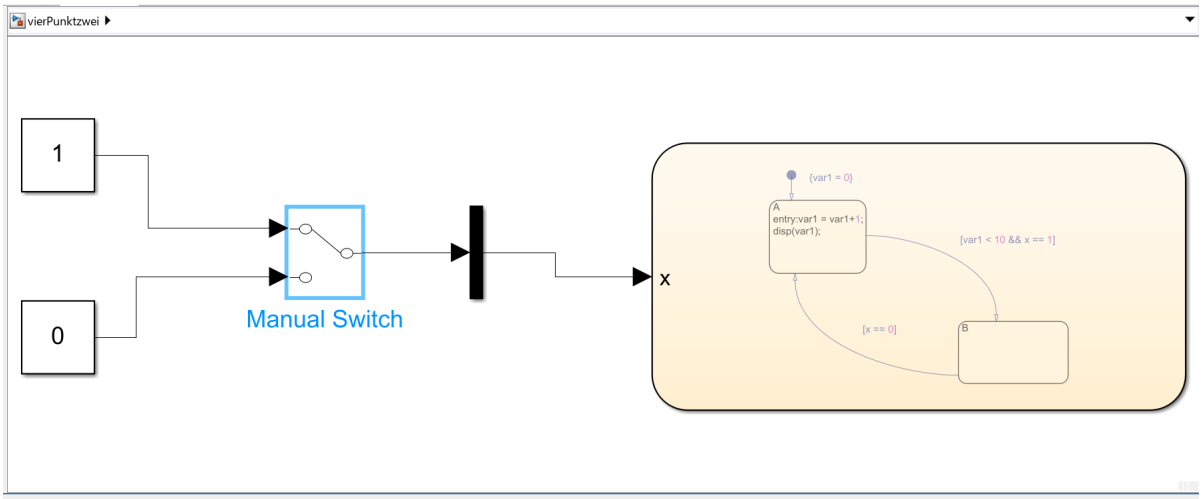


Abb. 8: Von "außen" ist schon zu sehen, dass innerhalb des Charts mehr Informationen stehen als ursprünglich in Aufg.2 (Abb.3).

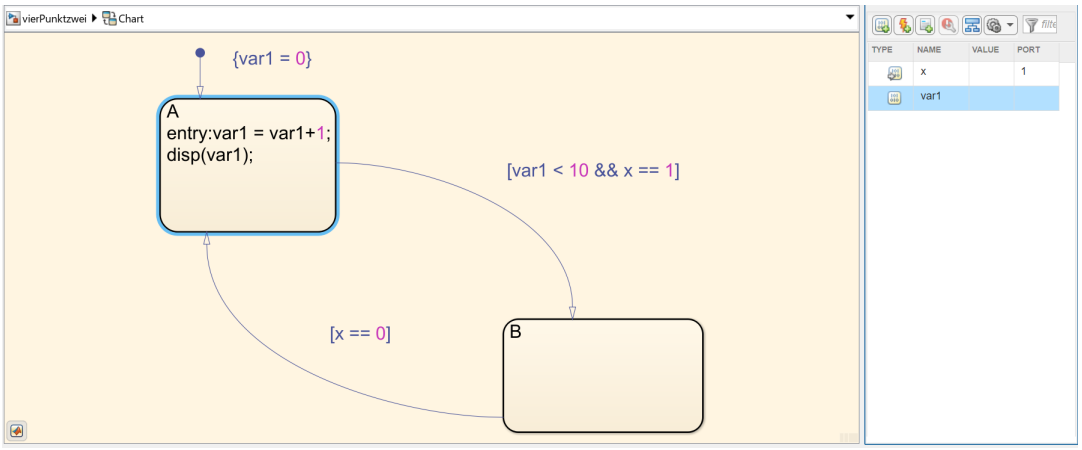


Abb. 9: Im State A ist nun weit mehr definiert als vorher.

Den Änderungen in *State A* entsprechend hat sich auch die Symboltabelle verändert.

Type	Name	Value	Port
InputData	x	-	1
LocalData	var1	-	-

e)

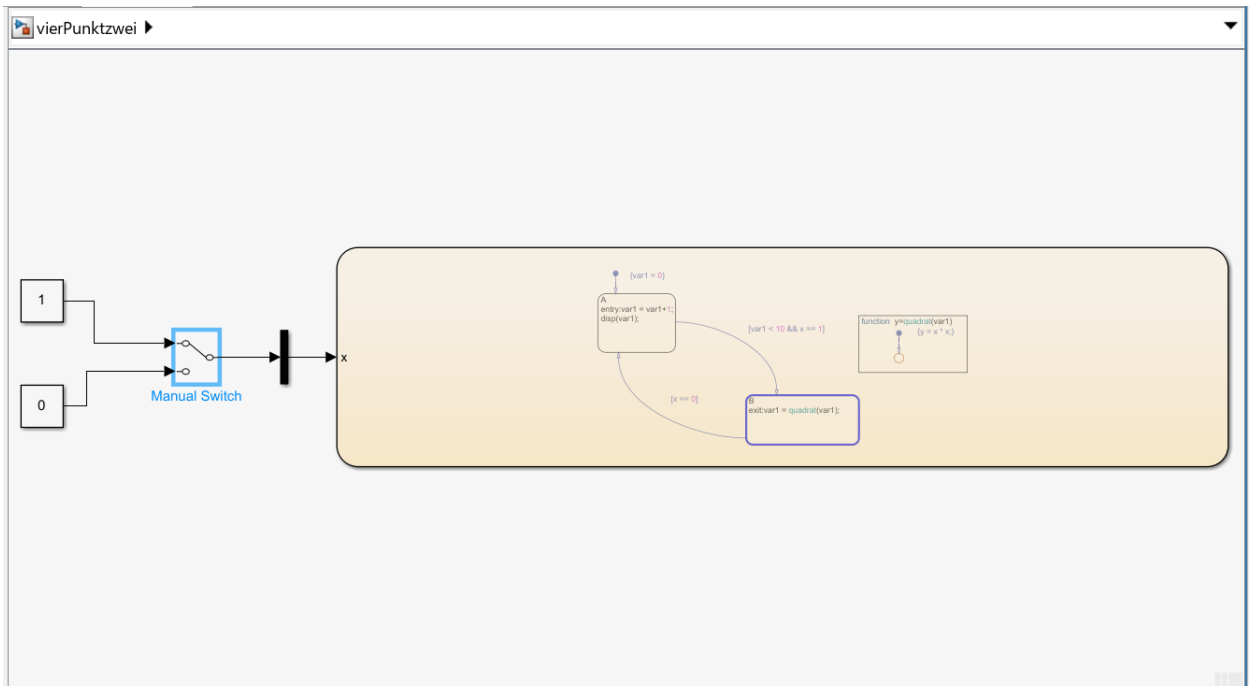


Abb. 10: Nächste Erweiterung.

Bei dem bereits zusehende dritten State, handelt es sich nicht direkt um ein State, sondern um die geforderte *grafische Funktion*. Dabei handelt es sich um einen separaten Stateblock, der keine Verbindungen zu den anderen States benötigt. Die Funktion wird vergleichsweise klassisch aufgerufen wie Funktionen in Java oder C/C++.
In *State B* wird ein Variable ‘var1‘ deklariert. Diese bekommt den Rückgabewert der aufgerufenen Funktion, hier ‘quadrat(var1)‘, zugewiesen.

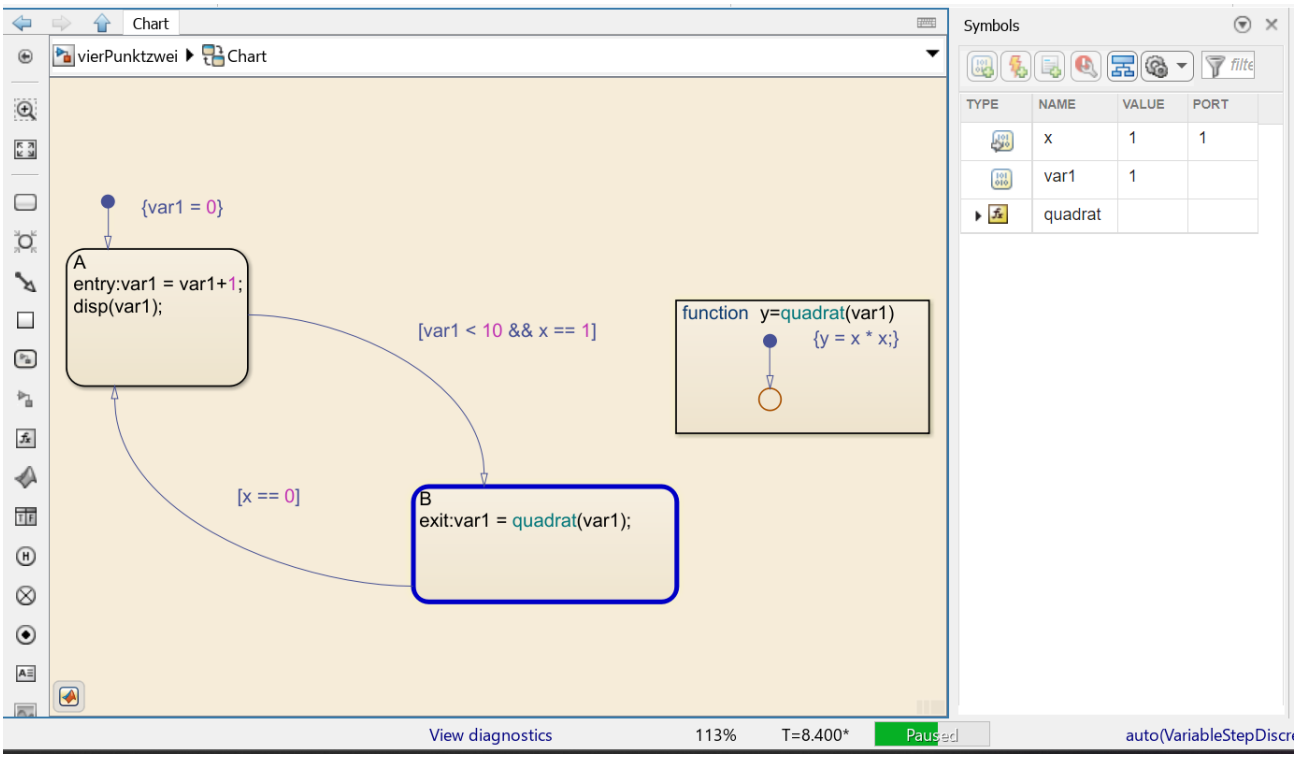


Abb. 11: Erweiterung durch eine grafische Funktion.

4 Anwendungsbeispiel - Mischer

a)

Aus der 1.Differntialgleichung

$$c \cdot m \cdot \frac{d\vartheta(t)}{dt} = P(t) - O \cdot k \cdot \vartheta(t)$$

lässt sich folgende Übertragungsfunktion erzeugen

$$G(s) = \frac{\vartheta(s)}{P(s)}$$

$$= \frac{1}{c \cdot m + O \cdot k}$$

$$= \frac{\frac{1}{O \cdot k}}{\frac{c \cdot m}{O \cdot k} s + 1}$$

Mit $K = \frac{1}{O \cdot k} = 1$ und $T_1 = c \cdot m \cdot k = 30sec$ sieht die Übertragsfunktions für den Mischer so aus:

$$G(s) = \frac{0.001}{30s + 1}$$

Es handelt sich damit um ein PT-1-Glied.

b)

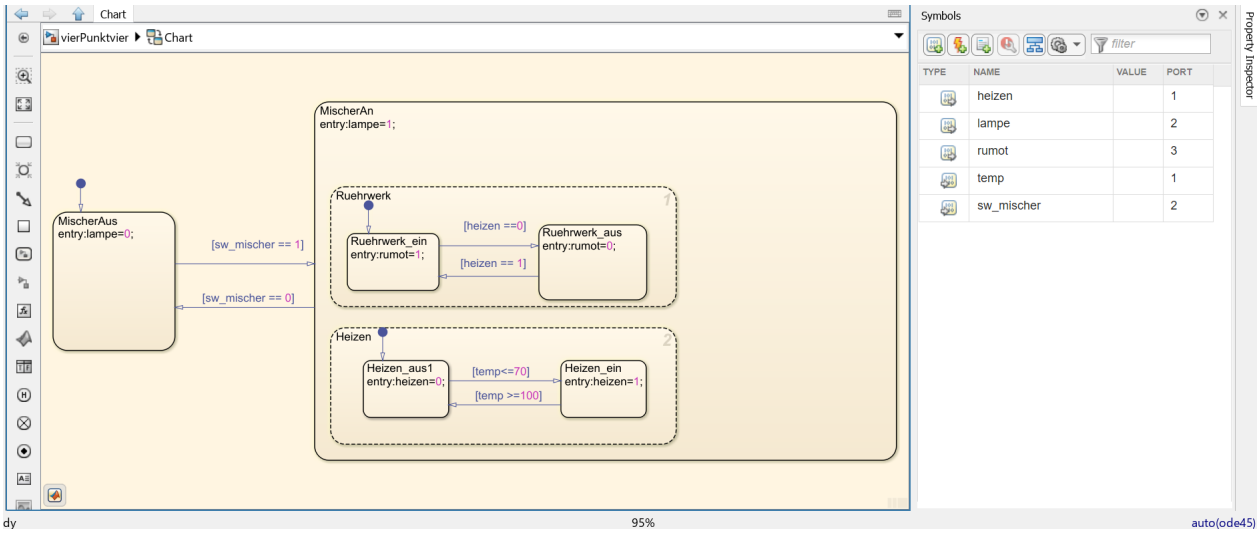


Abb. 12: Erstelltes Chart für die Mischersteuerung.

c)

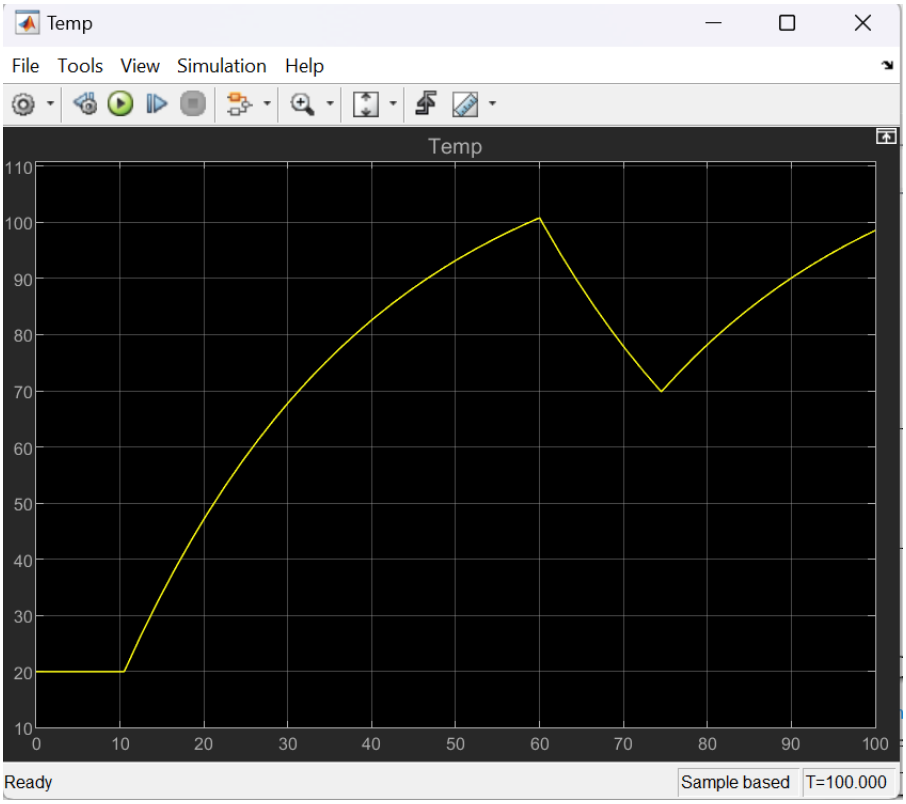


Abb. 13: Der Temperaturverlauf der Heizung. Zu erkennen sind der Heizvorgang von $0 - 70\,°C$ *und das Abkühlen von* $100 - 70\,°C$

d)

Aus dem Graph (Abb. 14) lässt sich die Zeit sehr gut ablesen. Die Dauer des

- Heizens: $49\ sec\ = 60 - 11$
- Abkühlen: $14\ sec\ = 74 - 60$