technische universität dortmund

FAKULTÄT MASCHINENBAU

IT in Produktion und Logistik

BACHELORARBEIT

Alexander Wuttke

Datenerfassung an einem Modell eines automatischen Hochregallagers zum Aufbau einer Simulationsdatenbasis

Studiengang	B.Sc. Maschinenbau
Matrikelnummer	169108
Thema ausgegeben am	21.11.2019
Arbeit eingereicht am	13.01.2020
Prüfer	Dr.-Ing. Dipl.-Inform. Anne Antonia Scheidler
Betreuer	M.Sc. Joachim Hunker

Inhaltsverzeichnis

1 Einleitung

Das Ziel von Unternehmen ist die Erzielung eines möglichst hohen Maßes an Rentabilität (Pfohl 2016). Um dieses Ziel zu erreichen, sehen sich die Unternehmen einer Vielzahl von Herausforderungen, wie einem sich verschärfenden Wettbewerb, kürzeren Arbeitszeiten, hohen Lohnnebenkosten und einer gestiegenen Anforderung an den Servicegrad, gegenüber (Bichler und Schröter 2004). Weitere Herausforderungen, die besonders für die Lagersysteme der Unternehmen eine Rolle spielen, sind die gestiegene Kundenindividualität, aus der eine erhöhte Produktvielfalt folgt, und die gestiegene Auftragslast (ten Hompel et al. 2018).

Um diesen Herausforderungen zu begegnen, werden flexible und automatische Lagersysteme, wie das automatische Hochregallager, eingesetzt (ten Hompel et al. 2018; Bichler und Schröter 2004). Da automatische Hochregalläger aber auch einen hohen Investitionsbedarf aufweisen (Vahrenkamp und Kotzab 2012) und im Falle des Versagens des Lagersystems nicht nur mit erheblichen Problemen im Lager selbst, sondern auch Störungen in den vor- und nachgeschalteten Bereichen wie der Vorfertigung, Montage oder dem Versand zu erwarten sind (Jünemann 1988), ist eine bestmögliche Planung des entstehenden automatischen Hochregallagers wichtig.

Als planungsunterstützende Methode hat sich die Simulation bewährt (Clausen 2008) und ist für komplexe Lagersysteme, wie dem automatischen Hochregallager, „nicht mehr wegzudenken" (Krieg 1985, S. 82). Anwendung findet Simulation dann, wenn analytische Methoden nicht benutzt werden können, weil das vorliegende Problem zu komplex ist, oder keine fachliche Realisierbarkeit gewährleistet ist (Law 2015). Die Simulation kann z.B. zur Untersuchung der Platzausnutzung, der benötigten Kapazität, oder der besten Lagerstrategie eingesetzt werden (Krieg 1985) und liefert abgesicherte, nachvollziehbare Planungsergebnisse (Gutenschwager et al. 2017). Die Richtigkeit und Übertragbarkeit der Simulationsergebnisse sind von erheblicher, teilweise von existenzieller, Bedeutung für das Unternehmen (Rabe et al. 2008; Bichler und Schröter 2004).

Die Methode der Simulation basiert auf Daten, die in einer Simulationsdatenbasis bereitgestellt werden (VDI 2014). Dabei gilt, die Ergebnisse einer Simulation können nur so gut sein, wie die Daten, die sie treiben (Bode und Hoya 1985). Um schon vor dem Bau eines Hochregallagers auf Daten zurückgreifen zu können, kann ein physisches Modell des zu planenden Hochregallagers gebaut und an diesem physischen Modell Experimente im Vorfeld durchgeführt werden (Arnold und Furmans 2019). Die sich in dem Experiment ergebenden Daten können mittels Sensoren erfasst und zum Aufbau einer Simulationsdatenbasis benutzt werden (VDI 2001). In der Literatur ist, nach Kenntnis des

Autors, keine allgemeine Simulationsdatenbasis für automatische Hochregalläger entwickelt worden. Insbesondere sind keine Methoden zur Datenerfassung an einem physischen Modell dargelegt und wie diese Daten zum Aufbau einer Simulationsdatenbasis genutzt werden können.

Das Ziel dieser Arbeit ist das Identifizieren von Daten, die zur Simulation eines automatischen Hochregallagers verwendet werden können und durch Datenerfassung an einem physischen Modell eines automatischen Hochregallagers gewonnen werden. Weiterhin ist eine Zielsetzung, anhand eines Experiments an einem physischen Modell, mittels Sensoren, exemplarisch Daten für eine Simulationsdatenbasis zu erheben und für die Weiterverarbeitung zur Verfügung zu stellen. Um diese Ziele zu erreichen, wird die Erfüllung mehrerer Teilziele vorausgesetzt.

Das erste Teilziel ist die Schaffung eines Verständnisses von automatischen Hochregallägern, um deren Charakteristika und Funktionsweisen nachvollziehen zu können. Im zweiten Teilziel geht es um die Einordnung, aus welchem Grund eine Simulation von automatischen Hochregallägern sinnvoll ist und wie eine Simulationsstudie durchgeführt werden kann. Das dritte Teilziel ist die Untersuchung des Aufbaus einer Simulationsdatenbasis und wie sie durch Datenerfassung an einem physischen Modell aufgebaut werden kann. Ein Schwerpunkt ist dabei die Identifizierung der Daten für die Simulationsdatenbasis anhand von für die Praxis relevanter Kennzahlen. Das letzte Teilziel ist die Entwicklung und Ausführung eines Experiments an einem physischen Modell eines automatischen Hochregallagers, währenddessen eine exemplarische Datenerfassung durchgeführt werden kann.

Zur Erarbeitung der Ziele wird methodisch wie folgt vorgegangen. Zuerst wird eine Einführung in Läger, Palettenläger und Hochregalläger gegeben, um die Charakteristika von Hochregallägern und deren Einordnung nachvollziehen zu können. Danach wird eine Betrachtung von automatischen Hochregallägern und deren Merkmalen durchgeführt. Als nächstes wird in die Methode der Simulation eingeführt und ihre Anwendungen und Nutzen im Bereich der Produktion und Logistik erläutert, damit die Notwendigkeit einer Simulationsdatenbasis klar wird. Um zu definieren, welche Daten für eine Simulationsdatenbasis erhoben werden müssen, wird im nächsten Schritt zunächst untersucht, welche Ergebnisse in einer Simulationsstudie erarbeitet werden. Nun wird untersucht, wie von den geforderten Ergebnissen einer Simulationsstudie, auf die benötige Simulationsdatenbasis geschlossen werden kann. Um im weiteren Verlauf der Arbeit exemplarisch eine Simulationsdatenbasis aufbauen zu können, wird in die Grundlagen von Mikrocontrollern, Peripheriegeräten und Sensoren eingeführt und einige gängige Sensoren, die für die Erfassung der Daten des automatischen Hochregallagers gebraucht werden, werden vorgestellt. Damit Daten für die Simulationsdatenbasis

identifiziert werden können, wird der Zusammenhang von Daten, Kennzahlen und Zielen erläutert, sowie nötiges Wissen zu diesen Themen vermittelt. Außerdem wird das physische Modell des automatischen Hochregallagers, welches für das zu entwickelnde Experiment benutzt werden soll, beschrieben. Auf diesen Grundlagen aufbauend wird im weiteren Verlauf der Arbeit ein möglicher Aufbau einer Simulationsdatenbasis für automatische Hochregalläger erarbeitet. Begonnen wird mit dem Aufbau eines Zielsystems, um die Ziele automatischer Hochregalläger nachvollziehen zu können. Aus den Zielen des Zielsystems werden daraufhin Kennzahlen abgeleitet, mit dem Ziel, möglichst viele, für die Praxis relevante Fragestellungen beantworten zu können. Basierend auf den Kennzahlen werden Daten zu deren Berechnung abgeleitet, die die Simulationsdatenbasis für automatische Hochregalläger bilden. Nachdem die Daten zum Aufbau der Simulationsdatenbasis identifiziert wurden, wird zunächst eine passende Auswahl von Sensoren, unter Beachtung der Restriktionen, die sich durch das zur Verfügung stehende physische Modell des automatischen Hochregallagers ergeben, getroffen. Es wird auch eine Möglichkeit zur Speicherung der erfassten Daten auf einem Computer in einem geeigneten Datenformat erläutert. Im Folgenden wird die entwickelte Simulationsdatenbasis exemplarisch, durch Datenerfassung an dem physischen Modell, aufgebaut. Nachdem spezifiziert wurde, welche Daten erfasst werden sollen und welche Sensoren dazu benutzt werden sollen, wird ein Experiment geplant. Während dieses Experiments erfassen die verbauten Sensoren die für die Simulationsdatenbasis benötigten Daten. Dazu wird zunächst die Ausgangssituation bei der Verwendung des physischen Modells erklärt. Weitere Punkte zur Planung des Experiments sind das Treffen von Vereinfachungen, die benötigt werden, um Restriktionen des physischen Modells zu umgehen, die Entwicklung eines Ablaufs und die Möglichkeit der Speicherung von Daten, damit in späteren Arbeiten darauf zugegriffen werden kann. Bevor das Experiment durchgeführt werden kann, müssen Anpassungen an dem physischen Modell vorgenommen werden, wie z.B. die Integration zusätzlicher Sensoren. Außerdem wird die Steuerungssoftware, die für den Ablauf des Experiments und der Datenerfassung verantwortlich ist, vorgestellt. Sobald das physische Modell den gewünschten Funktionsumfang fehlerfrei leisten kann, wird das geplante Experiment durchgeführt und dokumentiert. Im Anschluss daran wird das durchgeführte Experiment einer Bewertung unterzogen. Die Arbeit endet mit einer Zusammenfassung und einem Ausblick.

2 Läger

Um Daten für eine Simulationsdatenbasis eines automatischen Hochregallagers zu identifizieren, muss die Funktionsweise des zu betrachtenden Gegenstands, also dem automatischen Hochregallager, bekannt sein. Zusätzlich zu der Funktionsweise müssen, um ein physisches Modell erstellen oder daran forschen zu können, Grundlagen über den technischen Aufbau automatischer Hochregalläger gegeben sein. In den folgenden Abschnitten werden die nötigen Grundlagen automatischer Hochregalläger dargelegt. Dazu werden zunächst Grundlagen von Lägern, Palettenlägern und nicht-automatisierte Hochregalläger diskutiert, um zu sehen, welche allgemeinen Konzepte auf automatische Hochregalläger übertragbar sind und wie es sich von anderen Lagertypen unterscheidet.

2.1 Grundlagen von Lägern

Laut dem Verein Deutscher Ingenieure (VDI) (1970) ist lagern das geplante Liegen eines Guts im Materialfluss. In Lägern werden Güter gelagert, die erst zu einem späteren Zeitpunkt benötigt werden (ten Hompel 2018). Da in diesem Zustand der Zeitüberbrückung keine Wertschöpfung stattfindet, würden die Just-in-time und Lean Management Bewegung die Lagerhaltung am liebsten abschaffen (Frazelle 2016). Dennoch sind Läger in der Praxis unumgänglich und sogar sinnvoll, da sie wichtige Aufgaben erledigen können, die ten Hompel et al. (2018, S.52- 53) wie folgt beschreiben:

- Asynchrone Zu- und Abgänge zwischen Bereichen oder Anlagen
- Mengenausgleich, z. B. im Rahmen der Fertigung wirtschaftlicher Losgrößen
- Sicherstellung der Auslastung kostenintensiver Produktionsanlagen, z. B. bei Störungen, Lieferengpässen, Verkehrsproblemen usw.
- Nutzung kostenoptimierter Bestellmengen durch die Gewährung von Rabatten
- Ausnutzung der Transportkapazitäten im Verkehr
- Saisonale Schwankungen im Absatzverhalten
- Schaffung eines hohen Lieferservices durch schnelle Befriedigung von Aufträgen oder Kundenwünschen bzw. Sicherstellung der Lieferfähigkeit insgesamt
- Lagerung zur Wertsteigerung (durch Reifung) oder zwecks spekulativer Absichten
- Erfüllung von Zusatzaufgaben, wie z. B. die Bereitstellung zur Kommissionierung

Die Aufgaben, die Läger erfüllen, können in vier Grundtypen unterteilt werden: Bevorratung, Pufferung, Verteilung und Sammlung (ten Hompel et al. 2018). Die primäre Aufgabe von Vorrats- und Pufferlägern ist die Überbrückung von Zeit (Schulte 2016). Sie unterscheiden sich dadurch, dass Vorratsläger eher wenige Umschläge, dafür aber längere Einlagerungsdauern aufweisen und Pufferläger eine hohe Umschlaghäufigkeit mit niedriger Einlagerungsdauer (ten Hompel et al. 2018). Verteilläger und Sammelläger

zeichnen sich durch das Ändern der Lagereinheiten zwischen Zu- und Abgang aus. Ändert sich die Zusammensetzung der Ladeeinheiten zwischen Ein- und Auslagerung gleich, wird das Lager auch Kommissionierlager genannt. Ist dies nicht der Fall, wird es Einheitenlager genannt (ten Hompel et al. 2018). Verteilläger haben die Aufgabe, aus den eingehenden Lagereinheiten einzelne Ladungsmengen zu kommissionieren, zu sortieren und zu konsolidieren. Daher werden sie besonders oft in Rohteil- oder Zukaufsteillägern eingesetzt, da hier die eintreffenden Lagereinheiten zumeist mehrere Teileinheiten beinhalten (ten Hompel et al. 2018). Sammelläger hingegen fügen mehrere Teilmengen einer Transportladung zu einer großen Gesamtlagereinheit zusammen (Vahrenkamp und Kotzab 2012).

Die sich aus der Lagerhaltung ergebenden Vorteile stehen den Lagerhaltungskosten gegenüber. Sie bestehen zum einen aus den Lagerungs- und zum anderen aus den Kapitalbindungskosten (Schulte 2016). Die Lagerungskosten werden durch Abschreibungen auf das Gebäude oder Technik, Mietkosten, Personalkosten des Lagers, Energiekosten, etc. verursacht (Bichler et al. 2011). Für die Kapitalbindungskosten ist die Bestandshöhe und die damit verbundenen Kosten für das gebundene Kapital, Versicherungen, Steuern und Wertminderungen verantwortlich (Bichler et al. 2011). Die Lagerhaltungskosten zu senken ist dabei ein kompliziertes Problem da Zielkonflikte auftreten. Als Beispiel ist der Zielkonflikt der Bestandshöhe zu nennen. Niedrige Bestände verursachen zwar wenig Bestandskosten, gleichzeitig entstehen aber Kosten durch Fehlbestände (ten Hompel 2018). Bei hohen Beständen sind zwar die Kosten durch Fehlbestände niedrig, die Bestandskosten sind aber hoch. Somit gibt es einen optimalen Bestand, der je nach Situation die beiden Einflussfaktoren, Bestandskosten und Kosten durch Fehlbestände, ermittelt werden muss (ten Hompel 2018). Dieser Zielkonflikt ist in der Abbildung 2.1 dargestellt.

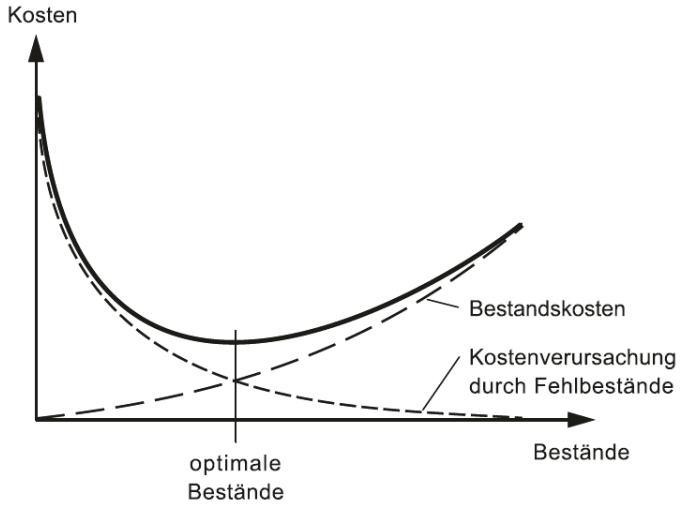

Abbildung 2.1: Zielkonflikt der Lagerhaltung aus ten Hompel et al. (2018, S. 53)

Läger können nach verschiedenen Kriterien unterteilt werden. Unter anderem sind Unterscheidungen nach der Lagertechnik, Lagereinrichtung, der Position des Lagergutes während der Lagerdauer oder des Automatisierungsgrades möglich (Ehrmann 2012).

Nach dem Kriterium der Lagertechnik können Läger in Bodenläger ohne Lagerhilfsmittel, Blockläger, Zeilenläger, oder Regalläger eingeteilt werden (Ehrmann 2012). Die Bodenlagerung ohne Lagerhilfsmittel ist die simpelste Form der Lagerung, bei der die Güter ohne feste Einrichtung auf dem Boden gelagert werden (ten Hompel et al. 2018). Bei der Blocklagerung werden die Güter, gegebenenfalls gestapelt, in zusammenhängenden Blöcken auf dem Boden gelagert (Ehrmann 2012). Bei der Zeilenlagerung werden die Güter mit einem Abstand untereinander in langen Reihen angeordnet (Schulte 2016). Regalläger bestehen zum Teil aus mehreren Ebenen, auf denen Güter in Regalsystemen gelagert werden (Ehrmann 2012).

Läger können auf zwei Arten organisiert werden. Die eine Organisationsform ist die Festplatzlagerung, bei der jedes Gut gleicher Art auf einen bestimmten Lagerplatz eingelagert wird (Vahrenkamp und Kotzab 2012). Diese Form ist durch hohe Zugriffsgeschwindigkeiten und einer hohen Zugriffssicherheit gekennzeichnet (Bichler und Schröter). Die andere Organisationsform ist die chaotische Lagerung. Sie setzt eine EDV-gestützte Lagerverwaltung voraus, die nach Kriterien wie Umschlagshäufigkeit, Zugriffszeit, Gewicht, etc. einen Lagerplatz auswählt (Vahrenkamp und Kotzab 2012). Einer der größten Vorteile der chaotischen Lagerung ist die Lagervolumeneinsparung von bis zu 30% gegenüber der Festplatzlagerung (Bichler und Schröter 2004). Eine Sonderform der chaotischen Lagerung ist die chaotische Lagerung mit Zonung. Verbunden mit einer ABC-Einteilung von Gütern, die nach der Zugriffshäufigkeit vorgenommen wird, werden jedem A- B- und C-Artikel eine bestimmte Zone zugeordnet, innerhalb der eine chaotische Lagerung durchgeführt wird (Schulte 2016).

Es kann auch eine Unterteilung von Lägern in dynamische und statische Läger vorgenommen werden (Bichler und Schröter 2004). Dynamische Läger zeichnen sich durch das Bewegen von Gütern zwischen Ein- und Auslagerung aus. Beispielsweise sind Durchlaufregale dynamische Läger (Schulte 2016). Statische Läger hingegen bewegen die Güter zwischen dem Ein- und Auslagern nicht. Beispiele für statische Läger sind Palettenläger (Ehrmann 2012).

2.1 Palettenläger

Palettenläger sind Läger mit Palettenregalen, in denen Stückgutlagerung mithilfe eines Ladehilfsmittels erfolgt (Jünemann und Schmidt 2000). Als Stückgut gilt ein Gut, das fest ist, sich während des Transportvorganges nicht ändert und als Einheit gehandhabt werden kann. Den Dimensionen sind keine Grenzen gesetzt, außerdem ist es nicht erheblich, ob

es unverpackt, verpackt, aus einem Material, oder aus mehreren Materialien zusammengesetzt ist (Martin 2017). Beispiele für Stückgut sind Pakete, Kisten, Fertigungs- und Montageteile, Flaschen, Maschinen und Container (Martin 2017). Die Ladehilfsmittel, synonym auch Ladungsträger (Martin 2017), zur Stückgutlagerung sind zumeist Europaletten, Chemiepaletten, Corletten oder Gitterboxen (ten Hompel et al. 2018). Die Kombination zwischen einem Stückgut und einem Ladehilfsmittel wird Lagereinheit genannt. Ein Lagerplatz bezeichnet einen Ort, auf denen Lagereinheiten gelagert werden können. Je nach Bauweise können mehrere Lagerplätze ohne bauliche Trennung nebeneinander angeordnet werden. Ein solcher Verbund wird Lagerfach genannt (ten Hompel und Heidenblut 2011).

Palettenläger können in verschiedenen Bauweisen ausgeführt werden, die einen Betrieb entweder als Einplatz- oder Mehrplatzsystem ermöglichen (Jünemann und Schmidt 2000). Bei der Einplatzlagerung werden die Ladehilfsmittel quer, also stirnseitig Richtung Regalstützen, eingelagert und füllen ein ganzes Lagerfach aus (ten Hompel und Schmidt 2010). Wenn ein Palettenlager mit dem Prinzip der Mehrplatzlagerung arbeitet, werden die Lagereinheiten längsseitig in das Lagerfach eingelagert (ten Hompel und Schmidt 2010). Meistens werden bei diesem Prinzip drei bis fünf Lagereinheiten nebeneinander in einem Lagerfach gelagert (ten Hompel und Schmidt 2010). Zur Bedienung von Palettenlägern können sowohl Regalstapler als auch Regalbediengeräte eingesetzt werden (Koether 2007).

Zu den Vorteilen von Palettenlägern zählen eine hochflexible Zugriffsleistung, die Fähigkeit zur Anpassung an geänderte Anforderungen, ein hoher Ordnungseffekt und eine gute Raumnutzung (Bichler und Schröter 2004).

Eine Form von Palettenlägern, die eine besonders hohe Bauweise aufweisen und dadurch vor zusätzliche Herausforderungen gestellt ist, wird im folgenden Abschnitt betrachtet.

2.2 Hochregalläger

Eine Form der Palettenläger stellen Hochregalläger dar, welche seit 1962 in Deutschland zu finden sind (Czeguhn 1985). Mittlerweile sind Hochregalläger, durch ein hohes Maß an Standardisierung und der dadurch ermöglichten Automatisierung, der mit am weitesten verbreitete Lagertyp (Frank 2008).

Sie sind Palettenläger mit einer Bauhöhe ab 12m (ten Hompel et al. 2018). Diese Bauhöhe ist die Höhe, ab der das Lager nicht mehr mit Staplern bedient werden kann (Brandes, 1997, S. 16). Die größten Hochregalläger sind bis zu 200m lang (Frank 2008) und erreichen eine Höhe von bis zu 55m (ten Hompel et al. 2018). Diese Dimensionen können nur durch die sogenannte Silobauweise eines Hochregallagers erreicht werden (ten Hompel et al. 2007), in der Lagerplätze, Dach- und Wandkonstruktion eine bauliche

Einheit bilden (Jünemann 1971). Kleinere Hochregalläger können auch in festen Gebäuden untergebracht werden (ten Hompel et al. 2007). Die Lagerfächer eines Hochregallagers können auch als Regalfach bezeichnet werden (ten Hompel et al. 2018).

Damit die Lagereinheiten nicht in einer einzigen, sehr langen Zeile gelagert werden, werden in einem Hochregallager mehrere Zeilen vorgesehen (ten Hompel et al. 2018). Der Raum zwischen den Zeilen wird Gasse genannt (Schulte 2016). In einer Gasse bewegen sich die Regalbediengeräte (ten Hompel et al. 2018).

Hochregalläger werden als reine Vorratsläger betrieben, in einigen Fällen mit zusätzlicher Kommissionierungsfunktion (Bichler und Schröter 2004). Auch ist keine Einschränkung hinsichtlich bestimmter Warengruppen, Fertigungsbereiche oder Branchen gesetzt (VDI 1994).

Für Anwendungsfälle, in denen eine hohe Lagerleistung benötigt wird, oder auch für Spezialanwendungen, wie z.B. Tiefkühllager, die bis zu -40 Grad Celsius erreichen, bietet sich der Einsatz automatisierter Hochregalläger an (Bichler et al. 2010).

2.3 Automatische Hochregalläger

Ist zwischen der Übernahme eines Lagergutes und der Wiederherausgabe dessen kein menschlicher Eingriff nötig, wird ein Lager als automatisiert bezeichnet (Brandes 1997). Standardisierte Lagereinheiten, neuzeitliche Bautechniken und neue technische Mittel der Mechanik, Elektrotechnik und Datenverarbeitung begünstigten die Automatisierung der Hochregalläger und dem damit verbundenen Wandel vom personalintensiven zum kapitalintensiven Lager (VDI 1994).

Laut VDI (1994) Richtlinie lässt sich ein automatisches Hochregallager in den Fördermittelbereich und den Hochregallagerbereich unterteilen. Der Fördermittelbereich umfasst die für einen automatisierten Ablauf benötigten Fördermittel, wie beispielsweise Rollenbahnen, Kettenförderer, Hängebahnen, Verteilwagen usw. Außerdem werden in diesem Bereich die Lagereinheiten identifiziert und dadurch Material mit Daten verknüpft. Die Stelle, an der die Identifikation stattfindet, wird Identifikationspunkt genannt. Auch beim Auslagern wird die Identität der auszulagernden Lagereinheit überprüft. Dieser Punkt wird Kontrollpunkt genannt und gehört, wie der Identifikationspunkt, zum Fördermittelbereich. Eine schematische Darstellung eines mehrgassigen Hochregallagers und der Positionierung des Identifikationspunkts (in der Abbildung 2.2 i-Punkt) und des Kontrollpunkts (in der Abbildung 2.2 k-Punkt) ist in Abbildung 2.2 wiedergegeben.

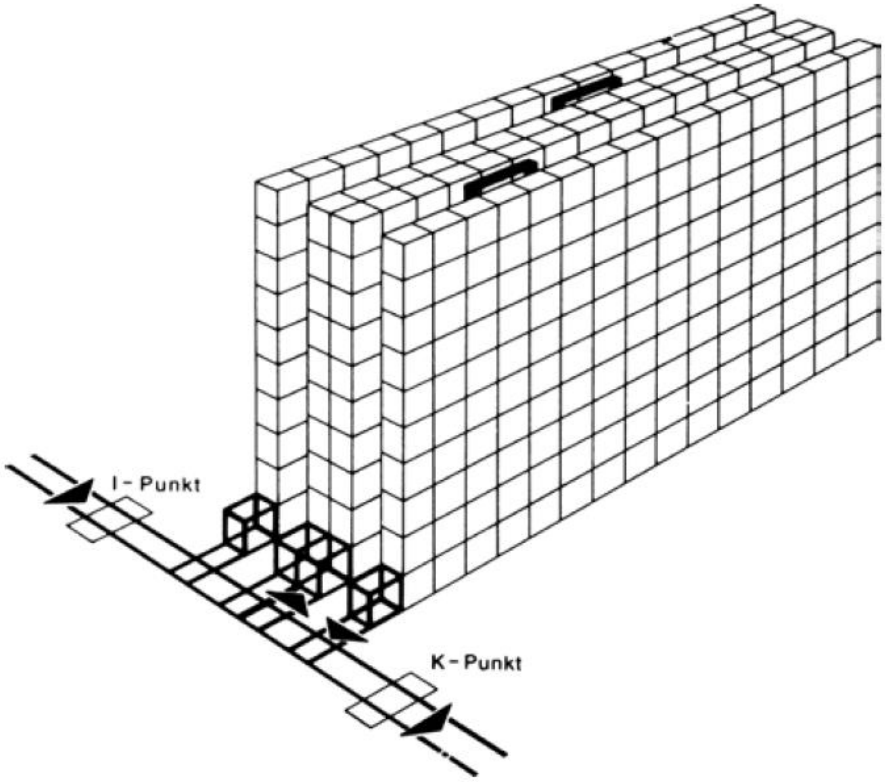

Abbildung 2.2: "Konventionelles" Hochregallager mit dem Zu- und Abfördersystem aus Knepper (1983, S. 219)

Zur Bedienung eines automatischen Hochregallagers werden vollautomatisierte Regalbediengeräte eingesetzt, die auf Trag- und Führungsschienen die Regallagerplätze erreichen können (Arnold und Furmans 2019). In den meisten Fällen wird jeweils ein Regalbediengerät pro Regalgasse eingesetzt, jedoch ist es bei niedrigen Zugriffszahlen auch möglich, mit nur einem Regalbediengerät und Umsetzbrücken an den Regalenden, bzw. Weichensystemen, das gesamte Hochregallager zu betreiben (Bichler und Schröter 2004). Sie bestehen aus einem Fahrwerk, einem Mast und einem Hubschlitten (Koether 2007). Regalbediengeräte sind in der Lage, sich gleichzeitig horizontal und vertikal, also diagonal, zu bewegen und somit Regalfächer auf kürzestem Weg anzufahren (Koether 2007). Es ist jedoch so, dass die Regalbediengeräte in Längsrichtung bis zu drei- bis sechsmal schneller fahren können, als sie horizontal unter Last heben können. Dies führt dazu, dass Hochregalläger in der Regel dreimal länger gebaut werden, als sie hoch sind (Bichler et al. 2010). Zur Aufnahme einer Palette sind die Hubschlitten der Regalbediengeräte mit Lastaufnahmemittel ausgestattet, die sich orthogonal zu den Bewegungsrichtungen des Regalbediengeräts bewegen können, und somit die Lagereinheiten in den Lagerfächern positionieren (ten Hompel et al. 2018). Ein solches Lastaufnahmemittel ist in Abbildung 2.3 zu sehen.

Abbildung 2.3: Lastaufnahmemittel eines Regalbediengeräts aus ten Hompel (2018, S. 201)

Eine andere Betriebsweise ist die Benutzung von Shuttle-Systemen anstatt der vollautomatisierten Regalbediengeräte (Arnold und Furmans 2019). Durch den Einbau von Schienen in jeder Regalebene ist ein parallelisierter Zugriff mittels mehrerer Fahrzeuge möglich (Arnold und Furmans 2019). Es sei außerdem erwähnt, dass automatische Hochregalläger dieselben morphologischen Merkmale eines automatischen Kleinteilelagers aufweisen (Brandes 1997).

Soll eine Lagereinheit in ein Hochregallager eingelagert werden, wird dies Einlagerungsauftrag genannt (Schulte 2012). Ein Auslagerungsauftrag ist der Auftrag, eine Lagereinheit aus dem Hochregallager auszulagern (Schulte 2012). Um einen Auslieferungsauftrag durchzuführen, muss das Regalbediengerät durch das Lastaufnahmemittel die einzulagernde Lagereinheit aufnehmen, zu dem vorgesehenen Lagerregal bringen, dort einlagern und zum Ausgangspunkt zurückkehren. Ein solcher Ablauf wird auch Einzelspiel genannt, welches sich sowohl aus dem produktiven Zeitanteil des Transports der Lagereinheit zum Lagerfach, als auch aus dem unproduktiven Teil der Leerfahrt zurück zum Ausgangspunkt, zusammensetzt (Arnold und Furmans 2019). Die Durchführung eines Auslagerungsauftrags, bei dem zunächst eine Leerfahrt stattfindet und dann der produktive Schritt des Auslagerns folgt, wird auch Einzelspiel genannt. Um die Zeit zu minimieren, in denen unproduktive Leerfahrten gefahren werden, wird das Doppelspiel eingeführt. Bei einem Doppelspiel wird zunächst

eine Lagereinheit eingelagert, jedoch wird nun keine Leerfahrt zum Ausgangspunkt durchgeführt, sondern sofort eine andere Lagereinheit ausgelagert (Arnold und Furmans 2019).

Abbildung 2.4 zeigt ein reales, automatisches, mehrgassiges Hochregallager.

Abbildung 2.4: Mehrgassiges, automatisches Hochregallager aus ten Hompel et al. (2018, S.202)

3 Simulation in Produktion und Logistik

In diesem Kapitel wird zunächst die Methode der Simulation vorgestellt. Daraufhin werden Vorgehensweisen zur Anwendung von Simulation innerhalb einer Simulationsstudie erläutert. Im Anschluss wird eine detaillierte Betrachtung von Daten durchgeführt, die, gruppiert, eine Simulationsdatenbasis darstellen und als Ausgangspunkt der Simulation dienen. Um im weiteren Verlauf der Arbeit ein Experiment zur Datenerfassung an einem physischen Modell durchführen zu können, werden weiterhin Wege zur Datenerfassung aufgezeigt und speziell auf Datenerfassung durch Mikrocontroller, Peripheriegeräte und Sensoren eingegangen. Neben der Datenerfassung wird auch die Datenspeicherung thematisiert, ohne die eine Simulation zu einem späteren Zeitpunkt nicht möglich wäre. Nachdem die Grundlagen gegeben wurden, wofür Daten gebraucht werden und wie sie erfasst werden, werden auch Grundlagen dazu erläutert, wie eine Auswahl von zu erfassenden Daten getroffen werden kann. Dazu wird eine Beziehung zwischen Daten, Kennzahlen und Zielen aufgezeigt und einige, gängige Ziele von Unternehmen, der Logistik, der Lagerhaltung und automatischer Hochregalläger sowie einige Kennzahlen zu deren Beschreibung aus der Literatur genannt. Den Abschluss dieses Kapitels bildet die Betrachtung eines physischen Modells eines automatischen Hochregallagers, welches dieser Arbeit zur Verfügung steht und an dem das experimentelle Erfassen von Daten durchgeführt wird.

3.1 Grundlagen von Simulation in Produktion und Logistik

Simulation ist eine Methode zur Lösung von Problemen, bei der anhand von Experimenten mit Simulationsmodellen Rückschlüsse auf das Verhalten von dynamischen Systemen gezogen werden können (Gutenschwager et al. 2017). Systeme sind nach DIN IEC 60050-351 als Menge von in Beziehung stehender Elemente, die von ihrer Umwelt abgegrenzt sind und als ein Ganzes gesehen werden können, definiert. Dynamische Systeme sind abhängig von der Zeit und durch einen Zeitfortschritt charakterisiert, im Gegensatz zu statischen Systemen, in denen die Ausgangsgröße zeitgleich mit den Eingangsgrößen feststeht (Wunsch und Schreiber 2005). Im Kontext von Produktion und Logistik sind solche Systeme gemeint, die eine enge Verbindung zum Materialfluss besitzen (Gutenschwager et al. 2017). Somit sind auch insbesondere die der Arbeit zugrundeliegenden automatischen Hochregalläger Systeme, auf die die Methode Simulation angewendet werden kann. Die vereinfachte Nachbildung eines Systems wird Modell genannt (Robinson 2004). Als Simulationsmodell werden solche Modelle bezeichnet, die als ablauffähiges Modell zur experimentellen Analyse von Systemen, wie sie die Simulation anhand von Experimenten durchführt, benutzt werden

(Gutenschwager et al. 2017). Der VDI definiert Simulation als "Nachbilden eines Systems mit seinen dynamischen Prozessen in einem experimentierbaren Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind; insbesondere werden die Prozesse über die Zeit entwickelt" (2014). Diese Definition ähnelt der einiger anderer Autoren (vgl. Banks 1998; Law 2015; Robinson 2004).

Für die Untersuchung von Systemen gibt es neben der Simulation weitere Methoden. Zunächst ist es möglich, Experimente am Originalsystem selbst durchzuführen (Law 2015). Ist dies nicht möglich, weil beispielsweise eine Unterbrechung des Betriebs zum Experimentieren nicht wirtschaftlich ist, oder das System noch nicht existiert, müssen die Experimente an einem Modell des Systems durchgeführt werden (Law 2015). Diese Modelle können nach ihrer Beschreibungsform typisiert werden. Gutenschwager et al. (2017) unterscheiden dabei zwischen physischen Modellen und abstrakten, formalen Modellen zu denen mathematische Modelle, grafische Modelle und Computerprogramme gehören. Law (2015) hingegen unterscheidet nur zwischen physischen Modellen und mathematischen Modellen. Tiefergehend zeigt Law (2015) zwei Wege zur Untersuchung von mathematischen Modellen auf: die analytische Lösung und die Simulation. Abbildung 3.1 zeigt die von Law entwickelte Baumstruktur zur Visualisierung von Wegen zur Systemuntersuchung.

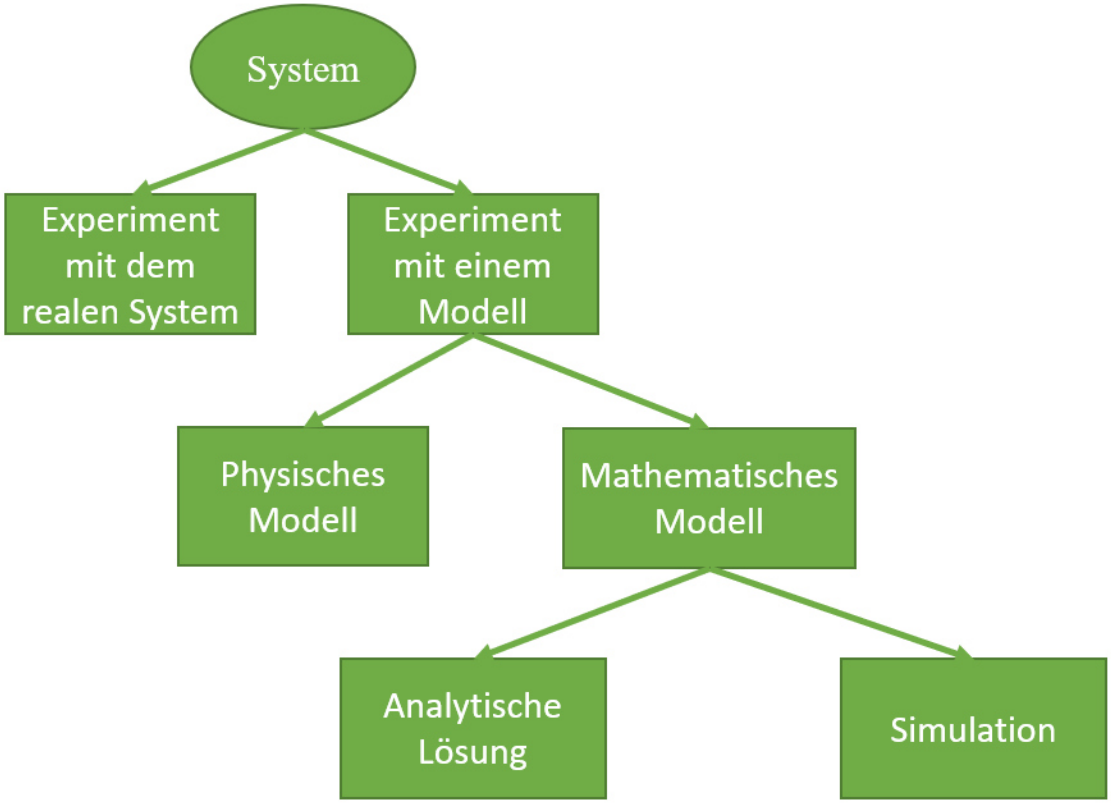

Abbildung 3.1: Überblick über Ansätze zur Untersuchung von Systemen nach Law (2015, S. 4)

In Abbildung 3.1 können an drei Stellen jeweils zwei verschiedene Wege eingeschlagen werden. Anhand dieser Wegfindung kann nachvollzogen werden, wann Simulation eingesetzt werden sollte. Bei der Wahl des ersten Weges muss entschieden werden, ob es sinnvoll ist, die Experimente am eigentlichen, zu untersuchenden System auszuführen, oder an einem Modell. Gründe, die gegen Experimente am eigentlichen System sprechen, können beispielsweise wirtschaftlicher Natur sein, da Anpassungen, verbunden mit Investitionen, am System vorgenommen werden müssten, oder die Anlage nicht zur Wertschöpfung zur Verfügung steht. Existiert das eigentliche System noch nicht, oder befindet es sich gerade noch im Bau, müssen Modelle verwendet werden. Als zweites ist zu entscheiden, ob zum Experimentieren ein physisches oder mathematisches Modell benutzt werden soll. Die Gründe für die Benutzung eines physischen Modells sieht Law hauptsächlich im Bereich von Lernzwecken. Zuletzt werden analytische Methoden und Simulation gegenübergestellt. Analytische Methoden bieten sich immer dann an, wenn ein Modell ausreichend simpel ist. Sind analytische Methoden zu langsam, oder können fachlich nicht realisiert werden, bietet sich Simulation an (Law, 2015).

In der Definition des VDI (2014) von Simulation wird allgemein von experimentierbaren Modellen gesprochen. Somit werden auch physische Modelle technischer Systeme, wie z.B. physische Modelle automatischer Hochregalläger, eingeschlossen. Diese Art von Simulation wird physische Simulation genannt (Robinson 2004).

Neben der Typisierung nach der Beschreibungsform führen Gutenschwager et al. (2017) weitere Klassifikationskriterien für Systeme und Modelle an. In Bezug auf die Zeit und den Systemzuständen kann eine Unterscheidung zwischen diskreter und kontinuierlicher Ausprägung getroffen werden. So werden bei der zeitdiskreten Betrachtung nur eine abzählbare Menge von Zeitpunkten untersucht, wohingegen bei kontinuierlicher Betrachtung unendlich viele Zeitpunkte untersucht werden. Den Unterschied zwischen zustandsdiskreten und zustandskontinuierlichen Systemen zeigen Gutenschwager et al. (2017) an dem Beispiel eines Behälters für Schüttgut. Bei zustandsdiskreter Betrachtung hat das System eine abzählbare Menge an Zuständen, nämlich leer und voll. Die zustandskontinuierliche Betrachtung kann das System mit einem Füllgrad beschreiben, der zwischen 0% und 100% Füllung unendlich viele Werte annehmen kann. Innerhalb eines Systems müssen jedoch nicht nur ausschließlich diskrete bzw. kontinuierliche Zustandsmengen oder diskrete bzw. kontinuierliche Zeitmengen vorkommen, denn auch jegliche Kombination von Teilsystemen unterschiedlicher Ausprägung sind möglich. Solche Systeme werden laut Gutenschwager et al. (2017) als kombinierte Systeme bezeichnet. Wenn Zufälle in einem System eine Rolle spielen werden sie stochastisch genannt. Ist dies nicht der Fall sind sie deterministisch. Auch Law (2015) kategorisiert Modelle mit den Ausprägungsmerkmalen statisch-dynamisch, deterministisch-stochastisch und kontinuierlich-diskret. Basierend darauf unterscheidet er zwischen drei

Simulationsarten: der kontinuierlichen Simulation, der ereignisdiskreten Simulation und der Monte-Carlo-Simulation. Für diverse weitere Klassifikationskriterien für Systeme und Modelle wird auf Gutenschwager et al. (2017) verwiesen Weitere Informationen zu den verschiedenen Simulationsarten sind in Law (2015) zu finden.

In der Produktion und der Logistik findet die Simulation laut VDI (2014) sowohl in innerbetrieblichen logistischen Ketten, in unternehmensübergreifenden Supply Chains und in jeder Phase des Lebenszyklus' technischer Systeme, also während der Planungs-, Realisierungs- und Betriebsphase, Anwendung. Zumeist werden im Bereich von Produktion und Logistik technische Anlagen, wie z.B. automatische Hochregallager, oder Abläufe, wie z.B. Einlagerungsvorgänge von Regalbediengeräten, simuliert (März et al. 2011). Etwa 80% der Untersuchungen werden im Bereich der Planungsphase angesiedelt, die restlichen 20% verteilen sich auf die Realisierungs- und Betriebsphase, wobei die Realisierungsphase einen größeren Anteil einnimmt (Jahangirian et al. 2010; Smith 2003; Gutenschwager et al. 2017). Zur Simulation des dynamischen Verhaltens dieser Systeme wird fast ausschließlich die ereignisdiskrete Simulation eingesetzt (März et al. 2011). Typische Anwendungsfelder in der Planungsphase sind die Ermittlung von Kapazitätsgrenzen, die Findung von Schwachstellen oder die Neuplanung von Anlagen (VDI 2014). In Abbildung 3.2 wir ein schematischer Verlauf der Kosten einer Projektumsetzung gezeigt.

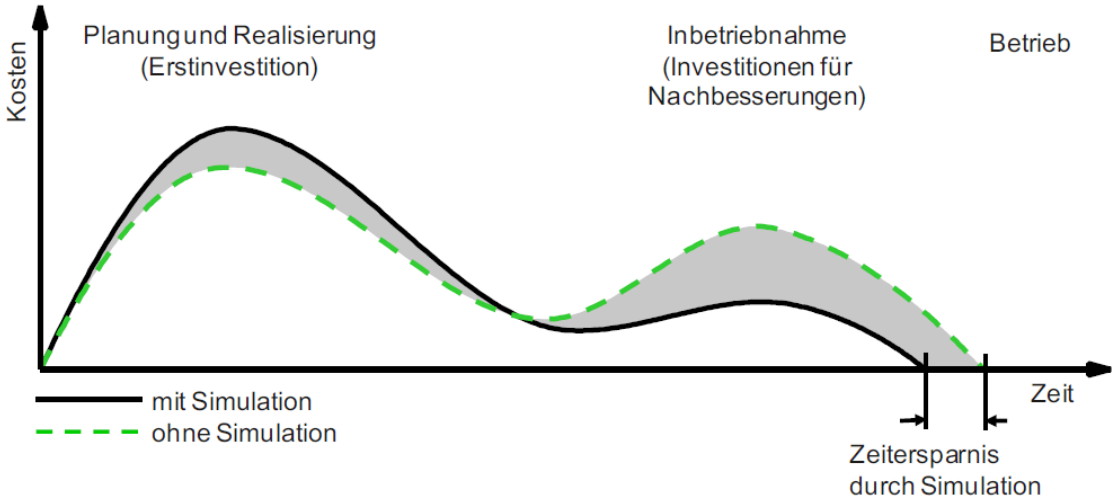

Abbildung 3.2: Schematischer Verlauf der Kosten von der Planung bis zur Inbetriebnahme aus Gutenschwager et al. (2017, S. 48)

Der in Abbildung 3.2 gezeigte Verlauf zeigt, wie sich die Anwendung von Simulation auf eine Projektumsetzung auswirkt. Zwar werden mit Simulation in der Planungs- und Realisierungsphase zunächst höhere Kosten verursacht, jedoch werden diese bei der Inbetriebnahme kompensiert und sogar Ersparnisse erzielt. Zusätzlich birgt der Einsatz

von Simulation eine Zeitersparnis. Projektumsetzungen mit Beteiligung von Simulationen werden auch Simulationsstudien genannt (Bernhard et al. 2007).

3.2 Vorgehensmodell einer Simulationsstudie

Als Simulationsstudie werden solche Projekte bezeichnet, in denen die Methode der Simulation eingesetzt wird (Gutenschwager et al. 2017).

Für das Vorgehen zur Durchführung einer Simulationsstudie gibt es viele verschiedene Herangehensweisen. In der Literatur finden sich unter Anderem Vorschläge von Landry et al. (1983), Banks et al. (2004) und Law (2007). Zwar sind Komplexität, Umfang und Detailgrad der Vorgehensmodelle unterschiedlich, jedoch sind in allen Modellen ähnliche Prozesse, wie z.B. die Systemanalyse, zu identifizieren (Gutenschwager et al. 2017; Robinson 2004). Im deutschsprachigen Raum hat das Vorgehensmodell von Rabe et al. (2008) für den Bereich Produktion und Logistik eine große Bedeutung, wie durch dessen Verwendung in der VDI Richtlinie 3633 (2014) zu sehen ist. Auch in dieser Arbeit wird, wenn von einem (Simulations-)Vorgehensmodell gesprochen wird, auf das von Rabe et al. (2008) entwickelte verwiesen. In Abbildung 3.3 ist das Simulationsvorgehensmodell nach Rabe et al. (2008) zu sehen.

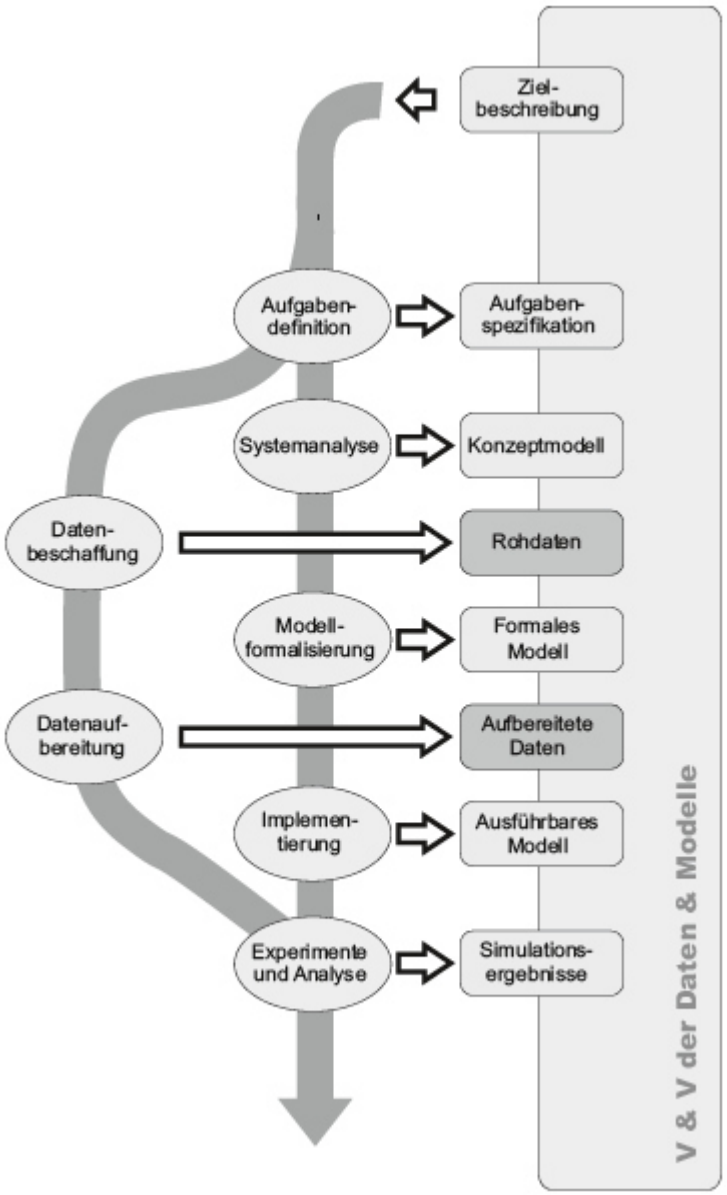

Abbildung 3.3: Simulationsvorgehensmodell nach Rabe et al. (2018, S. 5)

Das Vorgehensmodell aus Abbildung 3.3 gliedert sich in sieben Phasen, die in der Abbildung durch Ellipsen dargestellt sind. Ein Richtungspfeil zeigt an, in welcher Reihenfolge die Phasen durchlaufen werden. Nicht alle Phasen werden sukzessive durchlaufen, da zwei der Phasen parallel ausgeführt werden. Zu Jeder Phase sind zugehörige Phasenergebnisse, durch abgerundete Rechtecke gekennzeichnet (ausgenommen die "Zielbeschreibung"), mit Pfeilen zugeordnet. Die Eingangsinformation ist die "Zielbeschreibung". Außerdem ist ein großes Rechteck, das alle Phasenergebnisse und die "Zielbeschreibung" teilweise einschließt, mit der Beschriftung "Verifikation und Validierung der Daten & Modelle" zu sehen, welches verdeutlichen soll, dass die Verifikation und Validierung (V&V) ständig während der Simulationsstudie auf die jeweiligen Phasenergebnisse angewendet werden soll.

Die sieben Phasen können wie folgt beschrieben werden:

- Aufgabendefinition: In der Phase der Aufgabendefinition wird die Zielbeschreibung des Projekts präzisiert und vervollständigt. Das Ergebnis ist eine Aufgabenspezifikation, der alle Beteiligten zustimmen können und im vorgesehenen Zeit- und Kostenrahmen umsetzbar ist.
- Systemanalyse: Während der Phase der Systemanalyse wird das zu entwickelnde Simulationsmodell mit seinen Zielsetzungen, Eingaben, Ausgaben, Elementen und Beziehungen, Annahmen und Vereinfachungen dokumentiert. Hieraus entsteht ein Konzeptmodell zur Beschreibung des Übergangs, welche Aufgaben gelöst werden sollen und auch wie diese Lösung aussehen könnte. Der Umfang des Modells und der Grad der Detaillierung wird hier festgelegt.
- Modellformalisierung: Die Elemente und Beziehungen des vorher erarbeiteten Konzeptmodells werden während der Modellformalisierung in ein formales Modell überführt, damit eine Implementierung ohne weitere Analysen und Rückfragen möglich ist.
- Implementierung: In der Phase der Implementierung wird das formale Modell umgesetzt. Dabei muss auf die Eigenschaften des später verwendeten Simulationswerkzeugs Rücksicht genommen werden. Das Phasenergebnis stellt das ausführbare Modell dar.
- Experimente und Analyse: Diese Phase führt die aufbereiteten Daten und das ausführbare Modell zusammen. Die Schlussfolgerungen für das zu untersuchende System werden hier durch Simulation gewonnen.

Die beiden folgenden Phasen "Datenbeschaffung" und "Datenaufbereitung" werden parallel zu den Phasen "Systemanalyse", "Modellformalisierung" und "Implementierung" ausgeführt:

- Datenbeschaffung: Das Ziel der Datenbeschaffung ist die Bereitstellung von Rohdaten, die in Struktur und Format möglichst unmittelbar aus den Datenquellen hervorgehen. Welche Daten bereitzustellen sind, geht aus der Aufgabenspezifikation und dem Konzeptmodell hervor. Eine Besonderheit durch die parallele Ausführung der Phasen ist, dass Restriktionen in der Beschaffbarkeit der Daten das Konzeptmodell beeinflussen können.
- Datenaufbereitung: Die Aufgabe der Datenaufbereitung ist die Überführung der Rohdaten in eine Form, die für das ausführbare Modell verwendbar ist.

Die zu jeder Phase parallel auszuführende V&V hat die Aufgabe, die Gefahr von fehlerhaften Aussagen und der damit verbundenen Gefahr von Fehlentscheidungen zu vermindern. Der Verifikationsbegriff steht dabei für die Überprüfung, ob eine Transformation zwischen Beschreibungsarten eines Modells korrekt verlaufen ist. Die

Validierung überprüft das Modell kontinuierlich dahingehend, dass das Modell das Verhalten des abgebildeten Systems mit ausreichend hoher Genauigkeit wiedergibt. Eine weitere Aufgabe der Validierung ist die Überprüfung, ob die aus einem realen System stammenden Daten valide sind und für eine Simulationsstudie verwendet werden können. Für eine detailliertere Beschreibung des Simulationsvorgehensmodell wird auf Rabe et al. (2008) verwiesen.

Die Phasen Datenbeschaffung und Datenaufbereitung können laut Bernhard und Wenzel (2005) und Bernhard et al. (2007) in detaillierte Phasen eingeteilt werden. Zur differenzierteren Betrachtung wird die Phase „Datenbeschaffung“ aus dem Vorgehensmodell nach Rabe et al. (2008) durch die drei Phasen „Zieldefinition“, „Informationsidentifikation“ und „Erhebung & Erfassung“ ersetzt. Die Phase der „Datenaufbereitung“ wird durch die Phasen „Strukturierung & statistische Analyse“ und „Datennutzbarkeitsprüfung“ ersetzt. Laut Bernhard et al. (2007) werden durch den höheren Detailgrad des Vorgehensmodells der Daten- und Informationsgewinnungsprozess besser abgebildet und eine größere Wichtigkeit innerhalb des Vorgehensmodells klar. Das (Simulations-)Vorgehensmodell nach Bernhard et al. (2007) wird in der folgenden Abbildung 3.4 gezeigt.

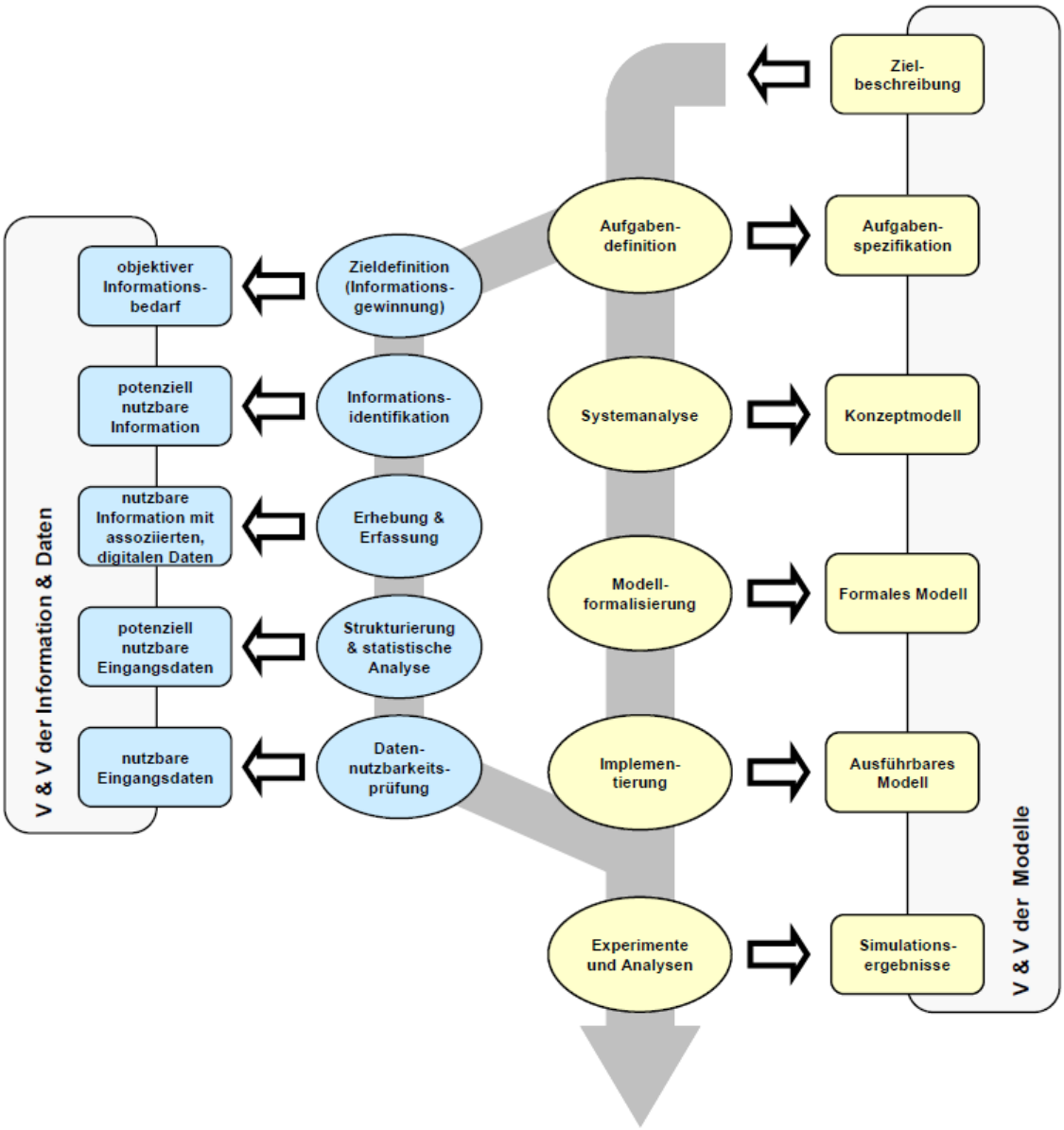

Abbildung 3.4: Simulationsvorgehensmodell nach Bernhard et al. (2007, S. 6)

Im nächsten Abschnitt wird der Begriff der „Daten“, welcher eine große Bedeutung in den Vorgehensmodellen nach Rabe et al. (2008) und Bernhard et al. (2007) spielt, näher betrachtet.

3.3 Daten

In den vorangehenden Abschnitten 3.1 und 3.2 wurde gezeigt, dass zur Entwicklung eines Simulationsmodells Daten notwendig sind. Eine Definition von Daten und eine Abgrenzung zu anderen Begriffen kann anhand der Wissenstreppe nach North, deren erste drei Stufen in Abbildung 3.5 gezeigt werden, formuliert werden.

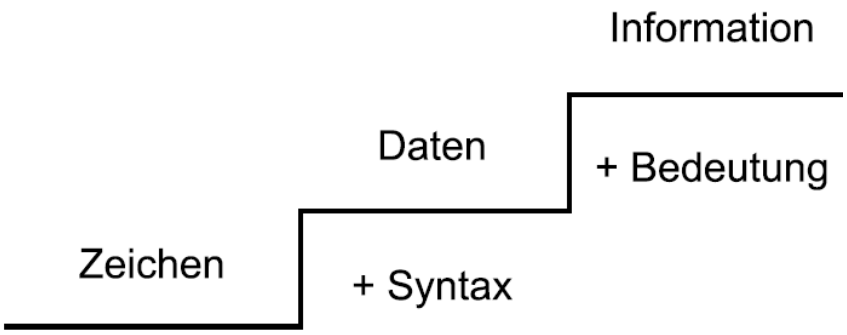

Abbildung 3.5: Ausschnitt der Wissenstreppe nach North (2016, S. 37)

Die drei Treppenstufen entsprechen jeweils einem der Begriffe „Zeichen“, „Daten“ und „Informationen“. Die unterste und damit elementarste Treppenstufe ist die der Zeichen, die jeweils einen einzelnen Buchstaben, eine Ziffer oder ein Sonderzeichen darstellt. Eine Menge von Zeichen, die durch eine Ordnungsregel (Syntax) gebildet wird, ist laut North als Datum (Singular von Daten) definiert. Daten nehmen die zweite Stufe in der Wissenstreppe ein und sind gegenüber der nächsten Stufe, die der Informationen, dadurch abgegrenzt, dass Daten nicht mit einer Bedeutung verknüpft sind. Es existieren noch weitere Treppenstufen mit den Benennungen „Wissen“, „Handeln“, „Kompetenzen“ und „Wettbewerbsfähigkeit“, da sie jedoch nicht von Relevanz für diese Arbeit sind, wird für weitere Betrachtung auf North verwiesen.

Diese Definition und Abgrenzung des Begriffs Daten deckt sich mit denen verschiedener Autoren (vgl. Krcmar 2011; Probst et al. 2006) und hat sich allgemein in der Fachliteratur durchgesetzt (Cleve und Lämmel 2014; Scheidler 2017). Dürr (2018, S. 38) kommt zu der Feststellung, dass „Daten die Mittel zur Formulierung von Informationen betrieblicher Relevanz sind“.

Laut Cleve und Lämmel (2014) ist es möglich, Daten zu typisieren. Sie unterscheiden dabei zwischen drei Datentypen:

- Nominale Daten: Können nur verglichen werden, ob sie identisch sind, es ist also keine Rangfolge bildbar. Beispiel: schwarz, weiß, rot
- Ordinale Daten: Haben zwar eine Ordnungsrelation, eignen sich aber nicht zum Rechnen. Beispiel: klein, mittel, groß
- Metrische Daten: Können geordnet werden und zum Rechnen benutzt werden. Beispiel: 1, 2, 3

Weiterhin können metrische Daten tiefergehend unterteilt werden:

- Diskrete Daten: Es gibt eine endliche Anzahl von möglichen Werten. Beispiel: Schulnoten

- Kontinuierliche Daten: Können beliebige reelle Zahlenwerte innerhalb eines Definitionsbereichs annehmen. Beispiel: Geschwindigkeit eines Autos

Ein Datum allein reicht nicht aus, um eine Simulation zu treiben. Daher werden alle Daten, die für die Simulation benötigt werden, konsolidiert.

3.4 Simulationsdatenbasis

In Abschnitt 3.3 wurde der Begriff „Daten“ definiert und erklärt. Ein wichtiger Nutzen von Daten ist ihre Verwendung für Simulation, die ohne Daten nicht möglich wäre (Robinson 2004). Die Simulationsdatenbasis, in der die Daten gesammelt sind, stellt dabei die Schnittstelle dar, durch die ein Simulationswerkzeug auf Daten zugreifen kann (VDI 2014). Daher sollten die Daten der Simulationsdatenbasis in einem gängigen Datenformat zur Verfügung gestellt werden (VDI 2014).

Grundsätzlich besteht eine Simulationsdatenbasis aus Systemlastdaten, Organisationsdaten und technischen Daten (VDI 2014). Eine Erklärung der drei genannten Begriffe ist in Wenzel und Meyer (1993) zu finden. Demnach bezeichnen technische Daten die Charakteristika des Betrachtungsgegenstands, wie Längen oder Geschwindigkeiten. Organisatorische Daten enthalten Daten über Strategien und Verhaltensregeln. Dazu zählen z.B. die Arbeitszeitorganisation, oder die Ressourcenzuordnung. Alle Daten über dynamische Prozesse, wie z.B. Produktions- oder Transportaufträge, werden Systemlastdaten genannt.

Die Daten einer Simulationsdatenbasis unterliegen Qualitätsansprüchen. Eine hohe Qualität der Daten muss gegeben sein, da die Simulationsdatenbasis den Grundpfeiler für jede Berechnung einer Simulation darstellt, und somit die Ergebnisse einer Simulation auch nur so gut sein können, wie die Simulationsdatenbasis selbst (Bode und Hoya 1985). Schulte (2012) definiert für logistische Daten zur Planung und Steuerung von System in Produktion und Logistik die Qualitätsansprüche wie folgt:

- Aktualität
- Verfügbarkeit
- Zuverlässigkeit
- Sicherheit
- Genauigkeit

Generell sollte die Simulation eines Systems möglichst weitgehend auf Realdaten, oder zumindest von Realdaten abgeleiteten Daten, basieren, um Laborbedingungen zu vermeiden (VDI 1997). Eine Möglichkeit, reale Daten für eine Simulationsdatenbasis zu bekommen, liegt in der Datenerfassung (Schulte 2016).

3.5 Datenerfassung

Im vorherigen Abschnitt 3.4 wurde beschrieben, dass für eine erfolgreiche Simulationsstudie eine Simulationsdatenbasis verwendet werden muss, deren Daten eine hohe Qualität aufweisen und möglichst durch Datenerfassung an realen Systemen erhoben wurden.

Hansen (1986, S. 433) definiert Datenerfassung als „die Entnahme von Daten realer Prozesse nach definierten Anforderungen der ihnen zugeordneten Datenverarbeitungsprozessen; diese Anforderungen spezifizieren im Einzelnen den Entnahmeprozess hinsichtlich des materiellen Inhalts der Daten, der Form der Daten und der Zeit“. Nach DIN 19222 (1985) stellt die Datenerfassung den Prozess des Messens, Zählens, oder der Signalumformung dar, um entweder analoge oder digitale Daten zu gewinnen. Daten, die durch Messungen erfasst wurden, werden auch Messdaten genannt (Bernstein 2014).

Datenerfassung kann manuell, halbautomatisch, oder automatisch durchgeführt werden (Heinzl und Nusswald 1996). Automatische Datenerfassung werden solche Situationen genannt, in denen ein Gerät zur Datenerfassung selbst die Messung auslöst, durchführt und das Ergebnis erhält (Heinzl und Nusswald 1996). Eine Erfassung von Messdaten ist kaum noch ohne Verbindung zu einem Computer üblich (Bernstein 2014). Eine Unterscheidung der Datenerhebung kann hinsichtlich der Intention der Erhebung getroffen werden. Werden für eine Simulationsstudie schon bestehende Daten benutzt, wird von einer Sekundärerhebung gesprochen und der Prozess, Daten spezifisch für den Zweck der Simulation zu erheben, wird Primärerhebung genannt (Gutenschwager et al. 2017).

Die Datenerfassung ist ein wichtiger Teil einer Simulationsstudie, wie die Vorgehensmodelle von Rabe et al. (2008) und Bernhard et al. (2013) zeigen (vgl. Abschnitt 3.2). In der Praxis erfordert die Datenerfassung einen hohen Anteil an Arbeitszeit einer Simulationsstudie und im Verhältnis zur Datenverarbeitung liegt ihr Anteil bei bis zu 90% der aufgewendeten Zeit (Hansen, 1986). Absolut gesehen, kann der Prozess der Datenerfassung bis zu 40% der gesamten Projektzeit ausmachen (Trybulla 1994) und stellt eines der größten Gründe für das Scheitern von Simulationsstudien dar (Perera und Liyanage 2000).

Trotz ihrer Wichtigkeit stellen sowohl Onggo und Hill (2014) als auch Skoogh und Johansson (2008) in Literaturrecherchen fest, dass Methoden zur Datenbeschaffung und Datenerfassung, nur unzureichend thematisiert werden, selbst unter populären Büchern über Simulation wie Pidd (2003), Law (2007), Banks et al. (2004) und Robinson (2004).

Eine wichtige Komponente moderner Datenerfassung stellt der Mikrocontroller dar (Bringschulte und Ungerer 2010), welcher im anschließenden Abschnitt vorgestellt wird.

3.6 Mikrocontroller und Peripheriegeräte

Mikrocontroller sind elektronische Bauteile, die zur Lösung von Steuerungs-, Kommunikations- und Datenverarbeitungsaufgaben eingesetzt werden (Brinkschulte und Ungerer 2010). Sie bestehen aus einem Mikroprozessor, der die Zentraleinheit, auch bekannt als Central Processing Unit oder CPU, des Systems darstellt, einem Programmspeicher, einem Arbeitsspeicher und einigen Peripheriegeräten (Schossig 1993). Zu den Peripheriegeräten, wie alle Komponenten genannt werden, die nicht der Prozessor oder einer der Speicher selbst sind, gehören unter anderem Input/Output Ports, Analog/Digital Wandler (A/D-Wandler), Zeitgeber und serielle Schnittstellen (Brinkschulte und Ungerer 2010).

Input/Output Ports (I/O Ports) sind die Ein- und Ausgänge, die die Schnittstelle des Mikrocontrollers zur Außenwelt, durch Übertragung von Daten und Signalen, darstellen (Schossig 1993). Zu einem I/O Port gehören mehrere Input/Output Pins (I/O Pins), auf denen jeweils ein Datum oder ein Signal übertragen werden kann (Wüst 2010). Die Pins eines I/O Ports können nur digitale Signale lesen und ausgeben, also die Werte „LOW" und „HIGH", was den Werten 0 und 1 entspricht (Schossig 1993).

Um auch analoge Signale, wie z.B. das Signal eines Temperatur-Sensors, verarbeiten zu können, ist ein Mikrocontroller mit A/D-Wandlern ausgestattet, der die analogen Signale in die für den Mikrocontroller verarbeitbaren digitalen Werte umwandelt (Brinkschulte und Ungerer 2010).

Der Zähler ist ein Peripheriegerät, das Ereignisse, wie z.B. Takte oder Spannungsabfälle, zählen kann (Schossig 1993). Ein Zähler kann auch softwaretechnisch realisiert werden, indem eine Variable im Programmcode des Mikrocontrollers das Zählen übernimmt (Schossig 1993).

Ein weiteres wichtiges Peripheriebauteil ist der Zeitgeber. Mit diesem Bauteil ist es möglich, eine Zeitmessung vorzunehmen, wie z.B. die vergangene Zeit seit Beginn des Programmstarts (Schossig 1993). Um die Zeitmessung durchzuführen, zählt das Bauteil die Anzahl von einem ihm zugeführten Takt, der entweder extern oder intern zugeführt werden kann (Brinkschulte und Ungerer 2010). Jedoch ist zu beachten, dass durch Verwendung des internen Taktes eine ungenaue Zeitmessung durchgeführt wird (Wannemacher und Halang 1994). Der Zeitgeber kann nur Zeitspannen relativ zum Beginn des Startens des Mikrocontrollers messen, die Bestimmung der aktuellen realen Uhrzeit oder des Datums sind mit ihm nicht möglich (Schossig 1993). Für diesen Zweck werden Echtzeituhren eingesetzt, die zwar auch wie der interne Zeitgeber des

Mikrocontrollers funktionieren, durch eine eigene, langlebige Batterie aber über einen deutlich größeren Zeitraum die Zeit messen können (Wüst 2011).

Ein Mikrocontroller ist meist mit Peripheriegeräten zur seriellen Übertragung ausgestattet (Schossig 1993). Zu den Übertragungsprotokollen, die benutzt werden können, gehören z.B. Universal Asynchronous Receiver/Transmitter (UART), Serial Peripheral Interface (SPI) und Inter-Integrated-Circuit-Bus (I^2C) (Gehrke et al. 2016). Sie werden benutzt, um mit anderen Peripheriegeräten wie z.B. Sensoren oder auch einem Computer zu kommunizieren und Daten auszutauschen (Gehrke et al. 2016). Die in der Praxis hauptsächlich genutzte serielle Schnittstelle zwischen einem Mikrocontroller und einem Computer ist die UART-Schnittstelle (Gehrke et al. 2016). Für eine tiefergehende Diskussion der Übertragungsprotokolle und ihrer Funktionsweise wird auf Gehrke et al. (2016) verwiesen.

Im Folgenden wird das Mikrocontroller-Board "Arduino Mega 2560" der Plattform Arduino vorgestellt, welche das am weitesten verbreitete Mikrocontroller-Board für die Forschung anbietet (Nayyar und Puri 2016). Arduino ist eine Plattform für open-source Hardware und Software, die auch eine für Mikrocontroller optimierte Entwicklungsumgebung für Programme zur Verfügung stellt (Arduino 2020). Die Vorteile liegen in der Plattformunabhängigkeit, Kosteneffektivität, einfachen Programmierung, open-source Hardware und open-source Software (Nayyar and Puri 2016). Weitere Informationen zur Arduino Plattform sind auf der Website von Arduino zu finden (Arduino 2020).

Der Arduino Mega 2560 ist ein Mikrocontroller-Board, das auf dem ATmega2560 Mikrocontroller basiert (Arduino 2020). Einige der wichtigsten Eigenschaften des Boards sind in Tabelle 2.1 aufgeführt

Tabelle 2.1: Auszug aus den Eigenschaften des Arduino Mega 2560 (Arduino 2020)

Merkmal	Ausprägung
Mikroprozessor	ATmega2560
Betriebsspannung	5V
Interner Spannungswandler	3,3V
I/O Pins	54
Konnektivität	USB
Übertragungsprotokolle	UART, SPI, I^2C

Aufgrund der in Tabelle 2.1 vorgestellten Eigenschaften bewerten Nayyar und Puri (2016) den Arduino Mega 2560 als Mikrocontroller-Board für Fortgeschrittene, das für

komplexe Experimente, wie mit einem Roboter, einen leistungsstarken Mikrocontroller mit vielen benötigten Peripheriegeräten zur Verfügung stellt. In Abbildung 3.6 ist das vorgestellte Mikrocontroller-Board zu sehen.

Abbildung 3.6: Arduino Mega 2560 (Arduino 2020)

Neben den Peripheriegeräten kann sich ein Mikrocontroller auch Sensoren, welche im nächsten Abschnitt beschrieben werden, zunutze machen.

3.7 Sensoren

3.7.1 Grundlagen von Sensoren

Hering und Schönfelder (2018, S. 1) definieren Sensoren als Bauteile "zur quantitativen und qualitativen Messung von physikalischen, chemischen, klimatischen, biologischen und medizinischen Größen", somit messen Sensoren keine elektrischen Größen. Ein Sensor besteht aus den zwei Bestandteilen "Sensor-Element" und "Auswerte-Elektronik" (Hüning 2016). Innerhalb eines Sensors werden durch das Sensor-Element eine Eingangsgröße, welche die zu messende Größe darstellt, und einer Störgröße in ein elektrisches Ausgangssignal umgewandelt, welches anschließend durch die Auswerte-Elektronik zu einem Sensor-Ausgangssignal aufbereitet wird (Hering und Schönfelder 2018). Dieses Prinzip wird schematisch in Abbildung 3.7 wiedergegeben.

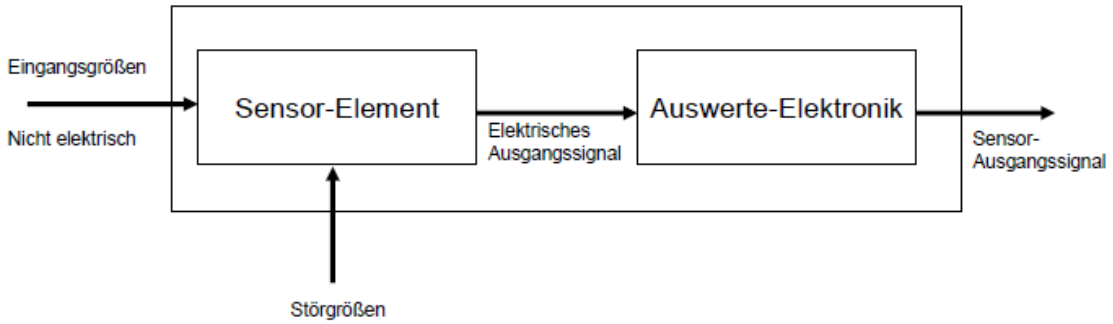

Abbildung 3.7: Wirkprinzip von Sensoren aus Hering und Schönfelder (2018, S. 1)

Sensoren können nach verschiedenen Kriterien eingeteilt werden. Zum einen werden "aktive Sensoren", die die Umwandlung der Messgröße in eine elektrische Größe ohne äußere Hilfsspannung vornehmen, wie z.B. eine Lichtschranke, und "passive Sensoren", die eine Hilfsspannung benötigen, wie z.B. ein Dehnungsmessstreifen, unterschieden (Hering und Schönfelder 2018). Weitere Einteilungen können hinsichtlich der zu messenden Größe selbst, z.B. der Temperatur, oder dem Messprinzip, wie z.B. optoelektronischer Sensoren, vorgenommen werden (Schanz 1988).

3.7.2 Gängige Sensoren

In diesem Abschnitt wird eine Auswahl von gängigen optoelektronischen Sensoren und Sensoren zur automatischen Identifikation vorgestellt. Für eine ausführlichere Beschreibung gängiger Sensoren sei auf Bernstein (2014) verwiesen.

Zu den wichtigsten optoelektronischen Sensoren gehören Einweg-Lichtschranken und Reflexions-Lichttaster, mit denen Objekte detektiert werden können (Hering und Schönfelder 2018).

Bei Einweg-Lichtschranken werden Lichtsender und Lichtempfänger getrennt positioniert, dabei muss jedoch sichergestellt werden, dass der Lichtstrahl des Senders auf den Empfänger fällt (Schanz 1988). Die Funktionsweise einer Einweg-Lichtschranke wird in Abbildung 3.8 schematisch dargestellt. Wie dort zu sehen ist, wird durch die Unterbrechung des Lichtstrahls, beispielsweise durch ein Objekt, die Helligkeit am Empfänger verändert.

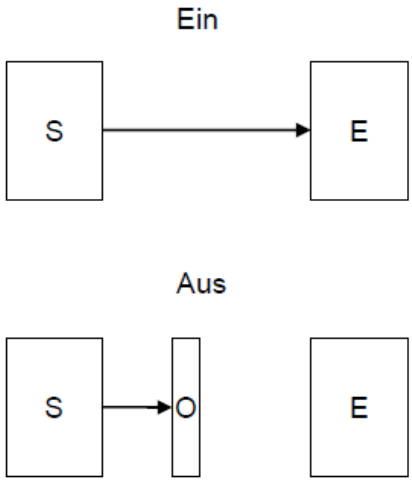

Abbildung 3.8: Wirkprinzip einer Einweg-Lichtschranke aus Hering und Schönfelder (2018, S. 83)

Bei Reflexions-Lichttastern, die im sogenannten Reflexionsbetrieb arbeiten, werden Lichtsender und -empfänger in einem Gehäuse untergebracht (Schanz 1988). Das Licht des Senders wird an dem zu detektierenden Objekt reflektiert und durch den Empfänger aufgenommen. Ist das Objekt nah genug am Reflexions-Lichttaster, wird ein ausreichend hoher Anteil von Licht reflektiert und der Sensor sendet ein entsprechendes Signal (Bernstein 2014). Bei Reflexions-Lichttastern ist darauf zu achten, dass das zu detektierende Objekt einen ausreichend hohen Anteil an Licht reflektieren kann (Hering und Schönfelder 2018). Die Funktionsweise des Reflexions-Lichttasters kann in Abbildung 3.9 nachvollzogen werden.

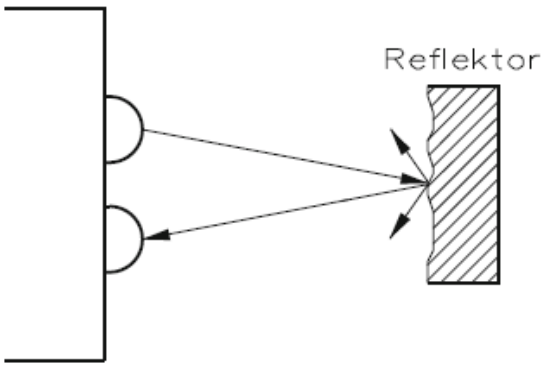

Abbildung 3.9: Wirkprinzip eines Reflexions-Lichttasters aus Bernstein (2014, S. 288)

Eine automatische Identifikation von Objekten wird für gewöhnlich durch einen "Barcode-Laserscanner", eine "Auto-Ident-Kamera" oder ein "RFID-Lesegerät" ermöglicht (Hering und Schönfelder 2018). Bei Verwendung von Barcode-

Laserscannern wird auf dem Objekt ein Barcode, welcher Informationen in einem binären Code, der durch eine Abfolge von parallelen, schwarzen Strichen unterschiedlicher Breite codiert ist, platziert, der durch den Scanner erfasst werden kann (ten Hompel et al. 2008). Da der Barcode-Laserscanner optisch arbeitet, wird eine direkt Sichtlinie auf den Barcode benötigt (Hesse und Schnell 2018). Durch den einfachen Aufbau, einer weitreichenden Standardisierung und geringe Kosten, stellen Barcodes die in der Praxis am weitesten verbreitete Methode zur Objektidentifikation dar (ten Homel et al. 2018). Durch Auto-Ident-Kameras können, entgegen der eindimensionalen Codes des Barcodes, auch zweidimensionale "Matrixcodes" erkannt werden, über die eine größere Menge von Daten über einen kleineren, auf dem Objekt befindlichen Code, hinsichtlich der Abmaße, übertragen werden kann (Hering und Schönfelder 2018). Eine weitere Anwendung von Auto-Ident-Kameras ist die Erkennung von Schriften auf einem Objekt (Hering und Schönfelder 2018). Da auch die Auto-Ident-Kamera optisch arbeitet, wird ein direkter Sichtkontakt zu dem zu scannenden Code erfordert.

RFID-Lesegeräte nutzen induktive oder elektromagnetische Wellen, um binär codierte Daten eines RFID-Transponders zu empfangen und benötigen somit keinen direkten Sichtkontakt zu dem Objekt, sondern einzig die Reichweite des RFID-Lesegeräts zum Empfangen der Wellen ist der limitierende Faktor (ten Hompel et al. 2008). Die auf dem Objekt zu platzierenden RFID-Transponder existieren in verschiedenen Bauformen, z.B. als Etikett oder Plastikkarte, und verschiedenen Funktionsweisen. Zum einen werden aktive RFID-Transponder versendet, die von sich aus ein Signal senden können, das das RFID-Lesegerät empfangen kann (ten Hompel et al., 2018). Passive Sensoren hingegen haben keine eigene Stromversorgung und können die Signale nicht selbstständig aussenden (Hunt et al. 2007). Ihnen wird durch das RFID-Lesergerät bei einem Lesevorgang Strom induziert, der sich, sobald ausreichend Strom induziert wurde, entlädt und dabei elektromagnetische Wellen entsprechend des gespeicherten Codes des RFID-Transponders aussendet, welche vom RFID-Lesegerät empfangen werden (Hunt et al. 2007). Neben dem erwähnten Vorteil, dass kein direkter Sichtkontakt zum Objekt bestehen muss, liegen die Vorteile der RFID-Technologie in der Beschreibbarkeit von Transpondern, die dynamisch an z.B. Kundenwünsche angepasst werden können, der Möglichkeit, mehrere Objekte auf einem Ladehilfsmittel gleichzeitig zu erfassen, und der Fähigkeit, große Mengen an Daten zu speichern (Hänisch 2017).

3.8 Speicherung von Daten

Damit auf erfasste Daten zu späteren Zeitpunkten zugegriffen werden kann, beispielsweise durch ein Simulationswerkzeug, müssen sie persistiert werden. Simulationswerkzeuge können über Schnittstellen, wie Dateischnittstellen, die

Textdateien austauschen, oder Datenbankschnittstellen, auf gespeicherte Daten zugreifen (Gutenschwager et al. 2017). Typische Formate für Textdateien stellen dabei das „Comma Separated Values-Format“ oder das „Extensible Markup Language-Format“ (XML) dar (Vonhoegen 2018).

Besonders XML stellt ein in der Industrie weit verbreitetes Textformat dar, welches einen hohen Standardisierungsgrad besitzt (Frank 2001). XML ist eine Auszeichnungssprache, die eine Untermenge der „Standard Generalized Markup Language“ ist (Moos 2008) und durch das World Wide Web Consortium, auch W3C genannt (W3C 2020), definiert wird.

Vorteile von XML-Dokumenten liegen in der Maschinen- und Menschenlesbarkeit, in der selbsterklärenden Natur einer Auszeichnungssprache, der Portabilität und der guten Strukturierung (Bourret 2005). Nachteilig sind die geringe Geschwindigkeit beim Lesen und Schreiben des Dokuments, sowie der hohe Aufwand zur Strukturierung eines XML-Dokuments (Bourret 2005). Bourret (2005) weist darauf hin, dass, wenn die grundlegendste Definition von Datenbanken, nämlich dass sie eine Sammlung von Daten sind, betrachtet wird, auch XML-Dokumente als Datenbank gesehen werden können.

Zu Beginn eines XML-Dokuments wird die XML-Deklaration angegeben, welche zumindest die Versionsangabe von XML und die Codierung enthält (Skulschus und Wiederstein 2012). Der weitere Aufbau eines XML-Dokuments besteht aus XML-Elementen, im englischen auch Tags genannt, die mit einem Start-Tag und einem End-Tag die Inhalte, wie z.B. Daten, umschließen (Vonhoegen 2018). Die Tags haben keine vordefinierte Bedeutung, sondern können durch den Nutzer definiert werden (Skulschus und Wiederstein 2012). Attribute können innerhalb des Start-Tags, zur zusätzlichen Beschreibung der eingeschlossenen Inhalte, benutzt werden (Vonhoegen 2018).

Um zu überprüfen ob XML-Dokumente eine interpretierbare Struktur aufweisen, können sie auf den Zustand „wohlgeformt“ untersucht werden (Vonhoegen 2018). Das W3C (W3C 2020) definiert die Regeln für ein wohlgeformtes Dokument wie folgt:

- Ein XML-Dokument hat nur ein Wurzelelement, dem alle anderen untergeordnet sind
- Jedes Element braucht einen Start-Tag und einen End-Tag
- Elemente in einem Dokument müssen ebenentreu paarig verschachtelt werden
- Die Attribute eines Tags dürfen nicht denselben Namen haben

Zwar wurde bisher die Fragen beantwortet, wofür Daten benötigt werden und wie sie zu beschaffen sind, jedoch wurde die Auswahl von Daten bisher nicht erläutert. Für eine Vorgehensweise zur Auswahl von Daten wird zunächst Wissen über Kennzahlen und Ziele benötigt, welches im nächsten Abschnitt vermittelt wird.

3.9 Kennzahlen

3.9.1 Grundlagen von Kennzahlen

Kennzahlen sind Werte, die als quantitative Bewertungsgrundlage eines Unternehmens oder einzelnen Bereichen dienen, damit Entscheidungsträgern frühzeitig Entwicklungen und Schwachstellen aufgezeigt werden (Bichler et al. 2011). Insbesondere können Kennzahlen dazu dienen, Ziele zu quantifizieren und den Erfüllungsgrad gesetzter Ziele zu messen (Groll 1991; Schuh 2013). Außerdem können Kennzahlen von Simulationswerkzeugen benutzt werden, um Ergebnisse durch Simulation zu erhalten, zum anderen ist es aber auch möglich, aus den Ergebnissen von Simulationsläufen Kennzahlen zur Bewertung des zu untersuchten Systems zu ermitteln (Gutenschwager et al 2017).

Jede Branche bedient sich dabei unterschiedlicher, an die Charakteristiken angepasste, Kennzahlen (Preißler 2010), wie auch die Logistikbranche, in der Kennzahlen zum Management und Controlling von Logistiksystemen verwendet werden (ten Hompel und Heidenblut 2011). Kennzahlen stellen ein unverzichtbares Werkzeug für jede Unternehmensführung dar, um aus der Fülle der Informationenen das Wichtigste herauszufiltern und basierend darauf fundierte unternehmerische Entscheidungen zu treffen (Preißler 2010).

Bürkeler (1977, S. 6) definiert den Begriff Kennzahlen als "betrieblich relevante, numerische Informationen". Anhand dieser Definition kann eine Einordnung von Kennzahlen durch die Wissenstreppe von North (vgl. Abschnitt 3.3) erfolgen. Zum einen bestehen Kennzahlen laut Bürkeler (1977) nur aus numerischen Zeichen und zum anderen sind Kennzahlen nicht nur Daten, sondern Informationen, die mit einer Bedeutung versehen sind. Diese Bedeutung ist wichtig, um überhaupt Aussagen über das zu untersuchende System treffen zu können, denn "Daten allein sprechen nicht – sie müssen übersetzt werden" (Ossola-Haring 2006, S. 7).

Basieren können Kennzahlen auf einer mathematischen beschreibbaren Eigenschaft, einer vergleichenden Bewertung oder der Aggregierung von Messwerten (ten Hompel und Heidenblut 2011). Insbesondere das Aggregieren von Messwerten bzw. Daten (vgl. Abschnitt 3.9.1) wird in dieser Arbeit verwendet, um Kennzahlen zu berechnen.

Die Definition von Kennzahlen umfasst bei allen Autoren zwar immer Verhältniszahlen (Siegwart et al. 2010; Schott 1991), jedoch seltener Absolutzahlen (Bürgi 1985), die oft auch Grundzahlen genannt werden (Ehrmann 2012). Absolutzahlen sind Zahlen, die unabhängig von anderen Zahlen dargestellt werden, wie z.B. der Umsatz oder eine Bilanzsumme. Verhältniszahlen dagegen setzten zwei Zahlen in Beziehung, wie z.B. der Beschäftigungs- oder Nutzungsgrad (Siegwart et al. 2010). Wie Groll (1991) feststellt,

werden in der Praxis absolute Zahlen, wie z.B. der Umsatz oder eine Lagerkapazität, unbestritten zur Informationsgewinnung über Systeme verwendet. Aus diesem Grund werden im Rahmen dieser Arbeit auch absolute Zahlen, also Kennzahlen, aufgefasst.

Verhältniszahlen können in drei Kategorien eingeteilt werden. Die nach Siegwart et. al (2010) wichtigste Kategorie ist die der Beziehungszahlen, in der inhaltlich ungleichartige Daten zueinander ins Verhältnis gesetzt werden. Ein Beispiel ist der Umsatz je beschäftigte Person (Wolf 1977). Allgemein gilt, dass der Zähler die Beobachtungszahl und der Nenner die Bezugszahl ist (Siegwart et. al 2010). Die zweite Kategorie setzt inhaltlich gleichartige Daten zueinander in Verhältnis (Siegwart et. al 2010). Kennzahlen aus dieser Kategorie werden Gliederungszahlen genannt (Siegwart et. al 2010). Relative Häufigkeiten sind immer Gliederungszahlen (Siegwart et. al 2010). Ein Beispiel ist der Anteil einer Teilmenge an einer Gesamtmenge (Siegwart et. al 2010). Die dritte Kategorie ist die der Messzahlen. Mit ihnen werden relative Veränderungen, also Entwicklungen, ausgedrückt (Siegwart et. al 2010). Dazu wird eine Basiszahl ausgewählt, beispielsweise der Umsatz des Jahres 2017, auf die eine andere Messung, wie der Umsatz des Jahres 2018, bezogen wird (Siegwart et. al 2010).

Eine weitere Unterteilung von Kennzahlen kann in monetäre und nichtmonetäre Kennzahlen vorgenommen werden (Kaplan und Norton 1996). Monetär bezieht sich in diesem Fall auf klassische betriebswirtschaftliche Kennzahlen der Finanzsphäre (Stenner 2004). Nach Weber (2006) ist mit ausschließlich monetären Kennzahlen nur eine unzureichende Beurteilung von komplexen Unternehmenssituationen möglich. Daher sollten, um in der Praxis nicht zu versagen, auch nichtmonetäre Kennzahlen hinzugezogen werden (Preißler 2010). Ohnehin sind sie oftmals Frühindikatoren für finanzielle Ergebnisse (Stenner 2004).

In der Logistik können Kennzahlen nach dem System von Schulte (2016) kategorisiert werden. Er unterscheidet zwischen:

- Struktur- und Rahmenkennzahlen, die sich auf Aufgabenumfang, Kapazität oder Kosten beziehen. Beispiele: Transportvolumen pro Periode, Anzahl von Fördermitteln, oder Gesamtkosten der Lagerhaltung
- Produktivitätskennzahlen, die sich auf die Produktivität von Mitarbeitern und technischen Betriebseinrichtungen in Form von Mengengrößen, Zeitgrößen oder Auslastungsgrößen beziehen. Beispiele: Sendungen pro Personalstunde, Warenannahmezeit pro Sendung, Auslastungsgrad von Transportmitteln
- Wirtschaftlichkeitskennzahlen, die sich auf die Logistikkosten von Logistikleistungen pro Leistungseinheit oder Verhältnisse von Kosten, bzw. Erlösgrößen, beziehen. Beispiele: Distributionskosten je Auftrag, Anteil Auftragsabwicklungskosten am Umsatz

- Qualitätskennzahlen, die sich auf Anteilswerte oder Zeitgrößen, die den Grad der Zielerreichung oder der Qualität widerspiegeln, beziehen. Beispiele: Anteil verspäteter Lieferungen an Gesamtzahl von Lieferungen, durchschnittliche Lieferzeit

Logistische Kennzahlen können außerdem, analog zur Unternehmensstruktur, in drei hierarchische Ebenen eingeteilt werden. Die sich somit ergebene Kennzahlenpyramide gliedert sich in folgende Bereiche (Bichler et al. 2011; Bichler und Schröter 2004):

- Strategisch: Kennzahlen für die Unternehmensführung. Umfasst Kennzahlen zur Erreichung betrieblicher Oberziele
- Dispositiv: Kennzahlen für die Bereichsleitung
- Operativ: Kennzahlen für Abteilungsleiter und Gruppen

Durch diese Zuordnung soll sichergestellt werden, dass die jeweiligen Entscheidungsträger nur die Kennzahlen bekommen, die für sie relevant sind. Somit sollen überflüssige Informationen gefiltert und eine Fokussierung betrieben werden (Bichler und Schröter 2004).

Eine einzelne Kennzahl bildet nur einen Sachverhalt eines größeren Systems ab und hat somit nur eine begrenzte Aussagekraft (Wolf 1977). Um ein System ganzheitlich zu erfassen ist es zwangsläufig nötig, mehrere Kennzahlen zu bilden (Oeller 1979). Daher werden verschiedene Kennzahlen zu einem sogenannten Kennzahlsystem zusammengefasst (ten Hompel 2011). Diese Systeme sollten nur die wichtigsten und nötigen Kennzahlen beinhalten, um einer Informationsüberflutung vorzubeugen (Preißler 2010). In Kennzahlensystemen werden Kennzahlen in eine Beziehung gebracht und hierarchisch angeordnet. Kennzahlen mit höherer Aussagekraft sind solchen mit niedrigerer übergeordnet. Die Kennzahlen an der Spitze der Hierarchie werden auch Spitzenkennzahlen oder Key Performance Indicators (Schröder 2017) genannt. In der Logistik sind das Auftrags- und Materialdurchlaufzeit, Lieferservicegrad, Bestand, Umschlagshäufigkeit und Logistikkosten (Frank 2008).

Kennzahlensysteme können in drei Arten unterteilt werden: Rechensystem, Ordnungssystem und Zielsystem (Groll et al. 2003).

In einem Rechensystem sind Kennzahlen durch mathematische Transformationsregeln verbunden (Groll et al. 2003). Ausgehend von einer Spitzenkennzahl wird durch mathematische Operationen eine sukzessive Zerlegung dieser Spitzenkennzahl in Unterkennzahlen vollzogen (Groll et al. 2003). Dieses System eignet sich besonders zur Untersuchung des Einflusses einer bestimmten Untersuchungskennzahl auf das Gesamtsystem (Groll et al. 2003). Für komplexe Systeme ist dieser Ansatz durch seinen niedrigen Informationsgehalt nicht geeignet. Bekannte Beispiele sind das DuPont System (Schott 1991) oder das ZVEI-System (Ehrmann 2012).

In einem Ordnungssystem bestehen zwischen den Kennzahlen nur nicht quantifizierbare Beziehungen, es werden also Zusammenhänge nicht mathematisch, sondern sachlogisch dargestellt (Reichmann und Lachnit 1976). Mit dieser Vorgehensweise können Systeme zwar systematisch und vollständig erfasst werden, jedoch ist dazu eine große Anzahl von Kennzahlen, verbunden mit hohem Datenbeschaffungs und -aufbereitungsaufwand, nötig (Groll et al. 2003). Ein Beispiel für ein Ordnungssystem ist das Rentabilitäts-Liquiditäts-Kennzahlensystem von Reichmann und Lachnit (1976).

Bei einem Zielsystem werden zunächst zu erreichende Ziele definiert, aus denen entsprechende Kennzahlen abgeleitet werden (Weber et al. 1995). Es gibt verschiedene hierarchische Ebenen, in denen übergeordnete Ziele von untergeordneten Zielen unterschieden werden (Syska 1990).

Eine Vorgehensweise zum Aufbau eines Kennzahlensystems in der Logistik kommt von Lochthowe (1990). Anhand der Zielstruktur eines Unternehmens werden Kennzahlen abgeleitet und nach der Kategorisierung der Kennzahlenpyramide (vgl. Abschnit 3.9.2) in strategische, taktische und operative Kennzahlen unterschieden. Die Relevanz der Kennzahlen ergibt sich somit durch ihre Stellung innerhalb der Unternehmensziele. Eine ähnliche Herangehensweise wird von Syska (1990) gewählt. Syska (1990) leitet zunächst ein allgemeines Zielsystem der Logistik aus den allgemeinen Zielen eines Unternehmens ab. Aus dem Zielsystem der Logistik werden weitere Ziele der Lagerhaltung identifiziert. In einem weiteren Schritt werden zu den identifizierten Zielen Kennzahlen zu deren Beschreibung gefunden. Einen ähnlichen Ansatz zu Lochthowe (1990) und Skyska (1990) wählen auch Pfohl und Zöllner (1991). Solche Ansätze, die Kennzahlen aus übergeordneten Unternehmenszielen ableiten, werden auch Top-Down-Ansätze genannt (Weber et al. 1995).

3.9.2 Kennzahlen zur Quantifizierung von Zielen der Lagerhaltung

Im vorherigen Abschnitt wurde der Top-Down-Ansatz zur Bestimmung von Zielen der Lagerhaltung vorgestellt. Ziele werden als Absichtserklärung eines zukünftigen Zustands gesehen (Ehrmann 2012). Um den aktuellen Zustand der Zielerfüllung zu quantifizieren werden Kennzahlen verwendet. Im Folgenden werden Ziele von Unternehmen, der Logistik und der Lagerhaltung vorgestellt, die durch Kennzahlen quantifiziert werden können (Schuh).

Als Hauptziele von Unternehmen identifizieren Pfohl (2016) und Heiserich et al. (2011) die folgenden drei:

- Rentabilitätsziele
- Marktstellungsziele
- Soziale Ziele

Diesen Zielen sind alle anderen Ziele eines Unternehmens untergeordnet. Diese drei Ziele lassen sich jedoch noch weiter zu dem obersten Ziel, der Überlebenssicherung des Unternehmens, verdichten (Pfohl 2016).

Die Ziele der Logistik können durch die "7R der Logistik" beschrieben werden, welche beschreiben, welche Dinge der Logistik "richtig" laufen müssen (Heiserich et al. 2011). In der IT-gestützten Logistik, werden sogar die "8R der Logistik" gefordert:

- Das richtige Produkt
- In der richtigen Menge
- In der richtigen Qualität
- Am richtigen Ort
- Zur richtigen Zeit
- Zu den richtigen Kosten
- Für den richtigen Kunden
- Mit den richtigen Informationen

(Hausladen 2016)

Ziele für die Lagerhaltung können beispielsweise in Vahrenkamp und Kotzab (2012) gefunden werden, welche die Ziele der Lagerhaltung, mit Fokus auf betriebswissenschaftlichen Zielen, angeben:

- Sicherung einer hohen Lieferbereitschaft gegenüber dem Kunden
- Minimierung der Fälle, in denen das Lager keine Vorräte besitzt
- Sicherheit und Schutz des Lagergutes
- Minimierung der Lagerkosten
- Hohe Flexibilität

Brandes (1997) definiert Ziele der Lagerhaltung hinsichtlich des operativen Betriebs:

- Termingerechte Auslagerung
- Kein Überlauf der Pufferkapazitäten auf der Einlagerungsseite
- Optimale Lagerraumausnutzung
- Minimierung des Energieverbrauchs
- Hohe Verfügbarkeit
- Vermeidung ungünstiger statischer Belastungen der Regalanlage
- Durchsatzmaximierung

Einige wichtige Kennzahlen die zur Quantifizierung von Zielen, wie den in diesem Abschnitt genannten, werden im nächsten Abschnitt vorgestellt.

3.9.3 Kennzahlen von Lägern

In diesem Absatz werden einige gängige Kennzahlen der Lagerhaltung vorgestellt. Ein Problem für Kennzahlen in Produktion und Logistik, und somit auch Lägern als Teil der Produktion und Logistik, erwächst durch die in der Logistik erbrachten Dienstleistungen, die schwerer messbar sind als Sachleistungen und die Leistungsdefinition erschweren (Weber et al. 1995). Auch bei der Ermittlung des wirtschaftlichen Erfolgs eines Lagers und dessen Darstellung mit monetären Kennzahlen ist es problematisch, dass Läger keine Marktleistung erbringen und der Erfolg somit nur indirekt gemessen werden kann (Weber et al. 1995).

Im Kontext von Lägern werden Kennzahlen zur Berechnung, Beurteilung, Vergleichung, Planung und laufenden Kontrolle eingesetzt (Martin 2017). Im Folgenden sind einige Kennzahlen zur Beschreibung von Lägern aufgeführt.

Im Verlauf des Abschnitts liegt der Fokus nicht auf Rahmenkennzahlen, wie beispielsweise der Lagerkapazität. Zwar sind auch Rahmenkennzahlen wichtige und gängige Kennzahlen, doch stellen sie keine Daten dar, die während eines Experiments erfasst werden sollten, da sie keine Systemlastdaten darstellen (vgl. Abschnitt 3.4).

Die Lagerleistung beschreibt die Leistung, also die umgeschlagene Menge des Lagers pro Zeiteinheit, des Lagers (Bauer 1985). Sie ist durch die Anzahl von Ein- und Auslagerungen pro Zeiteinheit definiert (Jünemann 1970). Über die Zeit gesehen ist die Lagerleistung nicht konstant, da sie verschiedenen Schwankungen unterliegt (Bauer 1985). Laut der VDI-Richtlinie 4480 (1998) ist die Lagerleistung neben der Lagerkapazität die wichtigste Kennzahl von Lägern.

$$Lagerleistung = \frac{Menge\ umgeschlagener\ Ladeeinheiten}{Zeit}$$

(Jünemann 1970, S. 111)

Eine weitere wichtige Kennzahl zur Beschreibung von Lägern ist der durchschnittliche Lagerbestand, der den Mittelwert zwischen Anfangs- und Endwert darstellt (Preißler 2010).

$$Durchschnittlicher\ Lagerbestand = \frac{Anfangsbestand + Enbestand}{2}$$

(Preißler 2010, S. 180-181)

Der Füllungsgrad beschreibt den Quotienten zwischen den absoluten Zahlen durchschnittlicher Lagerbestand und Lagerkapazität (Bauer 1985).

$$Füllungsgrad = \frac{Durchschnittlicher\ Lagerbestand}{Lagerkapazität}$$

(Bauer 1985, S. 15-20)

Die Lagerbestandsstruktur ist eine Kennzahl zur Beschreibung der Zusammensetzung des Gesamtlagerbestandes nach bestimmten Kriterien (Ossola-Haring 2006). So kann sie z.B. den Anteil von Rohstoffen oder Halbzeugen gemessen am Gesamtlagerbestand widerspiegeln.

$$Lagerbestandsstruktur = \frac{\text{Lagerbestand nach Art (oder Alter, Qualität, Mode ...)}}{\text{(Gesamt−)Lagerbestand}}$$

(Ossola-Haring 2006, S. 436)

Der Servicegrad der Lagerhaltung ist eine Kennzahl zur Beschreibung der Lieferbereitschaft von Artikeln (ten Hompel und Heidenblut 2011). Lieferbereitschaft liegt dann vor, wenn ein Artikel zum Zeitpunkt der Nachfrage sofort verfügbar ist (Ehrmann 2012).

$$Servicegrad\ der\ Lagerhaltung = \frac{Anzahl\ bedienbarer\ Auslagerungsaufträge}{Anzahl\ Auslagerungsaufträge}$$

(ten Hompel und Heidenblut 2011. S. 281)

Die Bestandskosten des Lagers geben an, wie hoch die entstehenden Kosten für das Unternehmen durch das Halten und Finanzieren von Beständen sind (Ehrmann 2012). Vereinfachend kann ein Bestandskostensatz ermittelt werden, der von mehreren Faktoren abhängig ist, anstatt mit vielen einzelnen Termen rechnen zu müssen (Heiserich 2011).

$$Bestandskosten = Lagerbestand * Bestandskostensatz$$

(Ehrmann 2012, S. 569)

Eine ähnliche Berechnung wird für die Kosten durch Fehlmengen durchgeführt. Sie ergeben sich aus dem Produkt der Fehlmenge und einem Fehlmengenkostensatz (Bichler und Schröter 2012).

$$Kosten\ durch\ Fehlbestände = Fehlmengen * Fehlmengenkostensatz$$

(Bichler und Schröter 2004, S. 187)

3.9.4 Kennzahlen von automatischen Hochregallägern

Nachdem allgemeine Kennzahlen für Läger im Abschnitt 3.7.2 betrachtet wurden, werden in diesem Abschnitt Kennzahlen von automatischen Hochregallägern betrachtet. Zwar gelten alle Kennzahlen der Läger auch für automatische Hochregalläger, andersherum ist dies nicht der Fall.

Da die Spielzeiten (vgl. Abschnitt 2.4) in Lägern, wie automatischen Hochregallägern, in denen verschiedene Positionen bzw. Lagerfächer angefahren werden, nicht für alle

Lagerfächer dieselben sind, werden die mittleren Spielzeiten als Kennzahlen eingeführt (ten Hompel et al. 2018). Nach Koether (2007) stellt die Spielzeit die wesentliche Kennzahl eines Bediengeräts und dessen Umschlagsleistung dar und werden in das mittlere Einzelspiel und das mittlere Doppelspiel unterteilt.

$$Mittlere\ Einzelspielzeit = \frac{Dauer\ aller\ Einzelspiele}{Anzahl\ von\ Einzelspielen}$$

(Koether, 2007, S. 88)

$$Mittlere\ Doppelspielzeit = \frac{Dauer\ aller\ Doppelspiele}{Anzahl\ von\ Doppelspielen}$$

(Koether, 2007, S. 88)

Eine weitere charakteristische Kennzahl automatischer Hochregalläger ist der Raumnutzungsgrad, welcher das Volumen der Lagereinheit und das Volumen des Lagerraums in Beziehung setzt (Bichler und Schröter 2012).

$$Raumnutzungsgrad = \frac{Volumina\ der\ Lagereinheiten}{Volumen\ des\ Lagerraums}$$

(Bichler und Schröter 2004, S. 188)

Im nächsten Abschnitt folgt eine Beschreibung eines physischen Modells, an dem die in diesem Abschnitt dargestellten Kennzahlen exemplarisch bestimmt werden können.

3.10 Physisches Modell eines automatischen Hochregallagers

Physische Modelle sind "in der Regel zu zeit- und kostenaufwendig" (Brandes 1997, S. 57), um sie für Experimente zu benutzen. Law (2015) sieht die Anwendung von physischen Modellen eher im Bereich von Lernmodellen. Jedoch wurde bei den genannten Autoren und in weitgehender Literaturrecherche keine Forschungsarbeiten gefunden, die das Thema der Datengewinnung an einem physischen Modell zum Aufbau einer Simulationsdatenbasis untersuchen.

Am Fachgebiet IT in Produktion und Logistik (ITPL 2020) der Fakultät Maschinenbau der Technischen Universität Dortmund existiert ein physisches Modell eines automatischen Hochregallagers, welches für Forschungszwecke dieser Arbeit zur Verfügung steht. Ein Foto des physischen Modells ist in Abbildung 3.10 gezeigt.

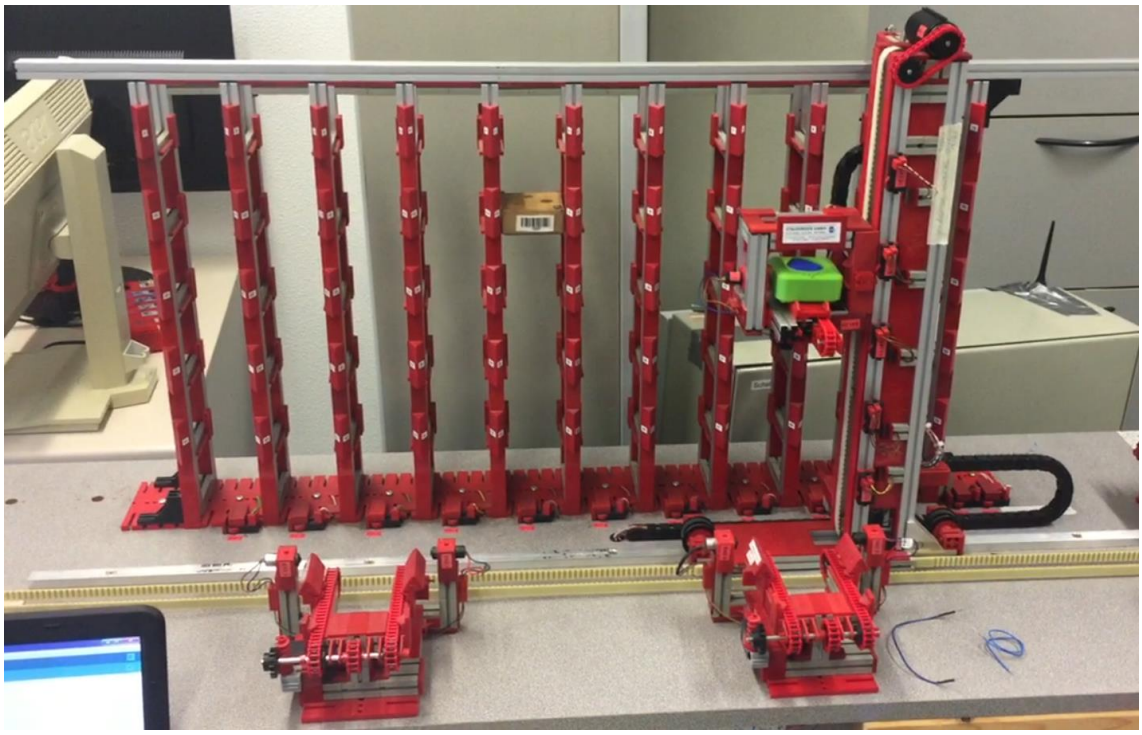

Abbildung 3.10: Physisches Modell eines automatischen Hochregallagers am Fachgebiet IT in Produktion und Logistik

Das Modell wurde durch das Unternehmen Staudinger GmbH (Staudinger GmbH 2020) gebaut und weist eine Lagerkapazität von 50 Regalfächern auf, die sich aus den zehn Spalten und fünf Reihen des Hochregallagers errechnen lässt. Die Regalfächer weisen keinen durchgehenden Boden auf. Die Lagereinheiten werden durch zwei kleine Absätze an den Regalfachwänden in den Regalfächern gehalten. Das Modell wird durch ein automatisches Regalbediengerät bedient und kann mit zwei Transportbändern verbunden werden. Für Informationen über die Funktions- und Bauweise realer automatischer Hochregalläger sei auf Kapitel 2 verwiesen.

Das Modell weist fünf Motoren auf, die für die Bewegungen zuständig sind. Ein Motor ermöglicht die Längsbewegung des Regalbediengeräts, ein weiterer Motor eine vertikale Bewegung und ein dritter Motor kann das Lastaufnahmemittel orthogonal zu den Bewegungsrichtungen des Regalbediengeräts bewegen. Zwei weitere Motoren werden zum Antrieb der beiden Transportbänder benutzt.

Zur Positionsbestimmung des Regalbediengeräts werden Taster eingesetzt. An jeder Spalte ist ein Taster verbaut, der durch das Fahrwerk ausgelöst wird und somit die aktuelle Spalte bestimmt werden kann. Zur Bestimmung der aktuellen Zeile wurden Taster am Mast des Regalbediengeräts verbaut, welche durch den Hubschlitten ausgelöst werden. Jede Regalreihe hat dabei zwei Taster verbaut, die jeweils einer Unter- und einer Oberstellung entsprechen. Auch für das Lastaufnahmegerät wurden drei Taster verbaut, die drei verschiedenen Stellungen, der Vorder-, Mittel- und Hinterstellung, entsprechen.

Eine Steuerungseinheit oder eine Stromversorgung für das Modell standen nicht zur Verfügung. Gegenüber dem Originalzustand, der durch das Unternehmen Staudinger GmbH hergestellt wurde, wurden nur einige Motortreiber im Zuge früherer Arbeiten an dem Modell hinzugefügt.

4 Aufbau einer Simulationsdatenbasis für automatische Hochregalläger

Da die Simulationsdatenbasis ein automatisches Hochregallager beschreiben soll, sind zunächst Kenntnisse über den Aufbau und die Funktionsweise von automatischen Hochregallägern, wie sie in Kapitel 2 beschrieben sind, eine nötige Voraussetzung, um passende Daten zu dessen Beschreibung finden zu können. In Kapitel 3 wurden die theoretischen Grundlagen dargelegt, um Daten zum Aufbau einer Simulationsdatenbasis abzuleiten, und an einem physischen Modell eines automatischen Hochregallagers zu erfassen. In diesem Kapitel werden diese Grundlagen dazu benutzt, den Aufbau einer Simulationsdatenbasis für automatische Hochregalläger zu beschreiben. Dies umfasst die Ableitung von Daten, die zur Bildung einer Simulationsdatenbasis benutzt werden können, sowie die Auswahl von Peripheriegeräten, die diese Daten während eines Experiments mit einem physischen Modell erfassen können.

Das erste, zu lösende Problem stellt dabei die Identifizierung von Daten für die Simulationsdatenbasis dar. Das zweite Problem besteht darin, wie die Daten für die Simulationsdatenbasis während des Experiments am physischen Modell erfasst werden können. Außerdem ist ein drittes Problem zu lösen, welches darin besteht, dass die erfassten, einzelnen Daten zu einer Simulationsdatenbasis zusammengefügt werden müssen und in einer geeigneten Form gespeichert werden, damit sie für eine Simulation verwendet werden können.

Die Herangehensweise, die in diesem Kapitel zum Aufbau einer Simulationsdatenbasis für automatische Hochregalläger verwendet wird und die drei genannten Probleme löst, wird im nächsten Abschnitt vorgestellt.

4.1 Vorstellung der Herangehensweise

Um die drei im vorherigen Abschnitt beschriebenen Probleme, nämlich welche Daten erfasst werden müssen, wie sie erfasst werden und wie sie für eine Simulation zugänglich gemacht werden, zu lösen, wird wie folgt vorgegangen.

Zur Lösung des ersten Problems empfiehlt es sich, wie in Abschnitt 3.2 beschrieben, zielgerichtet zu arbeiten und die Daten für die Simulationsdatenbasis anhand einer Aufgabenspezifikation und eines Konzeptmodells auszuwählen. Da im Rahmen dieser Arbeit keine konkrete Aufgabenspezifikation und kein konkretes Konzeptmodell vorgegeben ist, können die Daten nicht aus ihnen abgeleitet werden.

Durch das Fehlen einer konkreten Aufgabenspezifikation und eines Konzeptmodells werden Daten für eine allgemeine Simulationsdatenbasis gesucht. Eine allgemeine Simulationsdatenbasis zeichnet sich dadurch aus, dass sie es der Simulation ermöglicht,

möglichst viele gängige Ergebnisse ermitteln zu können. Die allgemeine Simulationsdatenbasis und ihre Daten lassen sich somit aus möglichst vielen gängigen Ergebnissen ableiten. In Abschnitt 3.9 wurde gezeigt, dass Ergebnisse einer Simulationsstudie, und somit einer Simulation und ihrer Simulationsdatenbasis, Kennzahlen sein können. Daher macht eine allgemeine Simulationsdatenbasis eine möglichst hohe, umfassende Zahl von in der Praxis relevanten Kennzahlen berechenbar. Zwischen Daten und Kennzahlen besteht ein Zusammenhang, der in Abschnitt 4.4 beschrieben wird. Es wird gezeigt, dass Daten aus Kennzahlen ableitbar sind und somit das Problem der Identifizierung von Daten, zur Beschreibung von automatischen Hochregallägern gleichermaßen durch die Identifizierung von Kennzahlen automatischer Hochregalläger und deren Ableitung zu Daten, lösbar ist. Dieser zusätzliche Schritt, zunächst Kennzahlen und daran anschließend erst Daten abzuleiten, bietet sich deswegen an, da Kennzahlen die Ergebnisse einer Simulationsstudie, für deren Zweck die Simulationsdatenbasis aufgebaut wird, sein können (vgl. Abschnitt 3.9), und somit eine zielgerichtete Auswahl von Daten erfolgt.

Die Herangehensweise zur Lösung des Problems orientiert sich an den Herangehensweisen aus der Literatur zur Identifizierung von Kennzahlen, die in Abschnitt 3.9.1 vorgestellt wurden. Die vorgestellten Herangehensweisen leiten Kennzahlen zum Aufbau von Kennzahlsystemen ab, jedoch können, auch wenn das Ziel dieser Arbeit nicht der Aufbau eines Kennzahlensystems ist, die vorgestellten Herangehensweisen zur Ableitung von Kennzahlen benutzt werden. Es wurden zwei allgemeine Methoden, basierend auf der Untersuchungsrichtung, identifiziert. Die eine Herangehensweise ist der Bottom-up Ansatz, der von Daten auf Kennzahlen und sogar übergeordnete Ziele schließt. Da dieser Ansatz auf Daten basiert und nicht wie gefordert Daten identifiziert, ist der Bottom-up Ansatz für das vorliegende Problem ungeeignet. Das umgekehrte Prinzip des Top-down Ansatzes hingegen, also das Ableiten von Kennzahlen aus übergeordneten Zielen, erfüllt die Zielsetzung und prägt die Untersuchungsrichtung dieses Kapitels.

In Anlehnung an die Top-down Vorgehensweisen von Lochthowe, Syksa und Weber (vgl. Abschnitt 3.9.1) werden zunächst in Abschnitt 4.2 Ziele des zu untersuchenden Systems, in diesem Fall von automatischen Hochregallägern, identifiziert. Hierbei wird hierarchisch vorgegangen, beginnend mit den übergeordneten Zielen des Unternehmens und gefolgt von den Zielen der Logistik und den Zielen der Lagerhaltung. Nachdem durch die identifizierten Ziele das Zielsystem (vgl. Abschnitt 3.9.1) festgelegt wurde, werden in Abschnitt 4.3 Kennzahlen zur Beschreibung der einzelnen Ziele ausgewählt (vgl. Abschnitt 3.9.2). Anschließend werden in Abschnitt 4.4 die Daten, die zur Berechnung der ausgewählten Kennzahlen benötigt werden, aus den Kennzahlen zur Beschreibung der Ziele abgeleitet.

Nachdem die Daten für eine allgemeine Simulationsdatenbasis zur Simulation automatischer Hochregalläger identifiziert wurden, wird das zweite Problem gelöst. Die Daten sollen im Laufe eines Experiments mit einem physischen Modell eines automatischen Hochregallagers erfasst werden. Zur Datenerfassung werden Peripheriegeräte und Sensoren verwendet (vgl. Abschnitt 3.6 und Abschnitt 3.7). Bevor die Peripheriegeräte zur Datenerfassung ausgewählt werden, werden in Abschnitt 4.5 Anforderungen, die sich durch die Problemdomäne an die Peripheriegeräte ergeben, erläutert. Danach werden in Abschnitt 4.6 allen identifizierten Daten für die allgemeine Simulationsdatenbasis Peripheriegeräte oder Sensoren zugeordnet, die für deren Erfassung während eines Experiments geeignet sind.

Das Ergebnis der Lösung des zweiten Problems sind einzelne Daten, die durch Peripheriegeräte erfasst werden. Um eine Simulationsdatenbasis zu bilden, müssen sie in einer geeigneten Struktur zusammengeführt und gespeichert werden (vgl. Abschnitt 3.8). Dies stellt das dritte Problem dar und wird am Ende dieses Kapitels in den Abschnitten 4.7 und 4.8 behandelt.

4.2 Auswahl von Zielen für ein Zielsystem eines automatischen Hochregallagers

Wie in Abschnitt 4.1 beschrieben, werden in diesem Abschnitt zunächst allgemeine Ziele einer Simulationsstudie über automatische Hochregalläger aus der Literatur abgeleitet.

Ein Verbund von Einzelzielen zur Beschreibung der Ziele eines Betrachtungsgegenstandes, wie z.B. eines automatischen Hochregallagers, wird Zielsystem genannt (vgl. Abschnitt 3.9.1). Abschnitt 3.9.1 zeigt auf, dass innerhalb eines Zielsystems eine Zielhierarchie existiert, also einige Ziele anderen übergeordnet sind. Die Lagerhaltung und die damit eingeschlossenen Läger als Teilsystem der Logistik, sind mitunter für die Erreichung der allgemeinen Logistikziele und Unternehmensziele verantwortlich. Im Falle eines Zielsystems für ein automatisches Hochregallager stellen Unternehmensziele die höchste Ebene dar. Die nächsthöhere Ebene wird durch die Logistikziele gebildet. Die Ziele der Lagerhaltung und somit auch der automatischen Hochregalläger repräsentieren die niedrigste Hierarchiestufe. Die Hierarchie eines Zielsystems für automatische Hochregalläger ist in Abbildung 4.1 zu sehen.

Abbildung 4.1: Hierarchie eines Zielsystems für automatische Hochregalläger

Im Folgenden werden exemplarisch einige der wichtigsten Ziele der jeweiligen Hierarchieebenen behandelt. Begonnen wird mit Zielen der höchsten Hierarchiestufe, also den Unternehmenszielen. Jedes weitere Ziel im Zielsystem lässt sich diesen Hauptzielen unterordnen. In Abschnitt 3.9.2 werden die Hauptziele von Unternehmen wie folgt angegeben:

- Rentabilitätsziele
- Marktstellungsziele
- Soziale Ziele

Aus diesen Hauptzielen von Unternehmen lassen sich viele Ziele von Unternehmen ableiten, um eine detaillierte Übersicht der Unternehmensziele zu gewinnen (vgl. Abschnitt 3.9.2). Für das Problem dieser Arbeit ist diese Ableitung jedoch nicht zielführend und wird daher nicht durchgeführt. Es reicht das Wissen, dass durch die Untersuchung der untergeordneten Logistikziele und Lagerhaltungsziele ein Beitrag zur Führung eines Unternehmens über Unternehmensziele geleistet wird und die Ziele der Logistik und der Lagerhaltung diesen untergeordnet sind. In dem aufzustellenden Zielsystem bilden die genannten Ziele die höchste hierarchische Ebene.

Die Ziele der zweithöchsten Ebene sind die Ziele der Logistik, die aus den Unternehmenszielen ableitbar sind. Ihre Hauptziele können, wie in Abschnitt 3.9.2 beschrieben, durch die "8R" der Logistik ausgedrückt werden:

- Das richtige Produkt
- In der richtigen Menge
- In der richtigen Qualität
- Am richtigen Ort
- Zur richtigen Zeit
- Zu den richtigen Kosten
- Für den richtigen Kunden
- Mit den richtigen Informationen

Es wurde bewusst eine Entscheidung gegen eine ausschließliche Zielbestimmung durch die „klassischen 7R" der Logistik getroffen. Durch Anwendung der „modernen" 8R der Logistik wird den Technologien, die im Rahmen der IT-Gestützten Logistik die „richtigen Informationen" benutzen, ein größeres Maß an Bedeutung zugewiesen.

Zuletzt werden Ziele der Lagerhaltung und somit automatischer Hochregalläger identifiziert. In Abschnitt 3.9.2 wurden zwei Zielsysteme der Lagerhaltung vorgestellt. Die aufgeführten Ziele, in dem Zielsystem von Vahrenkamp und Kotzab (2012) und dem Zielsystem von Brandes (1997), weisen keine Übereinstimmung auf. Dies lässt den Schluss zu, dass die Ableitung von Zielen der Lagerhaltung aus den Zielen der

Unternehmen und Logistik einen Interpretationsspielraum bietet. Vahrenkamp und Kotzab (2012) fokussieren sich auf die betriebswirtschaftlichen Funktionen der Lagerhaltung. Brandes‘ (1997) Fokus liegt auf Zielen für den Lagerbetrieb.

Die Ziele, die Vahrenkamp und Kotzab (2012) definieren, lauten wie folgt:

- Sicherung einer hohen Lieferbereitschaft gegenüber dem Kunden
- Minimierung der Fälle, in denen das Lager keine Vorräte besitzt
- Sicherheit und Schutz des Lagergutes
- Minimierung der Lagerkosten
- Hohe Flexibilität

Die Ziele nach Brandes (1997) sind:

- Termingerechte Auslagerung
- Kein Überlauf der Pufferkapazitäten auf der Einlagerungsseite
- Optimale Lagerraumnutzung
- Minimierung des Energieverbrauchs
- Hohe Verfügbarkeit
- Vermeidung ungünstiger statischer Belastungen der Regalanlage
- Durchsatzmaximierung

Nach Meinung des Autors, können dem bisher durch die Literatur abgeleiteten Zielsystems des automatischen Hochregallagers weitere Ziele hinzugefügt werden. Zum einen ergibt sich ein weiteres Ziel aus dem Zielkonflikt der Lagerhaltung (vgl. Abschnitt 2.1). Aus den gegensätzlichen Zielen, möglichst geringe Bestandskosten zu haben, aber dennoch die Kosten durch Fehlmengen zu senken, lässt sich ein Ziel identifizieren. Dieses Ziel ist das Erreichen eines optimalen Lagerbestands.

Zum anderen ist es von Bedeutung, dass die Lagereinheiten sich identifizieren können und möglichst alle Informationen, die benötigt werden, bereithalten. Dieses Ziel lässt sich aus dem Logistikziel der „richtigen Informationen" ableiten. Das Ziel wurde ausgewählt, um zu zeigen, dass ein reales automatisches Hochregallager einen Beitrag zu Umsetzung der Paradigmen der Industrie 4.0 leisten kann.

Die Basis für die weiteren Schritte in dieser Arbeit ist das Zielsystem, das sich aus den in diesem Abschnitt identifizierten Zielen zusammenfassen lässt und in Abbildung 4.2 dargestellt ist. Das pyramidenförmige Schaubild spiegelt die hierarchische Struktur des Zielsystems wider. An der Spitze stehen die übergeordneten Ziele des Unternehmens. Darunter befinden sich die Ziele der Logistik, die durch die "8R" der Logistik ausgedrückt werden. Auf der niedrigsten Stufe sind alle konkreten Ziele der Lagerhaltung und automatischer Hochregalläger zu finden.

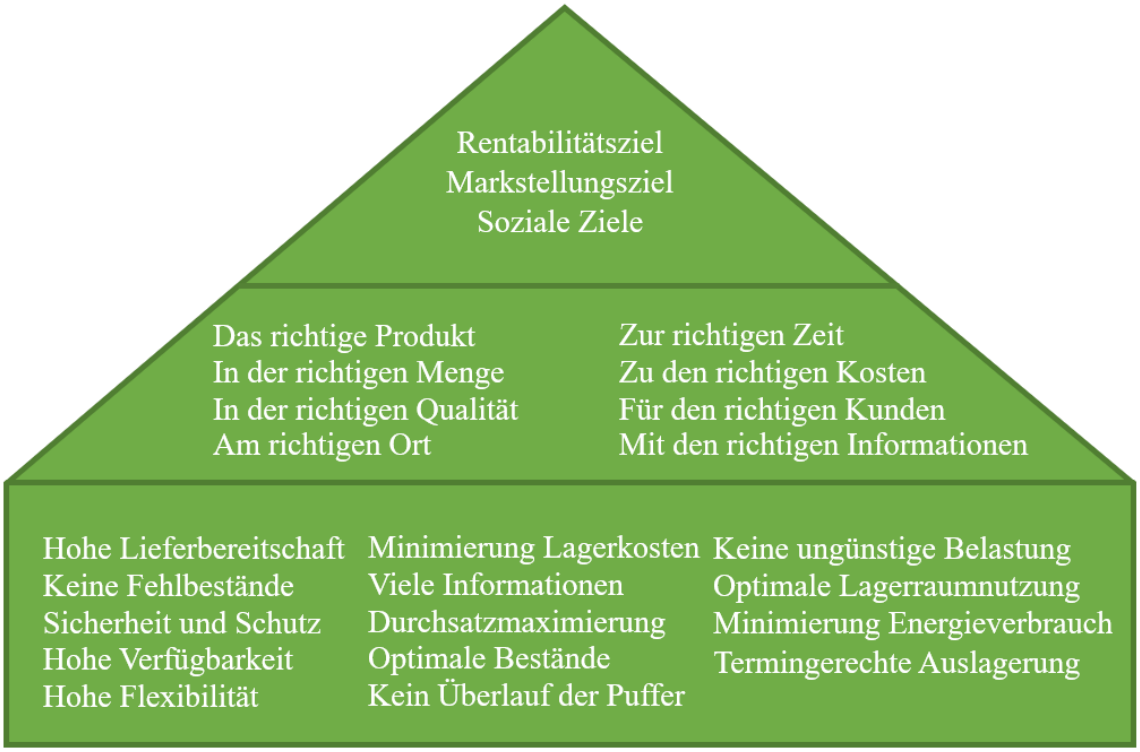

Abbildung 4.2: Zielsystem eines automatischen Hochregallagers

4.3 Ableiten von Kennzahlen aus dem Zielsystem

Im vorherigen Abschnitt wurde das Zielsystem eines automatischen Hochregallagers aufgebaut. Damit Entscheidungsträger bewerten können, in welchem Umfang die jeweiligen Ziele erfüllt werden, müssen sie quantifiziert werden (vgl. Abschnitt 3.9.1). Wie in Abschnitt 3.9.1 beschrieben, werden Kennzahlen zur Quantifizierung von Zielen eingesetzt. In diesem Abschnitt werden Kennzahlen zur Quantifizierung der Ziele des erarbeiteten Zielsystems aus Abschnitt 4.2 abgeleitet.

Bei der Ableitung von Kennzahlen wird sich auf die gängigsten Kennzahlen zur Beschreibung von automatischen Hochregallägern beschränkt. In Abschnitt 3.9.3 bis 3.9.4 wurden bereits gängige Kennzahlen der Logistik, Läger und automatischer Hochregalläger identifiziert. Alle diese Kennzahlen können Kennzahlen sein, die Ziele eines automatischen Hochregallagers quantifizieren, da das automatische Hochregallager als ein Teilsystem der Logistik und Lagerhaltung zur Erfüllung derer Ziele beiträgt.

Da einige Ziele es außerdem erfordern, durch mehrere Kennzahlen quantifiziert zu werden, werden einige Ziele nicht mit der nötigen Vollständigkeit erfasst (vgl. Abschnitt 3.9.2), da diese zusätzlichen Kennzahlen gegebenenfalls nicht gängige Kennzahlen, wie eingehend gefordert, sind.

Das Ableiten von Kennzahlen aus dem Zielsystem wird exemplarisch durchgeführt. Dazu werden Ziele der Lagerhaltung untersucht, die dem Logistikziel der niedrigen Kosten und

somit auch dem Unternehmensziel der Rentabilität untergeordnet sind. Außerdem wird davon abgesehen, Kennzahlen, die Rahmenkennzahlen nach der Definition von Schulte sind (vgl. Abschnitt 3.9.1), abzuleiten, da sie keine Daten sind, die sich über die Zeit des Experiments verändern und ein Erfassung durch Peripheriegeräte benötigen.

Zunächst wird das Ziel der Durchsatzmaximierung betrachtet. Zur Beurteilung dessen Zielerfüllung können mehrere Kennzahlen abgeleitet werden. Die charakteristischste Kennzahl, die der VDI sogar als die wichtigste Kennzahl eines Lagers beschreibt (vgl. Abschnitt 3.9.3), stellt die Lagerleistung dar. Sie gibt den Durchsatz des Lagers bezogen auf einen Zeitraum an, was eine direkte Quantifizierung des Ziels darstellt. Zur differenzierteren Betrachtung des Ziels können weitere Kennzahlen abgeleitet werden, die einen Einfluss auf den Durchsatz haben. Einen großen Einfluss haben dabei die in Abschnitt 3.9.4 vorgestellten Kennzahlen mittlere Einzelspielzeit und mittlere Doppelspielzeit, da sie maßgeblich für die Lagerleistung verantwortlich sind (vgl. Abschnitt 3.9.4).

Zur Beschreibung des Ziels der Minimierung der Lagerkosten, ist eine Vielzahl von Kennzahlen nötig. Diese müssen nicht immer monetärer Natur sein. So ist beispielsweise der Füllungsgrad (vgl. Abschnitt 3.9.3) eine wichtige Kennzahl eines Lagers zur Beurteilung von Einsparungspotentialen der Kosten, denn ein durchweg niedriger Wert des Füllungsgrads impliziert eine zu hoch gewählte Lagerkapazität und somit unnötige Kosten, die eingespart werden könnten. Eine weitere relevante Kennzahl zur Beschreibung dieses Ziels ist der durchschnittliche Lagerbestand. Durch die mit einem Lagerbestand verknüpften Kapitalbindungskosten ist es sinnvoll, möglichst geringe durchschnittliche Lagerbestände aufzuweisen, um auch die Kapitalbindungskosten so niedrig wie möglich zu halten.

Auch wenn der Lagerbestand für das Ziel der niedrigen Kosten so niedrig wie möglich gehalten werden soll, muss darauf geachtet werden, dass auch andere Ziele ausreichend gut erfüllt werden. Daher ist es ein wichtiges Ziel eine optimale Bestandshöhe zu erreichen. Wie in Abschnitt 2.1 gezeigt, ist die optimale Bestandshöhe abhängig von den monetären Kennzahlen Bestandskosten und Kostenverursachung durch Fehlbestände. Da dieser Zielkonflikt eine zentrale Herausforderung der Lagerhaltung ist (vgl. Abschnitt 2.1), ist der Einsatz dieser beiden Kennzahlen von Bedeutung. Jedoch gibt es noch weitere Kennzahlen, die bei der Bestimmung der optimalen Bestandhöhe helfen. So kann unter Zunahme der Kennzahl der Lagerbestandsstruktur, die den Anteil einzelner Güter anteilig am Gesamtlagerbestand wiedergibt, präziser ermittelt werden, durch welches Gut die entstandenen Fehlkosten verursacht wurden. Wenn der Lagerbestand für spezifische Güter den Wert Null hat, weist dies darauf hin, dass das Gut bei jeder neuen Bestellung potenziell Fehlkosten verursachen kann. Ohne diese zusätzliche Information könnten die

Bestände nur pauschal für alle Güter erhöht werden, da das Problem nicht weiter eingegrenzt werden kann. Eine weitere wichtige Kennzahl zur Bewertung des Ziels ist der Servicegrad der Lagerhaltung, der angibt, wie viele Güter bei Anforderung sofort lieferbar sind. Niedrige Werte des Servicegrades können auf zu niedrig gewählte Sicherheitsbestände schließen lassen, die erhöht werden sollten und somit die optimale Bestandshöhe beeinflussen.

Ein weiteres Ziel der Lagerhaltung, das das Ziel der niedrigen Logistikkosten beeinflusst, ist, dass der zur Verfügung stehende Lagerraum bestmöglich genutzt wird. Somit wird unnötig hoher Platzbedarf, der unnötige Kosten verursacht, vermieden, was besonders in Lägern, wie Läger die Lebensmittel beinhalten und kühlen müssen, zu hohen Einsparpotentialen führt. Eine Kennzahl zur Quantifizierung des Ziels der optimalen Lagerraumausnutzung stellt der Raumnutzungsgrad dar.

4.4 Ableitung von Daten zur Bildung der Kennzahlen

Nachdem im vorangehenden Abschnitt Kennzahlen zur Quantifizierung von Zielen eines automatischen Hochregallagers abgeleitet wurden, werden in diesem Abschnitt Daten zur Bildung der Kennzahlen abgeleitet. Diese Daten werden, wie in Abschnitt 4.1 beschrieben, für eine allgemeine Simulationsdatenbasis benötigt.

Zur Ableitung von Daten aus Kennzahlen muss zunächst untersucht werden, wie diese beiden Begriffe in Beziehung zueinander stehen. In Abschnitt 3.3 wird gezeigt, dass Kennzahlen Daten sind, die mit einer Bedeutung versehen sind, und somit eine Information bilden. Daher muss es für jede Kennzahl möglich sein, sie in ein Datum und eine Bedeutung aufzutrennen. Im Falle von Verhältniszahlen (vgl. Abschnitt 3.9.1) müssen sogar mindestens zwei Daten einer Kennzahl identifiziert werden. Diese identifizierten Daten werden mithilfe von mathematischen Operationen und unter Zuhilfenahme einer Bedeutung zur Bildung einer Verhältniszahl genutzt.

Zur Ableitung von Daten, um Kennzahlen zu bilden, ist die Formel zur Berechnung der Kennzahl ausschlaggebend. Jeder in der Berechnungsformel auftretenden Bezeichnung, die eine messbare Größe beschreibt, müssen ermittelbare Daten zugewiesen werden. Zwar haben Daten streng nach Definition keine Bedeutung, also haben keine Bezeichnung, jedoch ist es ohne eine Bezeichnung nicht möglich, eine Beschreibung der zu bestimmenden Daten anzugeben.

Die abzuleitenden Kennzahlen werden in derselben Reihenfolge untersucht, wie sie in Abschnitt 4.3 abgeleitet wurden. Für jede dieser Kennzahlen sollen Daten abgeleitet werden, welche später durch Messungen am physischen Modell bestimmt werden können.

Zuerst wird die Kennzahl der Lagerleistung betrachtet. Sie ist eine Vergleichszahl und setzt sich somit aus einem Zähler und einem Nenner zusammen, die jeweils verschiedene physikalische Größen beschreiben (vgl. Abschnitt 3.9.1). Der Zähler enthält die Summe der Daten für die Anzahl von Einlagerungen und der Anzahl von Auslagerungen. Die Daten müssen während des Experiments getrennt erfasst werden und können später durch die Simulation zur Kennzahlbildung benutzt werden. Der Nenner erfordert eine Zeitspanne, die den Beobachtungszeitraum beschreibt.

Als nächste Kennzahlen wurden die mittlere Einzelspielzeit und mittlere Doppelspielzeit abgeleitet. Beide Kennzahlen sind Mittelwerte und gehören daher zu den absoluten Zahlen (vgl. Abschnitt 3.9.1). Absolute Zahlen sind nur von einer physikalischen Größe abhängig, in diesem Fall ist es die Dauer eines jeden Einzel- bzw. Doppelspiels. Zusätzlich dazu muss, da es Mittelwerte sind, die Anzahl der Einzel- bzw. der Doppelspiele erfasst werden, was laut Definition der absoluten Zahlen jedoch nicht als andere physikalische Größe gewertet wird (vgl. Abschnitt 3.9.1).

Die nächste abgeleitete Kennzahl ist der Füllungsgrad (vgl. Abschnitt 3.9.3). Bei dieser Verhältniszahl müssen als Daten für den Zähler der Anfangs- und der Endbestand des Experiments erfasst werden. Sie werden mit der Strukturzahl der "Lagerkapazität" in Beziehung gesetzt, welche jedoch nicht während des Experiments erfasst wird (vgl. Abschnitt 4.3). Die absolute Zahl "durchschnittlicher Lagerbestand" (vgl. Abschnitt 3.9.3), welche als nächstes abgeleitet wurde, bedient sich derselben Daten, wie der Zähler der Kennzahl "Füllungsgrad", also dem Anfangs- und Endbestand. Somit wird auch die Beziehung zwischen den Kennzahlen "Füllungsgrad" und "durchschnittlicher Lagerbestand" klar. Denn der "Füllungsgrad", als Verhältniszahl, setzt die beiden absoluten Zahlen "durchschnittlicher Lagerbestand" und "Lagerkapazität" in Beziehung (vgl. Abschnitt 3.9.3).

Durch die Ableitung des Ziels des Zielkonfliktes der Lagerhaltung (vgl. Abschnitt 2.1) wurden darauffolgend die Kennzahlen „Bestandskosten“ (vgl. Abschnitt 4.3) und „Kostenverursachung durch Fehlbestände“ (vgl. Abschnitt 4.3) abgeleitet. Für die Verhältniszahl „Bestandskosten“ werden zunächst wieder die Daten für den Anfangs- und Endbestand benötigt, jedoch werden sie in der Formel für die Bestandskosten zur Quotientenbildung mit einem Lagerkostensatz benutzt. Das Datum des Lagerkostensatzes kann jedoch als Strukturzahl (vgl. Abschnitt 4.3) nicht während eines Experiments bestimmt werden. Auch zur Bildung der Kennzahl „Kostenverursachung“ durch Fehlbestände werden Daten benötigt, die nicht während eines Experiments erfasst werden können. Aus der Formel für die Kennzahl (vgl. Abschnitt 3.9.3) geht hervor, dass zum einen die Menge der fehlenden Güter erfasst werden muss, diese aber zum anderen mit

einem Kostensatz in Beziehung gesetzt werden muss, der nicht während eines Experiments erfasst werden kann.

Als nächstes wurde die Vergleichskennzahl „Lagerbestandsstruktur“ abgeleitet. Die Formel der Kennzahl aus Abschnitt 3.9.3 zeigt, dass zur Bildung der Kennzahl vier Daten benötigt werden. Dabei wird für den Zähler der Anfangs- und Endbestand einer Gruppe von Gütern gesucht, die der Anwender festlegen kann. Ein Beispiel wäre der Anfangs- und Endbestand aller Güter, deren Mindesthaltbarkeitsdatum weniger als fünf Tage in der Zukunft liegt. Für den Nenner wird der Anfangs- und Endbestand aller Güter benötigt.

Die absolute Zahl „Servicegrad der Lagerhaltung“ (vgl. Abschnitt 3.9.3) wurde als nächstes abgeleitet. Die Formel zeigt, dass auch der Servicegrad der Lagerhaltung ein Mittelwert ist und zu seiner Bestimmung die Daten der Lieferbereitschaft für jeden Auslagerungsauftrag und die Anzahl aller Auslagerungsaufträge benötigt werden.

Die letzte, abgeleitete Kennzahl ist der Raumnutzungsgrad (vgl. Abschnitt 3.9.4). Zu der Berechnung dieser Verhältniszahl werden die Daten der Volumina aller eingelagerten Lagereinheiten benötigt und mit der Rahmenkennzahl des Volumens des Lagerraums verglichen.

In Tabelle 4.1 werden die zur Bildung der Kennzahlen benötigten Daten, welche in diesem Abschnitt abgeleitet wurden, zusammengefasst.

Tabelle 4.1: Daten zur Kennzahlbildung

Kennzahl	Benötigte Daten
Lagerleistung	Anzahl von Einlagerungen Anzahl von Auslagerungen Bezugszeitraum
Mittlere Einzelspielzeit	Dauer aller Einzelspiele Anzahl von Einzelspielen
Mittlere Doppelspielzeit	Dauer aller Doppelspiele Anzahl von Doppelspielen
Füllungsgrad	Anfangsbestand Endbestand Lagerkapazität
Durchschnittlicher Lagerbestand	Anfangsbestand Endbestand

Bestandskosten	Anfangsbestand Endbestand Lagerkostensatz
Kosten durch Fehlbestände	Fehlmengen Fehlmengenkostensatz
Lagerbestandsstruktur	Anfangsbestand nach Kategorie Endbestand nach Kategorie Anfangsbestand Endbestand
Servicegrad der Lagerhaltung	Lieferbereitschaft für alle Auslagerungsaufträge Anzahl von Auslagerungsaufträgen
Raumnutzungsgrad	Volumina aller Lagereinheiten Volumen des Lagerraums

4.5 Anforderungen an Peripheriegeräte zur Datenerfassung

In Abschnitt 4.4 wurden Daten abgeleitet, die während eines Experiments an einem physischen Modell eines automatischen Hochregallagers erfasst und zum Aufbau einer allgemeinen Simulationsdatenbasis benutzt werden können. Wie einleitend in Kapitel 4 beschrieben, gilt es nach der Ableitung der Daten das Problem zu lösen, wie eine Datenerfassung während des Experiments realisiert werden kann.

Die Datenerfassung an einem physischen Modell wird durch Peripheriegeräte, zu denen auch Sensoren gehören, durchgeführt (vgl. Abschnitt 3.5). Für jedes zu erfassende Datum muss ein Peripheriegerät ausgewählt werden, welches die Aufgabe der Datenerfassung übernimmt. Bevor eine Auswahl an Peripheriegeräten zur Erfassung der in Tabelle 4.1 abgeleiteten Daten getroffen wird, werden Anforderungen, die sich durch Restriktionen des physischen Modells ergeben, gestellt. Dabei wird davon ausgegangen, dass nur wenige Experimente an dem physischen Modell durchgeführt werden. Würde eine große Menge von Experimenten über einen längeren Zeitraum durchgeführt, kämen zusätzliche Anforderungen, wie der Wartungsaufwand, die Austauschbarkeit, die Robustheit gegen Verschmutzung und Langzeitstabilität, in Frage.

Die grundlegende Anforderung an die Sensoren ist, dass sie das geforderte Datum erfassen können. Die Erfassung der Messwerte muss in ausreichender Genauigkeit, Wiederholbarkeit und Auflösung möglich sein (vgl. Abschnitt 3.7.1).

Eine weitere Anforderung an Sensoren ist die Möglichkeit, sie in den zur Verfügung stehenden Bauraum des physischen Modells integrieren zu können. Dazu müssen die Sensoren in einer ausrechend kleinen Bauform zur Verfügung stehen. Andernfalls ist eine Integration in das bestehende physische Modell nicht möglich.

Bezüglich der Integration der Sensoren in das physische System ist es weiterhin eine Anforderung, dass der Aufwand zur Integration in einem für das Projekt angemessenen Rahmen bleibt. Dazu gehört auch, dass die Peripheriegeräte mit einer für physische Modelle gängigen Spannung versorgt werden können, um zusätzliche Spannungswandlungen zu verhindern. Gängige Spannungen sind dabei 24V, 5V und 3,3V.

Eine Einschränkung zur Auswahl von Sensoren ist auch finanzieller Natur. Die zu benutzenden Sensoren dürfen den finanziellen Rahmen dieses Projekts nicht übersteigen.

Damit die Messdaten eines Sensors zur Bildung von Kennzahlen benutzt werden können, müssen sie für das Steuerungssystem des Sensors auswertbar sein. Dies bedeutet, dass der Sensor über eine geeignete Schnittstelle mit dem Steuerungssystem kommunizieren kann. Als Beispiel kann die Kommunikation über SPI zwischen einem Sensor und einem Mikrocontroller genannt werden (vgl. Abschnitt 3.6).

Die nachfolgende Tabelle 4.2 fasst die aufgestellten Anforderungen in einem Anforderungskatalog zusammen.

Tabelle 4.2: Anforderungskatalog für Peripheriegeräte zur Verwendung von Datenerfassung an einem physischen Modell eines Hochregallagers

Anforderung	Beispielhafte Ausprägung
Ausreichende Datenqualität	Genauigkeit, Wiederholbarkeit, Auflösung
Mögliche Integration in das physische Modell	Kleine Bauform
Geringer Aufwand	Standard Versorgungsspannung
Geringe Kosten	Bereits im Modell vorhanden
Auswertbarkeit der Messergebnisse	Benutzung gängiger Schnittstellen

4.6 Auswahl von Peripheriegeräten zur Erfassung der Daten

Nachdem in Abschnitt 4.5 Anforderungen an Peripheriegeräte zur Datenerfassung an einem physischen Modell gestellt wurden, können nun mithilfe dieses Anforderungskatalogs, der in Tabelle 4.2 zusammengefasst wird, die Peripheriegeräte zur Erfassung der in Tabelle 4.1 aufgelisteten Daten ausgewählt werden. Gängige

Peripheriegeräte und insbesondere Sensoren, die für die Datenerfassung benutzt werden, werden in Abschnitt 3.6 und Abschnitt 3.7 näher vorgestellt.

Für die Erfassung von Anzahlen bietet sich die Verwendung eines Zählers (vgl. Abschnitt 3.6) an. Die einfachste Form eines Zählers basiert auf einer softwaretechnischen Lösung, die eine Variable innerhalb eines Programms benutzt, um eine Zählung durchzuführen (vgl. Abschnitt 3.6). Zur Ausführung dieser Lösung bieten sich Mikrocontroller an, die Mini-Rechner sind, die ein Programm ausführen können und dabei günstig und klein sind (vgl. Abschnitt 3.6). Weiterhin weisen sie viele Schnittstellen (vgl. Abschnitt 3.6) auf und lassen sich mit wenig Aufwand in ein physisches Modell integrieren. Zudem sind sie bewährte Bauteile in der Forschung aufgrund ihrer hohen Zuverlässigkeit (vgl. Abschnitt 3.6). Daten, die eine Anzahl wiedergeben und durch eine softwaretechnische Lösung ermittelt werden sollen, sind:

- Anzahl Einzelspiele
- Anzahl Doppelspiele
- Anzahl von Einlagerungen
- Anzahl von Auslagerungen

Einen Sonderfall stellt dabei die Anzahl von Einlagerungsaufträgen dar. Zur Erfassung dieses Datums wird nicht ein softwaretechnischer Zähler, sondern die Verwendung einer Einweglichtschranke (vgl. Abschnitt 3.7.2) ausgewählt, die am Identifikationspunkt (vgl. Abschnitt 2.4) positioniert ist. Sie gibt ein digitales Signal aus, sobald sich die Helligkeit auf der Messstrecke maßgeblich ändert, also z.B. ein Gut angeliefert wird. Da die Genauigkeit und Wiederholbarkeit der Messung allein von der Helligkeit abhängen, muss diese möglichst konstant gehalten werden. Potenzielle Fehlerquellen könnten somit deutlich abweichende Lichtverhältnisse, z.B. zwischen einem dunklen und hellen Raum, oder auch ungewollte Objekte innerhalb der Messstrecke sein. Wenn diese Fehlerquellen jedoch bedacht werden, sind Lichtschranken eine zuverlässige und günstige Lösung. Es wird keine Reflexlichttaster verwendet, da die Einweg-Lichtschranke nicht von den Reflexionseigenschaften der Lagereinheiten abhängig sind und genug Bauraum zur Verfügung steht.

Zur Erfassung aller Daten, die einen Zeitraum bzw. eine Dauer wiedergeben, bietet sich die Verwendung eines Zeitgebers (vgl. Abschnitt 3.6) an. Ein Zeitraum bzw. eine Dauer ist durch einen Anfangs- und einen Endzeitpunkt charakterisiert. Daher reicht es aus, diese beiden Zeitpunkte zu erheben und die Dauer durch einfach Subtraktion des Anfangs- vom Endzeitpunkt zu berechnen. Die Genauigkeit eines Zeitgebers ist maßgeblich von dem ihm zugeführten Takt abhängig (vgl. Abschnitt 3.6). Die Abweichung von der tatsächlich vergangenen Zeit befindet sich im niedrigen Nanosekundenbereich und ist damit in jedem Fall für die Ansprüche der Problemdomäne

ausreichend gering. Ein Vorteil der Benutzung von Zeitgebern ist, dass sie standardmäßig in Mikrocontrollern zum Einsatz kommen (vgl. Abschnitt 3.6) und daher günstig, klein und auswertbar sind und kein Mehraufwand der Integration bei der Benutzung eines Mikrocontrollers entsteht. Zu den beschriebenen Daten gehören:

- Anfangszeitpunkt jedes Einzelspiels
- Endzeitpunkt jedes Einzelspiels
- Anfangszeitpunkt jedes Doppelspiels
- Endzeitpunkt jedes Doppelspiels
- Anfangszeitpunkt des Beobachtungszeitraums
- Endzeitpunkt des Beobachtungszeitraums

Für die Erfassung der Daten "Anfangsbestand nach Kategorie" und "Endbestand nach Kategorie" ist es nötig, jedes Gut im Lager zu identifizieren und kategorisieren zu können. In Abschnitt 3.7.2 wurden zwei gängige Verfahren der Logistik zur Identifikation von Gütern vorgestellt. Zum einen wurde das Barcode-Verfahren vorgestellt, bei dem, für einen Barcodescanner sichtbar, ein Barcode auf dem Gut platziert werden muss, damit eine Zahlenfolge ausgelesen werden kann. Anhand dieser Zahlenfolge können Informationen, z.B. die Kategorie des Gutes, ausgelesen werden. Die Nachteile dieses Verfahrens ergeben sich durch die Forderung, dass der Barcode an einer bestimmten, für den Barcodescanner sichtbaren, Stelle platziert sein muss und eine direkte Sichtlinie zwischen dem Barcodescanner und dem Barcode besteht. Das zweite Verfahren, das die Nachteile des Barcode-Verfahrens nicht aufweist, ist die Verwendung der RFID-Technologie. Durch die Verwendung dieser Technik muss nicht sichergestellt werden, dass eine Sichtlinie zwischen dem Code auf dem Objekt und dem Lesegerät besteht. Der RFID-Transponder des Objekts muss lediglich in der Reichweite des RFID-Lesegeräts sein, um gelesen werden zu können.

Die Daten für die Lagervolumina einer jeden Lagereinheit könnten prinzipiell zwar durch Gewichtssensoren erfasst werden, da in physischen Modellen jedoch meistens standardisierte Modelle von Lagereinheiten mit demselben Gewicht zum Einsatz kommen, würde eine Gewichtsbestimmung über einen Gewichtssensor keinen Sinn ergeben. Ein anderer Weg zur Erfassung des Gewichts ist es, das Gewicht der Lagereinheit auf der Lagereinheit selbst, z.B. auf einen RFID-Transponder, zu speichern. Mit einem RFID-Lesegerät können diese Daten dann erfasst werden. Die Begründung, warum ein RFID-Lesegerät benutzt wird, ist analog zu der Begründung des vorhergehenden Absatzes.

Zusammenfassend ergeben sich die benötigten Peripheriegeräte wie folgt:

- Zähler (softwaretechnisch, Mikrocontroller)

- Einweg-Lichtschranke
- Zeitgeber (Mikrocontroller)
- RFID-Lesegerät

Aus diesen Ergebnissen kann der Schluss gezogen werden, dass die Verwendung eines Mikrocontrollers und seiner Peripheriegeräte (vgl. Abschnitt 3.6) ein wichtiges Bauteil zur Datenerfassung an physischen Modellen automatischer Hochregalläger ist. Ein zusätzlicher Vorteil des Mikrocontrollers ist, dass er die übrigen benötigten Peripheriegeräte, also die Einweglichtschranke und das RFID-Lesegerät, steuern kann und deren Messdaten in seinem Speicher zwischenlagern kann. Durch die Verfügbarkeit vieler Schnittstellen bietet der Mikrocontroller anderen Bauteilen oder Geräten eine hohe Verfügbarkeit der erfassten Daten an.

4.7 Datenpakete

Bisher wurden in diesem Kapitel Daten für eine allgemeine Simulationsdatenbasis eines automatischen Hochregallagers abgeleitet und Peripheriegeräte gefunden, um sie während eines Experiments zu bestimmen. Um eine Simulation basierend auf der allgemeinen Simulationsdatenbasis ausführen zu können, muss das Simulationswerkzeug auf die Simulationsdatenbasis zugreifen können. Da Simulationen für gewöhnlich auf Computern ausgeführt werden, muss ein Weg gefunden werden, wie die Daten der Simulationsdatenbasis auf den Computer gelangen können. Dieses Problem erfordert eine Kommunikation zwischen den Daten erfassenden Peripheriegeräten und dem Computer, was standardmäßig über eine UART-Schnittstelle geschieht (vgl. Abschnitt 4.6). Eine Lösung des Problems kann durch einen Mikrocontroller (vgl. Abschnitt 4.6) realisiert werden. Wie im vorherigen Abschnitt abschließend festgestellt wurde, bietet sich die Verwendung eines Mikrocontrollers an, da der Mikrocontroller andere Peripheriegeräte steuern kann (vgl. Abschnitt 4.6), die Daten so konsolidiert und durch seine UART-Schnittstelle (vgl. Abschnitt 4.6) die Daten an einen Computer übermitteln kann. Die Übermittlung der Daten erfolgt durch Datenpakete, die in diesem Abschnitt entwickelt werden.

Datenpakete werden im Umfang dieser Arbeit als ein Verbund von Daten verschiedenen Typs definiert. Jedes Datum des Datenpakets wird einem Segment innerhalb des Datenpakets zugewiesen. Dabei weisen alle Datenpakete eine Reihenfolge der jeweiligen Segmente auf, die im Folgenden erläutert wird. Jedes Datenpaket besteht aus mindestens zwei Segmenten mit jeweiligen Daten. Das Datum des ersten Segments ist eine Ganzzahl, zur Kategorisierung des Datenpakets, welche nun als Kategorisierungszahl (Kz) definiert wird. Dies ist eine wichtige Information für den Empfänger, also den Computer, da er nach dieser ersten Information erkennt, wie viele Segmente sich in dem Datenpaket

befinden und welche Bedeutung die jeweiligen Daten der Segmente haben. Dies setzt voraus, dass der Empfänger die nötigen Informationen über den Aufbau der Datenpakete besitzt und muss im Vorfeld sichergestellt werden. Das zweite Datum eines Datenpakets stellt den Zeitpunkt dar. Dieser Zeitpunkt entspricht der aktuellen, realen Uhrzeit und kann als Datenanreicherung gesehen werden, um den Informationsgehalt und die Verwendbarkeit der Daten zu steigern. Genau aus diesen Gründen wird der Zeitpunkt dem Datenpaket hinzugefügt, da dadurch, gerade bei längeren Experimenten, eine bessere zeitliche Einordnung der erfassten Daten erfolgen kann. Alle weiteren Daten der Datenpakete sind individuelle Daten verschiedenen Typs, wie z.B. eine Zeitdauer oder die Artikel-Kategorie. Es ist zu beachten, dass Datenpakete desselben Typs, also mit der gleichen Kategorisierungszahl, immer die gleiche Reihenfolge der Daten aufweisen müssen.

Insgesamt werden im Rahmen dieser Arbeit fünf Datenpakete identifiziert, die im weiteren Verlauf des Abschnitts vorgestellt werden. Es wird darauf verzichtet, dass Rahmenkennzahlen (vgl. Abschnitt 4.3) durch Datenpakete übermittelt werden, da davon ausgegangen wird, dass diese dem Computer im Vorfeld zur Verfügung stehen. Dies betrifft die Daten "Lagerkapazität", "Volumen des Lagerraums", "Lagerkostensatz" und den Fehlkostensatz. Weiterhin wird auch der Anfangsbestand nicht durch Datenpakete übermittelt. Es wird vorausgesetzt, dass dem Computer der Anfangsbestand zu Beginn des Experiments bereits bekannt ist. Abgesehen von diesen Ausnahmen werden alle Daten, die in Tabelle 4.1 zu sehen sind, entweder direkt durch Datenpakete an den Computer übermittelt, oder sind aus den übermittelten Daten berechenbar. Die ersten beiden Daten, die in den ersten beiden Segmenten der Datenpakete übermittelt werden, wurden bereits im letzten Absatz vorgestellt.

Das erste Datenpaket mit der Kategorisierungszahl 1 übermittelt Daten, die sich durch ein Einzelspiel ergeben. Der Zeitpunkt, der in Segment zwei übertragen wird, stellt dabei den Zeitpunkt dar, an dem die Einweglichtschranke am Identifikationspunkt (vgl. Abschnitt 2.4) die Lagereinheit detektiert hat. Das erste individuelle Datum, welches im dritten Segment Platz findet, ist die Dauer des Einzelspiels, welche zuvor durch den Mikrocontroller berechnet werden muss. Das Datum in Segment vier wird zur Unterscheidung benötigt, ob das ausgeführte Einzelspiel eine Einlagerung oder eine Auslagerung durchgeführt hat. Im nächsten Segment wird eine Information darüber gegeben, welches Regalfach durch die Ein- oder Auslieferung bedient wurde. Im sechsten Segment wird ein Datum übermittelt, welches zur Spezifizierung einer Kategorie der Lagereinheit, wie z.B. welcher Kategorie einer ABC-Einteilung sie angehört, dient. Als letztes wird das Datum übertragen, welches das Volumen der Lagereinheit repräsentiert.

Das Datenpaket mit der Kategorisierungszahl 2, welches Daten über Doppelspiele übermittelt, hat einen ähnlichen Aufbau wie das erste Datenpaket. Der Zeitpunkt stellt auch hier dar, wann die Lichtschranke am Identifikationspunkt die Lagereinheit detektierte. Da bei einem Doppelspiel immer sowohl eine Ein- als auch eine Auslagerung stattfinden, entfällt die im ersten Datenpaket getroffene Unterscheidung im dritten Segment. Stattdessen wird im zweiten Datenpaket dort schon die Dauer übermittelt. Segment vier spezifiziert das Regalfach, in das eingelagert wurde. Segment fünf übermittelt die Kategorie der eingelagerten Lagereinheit und Segment sechs übermittelt das Datum des Volumens der eingelagerten Lagereinheit. In den nächsten drei Segmenten werden analog zu den Daten der Segmente vier bis sechs alle Informationen zu der ausgelagerten Lagereinheit übermittelt.

Daten über Fehlbestände werden über das Datenpaket mit der Kategorisierungszahl 3 übertragen. Der Zeitpunkt ist dabei der Zeitpunkt des Auslagerungsauftrags. Es wird zusätzlich zur Kategorisierungszahl und dem Zeitpunkt nur ein weiteres Datum übertragen. Dieses Datum, welches sich in Segment drei des Datenpakets befindet, ist die Kategorie der Lagereinheit. So kann differenzierter betrachtet werden, welche Kategorie von der Lagereinheit durch den Auslagerungsauftrag angefordert, aber nicht geliefert wurde.

Die Datenpakete mit den Kategorisierungszahlen 4 und 5 stellen den Start- bzw. Endpunkt des Beobachtungszeitraums dar. Da es keine weiteren relevanten Informationen zu dem Beginn und Ende des Beobachtungszeitraums gibt, sind diese zwei Daten, also die Kategorisierungszahl und der Zeitpunkt, ausreichend.

Es können weitere Datenpakete zum Aufbau einer Simulationsdatenbasis gefunden werden. Denkbar wäre z.B. ein Datenpaket, was der Simulation ermöglicht, die Zeit zu messen, die zwischen einem Einlagerungsauftrag und der tatsächlichen Einlagerung verstreicht. Da mit den vorgestellten Datenpaketen alle Daten aus Tabelle 4.1 dem Computer mitgeteilt werden können, werden im Rahmen dieser Arbeit keine zusätzlichen Datenpakete ermittelt.

Der Inhalt der Segmente, der im Verlauf dieses Abschnitts entwickelten Datenpakete, wird in Tabelle 4.3 zusammengefasst.

Tabelle 4.3: Datenpakete zur Übertragung von Daten einer allgemeinen Simulationsdatenbasis für automatische Hochregalläger

Kz	Datum 1	Datum 2	Datum 3	Datum 4	Datum 5	Datum 6	Datum 7	Datum 8
1	Zeitpunkt	(E)/(A)	Dauer	Fach	Kat.	Vol.	-	-
2	Zeitpunkt	Dauer	Fach (E)	Kat. (E)	Vol. (E)	Fach (A)	Kat. (A)	Vol. (A)
3	Zeitpunkt	Kat. (A)	-	-	-	-	-	-

4	Zeitpunkt	-	-	-	-	-	-	-
5	Zeitpunkt	-	-	-	-	-	-	-

(E) = Einlagerung; (A) = Auslagerung; (E) / (A) = Ein- oder Auslagerung; Kat. = Artikel-Kategorie; Vol. = Volumen

4.8 Speicherung der Daten

Im vorherigen Abschnitt wurden Datenpakete entwickelt, die zur Übermittlung von Daten zwischen einem Mikrocontroller und einem Computer verwendet werden können. Damit das Simulationsprogramm des Computers zu einem späteren Zeitpunkt auf die übermittelten Daten zugreifen kann, müssen sie in einer Form gespeichert werden.

Ein Standardformat zur Speicherung von Daten ist XML (vgl. Abschnitt 3.8). In einem XML-Dokument können die Daten menschenlesbar, selbsterklärend, portierbar und strukturiert gespeichert werden (vgl. Abschnitt 3.8). Aufgrund dieser Vorteile bietet sich eine Verwendung von XML zur Lösung des Problems der persistenten Speicherung der Daten an.

Eine Anforderung an XML-Dokumente ist, dass sie wohlgeformt sind (vgl. Abschnitt 3.8). Um als wohlgeformt zu gelten, müssen Regeln eingehalten werden, die in Abschnitt 3.8 festgehalten sind. Eine mögliche Struktur, die als allgemeine Simulationsdatenbasis dienen kann und ein wohlgeformtes XML-Dokument darstellt, wird im Folgenden entwickelt.

Zunächst wird ein Wurzelelement benötigt. Alle folgenden Elemente und Daten werden diesem Wurzelelement untergeordnet sein. Als Name des Elements wird „experimentData“ ausgewählt. Dem Wurzelelement direkt untergeordnet sind zwei Elemente. Das erste Element bekommt den Namen „frameIndicators“ und wird als übergeordnetes Element für alle Rahmenkennzahl nach Schulte (vgl. Abschnitt 3.9.1) benutzt. Da das XML-Dokument eine allgemeine Simulationsdatenbasis darstellen soll und Rahmenzahlen wichtige Informationen für das Simulationswerkzeug repräsentieren (vgl. Abschnitt 3.9.1), wurden sie in das XML-Dokument inkludiert. Die Rahmenkennzahlen werden nicht durch übermittelte Daten des Mikrocontrollers spezifiziert, sondern manuell durch den Benutzer eingetragen. Das andere Element hat den Namen „receivedData“ und ist das übergeordnete Element für die Datenpakete, welche durch den Mikrocontroller übermittelt werden und in dem XML-Dokument gespeichert werden sollen. Es wird eine Unterscheidung zwischen Rahmenkennzahlen und Daten aus einem Datenpaket getroffen, da, wie in Abschnitt 4.3 beschrieben, Rahmenkennzahlen nicht während des Experiments erhoben werden und somit semantisch nicht zu dem Element „receivedData“ gehören.

Jedes Datenpaket hat ein weiteres übergeordnetes Element „dataPackage“, welches zur Unterscheidung der verschiedenen, übermittelten Datenpakete dient und dem Element „received Data“ untergeordnet ist. Die Daten der Datenpakete werden jeweils durch ein weiteres Element, welches als Namen die Bezeichnung des jeweiligen Datums trägt, dem Element „dataPackage“ untergeordnet. Analog zu den Daten der Datenpakete werden auch die Rahmenkennzahlen mit zusätzlichen Elementen dem übergeordneten Element „frameIndicators“ zugeordnet. Der Aufbau des XML-Dokuments, mit einem exemplarischen Datenpaket und zwei Rahmenzahlen gefüllt, wird in Abbildung 4.3 gezeigt. Dieses XML-Dokument stellt eine mögliche Simulationsdatenbasis für automatische Hochregalläger dar.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<experimentData>
    <frameIndicators>
        <Lagerkapazität>50</Lagerkapazität>
        <AnzahlGassen>1</AnzahlGassen>
    </frameIndicators>
    <receivedData>
        <dataPackage>
            <Kategorisierungszahl>1</Kategorisierungszahl>
            <Datum1>10:42:11</Datum1>
            <Datum2>E</Datum2>
            <Datum3>7824</Datum3>
            <Datum4>3/8</Datum4>
            <Datum5>A</Datum5>
            <Datum6>4</Datum6>
        </dataPackage>
    </receivedData>
</experimentData>
```

Abbildung 4.3: Schema des XML-Dokuments

5 Exemplarische Erfassung von Daten an einem physischen Modell eines automatischen Hochregallagers

In Kapitel 4 wurde der theoretische Aufbau einer Simulationsdatenbasis für automatische Hochregalläger mithilfe von Datenerfassung an einem physischen Modell beschrieben. Zur Validierung dieser theoretischen Vorgehensweise wird in diesem Kapitel, mit der Methode des Experiments, exemplarisch eine Simulationsdatenbasis nach der vorgestellten Vorgehensweise aufgebaut.

Dazu wird zunächst in Abschnitt 5.1 ein Überblick über die Ausgangssituation gegeben, in der die Planung des Experiments beginnt. In Abschnitt 5.2 wird die Planung des Experiments durchgeführt, die die Verwendung von Lagereinheiten, die Erstellung eines Ablaufplans und den Lagerbestand zu Beginn des Experiments umfasst. In Abschnitt 5.3 werden alle nötigen Maßnahmen ergriffen, um die in Kapitel 4 entwickelte Vorgehensweise durch den in Abschnitt 5.2 entwickelten Ablaufplan des Experiments umzusetzen. Nachdem alle Vorbereitungen für das Experiment betrachtet wurden, wird in Abschnitt 5.4 die Durchführung des Experiments dokumentiert und die Ergebnisse dargestellt. Die Bewertung des durchgeführten Experiments wird abschließend in Abschnitt 5.5 gegeben.

5.1 Ausgangssituation

Das physische Modell eines automatischen Hochregallagers, an dem das exemplarische Erfassen von Daten durchgeführt werden soll, wurde bereits ausführlich in Abschnitt 3.10 beschrieben. Alle mechanischen Komponenten waren in einem beanstandungslosen Zustand und die bereits integrierten Sensoren, Taster und Motoren waren funktionsfähig. Lediglich eine Spannungsversorgung und eine Steuerung fehlten gänzlich und mussten bereitgestellt werden.

5.2 Planung eines Experiments

5.2.1 Lagereinheiten des Experiments

Eine Lagereinheit setzt sich aus dem Ladehilfsmittel und einer Ladung zusammen (vgl. Abschnitt 2.2). Für das zu planende Experiment kann die Lagereinheit vereinfachend als ein Ganzes betrachtet werden, um den Modellieraufwand, besonders hinsichtlich der physischen Repräsentation, zu verringern. Um Daten bezüglich einer Lagerbestandsstruktur zu ermitteln (vgl. Abschnitt 3.9.3) muss es eine Art Unterscheidungsmerkmal geben. Für dieses Experiment wurde als

Unterscheidungsmerkmal eine Klassifizierung nach der ABC-Einteilung (vgl. Abschnitt 2.1) vorgenommen, die sich nach der Zugriffshäufigkeit richtet. Somit gibt es insgesamt drei zu unterscheidende Arten von Lagereinheiten, die im Folgenden als A-Artikel, B-Artikel und C-Artikel bezeichnet werden. Die Information über das Unterscheidungsmerkmal muss mit einem geeigneten Verfahren, während des Experiments, erfasst werden.

Weiterhin werden zur Berechnung des Raumnutzungsgrades (vgl. Abschnitt 3.9.3) Daten über die Volumina jeder Lagereinheit benötigt. Dabei müssen die Volumina nicht unbedingt dem realen Volumen des physischen Modells der Lagereinheiten entsprechen. Um eine Variation der Volumina zu erzeugen, wie sie oftmals in realen Situationen auftreten, können entsprechend berechnete, unterschiedliche Volumenwerte einer Lagereinheit zugewiesen werden. Der Vorteil liegt darin, dass die Lagereinheiten in einer Einheitsgröße hergestellt werden können, trotzdem aber Untersuchungen für verschiedene Volumina möglich sind. Es muss nur sichergestellt werden, dass die zugewiesenen Volumina das maximale Volumen der Regalfächer des physischen Modells nicht überschreiten. Es sei darauf hingewiesen, dass die reine Betrachtung der Volumina zu Problemen führen könnte. Ein möglicherweise auftretendes Problem ist, dass zwar das Volumen niedriger ist als das maximale Volumen des Lagerfachs, die tatsächliche Höhe der Lagereinheit jedoch die maximal zulässige Höhe eines Lagerfachs übersteigt.

5.2.2 Ablaufplanung und Vereinfachungen

Das Ziel des Experiments ist das exemplarische erfassen von Daten zum Aufbau einer Simulationsdatenbasis. Deshalb ist es sinnvoll, möglichst alle Daten, die erfasst werden sollen, in verschiedenen Zuständen zu erfassen, um eine fehlerfreie Erhebung nachweisen zu können. Das Experiment erhebt keinen Anspruch darauf, einen möglichst realitätsnahen Ablauf abzubilden, sondern wird in vereinfachter Form durchgeführt.

Die zu erfassenden Daten und die dafür nötigen Sensoren können Abschnitt 4.6 entnommen werden. Der Ablauf des Experiments soll sicherstellen, dass ausreichend Daten für eine exemplarische Simulation zur Verfügung stehen, die im Rahmen dieser Arbeit aber nicht durchgeführt wird. Damit über den Beobachtungszeitraum für jedes Datum nicht ausschließlich gleiche Messwerte erfasst werden, müssen verschiedenartige Situationen eintreten, die ungleiche Messwerte bedingen. Folgende Situationen müssen durchgeführt werden, bzw. sich während des Experiments ergeben:

- Einzelspiele unterschiedlicher ABC-Artikeln und Volumina, zur Variation der mittleren Einzelspielzeit, des durchschnittlichen Lagerbestands, der Lagerleistung, des Füllungsgrades, der Lagerbestandsstruktur und des Raumnutzungsgrades

- Doppelspiele mit unterschiedlichen ABC-Artikeln und Volumina, zur Variation der mittleren Doppelspielzeit, des durchschnittlichen Lagerbestands, der Lagerlagerleistung, des Füllungsgrades, der Lagerbestandsstruktur und des Raumnutzungsgrades
- Auslagerungsauftrag mit Lieferbereitschaft, zur Variation des Servicegrades der Lagerhaltung
- Auslagerungsauftrag ohne Lieferbereitschaft, zur Variation des Servicegrades der Lagerhaltung

Durch das Durchführen dieser vier Situationen kann sichergestellt werden, dass über den Beobachtungszeitraum für jedes Datum nicht ausschließlich gleiche Messwerte erfasst werden. Dabei reicht es aus, wenn jede dieser Situationen mindestens einmal eintritt.

Um die genannten Situationen herbeizuführen, muss das physische Modell des automatischen Hochregallagers Ereignissen ausgesetzt werden. Die beiden Ereignisse, die eintreten können, um das Hochregallager zu einer Reaktion zu bringen, sind der Einlagerungs- und der Auslagerungsauftrag (vgl. Abschnitt 2.4). Mit diesen beiden Ereignissen können alle geforderten Situationen herbeigeführt und untersucht werden.

Wird ein Einlagerungsauftrag ausgeführt und es wird ein Auslagerungsauftrag gestellt, bevor der Prozess der Einlagerung vollendet ist, führt das System ein Doppelspiel aus. Ist dies nicht der Fall, wird ein Einzelspiel ausgeführt. Zusätzlich müssen sowohl ein Auslagerungsauftrag gestellt werden, der durch den Lagerbestand bedient werden kann, als auch ein Auslagerungsauftrag, der aufgrund von Fehlbeständen nicht bedient werden kann.

Zur Steuerung des Ablaufs des Experiments wird, wie in Abschnitt 4.6 empfohlen, ein Mikrocontroller benutzt, der zusätzlich auch die Datenerfassung durchführt, so entsteht kein Mehraufwand durch Integration mehrerer Bauteile. Die Steuerung des Ablaufs umfasst auch die Auslösung der eingehend beschriebenen Ereignisse des Einlagerungs- und Auslagerungsauftrags.

Wie in Abschnitt 5.2.1 geschrieben, werden Ladungen und somit die Lagereinheiten in drei Artikel unterteilt. Diese Einteilung folgt der ABC-Einteilung, die Artikel ihrer Zugriffshäufigkeit nach kategorisiert (vgl. Abschnitt 2.1). Basierend auf dieser Einteilung ist es sinnvoll, zur Minimierung der Laufzeit die vorhandenen Lagerplätze in drei Lagerzonen, entsprechend den ABC-Artikeln, aufzuteilen. Das Ziel ist es, Artikel mit einer hohen Zugriffshäufigkeit mit möglichst geringen Fahrwegen ein- und auszulagern. Für das Experiment würde dies heißen, dass A-Artikel in den unteren, B-Artikel in den darüberliegenden und C-Artikel in den obersten Regalfächern gelagert werden. Da die Anzahl von A- und B-Artikeln zahlenmäßig niedriger ist, als die der C-Artikel, wird folgende Einteilung von Zonen verwendet:

- Zone für A-Artikel: Reihe 1
- Zone für B-Artikel: Reihe 2
- Zone für C-Artikel: Reihe 3-5

Als Lagerplatzvergabestrategie wird die chaotische Lagerung mit einer Zonung angewendet (vgl. Abschnitt 2.1). Somit können Artikel nur in ihren jeweiligen Zonen gelagert werden, jedoch ist innerhalb der Zone kein fester Platz reserviert. Die Zuteilung innerhalb der Zone geschieht zufällig zwischen allen noch nicht belegten Lagerplätzen.

Als Auslagerungsstrategie (vgl. Abschnitt 2.1) wird keine in der Praxis übliche gewählt. Da das Ziel nicht in der Untersuchung einer Ein- und Auslagerungsstrategie besteht, reicht eine einfachere Umsetzung der Problemlösung. Ein Auslagerungsauftrag fordert nicht eine spezielle Lagereinheit an, sondern fordert eine beliebige Lagereinheit einer spezifizierten Kategorie an. Dazu wird bei einem Auslagerungsauftrag immer die Lagereinheit ausgewählt, die den niedrigsten Reihenindex aufweist, vorausgesetzt es ist eine Lagereinheit der auszulagernden Kategorie in der Reihe verfügbar. Im Spezialfall der C-Artikel, die sich über drei Ebenen verteilen und somit die Reihenindizes dreifach vorkommen, wird die Annahme getroffen, das zuerst Artikel der dritten, dann der vierten und zuletzt der fünften Reihe ausgelagert werden. Wenn Fehlbestände vorliegen, also bei einem Auslieferungsantrag keine Lagereinheit der angeforderten Kategorie vorhanden ist, wird auch im Nachhinein, wenn wieder eine Lagereinheit der Kategorie verfügbar ist, nicht nachgeliefert.

Zum Aufbau der Simulationsdatenbasis werden während des Experiments insgesamt 14 Ereignisse ausgeführt, die sich auf sechs Einlagerungsaufträge und acht Auslagerungsaufträge verteilen. Als Beginn des Experiments und des Beobachtungszeitraums wird der Eingang des ersten Einlagerungsauftrags, registriert durch die Einweglichtschranke am Identifikationspunkt, gewertet. Das Ende des Experiments und des Beobachtungszeitraums ist dann erreicht, wenn die vorgesehenen 14 Ereignisse fehlerfrei durchgeführt wurden und das Regalbediengerät in die Ausgangsposition zurückgekehrt ist.

Die auszuführenden Ereignisse unterliegen einer vorbestimmten Reihenfolge. Der Zeitpunkt, an dem ein Einlagerungsauftrag eingeht, ist nicht exakt vorbestimmt und unterliegt einer Unsicherheit, da dieser Prozess manuell ausgeführt wird. Durch eine solche Abweichung ändert sich die Länge des Beobachtungszeitraums. Da die Ergebnisse des Experiments nur exemplarischer Natur und nicht an konkrete Vorgaben eines Anwendungsfalles gebunden sind, ist eine Abweichung des Beobachtungszeitraums für die Ergebnisse des Experiments irrelevant. Auslagerungsaufträge werden durch die Steuerungssoftware des Mikrocontrollers ausgelöst, sind aber so programmiert, dass sie die Reihenfolge einhalten. Da das Auslösen der Einlagerungsaufträge manuell geschieht,

ist der entwickelte Vorgang als halbautomatische Datenerfassung aufzufassen (vgl. Abschnitt 3.5)

Durch den Zeitpunkt der Auslösung der Auslagerungsaufträge durch den Mikrocontroller kann bestimmt werden, welche Art von Spiel (vgl. Abschnitt 2.4) das physische Modell des automatischen Hochregallagers ausführt. Zur Ausführung eines Doppelspiels wird der Auslagerungsauftrag direkt nach dem manuellen Auslösen des Einlagerungsauftrags ausgelöst. Sollen zwei aufeinander folgende Ein- und Auslagerungsaufträge durch zwei Einzelspiele durchgeführt werden, wird der Auslagerungsauftrag erst ausgelöst, nachdem das Einzelspiel des Einlagerungsauftrags vollendet wurde.

Die Reihenfolge von Spielen während des Experiments, die sich durch die Komposition der Ereignisse Ein- und Auslagerungsauftrag ergibt, wird in Tabelle 5.1 gezeigt

Tabelle 5.1: Reihenfolge von Spielen während des Experiments

Reihenfolge	Spiel	Kategorie (E)	Kategorie (A)
1	Einzelspiel	C	-
2	Einzelspiel	-	A
3	Doppelspiel	B	A
4	Einzelspiel	A	-
5	Doppelspiel	C	A
6	Einzelspiel	-	A
7	Einzelspiel	-	C
8	Doppelspiel	A	C
9	Doppelspiel	A	B
10	Einzelspiel	-	B

(E) = Einlagerung, (A) = Auslagerung

Der Ablauf des Tests kann nicht vollautomatisch durchgeführt werden, sondern erfordert manuelles Eingreifen. Dies geschieht sowohl am Identifikations- als auch am Kontrollpunkt. Am Identifikationspunkt werden die einzulagernden Lagereinheiten von Hand so platziert, dass die zuständige Einweglichtschranke eine Lagereinheit registriert und eine Abholung durch das Regalbediengerät ordnungsmäßig möglich ist. Am Kontrollpunkt werden die ausgelagerten Lagereinheit von Hand entfernt, um Platz für weitere Lagereinheiten am Kontrollpunkt zu schaffen.

5.2.3 Lagerbestand zu Experimentbeginn

Zu Beginn des Testlaufs werden sich bereits Lagereinheiten im Modell des Lagers befinden. Das Wissen darüber, welche Lagerplätze mit Lagereinheiten bereits belegt sind und welcher Kategorie diese angehören, wird der Steuerungssoftware im Vorfeld bereitgestellt. Obwohl theoretisch eine dynamische Bestimmung des Anfangsbestandes

möglich ist, indem an jedem Regalfach eine Auslagerung versucht wird, die ausgelagerte Lagereinheit zum Identifikationspunkt gebracht wird (falls das Regalfach belegt war), dort identifiziert und danach wieder an derselben Stelle eingelagert wird, ist dieses Verfahren ein unnötiger Mehraufwand, der nicht zielführend ist. Die Initialbelegung der Lagerplätze, für die die angegebenen Lagereinheiten zufällig in den vorgesehenen Zonen platziert wurden, des Experiments lautet wie folgt:

- Zwei A-Artikel
- Zwei B-Artikel
- Vier C-Artikel

Insgesamt hat das physische Modell eine Kapazität von 50 Lagereinheiten (vgl. Abschnitt 3.10). Es wird davon ausgegangen, dass jede dieser bereits eingelagerten Lagereinheiten konform der Einlagerungsstrategie nach der ABC-Einteilung eingelagert wurde und die entsprechenden Informationen auf den jeweiligen RFID-Transpondern gespeichert sind. Insgesamt sind zu Beginn bereits acht Lagereinheiten eingelagert, was einem Füllungsgrad von 16% entspricht. Der Anfangsbestand wurde so gewählt, dass einmal eine Situation eintritt, in der keine Lieferbereitschaft besteht. Dieser Fall tritt bei dem sechsten Spiel der Reihenfolge aus Tabelle 5.1 auf, in dem ein A-Artikel ausgelagert werden soll, aber kein A-Artikel mehr auf Lager ist.

5.3 Umsetzung des Experiments

5.3.1 Vorbereitung der Lagereinheiten

In Abschnitt 4.4 wurden zwei Anforderungen an Lagereinheiten gestellt. Zum einen müssen sie eindeutig, einer ABC-Einteilung folgend, identifizierbar sein und zum anderen müssen sie Daten über ihr Volumen bereitstellen können. In Abschnitt 4.6 wurde bereits die Verwendung der RFID-Technologie empfohlen (vgl. Abschnitt 3.7.2), welche auch die in Abschnitt 4.5 entwickelten Anforderungen erfüllen. Folglich müssen die Informationen auf einem RFID-Transponder gespeichert werden und der RFID-Transponder muss sich auf den physischen Modellen der Lagereinheiten befinden, um stets eine Identifikation zu gewährleisten. Wie in Abschnitt 5.2.1 beschrieben, sollen für das Volumen verschiedene Werte gespeichert werden.

Ein RFID-Transponder besitzt die Eigenschaft, mehrere Daten in einem internen Speicher zu speichern (vgl. Abschnitt 3.7.2). Für dieses Experiment muss ein Datum zur Kategorisierung und ein Datum zur Spezifizierung des Volumens gespeichert werden. Die RFID-Transponder der Lagereinheiten in diesem Experiment wurden mit zufälligen Volumenwerten im Bereich 1 bis 5 beschrieben, das maximale Volumen eines Regalfachs

wird mit 6 angegeben. Es wurden keine spezifischen Einheiten für das Volumen verwendet, da dieses für ein exemplarisches Erfassen von Daten nicht zielführend ist.

Um den RFID-Transpondern mitzuteilen, welche konkreten Datenwerte sie speichern sollen, müssen die Speicher beschrieben werden. Dazu wurde ein RFID-Lesegerät, welches gleichzeitig als RFID-Schreibgerät genutzt werden kann, und ein Mikrocontroller genutzt. Es wurde ein Programm entwickelt (Anhang A), welches mithilfe einer Bibliothek für das RFID-Lesegerät die gewünschten Daten auf den RFID-Transponder schreiben kann.

Der Entwurf eines physischen Modells einer Lagereinheit, das die gestellten Anforderungen erfüllt und einen integrierten RFID-Transponder besitzt, ist Thema des folgenden Absatzes.

5.3.2 Entwurf eines physischen Modells einer Lagereinheit

Da bisher keine physischen Modelle von Lagereinheiten für das physische Modell des automatischen Hochregallagers existierten, muss für die Durchführung des Experiments eine Lagereinheit entworfen und mehrfach hergestellt werden. Wie in Abschnitt 2.2 beschrieben, bestehen Lagereinheiten zum einen aus Ladehilfsmitteln und zum anderen aus einer Ladung. In Abschnitt 5.2.1 wurde als Vereinfachung bereits die Betrachtung der Lagereinheit als Ganzes getroffen. Dies bedeutet, dass das Modell nur aus einem Teil bestehen kann.

Hauptsächlich richtet sich die Geometrie des Modells der Lagereinheit nach den Regalfachmaßen des Modells des physischen Hochregallagers. Zusätzlich ergibt sich durch die Benutzung der RFID-Technologie die Forderung, dass ein RFID-Transponder auf dem Modell Platz finden muss (vgl. Abschnitt 3.8). Eine kompakte und günstige Form von RFID- Transpondern wird als Schlüsselanhänger angeboten. Da diese kompakt genug sind, um auf dem Modell der Lagereinheit Platz zu finden, wird diese Art von RFID- Transpondern benutzt. An der Oberfläche der Lagereinheit wird eine Aussparung, entsprechend der Geometrie des RFID-Transponders, vorgesehen, in die der RFID-Transponder gelegt werden kann.

Die Lagereinheiten werden mithilfe eines 3D-Druckers hergestellt. Insgesamt werden für den geplanten Ablauf 13 Lagereinheiten benötigt, die sich aus dem Anfangsbestand und den einzulagernden Lagereinheiten berechnen, unter der Voraussetzung, dass einmal ausgelagerte Lagereinheiten wieder eingelagert werden können. Ein Foto der entworfenen Lagereinheit im eingelagerten Zustand sind in der folgenden Abbildung 5.1 dargestellt. Ein Aufkleber auf der Frontseite der Lagereinheit dient der Identifizierung der Lagereinheit durch Menschen, da der Inhalt der gespeicherten Daten des RFID-Transponders ohne ein Lesegerät nicht ermittelbar ist.

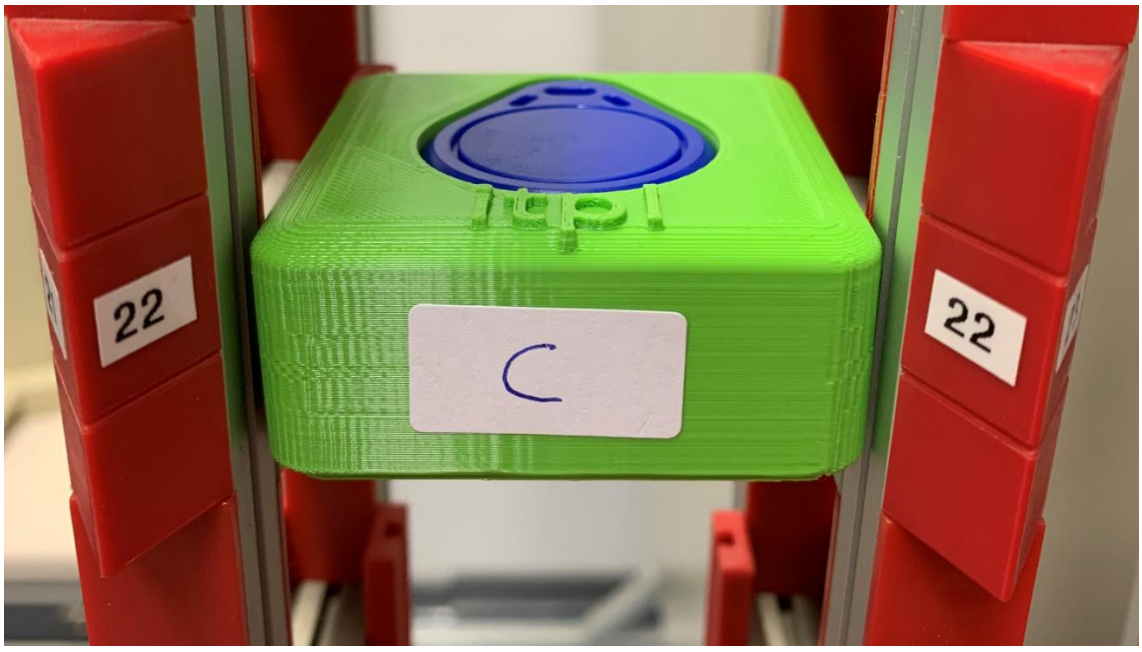

Abbildung 5.1: Physisches Modell einer Lagereinheit

Im nächsten Abschnitt werden Anpassungen des physischen Modells des automatischen Hochregallagers thematisiert, die zur Umsetzung des geplanten Experiments nötig sind.

5.3.3 Anpassungen am physischen Modell

Um den Ablauf des geplanten Experiments aus Abschnitt 5.2.2 umsetzten zu können, müssen Anpassungen am physischen Modell des automatischen Hochregallagers vorgenommen werden. Begonnen wird mit der Auswahl von zusätzlichen Bauteilen, die integriert werden müssen. Danach wird kurz die Stromversorgung des physischen Modells erklärt und abschließend die Position der neuen Bauteile im physischen Modell erläutert.

In Abschnitt 4.6 wurden bereits Klassen von Sensoren ausgewählt, die zur Datenerfassung benötigt werden. Nun werden reale Sensoren dieser Klassen ausgewählt und in das physische Modell integriert. Bis auf die Einweg-Lichtschranke fehlen dem physischen Modell alle in Abschnitt 4.6 identifizierten Peripheriegeräte bzw. Sensoren.

Zunächst wurde ein Mikrocontroller zur Steuerung der Ablaufplanung des Experiments ausgewählt. Durch die Vorteile, die die Plattform Arduino bietet (vgl. Abschnitt 3.6), wird der Mikrocontroller Arduino Mega 2560 (vgl. Abschnitt 3.6) verwendet. Er bietet durch viele Schnittstelen, wie UART, SPI und I^2C, eine gute Konnektivität. Durch die 54 zu Verfügung stehenden I/O Pins ist eine Steuerung aller Komponenten des physischen Modells gewährleistet. Die Betriebsspannung des Mikrocontrollers beträgt 5V, zusätzlich besitzt er einen internen Spannungswandler (vgl. Abschnitt 3.6), der es ermöglicht, Peripheriegeräte mit einer Spannung von 3,3V zu versorgen. Zusätzlich zu seiner

Steuerungsfunktion, bietet der Mikrocontroller, mit seinen integrierten Peripheriegeräten, einen Zeitgeber zur Zeitmessung und einen softwaretechnischen Zähler an (vgl. Abschnitt 3.6), welche in Abschnitt 4.6 als Bestandteil der Datenerfassung gefordert wurden.

Um das Lesen von Informationen auf Lagereinheiten zu ermöglichen, wird ein RFID-Lesegerät benötigt (vgl. Abschnitt 3.6). Da das bestehende Modell noch nicht mit einem RFID-Lesegerät ausgestattet war, musste ein solches integriert werden. Wie in Abschnitt 2.4 beschrieben, haben automatische Hochregalläger einen Identifikationspunkt und einen Kontrollpunkt. An diesen beiden Punkten sollen die Daten der Lagereinheit erfasst werden, im Falle dieses Experiments also durch RFID-Lesegeräte. Um den Aufwand der Integration gering zu halten, wird nur ein RFID-Lesegerät am Identifikationspunkt verwendet. Um das exemplarische Vorgehen zu zeigen, reicht es mit nur einem Lesegerät zu arbeiten, Erweiterungen des Modells können aber im Zuge weiterer Arbeiten hinzugefügt werden. Das verbaute RFID-Lesegerät der Firma joy-it basiert auf dem NXP MFRC-522-Modul (MFRC-522 2020) und wird mit einer Spannung von 3,3V betrieben. Um mit dem Lesegerät zu kommunizieren, bietet es eine SPI-Schnittstelle an.

Ein weiteres Peripheriegerät, welches in das physische Modell integriert werden soll, wurde in Abschnitt 4.7 zur Datenanreicherung identifiziert. Dort wird gefordert, dass die reale Uhrzeit mit jedem Datenpaket versendet werden soll. Zur Bestimmung der realen Uhrzeit wird eine Echtzeituhr benötigt (vgl. Abschnitt 3.6). Für dieses Experiment wurde eine Echtzeituhr auf Basis des DS3231-Moduls ausgewählt, welches eine Kommunikation über I^2C ermöglicht und mit 5V betrieben wird (DS3231 2020).

Zum Betrieb des physischen Modells wird eine Stromversorgung benötigt. Die Stromversorgung wurde über ein Labornetzgerät hergestellt. Das Labornetzgerät liefert eine Versorgungspannung von 24V und einer Stromstärke von 500mA. Da der Mikrocontroller und einige Peripheriegeräte nicht über eine Spannung von 24V versorgt werden können, sondern eine niedrigere Spannung benötigen, mussten Spannungswandler eingesetzt werden. Insgesamt werden drei verschiedene Spannungen benötigt: 24V, 5V und 3,3V.

Mit 24V wurden alle Motoren und Sensoren versorgt, die bereits im Modell verbaut waren. Das betrifft die Motoren des Regalbediengeräts, den Reflexlichttaster im Regalbediengerät und die Einweg-Lichtschranken des Identifikations- und Kontrollpunktes (vgl. Abschnitt 2.5).

Eine Sonderrolle nehmen die Taster zur Positionsbestimmung im Regalbediengerät ein. Da sie über dieselbe Leitung wie die Reflexlichtschranke mit Strom versorgt wurden, mussten diese Taster ebenfalls mit 24V belastet werden. Beim Auslösen der Taster würde somit ein 24V Signal an den Mikrocontroller gegeben werden, was zu einer Überschreitung der maximal verträglichen Spannung führen (vgl. Abschnitt 3.6) und

somit wahrscheinlich den Mikrocontroller schädigen würde. Um dies zu verhindern musste die maximale Spannung auf einen verträglichen Wert, in diesem Fall 4,5V, begrenzt werden. Dies wurde mithilfe eines passend ausgelegten Spannungsteilers (vgl. Schossig 1993) gelöst.

Ein Wert von 5V entspricht der sogenannten Logikspannung. Mit dieser Spannung wurde der Mikrocontroller, die Echtzeituhr, sowie alle übrigen Taster zur Positionsbestimmung versorgt, was einen Spannungsteiler aufseiten der Taster obsolet machte. Die Spannungswandlung von 24V auf 5V wurde mit einem LM2596 (2020) realisiert.

Eine Spannung von 3,3V wurde benötigt, um das RFID-Lesegerät mit Spannung zu versorgen. Für diese Spannungswandlung konnte ein interner Spannungswandler des Mikrocontrollers (vgl. Abschnitt 3.6) benutzt werden.

Nachdem der Mikrocontroller und weitere Peripheriegeräte ausgewählt und ihre Spannungsversorgung erläutert wurden, wird nun die Integrierung dieser Bauteile in das physische Modell erläutert. Der Mikrocontroller wurde nicht fest mit dem Gestell des physischen Modells verbunden, sondern abseits davon und über Kabel. Auch die Echtzeituhr wurde nicht fest mit dem Gestell verbunden, sondern wurde abseits durch Kabel mit dem Mikrocontroller verbunden. Die Spannungsteiler, der Spannungswandler und ansonsten nötige Verkabelung wurden auf Steckplatinen realisiert. Sie wurden mit Kabeln jeweils mit dem physischen Modell, dem Mikrocontroller und dem Labornetzgerät verbunden.

Zur Befestigung des RFID-Lesegeräts wurde eine Haltevorrichtung entworfen und mit einem 3D Drucker verwirklicht. Die Haltevorrichtung des Lesegeräts besteht aus einem höhenverstellbaren Ständer und einem Kragarm. Durch die Höhenverstellbarkeit ist es gewährleistet, dass der Abstand zwischen RFID-Lesegerät und RFID-Transponder eingestellt werden kann. Dies ist wichtig, da die Reichweite des RFID Sender/Empfänger-Paars ausreichend gering sein muss, um eine reibungslose Kommunikation zu gewährleisten (vgl. Abschnitt 3.6). In Abbildung 5.2 ist das ausgedruckte Bauteil, welches an der schwarzen Färbung zu erkennen ist, nach seiner Integration in das physische Modell zu sehen. Zusätzlich ist das RFID-Lesegerät bereits über Schrauben mit dem Kragarm verbunden.

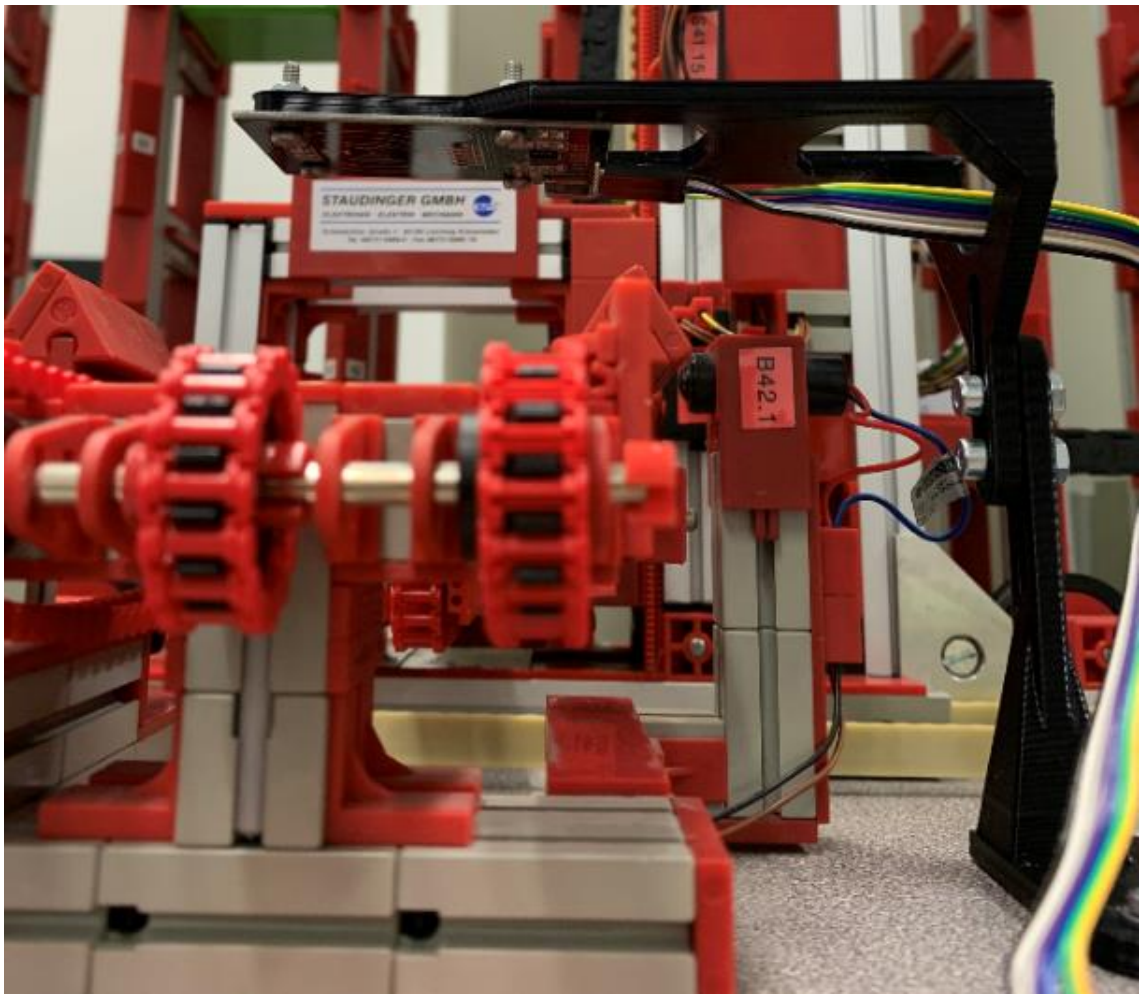

Abbildung 5.2: Halterung eines RFID-Lesegeräts

Im Folgenden ist eine Auflistung gegeben, die Bauteile, die zusätzlich in dem physischen Modell verbaut wurden, oder zu dessen Steuerung benutzt wurden, beinhaltet. Ausgeschlossen sind dabei Kabel und Widerstände für die Spannungsteiler.

- Mikrocontroller: Arduino Mega 2560
- RFID-Lesegerät: NXP MFRC-522
- Echtzeituhr: DS3231
- Spannungswandler: LM2596

5.3.4 Steuerungssoftware

Die Steuerungssoftware (Anhang B) für das physische Modell des automatischen Hochregallagers wurde mithilfe der Arduino Entwicklungsumgebung entwickelt (Arduino 2020), da der benutzte Mikrocontroller von dem Unternehmen Arduino hergestellt wird und die Entwicklungsumgebung optimiert für dessen Programmierung ist. Die Steuerungssoftware ist verantwortlich für die Datenerfassung und den Ablauf des Experiments.

Der Funktionsumfang der Steuerungssoftware umfasst insgesamt sieben Funktionen. Diese werden im Folgenden kurz vorgestellt.

Die erste Funktion ist die der Positionierung. Diese Funktion erlaubt es, das Regalbediengerät vor einem über eine x- und y-Koordinate spezifizierten Regalfach zu positionieren. Die x-Achse entspricht dabei einer Bewegung längs des Lagers, die y-Achse einer vertikalen Bewegung. Dazu wird zunächst eine horizontale Bewegung entlang der x-Achse ausgeführt, bis die gewünschte x-Koordinate erreicht ist. Danach wird die horizontale Bewegung entlang der y-Achse durchgeführt. Da eine Bestimmung der Koordinaten über die verbauten Taster des physischen Modells geschieht (vgl. Abschnitt 3.10), ist eine diagonale Bewegung, wie sie in realen Hochregallägern stattfindet (vgl. Abschnitt 2.4), nicht möglich.

Zwei weitere Funktionen sind das Ein- und Auslagern der Lagereinheiten. Sie setzen voraus, dass das Regalbediengerät bereits am Ort der Ein- oder Auslagerung positioniert ist. Bei der Einlagerung in das Hochregallager fährt das Regalbediengerät in die Oberstellung des Lagerfachs (vgl. Abschnitt 3.10). Nach Erreichen der Oberstellung wird die Lagereinheit in das Hochregallager gefahren, bis die Vorderstellung des Regalbediengerätes erreicht ist. Nun wird das Regalbediengerät bis zur Unterstellung des Lagerfachs heruntergefahren. Bei diesem Übergang, von der Lagerfach Oberstellung in die Unterstellung, wird die Lagereinheit durch seitliche Absätze im Lagerfach gehalten. Daraufhin wird, nach dem Zurückfahren des Regalbediengeräts von der Vorder- in die Mittelstellung, die Ausgangsposition erreicht. Für die Auslagerung aus dem Hochregallager wird der Prozess in umgekehrter Reihenfolge durchlaufen. Weiterhin ist es mit diesen Funktionen möglich, Lagereinheiten vom Identifikationspunkt aufzunehmen und Lagereinheiten am Kontrollpunkt abzulegen. Die Aufnahme am Identifikationspunkt geschieht analog zu einem Auslagerungsvorgang aus dem Hochregallager, jedoch wird das Regalbediengerät nicht in Richtung seiner Vorder- sondern in die Hinterstellung bewegt. Das Ablegen einer Lagereinheit am Kontrollpunkt wird durch den umgekehrten Prozess der Abholung am Identifikationspunkt realisiert.

Das Detektieren neuer Lagereinheiten am Identifikationspunkt und das Auslesen der Daten des RFID-Transponders stellt eine weitere Funktion dar. Mithilfe der Einweglichtschranke (vgl. Abschnitt 3.7.2) wird zunächst geprüft, ob sich ein Objekt am Identifikationspunkt befindet. Ist dies der Fall, kann durch das RFID-Lesegerät, welches sich am Identifikationspunkt befindet (vgl. Abschnitt 2.4), der Inhalt des RFID-Transponders der Lagereinheit ausgelesen werden. Somit werden Daten über die Kategorie der Lagereinheit und dessen Volumen erfasst.

Nachdem eine Lagereinheit am Identifikationspunkt abgeholt wurde, wird eine weitere Funktion der Steuerungssoftware benötigt, um zu bestimmen, in welches Lagerfach die

Lagereinheit eingelagert werden soll. Daher wurde eine Funktion zur Umsetzung der in Abschnitt 5.2.2 erläuterten Einlagerungsstrategie umgesetzt. Auch die Umsetzung der Auslagerungsstrategie ist Teil der Steuerungssoftware.

Eine weitere wichtige Funktion der Steuerungssoftware ist, dass sie die Datenpakete aus Abschnitt 4.7 an einen angeschlossenen Computer senden kann. Die Software sammelt alle für ein Datenpaket benötigte Daten, reichert sie mit der aktuellen Uhrzeit der Echtzeituhr an, und versendet sie über eine geeignete Schnittstelle.

5.3.5 Bereitstellung der erfassten Daten

Um die Daten für eine Simulation nutzen zu können, müssen sie der Simulation durch eine Simulationsdatenbasis zugänglich gemacht werden (vgl. Abschnitt 3.4). Dazu werden die Daten persistent mit einem geeigneten Verfahren außerhalb des Mikrocontrollers gespeichert. Aufgrund des hohen Standardisierungsgrades, der Popularität, einfacher Implementation, Menschenlesbarkeit und der Möglichkeit zur Verwendung eigener Tags, eignet sich besonders eine Datenspeicherung in einem XML-Dokument (vgl. Abschnitt 4.8). Somit wird die in Abschnitt 3.4 aufgezeigte Forderung, in der eine Verfügbarmachung der Simulationsdatenbasis in einem gängigen Format gefordert wird, durch die Popularität und den Standardisierungsgrad, erreicht. Die Menschenlesbarkeit und die anpassbaren Tags ermöglichen es, dass das Dokument selbsterklärend sein kann und somit die Zugänglichkeit anderer Personen zu der Simulationsdatenbasis stark vereinfacht wird.

Da Simulationen für gewöhnlich nicht auf einem Mikrocontroller stattfinden, ist eine Speicherung der Daten auf dem Mikrocontroller nicht sinnvoll, sondern sollte auf einem Computer stattfinden (vgl. Abschnitt 3.5). Damit die Daten auf einem Computer gespeichert werden können, müssen sie zunächst vom Mikrocontroller übertragen werden. Da Computer standardmäßig einen Baustein zum Empfangen serieller Daten durch UART über eine USB verbaut haben, und die Datenübertragung durch das UART-Verfahren das in der Praxis hauptsächlich genutzte Verfahren darstellt, bietet sich für die Übertragung besonders das UART-Verfahren des Mikrocontrollers über eine USB-Verbindung an (vgl. Abschnitt 3.6).

Das serielle Senden von Daten ist mit einem Mikrocontroller einfach möglich (vgl. Abschnitt 3.6), jedoch ist auf Seiten des Computers ein höherer Aufwand zur Verarbeitung der Daten nötig. Für diesen Zweck wurde ein Programm in der Sprache C++ (Will 2018) geschrieben, das auf einem Windows Computer ausgeführt werden kann und als Empfängersoftware, für die durch den Mikrocontroller gesendeten Daten, dient (Anhang C). Die Programmiersprache C++ ermöglicht es mithilfe einer durch Arduino zur Verfügung gestellten Bibliothek (Arduino 2020), eine serielle Kommunikation über

eine USB-Schnittstelle zu führen. Mithilfe einer Bibliothek zur Bearbeitung von XML Dokumenten speichert das erwähnte Programm die vom Mikrocontroller gesendeten Daten in einer XML Datenstruktur. Die verwendete XML-Bibliothek nennt sich pugixml, weitere Informationen sind auf der Website der Bibliothek zu finden (pugixml 2020). Um den Experimentablauf visuell zu unterstützen, eine Kontrolle während des Experiments zu ermöglichen und dem Benutzer einen einfacheren Zugang zu dem Programm zu ermöglichen, besitzt das Programm zusätzlich eine grafische Oberfläche, für die zwei zusätzliche Bibliotheken benutzt wurden. Zum einen wurde die Bibliothek Simple and Fast Multimedia Library (SFML) benutzt, welche eine API zu verschiedenen Multimedia-Komponenten zur Verfügung stellt (SFML 2020). Als Ergänzung wurde eine zusätzliche Bibliothek, das Texus' Graphical User Interface, kurz TGUI, verwendet (TGUI 2020). Es erweitert die Funktionspalette der SFML Bibliothek um einige Widgets.

Die Kommunikation zwischen Mikrocontroller und des erwähnten Programms geschieht über Datenpakete, die der Mikrocontroller sendet und in Abschnitt 4.7 spezifiziert wurden. Es sei erwähnt, dass die Kommunikation der Geräte ausschließlich in eine Richtung, nämlich vom Mikrocontroller zum Computer, verläuft. Das Programm wartet während des Experiments auf Datenpakete und schreibt sie, sobald sie erfolgreich gelesen wurden, in die XML Datei und in eine Ausgabebox der Grafischen Oberfläche (GUI) des Programms. Über einen Button kann der Anwender das Warten auf Datenpakete starten und beenden. Wird das Warten auf Datenpakete beendet, wird auch das XML-Dokument durch das Programm gespeichert. Das verwendete Format des XML-Dokuments entspricht der in Abschnitt 4.8 entwickelten Struktur. Abbildung 5.3 zeigt die Benutzeroberfläche während der Beobachtung eines Experiments.

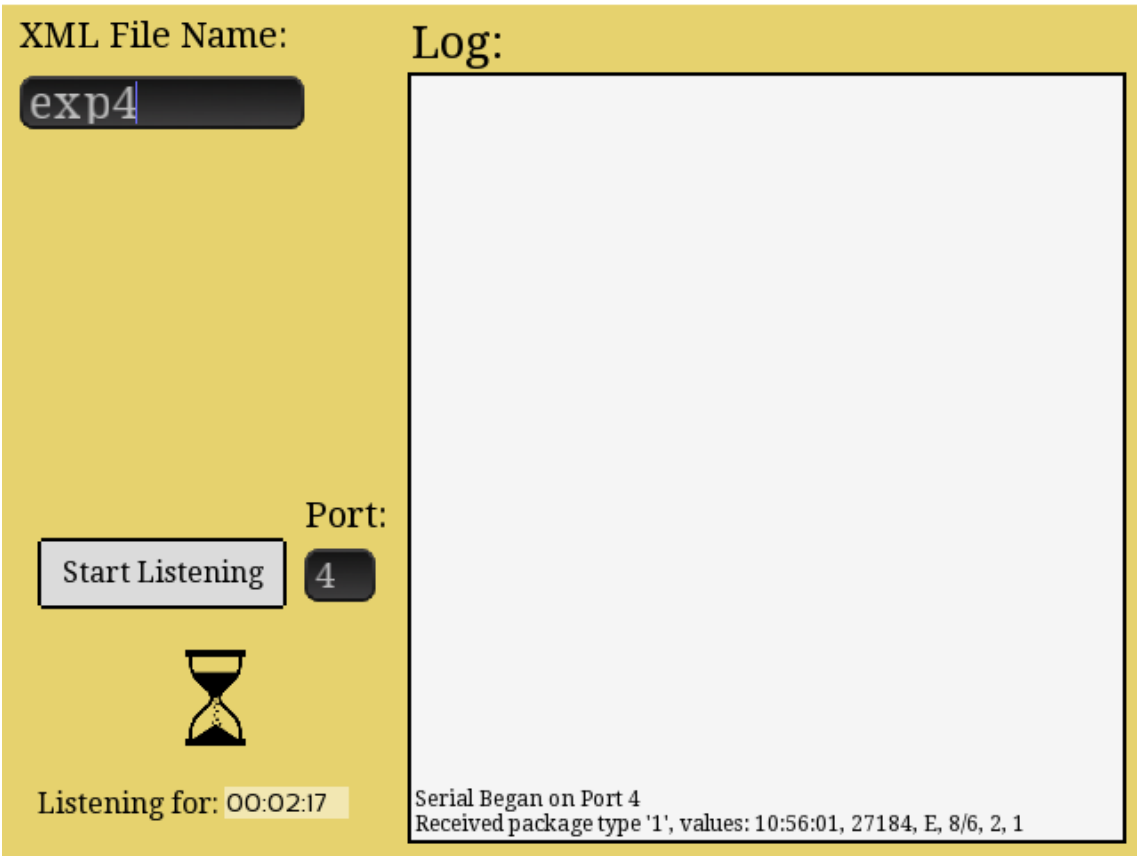

Abbildung 5.3: Benutzeroberfläche des Programms zum Empfangen von Datenpaketen

5.4 Durchführung des Experiments

Der Ablauf des Experiments und dessen Planung wurde in Abschnitt 5.2 erklärt. Die Umsetzung des in Abschnitt 5.2 entwickelten Ablaufs wurde in Abschnitt 5.3 betrachtet. In diesem Abschnitt wird das entwickelte Experiment durchgeführt und eine exemplarische Simulationsdatenbasis durch Datenerfassung aufgebaut.

Zunächst wurden einige Vorbereitungen getroffen. Es wurde sichergestellt, dass alle Lagereinheiten, wie in Abschnitt 5.2.1 beschrieben, Daten bezüglich ihrer Kategorie und ihres Volumens auf dem zugehörigen RFID-Transponder aufweisen. Außerdem wurde eine Integration aller zusätzlicher Bauteile, wie sie in Abschnitt 5.3 beschrieben ist, durchgeführt.

Nachdem die Vorbereitungen abgeschlossen waren, konnte die Stromversorgung des physischen Modells des automatischen Hochregallagers hergestellt werden. Die Steuerungssoftware des Mikrocontrollers wurde so programmiert, dass der Ablauf des Experiments erst startet, sobald die erste Lagereinheit manuell auf den Identifikationspunkt gelegt wird. Bevor das Experiment auf diese Weise gestartet wurde, wurde die Empfängersoftware (vgl. Abschnitt 5.3) auf dem Computer gestartet, eine

Verbindung des Computers mit dem Mikrocontrollers über USB hergestellt und die Überwachung der UART-Schnittstelle aktiviert, damit die vom Mikrocontroller gesendeten Datenpakete durch den Computer empfangen und zu einem XML-Dokument verarbeitet werden konnten.

Der Beginn des Experiments wurde durch das manuelle Platzieren einer Lagereinheit am Identifikationspunkt gestartet. Der weitere Verlauf der halbautomatischen Datenerfassung entsprach dem in Abschnitt 5.3.2 entwickelten Ablauf und konnte ohne Probleme durchgeführt werden. Das Ende des Experiments wurde erreicht, nachdem die Steuerungssoftware das letzte Ereignis ausgelöst hatte, das Regalbediengerät wieder in die Ausgangsstellung zurückgekehrt war, und das "Ende-Datenpaket" durch den Mikrocontroller versendet wurde. Um die erfassten Daten am Computer in dem über den Zeitraum des Experiments aufgebauten XML-Dokument zu speichern, musste der entsprechende Button der GUI betätigt werden. Ein Ausschnitt des erhaltenen XML-Dokuments wird in Abbildung 5.4 gezeigt. Das vollständige XML-Dokument ist dem Anhang (D) beigefügt.

```
<?xml version="1.0" encoding="UTF-8"?>
<experimentData>
    <frameIndicators>
    </frameIndicators>
    <receivedData>
        <dataPackage>
            <identifier>4</identifier>
            <timestamp>01:42:18</timestamp>
        </dataPackage>
        <dataPackage>
            <identifier>1</identifier>
            <timestamp>01:42:45</timestamp>
            <duration>27259</duration>
            <inOrOut>E</inOrOut>
            <rackBay>7/4</rackBay>
            <category>3</category>
            <volume>3</volume>
        </dataPackage>
        <dataPackage>
            <identifier>1</identifier>
            <timestamp>01:43:04</timestamp>
            <duration>18124</duration>
            <inOrOut>A</inOrOut>
            <rackBay>0/0</rackBay>
            <category>1</category>
            <volume>3</volume>
        </dataPackage>
```

Abbildung 5.4: Auszug des durch das Experiment erhaltenen XML-Dokuments

5.5 Fazit des durchgeführten Experiments

Die Durchführung des Experiments hat gezeigt, dass durch Datenerfassung an einem physischen Modell eines automatischen Hochregallagers die Schaffung einer

Simulationsdatenbasis möglich ist. Die in Kapitel 4 erarbeiteten theoretischen Grundlagen konnten auf das Experiment übertragen werden und lieferten Ergebnisse gemäß der in Kapitel 4 aufgestellten Forderungen. Die Auswahl des Mikrocontrollers, der übrigen Peripheriegeräte und Sensoren hat einen reibungslosen Ablauf des Experiments ermöglicht. Auch die Datenübertragung durch UART und ein Empfangen der Daten durch ein C++ Programm, welches gleichzeitig ein XML-Dokument erstellt, hat sich als funktionsfähige und einfach umzusetzende Möglichkeit zur Speicherung und Verfügbarmachung der erfassten Daten auf einem Computer bewährt.

In Abschnitt 3.4 wurde die Forderung beschrieben, dass Daten für eine Simulationsdatenbasis nicht unter Laborbedingungen, sondern unter realen Bedingungen erhoben werden sollten. Zwar stellt das physische Modell ein automatisches Hochregallager unter realen Bedingungen dar, doch werden durch die hohe Zahl an getroffenen Vereinfachungen und Limitationen durch die Bauweise des physischen Modells laborähnliche Bedingungen geschaffen. Eine Verbesserung dieses Umstandes kann durch eine realitätsnähere Bauweise des physischen Modells und der Reduzierung der vereinfachenden Annahmen erwirkt werden. Beispielsweise ist die fehlende Möglichkeit einer diagonalen Bewegung des Regalbediengeräts ein deutlicher Unterschied zu realen Systemen und wird sich maßgeblich auf die Spielzeit auswirken. Durch bauliche Maßnahmen, die eine Diagonalbewegung ermöglichen, könnte die Qualität der erfassten Daten "mittleren Einzelspielzeit" und der "mittleren Doppelspielzeit" womöglich deutlich gesteigert werden.

Nichtsdestotrotz hat sich das Experimentieren an einem physischen Modell als funktionsfähige Datenquelle erwiesen. Mögliche Anwendungen könnten Experimente an dem physischen Modell im Rahmen von Lehrzwecken sein, wie Law (2015, vgl. Abschnitt 3.10) bereits beschrieben hat, denn dafür können Daten in angemessener Genauigkeit erfasst und die annähernden Laborbedingungen vernachlässigt werden. Nach Einschätzung des Autors wäre auch eine Anwendung für Industrieprojekte denkbar, wenn ein realitätsnäheres, physisches Modell eines automatischen Hochregallagers zur Verfügung stehen würde. Denkbare Untersuchungsgegenstände könnten verschiedene Lagerstrategien und ihr Vergleich sein.

6 Zusammenfassung und Ausblick

Das Ziel dieser Arbeit ist die Identifizierung von Daten für eine allgemeine Simulationsdatenbasis zur Simulation von automatischen Hochregallägern und deren exemplarischer Erfassung an einem physischen Modell eines automatischen Hochregallagers.

Dazu wurden zunächst Grundlagen automatischer Hochregalläger, der Methode der Simulation, von Daten, der Datenerfassung und der Speicherung von Daten gegeben. Basierend auf identifizierten Ansätzen aus der Literatur wurde eine Herangehensweise zur Identifikation von Daten für eine allgemeine Simulationsdatenbasis entwickelt. Diese Herangehensweise beginnt mit dem Identifizieren von Zielen automatischer Hochregalläger, die sich aus Zielen der Lagerhaltung, der Logistik und des Unternehmens ableiten lassen. Nach der Identifizierung von Zielen wurden Kennzahlen zu deren Quantifizierung abgeleitet, die in der Praxis gängig sind. Durch Ableiten von Daten aus den gefundenen Kennzahlen konnten Daten für eine allgemeine Simulationsdatenbasis identifiziert werden, die dazu benutzt werden können, wichtige Kennzahlen und Ziele automatischer Hochregalläger mithilfe einer Simulationsstudie zu untersuchen. Weiterhin wurden Wege gefunden, die Daten der allgemeinen Simulationsdatenbasis während eines Experiments an einem physischen Modell eines automatischen Hochregallagers zu erfassen. Dazu wurden für jedes zu erfassende Datum ein Peripheriegerät oder ein Sensor ausgewählt, damit eine Erfassung am physischen Modell erfolgen kann. Es wurde auch ein Weg zur Verfügbarmachung der erfassten Daten an einem Computer vorgestellt, indem Datenpakete für eine Kommunikation zwischen einem Mikrocontroller, welche die Sensoren steuert, und dem Computer erarbeitet wurden und eine Form der Datenspeicherung auf dem Computer erörtert wurde.

Für das exemplarische Erfassen der Daten in einem Experiment wurde das auszuführende Experiment zunächst geplant. Dazu gehörten die Rolle von Lagereinheiten, eine Ablaufplanung des Experiments, nötige Vereinfachungen und ein Initialbestand. Für die Umsetzung des geplanten Experiments waren mehrere Schritte nötig. Es wurde ein physisches Modell einer Lagereinheit entworfen und deren Verwendung mit der RFID-Technologie thematisiert. Es wurden Anpassungen an dem physischen Modell vorgenommen, damit alle in Kapitel 4 vorgestellten theoretischen Ergebnisse validiert werden konnten. Für die Umsetzung wurden drei Programme entwickelt. Das erste Programm kann RFID-Transponder mit benutzerdefinierten Daten beschreiben. Das zweite Programm stellt die Steuerungssoftware des Mikrocontrollers dar, der auch für den Experimentablauf verantwortlich war. Das dritte Programm wird verwendet, um die im theoretischen Teil entwickelten Datenpakete an einer UART-Schnittstelle zu empfangen und in ein XML-Dokument umzuwandeln, welches der Verfügbarmachung

der erfassten Daten dient. Zuletzt wurden das geplante Experiment ausgeführt, die Ergebnisse vorgestellt und ein Fazit des Experiments gezogen.

Das Experiment hat gezeigt, dass eine Datenerfassung an einem physischen Modell eines automatischen Hochregallagers möglich ist und auch wie dieser Prozess realisiert werden kann. Bisher wurden nur einige ausgewählte Daten erfasst. Ziel anschließender Arbeiten könnte die Identifikation weiterer Daten für eine allgemeine Simulationsdatenbasis automatischer Hochregalläger sein und Wege zu finden, diese an einem physischen Modell zu bestimmen. Außerdem könnten Untersuchungen angestellt werden, in denen die erfassten Daten in einer Simulationsstudie verwendet werden und die Qualität der erzielten Ergebnisse bewertet wird. Untersuchungsgegenstand einer solchen Simulationsstudie könnte beispielsweise eine Lagerstrategie oder eine Schwachstellenfindung sein. Weitere Arbeiten könnten untersuchen, inwieweit die Ergebnisse der Experimente skalierbar sind. Für ein mehrgassiges automatisches Hochregallager mit vielen Lagerplätzen müsste ein entsprechend großes physisches Modell gebaut werden, doch vielleicht reicht es aus, nur eine Gasse mit einer niedrigeren Anzahl von Lagerplätzen nachzubilden, daran Experimente auszuführen und die Ergebnisse am Ende zu skalieren. Außerdem wäre es denkbar, die Anforderungen, die sich an den Bau des physischen Modells ergeben, zu untersuchen.

Generell ist festzuhalten, wie in Abschnitt 3.5 gezeigt wurde, dass das Thema der Datenerfassung an einem physischen Modell, zum Zweck der Datenbeschaffung für eine Simulationsstudie, bisher nur unzureichend erforscht wurde und diesbezüglich eine Forschungslücke besteht. Nach Meinung des Autors sind weitere Forschungen zu diesem Thema vielversprechend und könnten dazu beitragen, die Planung weiter zu verbessern und kostengünstiger zu gestalten.

Literaturverzeichnis

Arduino: Arduino. 2020. URL: https://www.arduino.cc/ (zuletzt geprüft am 13.1.2020).

Arnold, D.; Furmans, K.: Materialfluss in Logistiksystemen. 7. Aufl. Berlin, Heidelberg: Springer, 2019.

Banks, J.: Handbook of Simulation. Principles, Methodology, Advances, Applications, and Practice. New York: Wiley, 1998.

Banks, J.; Carson, J.; Nelson, B.; Nicol, D.: Discrete-Event System Simulation. 4. Aufl. New Jersey: Prentice-Hall, 2004.

Bauer, P.: Planung und Auslegung von Palettenlagern. Berlin, Heidelberg: Springer, 1985.

Bernhard, J.; Dragan, M.; Wenzel, S.: Evaluation und Erweiterung der Kriterien zur Klassifizierung von Visualisierungsverfahren für GNL. Dortmund: Sonderforschungsbereich 559, 2005.

Bernhard, J.; Dragan, M.; Wenzel, S.: Bewertung der Informationsgüte in der Informationsgewinnung für die modellgestützte Analyse großer Netze der Logistik. Dortmund: Sonderforschungsbereich 559, 2007.

Bernstein, H.: Messelektronik und Sensoren. Grundlagen der Messtechnik, Sensoren, analoge und digitale Signalverarbeitung. Wiesbaden: Vieweg, 2014.

Bichler, K.; Krohn, R.; Riedel, G.; Schöppach, F.: Beschaffungs- und Lagerwirtschaft. Praxisorientierte Darstellung der Grundlagen, Technologien und Verfahren. 9. Aufl. Wiesbaden: Gabler, 2010.

Bichler, K.; Krohn, R.; Philippi, P.: Gabler Kompaktlexikon Logistik. 2. Aufl. Wiesbaden: Gabler, 2011.

Bichler, K.; Schröter, N.: Praxisorientierte Logistik. 3. Aufl. Stuttgart: Kohlhammer, 2004.

Bode, W.; Hoya, U.: Verbesserte Planungsbasen durch EDV-gestützte Systeme. In: Jünemann, R. (Hrsg.): Materialfluß und Logistiksysteme. Entwicklungen und Perspektiven. Dortmund, 1985, S. 101-105.

Bourret, R.: XML and Database. Felton: Bourret, 2005.

Brandes, T.: Betriebsstrategien für Materialflußsysteme unter besonderer Berücksichtigung automatisierter Lager. Aachen: Shaker, 1997.

Brinkschulte, U.; Ungerer, T.: Mikrocontroller und Mikroprozessoren. 3. Aufl. Berlin, Heidelberg: Springer, 2010.

Bürgi, A.: Führen mit Kennzahlen. Ein Leitfaden für den Klein- und Mittelbetrieb. 4. Aufl. Bern, 1985.

Bürkeler, A.: Kennzahlensysteme als Führungsinstrument. Zürich, 1977.

Clausen, U.: Entsorgung und Kreislaufwirtschaft. In: Arnold, D.; Isermann, H.; Kuhn, A.; Tempelmeier, H.; Furmans, K. (Hrsg.): Handbuch Logistik. 3 Aufl. VDI-Buch. Berlin: Springer, 2008, S. 487 – 524.

Cleve, J.; Lämmel, U.: Data Mining. München: Oeldenbourg, 2014.

Czeguhn, K.: Planung, Ausführung und Betrieb von Hochregallägern. In: Dorloff, F.-D.; Roth, P.: Service- und Materialmanagement. Wiesbaden: Gabler, 1985, S. 131 – 152.

DIN 19222: Leittechnik. Begriffe. 2001.

DIN IEC 60060-351: Internationales Elektrotechnisches Wörterbuch. Teil 351: Leittechnik. Berlin: Beuth, 2014.

DS32321: Extremely Accurate I^2C-Integrated RTC/TCXO/Crystal. URL: https://datasheets.maximintegrated.com/en/ds/DS3231.pdf (zuletzt geprüft am 13.1.2020).

Dürr, H.: Datenerfassung in der kommerziellen Datenverarbeitung. Berlin: DeGruyter, 2018.

Ehrmann, H.: Logistik. 7. Aufl. Herne: Kiehl, 2012.

Frank, U.: Standardisierungsvorhaben zur Unterstützung des elektronischen Handels: Überblick über anwendungsnahe Ansätze. Wirtschaftsinformatik, 43 (2001) 3, S. 282-293.

Frank, T.: Lagersysteme. In: Arnold, D.; Isermann, H.; Kuhn, A.; Tempelmeier, H.; Furmans, K. (Hrsg.): Handbuch Logistik. 3 Aufl. VDI-Buch. Berlin: Springer, 2008, S. 645 – 667.

Frazelle, E.: World-Class Warehousing and Material Handling. 2. Aufl. New York: McGraw Hill, 2016.

Gehrke, W.; Winzker, M.; Urbanski, K.; Woitowitz, R.: Digitaltechnik. Grundlagen, VHDL, FPGAs, Mikrocontroller. 7. Aufl. Wiesbaden: Vieweg, 2016.

Groll, K.: Erfolgssicherung durch Kennzahlensysteme. 4. Aufl. Freiburg: Haufe, 1991.

Groll, M.; Weber, J.; Bacher, A.: Steuerung der Supply Chain. Aber mit welchen Instrumenten? Vallendar: WHU, 2003.

Gutenschwager, K.; Rabe, M.; Spieckermann, S.; Wenzel, S.: Simulation in Produktion und Logistik. Grundlagen und Anwendungen. Berlin, Heidelberg: Springer, 2017.

Hänisch, T.: Grundlagen Industrie 4.0. In: Andelfinger, V.; Hänisch, T.: Industrie 4.0. Wie cyber-physische Systeme die Arbeitswelt verändern. Wiesbaden: Gabler, 2017.

Hansen, H.: Wirtschaftsinformatik I. Grundlagen betrieblicher Informationsverarbeitung. 7. Aufl. Stuttgart: Fischer, 1996.

Hausladen, I.: IT-gestützte Logistik. Systeme – Prozesse – Anwendungen. 3. Aufl. Wiesbaden: Gabler, 2016.

Heinz, K.; Nusswald, M.: Logistikdaten effizient erfassen: praxisorientierte Auswahl von Methoden. Dortmund: Praxiswissen, 1996.

Heiserich, O.; Helbig, K.; Ullmann, W.: Logistik. Eine praxisorientierte Einführung. 4. Aufl. Wiesbaden, Gabler, 2011.

Hering, E.; Schönfelder, G.: Sensoren in Wissenschaft und Technik. Funktionsweise und Einsatzgebiete. 2. Aufl. Wiesbaden: Vieweg, 2018.

Hesse, S.; Schnell, G.: Sensoren für die Prozess und Fabrikautomation. Funktion – Ausführung – Anwendung. 7. Aufl. Wiesbaden: Vieweg, 2018.

Hüning, F.: Sensoren und Sensorenschnittstellen. Berlin: Oeldenbourg, 2016.

Hunt, V.; Puglia, A.; Puglia, M.: RFID – A guide to radio frequency identification. New Jeresey: Wiley, 2007.

ITPL: Fachgebiet IT in Produktion und Logistik. 2020. URL: http://www.itpl.mb.tu-dortmund.de/cms/de/Fachgebiet/Home/ (zuletzt geprüft am 13.1.2020).

Jahangirian, M.; Eldabi, T.; Naseer, A.; Stergioulas, L.; Young, T.: Simulation in Manufacturing and Business. A Review. European Journal of Operational Research, 203 (2010) 1, S. 1 – 13.

Jünemann, R.: Beitrag zur Planungsmethode des Stückgutlagers in Industrie- und Handelsunternehmen. Berlin, 1970.

Jünemann, R.: Systemplanung für Stückgutläger. Mainz: Krausskopf, 1971.

Jünemann, R.; Schmidt, T.: Materialflußsysteme. Systemtechnische Grundlagen. 2. Aufl. Berlin, Heidelberg: Springer, 2000.

Kaplan, R.; Norton, D.: The balanced scorecard. Translating strategy into action. Boston: Harvard Business School Press, 1996.

Knepper, L.: Verstärkte materialflußtechnische Integration von Hochregallagern in Produktions- und Umschlaganlagen. Internationaler Logistik Kongreß, Dortmund, 1983, S. 210 - 224.

Koether, R.: Technische Logistik. 3. Aufl. München: Hanser, 2007.

Krcmar, H.: Einführung in das Informationsmanagement. Berlin, Heidelberg: Springer, 2011.

Krieg, W.: Anforderungen an die Simulationstechnik und Lösungsansätze. In: Jünemann, R. (Hrsg.): Materialfluß und Logistiksysteme. Entwicklungen und Perspektiven. Dortmund, 1985, S. 81-90.

Landry, M.; Malouin, J.; Oral, M.: In search of a valid view of model validation for operations research. European Journal of Operational Research 66 (1993) 2, S. 161 – 167.

Law, A.: Simulation Modeling and Analysis. 4. Aufl. New York: McGraw-Hill, 2007.

Law, A.: Simulation Modeling and Analysis. 5. Aufl. New York: McGraw-Hill, 2015.

LM2596: Simple Switcher Power Converter. URL: http://www.ti.com/lit/ds/symlink/lm2596.pdf (zuletzt geprüft am 13.1.2020).

Lochthowe, R.: Logistik-Controlling. Entwicklung flexibilitätsorientierter Strukturen und Methoden zur ganzheitlichen Planung, Steuerung und Kontrolle der Unternehmenslogistik. Frankfurt am Main: Lang, 1990.

Martin, H.: Transport- und Lagerlogistik. Systematik, Planung, Einsatz und Wirtschaftlichkeit. 10. Aufl. Wiesbaden: Vieweg, 2017.

MFRC-522: RFID Modul. URL: https://www.joy-it.net/files/files/Produkte/SBC-RFID-RC522/SBC-RFID-RC522-Anleitung.pdf (zuletzt geprüft am 13.1.2020).

Moos, A.: XQuery und SQL/XML in DB2-Datenbanken. Verwaltung und Erzeugung von XML-Dokumenten in DB2. Wiesbaden: Vieweg, 2008.

Nayyar, A.; Puri, E.: A Review of Arduino Board's, Lilypad's & Arduino Shields. International Conference on Computing for Sustainable Global Development, 2016.

North, K.: Wissensorientiere Unternehmensführung. Wissensmanagement gestalten. 6. Aufl. Wiesbaden: Gabler, 2016.

Oeller, K.: Systemorientierte Unternehmensführung mit Hilfe kybernetischer Kennzahlensysteme. In: Malik, F. (Hrsg.): Praxis des systemorientierten Managements. Stuttgart, 1979.

Onggo, B.; Hill, J.: Data identification and data collection methods in simulation: a case study at ORH Ltd. Journal of Simulation 8 (2014), S. 195-205.

Ossola-Haring, C.: Handbuch Kennzahlen zur Unternehmensführung. Kennzahlen richtig verstehem, verknüpfen und interpretieren. Landsberg am Lech: mi-Fachverlag, 2006.

Perera, T.; Liyanage, K.: Methodology for rapid identification of input data in the simulation of manufacturing systems. Simulation Practice and Theory 7 (2000), S. 645–656.

Peter, P.: Logistik-Datenmanagement und planungsbegleitende Datenbasis. In: Jünemann, R. (Hrsg.): Integrierte Materialflußsysteme. Köln: Verlag TÜV Rheinland, 1988, S. 39 - 45.

Pfohl, H.; Zöllner, W.: Effizienzmessung in der Logistik. Die Betriebswirtschaft 51 (1991) 3, S. 323 – 339.

Pfohl, H.-C..: Logistikmanagement. 3. Aufl. Berlin, Heidelberg: Springer, 2016.

Pidd, M.: Tools for Thinking: Modelling in Management Science. 2. Aufl. Chichester: Wiley, 2003.

Preißler, P.: Betriebswirtschaftliche Kennzahlen. Formeln, Aussagekraft, Sollwerte, Ermittlungsintervalle. Berlin: Oldenbourg, 2010.

Probst, G.; Raub, S.; Romhardt, K.: Wissen managen. Wie Unternehmen ihre wertvollste Ressource optimal nutzen. 5. Aufl. Wiesbaden: Gabler, 2010.

Pugixml: Light-weight C++ XML processing library. URL: https://pugixml.org/ (zuletzt geprüft am 13.1.2020).

Rabe, M.; Wenzel, S.; Spieckermann, S.: Verifikation und Validierung für die Simulation in Produktion und Logistik. Vorgehensmodelle und Techniken. Heidelberg, Berlin: Springer, 2008.

Reichmann, T.; Lachnit, L.: Kennzahlen. Planung, Steuerung und Kontrolle mit Hilfe von Kennzahlen. Zfbf, 1976, S. 705 – 723.

Robinson, S.: Simulation. The Practice of Model Development and Use. Chichester: Wiley, 2004.

Rose, O.; März, L.: Simulation. In: März, L.; Krug, W.; Rose, O.; Weigert, G. (Hrsg.): Simulation und Optimierung in Produktion und Logistik. Praxisorientierter Leitfaden mit Fallbeispielen. Berlin, Heidelberg: 2011, S. 13 – 20.

Schanz, G.: Sensoren – Fühler der Meßtechnik. Ein Handbuch der Meßwertaufnahme für den Praktiker. 2. Aufl. Heidelberg: Dr. Alfred Hüthig Verlag, 1988.

Scheidler, A.: Methode zur Erschließung von Wissen aus Datenmustern in Supply-Chain-Datenbanken. Göttingen: Cuvillier, 2017.

Schossig, D.: Mikrocontroller. Aufbau, Anwendung und Programmierung. München: te-wi-Verl, 1993.

Schott, G.: Kennzahlen, Instrument der Unternehmensführung. 6. Aufl. Wiesbaden: Vieweg, 1991.

Schröder, F.: Vorgehensweise zur Implementierung von logistischen Kennzahlensystemen im Umfeld der Automobilindustrie. München, 2017.

Schuh, G.; Stich, V.: Logistikmanagement. 2. Aufl. Berlin, Heidelberg: Springer, 2013.

Schulte, C.: Logistik. Wege zur Optimierung der Supply Chain. 7. Aufl. München: Vahlen, 2016.

SFML: Simple and Fast Multimedia Library. 2020. URL: https://www.sfml-dev.org/ (zuletzt geprüft am 13.1.2020).

Siegwart, H.; Reinecke, S.; Sander, S.: Kennzahlen für die Unternehmensführung. 7. Aufl. Stuttgart: Haupt Verlag, 2010.

Skoogh, A.; Johansson, B.: A methodology for input data management in discrete event simulation projects. In: Mason, S.; Hill, R.; Mönch, L.; Rose, O.; Jefferson, T.; Fowler, J. (Hrsg.): Proceedings of the 2008 Winter Simulation Conference. Miami: IEEE, 2008.

Skulschus, M.; Wiederstein, M.: XML: Standards und Technologien. 2. Aufl. Berlin: Comelio Medien, 2012.

Smith, J.: Survey on the Use of Simulation für Manufacturing System Design and Operation. Journal of Manufacturing Systems 22 (2003) 2, S. 157-161.

Staudinger GmbH: Automatisierungstechnik. 2020. URL: https://www.staudinger-est.de/ (zuletzt geprüft am 13.1.2020).

Stenner, F.: Der Einsatz von Kennzahlen zur strategischen Unternehmenssteuerung. In: Fischer, M. (Hrsg.): Handbuch Wertemanagement in Banken und Versicherungen. Wiesbaden: Gabler, 2004.

Syska, A.: Entwicklung einer Vorgehensweise zur Bildung von betriebsspezifischen Logistik-Kennzahlsystemen. Berlin, Heidelberg: Springer, 1990.

Ten Hompel, M.; Heidenblut, V. (Hrsg.): Taschenlexikon Logistik. Abkürzungen, Definition und Erläuterungen der Wichtigsten Begriffe aus Materialfluss und Logistik. 3. Aufl. Berlin, Heidelberg: 2011.

Ten Hompel, M.; Schmidt, T.; Nagel, L.: Materialflusssysteme. Förder- und Lagertechnik. 3. Aufl. Berlin, Heidelberg: Springer, 2007.

Ten Hompel, M.; Schmidt, T.; Dregger, J.: Materialflusssysteme. Förder- und Lagertechnik. 4. Aufl. Berlin, Heidelberg: Springer, 2018.

TGUI: Texus' Graphical User Interface. 2020. URL: https://tgui.eu/ (zuletzt geprüft am 13.1.2020).

Trybula, W.: Building simulation models without data. IEEE International Conference on Systems, Man and Cybernetics. Humans, Information and Technology 1 (1994), S. 209-214.

Vahrenkamp, R.; Kotzab, H.: Logistik. Management und Strategien. 7. Aufl. München: Oeldenbourg, 2012.

VDI: VDI-Richtlinie 2411 Blatt 1: Begriffe und Erläuterungen im Förderwesen. Berlin: Beuth, 1970.

VDI: VDI-Richtlinie 2690 Blatt 1: Material- und Datenfluß im Bereich von automatisierten Hochregallagern. Grundlagen. Berlin: Beuth, 1994.

VDI: VDI-Richtlinie 3633 Blatt 1: Simulation von Logistik-, Materialfluss- und Produktionssystemen. Grundlagen. Berlin: Beuth, 2014.

VDI: VDI-Richtlinie 3633 Blatt 3: Simulation von Logistik-, Materialfluss- und Produktionssystemen. Experimentplanung und -auswertung. Berlin: Beuth, 1997.

VDI: VDI-Richtlinie 4400 Blatt 1: Logistikkennzahlen für die Beschaffung. Berlin: Beuth, 2001.

VDI: VDI-Richtlinie 4480 Blatt 1: Durchsatz von automatischen Lagern mit gassengebundenen Regalbediensystemen. Berlin, Beuth: 1998.

Vonhoegen, H.: XML. Einstieg, Praxis, Referent. 9. Aufl. Bonn: Rheinwerk Verlag, 2018.

W3C: World Wide Web Consortium. 2020. URL: https://www.w3.org/ (zuletzt geprüft am 13.1.2020).

Wannemacher, M.; Halang, W.: GPS-basierte Zeitgeber: Realzeitsysteme werden endlich "echt" zeitfähig. Berlin, Heidelberg: 1994.

Weber, J.; Großklaus, A.; Kummer, S. Nippel, H.; Warnke, D.: Methodik zur Generierung von Logistik-Kennzahlen. In: Weber, J. (Hrsg.): Kennzahlen für die Logistik. Stuttgart: Schäffer-Poeschel Verlag, 1995.

Weber, J.; Schäffer, U.: Einführung in das Controlling. 11. Aufl. Stuttgart: Schäffer-Poeschel Verlag, 2006.

Wenzel, S.; Meyer, R.: Methoden des Datenmanagement. In: Kuhn, A.; Reinhardt, A.; Wiendahl, H.: Handbuch Simulationsanwendungen in Produktion und Logistik. Wiesbaden: Vieweg, 1993.

Will, T.: C++. Das umfassende Handbuch. Bonn: Rheinwerk Verlag, 2018.

Wolf, J.: Kennzahlensysteme als betriebliche Führungsinstrumente. München, 1977.

Wüst, K.: Mikroprozessortechnik. Grundlagen, Architekturen, Schaltungstechnik und Betrieb von Mikroprozessoren und Mikrocontrollern. 4. Aufl. Wiesbaden: Vieweg, 2011.

Wunsch, G.; Schreiber, H.: Stochastische Systeme. 4. Aufl. Berlin, Heidelberg: Springer, 2005.

Abkürzungsverzeichnis

A/D Wandler	Analog/Digital Wandler
CPU	Central Processing Unit
GUI	Graphical User Interface
I^2C	Inter-Integrated Circuit
I/O Ports	Input/Output Ports
mA	Milliampere
RFID	Radio-Frequency Identification
SFML	Simple and Fast Multimedia Library
SPI	Serial Periphal Interface
TGUI	Texus' Graphical User Interface
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
V	Volt
VDI	Verein Deutscher Ingenieure
XML	Extensible Markup Language-Format

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang A: RFID-Schreiber

```
#include <SPI.h>
#include <MFRC522.h>

const int RST_PIN = 9;
const int SS_PIN = 10;

// Create MFRC522 instance
MFRC522 mfrc522(SS_PIN, RST_PIN);

const int categoryToWriteInAscii = 65;
const int volumeToWriteInAscii = 50;

void setup()
{
 // Initialization
 Serial.begin(9600);
 SPI.begin();
 mfrc522.PCD_Init();
 Serial.println("RFID writer started");
}

void loop() {
 // Set all keys to zero
 MFRC522::MIFARE_Key key;
 for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

 Serial.println("Waiting for transponder ...");
 // Search for a transponder
 bool foundTransponder = false;
 while(!foundTransponder)
 {
  if (mfrc522.PICC_IsNewCardPresent()) {
   if (mfrc522.PICC_ReadCardSerial()) {
    foundTransponder = true;
   }
  }
 }

 Serial.println("Found a transponder, start writing ...");

 byte buffer[16];
 byte block;
 MFRC522::StatusCode status;
 byte len;

 // Initialize buffer with ' ' which equals 32 in ASCII
 for(int i  = 0; i < 16; i++)
  buffer[i] = 32;

 // Access block 4
 block = 4;
 status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key, &(mfrc522.uid));
 if (status != MFRC522::STATUS_OK) {
  Serial.print("Authentication failed: ");
  Serial.println(mfrc522.GetStatusCodeName(status));
```

```
  return;
 }

 // Write to block 4
 // Set value for category
 buffer[0] = categoryToWriteInAscii;
 status = mfrc522.MIFARE_Write(block, buffer, 16);
 if (status != MFRC522::STATUS_OK) {
  Serial.print("MIFARE_Write() failed: ");
  Serial.println(mfrc522.GetStatusCodeName(status));
  return;
 }
 else
 {
  Serial.print("Category wrote: ");
  Serial.println(char(buffer[0]));
 }

 // Reset buffer
 for(int i  = 0; i < 16; i++)
  buffer[i] = 32;

 // Access block 5
 block = 5;
 status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key, &(mfrc522.uid));
 if (status != MFRC522::STATUS_OK) {
  Serial.print("Authentication failed: ");
  Serial.println(mfrc522.GetStatusCodeName(status));
  return;
 }

 // Write to block 5
 // Set value for category
 buffer[0] = volumeToWriteInAscii;
 status = mfrc522.MIFARE_Write(block, buffer, 16);
 if (status != MFRC522::STATUS_OK) {
  Serial.print("MIFARE_Write() failed: ");
  Serial.println(mfrc522.GetStatusCodeName(status));
  return;
 }
 else
 {
  Serial.print("Volume wrote: ");
  Serial.println(char(buffer[0]));
 }

 Serial.println("Success!");
}
```

Anhang B: Steuerungssoftware

```
#include <SPI.h>
#include <MFRC522.h>
#include <DS3231.h>

const int RST_PIN = 5;
const int SS_PIN = 53;

// Create MFRC522 instance
MFRC522 mfrc522(SS_PIN, RST_PIN);

// Init the DS3231 using the hardware interface
DS3231  rtc(SDA, SCL);

// We cannot get the length of an array easily, so store its length separately
const int numberOfzSwitches = 3;
const int zSwitches[3] = {4, 3, 2};
const int numberOfySwitches = 10;
const int ySwitches[10] = {26, 32, 27, 33, 28, 34, 29, 35, 30, 36};
const int numberOfxSwitches = 10;
const int xSwitches[10] = {38, 39, 40, 41, 42, 43, 44, 45, 46, 47};

const int zMotorIN1 = 25;
const int zMotorIN2 = 24;
const int yMotorIN1 = 64;
const int yMotorIN2 = 65;
const int xMotorIN1 = 48;
const int xMotorIN2 = 49;

const int iPointLightBarrier = 19;
const int cPointLightBarrier = 18;
const int svLightBarrier = 9;

const int iPointRow = 2;
const int cPointRow = 7;

const int numberOfLeds = 6;
const int leds[6] = {6, 7, 8, 15, 16, 17};

enum Axis {x, y, z};

int currentLevel = -1;
int currentRow = -1;

struct item
{
  int category = 0;
  int volume = 0;
};

item storage[10][5];

void setMotorStatus(Axis axis, bool on, bool forward = true)
{
  int motorIN1;
  int motorIN2;
```

```
 if(axis == x)
 {
  motorIN1 = xMotorIN1;
  motorIN2 = xMotorIN2;
 }
 else if (axis == y)
 {
  motorIN1 = yMotorIN1;
  motorIN2 = yMotorIN2;
 }
 else
 {
  motorIN1 = zMotorIN1;
  motorIN2 = zMotorIN2;
 }

 if(on)
 {
  if(forward)
  {
   digitalWrite(motorIN1, LOW);
   digitalWrite(motorIN2, HIGH);
  }
  else
  {
   digitalWrite(motorIN2, LOW);
   digitalWrite(motorIN1, HIGH);
  }
 }
 else
 {
  digitalWrite(motorIN1, LOW);
  digitalWrite(motorIN2, LOW);
 }
 // There might be false signals due to high inductions
 delay(30);
}

void getFromIdentificationPoint()
{
 moveToRow(iPointRow);
 moveToLevel(0);

 // Move backwards
 setMotorStatus(z, true, false);
 while(!digitalRead(zSwitches[0]))
 {
  // wait
 }
 setMotorStatus(z, false);

 // wait a little bit
 delay(200);

 // Go a little higher
 setMotorStatus(y, true, true);
 delay(300);
```

```
  setMotorStatus(y, false);

  // home z
  setMotorStatus(z, true, true);
  while(!digitalRead(zSwitches[1]))
  {
    // wait
  }
  setMotorStatus(z, false);
}

void bringToControlPoint()
{
  moveToRow(cPointRow);
  moveToLevel(0);

  // Move upwards
  setMotorStatus(y, true, true);
  while(!digitalRead(ySwitches[1]))
  {
    // wait
  }
  setMotorStatus(y, false);
  // We reached the higher position. Now move z bachwards
  setMotorStatus(z, true, false);
  while(!digitalRead(zSwitches[0]))
  {
    // wait
  }
  setMotorStatus(z, false);
  // We can now drop the load
  setMotorStatus(y, true, false);
  while(!digitalRead(ySwitches[0]))
  {
    // wait
  }
  setMotorStatus(y, false);
  // Now, drive z back to center position
  setMotorStatus(z, true, true);
  while(!digitalRead(zSwitches[1]))
  {
    // wait
  }
  setMotorStatus(z, false);
}

void moveToRow(int targetRow)
{
  // Ranges from 0 to 9
  if(targetRow >= 0 && targetRow <= 9)
  {
    // Are we already at the requested row?
    if(currentRow != targetRow)
    {
      // Figure out if we have to move right or left
      if(targetRow > currentRow)
        setMotorStatus(x, true, true);
      else
```

```
      setMotorStatus(x, true, false);

    // Now wait until the targetLevel was reached
    while(!digitalRead(xSwitches[targetRow]))
    {
      // Wait and check for extrem switches
      if(targetRow < currentRow && digitalRead(xSwitches[0]))
      {
        // Stop the motor and get caught in an infinite loop
        setMotorStatus(x, false);
      }
      if(targetRow > currentRow && digitalRead(xSwitches[9]))
      {
        // Stop the motor and get caught in an infinite loop
        setMotorStatus(x, false);
      }
    }
    // We have reached the targetRow
    currentRow = targetRow;
    setMotorStatus(x, false);
   }
  }
}

void moveToLevel(int targetLevel)
{
  // Ranges from 0 to 4
  if(targetLevel >= 0 && targetLevel <= 4)
  {
    // Are we already at the requested level?
    if(currentLevel != targetLevel)
    {
      // Figure out if we have to move upwards or downwards
      if(targetLevel > currentLevel)
        setMotorStatus(y, true, true);
      else
        setMotorStatus(y, true, false);

      // Now wait until the targetLevel was reached
      while(!digitalRead(ySwitches[targetLevel * 2]))
      {
        // Wait and check for extrem switches
        if(targetLevel < currentLevel && digitalRead(ySwitches[0]))
        {
          // Stop the motor and get caught in an infinite loop
          setMotorStatus(y, false);
        }
        if(targetLevel > currentLevel && digitalRead(ySwitches[8]))
        {
          // Stop the motor and get caught in an infinite loop
          setMotorStatus(y, false);
        }
      }
      // We have reached the targetLevel
      currentLevel = targetLevel;
      setMotorStatus(y, false);
    }
  }
```

```
}

void homeY()
{
 if(!digitalRead(ySwitches[0]))
 {
  // Just move downwards
  setMotorStatus(y, true, false);
  while(!digitalRead(ySwitches[0]))
  {
   // wait
  }
 }
 setMotorStatus(y, false);
 currentLevel = 0;
}

void homeX()
{
 if(!digitalRead(xSwitches[0]))
 {
  // Just move left
  setMotorStatus(x, true, false);
  while(!digitalRead(xSwitches[0]))
  {
   // wait
  }
 }
 setMotorStatus(x, false);
 currentRow = 0;
}

void homeZ()
{
 // The home position of z is the center position, so zSwitches[1]

 // Are we already in the center position?
 if(!digitalRead(zSwitches[1]))
 {
  // We aren't. So try to move forward
  setMotorStatus(z, true, true);
  while(!digitalRead(zSwitches[1]))
  {
   if(digitalRead(zSwitches[2]))
   {
    // wrong direction, so move backwards
    setMotorStatus(z, true, false);
   }
   // Other modes shouldnt be possible
  }
 }
 setMotorStatus(z, false);
}

String unloadItem(int category)
{
 // Try to find an item for the category
 int i = 0;
```

```
 int level = 0;
 bool found = false;
 if(category == 1)
 {
   while(!found && i < 10)
   {
     level = 0;
     if(storage[i][level].category == 1)
       found = true;
     else
       i++;
   }
 }
 else if(category == 2)
 {
   level = 1;
   while(!found && i < 10)
   {
     if(storage[i][level].category == 2)
       found = true;
     else
       i++;
   }
 }
 else
 {
   // We can search 3 levels
   level = 3;
   while(!found && level < 4)
   {
     if(storage[i][level].category == 3)
       found = true;
     else
     {
       i++;
       if(i > 9)
       {
         i = 0;
         level++;
       }
     }
   }
 }

 // found something?
 if(!found)
   return "";

 // We found one, unload it
 int volume = storage[i][level].volume;
 unload(i, level);
 bringToControlPoint();
 return String(i) + "/" + String(level) + "/" + String(volume);
}

String stockNewItem(item newItem)
{
 // We will use a level accordingly to the category
```

```
  // We assume there is always at least one free space in a level
  int level;
  if(newItem.category == 1)
    level = 0;
  else if(newItem.category == 2)
    level = 1;
  else
    level = random (2, 5);

  // Use a random, empty row
  int row;
  bool found = false;
  while(!found)
  {
    row = random(0, 10);
    if(storage[row][level].category == 0)
      found = true;
  }

  // We found the target row and level, stock it there
  stock(row, level);

  // And save it to the storage
  storage[row][level] = newItem;
  return String(row) + "/" + String(level);
}

void stock(int targetRow, int targetLevel)
{
  // Move to the lower position of the target level
  moveToRow(targetRow);
  moveToLevel(targetLevel);
  // After we reached the position, move to the higher position of the target level
  // Move upwards
  setMotorStatus(y, true, true);
  while(!digitalRead(ySwitches[targetLevel * 2 + 1]))
  {
    // wait
  }
  setMotorStatus(y, false);
  // We reached the higher position. Now move z forward
  setMotorStatus(z, true, true);
  while(!digitalRead(zSwitches[2]))
  {
    // wait
  }
  setMotorStatus(z, false);
  // We are now inside the HBR, move to the lower position again
  setMotorStatus(y, true, false);
  while(!digitalRead(ySwitches[targetLevel * 2]))
  {
    // wait
  }
  setMotorStatus(y, false);
  // Now, drive z back to center position
  setMotorStatus(z, true, false);
  while(!digitalRead(zSwitches[1]))
  {
```

```
   // wait
  }
  setMotorStatus(z, false);
  // We have finished storing
}

void unload(int targetRow, int targetLevel)
{
  // Move to the lower position of the target level
  moveToRow(targetRow);
  moveToLevel(targetLevel);
  // After we reached the position, move z inside
  setMotorStatus(z, true, true);
  while(!digitalRead(zSwitches[2]))
  {
   // wait
  }
  setMotorStatus(z, false);
  // Move upwards
  setMotorStatus(y, true, true);
  while(!digitalRead(ySwitches[targetLevel * 2 + 1]))
  {
   // wait
  }
  setMotorStatus(y, false);
  // We reached the higher position. Now move z to center position
  setMotorStatus(z, true, false);
  while(!digitalRead(zSwitches[1]))
  {
   // wait
  }
  setMotorStatus(z, false);
  // Move back to lower position
  setMotorStatus(y, true, false);
  while(!digitalRead(ySwitches[targetLevel * 2]))
  {
   // wait
  }
  setMotorStatus(y, false);
  // We have finished unloading, clear the storage
  storage[targetRow][targetLevel].category = 0;
  storage[targetRow][targetLevel].volume = 0;
}

item identifyUntilFound()
{
  // Only proceed when an item was found
  bool found = false;
  item newItem;
  while(!found)
  {
   newItem = identify();
   if(newItem.category > 0 && newItem.volume > 0)
   {
    found = true;
   }
  }
  return newItem;
```

```
}

item identify()
{
 // Set all keys to zero
 MFRC522::MIFARE_Key key;
 for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

 // Reset IsNewCardPresent
 mfrc522.PICC_HaltA();
 mfrc522.PCD_StopCrypto1();

 // Search for a transponder
 bool foundTransponder = false;
 while(!foundTransponder)
 {
  if (mfrc522.PICC_IsNewCardPresent()) {
   if (mfrc522.PICC_ReadCardSerial()) {
    foundTransponder = true;
   }
  }
 }

 //some variables we need
 byte buffer[18];
 byte block;
 byte len;
 MFRC522::StatusCode status;

 // Get Category
 block = 4;
 len = 18;

 // Access block 4
 status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key, &(mfrc522.uid));
 if (status != MFRC522::STATUS_OK) {
  return;
 }

 // Read block 4
 status = mfrc522.MIFARE_Read(block, buffer, &len);
 if (status != MFRC522::STATUS_OK) {
  return;
 }

 // When using -64, we convert an ASCII A to a 1
 int receivedCategory = buffer[0] - 64;

 // Access block 5
 block = 5;
 status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key, &(mfrc522.uid));
 if (status != MFRC522::STATUS_OK) {
  return;
 }

 // Read block 5
 status = mfrc522.MIFARE_Read(block, buffer, &len);
 if (status != MFRC522::STATUS_OK) {
```

```
    return;
  }

  // When we use -48, an ASCII 1 will become a normal 1
  int receivedVolume = buffer[0] - 48;

  item newItem;
  newItem.category = receivedCategory;
  newItem.volume = receivedVolume;

  return newItem;
}

void returnToIdentificationPoint()
{
  moveToRow(iPointRow);
  moveToLevel(0);
}


void setup() {
  Serial.begin(9600);
  SPI.begin();
  mfrc522.PCD_Init();
  randomSeed(analogRead(0));
  rtc.begin();

  for(int i = 0; i < numberOfySwitches; i++)
    pinMode(ySwitches[i], INPUT);

  for(int i = 0; i < numberOfzSwitches; i++)
    pinMode(zSwitches[i], INPUT);

  for(int i = 0; i < numberOfxSwitches; i++)
    pinMode(xSwitches[i], INPUT);

  for(int i = 0; i < numberOfLeds; i++)
    pinMode(leds[i], OUTPUT);

  pinMode(svLightBarrier, INPUT);
  pinMode(iPointLightBarrier, INPUT);
  pinMode(cPointLightBarrier, INPUT);

  pinMode(xMotorIN1, OUTPUT);
  pinMode(xMotorIN2, OUTPUT);
  pinMode(yMotorIN1, OUTPUT);
  pinMode(yMotorIN2, OUTPUT);
  pinMode(zMotorIN1, OUTPUT);
  pinMode(zMotorIN2, OUTPUT);

  for(int i = 0; i < numberOfLeds; i++)
    digitalWrite(leds[i], HIGH);

  delay (1000);

  for(int i = 0; i < numberOfLeds; i++)
    digitalWrite(leds[i], LOW);
```

```
  delay (1000);

  performExperiment();
}

void loop()
{
  // Do nothing
  delay(100);
}

void setInitialStock()
{
  // There are already 8 inside
  storage[0][0].category = 1;
  storage[0][0].volume = 3;

  storage[8][0].category = 1;
  storage[8][0].volume = 2;

  storage[2][1].category = 2;
  storage[2][1].volume = 3;

  storage[3][1].category = 2;
  storage[3][1].volume = 4;

  storage[1][2].category = 3;
  storage[1][2].volume = 1;

  storage[9][3].category = 3;
  storage[9][3].volume = 3;

  storage[2][3].category = 3;
  storage[2][3].volume = 1;

  storage[3][4].category = 3;
  storage[3][4].volume = 5;
}

void performExperiment()
{
  // Home everything
  homeZ();
  homeY();
  homeX();

  setInitialStock();

  // --------- Step 1 ----------
  // Wait for the first item
  while(digitalRead(iPointLightBarrier))
  {
    // wait
  }
  // Send start package
  String startPackage = "4;";
  startPackage += rtc.getTimeStr();
  Serial.println(startPackage);
```

```
  einzelspielIn();

  delay(1000);
  // --------- Step 2 ----------
  einzelspielOut(1);

  delay(1000);
  // --------- Step 3 ----------
  doppelspiel(1);

  delay(1000);
  // --------- Step 4 ----------
  einzelspielIn();

  delay(1000);
  // --------- Step 5 ----------
  doppelspiel(1);

  delay(1000);
  // --------- Step 6 ----------
  einzelspielOut(1);

  delay(1000);
  // --------- Step 7 ----------
  einzelspielOut(3);

  delay(1000);
  // --------- Step 8 ----------
  doppelspiel(3);

  delay(1000);
  // --------- Step 9 ----------
  doppelspiel(2);

  delay(1000);
  // --------- Step 10 ---------
  einzelspielOut(2);

  // finished, send endpackage
  String endPackage = "5;";
  endPackage += rtc.getTimeStr();
  Serial.println(endPackage);
}

void einzelspielIn()
{
  item newItem = identifyUntilFound();
  // Perform actions
  // Wait for a new item
  while(digitalRead(iPointLightBarrier))
  {
    // wait
  }
  long timeStart = millis();
  getFromIdentificationPoint();
  String target = stockNewItem(newItem);
  returnToIdentificationPoint();
```

```
 long timeEnd = millis();
 // Send package
 String package;
 package = "1";
 package += ";";
 package += rtc.getTimeStr();
 package += ";";
 package += String(timeEnd-timeStart);
 package += ";";
 package += "E";
 package += ";";
 package += target;
 package += ";";
 package += String(newItem.category);
 package += ";";
 package += String(newItem.volume);
 Serial.println(package);
}

void einzelspielOut(int category)
{
 // Perform actions
 long timeStart = millis();
 String target = unloadItem(category);
 returnToIdentificationPoint();
 long timeEnd = millis();
 // Send package
 String package;
 if(target != "")
 {
  package = "1";
  package += ";";
  package += rtc.getTimeStr();
  package += ";";
  package += String(timeEnd-timeStart);
  package += ";";
  package += "A";
  package += ";";
  package += target.substring(0, 3);
  package += ";";
  package += String(category);
  package += ";";
  package += target.substring(4);
 }
 else
 {
  package = "3";
  package += ";";
  package += rtc.getTimeStr();
  package += ";";
  package += String(category);
 }
 Serial.println(package);
}

void doppelspiel(int category)
{
 item newItem = identifyUntilFound();
```

```
 // Perform actions
 // Wait for a new item
 while(digitalRead(iPointLightBarrier))
 {
  // wait
 }
 long timeStart = millis();
 newItem = identifyUntilFound();
 getFromIdentificationPoint();
 String targetIn = stockNewItem(newItem);
 // And unload
 String targetOut = unloadItem(category);
 returnToIdentificationPoint();
 long timeEnd = millis();
 // Send package
 String package;
 package = "2";
 package += ";";
 package += rtc.getTimeStr();
 package += ";";
 package += String(timeEnd-timeStart);
 package += ";";
 package += targetIn;
 package += ";";
 package += String(newItem.category);
 package += ";";
 package += String(newItem.volume);
 package += ";";
 if(targetOut != "")
 {
  package += targetOut.substring(0, 3);
  package += ";";
  package += String(category);
  package += ";";
  package += targetOut.substring(4);
 }
 else
 {
  package += "0";
  package += ";";
  package += "0";
  package += ";";
  package += "0";
 }
 Serial.println(package);
}
```

Anhang C: Programm zum Empfang von Datenpaketen und Umwandlung in ein XML-Dokument

```
// ---------------------- FILE main.cpp ----------------------
#include "Application.hpp"

int main() {
	Application application;
	application.run();
}
// ---------------------- FILE Application.hpp ----------------------
#pragma once

#include "TGUI\Gui.hpp"
#include "TGUI\Loading\Theme.hpp"
#include "TGUI\Widgets\Panel.hpp"
#include "TGUI\Widgets\Picture.hpp"
#include "TGUI\Widgets\Canvas.hpp"
#include "TGUI\Widgets\ComboBox.hpp"
#include "TGUI\Widgets\Button.hpp"
#include "TGUI\Widgets\ProgressBar.hpp"
#include "TGUI\Widgets\ChatBox.hpp"
#include "TGUI\Widgets\EditBox.hpp"

#include "SFML\Graphics\Image.hpp"

#include "pugixml.hpp"

// Forward Declaration
namespace sf
{
	class RenderWindow;
}

class Application
{
private:
	bool initializeXMLDocument();
	void initializeGui();
	void handleInput();
	void updateGui();
	void updateClock();
	void draw();

	void listen();
	void processDataPackage(std::string &actualIncomingData);
	void storeDataPackage(std::vector<std::string> &segments);

	void startOrStopListening();

	// Configuration
	const int mStandardPort = 4;

	// Logic
```

```
        std::string xmlDocumentName;
        pugi::xml_document xmlDocument;
        pugi::xml_node receivedDataNode;
        bool mIsListening = false;

        // Gui core
        std::unique_ptr<sf::RenderWindow> mWindow;
        std::unique_ptr<tgui::Gui> mGui;
        tgui::Theme::Ptr mTheme;

        // Widgets
        tgui::Panel::Ptr mPanelMain;

        tgui::Label::Ptr mLabelChatBoxReceivedData;
        tgui::ChatBox::Ptr mChatBoxReceivedData;

        tgui::Button::Ptr mButtonStartOrEndListening;
        tgui::Picture::Ptr mPictureStatus;

        tgui::Label::Ptr mLabelXMLName;
        tgui::EditBox::Ptr mEditBoxXMLName;

        tgui::Label::Ptr mLabelPort;
        tgui::EditBox::Ptr mEditBoxPort;

        // Clock
        sf::Clock mClockListening;
        sf::RectangleShape mClockRect;
        sf::Text mClockText;
        sf::Font font;
        tgui::Label::Ptr mLabelClock;

public:
        Application();
        ~Application();

        void run();
};
// ---------------------- FILE Application.hpp ----------------------
#include "Application.hpp"

#include "SFML/Graphics.hpp"
#include "SerialClass.h"


Application::Application()
{
}


Application::~Application()
{
}

void Application::initializeGui()
{
        // Initialize Gui core
        mWindow = std::make_unique<sf::RenderWindow>(sf::VideoMode(640, 480), "UART to XML");
```

```
mGui = std::make_unique<tgui::Gui>(*mWindow);
mTheme = tgui::Theme::create("themes/theme1.txt");

// Initialize main panel
mPanelMain = mTheme->load("Panel");
mPanelMain->setPosition(0, 0);
mPanelMain->setSize(mWindow->getSize().x, mWindow->getSize().y);
mPanelMain->setBackgroundColor(sf::Color(230, 210, 110));
mGui->add(mPanelMain);

// Initialize Widgets
mLabelChatBoxReceivedData = mTheme->load("Label");
mLabelChatBoxReceivedData->setSize(400, 30);
mLabelChatBoxReceivedData->setPosition(230, 6);
mLabelChatBoxReceivedData->setTextSize(26);
mLabelChatBoxReceivedData->setTextColor(sf::Color::Black);
mLabelChatBoxReceivedData->setHorizontalAlignment(tgui::Label::HorizontalAlignment::Left);
mLabelChatBoxReceivedData->setText("Log:");
mPanelMain->add(mLabelChatBoxReceivedData);

mChatBoxReceivedData = tgui::ChatBox::create();
mChatBoxReceivedData->setTextSize(12);
mChatBoxReceivedData->setPosition(230, 40);
mChatBoxReceivedData->setSize(400, 430);
mPanelMain->add(mChatBoxReceivedData);


//mChatBoxReceivedData->addLine("Received: 10/20/390/10");
//mChatBoxReceivedData->addLine("Error occured", sf::Color::Red);

mLabelXMLName = mTheme->load("Label");
mLabelXMLName->setSize(160, 24);
mLabelXMLName->setPosition(10, 6);
mLabelXMLName->setTextSize(20);
mLabelXMLName->setTextColor(sf::Color::Black);
mLabelXMLName->setHorizontalAlignment(tgui::Label::HorizontalAlignment::Left);
mLabelXMLName->setText("XML File Name:");
mPanelMain->add(mLabelXMLName);

mEditBoxXMLName = mTheme->load("EditBox");
mEditBoxXMLName->setSize(160, 30);
mEditBoxXMLName->setPosition(10, 40);
mEditBoxXMLName->setTextSize(26);
mPanelMain->add(mEditBoxXMLName);

mButtonStartOrEndListening = mTheme->load("Button");
mButtonStartOrEndListening->setSize(140, 40);
mButtonStartOrEndListening->setPosition(20, 300);
mButtonStartOrEndListening->setText("Start Listening");
mButtonStartOrEndListening->connect("pressed", &Application::startOrStopListening, this);
mPanelMain->add(mButtonStartOrEndListening);

mLabelPort = mTheme->load("Label");
mLabelPort->setSize(80, 24);
mLabelPort->setPosition(170, 276);
mLabelPort->setTextSize(20);
mLabelPort->setTextColor(sf::Color::Black);
mLabelPort->setHorizontalAlignment(tgui::Label::HorizontalAlignment::Left);
```

```
	mLabelPort->setText("Port:");
	mPanelMain->add(mLabelPort);

	mEditBoxPort = mTheme->load("EditBox");
	mEditBoxPort->setSize(40, 30);
	mEditBoxPort->setPosition(170, 306);
	mEditBoxPort->setTextSize(20);
	mEditBoxPort->setText(std::to_string(mStandardPort));
	mPanelMain->add(mEditBoxPort);

	mPictureStatus = tgui::Picture::create();
	mPictureStatus->setPosition(90, 360);
	mPictureStatus->setTexture("themes/status_idle.png");
	mPictureStatus->setSize(60, 60);
	mPanelMain->add(mPictureStatus);

	// Clock
	mClockRect.setSize(sf::Vector2f(70, 18));
	mClockRect.setPosition(sf::Vector2f(125, 440));
	mClockRect.setFillColor(sf::Color(255, 255, 255, 120));

	font.loadFromFile("themes/Sansation.ttf");
	mClockText.setFont(font);
	mClockText.setCharacterSize(14);
	mClockText.setPosition(sf::Vector2f(126, 440));
	mClockText.setFillColor(sf::Color::Black);

	mLabelClock = mTheme->load("Label");
	mLabelClock->setSize(160, 20);
	mLabelClock->setPosition(20, 440);
	mLabelClock->setTextSize(16);
	mLabelClock->setTextColor(sf::Color::Black);
	mLabelClock->setHorizontalAlignment(tgui::Label::HorizontalAlignment::Left);
	mLabelClock->setText("Listening for:");
	mPanelMain->add(mLabelClock);
}

void Application::run()
{
	initializeGui();

	// Prepare time measurement
	sf::Clock clock;
	sf::Time elapsedTime = sf::Time::Zero;

	// Main loop
	while (mWindow->isOpen())
	{
		// Handle time
		elapsedTime = clock.restart();

		updateClock();
		handleInput();
		updateGui();

		if (mIsListening)
			listen();
	}
```

```
}

void Application::handleInput()
{
	sf::Event event;
	while (mWindow->pollEvent(event))
	{
		if (event.type == sf::Event::Closed)
			mWindow->close();

		mGui->handleEvent(event);
	}
}

void Application::updateGui()
{
	mWindow->clear();
	mGui->draw();
	draw();
	mWindow->display();
}

void Application::startOrStopListening()
{
	if (!mIsListening)
	{
		// Start listening
		mChatBoxReceivedData->addLine("Started Listening", sf::Color(51, 204, 51));
		mButtonStartOrEndListening->setText("Stop Listening");
		mClockListening.restart();
		mPictureStatus->setTexture("themes/status_listening.png");
	}
	else
	{
		// Stop listening
		mChatBoxReceivedData->addLine("Finished Listening", sf::Color(51, 204, 51));
		mButtonStartOrEndListening->setText("Start Listening");
		mPictureStatus->setTexture("themes/status_finished.png");
	}
	mIsListening = !mIsListening;
}

void Application::updateClock()
{
	if (mIsListening)
	{
		sf::Time elapsedTime = mClockListening.getElapsedTime();

		int hours = (int)elapsedTime.asSeconds() / 3600;
		int minutes = (int)elapsedTime.asSeconds() % 3600 / 60;
		int seconds = (int)elapsedTime.asSeconds() % 3600 % 60;

		// Create substrings
		std::string secondsString;
		if (seconds < 10) {
			secondsString = "0" + std::to_string(seconds);
		}
		else {
```

```
                secondsString = std::to_string(seconds);
            }
            std::string minutesString;
            if (minutes < 10) {
                minutesString = "0" + std::to_string(minutes);
            }
            else {
                minutesString = std::to_string(minutes);
            }
            std::string hoursString;
            if (hours < 10) {
                hoursString = "0" + std::to_string(hours);
            }
            else {
                hoursString = std::to_string(hours);
            }
            mClockText.setString(hoursString + ":" + minutesString + ":" + secondsString);
        }
        else
        {
            mClockText.setString("not started");
        }
}

void Application::draw()
{
    mWindow->draw(mClockRect);
    mWindow->draw(mClockText);
}

void Application::listen()
{
    std::string portString = mEditBoxPort->getText().toAnsiString();

    Serial* SP = new Serial(("\\\\.\\COM" + portString).c_str());

    if (SP->IsConnected())
    {
        mChatBoxReceivedData->addLine("Serial Began on Port " + portString, sf::Color(51, 204, 51));

        // Check XML
        if (!initializeXMLDocument())
        {
            mIsListening = false;
            mPictureStatus->setTexture("themes/status_failure.png");
            mButtonStartOrEndListening->setText("Start Listening");
        }

        std::string currentMessage = "";
        while (SP->IsConnected() && mIsListening)
        {
            char incomingData[256] = "";
            int dataLength = 255;
            int readResult = 0;

            // Read data and get read bytes
            readResult = SP->ReadData(incomingData, dataLength);
            //incomingData[readResult] = 0;
```

```
                        if (readResult != 0)
                        {
                                // There is data available
                                // Find actually written data to the buffer by using the length given in readResult
                                for (int readByte = 0; readByte < readResult; readByte++)
                                        currentMessage += incomingData[readByte];

                                // Is the message complete?
                                if (incomingData[readResult - 1] == 10)
                                {
                                        processDataPackage(currentMessage);
                                        // We can reset the currentMessage as it was processed
                                        currentMessage = "";
                                }
                        }

                        // Call updates and handle inputs, otherwise there are no updates and chances to stop the listening process
                        handleInput();
                        updateGui();
                }

                delete SP;

                mChatBoxReceivedData->addLine("Serial Ended", sf::Color(51, 204, 51));

                xmlDocument.save_file(xmlDocumentName.c_str());
                mChatBoxReceivedData->addLine("Saved XML file", sf::Color(51, 204, 51));
        }
        else
        {
                mChatBoxReceivedData->addLine("No Serial on Port " + portString + " available", sf::Color::Red);
                mButtonStartOrEndListening->setText("Start Listening");
                mPictureStatus->setTexture("themes/status_failure.png");
                mIsListening = false;
        }
}

void Application::processDataPackage(std::string &actualIncomingData)
{
        // At first, segment the string
        std::stringstream ss(actualIncomingData);
        std::string segment;
        std::vector<std::string> seglist;

        while (std::getline(ss, segment, ';'))
                seglist.push_back(segment);

        // Get rid of the last two characters as they are the carriage return
        seglist.back().pop_back();
        seglist.back().pop_back();

        // Prepare chatBox string
        std::string chatBoxString;

        // Now, we can identify the package
        int dataPackageType = std::stoi(*seglist.begin());
        chatBoxString = "Received package type '" + *seglist.begin() + "', values: ";
```

```
		// Print all values
		if (seglist.size() > 1)
		{
			for (std::vector<std::string>::iterator values = seglist.begin() + 1; values != seglist.end(); values++)
			{
				chatBoxString += "";
				chatBoxString += *values;
				chatBoxString += "";
				if(values != seglist.end() - 1)
					chatBoxString += ", ";
			}

			// Store it in xml
			storeDataPackage(seglist);
		}
		else
		{
			mChatBoxReceivedData->addLine("Package has not enough data", sf::Color::Red);
			return;
		}

		// Print chatBox string
		mChatBoxReceivedData->addLine(chatBoxString, sf::Color::Black);
}

bool Application::initializeXMLDocument()
{
		xmlDocumentName = "xmlDocuments/" + mEditBoxXMLName->getText().toAnsiString() + ".xml";

		// Try to load the specified XML, only proceed when the name is not already existant
		pugi::xml_parse_result result = xmlDocument.load_file(xmlDocumentName.c_str());

		if (result.status == pugi::xml_parse_status::status_ok)
		{
			// There is already a file with this name
			mChatBoxReceivedData->addLine("XML filename already in use", sf::Color::Red);
			return false;
		}

		// Otherwise, create a new document

		// Generate XML declaration
		auto declarationNode = xmlDocument.append_child(pugi::node_declaration);
		declarationNode.append_attribute("version") = "1.0";
		declarationNode.append_attribute("encoding") = "UTF-8";


		// Create base node
		pugi::xml_node root = xmlDocument.append_child("experimentData");

		// Add the received data node
		receivedDataNode = root.append_child("receivedData");
		return true;
}

void Application::storeDataPackage(std::vector<std::string> & segments)
{
		// Add a new received package
```

```
pugi::xml_node dataPackage = receivedDataNode.append_child("dataPackage");

// Add identifier
pugi::xml_node identifier = dataPackage.append_child("identifier");
identifier.append_child(pugi::node_pcdata).set_value(segments[0].c_str());

// Add timestamp
pugi::xml_node timestamp = dataPackage.append_child("timestamp");
timestamp.append_child(pugi::node_pcdata).set_value(segments[1].c_str());

// Now, the rest depends on the identifier
int id = std::stoi(segments[0]);
switch (id)
{
	case 1:
	{
		// Add duration
		pugi::xml_node duration = dataPackage.append_child("duration");
		duration.append_child(pugi::node_pcdata).set_value(segments[2].c_str());
		// Add inOrOur
		pugi::xml_node inOrOut = dataPackage.append_child("inOrOut");
		inOrOut.append_child(pugi::node_pcdata).set_value(segments[3].c_str());
		// Add rackBay
		pugi::xml_node rackBay = dataPackage.append_child("rackBay");
		rackBay.append_child(pugi::node_pcdata).set_value(segments[4].c_str());
		// Add category
		pugi::xml_node category = dataPackage.append_child("category");
		category.append_child(pugi::node_pcdata).set_value(segments[5].c_str());
		// Add volume
		pugi::xml_node volume = dataPackage.append_child("volume");
		volume.append_child(pugi::node_pcdata).set_value(segments[6].c_str());
	}
	break;

	case 2:
	{
		// Add duration
		pugi::xml_node duration = dataPackage.append_child("duration");
		duration.append_child(pugi::node_pcdata).set_value(segments[2].c_str());
		// Add rackBay
		pugi::xml_node rackBayIn = dataPackage.append_child("rackBayIn");
		rackBayIn.append_child(pugi::node_pcdata).set_value(segments[3].c_str());
		// Add category
		pugi::xml_node categoryIn = dataPackage.append_child("categoryIn");
		categoryIn.append_child(pugi::node_pcdata).set_value(segments[4].c_str());
		// Add volume
		pugi::xml_node volumeIn = dataPackage.append_child("volumeIn");
		volumeIn.append_child(pugi::node_pcdata).set_value(segments[5].c_str());
		// Add rachBay
		pugi::xml_node rackBayOut = dataPackage.append_child("rackBayOut");
		rackBayOut.append_child(pugi::node_pcdata).set_value(segments[6].c_str());
		// Add category
		pugi::xml_node categoryOut = dataPackage.append_child("categoryOut");
		categoryOut.append_child(pugi::node_pcdata).set_value(segments[7].c_str());
		// Add volume
		pugi::xml_node volumeOut = dataPackage.append_child("volumeOut");
		volumeOut.append_child(pugi::node_pcdata).set_value(segments[8].c_str());
	}
```

```
                break;

                case 3:
                {
                        // Add category
                        pugi::xml_node category = dataPackage.append_child("category");
                        category.append_child(pugi::node_pcdata).set_value(segments[2].c_str());
                }
                break;

                case 4:
                {
                        // No additional information needed
                }
                break;

                case 5:
                {
                        // No additional information needed
                }
                break;
        }
}
```

Anhang D: Erhaltenes XML-Dokument des Experiments

```
<?xml version="1.0" encoding="UTF-8"?>
<experimentData>
	<frameIndicators>
	</frameIndicators>
	<receivedData>
		<dataPackage>
			<identifier>4</identifier>
			<timestamp>01:42:18</timestamp>
		</dataPackage>
		<dataPackage>
			<identifier>1</identifier>
			<timestamp>01:42:45</timestamp>
			<duration>27259</duration>
			<inOrOut>E</inOrOut>
			<rackBay>7/4</rackBay>
			<category>3</category>
			<volume>3</volume>
		</dataPackage>
		<dataPackage>
			<identifier>1</identifier>
			<timestamp>01:43:04</timestamp>
			<duration>18124</duration>
			<inOrOut>A</inOrOut>
			<rackBay>0/0</rackBay>
			<category>1</category>
			<volume>3</volume>
		</dataPackage>
		<dataPackage>
			<identifier>2</identifier>
			<timestamp>01:43:42</timestamp>
			<duration>36277</duration>
			<rackBayIn>9/1</rackBayIn>
			<categoryIn>2</categoryIn>
			<volumeIn>2</volumeIn>
			<rackBayOut>8/0</rackBayOut>
			<categoryOut>1</categoryOut>
			<volumeOut>2</volumeOut>
		</dataPackage>
		<dataPackage>
			<identifier>1</identifier>
			<timestamp>01:43:55</timestamp>
			<duration>9981</duration>
			<inOrOut>E</inOrOut>
			<rackBay>3/0</rackBay>
			<category>1</category>
			<volume>4</volume>
		</dataPackage>
		<dataPackage>
			<identifier>2</identifier>
			<timestamp>01:44:40</timestamp>
			<duration>42140</duration>
			<rackBayIn>6/4</rackBayIn>
			<categoryIn>3</categoryIn>
```

<volumeIn>4</volumeIn>
<rackBayOut>3/0</rackBayOut>
<categoryOut>1</categoryOut>
<volumeOut>4</volumeOut>
</dataPackage>
<dataPackage>
<identifier>3</identifier>
<timestamp>01:44:41</timestamp>
<category>1</category>
</dataPackage>
<dataPackage>
<identifier>1</identifier>
<timestamp>01:45:07</timestamp>
<duration>24491</duration>
<inOrOut>A</inOrOut>
<rackBay>2/3</rackBay>
<category>3</category>
<volume>1</volume>
</dataPackage>
<dataPackage>
<identifier>2</identifier>
<timestamp>01:45:48</timestamp>
<duration>38755</duration>
<rackBayIn>4/0</rackBayIn>
<categoryIn>1</categoryIn>
<volumeIn>1</volumeIn>
<rackBayOut>9/3</rackBayOut>
<categoryOut>3</categoryOut>
<volumeOut>3</volumeOut>
</dataPackage>
<dataPackage>
<identifier>2</identifier>
<timestamp>01:46:20</timestamp>
<duration>29912</duration>
<rackBayIn>1/0</rackBayIn>
<categoryIn>1</categoryIn>
<volumeIn>4</volumeIn>
<rackBayOut>2/1</rackBayOut>
<categoryOut>2</categoryOut>
<volumeOut>3</volumeOut>
</dataPackage>
<dataPackage>
<identifier>1</identifier>
<timestamp>01:46:40</timestamp>
<duration>18561</duration>
<inOrOut>A</inOrOut>
<rackBay>3/1</rackBay>
<category>2</category>
<volume>5</volume>
</dataPackage>
</receivedData>
</experimentData>