Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Ali Varal

Konzeptionierung und Aufbau eines Testfeldes zur dezentralen Steuerung eines fahrerlosen Transportsystems

Fakultät Technik und Informatik
Department Maschinenbau und Produktion

Faculty of Engineering and Computer Science
Department of Mechanical Engineering and Production Management

Ali Varal

Konzeptionierung und Aufbau eines Testfeldes zur dezentralen Steuerung eines fahrerlosen Transportsystems

Masterarbeit eingereicht im Rahmen der Masterprüfung

im Studiengang Produktionstechnik und -management
am Department Maschinenbau und Produktion
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Erstprüfer: Prof. Dr. Henner Gärtner
Zweitprüfer: Prof. Dr.-Ing. Randolf Isenberg

Abgabedatum: 16.07.2018

Zusammenfassung

Ali Varal

Thema der Masterarbeit

Konzeptionierung und Aufbau eines Testfeldes zur dezentralen Steuerung eines fahrerlosen Transportsystems

Stichworte

Dezentrale Produktionssteuerung, Industrie 4.0, Fahrerlose Transportsysteme, Arduino

Kurzzusammenfassung

In dieser Arbeit wird ein einzigartiges Konzept für eine dezentrale Produktionssteuerung mit fahrerlosen Transportsystemen aufgestellt. Der Grundgedanke dabei ist es, dass die Auftragssteuerung nicht mehr durch eine übergeordnete Instanz in Form einer zentralen Produktionssteuerung, sondern direkt am Ort des Geschehens durch einzelne dezentrale Systeme erfolgt. Die auftragssteuernde Instanz sollen die Kundenaufträge selbst in Form von intelligenten Transportkisten darstellen, welche sich selbststeuernd durch die Produktion für die Erfüllung des Auftrages koordinieren. Weiterhin soll das Konzept in einer Modellierung basierend auf dem Arduino-System dargestellt werden, in der sich die fahrerlosen Transportfahrzeuge (FTF) um den Transport der Kisten kümmern. Die FTF sollen dabei gegenseitig um den Erhalt der Transport-aufträge konkurrieren und nicht mehr durch ein zentrales System mit Aufträgen versorgt werden.

Ali Varal

Title of the paper

Conceptual design and construction of a test field for decentralized control of an automated guided vehicle system

Keywords

Decentralized production control, Industry 4.0, Automated guided vehicle systems, Arduino

Abstract

In this work, a unique concept for decentralized production control with driverless transport systems is developed. The basic idea here is that order control is no longer carried out by a superordinate instance in the form of a central production control system, but directly at the location of the event by individual decentralized systems. The order controlling authority should represent the customer orders themselves in the form of intelligent transport boxes, which are coordinated autonomously by the production for the fulfilment of the order. Furthermore, the concept is to be presented in a model based on the Arduino system, in which the automated guided (AGV) vehicles take care of the transport of the boxes. The AGV are to compete with each other for the receipt of transport orders and are no longer to be supplied with orders by a central system.

Inhaltsverzeichnis

Abkürzungsverzeichnis

µC	Mikrocontroller
FTF	Fahrerlose Transportfahrzeuge
FTS	Fahrerlose Transportsysteme
GPS	Global Positioning System
HP	Hohe Priorität
IDE	Integrierte Entwicklungsumgebung
IoT	Internet-of-Things
IPH	Institut für Integrierte Produktion Hannover
IR	Infrarot
MRP	Manufacturing Ressources Planning
NA	Neuausrichtung
NP	Niedrige Priorität
NV	Nicht verfügbar
PP	Produktionsplanung
PPS	Produktionsplanung und -steuerung
QoS	Quality-of-Service
RFID	Radio Frequency Identification
ZS	Zeitslot

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Ausganssituation/Problemstellung

Durch den internationalen Wettbewerb und die zunehmende Individualisierung der Kundenwünsche stoßen die zentralen Steuerungssysteme an ihre Grenzen und die Unternehmen sind dazu gezwungen alternative Lösungen zu finden. Die Produkte werden komplexer und die Variantenvielfalt größer. All dieser Veränderungen wirken sich auf die Erreichung der logistischen Ziele aus, wie die Einhaltung der Liefertreue. Außerdem sind in der heutigen Produktion mehr externe Dienstleister involviert als zuvor, welches die gesamte Steuerung deutlich erschwert, denn die Produktion geht weit über das eigene Umfeld hinaus und die unternehmensübergreifende Zusammenarbeit in Lieferketten wird erheblich bedeutsamer.

Die dezentrale Produktionssteuerung ist ein Kernelement des Zukunftsschaubilds Industrie 4.0 - ein selbstgesteuertes Produktionssystem. Das Ziel dabei ist es eine autarke Produktion, und zwar unabhängig einer zentralen Instanz, zu ermöglichen, sodass beispielsweise Maschinen Ihren Nachschub an Material aus dem Lager selbst bestellen oder eigenständig Transportaufträge auslösen. Durch die Einführung der dezentralen Steuerung werden auch kleinere Produktionsbetriebe in der Lage sein, selbst Fertigungsmengen von einem Stück wirtschaftlich herzustellen und somit weiterhin eine Position im Markt aufrechterhalten können.

1.2 Zielsetzung und Abgrenzung

In dieser Arbeit wird eine dezentrale Steuerung in Verbindung mit einem fahrerlosen Transportsystem konzipiert, worin anhand eines zentralen Elementes die Produktion selbstständig gesteuert wird. Grundlegend baut diese Arbeit auf der Erfüllung von drei Teilzielen auf, die nachfolgend schwerpunktmäßig erläutert werden.

Teilziel 1: Konzeptionierung eines Systems zur dezentralen Steuerung von fahrerlosen Transportsystemen mit zugehörigem Produktionsszenario

Ein ganzheitliches Konzept für die dezentrale Steuerung von fahrerlosen Transportsystemen innerhalb einer Produktion bedingt vielerlei Anforderungen, die festgelegt und

aufeinander abgestimmt realisiert werden müssen. Primär geht es dabei um die Entscheidungs- und Steuerungslogik, wonach alle Handlungen der Entitäten im System geregelt werden müssen. Das Konzept soll eine funktionierende Steuerungslogik auf dezentraler Struktur aufweisen und die gesamte Produktionssteuerung ohne eine zentrale Instanz leiten können. Demnach werden die Entscheidungen direkt am Ort des Geschehens gefallen und nicht von außerhalb gesteuert.

Für fahrerlose Transportfahrzeuge (FTF) ist die Routenplanung ein essentieller Bestandteil, denn sie müssen sich „fahrerlos“ und unabhängig einer Kontrolle aus der Umwelt bewegen können. Um die eigenständige Bewegung möglich zu machen, benötigen diese FTF Algorithmen, womit sie ihre Fahrroute ausrechnen. Daher ist ein geeigneter Algorithmus auszusuchen und in das Konzept zu implementieren.

Zu dem Konzept gehört ebenso ein Testfeld zu entwerfen, auf dem die Ergebnisse erprobt werden müssen. Hierzu wird ein Produktionsszenario aufgestellt, in der ein fiktiver Teil einer Produktionsstätte abzubilden ist.

Teilziel 2: Technische Realisierung einer Teilentwicklung aus dem Konzept und die Validierung mittels eines kleinen Testszenarios

Neben dem Konzept ist im Zuge dieser Arbeit als praktische Leistung ein fahrerloses Transportfahrzeug zu entwickeln, welches mit einem Fahrzeugroboter basierend auf dem Arduino-System benötigte Funktionen zu erfüllen gilt. Der zu erfüllende Funktionsumfang wurde hierfür in den ersten Projektgesprächen mit den Teilnehmern gemeinsam abgestimmt (vgl. 5.2). Die Entwicklung des FTF und die Erfüllung der Anforderungen müssen anschließend mittels eines vereinfachten Testfeldes validiert werden.

Teilziel 3: Ausblick für das weitergehende Projekt mit geplantem Entwicklungsverlauf und zu treffenden Maßnahmen

Um das aufgestellte Konzept vollständig zu realisieren, werden weitere Schritte nötig werden, da im Rahmen dieser Arbeit nur eine Teilentwicklung durchgeführt wird. Dazu sollen die offenen Punkte für die Realisierung mit möglichen Ansätzen vorgestellt werden, woraus die Möglichkeit zur Entstehung weiterer studentischen Projekte resultiert.

1.3 Vorgehensweise

Grundlegend ist die Arbeit in drei Teile zu unterteilen. Zu Beginn werden Grundlagen zu den Themen vermittelt, die in dieser Arbeit zur Verwendung kommen werden. Da es eine sehr vielseitige Arbeit mit vielen unterschiedlichen Themen- und Technologiebereichen ist, gestaltet sich der Grundlagenteil sehr gemischt. Nach dem Abschluss der Grundlagenbeschreibung folgt ein Einstieg zu dem Themenfeld der Produktionssteuerung und die Betrachtung des Trends zur Dezentralisation der Produktion. Die Frage dabei ist, die zu untersuchen gilt, ob der Trend einer dezentralen Produktionssteuerung seine Berechtigung hat oder nicht. Abzuschließen ist der Abschnitt mit einer kritischen Betrachtung über den Vergleich zwischen der altbewährten zentralen und der angestrebten dezentralen Steuerung.

Nach dem oben beschriebenen Teil wird der theoretische Teil der Arbeit abgeschlossen und ist fortzuführen mit dem konzeptionellen Teil. Zunächst ist ein Fallbeispiel mit einem fiktiven Produktionsabschnitt aufzustellen, in der ein Bezug zu der Aufgabenstellung dieser Arbeit hergestellt werden muss. Anschließend werden die Komponenten des Konzeptes mit ihren einzelnen Funktionen beschrieben. Darauffolgend werden notwendige Modelle zur Auftragssteuerung und die Teilhabe durch die vorgestellten Komponenten erläutert. In dem darauffolgenden Abschnitt findet schließlich eine Teilvalidierung hinsichtlich des zu entwickelnden fahrerlosen Transportfahrzeugs statt.

Als dritter und letzter Teil folgt eine kritische Würdigung der erreichten Ergebnisse. Dafür werden Fragen aufgestellt, die zu beantworten sind, womit die Ergebnisse der Arbeit beschrieben werden. Um die Realisierung des gesamten Konzeptes voranzutreiben, werden zudem weitere Handlungsempfehlungen vorgeschlagen. Zuletzt wird die Arbeit mit einer kurzen Zusammenfassung und einem Ausblick auf das weitere Geschehen abgeschlossen.

Für Abbildungen, welche aus fremden Quellen entnommen sind, werden stets Quellverweise in Form von Fußnoten erstellt. Demnach sind alle Abbildungen ohne eine Fußnote selbst erstellt und demnach ohne Quellverweis. Dies gilt gleichermaßen für Tabellen oder andere Arten von Darstellungen. Zur Zitation und die Erstellung des Literaturverzeichnisses wird die frei verfügbare Software „Zotero“ genutzt. Sowohl wörtliche, als auch indirekte Zitate werden unmittelbar nach der Textstelle in Klammern angegeben. In den Abschnitten mit Programmierergebnissen, werden Funktions- und Variablenbezeichnungen fett und in Anführungszeichen dargestellt.

1.4 Abgrenzung

Das Konzept stellt keine Behauptung auf eine gesamtheitlich perfekte Produktion zu modellieren. Primär liegt der Fokus auf dem Erreichen einer funktionierenden Steuerungslogik für eine dezentrale Steuerung der Aufträge. Hierfür wird anlehnend auf ein fiktives Beispiel ein Produktionsszenario entworfen, in der es vor allem um die Fertigung von Stückgut mit realitätsfernen Prozessparametern in einer einstufigen Fertigung geht. Logistische Kenngrößen wie Durchlaufzeiten oder Transportzeiten werden hierbei nicht erfasst. Daher wird es in dieser Form nicht für den Einsatz an einer realen Produktionsstätte gerichtet werde, sondern lediglich um eine dezentrale Steuerung für ein fahrerloses Transportsystem zu demonstrieren und neue Erkenntnisse in der Forschungsarbeit zu erlangen. Ziel ist es eine Modelldarstellung des gesamten Systems aufbauend auf dem Entwicklungssystem Arduino aufzubauen. Daher können auch die Funktionsumfänge und -vorstellungen mit keiner realen Produktion verglichen werden. So kann ein Werkstück an einem Arbeitsplatz über die gesamte Bearbeitungsdauer nur verweilen, ohne dass irgendeine Aktion erfolgt.

Das Konzept wird zudem in dieser Arbeit nicht vollständig realisiert, weshalb nur die Teilentwicklung des fahrerlosen Transportfahrzeuges validiert werden kann. Das FTF dient dabei als Grundlage, welches im weiteren Verlauf der Konzeptrealisierung als Prototyp dienen wird.

2 Theoretische Grundlagen

In diesem Abschnitt werden theoretische Grundlagen zu den Themen behandelt, die während dieser Arbeit zur Verwendung gekommen sind, um ein besseres Verständnis im Hauptteil zu ermöglichen. Die Themen sind aufgrund der Vielfältigkeit der Arbeit nicht zusammenhängend und voneinander unabhängig zu betrachten. Zudem hat die Reihenfolge der behandelten Themen keine Relevanz zu dem Vorgehen im Konzeptteil.

2.1 Fahrerlose Transportsysteme

Definiert werden fahrerlose Transportsysteme (FTS) als flurgebundene Fördersysteme, in denen die enthaltenen Fahrzeuge autonom geführt werden. Gerade im Bereich des Transportes von Stückgütern erhalten fahrerlose Transportsysteme eine bedeutsamen Wert in der Industrie, mit denen die Erfüllung der logistischen Ziele im Bereich der Transportsysteme gerecht werden kann (Niemann, Baum, Fricke, & Overmeyer, 2006).

Die Bestandteile eines fahrerlosen Transportsystems sind („VDI-Richtlinie: VDI 2510 Fahrerlose Transportsysteme (FTS)“, 2005, S. 7):

- Fahrerlose Transportfahrzeuge
- FTS-Leitsteuerung
- Mittel zur Standortbestimmung und Lageerfassung
- Mittel zur Datenübertragung (zwischen Fahrzeugen und System)
- Infrastruktur und periphere Einrichtungen

Lediglich der Bereich der Infrastruktur und der peripheren Einrichtungen wird im Rahmen dieser Arbeit nicht behandelt. Zu allen weiteren Bereichen folgt im aufzustellenden Konzept ein Bezug in Form einer genauen Lösung.

2.1.1 FTS-Leitsteuerung

Die FTS-Leitsteuerung bildet den wichtigsten Untersuchungsbereich im Rahmen dieser Arbeit aus, da die dezentrale Steuerung als eine alternative Lösung den Einzug in die FTS-Leitsteuerung erfordert. Nach der VDI ist eine FTS-Leitsteuerung wie folgt definiert („VDI-Richtlinie: VDI 4451 Blatt 7 - Leitsteuerung für FTS“, 2015):

„Eine FTS-Leitsteuerung besteht aus Hard- und Software. Kern ist ein Computerprogramm, das auf einem oder mehreren Rechnern abläuft. Sie dient der Koordination mehrerer Fahrerloser Transportfahrzeuge und/oder übernimmt die Integration des FTS in die innerbetrieblichen Abläufe."

Die Leitsteuerung basiert demnach auf einer zentralen Steuerung, dessen zentrale Instanz ein „Computerprogramm" ist. Die Aufgaben der Leitsteuerung sind die **Integration** des FTS in seine Umgebung, Annahme von **Transportaufträgen** und das Bereitstellen von entsprechenden **Funktionsblöcken** für Aufgaben. Ohne eine übergeordnete Leitsteuerung können die Fahrzeuge miteinander nicht kommunizieren und agieren, da sie kaum eigene Entscheidungen treffen (Ullrich, 2014, S. 130). Die Kernaufgabe einer FTS-Leitsteuerung ist es Transportaufträge zu erledigen, welche von ausgewählten Nutzern beauftragt werden. Jeder dieser Aufträge ist gekennzeichnet durch eine ID, Quelle, Senke und weiteren Information wie z.B. Priorität oder dergleichen (Ullrich, 2014, S. 133).

Die Leitsteuerung ist in **drei Bereiche**[1] unterteilt:

- Benutzer-Interface
- Transportauftragsabwicklung
- Servicefunktionen

Benutzer-Interface:

Es geht hierbei um die Schnittstelle zwischen Mensch-zu-Maschine, aber auch um die Schnittstelle zwischen Maschine-zu-Maschine, wie z.B. WLAN-Protokolle.

Transportauftragsabwicklung:

Transportauftragsverwaltung: Der Erhalt von Transportaufträgen geschieht in der Regel nach der Reihenfolge. Auch Aufträge mit zeitlicher Einplanung durch übergeordnete Produktionsplanungssysteme sind möglich. Folglich werden die Aufträge nach ihrer Priorität und zeitlicher Einplanung stets auf ihre Ausführbarkeit geprüft. Die Ausführbarkeit ist gegeben, wenn in der Quelle Material bereitsteht und die Senke aufnahmebereit ist.

Fahrzeugdisposition: Bei der Fahrzeugdisposition wird dem Auftrag der günstigste FTF zuzuweisen und per Fahrauftrag an die Fahrauftragsverwaltung übergeben. Die Eigen-

[1] Die Bereiche werden nach (Ullrich, 2014, S. 135 ff.) erläutert. Auf den Bereich der Servicefunktionen wird nahstehend nicht näher eingegangen, da dies in dem aufzustellenden Konzept nicht behandelt wird.

schaft „günstig“ kann aus verschiedenen Vorgehen bestimmt werden, wobei im einfachsten Fall ein beliebig freies FTF ausgewählt wird. Weitere Kriterien hierfür sind vor allem die kürzeste Entfernung einer FTF zur Quelle, aber auch die Vermeidung von möglichen Hindernissen auf den Fahrstrecken, die Aufnahmemöglichkeit mehrerer Lasten an unterschiedlichen Orten und Prognosen des Systemzustands in naher Zukunft, wie z.B. die aktuelle Batteriekapazität.

Fahrauftragsabwicklung: Aus den Transportaufträgen werden die Fahr- und Aktionsaufträge generiert. Ein Fahrauftrag ist das Ausführen einer Fahrbewegung von einer Position A zur Position B. Als Aktionsaufträge gelten z.B. Lastaufnahme, Lastabgabe oder Batterieladen. Sie reguliert zudem die Verkehrslage durch die Verkehrsleitsteuerung für eine störungs- und kollisionsfreie Fahrt aller FTF.

2.1.2 Fahrerlose Transportfahrzeuge

Fahrerlose Transportfahrzeuge sind flurgebundene Förderfahrzeuge, die für den Materialtransport sorgen. Sie werden automatisch gesteuert und können sich ohne weitere Hilfe selbst bewegen („VDI-Richtlinie: VDI 2510 Fahrerlose Transportsysteme (FTS)“, 2005, S. 7). In vielen produzierenden Unternehmen werden bereits seit Jahrzenten fahrerlose Transportfahrzeuge eingesetzt, die den Materialtransport übernehmen. Durch den aufkommenden Trend zu einem höheren Automatisierungswunsch der Prozesse, geraten auch die fahrerlosen Transportfahrzeuge in Form von autonomen Robotern mehr in den Fokus (W. A. Günther, 2012, S. 13 f.). Der Aufbau eines FTF ist stets dem Anwendungsfall anzupassen, wie z.B. der Einsatz einer Rollenbahn, um Transportkisten aufnehmen zu können.

Für die Bewegungssteuerung von fahrerlosen Transportfahrzeugen (FTF) werden verschiedene Verfahren verwendet. Nachfolgend eine Auflistung dieser Verfahren ohne weitere Erläuterung ihrer Funktionsweise sowie der gegenseitigen Vor- und Nachteile (Niemann u. a., 2006):

- Steuerung durch optische Verfahren
- Steuerung durch induktive Verfahren
- Steuerung durch Magnetmarken
- Steuerung durch GPS
- Steuerung durch Transponder

2.2 IT-Entwicklung

Im Zuge der IT-Entwicklung ist zu beobachten, dass die Produktivität nicht in gleicher Weise gestiegen ist, wie die Investitionen in die Informations- und Kommunikationstechnologien getätigt wurden. Im Gegenteil findet eher eine Nichterfüllung der Erwartungen und somit kein positiver Zusammenhang zur Verbesserung der Produktion statt (Jahn, 2017, S. 5). Es entsteht eine negative Wirkungsbeziehung, obwohl die entwickelten Technologien immer günstiger und besser werden, welches als das Produktivitätsparadoxon der Informationstechnologie bezeichnet wird (Piller, 1998, S. 1). Dies bestätigt Prof. Dr. Kamyar Sarshar anhand zahlreicher Untersuchungen und sagt: *„Trotz zunehmender IT-Investitionen und eines hohen Innovationsgrades ist eine Steigerung der Produktivität, […] nicht eindeutig belegbar“* (Sarshar, 2015).

Nach der obigen Sensibilisierung in das Thema der IT und die erhoffte Produktivitätssteigerung in der Produktion, werden im Folgenden diverse Bereiche aus der IT-Entwicklung, welche im Rahmen dieser Arbeit zur Verwendung gekommen sind, näher beschrieben.

2.2.1 Mikrocontroller mit Fokus auf Arduino

Unter Mikrocontroller sind „Ein-Chip-Systeme“ zu verstehen, die neben dem Prozessor noch diverse weitere, dem Anwendungsfall angepasste Komponenten beinhalten. Anders ausgedrückt – Mikrocontroller sind spezielle Mikrorechner. Ausgezeichnet sind sie dadurch, dass sie kostengünstig in hohen Stückzahlen hergestellt und in einem großen Einsatzraum genutzt werden können, wie beispielsweise für Aufgaben in der Steuerung und Kommunikation (Metz, 2014, S. 1).

Bei der Auswahl von Mikrocontroller wird auf eine Vielzahl an Herstellern und Modellen gestoßen. Ein besonderes Augenmerk ist hierbei auf das Open-Source Projekt „Arduino“ zu werfen. Das im Jahr 2006 gestartete Projekt hat innerhalb kürzester Zeit Einzug in den verschiedensten Personenkreisen gefunden, wie Elektrotechniker, Informatiker und sogar Privatpersonen, die aufgrund der sehr einfach zu erlernenden Programmierung mittels der bereitgestellten IDE (Integrierte Entwicklungsumgebung) auf schnelle Weise ihren Weg in die Welt des Programmierens finden können (Metz, 2014, S. 9).

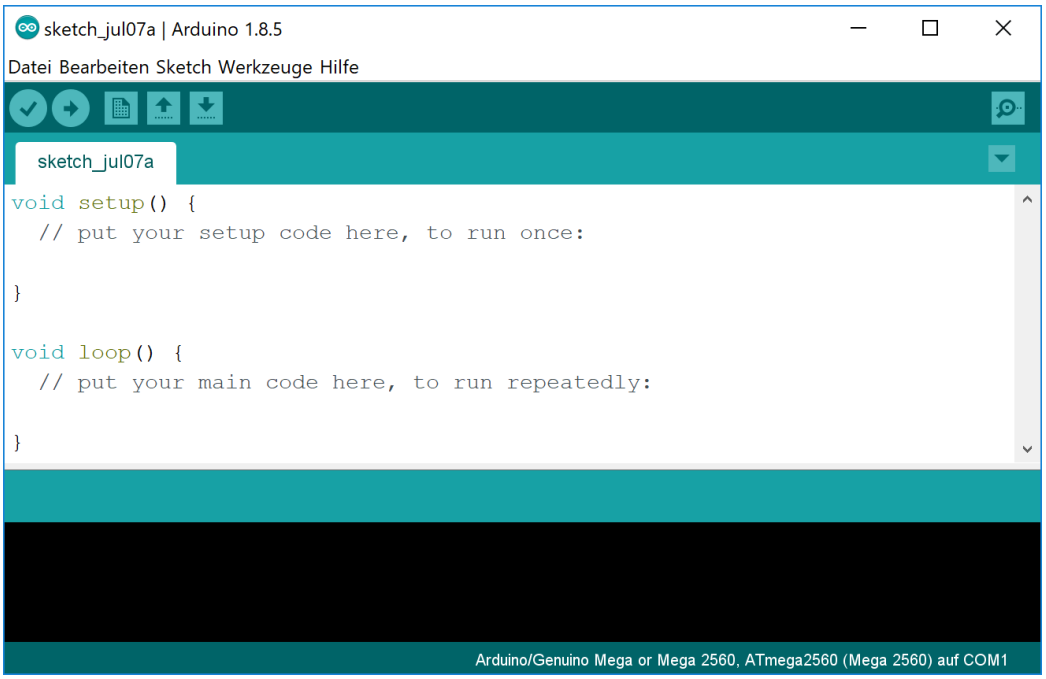


Abbildung 1: Arduino-IDE - Sketchaufbau

Bei der Programmierung wird mit „Sketches“ gearbeitet. Diese sind die eigentlichen Programmdateien und werden nach erfolgter Programmierung auf den Arduino hochgeladen. Bei aktiver Stromversorgung startet der Arduino umgehend den hochgeladenen Programmcode. Der Aufbau eines Sketches besteht aus zwei Teilen, dem Setup- und dem Loop-Teil. In dem Setup findet ein Initialisierungsprozess statt, worin diverse Einstellungen und Zuweisungen am Anfang des Programms **einmalig** erfolgen. Nachdem dieser Teil durchlaufen ist, wird es während des gesamten Durchlaufes nicht erneut aufgerufen. Der Loop-Teil hingegen, wie der Name schon hindeutet, ist ein ständiges Wiederholen des darin enthaltenen Programmcodes. Demnach kann in bestimmten Sequenzen immer wieder eine Abfrage eines zu beobachtenden Parameters erfolgen, wie beispielsweise die Raumtemperatur in einer Örtlichkeit.

Die Programmierung erfolgt in den beiden weit verbreiteten Sprachen **„C“** und **„C++“**. Anhand vieler Beispielprogramme und Bibliotheken wird die Programmierung gerade für Anfänger besonders vereinfacht. Bibliotheken verhelfen dabei, hochtechnologische Module mit komplexen Funktionen auf eine einfache Art und Weise zu verwenden, wodurch nur noch Anpassungen an die jeweiligen Bedürfnisse notwendig werden. Für die meisten Module gibt es solche Bibliotheken zu finden. Entweder über die in der IDE enthaltenen Bibliothek-Verwaltung oder über das Internet, wie beispielsweise auf GitHub[2].

[2] GitHub ist eine Online-Entwicklerplattform, in der die Entwickler ihre Software bereitstellen.

Nachfolgend werden einige Begrifflichkeiten und Verfahren in der Programmierung mit Arduino erläutert, die im Konzeptteil eingesetzt wurden. Verwendet wurde dabei die Programmiersprache „C“.

Pulsweitenmodulation (Pulse Width Modulation): Mit dem Verfahren der Pulsweitenmodulation ist durch sequentielles an- und ausschalten des Stroms möglich, die durchschnittliche Stromzufuhr an eine Komponente zu regeln. Dadurch wird beispielsweise die Steuerung von Motorgeschwindigkeiten ermöglicht. Soll ein Motor mit maximaler Geschwindigkeit in die vorgegebene Richtung drehen, wird der höchste PWM-Wert „255“ genutzt. Dadurch wird dem Motor durchgängig ein durchschnittlicher Strom von 5 V zugeführt. Analog dazu wird bei einer Geschwindigkeit von 50 %, ausgehend des Maximums, nur die Hälfte des höchsten PWM-Werts „127“ genutzt, wodurch die durchschnittliche Stromzufuhr auf ca. 2,5 V sinkt. Die mit der PWM-Methode zu steuernde Komponente muss dabei mit einem der vorhandenen PWM-Pins auf dem Arduino-Board angeschlossen werden. Ein Vergleich zwischen verschiedenen Zuständen, in Abhängigkeit des PWM-Wertes, ist in Abbildung 2 dargestellt.

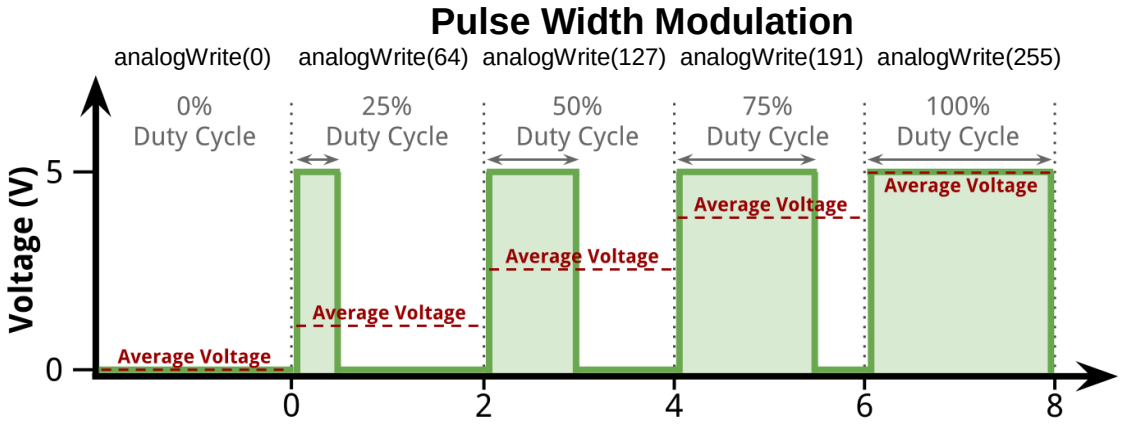


Abbildung 2: Durchschnittliche Stromzufuhr mit der Pulsweitenmodulation[3]

Verwendung von Datentypen und Methoden zur Konvertierung: Je nach Verwendung eines Datentyps, werden im Arbeitsspeicher des Arduinos Speicher reserviert. In Abbildung 3 werden die Datentypen der Sprache „C“ aufgeführt. Üblicherweise werden die Datentypen „char“, „int“, „float“ und „double“ verwendet. Zu dem Bereich der „Integer“ gehört außerdem ein oft gebrauchter Datentyp „byte“ zu, welches in der Abbildung nicht aufgeführt ist.

[3] (Bolt, 2013)

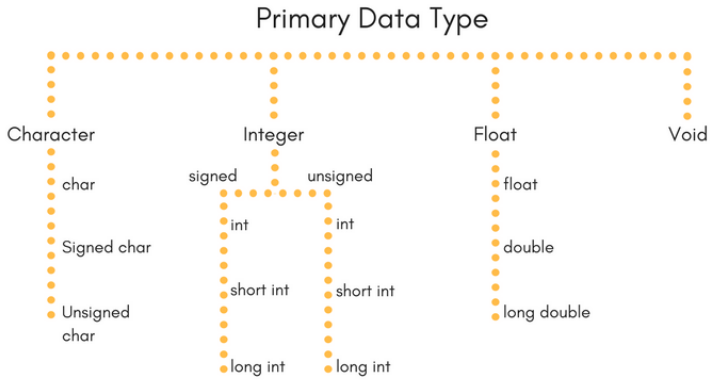


Abbildung 3: Datentypen in der Sprache "C"[4]

Der Datentyp des „String“ ist in der Sprache „C“ nicht vorhanden. Das Pendant dazu ist der Datentyp „char“, welches für Text und Zeichen genutzt wird und vom Aufbau her wie ein Array funktioniert. Daher muss für den Fall, dass ein „String“ als Datentyp benötigt wird, eine Konvertierung in ein „Stringobject“ stattfinden. Dies wird ermöglicht, indem die zu konvertierende Variable mit der Funktion „String“ umschlossen wird. Die Konvertierung findet demnach mit der Bezeichnung als „String(variable)“ statt. Diese Strings können zudem durch weitere String-Methoden in ein anderes Format konvertiert werden. Da keine tiefgründigen Programmierthemen behandelt werden, wird an dieser Stelle, bei Interesse, an die allgemeingültige Fachliteratur zu dem Themenbereich verwiesen.

2.2.2 Line-Tracking-Sensoren

Mikrocontroller sind wie bereits erwähnt sehr nützlich, um mit zusätzlichen Modulen Zustandsabfragen durchzuführen und daraus resultierend Ereignisse ausführen zu lassen. Insbesondere Arduino-Systeme sind nahezu unbegrenzt erweiterbar, sowohl durch eigene Entwicklungen als auch durch Module, die eingekauft werden können. Beispielsweise kann die Funktion eines Line-Tracking-Sensors eigenständig entwickelt werden, da sie auf einfache Art mit der Technologie der IR-Sensorik arbeiten. Dies ist nicht der weitere Betrachtungsgegenstand dieser Arbeit, weshalb es nachfolgend um die „Line-Tracking-Sensoren“ als fertige Kaufteile geht.

Mit Line-Tracking-Sensoren wird das Ziel verfolgt, auf einer schwarzen Linienspur die Orientierung von Fahrzeugrobotern zu ermöglichen. Der Sensor ermittelt durch das Emittieren und dem anschließenden Empfangen eines IR-Lichtes die Oberflächenfarbe des Bodens. Durch die zurückgemeldeten Rückgabewerte werden dem Fahrzeug die

[4] (Studytonight.com, 2018)

auszuführenden Aktionen vorgeschrieben. Dabei geht es um zweier Art von Rückgabewerten. Bei einer schwarzen Oberfläche meldet der Sensor den Wert **„0“** und bei einer weißen Oberfläche den Wert **„1“** zurück (Abbildung 4).

Abbildung 4: Erkennung der Oberflächenfarbe durch Line-Tracking-Sensoren[5]

Um die Oberflächenfarbe der Fahrbahn zu erkennen, werden die Sensoren üblicherweise auf der Unterseite des Fahrzeuges angebracht. Dabei gilt, je höher die Anzahl der Sensoren, desto genauer wird die Steuerung des Fahrzeuges. Die Mindestanzahl an Sensoren liegt bei zwei, denn nur bei einem Sensor können keine Veränderungen der Bewegung ermittelt werden.

Genaue Funktionsweise: Line-Tracking-Sensoren bestehen aus je einer IR-LED und einer Fotodiode. Die IR-LED emittiert ein Licht, welches durch die Oberfläche auf die Fotodiode reflektiert wird. Je nach der Stärke der Reflektion wird eine Ausgangsspannung geliefert, wobei für hellere Oberflächen größere und für dunklere Oberflächen kleinere Werte resultieren, da auf einer dunkleren Oberfläche die Reflektion geringer ist als auf einer helleren. Bei der Programmierung werden die Signale als ganze Zahlen abgefragt, sodass wie oben bereits beschrieben, die Rückgabewerte entweder **„0“** oder **„1“** sein können. Eine Veranschaulichung[6] in Abbildung 5 verdeutlicht die genaue Funktionsweise.

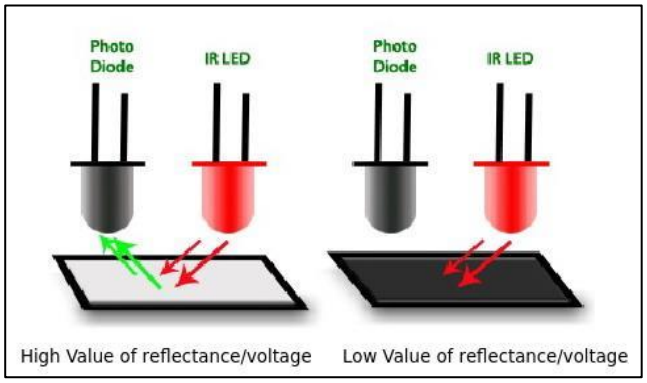


Abbildung 5: Funktionsweise von Line-Tracking-Sensoren

[5] (DFRobot, 2017a)
[6] (Sanjeev, 2014)

2.2.3 RFID für die Informationsverarbeitung

Das System der RFID-Technologie (Radio Frequency Identification) besteht aus einem RFID-Lesegerät und den zu lesenden RFID-Tags – auch als RFID-Transponder bezeichnet. Die Technologie wurde in den 1930er Jahren für militärische Zwecke – der Flugzeugidentifikation – eingeführt. Im Laufe der Zeit fand der Einzug in diversen weiteren Bereichen Einzug statt. So begann die Verwendung in den 1960er Jahren für Zugangskontrollen, indem Jahr 1978 für Tierkennzeichnungen und zuletzt in dem Jahr 1988 in der Industrie. Die industrielle Anwendung der RFID-Tags findet sich beispielsweise in der Funktion als ein digitales Typenschild wieder, worin diverse Produktinformationen festgehalten und im Produktionsprozess per Abruf zur Verfügung gestellt werden können (Jahn, 2017, S. 12).

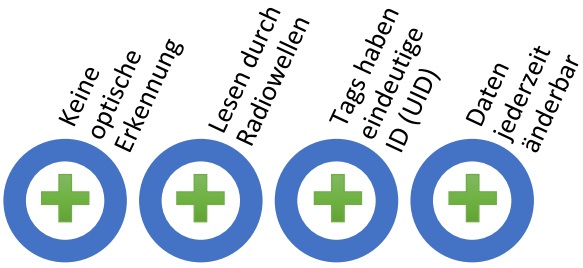


Abbildung 6: Vorteile der RFID-Technologie

In Abbildung 6 sind die Vorteile der RFID-Technologie aufgeführt. Für das Lesen eines RFID-Tags ist keine optische Erkennung notwendig, weshalb die Identifikation nicht durch Schmutz oder anderweitigem Abdecken des RFID-Tags behindert werden kann. Durch das Lesen mittels Radiowellen wird die Fehlerrate verringert und ist im Vergleich zu einem optischen Leseverfahren zuverlässiger, da keine optische Führung notwendig ist und eine Annäherung in den Bereich der Radiowellen ausreichend für den Beginn des Lesevorganges ist, sodass beispielsweise bei Inventuren eine Genauigkeit von 99 % erreicht wird (Jahn, 2017, S. 13 ff.). Eine Übertragung der Radiowellen mit der RFID-Technologie funktioniert in bestimmten Frequenzbändern. An dieser Stelle sei zu erwähnen, dass die im Konzeptteil genutzten RFID-Lesegeräte (MFRC522) im Hochfrequenzbereich auf 13,56 MHz arbeiten.

RFID-Tags werden in verschiedenen Typen und Speichergrößen, vertrieben durch diverse Anbieter. Nachfolgend geht es um die Tags des Herstellers NXP mit der Typbezeichnung **„MIFARE Classic 1K“**. Diese im 13,56 MHz funktionierenden Tags haben eine Speicherkapazität von insgesamt 1024 Bytes, wobei effektiv nur ca. 700 Bytes davon genutzt werden können. Der Grund dafür sind die Architektur des Speicheraufbaus und die enthaltenen Sicherheitsmechanismen. Ein RFID-Tag in dieser Speichergröße ist in 16 Sektoren á 4 Blocks unterteilt, welche insgesamt 64 einzelne Speicherbereiche ausmachen. In jedem dieser Sektoren ist der vierte Block für die Authentifizierung mittels

Sicherheitsschlüsseln reserviert, die bei dem Lesevorgang eines Blocks in dem jeweiligen Sektor von dem zu lesenden Gerät übermittelt werden müssen, denn ohne die Kenntnis über den Sicherheitsschlüssel kann keine Authentifizierung stattfinden und der Lesevorgang wird verweigert. Diese Blocks werden als **„Trailer Blocks“** oder **„Sector Trailer“** bezeichnet. Wie in Abbildung 7 dargestellt, besteht dieser Block aus zwei Sicherheitsschlüsseln, wobei der Zweite nur optional eingesetzt wird und bei Bedarf auch als Informationsspeicher dienen kann. Im Abschnitt 4.4.2 wird der Speicheraufbau anhand eines Beispiels beschrieben (Skyetek Inc., 2017, S. 5 ff.).

Byte Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	Key A						Access Bits				Key B (optional)					

Abbildung 7: Aufbau eines Trailer Blocks in RFID-Tags[7]

Die Informationen werden im Hexadezimal-System auf die jeweiligen Speicherbereiche der Tags geschrieben und sind daher beim reinen Aufrufen des Speicherinhaltes für das menschliche Auge nicht direkt lesbar. Bei der Programmierung hingegen wird der zu speichernde Wert als Text oder Zahl übermittelt und die Konvertierung in das Hexadezimal-System wird durch den Computer, in diesem Falle durch den Arduino, übernommen.

2.2.4 M2M-Kommunikation mit dem MQTT-Protokoll

Unter einer Maschine-zu-Maschine-Kommunikation ist nicht nur die Kommunikation zwischen den industriellen Maschinen zu verstehen. Der Begriff der Maschine umfasst weit mehr als z.B. Produktionsmaschinen (Dressler, 2013). Gerade im Hinblick auf die heutige Entwicklung der Technologien ist es möglich, nahezu alles kommunikationsfähig zu machen, wie z.B. durch die Anbringung von Mikrocontrollern (vgl. 2.2.1).

Die Kommunikation von Maschinen ist ein großes und insbesondere in den letzten Jahren ein für die produzierenden Unternehmen essentielles Thema aus Sicht der Bewegung in Richtung Industrie 4.0 geworden. Gemeint wird damit der Datenaustausch zwischen den kommunikationsfähigen Maschinen (Sendler, 2013, S. 11). Insbesondere der Aspekt zur selbststeuernden Produktion bedingt eine drahtlose Kommunikation, um Entscheidungswege realisieren zu können (Dressler, 2013). Darin liegt auch der Grundgedanke von Industrie 4.0, und zwar die Vernetzung aller beteiligten Systeme in der Pro-

[7] Nach (Skyetek Inc., 2017, S. 7)

duktion, die zur Wertschöpfung beitragen, indem sie die notwendigen Informationen ermitteln und bereitstellen (BMWi, 2016, S. 3). Als ein Oberbegriff für das Thema der M2M-Kommunikation gilt der Begriff des **„Internet-of-Things (IoT)“**, in der Dinge (Entitäten[8]) verschiedenster Art untereinander vernetzt werden und sich in ein Netzwerk zusammenschließen. Dazu gehört auch die Kommunikation zwischen den Dingen und den Menschen.

Für die Implementierung einer M2M-Kommunikation gibt es viele verschiedene Ansätze, wobei bis heute noch keines dieser als das Standardprotokoll für die M2M-Kommunikation im IoT festgelegt ist (Knauer, 2016). Nachfolgend eine Auflistung einiger bekannter Kommunikationsprotokolle mit dem Prinzip zur Veröffentlichung einer Nachricht:

- MQ Telemetry Transport (**MQTT**) – PubSub-Prinzip
- OPC Unified Architecture (**OPC UA**) – PubSub-Prinzip
- Constrained Application Protocol (**CoAP**) – Request/Response-Prinzip

Für das aufzustellende Konzept bedingt es einem Protokoll basierend auf dem **„PubSub-Prinzip“**, worin eine Kommunikation von der anfragenden Entität aus startet. Der Sender veröffentlicht aus eigenem Willen eine Nachricht, welches alle relevanten Teilnehmer erhalten. Dies ist erforderlich, da im Konzeptteil die dezentrale Steuerung durch den Auftrag in Form der Transportkiste eine „Push-Bewegung“ darstellt, indem sie die Ressourcen für die Auftragserfüllung anfragt (vgl. 4.2.2). Alternativ gibt es Protokolle, die mit dem „Request/Response-Prinzip“ arbeiten, in der die Kommunikationsteilnehmer angefordert werden zu antworten. Beide erwähnten Protokolle mit dem PubSub-Prinzip sind „leichtgewichtige“ Protokolle, die wenig Ressourcen hinsichtlich der Kapazitäten des Prozessors und des Speichers besitzen (Drolshagen, 2015, S. 6). Im Weiteren wird der Aufbau des MQTT-Protokolls näher vorgestellt, da es für das Zielsystem „Arduino“ die bessere Wahl darstellt. Der Einstieg in die M2M-Kommunikation mit den Arduinos ist aufgrund der breiten Verfügbarkeit und Anwendung durch diverse Personenkreise einfacher zu realisieren. Zudem erleichtert die verfügbare Bibliothek des Entwicklers „Miguel Balboa“ (Balboa, 2012/2018) mit seinen Beispielcodes die Implementierung, da lediglich Anpassungen mit kleinen Funktionserweiterungen an das eigene System durchzuführen sind.

Das MQTT-Protokoll ist ein TCP-basiertes Nachrichtenprotokoll, in der jeweils eine TCP-Verbindung von den Kommunikationsteilnehmern zu dem zentralen Element, dem MQTT-Broker, aufgebaut wird. Der MQTT-Broker ist das Herzstück des Protokolls, da

[8] Entität = „Eine physische/virtuelle Entität repräsentiert das zu betrachtende Objekt, […]. Ein wichtiger Aspekt in diesem Zusammenhang ist die Objektidentität. Sie ermöglicht es, ein bestimmtes Objekt zweifelsfrei zu identifizieren und anzusprechen.“ (Neubauer, 2014)

jegliche Nachrichten darüber an die Zielobjekte weitergeleitet werden (Obermaier, 2017, S. 2). Aufgrund dieser erwähnten Architektur des Protokolls herrscht keine direkte Verbindung zwischen den Kommunikationspartnern. In Abbildung 8 ist das PubSub-Prinzip in dem MQTT-Protokoll dargestellt. Die Publisher sind jene Entitäten, die Nachrichten veröffentlichen und die Subscriber jene, die die Nachrichten in Abhängigkeit ihres Themenbereiches empfangen. Die Kernaufgabe des MQTT-Brokers ist daher als die Bereitstellung der Nachrichten für die erforderlichen Kommunikationsteilnehmer zu beschreiben.

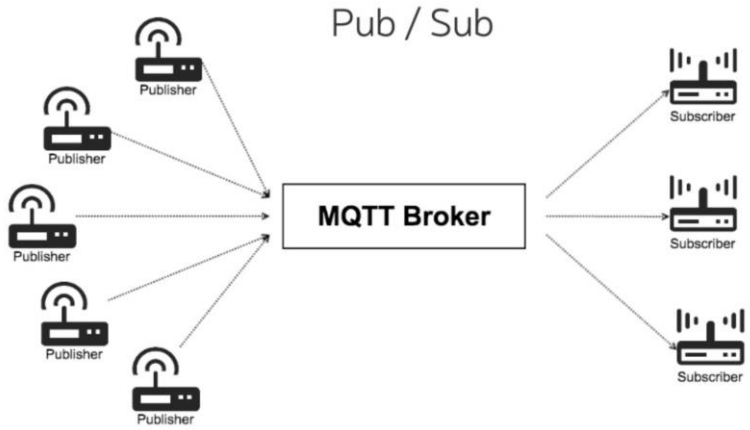


Abbildung 8: PubSub-Prinzip über MQTT-Broker[9]

Bei dem Veröffentlichen (Publish) einer Nachricht werden keine direkten Kommunikationspartner angeschrieben. Das Protokoll besagt, dass alle zu veröffentlichenden Nachrichten in einen zu benennenden Ordner gesendet werden. Die Übermittlung der Nachricht geschieht dabei durch das Abonnieren (Subscribe) des relevanten Ordners durch die Empfänger. Alle veröffentlichten Nachrichten sind demnach auf einen zu Beginn bestimmten Ordner adressiert. Diese Ordner werden als Topics bezeichnet und der Aufbau ist prinzipiell gleichzusetzten mit der Ordnerstruktur eines PC-Systems. Alle Topics können einzeln als Haupttopics existieren oder weitere Untertopics besitzen. Durch Untertopics sind somit beispielsweise Detailinformationen eines ganzen Systems abzufragen. Beispiele für Benennungen der Topics können lauten:

- Transportauftraege/
- Transportauftraege/2018
- Stoerungsmeldungen/Maschinen/AP_X1

Zusätzlich gibt es die Möglichkeit „Wildcards“ einzusetzen, womit eine genauere Steuerung der Abonnements möglich wird. Dazu gehören zum einen das „+“-Zeichen, welches als das Single-Level Wildcard bezeichnet wird. Damit ist es möglich alle Topics aus einer

[9] (Obermaier, 2014, Fol. 17)

Hierarchieebene zu abonnieren. Zum anderen gibt es das „#“-Zeichen, womit alle Untertopics des gewünschten Haupttopics abonniert werden können (Obermaier, 2017, S. 2).

Eine grundlegende Fragestellung in dem Bereich der Kommunikation liegt in der Beantwortung der Frage hinsichtlich der Sicherheit gegenüber das Risiko des Abhörens durch unbefugte Dritte. Die Kommunikation erfolgt, wie oben erwähnt über eine TCP-Verbindung, wofür die Ports 1883 und 8883 genutzt werden. Mit dem Port 1883 erfolgt eine unverschlüsselte und mit dem Port 8883 eine über SSL verschlüsselte Kommunikation. Der Grund für die Auswahlmöglichkeit liegt in der Betrachtung des Erhaltungswunsches der Leichtgewichtigkeit, um in dem zu kommunizierenden Netzwerk bei Bedarf ressourcenschonend kommunizieren zu können (Drolshagen, 2015, S. 6). Außerdem gibt es die Möglichkeit einer Verschlüsselung der Rohdaten, wodurch die Nachricht selbst mittels kryptografischer Ansätze verschlüsselt und erst dann danach veröffentlicht wird. (Drolshagen, 2015, S. 7). Zusätzlich kann der Zugriff auf klassische Art mittels einer Authentifizierung über Benutzername und zugehörigem Passwort eingeschränkt werden.

Das MQTT-Protokoll verfügt insgesamt über drei Level des Quality-of-Service (QoS) (

Abbildung 9). Das QoS beschreibt die Qualität in Form von verschiedener Abläufen, die das Erreichen der Nachricht bei den Empfängern sicherstellt. Bei dem niedrigsten Leven wird eine Nachricht gesendet, ohne das Erreichen zu prüfen. Demnach kann hier die Nachricht verloren gehen und der Absender erhält keine Kenntnis über den Zustand der Nachricht. Bei dem nächsten Level wird eine Nachricht mehrmals zugestellt, sodass eine Zustellung garantiert ist, und daher Duplikate vorkommen können. Der höchste QoS-Level sicher zu, dass eine Nachricht bei dem Empfänger garantiert nur ein einziges Mal ankommt. Daher ist bei Anwendungsfällen, worin eine erhöhte Sicherheit über das korrekte Erhalten einer Nachricht erforderlich ist, die höchste Stufe die geeignetere Wahl, wobei hierbei mehr Ressourcen benötigt werden und daher die Leichtgewichtigkeit des Protokolls darunter leidet.

Abbildung 9: QoS-Levels im MQTT-Protokoll[10]

[10] Entworfen nach: (Maritsch, Kittl, & Ebner, 2015, S. 220)

2.3 Routenfindung mit dem A*-Algorithmus

Der A*-Algorithmus gilt als einer der bekanntesten heuristischen Algorithmen zur Routenfindung, da sie sehr einfach zu verstehen und in der Anwendung einfach in die Entwicklung zu implementieren ist (Jena & Liebelt, 2015, S. 7). Das Ziel dabei ist es den kürzesten Weg innerhalb eines Gittergraphen zu ermitteln. Nachfolgend folgt eine Einführung in das Themengebiet der Graphalgorithmen und anschließend eine detaillierte Erläuterung zu der Anwendung des A*-Algorithmus.

2.3.1 Graphalgorithmen und Routenfindung auf Graphen

Die Anwendung von Graphenmodellen dienen zur Beschreibung struktureller Zusammenhänge. Diese werden mittels Knoten und Kanten dargestellt, wie beispielsweise für Routenplanungen oder Kommunikationsnetzwerke („TUM - Mathematik - M9“, o. J.). Mit den genannten Kanten wird die Verbindung zwischen den Knoten aufgebaut. Die Beschreibung eines Graphen erfolgt anhand der Klassifizierung als Tupel G = (V, E), wobei **G** für den zu beschreibenden Graphen, **V** für die Menge der Knoten (vertex) und **E** für die Menge der Kanten (edge) steht (Nebel & Wild, 2018, S. 295). Die Kanten können gerichtet und gewichtet werden. Eine gerichtete Kante bedeutet, dass ausgehend vom einem Knoten nur in eine bestimmte Richtung bewegt werden kann. Bei ungerichteten Kanten spielt die Richtung keine Rolle, wodurch Wege wie A $\rightarrow$ B und rückwärts B $\rightarrow$ A möglich sind. Jeder Knoten stellt eine potentielle anfahrbare Position dar, sofern dieses nicht durch ein Hindernis belegt ist.

Durch das Verbinden der Knoten mittels der Kanten werden Graphen erzeugt. Der optische Aufbau der Knotenpositionen dient in erster Linie nur für eine einfache Darstellung. So ist beispielsweise bei der Erstellung eines Graphen über die Bundesländer Deutschlands möglich, die Bundesländer in Form von Knoten exakt so positionieren, wie es auch in der Realität ist. Dies spielt hinsichtlich der mathematischen Bedeutung keine Rolle, sofern die Kanten zu jedem hinführenden Knoten richtig getroffen und gewichtet werden.

Oft, so wie es in dieser Arbeit auch der Fall ist (vgl. 4.3), bieten sich Gittergraphen bei der Routenfindung in Räumlichkeiten an. Dabei wird wie in dem Beispiel aus Abbildung 10 ein ungerichteter Gittergraph aufgestellt, welches anschließend für die weiterführende mathematische Betrachtung in einer Matrix dargestellt wird. Darin

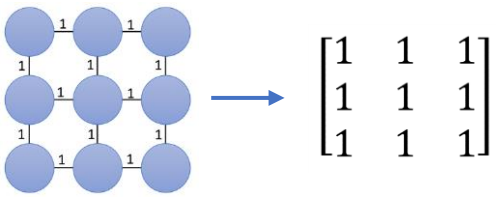


Abbildung 10: Beschreibung eines Gittergraphen als Matrix

sind alle Positionen der Knoten mit dem Wert **„1“** deklariert, welches den Zustand der Position beschreibt. Sollte eine Stelle in einer Räumlichkeit ein Hindernis darstellen, so existiert an der Position des Gittergraphen kein Knoten. In der Matrix-Darstellung werden die Hindernisse mit einer **„0“** deklariert.

Es stehen diverse Algorithmen zur Auswahl, wobei der Dijkstra- und der A*-Algorithmus zu den grundlegenden dieser Art gehören. Diese Algorithmen werden in Programmierumgebungen abgebildet, um rechenintensive Problemstellungen lösen zu können. Kleinere Aufgabentypen können in der Regel auch mit handschriftlicher Rechnung gelöst werden. Nachfolgend findet eine kurze Einführung zu dem in dieser Arbeit angewendetem A*-Algorithmus statt. Der A*-Algorithmus baut auf dem Dijkstra-Algorithmus auf, welches bei der Ermittlung der Route durch Einsatz einer zusätzlichen Information in Form einer Schätzfunktion (Heuristik) die Berechnung beschleunigt. Dadurch werden nicht alle erreichbaren Knoten auf dem Graphen, wie beim Dijkstra-Algorithmus, betrachtet, welches demnach zu einem schnelleren Ergebnis führt. Der A*-Algorithmus wird daher als ein informatives Suchverfahren bezeichnet (Velden, 2014). Aufgrund der schnelleren Lösungsfindung eignet sich das Verfahren zudem besser für den Einsatz mit einem Arduino-System.

2.3.2 Anwendung des A*-Algorithmus

Zunächst werden einige grundlegende Begriffe erläutert, die in der Beschreibung des A*-Algorithmus verwendet werden.

Tabelle 1: Grundlegende Begriffe in dem A*-Algorithmus

Begriff	Erläuterung
Warteschlange	Knoten, die für den Weg bis zum Betrachtungszeitpunkt in Frage gekommen sind, werden die Warteschlange aufgenommen. Der Knoten mit den geringsten Kosten befindet sich stets an der obersten Stelle.
Geschlossene Liste	Knoten, die nicht mehr untersucht werden müssen, da sie bereits in der Berechnung des Weges in Betracht gezogen wurden.
Kostenberechnung	*Formel:* f-Wert = G_{Kosten} + H_{Kosten} G_{Kosten} = Kosten für den bisherigen Weg zu dem aktuellen Knoten H_{Kosten} = Heuristik → geschätzte Kosten von dem aktuellen Knoten zum Zielknoten mittels einer Schätzfunktion

Für die Schätzfunktion ist keine feste Methode vorgegeben. Daher können unter anderem eine der folgenden Methoden angewendet werden:

- Manhattan-Methode (Szenarien mit 4 Bewegungsrichtungen)
- Euklidische Distanz (Szenarien mit 8 Bewegungsrichtungen – Diagonalen)

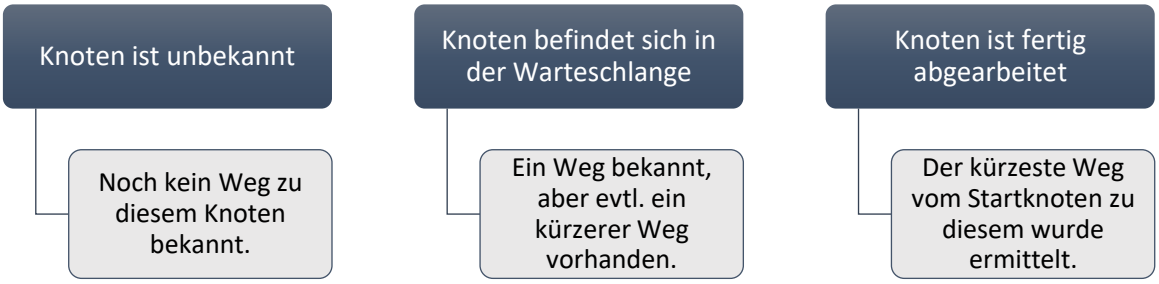
Die Manhattan-Methode eignet sich für Fahrzeuge mit vier Bewegungsrichtungen und die euklidische Distanz, wenn auch Diagonalen (45°) möglich und somit acht Bewegungsrichtungen ausführbar sind. Da das entwickelte Fahrzeug und die die zu befahrende Umgebung für die Ausführung von vier Bewegungsrichtungen konzipiert ist, folgt eine Erläuterung zu der Manhattan-Methode.

Erklärung Manhattan-Methode (Heuristik):

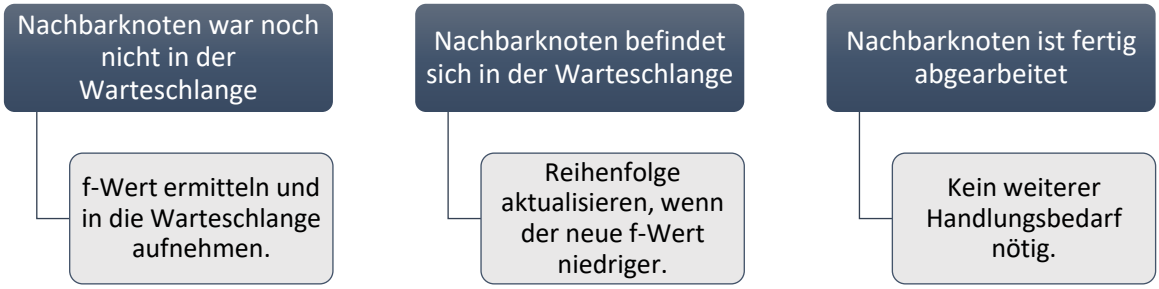
Die Manhattan-Methode stellt die einfachste Heuristik für die Ermittlung der Schätzkosten dar. Der Zustand jedes Knotens, ob befahrbare Position oder Hindernis, wird ignoriert und die Schätzkosten berechnet, als würde das Ziel auf direktem Weg erreicht werden können. Der Abstand des Weges vom Startpunkt zum Zielpunkt über die X- und Y-Koordinaten werden summiert und mit dem (hier einheitlichen) Kantenwert („1“) multipliziert. Dafür können die Koordinaten des aktuell betrachteten Knotens (x_1, y_1) und des Zielknotens (x_2, y_2) erfasst und die jeweiligen Schätzkosten (H) mittels folgender Gleichung ermittelt werden:

$$H = (x2 - x1) + (y2 - y1)$$

Jeder **Knoten** auf dem Graphen befindet sich in einem der folgenden 3 Zustände:



Jeder **Nachbarknoten** befindet sich in einem der folgenden 3 Zustände:



Vorgehen des A*-Algorithmus:[11]

Zunächst wird der Startknoten in die Warteschlange mit dem zugehörigen f-Wert aufgenommen. Anschließend wird dieser Knoten „expandiert“. Das Expandieren bedeutet die Berechnung des f-Wertes für jedes Nachbarknoten des expandierten (Kostenberechnung) und die anschließende Aufnahme in die Warteliste, sofern das aufzunehmende Nachbarfeld mit einem günstigeren f-Wert noch nicht in der Warteschlange vorhanden ist. Nach dem Expandieren eines Knotens wird sie in die geschlossene Liste aufgenommen. Ein bereits in der geschlossenen Liste vorhandener Knoten kann nicht erneut für die Wegermittlung in Betracht gezogen werden. Der obige Durchlauf wird so lange fortgeführt bis die Zielposition in die Warteschlange reinkommt und mit dem anschließenden Expandieren in die geschlossene Liste aufgenommen wird (Abbildung 11). Der tatsächlich hinterlegte Weg wird zuletzt über die Zusammensetzung des letzten f-Wertes ersichtlich.

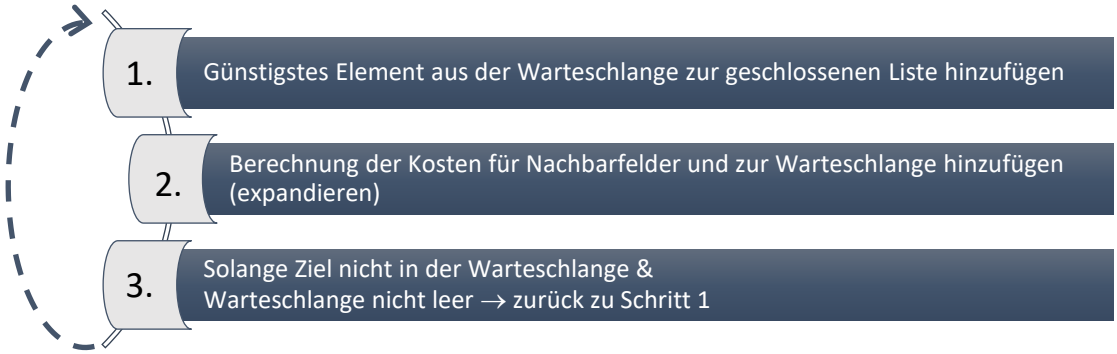
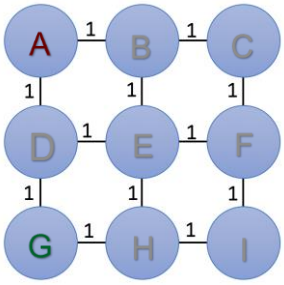


Abbildung 11: Schema zur Vorgehensweise bei dem A*-Algorithmus

Eine einfache und übersichtliche Schreibweise ist die Baumansicht. Dabei werden alle expandierten Knoten markant dargestellt, sodass auch der Weg zu einem Zielknoten visuell nachvollzogen werden kann.

Beispielaufgabe:



Heuristik-Tabelle

Knoten	H-Kosten
A	2
B	3
C	4
D	1
E	2
F	3
G	0
H	1
I	2

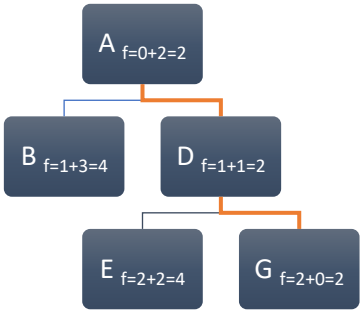


Abbildung 12: Lösung einer Beispielaufgabe mit Tree-Visualization zum A*-Algorithmus

[11] Nach (Lester, 2005) und (Velden, 2014)

Nebenbedingung: Bei gleichem f-Wert hat die alphabetische Reihenfolge Vorrang!

Für die Routenberechnung von dem Knoten **„A“** zu dem Knoten **„G“** wird eine Gesamtkostenwert von 2 ermittelt. Dies bedeutet zugleich auch die hinterlegte Strecke über den Weg **„A → D → G“**.

2.4 Minimum Viable Product

Das Minimum Viable Product (MVP) ist eine Methode der Produktentwicklung, welche von Eric Ries popularisiert worden ist. Gerade im Bereich der Startups ist es eine bewährte Methode. Ries definiert das Prinzip des MVP mit einer Version des Produktes, mit dem die maximale Menge an validiertem Wissen über die Anforderungen des Kunden erhalten wird (Ries, 2009b). Grundlegend ist die Intention, dass der Kunde in den Entwicklungsprozess eingebunden wird. Dazu wird in iterativen Teilschritten der Kunde über jeden Meilenstein der Entwicklung informiert und seine Meinung eingeholt. Diese iterativen Teilschritte haben vor allem einen Zweck – ein Ergebnis zu liefern, welches mit minimalem Aufwand die Anforderungen des Kunden erfüllt. Durch die Methode des MVP wird demnach die Frage erörtert, wie der günstigste und schnellste Weg ausschaut, um weitere Erkenntnisse über die konkreten Anforderungen des Kunden zu sammeln (Kniberg, 2016). Statt den Kunden mit vielen Funktionen zu beeindrucken, die erst bei der Lieferung zu sehen sein werden, wird dadurch ermöglicht dem Kunden nur die Funktionen zu liefern, die explizit gebraucht werden (Ries, 2009a). In der Produktentwicklung von heute geht es vor allem um Geschwindigkeit, früher Kundenkontakt und Ästhetik, wofür die Methode des MVP passend angewendet werden kann (Fleisch, Weinberger, & Wortmann, 2017, S. 13).

Die MVP-Methode wird an dem Beispiel von Kniberg (Kniberg, 2016) deutlich gemacht (Abbildung 13). In dem Beispiel geht es um die Bestellung eines Autos, mit dem der Kunde sich von einem Ort „A“ zu dem Ort „B“ bewegen möchte. Er fängt vorerst an, ein falsches Verständnis der Methode zu beseitigen, indem ein falsches Vorgehen erläutert wird. Teilschritte sollen dabei nicht die Bereitstellung einzelner Komponenten eines Fahrzeuges bedeuten, sondern sind dabei stets Lösungen zu erreichen, die die Anforderung des Kunden mit minimalem Aufwand erfüllen. So bringt es dem Kunden nichts, wenn im ersten Schritt nur ein Reifen vorgestellt wird, da damit der Kunde seine Anforderungen nicht erfüllen und demnach keine Meinung abgeben kann. Dieses falsche Vorgehen wird fortgeführt, bis der Kunde im letzten Schritt erst ein funktionsfähiges Auto erhält, welches erst zu dem Zeitpunkt die Anforderungen erfüllt. Was dabei an der Methode verloren gegangen ist, ist das Einbeziehen des Kunden mit seinen Meinungen zu

den Teillösungen. Dadurch konnte im Entwicklungsprozess kein Einfluss des Kunden erreicht werden und wurde letztendlich „nur“ ein Auto geliefert.

Das richtige Vorgehen stellt Kniberg im unteren Teil der Abbildung dar. Innerhalb der Produktentwicklungsphase entwickelt das Entwicklungsteam nach dem MVP-Prinzip erstmal das kleinste mögliche Produkt, wobei hier dem Kunden im ersten Entwicklungsschritt ein Skateboard vorgestellt wird. Dem Kunden wird mitgeteilt, dass diese lediglich die ersten iterativen Schritte der Entwicklung sind, um keine Enttäuschung hervorzurufen, und bittet den Kunden dieses für ein anschließendes Feedback zu testen. Der Kunde ist damit nicht zufrieden, kann aber dennoch seine Anforderung erfüllen, da hiermit die Anforderung als Transportmöglichkeit von A nach B gegeben ist. Der Entwicklungsverlauf ist so fortzuführen, dass bezogen auf das Feedback des Kunden weitere Funktionen hinzugefügt oder sogar eliminiert werden, bis im letzten Entwicklungsschritt der Kunde das gewünschte Produkt erhält. Durch dieses Vorgehen hat der Kunde im Laufe des Entwicklungsprozesses erkannt, dass ein Cabrio die bessere Wahl ist, da ihm der frische Wind beim Fahren gefallen hat.

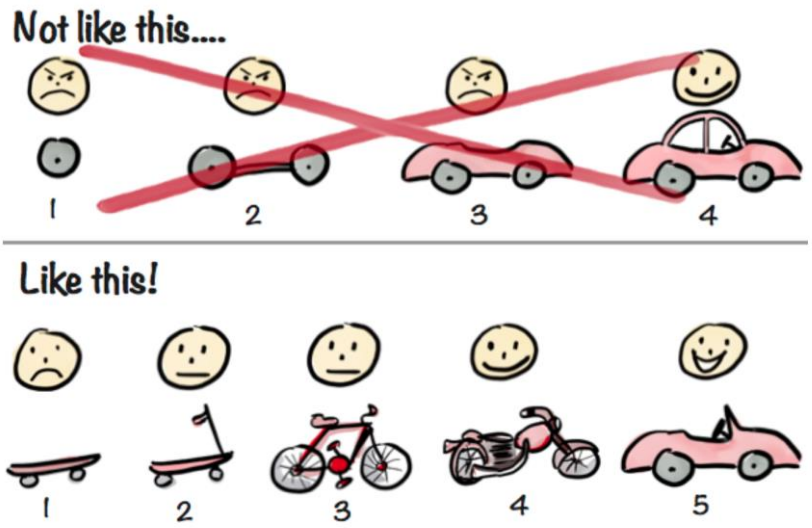


Abbildung 13: MVP-Entwicklungsbeispiel von Kniberg

Der größte Vorteil an dieser Methode liegt in der Validierung der Teilschritte seitens des Kunden. Da dem Kunden Zwischenlösungen vorgestellt werden, ist der Wahrscheinlichkeit von Fehlentwicklungen entgegengewirkt.

3 Trend zur dezentralen Produktionssteuerung

Das Ziel in diesem Kapitel ist die Untersuchung des Trends zur dezentralen Produktionssteuerung. Warum soll dezentral gesteuert werden und was sind die Vorzüge gegenüber der altbewährten zentralen Steuerung? Angefangen mit der Erklärung allgemeiner Grundzüge der Produktionsplanung und -steuerung sollen die wesentlichen Merkmale und Stärken/Schwächen der dezentralen Steuerung aufgezeigt werden. Zudem soll betrachtet werden, ob eine dezentrale Steuerung von FTS vorteilhafter ist.

3.1 Einführung in die Produktionsplanung und -steuerung

Eine Produktion lässt sich als ein Transformationsprozess beschreiben, worin ausgehend von einem eingesetzten Input der gewünschte Output erwartet wird. Ein Transformationsprozess im Falle einer Produktion besteht aus dem Input in Form von Produktionsfaktoren, wie Rohstoffe oder Ressourcen, und dem Output in Form von Produkten (Abbildung 14). Der Erfolg dieses Transformationsprozesses hinsichtlich der logistischen Zielgrößen wie hohe Termintreue, kurze Durchlaufzeit, hohe Auslastung, niedriger Bestand und geringe Kosten (Nyhuis & Wiendahl, 2012, S. 10) ist wesentlich davon abhängig, in welcher Art und Weise dieser Prozess geplant und gesteuert wird. Es handelt sich dabei konkret um die Produktionsplanung und -steuerung (PPS). Bei erfolgreichem Einsatz steigt die Produktivität der gesamten Produktion und umgekehrt sinkt sie bei einer ungeeigneten Steuerungsweise (Dangelmaier, 2009, S. 3).

Abbildung 14: Transformationsprozess der Produktion[12]

Anfang der 1980er Jahre wurde der Begriff der Produktionsplanung und -steuerung geprägt und ist seitdem ein essentieller Bestandteil der Wissenschaft und der Industrie (Praxis), um Material- und Zeitwirtschaft übergreifend beschreiben und angehen zu können. In diesem Zusammenhang kooperieren beide Seiten und adaptieren ihre gewonnen Erkenntnisse gegenseitig (Dangelmaier, 2009, S. 3).

[12] (Dangelmaier, 2009, S. 3)

Die Produktionsplanung (PP) lässt sich hierarchisch in drei Arten unterteilen. Zum einen gibt es die **strategische Produktionsplanung**, welches die Aufgabe hat eine wettbewerbsfähige Produktion zu etablieren oder eine bestehende aufrechtzuerhalten. Die **taktische Produktionsplanung** bestimmt die zu fertigenden Produkte und trifft Entscheidungen über deren Gestaltung. Außerdem werden die Personal- und Betriebsmittelkapazitäten und die organisatorischen Angelegenheiten der Produktion behandelt. Eine andere Bezeichnung für die Produktionsplanung und -steuerung (PPS) ist die **operative Produktionsplanung**, in der ausgehend der strategischen und taktischen PP der bestmögliche Einsatz der Produktionsfaktoren zu realisieren ist, um dem Unternehmen einen wirtschaftlichen Erfolg zu erbringen (Dangelmaier, 2009, S. 9). Lödding weist darauf hin, dass ein guter und realistischer Produktionsplan für eine hohe Zielerreichung notwendig ist und dieser Plan, vorausgesetzt bei Vorhandensein einer funktionierenden Produktionssteuerung, umzusetzen gilt (Lödding, 2016, S. 2).

Im weiteren Teil der Arbeit wird aufgrund des Themenschwerpunktes der Aspekt der Produktionssteuerung näher betrachtet. Die wesentliche Aufgabe der Produktionssteuerung ist die Umsetzung der Produktionsprogrammplanung, wohingegen in der heutigen Praxis die beiden Phasen der Planung und der Steuerung in PPS-Systemen (z.B. ERP) gemeinsam agieren. Die Auftragsfreigabe ist die Kernaufgabe, in der die Planung so gestaltet wird, dass jegliche Kapazitätsengpässe vermieden werden. Darüber hinaus gewährleistet es die Verfügbarkeit der Betriebsmittel und Werkstoffe. (Wikis der Freien Universität Berlin, Grote, 2013). Westkämper und Bauernhansl beschreiben die Ziele der Produktionssteuerung folgendermaßen: *„Ihr Ziel ist die termin-, mengen- und qualitätsgerechte Lieferung von Produkten an die jeweiligen Kunden mit minimalen Beständen und Vorräten an Material, unfertigen und fertigen Erzeugnissen und zugleich der Maximierung der Nutzung vorhandener personeller und technischer Ressourcen.“* (Westkämper & Bauernhansl, 2014, S. 13)

In der Produktionssteuerung werden, abhängig von dem Zentralisierungsgrad, zwischen folgenden Verfahren der Steuerung unterschieden (Grinninger, 2012, S. 117):

- zentral organisierte Verfahren
- bereichsweise zentral organisierte Verfahren
- dezentral organisierte Verfahren

Der Unterschied zwischen diesen Verfahren ist die Art der Auslösung eines Fertigungsauftrages und wie dieser Fertigungsauftrag zu steuern ist. Grundlegend existieren zwei Arten der Auftragsauslösung. Zum einen gibt es das plangesteuerte Push-Prinzip, welches als das Schiebeprinzip bezeichnet wird. Darin werden die Folgeaufträge, unabhän-

gig der Betrachtung nachfolgender Fertigungs-/Arbeitsstationen, durch den Input ausgelöst. Zum anderen gibt es das bedarfsgesteuerte Pull-Prinzip, welches als das Ziehprinzip bezeichnet wird. Eine Auftragsauslösung in einem Ziehprinzip wird meist mit der dezentralen Steuerung in Verbindung gebracht, da die Aufträge, beispielsweise durch den Einsatz von Kanban-Karten, selbst ausgelöst werden. Nachfolgend ist in Tabelle 2 eine Gegenüberstellung ihrer Merkmale in Betracht zur Auftragsauslösung und ihres Zieles dargestellt (Grinninger, 2012, S. 118 f.).

Tabelle 2: Gegenüberstellung von Merkmalen zur Auftragsauslösung

	Auftragsauslösung	**Ziel**
Push-Prinzip (planbedarfsorientiert)	Der Input löst die nächsten Arbeitsschritte aus.	Den Auftrag so steuern, dass der Auftrag zum Liefertermin fertiggestellt ist.
Pull-Prinzip (verbrauchsorientiert)	Der Bedarf an der vorgelagerten Arbeitsstation löst den Auftrag entgegen der Materialflussrichtung aus.	Innerhalb einer bestimmten Zeitspanne den Bedarf sicherstellen.

3.2 Arten der Produktionssteuerung

Wie in dem vorigen Abschnitt beschrieben, erfolgt die Steuerung einer Produktion entweder zentralisiert oder dezentralisiert. Nachfolgend sollen die beiden Arten der Produktionssteuerung kurz in ihren wesentlichen Merkmalen erläutert und anschließend einer kritischen Betrachtung unterzogen werden. Vor allem gilt es zu beantworten, ob eine dezentrale Steuerung für fahrerlose Transportsysteme (FTS) vorteilhafter ist, da sie in der Regel zentral gesteuert werden (vgl. Abschnitt 2.1.1).

3.2.1 Zentrale Produktionssteuerung

Die zentrale Produktionssteuerung ist als die altbewährte und die üblicherweise eingesetzte Steuerung zu beschreiben. Die Basis dafür ist das MRP II-Konzept (Manufacturing Resources Planning), welches im deutschen Raum mit dem Begriff PPS-System gleichzusetzen ist (Kurbel, 2016). Dazu gehören Verfahren, in denen ausgehend einer zentralen Organisationseinheit die Produktion hinsichtlich eines Kundenauftrages bis zur Fertigungsebene geplant und der Fortschritt kontinuierlich überwacht wird, um bei Abweichungen aus der zentralen Ebene eingreifen zu können, wodurch folglich ein neuer aktualisierter Plan erstellt wird (Fischer, 1997, S. 121). Die zentrale Planungsinstanz plant alle notwendigen Aktivitäten innerhalb der Produktion und trifft die Entscheidungen, welches genauso auch die Einbindung weiterer Netzwerkpartner betrifft (Schuh & Stich,

2012, S. 102). Mit der Zentralisierung wird das Ziel verfolgt, Systeme zu vereinen und somit den Koordinationsaufwand zu minimieren (Schuh & Stich, 2012, S. 302).

3.2.2 Dezentrale Produktionssteuerung

Mit dem Beginn der industriellen Revolution 4.0 ist der Begriff einer dezentralen Steuerung mehr in den Fokus gerückt. Dabei wurde die Dezentralisierung von Planungs- und Steuerungsaufgaben als ein wesentlicher Bestandteil in der Diskussion um die Optimierung der Produktion in den vergangenen Jahren einbezogen. (Loos & Allweyer, 2013, S. 83). Die betriebswirtschaftlichen Ziele sind mit den heutigen technologischen Möglichkeiten weitaus einfacher zu erreichen als zuvor. So können bereits bestehende Technologien, wie beispielsweise RFID oder Sensorik, in der Produktion auf eine vollkommen neue Art und Weise eingesetzt werden, wobei stets der betriebswirtschaftliche Aspekt zu beachten gilt, um keine Fehlinvestitionen in IT-Systeme zu tätigen (Jahn, 2017, S. 5) (vgl. 2.2).

Ein Kernelement dieser vierten industriellen Revolution ist der Begriff der dezentralen Steuerung mittels Cyber-Physischer Systeme (CPS). Dabei handelt es sich um ein System, in welchem eine situative Selbststeuerung stattfindet. Ein ständiger Informationsaustausch zwischen „intelligenten Objekten“, welche während des Produktionsprozesses anhand vordefinierter Regeln eigenständig Entscheidungen treffen. In einem CPS sind die Produktionsressourcen räumlich verteilt. Produktionsbereiche werden in einem Netzwerk zusammengeschlossen und arbeiten eigenständig. Durch die Einbindung der Informations- und Kommunikationstechnologie (ITK) (Furmans, 2015, S. 4) und der Cyber-Physischen Systeme (CPS) in die Produktionsumgebung entsteht das Cyber-Physische Produktionssystem (CPPS), welches die intelligente Fabrik (Smart Factory) bildet (Botthof & Hartmann, 2015, S. 99 f.).

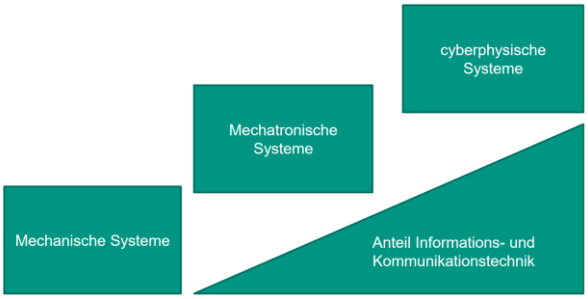


Abbildung 15: Einfluss des Einsatzes der ITK[13]

[13] (Furmans, 2015, Fol. 4)

In einer dezentralen Steuerung ist das Ziel, unabhängig einer zentralen Instanz, autark den gesamten Produktionsablauf selbst steuern zu lassen. Somit entfällt die operative Planung und es findet eine Selbstorganisation statt. Übliche Komponenten in dem Produktionssystem, wie Maschinen, Transportkisten, Transportfahrzeuge und Weitere, sind intelligent und verfügen über die Fähigkeit zu kommunizieren. Somit können sie steuern und gesteuert werden – unabhängig von ihrer örtlichen Lage und des Zustandes, vorausgesetzt, dass die Kommunikation durchgehend möglich ist. Dadurch wird eine Feinplanung in Echtzeit erreicht. Dies wird ermöglicht durch die Anwendung der Komponenten als eingebettete Systeme in dem Internet der Dinge (IoT), worin sie mittels Sensoren und Aktoren ausgestattetet sind und sich als kleine Computer in das Produktionsnetzwerk einbinden (BMBF, 2013, S. 6).

3.3 Kritische Betrachtung beider Arten der Produktionssteuerung

Ein großer Vorteil der zentralen Steuerung ist, dass das gesamte Wissen der Planung an einem Punkt gesammelt und die Entwicklung benötigter Tools von einer zentralen Stelle aus geleitet wird (Jodlbauer, 2008, S. 107). Dadurch wird vor allem möglich Standardisierungen aufzubauen, welche eine erhöhte Planungssicherheit mitbringt. Außerdem beschreibt Schuh die mögliche Notwendigkeit einer Zentralisierung mit der Aussage: *„Aus einer übergeordneten Sicht, beispielsweise der Netzwerksicht, kann eine zentrale Koordination von Planungs- und Steuerungsprozessen, zumindest auf einer groben Planungsebene, sinnvoll oder sogar erforderlich sein.“* (Schuh & Stich, 2012, S. 302).

Kritik zu zentral: Fischer: S. 121 unten

Mit der Dezentralisierung herrscht eine Unabhängigkeit zwischen mehreren Systemen, sodass in dezentralen Prozessabläufen Entscheidungen schneller getroffen werden können, wodurch es möglich ist, ebenso schnell auf Änderungen seitens der Kunden oder des Marktes zu reagieren. Dezentrale Systeme sind durch ihre kurze Entscheidungswege, dem geringen Koordinationsaufwand und der erhöhten Flexibilität gekennzeichnet (Schuh & Stich, 2012, S. 301), welche insbesondere aus Sicht der Produktionssteuerung ein bedeutsamer Vorteil ist, denn je flexibler die Möglichkeiten der Einwirkung in den Produktionsplan sind, desto schneller können die erforderlichen Änderungen umgesetzt werden. Bei einer zentralen Planung hingegen herrscht durch Änderungen oder Störungen eine ständige Alterung der Planungsdaten. Darin wird z.B. mit einer Planung durch ERP, welcher als übergeordnetes IT-System fungiert (zentrale Instanz), in der Regel

eine Produktionsplanung von einem Jahr durchgeführt (vgl. Jahn, S.3). Eine zentrale Produktionssteuerung ist daher entgegen einer dezentralen sehr unflexibel. Entscheidungen erfordern immer eine zentrale Instanz, welches beispielsweise zu größeren Stillständen durch Störungen führen kann. Bei einer dezentralen Produktionssteuerung wird von der zentralen Instanz lediglich ein Liefertermin oder ein anderweitiges Ziel genannt und der Rest wird unter den intelligenten Objekten selbst gesteuert und geplant. Diese Tatsache bringt eine erhebliche Flexibilisierung mit sich, wovon sowohl die produzierenden Unternehmen als auch die Kunden profitieren.

Der Wandel von Produktionssystemen geht dahin, dass nur noch modulare Arbeitssysteme bestehen, welche untereinander kommunizieren. Somit werden ganzheitliche Produktionsprozesse zerlegt und in einzelne dezentrale Einheiten unterteilt. Dies führt dazu, dass die Produktion bei Anpassungen oder Änderungen, ohne des Gesamtsystem zu stören, fortfahren kann (Jahn, 2017, S. 5). Mit der Einführung dieser neuen Struktur in die Fabrikumgebung steigt auch die Komplexität, welches unter Kontrolle gehalten werden muss. Aufgrund dessen müssen diverse Standardisierungen über die gesamte Wertschöpfungskette aufgebaut werden (Botthof & Hartmann, 2015, S. 99 f.).

Kleinere und mittelständische Unternehmen stehen heute vor neuen Herausforderungen, denn die Marktsituation und die Kundenwünsche verändern sich zunehmend. Ohne Optimierungsmaßnahmen in Bezug auf die Kostenreduzierung oder die Einhaltung der Liefertermine wird es schwierig sein, die Position im Markt aufrechterhalten zu können. Dies vor allem aus dem Grund, dass sich die Kundenwünsche immer stärker individualisieren (Fay, 2016) und somit die Produktionssteuerung komplexer und kostenintensiver wird. Außerdem werden immer mehr externe Unternehmen eingebunden und zunehmend in Lieferketten gearbeitet, welche zu einer noch höheren Komplexität der Steuerung führen (M. Günther, 2016).

Ein nicht zu vernachlässigender Punkt in dem Thema der Selbststeuerung ist der Begriff „Mensch“ als Funktion in dem System. Durch zu starke intelligente und autonome Arbeitsprozesse wird der Mensch in seiner Handlungsweise immer mehr eingeschränkt. Daher sollte die Zusammenarbeit zwischen Mensch-Maschine nicht außer Betracht gelassen werden, um keine Nachteile bei den Arbeitnehmern hervorzurufen.

Abschließend sei zu erwähnen, dass eine feste Aussage, ob und wie viel dezentrale Steuerung für jeweilige Unternehmen notwendig ist, gesondert behandelt werden muss. Daher ist die Aussage, dass eine dezentrale Steuerung für alle Unternehmen in gleicher Weise wichtig ist, nicht möglich, denn der Grad der Dezentralität ist anhand gewisser Prozessszenarien individuell zu ermitteln (Lass & Theuer, o. J.).

3.4 Dezentrale Steuerung bei fahrerlosen Transportsystemen

Fahrerlose Transportsysteme werden klassisch zentral gesteuert (vgl. 2.1.1). Die Fahrzeuge erhalten lediglich die Information über die auszuführenden Transportaufträge und führen diese aus, wobei die gesamte Koordinationsaufgabe von der FTS-Leitsteuerung übernommen wird (Ullrich, 2014, S. 181). Daher ist die Anwendbarkeit einer dezentralen Steuerung für FTS zu untersuchen, welches im Zuge dieser Arbeit Einzug im Konzept der Steuerung finden wird.

Aufgrund der Wechselseitigkeit von Anforderungen an die Produktion, wie der Wunsch nach immer mehr kundenspezifischen individualisierten Produkten, werden flexiblere, anpassungsfähigere und intelligentere industrielle Fertigungssysteme gefordert, welche zudem eine große Herausforderung für die Intralogistik bedeuten (Walenta, Schellekens, Ferrein, & Schiffer, 2017, S. 1). Daher sind Materialflüsse sehr effizient zu gestalten, wozu auch die Planung und Steuerung des fahrerlosen Transportsystems gehört. Schnellere Reaktionen und Treffen von Maßnahmen auf ungeplante Ereignisse machen den gesamten Prozess effizienter, als das Warten auf die Mitwirkung einer zentralen Instanz von „außerhalb“. Je größer der Wunsch nach Unabhängigkeit der Fahrzeuge und der damit einhergehenden Intelligenzerweiterung, um erforderliche Informationen beschaffen und Entscheidungen treffen zu können, wird der Aspekt der Zusammenarbeit und die Kommunikation zwischen den Fahrzeugen und anderen peripheren Geräten immer wichtiger (Ullrich, 2014, S. 181).

Feldmann und Wolfgang weisen auf die Veränderungen der Produktionsbedingungen und -umgebungen und die damit nicht mehr zufriedenstellenden Lösungen mittels zentraler Steuerung der fahrerlosen Transportfahrzeuge hin. Aufgrund der zentralen Steuerung wird keine Anbindung an das Geschehen vor Ort realisiert, sodass den Fahrzeugen ein unzureichender Entscheidungsraum gegeben wird. Wenn sie beispielsweise bei den auszuführenden Fahrten auf Probleme stoßen, müssen sie dabei erst von der zentralen Instanz zur Problemlösung unterstützt werden. Außerdem sprechen sie von einem weiteren Nachteil der zentralen Steuerung – dem Single-Point-of-Failure, das durch den Ausfall des zentralen Steuerungssystems einen Totalausfall des gesamten Transportsystems beschreibt (Feldmann & Wolfgang, 2006, S. 261).

Die Auftragsvergabe in einem FTS gehört zu den Hauptaufgaben einer FTS-Leitsteuerung (vgl. 2.1.1). In einer dezentralen Steuerung koordinieren alle FTF die Aufträge selbst, indem sie durch Auftragsverhandlungen die auszuführenden Aufträge erhalten.

Diese Aufgabe wird durch keine zentrale Instanz erledigt, weshalb die autonomen Entitäten, kooperierend und mit vorgegebenen Entscheidungswegen, dieser Aufgabe eigenständig nachgehen (Schwarz, 2014, S. 27).

Zusammengefasst ist die Einbringung einer dezentralen Steuerung für FTS ein richtiger Weg in Richtung Industrie 4.0. FTS als Materialtransportsysteme sind in der heutigen Produktion nicht wegzudenken. Da mit Industrie 4.0 eine Selbststeuerung der Produktion erreicht werden möchte, sind auch FTS davon betroffen und müssen sich zumindest hinsichtlich ihrer Auftragssteuerung verändern.

4 Konzept einer dezentralen Steuerung für FTS

Wie in dem vorigen Kapitel beschrieben, ist der Einsatz einer dezentralen Steuerung besonders vorteilhaft für FTS, worin die FTF unabhängig einer zentralen Instanz mit Transportaufträgen versorgt werden. In dem nachstehenden Konzept wird die dezentrale Produktionssteuerung in Anwendung mit einem fahrerlosen Transportsystem (FTS) vorgestellt. Das zentrale Element des Konzeptes sind „intelligente Transportkisten“, welche ein Auftrag durch die gesamte Fertigung begleiten. Die Transportkisten übernehmen dabei die Durchführung der dezentralen Auftragssteuerung im Produktionsprozess. Neben den Transportkisten gibt es weitere an dem digitalen Produktionsgeschehen teilhabende „intelligente Komponenten“, wie die fahrerlosen Transportfahrzeuge für den Transport der Transportkisten, das Kistenregal für die Auftragsvorbereitung und die Arbeitsplätze.

Das Konzept umfasst die Erfüllung folgender Anforderungen:

- Entwurf eines Testfeldes basierend auf ein fiktives Produktionsszenario
- Modell für Kommunikation der Entitäten
- Auftragsvergabemodell für die fahrerlosen Transportfahrzeuge und Arbeitsplätze zur Buchung von Ressourcen

4.1 Produktionsszenario – Fallbeispiel

Das (fiktive) Produktionsunternehmen „ABC Spielzeuge GmbH“ hält eine führende Position auf dem Markt der Spielzeugfertigung als Produzent für diverse Spielzeugmarken. Um den Unternehmenserfolg zu verbessern, sieht das Management eine Erweiterung der Produktionsstätte mit einem fahrerlosen Transportsystem vor, welches dezentral gesteuert werden soll. Für die Erprobung des neuen Systems möchten sie mit der Erweiterung in einem kleinen Teil der Fertigung beginnen. Das Forschungsteam „HAW Smart Production“ erhält den Auftrag dieses Vorhaben zu modellieren, um die Ergebnisse und Erkenntnisse aus der Arbeit dem Management der „ABC Spielzeuge GmbH“ als Entscheidungsgrundlage zur Verfügung zu stellen.

In dem zu betrachtenden Produktionsabschnitt geht es um die Umformung von Blechteilen zu Spielzeuggeschirr. Der Kunde hat die Möglichkeit aus einer vorgegebenen Produktpalette (Abbildung 16) folgende Produkte zu bestellen:

- Untertasse; Tasse; Teekanne; Tablett

Abbildung 16: Übersicht der Produktpalette

Dabei handelt es sich um keine Fließfertigung, sondern um die Fertigung von Einzelstücken basierend auf dem One-Piece-Flow-Prinzip ohne Mitarbeiteranbindung. Die fahrerlosen Transportfahrzeuge müssen daher Stückgüter in den Transportkisten transportieren, wobei durch das Ersetzen der Mitarbeiter, hinsichtlich ihrer Transportaufgabe, die Bindungsfreiheit genauso beibehalten wird.

Als Ausgangslage für die Aufstellung des Produktionsszenarios dient hierbei das Swimming-Pool-Beispiel aus Abbildung 17 von Prof. Dr. Henner Gärtner aus der Vorlesungsreihe „Digitale Produktion“ (Gärtner, 2018, Fol. 29), worin die Vorstellung einer dezentralen Steuerung mit der Anwendung von FTS stattgefunden hat. In diesem Rahmen wurde eine Fertigungslinie bestehend aus drei Maschinengruppen vorgestellt. In der Darstellung steht jede Zelle für eine Zeitspanne, in der sich die Aufträge (rote Rechteckform) durch die Fertigung bewegen. Die Breite der einzelnen Arbeitsplätze aus den Maschinengruppen wiedergeben die benötigte Bearbeitungsdauer für den jeweiligen Fertigungsschritt. So benötigt die Maschine X1 dreifach länger für den selben Arbeitsgang als die Maschine X2.

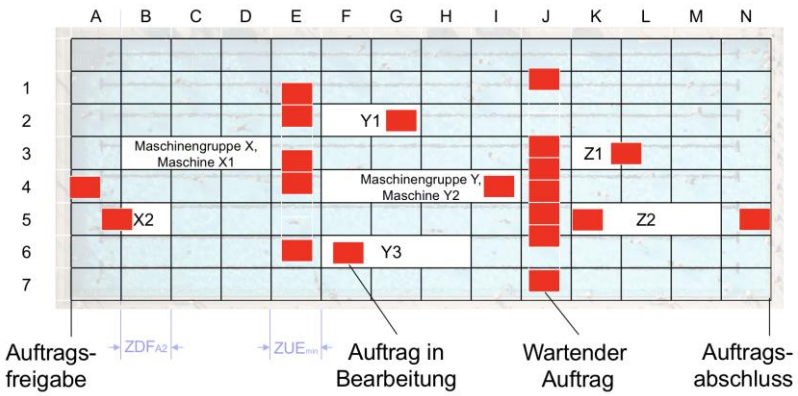


Abbildung 17: Fertigungslinie des Swimmingpool-Beispiels zur dezentralen Steuerung

Demnach beinhaltet das Produktionsszenario insgesamt drei Fertigungsschritte mit sieben Arbeitsplätzen. Der genaue Prozessablauf der Fertigung wird in dem Abschnitt 4.3 mit der Vorstellung des aus dem Szenario entwickelten Testfeldes näher erläutert.

4.2 Grundlegende Konzeptidee

Zur Einführung in das Konzept werden nacheinander alle zu entwickelnden Komponenten erläutert und anschließend mit ihren Funktionen näher beschrieben, welche für die Modellierung des aufgestellten Fallbeispiels und die dezentrale Produktionssteuerung mit FTS notwendig sind. Detailinformationen zu einigen dieser Funktionen werden in den darauffolgenden Abschnitten behandelt. Die einzelnen Komponenten und die zu erfüllenden Funktionen resultieren aus den ersten Projektgesprächen, worin die Anforderungen an das Projekt besprochen wurden. Gemäß diesen Anforderungen wurde das Funktionsspektrum der Komponenten aufgestellt.

Als Entwicklungssystem für alle Komponenten werden Arduino-Mikrocontroller verwendet, um sie mit „Intelligenz“ auszustatten. Mit der Ausrüstung einzelner Komponenten (nachfolgend als Entitäten bezeichnet) in der Produktion durch Mikrocontroller können diese dazu befähigt werden, je nach Zustand und Begebenheit, anhand vordefinierter Routinen und Regeln, Entscheidungen zu treffen und demnach zu agieren. Unter Mikrocontrollern gibt es diverse Modelle, wobei in diesem Konzept durchgehend Arduinos eingesetzt werden. Durch diese Erweiterung wird eine Beteiligung im digitalen Produktionsgeschehen ermöglicht. Es sei zu erwähnen, dass die Intelligenz, von der hier die Rede ist, keine selbst erlernende künstliche Intelligenz ist, sondern eine, die lediglich innerhalb eines festen Routine- und Regelwerks Entscheidungen treffen kann. Nähere Details dazu folgt in dem Abschnitt 4.4 unter dem Themenfeld der M2M-Kommunikation und der Informationsverarbeitung. Daher zunächst eine kurze Übersicht in Tabelle 3 zu den in dem System vorhandenen Entitäten. Jede aufgeführte Kernaufgabe ist mit einem eigenen Arduino zu realisieren.

Tabelle 3: Übersicht der im System vorhandenen Entitäten

Entität	Kernaufgabe
Intelligente Transportkiste	Dezentrale Steuerung des Kundenauftrages
Fahrerloses Transportfahrzeug	Fahrzeug- und Komponentensteuerung
Kistenregal	Zuweisung von Aufträgen an die Transportkisten
Arbeitsplätze	Annahme und Abgabe von Transportkisten

4.2.1 Annahme zur zeitlichen Betrachtung

Vor der Erläuterung der einzelnen Entitäten ist eine Annahme zur zeitlichen Betrachtung für die Modellierung aufzustellen. Eine große Herausforderung in solchen Modellierungen ist es die Zeiten in realer Form zu erfassen. Die Schwierigkeit liegt darin, dass sehr viele, von sich unabhängige und entkoppelte, Entitäten auf Basis der Arduinos gemeinsam in dem System agieren werden. Demnach ist eine alternative Form der Zeiterfassung vorzunehmen. Um die gesamte Betrachtung quantitativ beurteilen zu können, bedingt es einer einheitlichen Basis, womit der zeitliche Aspekt für alle Beteiligten gleichermaßen Anwendung finden kann. Die Idee ist es die Zeit mittels **„Zeitslots“** zu beschreiben. Bei einem Zeitslot handelt es sich um Zeitspannen, die durch das Eintreffen festgelegter Bedingungen abgeschlossen werden. Eine Bedingung für das Eintreffen für den Abschluss eines Zeitslot aus Sicht einer Entität kann dabei beispielsweise die Bewegung eines fahrerlosen Transportfahrzeuges um eine Position bedeuten. Daher spielt es keine Rolle, ob Aktionen zu unterschiedlichen Zeitslots, aus Sicht der realen Zeitbetrachtung, länger oder kürzer dauern. Somit ist gewährleistet, dass unabhängig von den individuellen Abweichungen einzelner Entitäten die zeitliche Betrachtung dennoch qualitativ in Ordnung ist. Eine genaue Erläuterung zu diesem Verfahren folgt in dem Abschnitt 4.5.2.

4.2.2 Auftragssteuernde intelligente Transportkisten

In diesem Konzept eines dezentralen Steuerungsmodells spielen die Transportkisten die Hauptrolle, da sie den aufgegebenen Kundenauftrag selbst darstellen. Ihre Aufgabe ist es den Auftrag durch die erforderlichen Fertigungsschritte zu bewegen, indem durch die Anbringung eines Mikrocontrollers an die Transportkiste eine Teilnahme im digitalen Produktionsgeschehen ermöglicht wird. Sie sind daher die wesentliche Instanz mit der Aufgabe die Produktion dezentral mit Effizienz und Selbstorganisation zu steuern. Diese Aufgabe ist sinnhaft mit dem Besuch einer Person in einem Jahrmarkt zu vergleichen. Der Besucher weiß zu Beginn welche Attraktionen von ihm besucht werden müssen. Doch er steht frei zu entscheiden, wann er welche Attraktion besuchen wird. Sollte an einem Wunschort beispielsweise eine sehr lange Warteschlange bestehen, wählt der Besucher eine andere Attraktion aus, mit dem er dennoch effizient seine Anforderung erfüllen kann. Auf dieselbe Art und Weise bewegen sich die Aufträge in Gestalt der Transportkisten durch die Produktion und planen die nächsten Bearbeitungsschritte der zu fertigenden Produkte ein, um innerhalb gegebener Frist anforderungserfüllend den Prozess abzuschließen. Sie stehen dadurch in der Position des entscheidungszentralen Elementes in dem gesamten System. Demnach werden die Entscheidungen, die das

Produktionsprogramm beeinflussen, durch diese intelligenten Transportkisten gefallen. Sie stehen kontinuierlich in Kommunikation mit den weiteren Entitäten aus dem System.

Diese Art einer dezentralen Steuerung ist entgegen der Beschreibung aus dem Abschnitt 3.1 statt einer bedarfsorientierten Steuerung auf dem Pull-Prinzip zu vergleichen mit dem Push-Prinzip, worin der Auftrag planorientiert gesteuert wird. Die Auftragsauslösung erfolgt durch die Transportkiste, dem Input. Die in den bisherigen Literaturwerken aufgefasste Definition einer dezentralen Steuerung, verglichen mit der Kanban-Methode, ist daher nicht mehr exakt zutreffend, da mit der Vorstellung einer selbstorganisierten Produktion aus dem Industrie 4.0 hat der Begriff der dezentralen Produktionssteuerung eine neue Bedeutung angenommen.

Abbildung 18: Funktionen einer Transportkiste

Träger von (Halb-)Erzeugnissen: Die fahrerlosen Transportfahrzeuge (FTF) benötigen ein Mittel, mit dem die Erzeugnisse durch die Produktion bewegt werden können, wofür Transportkisten zum Einsatz kommen. Zu Beginn eines Auftrages wird dem abzuholenden FTF eine leere Transportkiste übergeben. Die Übergabe der Transportkiste an die FTF geschieht durch ein automatisiertes Regalsystem. Die Idee dabei ist der Einsatz eines automatisierten Lagersystems, woraus mittels eines angeschlossenen Förderbandes die Kiste dem FTF übergeben werden kann. Unmittelbar danach erhält diese Transportkiste die relevanten Auftragsinformationen und begleitet den Auftrag durch den gesamten Produktionsprozess. Nachdem der Produktionsprozess abgeschlossen ist, werden die Erzeugnisse aus der Transportkiste über ein Förderband in ein zugewiesenes Lager transportiert. Die erwähnten Systeme zur Übergabe und Abnahme der Transportkisten sind lediglich Annahmen und kein Bestandteil der Modellierung und sind erst erforderlich, wenn das Konzept in der Realität umgesetzt wird. Daher wird die Übergabe und Abnahme der Transportkiste manuell durchgeführt.

Dezentrale Steuerung der Produktion: Die intelligenten Transportkisten dienen in erster Linie dazu, die Produktion dezentral zu steuern. Angefangen von der Auftragsauslösung bis hin zum letzten Prozessschritt gehen alle Entscheidungen von der Transportkiste aus, indem es in ständiger Kommunikation mit den anderen Entitäten im System ist und diese je nach Bedarf anspricht, um Ressourcen zu buchen. Somit ist beispielsweise die

Transportkiste in der Lage, Kapazitäten eines Arbeitsplatzes zu buchen, um an diesem den nächstfolgenden Bearbeitungsschritt durchführen zu lassen.

Auftragsdaten tragen/aktualisieren: Mit der Transportkiste werden im gesamten Produktionsablauf alle benötigten Daten mitgetragen, ohne dass eine Anbindung an eine weitere zentrale Leitstelle benötigt wird. So werden über den gesamten Prozess entlang auf der Transportkiste Informationen gespeichert und gelesen. Die Informationsträger in Form von RFID-Tags werden auf der Unterseite einer Transportkiste angebracht. Anhand diesem ist es möglich an allen relevanten Stationen benötigte Informationen abzurufen oder Informationen zu aktualisieren. So können an einem Arbeitsplatz die auszuführenden Arbeitsschritte gelesen und am Ende des Prozesses die erfolgte Bearbeitung bestätigt werden. Sobald ein Auftrag begonnen wurde, wird der Auftragsstatus aktualisiert. Mit jedem weiteren Arbeitsschritt findet ebenso eine Aktualisierung des Bearbeitungsstandes statt.

4.2.3 Kistenregal

Das Kistenregal erfüllt die Aufgabe der Auftragsvorbereitung, in der die Verwaltung aller alle offenen Kundenaufträge geführt werden. In dem Produktionsszenario sind insgesamt sieben Arbeitsplätze vorhanden und für den Fall, dass alle Arbeitsplätze aktiv arbeiten und zudem beide FTF (vgl. 4.2.4) einen Transportauftrag ausführen, sollten im Inventar mindestens 10 Transportkisten vorhanden sein, um ausreichende Verfügbarkeit für maximale Kapazitätsauslastung gewährleisten zu können. Das Kistenregal sollte bei Erreichen eines Bestandes von drei Transportkisten durch eine externe Arbeitskraft mit den abgearbeiteten Transportkisten aufgefüllt werden.

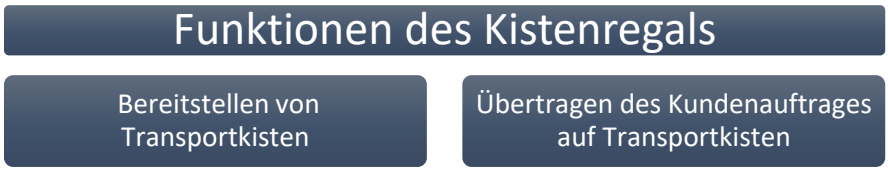


Abbildung 19: Funktionen des Kistenregals

Bereitstellen von Transportkisten: Die Bereitstellung aller Transportkisten an die FTF erfolgt durch das System des Kistenregals. Nach Abschluss jeden Kundenauftrages werden die RFID-Tags der Transportkisten umgehend zurückgesetzt, damit es erneut in das Kistenregal aufgenommen werden kann.

Übertragung des Kundenauftrages auf Transportkisten: Um der Transportkiste den zu bearbeitenden Kundenauftrag mitzuteilen, werden die auftragsspezifischen Daten auf

das RFID-Tag der Transportkiste geschrieben. Hierfür wird dem Arduino-Mikrocontroller des Kistenregals ein RFID-Lesegerät angebracht.

4.2.4 Fahrerlose Transportfahrzeuge

Fahrerlose Transportfahrzeuge sind in einem fahrerlosen Transportsystem unabdingbar. Daher ist hier nicht die Frage ob FTF benötigt werden, sondern wie viele davon in diesem Konzept einzusetzen sind. Die Anzahl der Fahrzeuge sind **vorerst auf max. zwei** zu beschränken, da das Szenario für eine höhere Anzahl zu klein dimensioniert ist und nicht mehr sinnvoll wäre. Einer Erweiterung hingegen steht nichts im Wege, sofern das Szenario angepasst wird. Die Fahrzeuge werden mit dem Arduino-System zu entwickeln. Näheres zu der Entwicklung eines Fahrzeuges folgt in dem fünften Kapitel. Die zu erfüllenden Funktionen eines FTF im Rahmen dieser Arbeit, unter anderem durch die Definition des VDI (vgl. 2.1), werden nachstehend beschrieben.

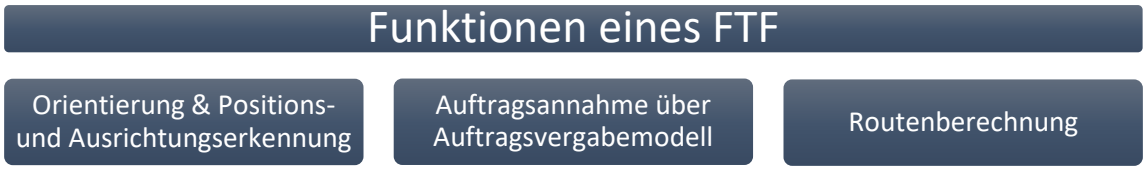


Abbildung 20: Funktionen eines FTF

Orientierung: Die FTF müssen sich während des Fahrens auf dem aufzustellenden Testfeld (vgl. 4.3) mit einer entsprechenden Methode Bewegungssteuerung orientieren können. Vorweg sei zu erwähnen, dass die ausgewählte Methode auf eine spurgebundene Art erfolgen wird, da aufgrund des Modelliervorhabens mit dem Arduino-System und dem zeitlichen Rahmen dieser Arbeit besser geeignet ist (vgl. 5.2.2).

Positions- und Ausrichtungserkennung: Aufgrund dessen, dass keine besondere technische Lösung wie GPS[14] für die Positions- und Ausrichtungserkennung verwendet wird, ist eine Methode zu entwickeln, wodurch die FTF ihre aktuelle Position auf dem aufzubauenden Testfeld zu jeder Zeit kennen und laufend zurückmelden (vgl. 5.2.3).

Auftragsannahme über Auftragsvergabemodell: Mit der Entwicklung eines Auftragsvergabemodells kommt ein essentieller Bestandteil ins Konzept, welches für die dezentrale Steuerung notwendig ist. Dadurch wird ermöglicht, dass alle intelligenten Entitäten in dem System – unabhängig einer zentralen Instanz – untereinander in Auftragsverhandlungen treten können. Mit der Auslösung eines Auftrages durch eine Transportkiste tre-

[14] GPS: Global Positioning System (Globales Positionsbestimmungssystem)

ten alle im System vorhandenen FTF in eine Auktion, gemäß der Beschreibung im Auftragsvergabemodell, und konkurrieren gegeneinander um den Erhalt des Transportauftrages. Somit wird dezentral entschieden, durch welchen FTF ein Auftrag auszuführen ist (vgl. 4.5).

Routenberechnung: Durch den Einsatz eines Algorithmus zur Routenberechnung, wird das FTF fähig erforderliche Transportrouten zu ermitteln. Bei der Wahl des Algorithmus wurde auf die Eigenschaften der Ermittlung der kürzesten Route und die schnelle Berechnungsdauer Wert gelegt, da der Arduino begrenzte Ressourcen zur Verfügung stellt. Die Entscheidung fällt dabei auf den A*-Algorithmus, wobei der Algorithmus noch zu erweitern gilt, sodass eine kollisionsfreie Routenberechnung mehrerer FTF möglich wird.

4.2.5 Arbeitsplätze

Die FTF erhalten durch die Transportkiste die Anfrage für die Ausführung eines Transportauftrages. Bevor diese Anfrage an die FTF erfolgen kann, ist zunächst einmal die nächst anstehende Arbeitsstation des relevanten Auftrages näher zu betrachten. Wenn beispielsweise ein Teilerzeugnis aus der Maschinengruppe X zu der Maschinengruppe Y transportiert werden muss, gilt es zu prüfen, welcher der Maschinen aus der Maschinengruppe Y verfügbar ist. Somit wäre ein Auftrag vorerst abzulehnen, sofern für einen Auftrag kein weiterführender Arbeitsplatz planbar ist. Erst nach erfolgreicher Buchung der Ressourcen eines Arbeitsplatzes treten die FTF in eine Verhandlung im Rahmen des Auftragsvergabemodells (vgl. 4.5) ein. Die Arbeitsplätze sind demnach ebenso in das Produktionsnetzwerk einzubinden, denn auch sie beeinflussen die Entscheidungen der Transportkiste, indem Sie aktiv an der Ressourcenplanung teilnehmen.

Abbildung 21: Funktionen eines Arbeitsplatzes

Ressourcen bereitstellen: Alle Arbeitsplätze werden durch die Buchung von Kapazitäten gesteuert. In einem 3-Schicht-Betrieb stellt die Ressource des Arbeitsplatzes, bei einer Annahme, dass ein Zeitslot einer Minute entspricht, jeweils eine Kapazität von 3600 ZS zur Verfügung. Die Transportkisten befinden sich im Produktionsprozess auf Ebenen (Abbildung 22), sodass sie für eine Buchungsanfrage nur die nachstehende Ebene betrachten können. Daher ist im Vorfelde nicht möglich den übernächsten Arbeitsvorgang

zu buchen. Durch dieses Vorgehen wird erreicht, dass in kleineren Schritten eine Detailplanung stattfindet, um bei ungeplanten Ereignissen nicht weitere Planungen zu verwerfen und somit die Planungsunsicherheiten zu vermeiden.

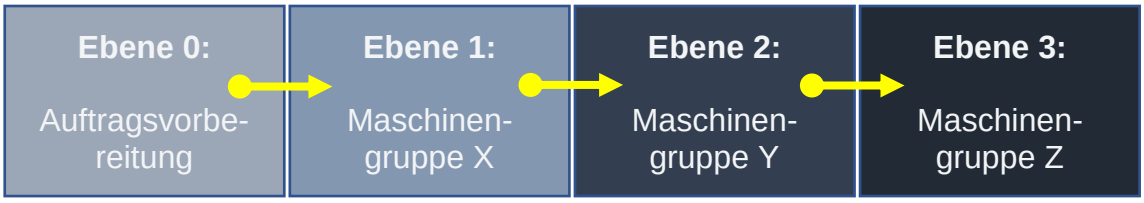


Abbildung 22: Betrachtungsebenen für Buchungsanfrage von Transportkisten

Arbeitsvorgang durchführen: Ankommende Transportkisten müssen manuell auf die Position des Arbeitsplatzes hingestellt werden, in denen die vorgesehenen Arbeitsvorgänge durchgeführt. Wie zuvor erwähnt werden keine aktiven Arbeiten in dieser Modellierung vorgenommen. Die Transportkiste verweilt daher für die vorgesehene Bearbeitungszeit auf der Position des Arbeitsplatzes und plant nach Abschluss der Bearbeitung den nächsten Fertigungsschritt ein.

Aktualisierung des RFID-Tags: Nach abgeschlossener Bearbeitung erfolgt eine Zustandsrückmeldung auf dem RFID-Tag der jeweiligen Transportkiste. Dazu wird auf vorgesehener Stelle die Information erfasst.

4.3 Testfeld zu dem Produktionsszenario

Das Testfeld ist jene Komponente der Modellierung, auf welchem der gesamte Prozess durchgespielt wird. An dieser Stelle sei vorerst zu erwähnen, dass mit dem Begriff „Testfeld“ eine Plattform zu verstehen ist, worin ein Produktionsabschnitt abgebildet ist und dieses dazu dient, dass darin Produktionsszenarien mit einem fahrerlosen Transportsystem untersucht werden können.

Die betrachtete Fertigungslinie besteht, wie bereits in 4.1 erwähnt, aus **drei Maschinengruppen** mit **drei Fertigungsschritten**, die ein zu fertigendes Produkt durchlaufen muss. In der nachfolgenden Tabelle 4 sind die Prozessschritte und Bearbeitungszeiten für die Fertigung aufgeführt. Dabei hat jede Maschine der Maschinengruppen unterschiedliche Bearbeitungszeiten, wodurch es zu variablen Durchlaufzeiten je Auftrag kommen kann. Die Bearbeitungszeiten sind in ZS (Zeitslots) angegeben und entsprechen in der jeweiligen Dauer dem Aufbau aus der Abbildung 17, welches die Grundlage für das Produktionsszenario darstellt. Diese sind gültig für alle Produkte. Die Bearbeitungszeiten sind feste maschinenbezogene Werte, wobei jeder Artikel eine spezifische Äquivalenzzahl besitzt, welches mit dem Maschinenwert multipliziert wird. Durch die

Äquivalenzzahl wird der Maschinenwert mit einer prozentualen Gewichtung verändert, wobei der Wert „1“ für keine Veränderung steht. Das Ergebnis wird gerundet, um diesem exakt einem Zeitslot zuordnen zu können. Durch dieses Vorgehen wird eine realistische und dynamische Simulation ermöglicht. Die Äquivalenzzahlen sind durch Schätzung der Artikelkomplexität in Abbildung 23 aufgeführt. So beträgt die Bearbeitungsdauer einer Untertasse an dem Arbeitsplatz „AP_X1“ nur 15 ZS. Im weiteren Verlauf der Arbeit wird für die Äquivalenzzahl die Bezeichnung „Faktor“ verwendet.

Tabelle 4: Prozessschritte und Bearbeitungszeiten der Arbeitsplätze

	Arbeitsplatz	Bearbeitungszeit [ZS]	Arbeitsvorgang
Maschinengruppe X	AP_X1	30	Spanende Bearbeitung
	AP_X2	10	
Maschinengruppe Y	AP_Y1	20	Umformen
	AP_Y2	40	
	AP_Y3	30	
Maschinengruppe Z	AP_Z1	10	Montage-arbeitsplatz
	AP_Z2	30	

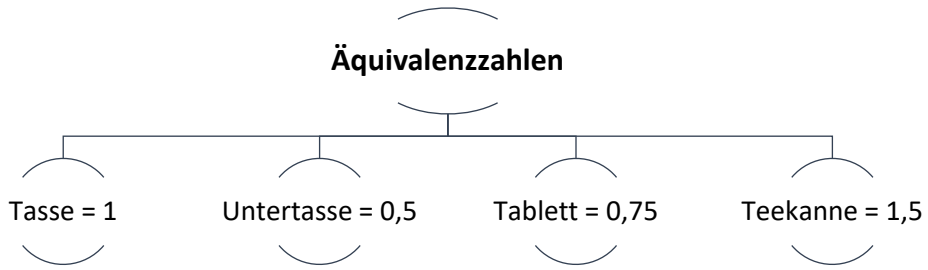


Abbildung 23: Äquivalenzzahlen der Produkte für die Berechnung der spezifischen Bearbeitungszeiten

Konzipiert ist das Testfeld in einer Gitterstruktur mit schwarzen Linienspuren. Dazu wurde das Swimmingpool-Beispiel in eine Linienstruktur transformiert (Abbildung 24), in die Fahrwege innerhalb der Fertigungslinie mit schwarzen Spuren dargestellt sind. Alle Koordinaten darin, außer die der Arbeitsplätze, sind befahrbar. Die Anfahrpunkte der Arbeitsplätze sind mit einem grünen Punkt gekennzeichnet. Diese Darstellung ist ebenso gleichzusetzen mit einer Matrix, wie bereits im Grundlagenteil zu den Graphalgorithmen beschrieben. Sofern die Position befahrbar ist, erhält es den Zellenwert 1 und bei einem Hindernis den Wert 0. Spurverbindungen können nur zwischen befahrbaren Punkten existieren. Hindernisse sind in zwei Arten zu unterteilen. Zum einen fest definierte Hindernisse, wie Arbeitsplätze, und zum anderen variable Hindernisse, welche durch die momentane Belegung der Positionen durch die FTF erzeugt werden. Andere Arten von Hindernissen, wie Produktionsmitarbeiter, werden an dieser Stelle vernachlässigt.

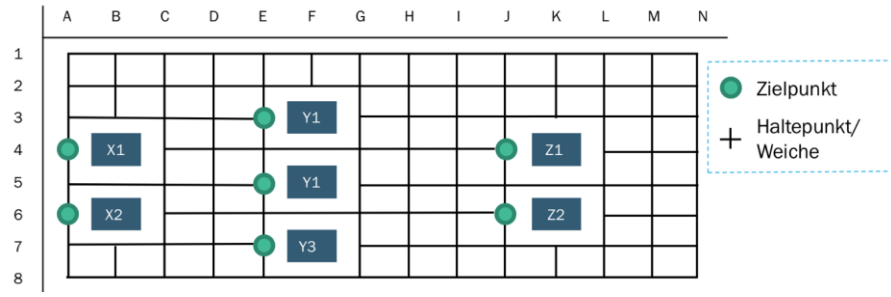


Abbildung 24: Transformation des Swimmingpools auf die Linienstruktur

Jede der Positionen, auch Knoten genannt, dieser Matrix hat den gleichen Abstand zu seinen Nachbarpositionen. Demnach ist die Kantengewichtung identisch und mit dem Wert „1“ anzunehmen. Die Spuren besitzen in Abständen von mindestens einheitlichen 30 cm jeweils einen Positionsmarker, welche gemäß den Indizes einer Matrix fortlaufend geführt wird (Abbildung 25). Die Abstandsgröße ist hinsichtlich der Objektgröße des entwickelten FTF ausgewählt. Da das Testfeld modular aufzubauen ist, kann sich im Laufe der Entwicklung bei Bedarf für einen größeren Abstand entschieden werden. Als Positionsmarker werden RFID-Tags eingesetzt, worin die Informationen zu der jeweiligen Position enthalten sind. Nähere Informationen zu dem Aufbau dieser Positionsmarker folgt in dem Abschnitt 5.2.3.

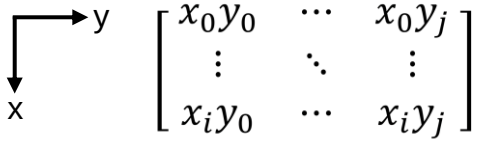


Abbildung 25: Notation für Positionsmarken

Der Transport und Aufbau des gesamten Testfeldes ist einfach zu gestalten. In

Abbildung 26 ist das aufzubauende Testfeld in einer 3D-Ansicht skizziert. Dabei stellen die kleinen abgerundeten Rechtecke mögliche Anfahrpunkte für die FTF dar. Gemäß der aufgestellten Notation sind die zugehörigen Positionsmarker mit den Koordinaten beschrieben. Aufgrund dessen, dass die FTF sich durch IR-Sensoren auf der schwarzen Spur orientieren, müssen die RFID-Tags unter der schwarzen Linienspur angebracht werden, da die weiße Farbe der RFID-Tags die Sensoren täuschen können. Alternativ gibt es auch RFID-Tags in dunkler Farbe, aber die Größe, wobei die Größe nicht größer als die der Spur sein darf.

Ein Arbeitsplatz entspricht einem Platzbedarf von exakt zwei Punkten des Feldes. Die Arbeitsplätze gehören zu jenen Komponenten im System, die als feste Hindernisse in der Routenplanung gelten. Somit stehen die Koordinaten dieser Arbeitsplätze für das Fahren der FTF nicht zur Verfügung. Jeder dieser Arbeitsplätze besitzt am Anfang eine

Quelle für die Warenannahme und am Ende eine Senke für die Zurverfügungstellung der (Halb-)Erzeugnisse an die FTF. Die fertiggestellten Erzeugnisse werden nach dem letzten Arbeitsgang zur Position **„x6y9“** transportiert, um diese auf das (fiktive) Förderband des automatisierten Regalsystems abzulegen, woraus die Versandvorbereitung an den Kunden erfolgt. An dieser Stelle nimmt die Transportkiste wieder den Leerzustand ein, indem der RFID-Tag zurückgesetzt wird. Das FTF begibt sich nach der Übergabe der Transportkiste zu seiner Ladestation zurück, sofern kein weiterfolgender Auftrag eingeplant ist.

Damit das Testfeld nicht überdimensioniert ist, wurde das in die Gitterstruktur transformierte Fertigungslinie hinsichtlich der vorhandenen Knoten komprimiert, wodurch das finale Testfeld eine Dimension von G = (56, 74) besitzt. Das Testfeld weist 56 Knoten und 74 ungerichtete Kanten auf. Somit sind insgesamt 56 Positionen durch die FTF befahrbar, welche durch dementsprechende 56 RFID-Tags gekennzeichnet sind. Das Testfeld muss modular gestaltet sein, um weiterfolgende Erweiterungen gewährleisten zu können. Mit einem fest definierten Feld ist es nicht möglich umfassende Forschungsarbeit zu erreichen. Sollten in Zukunft weitere Szenarien erprobt werden, so ist dieses von essentieller Natur. Unabhängig davon ist der modulare Aufbau von Produktionsstätten eine Kernidee der dezentralen Produktion in Industrie 4.0 (vgl. 3.3). Daher gestaltet sich das praktische Aufbauen des Testfeldes so auf, dass bei nur die RFID-Tags als solches existieren und je nach Zeit und Ort das Testfeld mit einer gewissen Vorbereitungsdauer aufgebaut werden muss. Zusätzlich werden neben den RFID-Tags schwarzer PVC-Band mit einer Breite von ca. 20 mm benötigt, um die Fahrwege darzustellen.

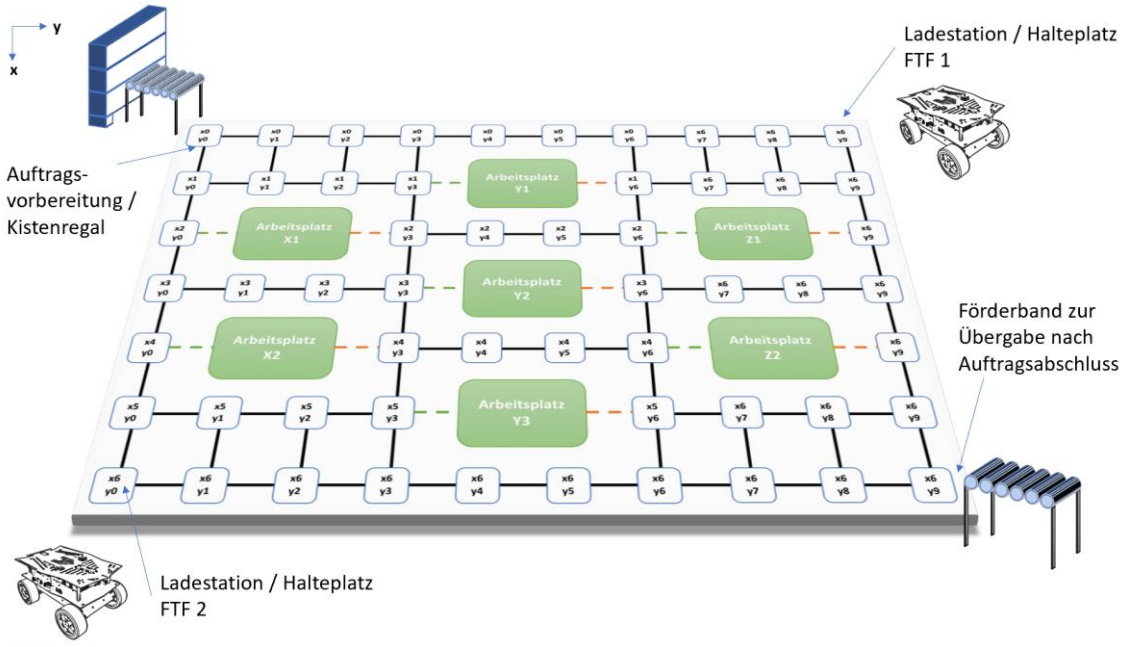


Abbildung 26: 3D-Darstellung des Testfeldes

4.4 M2M-Kommunikation und Informationsverarbeitung

4.4.1 M2M-Kommunikation über MQTT-Protokoll

Die dezentrale Steuerung einer Produktion erfordert die Vernetzung aller Entitäten in dem System, um zwischen allen einzelnen Beteiligten eine Verbindung herzustellen, in der sie fähig sind miteinander zu kommunizieren. Dies ist erforderlich für das Treffen von gezielten und präzisen Entscheidungen für jegliche Ereignisse innerhalb der Produktion. Die Kommunikation ist zwischen den Transportkisten, den FTF, den Arbeitsplätzen und dem Auftraggeber (erstmalige Auftragseingabe über Software) aufzubauen, wobei die wichtigste Kommunikation zwischen den Transportkisten, die die Transportaufträge auslösen, und den FTF, die mit der Ausführung dieser Transportaufträge beauftragt werden, herrscht. Die Kommunikationsfähigkeit dient in erster Linie für die Durchführung von Auftragsverhandlungen, initiiert durch die Transportkiste, mit den Arbeitsplätzen und den FTF, wofür im Abschnitt 4.5 ein Auftragsvergabemodell beschrieben wird.

Als M2M-Kommuniaktionsprotokoll wird das MQTT verwendet. Die Entitäten stehen mit diesem Protokoll in unabhängiger und entkoppelter Weise in Kommunikation. Unabhängigkeit bedeutet, dass keines der Beteiligten zwingend anwesend oder gar dem gegenüber vorher bekannt sein muss. Es besteht zu jedem Moment die Möglichkeit Beteiligte hinzuzufügen oder zu entfernen, ohne dass das System dadurch beeinflusst wird. Aufgrund der Architektur bedeutet dies folglich eine Flexibilität hinsichtlich der Erweiterung des Gesamtsystems, wodurch bspw. weitere FTF ohne große Probleme in das aufgestellte Konzept integriert werden können.

Das Herzstück in dem MQTT-Protokoll ist der MQTT-Broker als verbindendes Element aller Entitäten im System. Jegliche Nachrichten des Publisher-Clients werden über den Broker an die weiteren Clients, die dem relevanten Topic als Subscriber folgen, gesendet. Ein MQTT-Broker kann zum einen als ein fertig einsetzbares System eingekauft werden, wofür es auf dem Markt diverse Anbieter gibt. Zum anderen besteht auch die Möglichkeit mittels eines Servers einen MQTT-Broker laufen zu lassen. Letzteres ist ohne größere Mühe und kostenlos zu realisieren, weshalb dies im weiteren Teil dieses Konzeptes zur Anwendung kommen wird. Der MQTT-Broker-Server des Eclipse Foundations mit der Bezeichnung „Eclipse Mosquitto“ (Beaton, 2018) ist eine passende Lösung hierfür. Der Hersteller stellt „Mosquitto“ folgendermaßen vor: *„Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers.”* Neben dem Source-Code werden Installationen für diverse Plattformen angeboten. Darunter gehören beispielsweise Microsoft Windows, Mac OS, diverse Linux

Distributionen, Raspberry Pi und iPhone. Im Downloadbereich der offiziellen Seite von Eclipse sind Anleitungen für die Einrichtung auf den jeweiligen Plattformen aufgeführt („Eclipse Mosquitto - Download“, 2018). Nachdem der Server gestartet ist, können die Clients eine Verbindung zu dem Broker aufbauen und anfangen gegenseitig Nachrichten zu senden/empfangen.

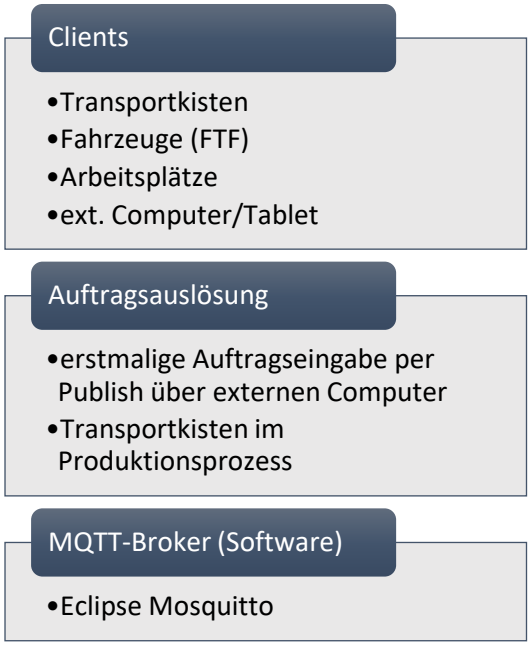


Abbildung 27: Komponenten im MQTT-Protokoll

In Abbildung 27 sind die vier miteinander zu kommunizierenden Clients aufgelistet. Der externe Client in Form eines Computers übernimmt die erstmalige Auftragsauslösung für die Produktionsaufträge. Dabei spielt es keine Rolle, ob ein PC oder eine anderweitige Hardware genutzt wird, denn das Protokoll ist Server-basiert und es genügt ein entsprechend eingerichtetes internetfähiges Gerät für die Anbindung an den MQTT-Server. Daher ist für diese Aufgabe kein Mikrocontroller zu verwenden. Hierfür eignet sich die Software des Entwicklers „Jens Deter“ mit der Bezeichnung „MQTT.fx“ aufgrund der einfachen Anwendbarkeit optimal. Die restlichen drei Clients besitzen einen Arduino, womit sie als eigenständige Entitäten im System je nach Bedarf kommunizieren können. Die Einbindung der Arduinos an den MQTT-Broker erfolgt mittels einer Bibliothek, auf dem im Grundlagenteil bereits verwiesen worden ist (vgl. 2.2.4).

Hinsichtlich des zu verwirklichenden Prozesses bestehen die Kommunikationswege aus den in der Abbildung 28 dargestellten Verbindungen. Daraus sind ebenso die Kommunikationspartner abzuleiten. Die Transportkiste als zentrales Element im gesamten System steht mit allen Entitäten im System in Verbindung. Angefangen mit dem Erhalt eines Auftrages aus der Warteliste der offenen Aufträge, bis hin zur Anfrage der Arbeitsplätze und FTF, um die für die Fertigung des Auftrages notwendigen Ressourcen zu buchen, wird eine selbststeuernde Produktion mittels dieser intelligenten Transportkiste ermöglicht.

Der externe Computer trägt die Aufgabe das System mit Produktionsaufträgen zu versorgen. Diese erstmalige Eingabe des Auftrages durch ein externes Gerät ist auch gleichzusetzen mit dem Eintragen eines Kundenauftrages in ein ERP-System, womit in einer zentralen Produktionssteuerung die Produktion gesteuert wird. Demnach erfolgt

eine Mitteilung an das Kistenregal, sobald ein neuer Kundenauftrag das Unternehmen erreicht. Diese Aufgabe stellt den alleinigen Anteil des menschlichen Handelns in Bezug auf die Produktionssteuerung innerhalb dieser dezentralen Steuerung. Alle weiteren Aufgaben der Produktionssteuerung werden durch die restlichen Entitäten, insbesondere der Transportkisten, direkt am Ort des Geschehens – also dezentral – gesteuert und ausgelöst. So wird beispielsweise unmittelbar nach dem Abschließen des Arbeitsganges an der Maschinengruppe X ein Folgeauftrag ausgelöst, indem diejenige Transportkiste – in der Funktion als ein Publisher – zunächst einen verfügbaren Arbeitsplatz aus der nächsten Ebene bucht und anschließend eine Nachricht an das zugehörige Topic für neue Transportaufträge sendet, um die FTF in eine Auktion eintreten zu lassen. Die Kommunikation in der Dreieckskonstellation (Schritte 3 und 4) zwischen der Transportkiste, den FTF und den Arbeitsplätzen wiederholt sich bis zum Abschluss des Kundenauftrages. Außerdem stehen die FTF und die Arbeitsplätze in keiner Weise miteinander in Kommunikation. Alle Anfragen an sie ergehen lediglich durch die auftragsanfragenden Transportkisten.

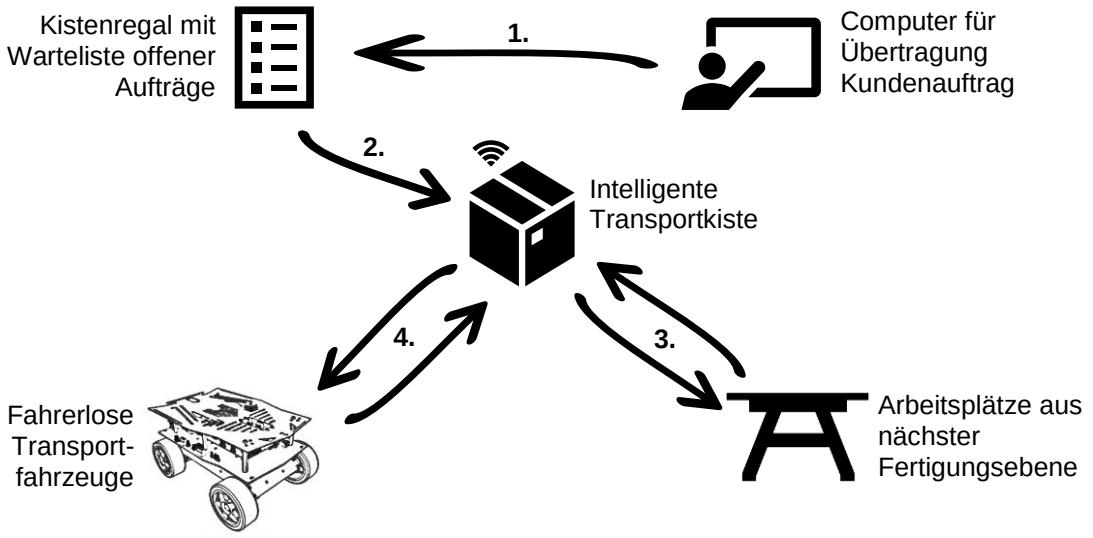


Abbildung 28: Übersicht der Kommunikationswege und -partner

4.4.2 Informationsverarbeitung mit RFID-Tags und MySQL-Datenbank

Für die Aufnahme und das Bereitstellen von Informationen im digitalen Produktionsgeschehen werden RFID-Tags eingesetzt. Der Einsatz der RFID-Tags beschränkt sich insgesamt in zwei Bereichen ein. Hierbei gilt es diese zwei unterschiedlichen Verwendungszwecke strikt zu unterscheiden, da zwischen beiden keine Relevanz herrscht. Sie müssen daher gesondert behandelt werden. In Abbildung 29 ist der Aufbau dieses Informa-

tionsträgers hierarchisch mit den jeweiligen Verwendungszwecken und der darin enthaltenen Informationen aufgelistet. Wie bereits erwähnt, werden die RFID-Tags als Positionsmarker für das Testfeld verwendet, womit das zu entwickelnde FTF seine aktuelle Position lokalisieren kann. Als zweiter Funktionsbereich gilt für sie das Tragen von Informationen hinsichtlich des auszuführenden Auftrages. Auftragsinformationen werden durch das Kistenregal, in der Funktion als Auftragsvorbereitung, dem RFID-Tag einer leeren Transportkiste geschrieben. Die Transportkiste besitzt zentral auf der Unterseite einen anfangs nicht beschriebenen RFID-Tag. Die Auftragsinformationen enthalten neben den Daten zur Zielposition relevante Artikeldaten, die in dem Produktionsprozess benötigt werden. Der Aufbau der Positionsmarker besteht je aus der Positionskoordinate und der Positionsklasse. Eine genaue Beschreibung dazu folgt in dem Abschnitt 5.2.3.

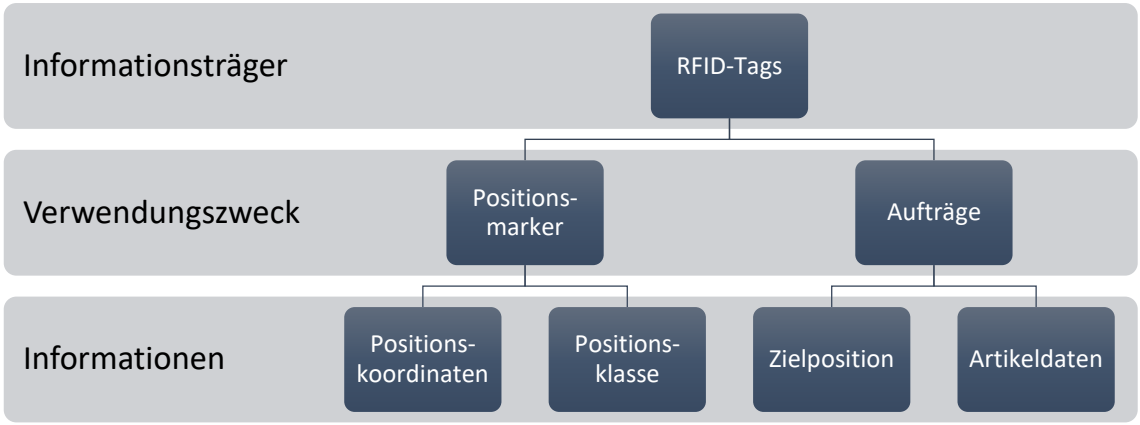


Abbildung 29: RFID-Tags als Informationsträger

Mit jedem Übertragen eines Kundenauftrages auf eine leere Transportkiste wird automatisch eine Verbindung zu der MySQL-Datenbank vorgenommen. In diesem werden diverse auftragsbezogene Daten geführt, die sowohl für die Produktion als auch für weitere prozessbezogene Analysen genutzt werden können. Bei den Artikeldaten geht es um Informationen, die für den Produktionsprozess relevant sind. Eine solche Datenbank hat eine Struktur gemäß Tabelle 5 aufzuweisen. Dazu gehören neben einer eindeutigen Artikel-ID, der Artikelname und der zugewiesene Faktor für die Errechnung der spezifischen Bearbeitungsdauer an den jeweiligen Arbeitsplätzen. Zudem sind noch zwei Bezüge zu anderen Datenbankeinträgen vorhanden. Zum einen für die jeweiligen Stücklisten mit Auflistung der benötigten Einzelteile und zum anderen eine Historie, in der alle bisher behandelten Produktionsaufträge zu dem jeweiligen Artikel mit den Ereignissen chronologisch aufgelistet werden. Durch das Letztere ist eine langfristige Rückverfolgbarkeit gewährleistet, um aussagekräftige Analysen und Schlüsse über die Prozesse treffen zu können. Anhand der eindeutigen Artikel-ID wird bei der Beauftragung einer Transportkiste der Bezug zu den weiteren benötigten Informationen aus der Datenbank

abgerufen. Daher wird beim Mitteilen von Kundenaufträgen an das Kistenregal die Artikel-ID genutzt. Alle geschriebenen Daten auf dem RFID-Tag der Transportkiste werden nach Abschluss des Kundenauftrages zurückgesetzt, damit es für den nächsten Auftrag verwendet werden kann.

Tabelle 5: MySQL Datenbankstruktur zu Artikeldaten aus dem betrachteten Sortiment

Artikel-ID	**Artikelname**	**Stückliste (Bezug)**	**Faktor**	**Lager-bestand**	**Historie (Bezug)**
Tas01	Tasse	ST_Tas01	1	…	H_Tas01
Unt01	Untertasse	ST_Unt01	0,5	…	H_Unt01
Tab01	Tablett	ST_Tab01	0,75	…	H_Tab01
Tee01	Teekanne	ST_Tee01	1,5	…	H_Tee01

Alle ausgelösten Aufträge erhalten zudem eine eindeutige Auftrags-ID. Die Notation dieser Auftrags-ID setzt sich dabei gemäß der Beschreibung in Abbildung 30 aus zwei Teilen zusammen. Der erste Teil ist das Datum des aktuellen Tages im Format „JJJJMMTT“. Dieser Teil ist über den gesamten Arbeitstag hinaus durchgehend gleich. Mit der Trennung durch einen Bindestrich folgt der zweite Teil aus der Kombination zwischen einer Zahl im Bereich von „0 bis 9“ und einem Buchstaben von „a-z“ nach deutscher Buchstabenordnung. Im ersten Auftrag ist es somit die Zahl „0“ und der Buchstabe „a“. Mit jedem weiteren neuen Kundenauftrag verändert sich zunächst der Buchstabenteil. Sobald der letzte Buchstabe erreicht ist, wird der Zahlenteil inkrementiert und der Buchstabenteil fängt wieder bei dem Buchstaben „a“ an. Insgesamt variiert diese ID mit einer Möglichkeit von 260 unterschiedlichen Auftrags-IDs pro Arbeitstag. Für eine Simulation im Rahmen dieser Arbeit ist diese Menge mehr als ausreichend. Der Grund für die Wahl einer Notation in dieser Dimension ist der vorausschauend langlebige Entwicklungsgedanke, sofern das Konzept in die Realität umgesetzt werden sollte.

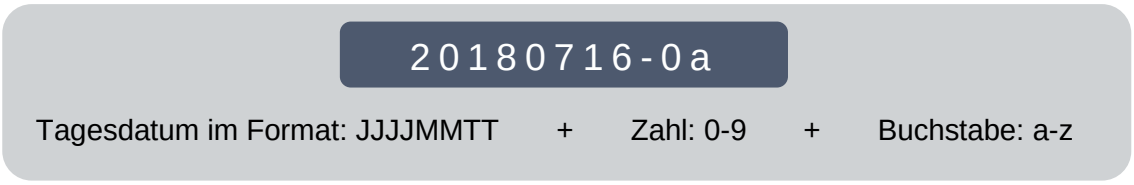


Abbildung 30: Notation der eindeutigen Auftrags-ID

Eingesetzt werden RFID-Tags mit der Bezeichnung MIFARE Classic 1K des Herstellers NXP. Die genaue Funktionsweise und der Aufbau dieser Technologie ist in dem zugehörigen Abschnitt 2.2.2 der Grundlagen erläutert. Alle bisher erwähnten Informationen

werden auf dem Speicher dieser RFID-Tags gespeichert. In Tabelle 6 ist exemplarisch der Speicherinhalt für einen neuen Kundenauftrag mit folgenden Informationen dargestellt:

- **Artikel-ID:** Tas01
- **Priorität:** NP (niedrige Priorität)
- **Zielposition:** Transport an AP_X1 mit Positionskoordinate „x2y1“

Nach Erhalt dieser drei Angaben werden die restlichen Informationen entweder generiert oder aus der MySQL-Datenbank abgerufen und anschließend auf den RFID-Tag mittels eines RFID-Lesegerätes gespeichert. Jeder Block eines RFID-Tags besitzt eine Größe von 16 Byte, welches exakt 16 Zeichen entspricht. Als Format für den Inhalt gilt das Hexadezimal-System. In der Darstellung wurde zur Veranschaulichung die Konvertierung ins Text-Format mit aufgeführt und hellblau hervorgehoben. Tatsächlich geschrieben werden ausschließlich Werte im Hexadezimal-System. Bytes, die in einem Block nicht beschrieben sind, enthalten den Wert „00“.

Im Speicherbereich „Sektor 3 / Block 1“ wird der jeweilige Auftragsstand festgehalten. Nach jeder erfolgten Bearbeitung wird dieser Speicherinhalt an dem jeweiligen Arbeitsplatz aktualisiert. Durch diese Information ist eine Fehlervermeidung bei der Belieferung der Arbeitsplätze möglich. Der Arbeitsplatz erkennt bei Lieferung, ob der vorige Fertigungsschritt durchlaufen wurde. Zudem kann eine Transportkiste infolge einer Störung aus dem Plan fallen, wonach sie durch die Information des aktuellen Bearbeitungsstandes umgehend die Planung für den nächsten Fertigungsschritt erledigen kann. Dabei gelten folgende Belegungen des Speicherbereiches, um den Bearbeitungsstand auszudrücken:

- **Auftragsvorbereitung:** 0
- **Maschinengruppe X durchlaufen:** 1
- **Maschinengruppe Y durchlaufen:** 2
- **Maschinengruppe Z durchlaufen:** 3
- **Auftrag abgebrochen:** 9

In dem Kapitel zu den weiteren Handlungsempfehlungen werden Hinweise zu der Nutzung von MySQL-Datenbanken mit dem Arduino-System gegeben.

Tabelle 6: Beispielhafter Speicherinhalt eines RFID-Tags für Transportkisten – Auftrag: Tasse

		Byte-Nummer des betrachteten Blocks – MIFARE Classic 1K																
Sektor	**Block**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**Beschr.**
…	**…**	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
3	**15**	KEY A						ACCESS BITS				KEY B						⊘
	14	**N**	**P**	00	00	00	00	00	00	00	00	00	00	00	00	00	00	Priorität
		4e	50															
	13	**0**		00	00	00	00	00	00	00	00	00	00	00	00	00	00	Status
		30																
	12	**1**	**.**	**5**	00	00	00	00	00	00	00	00	00	00	00	00	00	Faktor
		31	2e	35														
2	**11**	KEY A						ACCESS BITS				KEY B						⊘
	10	**S**	**T**	**_**	**T**	**a**	**s**	**0**	**1**	00	00	00	00	00	00	00	00	Stückliste
		53	54	5f	54	61	73	30	31									
	9	**T**	**a**	**s**	**s**	**e**	00	00	00	00	00	00	00	00	00	00	00	Artikel
		54	61	73	73	65												
	8	**T**	**a**	**s**	**0**	**1**	00	00	00	00	00	00	00	00	00	00	00	Artikel-ID
		54	61	73	30	31												
1	**7**	KEY A						ACCESS BITS				KEY B						⊘
	6	**2**	**0**	**1**	**8**	**0**	**7**	**1**	**6**	**-**	**0**	**a**	00	00	00	00	00	Auftrag-ID
		32	30	31	38	30	37	31	36	2d	30	61						
	5	**A**	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	Zielpositionsdaten
		57																
	4	**x**	**2**	**y**	**1**	00	00	00	00	00	00	00	00	00	00	00	00	
		78	33	79	31													
…	**…**	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

4.5 Auftragsvergabemodell

Unterteilt wird das Auftragsvergabemodell in zwei Arten. Sowohl Verhandlungen um den Erhalt eines Transportfahrzeuges durch die FTF, als auch die erfolgreiche Buchung eines Arbeitsplatzes als Ressource, gelten als Aufgabenbereiche, die durch dieses Modell abgedeckt werden. Obwohl beide voneinander unterschiedliche Aspekte behandeln, gehören sie trotz dessen in einem gewissen Rahmen zusammen, denn für die Beauftragung eines FTF für einen Transportauftrag wird vorerst die erfolgte Buchung eines Arbeitsplatzes vorausgesetzt. Demnach kann die Auftragsverhandlung zwischen den FTF erst dann stattfinden, wenn ein Arbeitsplatz aus der nächsten anzufahrenden Ebene erfolgreich gebucht ist. Das Starten von Anfragen und der damit zusammenhängenden Auftragsverhandlungen erfordern eine aktiv in der Produktion befindende Transportkiste mit einem offenen Kundenauftrag. Im Folgenden werden die Themen anhand des Beispiels eines Kundenauftrages für die Fertigung einer Tasse erläutert.

4.5.1 Aufnahme von neuen Kundenaufträgen in das System

Um dem Kistenregal den Eingang eines neuen Kundenauftrages mitzuteilen, wird mittels eines externen Computers eine Nachricht an das zugehörige Topic **„Auftraege“** gesendet. Der externe Computer spielt hierbei die Rolle des Vertriebes, wodurch die Produktion erfährt, welcher Artikel zu fertigen ist. Der Aufbau dieser Nachricht hat für das erwähnte Beispiel mit dem Auftrag einer Tasse die folgende Form aufzuweisen:

Tas01-NP

Die Mitteilung besteht aus zwei Teilen. Im ersten Teil wird auf die eindeutige Artikel-ID hingewiesen. Die Aufforderung ist es eine Tasse zu fertigen. Der zweite Teil gibt die Information über die Priorität des Auftrages an. Die Priorität kann in zwei Stufen angegeben werden. Entweder hat der Auftrag keine besondere Priorität und wird mit der Bezeichnung NP (niedrige Priorität) klassifiziert oder der Auftrag hat besonders hohe Priorität und erhält die Klassifikation HP (hohe Priorität). Beim Versenden der Nachricht ist auf die Groß- und Kleinschreibung zu achten. Das System des Kistenregals empfängt umgehend diese Nachricht, da es das Topic „Auftraege“ abonniert hat und der MQTT-Broker die Nachricht an alle abonnierten Clients weiterleitet. Der erhaltene Auftrag wird anschließend in einer MySQL-Datenbank in die Warteliste für offene Aufträge aufgenommen. Für die Warteliste wird das FIFO-Prinzip angewendet, wobei Aufträge mit hoher Priorität an die oberste Stelle geschoben werden.

Das Kistenregal arbeitet nach und nach alle vorhandenen Aufträge ab. Den Transportkisten werden die Aufträge zugewiesen und alle für den Auftrag relevanten Informationen auf den RFID-Tag der Transportkiste gespeichert. Ab diesem Zeitpunkt hängt die gesamte Steuerung ihrer Auftragsbearbeitung von den Transportkisten selbst ab.

4.5.2 Anwendung eines Zeitslot-Systems für Ressourcenplanung

Wie bereits in Abschnitt 4.2.1 beschrieben, wird für die zeitliche Betrachtung der Prozesse ein alternatives System aufgestellt. Dabei handelt es sich um ein Zeitslot-basiertes System, in der durch das Erfüllen einer für diese Zeitspanne bestimmten Aufgabe ein Zeitslot abgeschlossen wird. Das System ist gleichermaßen anzuwenden für alle Arten von Entitäten im System, weshalb ein Zeitslot erst abgeschlossen wird, wenn alle Entitäten ihre Aufgabe in dem betrachteten Zeitslot ausgeführt haben. Selbst wenn die Aufgabe nicht wie gewünscht erfüllt werden kann, wird zumindest eine Zustandsrückmeldung erwartet, um zu wissen, woran die Nichterfüllung der Aufgabe lag, wodurch weitere Maßnahmen ergriffen werden können. Zu den Aufgaben gehören beispielsweise das Bewegen des FTF um eine Positionskoordinate gemäß seiner zu fahrenden Route oder die anteilige Bearbeitungszeit während des Arbeitsvorgangs an einem Arbeitsplatz.

Mors beschreibt in seinem Modell des „Context-Aware Logistic Routing and Sheduling“ ein agentenbasiertes Verfahren zur konfliktfreien Routenplanung mit einem Ansatz zur Planung mit freien Zeitfenstern (free time windows). Das Ziel dabei ist es eine kollektiv optimale und realisierbare Routenplanung für das betrachtete System, wie z.B. ein fahrerloses Transportsystem mit den zugehörigen FTF als Agenten, hinsichtlich der begrenzten Kapazitäten zu schaffen. Das Modell wird als kontextsensitiv bezeichnet, denn die Agenten stehen stets unter dem Einfluss weiterer Agenten im System und müssen folglich Konsequenzen in Betracht ziehen, um sie in ihre Routenplanung einfließen zu lassen (Mors, Zutt, & Witteveen, 2007, S. 328).

Für die Planung der Ressourcen wird ein ähnliches Verfahren nach Mors verwendet. Dabei wird nicht der von ihm entwickelte Algorithmus, sondern lediglich ein vereinfachter Ansatz des freien Zeitslotsystems extrahiert und in die Entwicklung des A*-Algorithmus eingeführt. Die Idee ist es, eine Variable in die Routenfindungsfunktion einzubauen, die öffentlich für alle FTF zugänglich ist und durch eine Abfrage die Zustände der jeweiligen Positionen abfragen kann. Diese Variable ist eine Matrix in der Dimensionsgröße des Testfeldes, worin jede einzelne Zelle einen Zustand von 0 oder 1 aufweist, mit der die Information über die Belegung einer Position beschrieben wird. Der größte Vorteil dieses Ansatzes liegt in der Möglichkeit der dynamischen und kollisionsfreien Planbarkeit anhand der momentan belegten Positionen der FTF im Produktionsprozess. Aufgrund

dessen, dass die Bewegung der FTF auf eine spurgebundene Art funktioniert, ist eine solche Maßnahme unabdingbar, da sie sich sonst in der Fahrt gegenseitig blockieren würden. Daher muss schon in der Routenplanung diese Information berücksichtigt werden. Für FTF mit freier Bewegungsmöglichkeit im Produktionsbetrieb sieht das Ganze anders aus, da sie mittels Sensorik Hindernisse erkennen und Live-Optimierungen ihrer Route vornehmen. Hierbei sei noch zu erwähnen, dass die Frage der Kollisionsvermeidung nicht abschließend bearbeitet ist, denn die Fahrzeuge innerhalb dieser Modellierung besitzen aufgrund ihrer Architektur das Potenzial fehleranfällig zu sein, wodurch die erwartete momentane Position der FTF zu den jeweiligen Zeitslots abweichen kann. Wenn sich das Arduino-System eines FTF beispielsweise aufhängen sollte, kann das andere FTF keine Information über die Störung erhalten und geht davon aus, dass die Position des gestörten FTF zu dem nächsten Zeitslot frei befahrbar ist. In dem Abschnitt 7 für die weiteren Handlungsempfehlungen werden Ideen aufgeführt, welche als Anreize für die Beantwortung dieser Fragestellung dienen sollen. Wie in

Abbildung 31 dargestellt, können die FTF zwei Zustände annehmen[15]. Entweder sind sie für einen entsprechenden Zeitslot verfügbar (grün) oder bereits durch einen anderen Auftrag belegt (blau). Dazu sind im unteren Teil der Abbildung die abzufahrenden Positionen (Strecke) beschrieben. Da die Fahrwege und die Positionen den FTF durch Koordinaten mitgeteilt werden, ist für jede Belegung eines Zeitslots durch einen FTF auch dementsprechend eine Position auf den Fahrwegen belegt. Im rot umrandeten Bereich zu dem Zeitslot 2 sind beispielsweise beide FTF in Auftragsausführung und befinden sich zu diesem Moment auf unterschiedlichen Koordinaten. Durch die Belegung des FTF wird diese Positionskoordinate für den Moment absolut gesperrt, sodass kein anderes FTF dieses für seine Route in Betracht ziehen kann. Dies resultiert in der Planungsebene auf eine sichere Kollisionsvermeidung, welches wie oben beschrieben zunächst vollständig beschrieben und in der Entwicklungsphase erprobt werden muss.

Zeitslots [i]:	1	2	3	4	5
FTF 1	x	x	x		
FTF 2		x	x	x	

Strecke FTF 1	x0y1	x1y1	x1y2	
Strecke FTF 2		x2y2	x2y1	x1y1

Abbildung 31: Belegung der Positionskoordinaten durch die FTF je Zeitslot

[15] Diese Darstellung dient nur zur Veranschaulichung und stammt nicht aus einer Systemausgabe.

4.5.3 Auktionsmodell zur Verhandlung zwischen Transportkisten und Ressourcen

Das in den nächsten Abschnitten für das Auftragsvergabemodell verwendete Auktionsmodell baut auf dem Auktionsprotokoll des FIPA-Foundations auf („FIPA English Auction Interaction Protocol Specification“, 2011), welches von dem Institut für Integrierte Produktion Hannover im Rahmen ihres Forschungsprojektes „Dezentrale, agentenbasierte Selbststeuerung von Fahrerlosen Transportsystemen (FTS)“ angepasst wurde.

In

Abbildung 32 wird das Protokoll hinsichtlich der Kommunikation zwischen einem Initiator als Auftraggeber und den Beteiligten als Ressourcen in einem Auktionsmodell beschrieben. Der Initiator ist stets die Transportkiste in der Funktion als auftragsanfragende Entität, um Ressourcen zu buchen, wofür es in eine Kommunikation mit den Beteiligten eingeht. Sobald eine Anfrage bei den Beteiligten eingeht, prüfen sie vorerst die Ausführbarkeit des Auftrages und beantworten diese Frage entweder mit einem direkten Angebot, welches gemäß einer vorgeschriebenen Formel berechnet wird, oder sie lehnen den Auftrag ab, wenn eine Ausführbarkeit nicht möglich ist. Sobald alle Angebote eingehen, findet eine Auswertung des Auftraggebers statt, in der der Bieter mit dem günstigsten Angebot den Auftrag erhält. Das Modell dient dabei nur als Grundlage für den Aufbau des Auktionsablaufes.

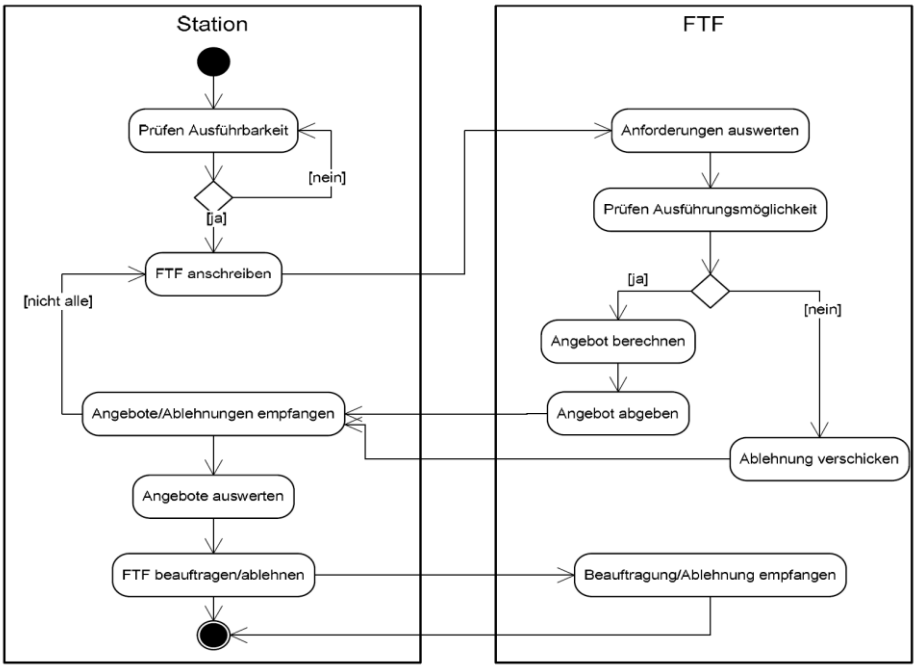


Abbildung 32: Auktionsmodell nach dem FIPA-Protokoll[16]

[16] (Institut für Integrierte Produktion, 2013, S. 15)

4.5.4 Buchungsanfrage an einen Arbeitsplatz

Unmittelbar nachdem die Transportkiste einen Auftrag durch das System des Kistenregals auferlegt bekommt, fängt es an eine verfügbare Arbeitsstation aus der nächsten Fertigungsebene zu suchen. Der aktuelle Bearbeitungsstand kann durch die Information aus dem RFID-Tag der Transportkiste abgerufen werden. Demnach weiß die Transportkiste, dass der Bearbeitungsstand noch in der Auftragsvorbereitung (Wert: 0) ist. Umgehend wird eine Anfrage an alle Arbeitsplätze aus der nächsten Fertigungsebene (Maschinengruppe X) gesendet. Insgesamt sind drei Fertigungsebenen vorhanden. Für jede einzelne Ebene sind für die MQTT-Kommunikation einzelne Topics zu definieren.

- **Maschinengruppe X:** Topic „**Ebene1/AP**“ und „**Ebene1/Anfrage**“
- **Maschinengruppe Y:** Topic „**Ebene2/AP**“ und „**Ebene2/Anfrage**“
- **Maschinengruppe Z:** Topic „**Ebene3/AP**“ und „**Ebene3/Anfrage**“

Die Nachricht geht hierbei aufgrund des aktuellen Bearbeitungsstandes an die „Ebene1“. Dabei wird kein externes Eintippen einer Nachricht wie zuvor benötigt, denn die Nachricht stellt sich selbst anhand von Bezügen zu den Informationen aus dem RFID-Tag zusammen. Somit werden stets aktuelle und korrekte Informationen abgerufen und übertragen. Der Aufbau der Nachricht ist in Abbildung 33 aufgeführt.

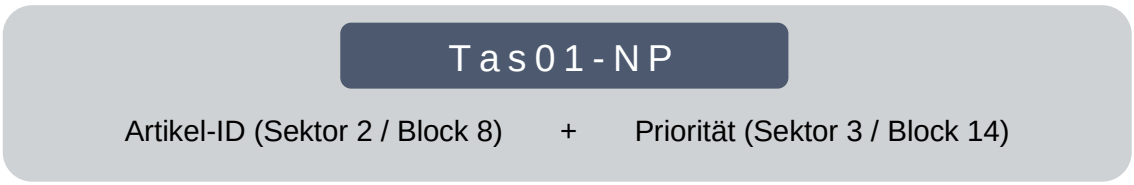


Abbildung 33: Aufbau der Nachricht zur Buchungsanfrage eines Arbeitsplatzes

Der Ablauf einer solchen Buchungsanfrage ist in dem Ablaufdiagramm aus Abbildung 34 dargestellt. Zu Beginn einer Buchungsanfrage erfolgt eine Nachricht durch die Transportkiste an alle Arbeitsplätze aus der nächsten Fertigungsebene. Für das erwähnte Beispiel geht die Nachricht bei dem Topic „Ebene1/AP“ ein, wodurch die zugehörigen Arbeitsplätze die Nachricht über den MQTT-Broker erhalten. Nachdem die Arbeitsplätze diese Nachricht empfangen, wird umgehend die jeweilige frühestmögliche Verfügbarkeit überprüft und eine Antwort in Form einer Nachricht an das Topic „Ebene1/Anfrage“ zurückgemeldet. Ist durch irgendeinen Arbeitsplatz eine Verfügbarkeit gewährleistet, so erhält die Transportkiste über diesem Topic die Angebote. Sollte kein Arbeitsplatz für die Bearbeitung zur Verfügung stehen, wie durch eine Störung, so wird die Auftragsanfrage abgelehnt, indem an den selben Topic eine Nachricht mit dem Inhalt „NV“ (nicht verfügbar) gesendet wird.

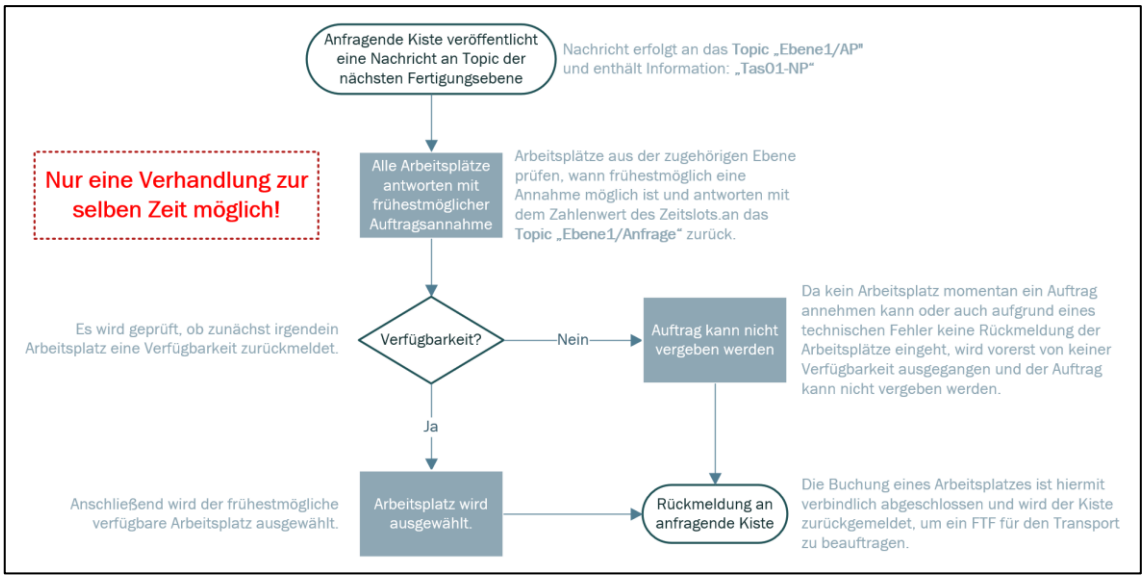
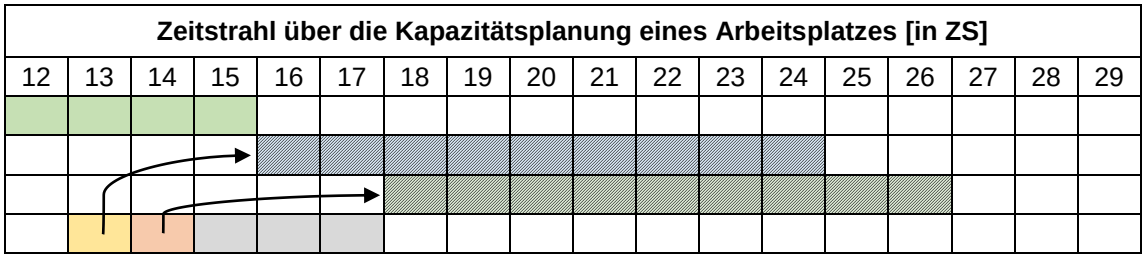


Abbildung 34: Ablaufdiagramm einer Buchungsanfrage an die Arbeitsplätze

Damit keine Anfragen von zwei Transportkisten zur gleichen Zeit kommen, ist in der Programmierung zu beachten, dass nur eine Verhandlung zur selben Zeit möglich ist. Das kann beispielsweise durch das automatische Abonnieren der Topics am Anfang einer Anfrage und das Beenden des Abonnements nach Abschluss der Anfrage ermöglicht werden. Zudem sollte durch eine Variablenprüfung der Status eines Arbeitsplatzes kenntlich gemacht werden, sodass daraus ersichtlich ist, dass momentan eine Anfrage läuft. Die zweite Anfrage müsste hierbei vorerst abgewiesen werden. Im nächsten Zeitslot wird eine erneute Anfrage durch die andere Transportkiste versucht, bis eine erfolgreiche Anfrage gestartet werden kann.

Die für die abschließende Buchungsanfrage benötigte Zeitspanne beträgt insgesamt 2 Zeitslots [ZS]. Sofern der Transportkiste die Ressourcen in dem benötigten Rahmen zugesichert sind, wird werden die Zeitslots für die Bearbeitung erstmal vorbehaltlich gebucht. Der Arbeitsplatz gibt dabei den frühestmöglichen Zeitslot an. Die Reservierung der Ressourcen ist auf eine Dauer von max. 2 Zeitslots zu beschränken, um der Transportkiste die Möglichkeit zu geben, ein FTF für den Transportauftrag zu buchen. Die Dauer der Beauftragung eines FTF dauert nur 1 Zeitslot, wobei eine Toleranz von einem weiteren Zeitslot gegeben wird, um die Möglichkeit technischer Fehler in der Kommunikation zu berücksichtigen. Sollte bis dahin keine Rückmeldung der anfragenden Transportkiste eingehen, wird die Reservierung storniert und wieder für weitere Aufträge freigegeben. Sobald die Transportkiste einen FTF mit einem Transportauftrag beauftragt hat, wird automatisch der Arbeitsplatz kontaktiert und die vorbehaltliche Reservierung endgültig in eine verbindliche Buchung geändert. Der Transportweg, welcher anfangs bei der Buchungsanfrage noch nicht bekannt ist, wird bei der vorbehaltlichen Buchung

außer Acht gelassen. Daher gibt es zwischen der vorbehaltlichen und der verbindlichen Kapazitätsbuchung einen anfangs nicht planbaren Verzug abhängig des Transportweges des beauftragten FTF. In diesem Beispiel benötigt das FTF insgesamt 3 Zeitslots für die Anfahrt an die Quelle des Arbeitsplatzes.



Legende			
	bestehender Auftrag		Buchungsanfrage
	Auftrag vorbehaltlich gebucht		Auftragsverhandlung FTF
	Auftrag verbindlich gebucht		Transport der Transportkiste

Abbildung 35: Zeitstrahl zur Kapazitätsplanung eines Arbeitsplatzes

Organisiert werden die Belegungen der Kapazitäten eines Arbeitsplatzes anhand von zugehörigen Arrays mit einer Größe von je 3600 Werten. Mit dieser Größe wird somit eine Auftragsplanung von genau einem Tag realisiert - mit der Annahme, dass ein Zeitslot einer Dauer von einer Minute in der Realität entspricht. Da es hierbei nur um die Konzeptionierung einer dezentralen Steuerung geht, ist eine längere Auftragsplanung aus Sicht einer gesamtheitlichen Produktion nicht erforderlich. Diese Arrays werden auf den jeweiligen Arduino-Mikrocontrollern der Arbeitsplätze geführt. Im Initialisierungsprozess erhält jeder Array-Index den Wert 0, welches die frei verfügbare Kapazität in dem jeweiligen Zeitslot aussagt. Für den Fall, dass ein Zeitslot reserviert ist, wird umgehend der zugehörige Array-Index mit einer 1 gefüllt. Durch das Referenzieren der Array-Indexe ist es möglich, einen bestimmten Zeitslot nach der Verfügbarkeit hin zu überprüfen.

Ein Nachteil an diesem Vorgehen sind die in der Kapazitätsplanung entstehenden Leerlaufzeiten, wie in diesem Beispiel innerhalb der Zeitspanne zwischen Zeitslot 16 und 17. Dies ist zugleich eine Chance, um in der Zukunft durch Optimierung des Konzeptes Leerlaufzeiten dieser Art beispielsweise für kurzfristige Sonderaufträge einzusetzen. In der Realität hingegen können durch solche Leerlaufzeiten kleinere Aufträge parallel zu einem Großauftrag gefertigt werden.

4.5.5 Vergabe des Transportauftrages an einen FTF

Der Eingang einer Anfrage bzgl. eines neuen Transportauftrages wird den FTF im Topic „T_Auftraege“ mitgeteilt. Hierzu werden durch die anfragende Transportkiste, in der

Funktion als Publisher, eine Textnachricht mit den anzufahrenden Positionskoordinaten veröffentlicht. Für einen Transportauftrag zum Arbeitsplatz „AP_X1“ erfolgt demnach folgende Nachricht:

x2y0

Die Koordinaten aller Arbeitsplätze sind den FTF anhand der auf dem Mikrocontroller gespeicherten Variablen bekannt. Mit der Veröffentlichung der obigen Nachricht werden alle im System vorhandenen FTF informiert, da in deren Initialisierungsprozess vordefiniert ist, dass das Topic „T_Auftraege“ abonniert wird. In Abbildung 36 ist das Kommunikationsschema dieses Beispiels zwischen den Entitäten im MQTT-Protokoll dargestellt. Nach dem Erhalt dieser Nachricht folgt umgehend eine Auftragsverhandlung. Dieses Schema gilt selbstverständlich gleichermaßen auch für eine Kommunikation zwischen der Transportkiste und den Arbeitsplätzen, worin die Arbeitsplätze dem Topic der jeweiligen Ebene folgen und die Anfragen von den Transportkisten erhalten.

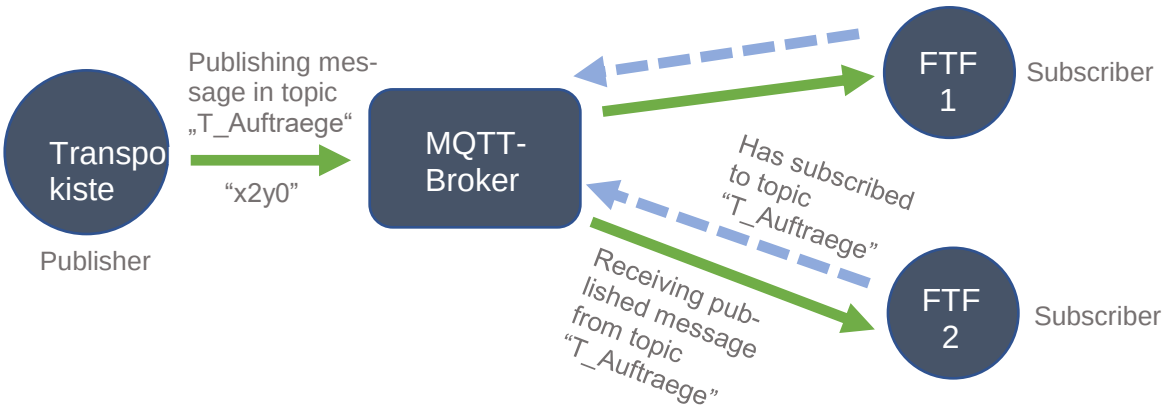


Abbildung 36: Kommunikation zwischen der anfragenden Transportkiste und den zu beauftragenden FTF mittels des MQTT-Protokolls

Beim Buchen eines Arbeitsplatzes sind die zurückzumeldenden Antworten seitens der Arbeitsplätze sehr einfach aufgebaut. Sie teilen lediglich ihre frühste Verfügbarkeit mit, womit sie in die Verhandlung treten. Die Auftragsvergabe an die FTF beinhaltet hingegen neben dem frühestmöglichen Verfügbarkeitszeitpunkt noch einen weiteren Wert, womit das Angebot in der Auftragsverhandlung zusammengesetzt wird – und zwar der Weg zum Auftragsort, ausgehend der Position des FTF zu dem frühestmöglichen Verfügbarkeitszeitpunkt. Anhand des A*-Algorithmus zur Routenfindung wird aus dieser Position vorerst der kürzeste Weg zum gewünschten Zielpunkt berechnet. Demnach besteht das Angebot mit einer Angabe in ZS aus der folgenden Gleichung:

$$\boldsymbol{Angebot_{FTF} = Verfuegbarkeitszeitpunkt + kuerzester\ Weg\ zum\ Ziel}$$

Der Prozessablauf einer solchen Auftragsverhandlung ist in Abbildung 37 dargestellt. Der Prozess startet mit dem Veröffentlichen einer Nachricht an das Topic für die Mitteilung von Transportaufträgen und endet mit der Rückmeldung aller Angebote der an der Auktion teilnehmenden FTF. Die Auktion ist einstufig aufgebaut, die entweder bei der ersten Runde erfolgreich ausgeht und den Auftrag vergibt, oder sie geht bei der Anfrage leer aus, weshalb ein neuer Versuch im nächsten Zeitslot getätigt werden muss. Die Angebote werden in Form von Textnachrichten an das Topic „T_Auftraege“ gesendet. Das niedrigste Angebot der eingehenden Nachrichten erhält anschließend den Transportauftrag.

Auch in diesem Vorgang ist nur eine Verhandlung zur selben Zeit möglich. Eine Maßnahme, um dies sicherzustellen, ist in dem vorigen Abschnitt beschrieben und hierfür analog anzuwenden.

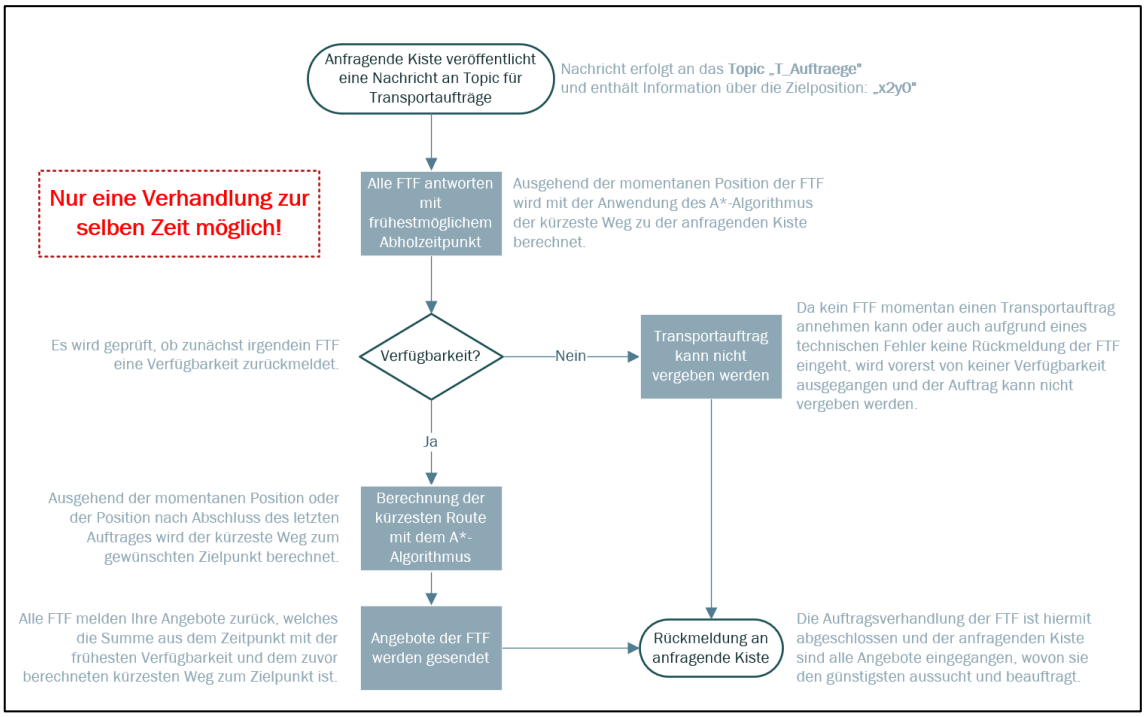


Abbildung 37: Ablaufdiagramm einer Auftragsverhandlung für den Erhalt eines Transportauftrages

Hiermit ist das Auftragsvergabemodell vollständig beschrieben, in der die Kommunikation der Transportkisten mit den Arbeitsplätzen und den FTF ermöglicht ist. Mittels dieser Funktionen haben die Transportkisten die Fähigkeit erlangt die Produktion gemäß ihrem Erfordernis zu steuern.

5 Entwicklung des FTF und Validierung mit vereinfachtem Testfeld

In diesem Abschnitt wird eine Teilentwicklung zu dem im vorigen Kapitel aufgestellten Konzept vorgestellt, welches die praktische Leistung dieser Arbeit darstellt. Mit dieser Teilentwicklung wird ein fahrerloses Transportfahrzeug in Form eines Prototyps entwickelt. Das entwickelte Fahrzeug ist zudem mit einem vereinfachten Testfeld zu validieren. Abgedeckt werden hierbei von den zuvor vorgestellten Entitäten daher das fahrerlose Transportfahrzeug und das Testfeld. Eine Kommunikation zwischen Entitäten findet daher nicht statt und demnach auch keine Auftragsvergabe in Form einer Auktion. Das Ziel dabei ist es ein FTF mittels eines manuell zugeführten Auftrages in Form eines RFID-Tags anzuweisen, einen Transportauftrag auszuführen. Die nötigen Schritte zur Realisierung des gesamten Konzeptes werden in dem nächsten Abschnitt 7 zu den weiteren Handlungsempfehlungen behandelt.

5.1 Vereinfachtes Testfeld

Im Konzeptteil wurde im Rahmen des aufgestellten Produktionsszenarios ein Testfeld entworfen, welches für die Validierung der Teilentwicklung aus diesem Abschnitt überdimensioniert ist. Das vorgestellte Testfeld aus 4.3 wird daher auf eine Dimension von G = (10, 13) komprimiert und erhält den dargestellten Aufbau aus Abbildung 38. Mit 10 Knoten und 13 Kanten ist diese Dimensionierung ausreichend, um die Funktionalitäten des FTF zu validieren. Hierin sind keine Arbeitsplätze vorhanden, sondern lediglich Positionen, die mittels der jeweiligen Positionskoordinaten angefahren werden. Zudem ist dem FTF eine feste Startposition zugewiesen, welches als eine Halteposition mit der Positionsklasse „H“ deklariert ist. Dies ist in der Realität gleichzusetzen mit einer fest definierten Ladestation eines FTF – einem festen Parkplatz.

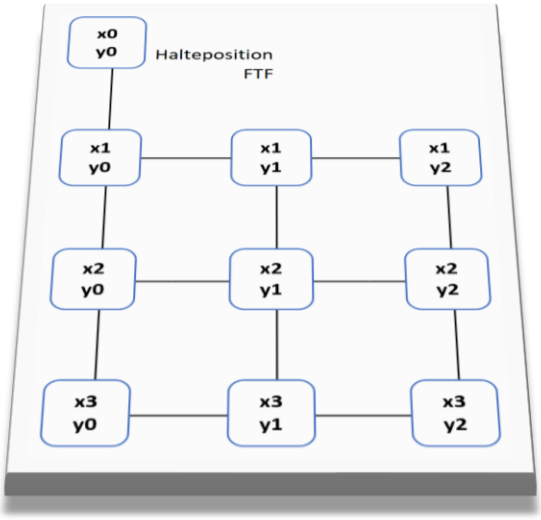


Abbildung 38: Komprimiertes Testfeld

5.2 Entwicklung eines fahrerlosen Transportfahrzeuges

Dieser Abschnitt behandelt das Thema der Entwicklung eines fahrerlosen Transportfahrzeuges (FTF). Als eine systematische Entwicklungsmethode wird hier das Prinzip des Minimum-Viable-Product (MVP) (vgl. 2.4) verwendet. Im Entwicklungsprozess wurden demnach für alle Entwicklungsschritte Teillösungen vorgestellt und anhand des Feedbacks Optimierungen vorgenommen. Durch die MVP-Methode ist daher sichergestellt, dass die Entwicklung und der Funktionsumfang validiert sind. Die Anforderung an das FTF ist es einen Transportauftrag von A nach B auszuführen. Insgesamt hat die Entwicklung sechs Schritte benötigt, welche in den folgenden Seiten im Detail beschrieben werden.

5.2.1 Aufbau und Sicherstellung der Fahrbewegungen

MVP-Schritt 1: Das fahrerlose Transportfahrzeug (FTF) wird mittels eines Arduino-Systems entwickelt. Grundlegend handelt es sich dabei um ein Fahrzeug-Roboter, das die Funktion eines FTF tragen wird. Es besteht die Möglichkeit ein Fahrzeug von Grund auf neu aufzubauen oder einen Bausatz zu erwerben und beliebig zu erweitern. In dieser Arbeit wurde sich für die letztere Variante entschieden und ein Bausatz von dem chinesischen Hersteller DFRobot, ein weltweit führender Anbieter von Robotik und Open-Source-Hardware, bestellt und aufgebaut. Dabei handelt es sich um den Bausatz **„Cherokey 4WD Arduino Mobile Robot“** (DFRobot, o. J.-a). Besondere Eigenschaften dieses Bausatzes sind unter anderem:

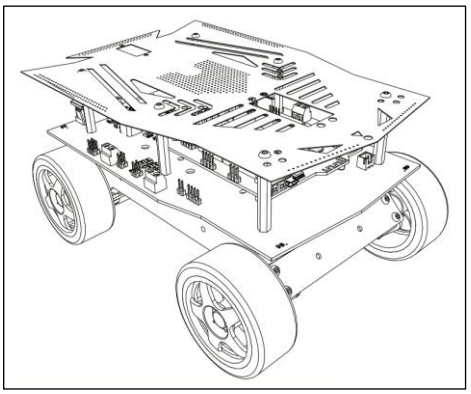


Abbildung 39: DFRobot Cherokey 4WD

- Kompatibel mit den gängigsten Arduino Mikrocontrollern (nicht inklusive)
- Eingebauter Motortreiber (L298P) in der Hauptplatine
- Erweiterungsplatine (oben) für die Installation weiterer Module
- Äußerst widerstandsfähiges Gehäuse aus Aluminiumlegierung
- 4x Gummireifen mit 65 mm Durchmesser
- 4x Getriebemotoren mit 6V und 160 RPM

Als Arduino-Modell kommt der **„Mega 2560“** zum Einsatz, welcher im Vergleich zu der weitbekannteren Variante „Uno“ noch leistungsstärker ist und mehr Anschlussmöglichkeiten bietet, welches aufgrund der Mehrzahl der einzusetzenden Module in dieser Arbeit ein großer Vorteil ist. Neben dem offiziellen Hersteller „Arduino“ gibt es auch diverse

andere Hersteller, die als „Replica/Clone“, also baugleiche Mikrocontroller, mit unterschiedlichen Namen vermarkten, da Arduino-Systeme ein Open-Source-Projekt sind (vgl. 2.2.1). Genauso wie der Fahrzeug-Bausatz stammt auch der Arduino Mega 2560 von dem Hersteller DFRobot. In Abbildung 40 wird der erwähnte Mikrocontroller dargestellt. Für die nachfolgende Erläuterung des Aufbaus wurde der Wiki-Artikel des Herstellers herangezogen. Auch in der Erläuterung verwendete Abbildungen stammen, soweit nicht anders verwiesen, aus dem genannten Wiki-Artikel (DFRobot, 2017b).

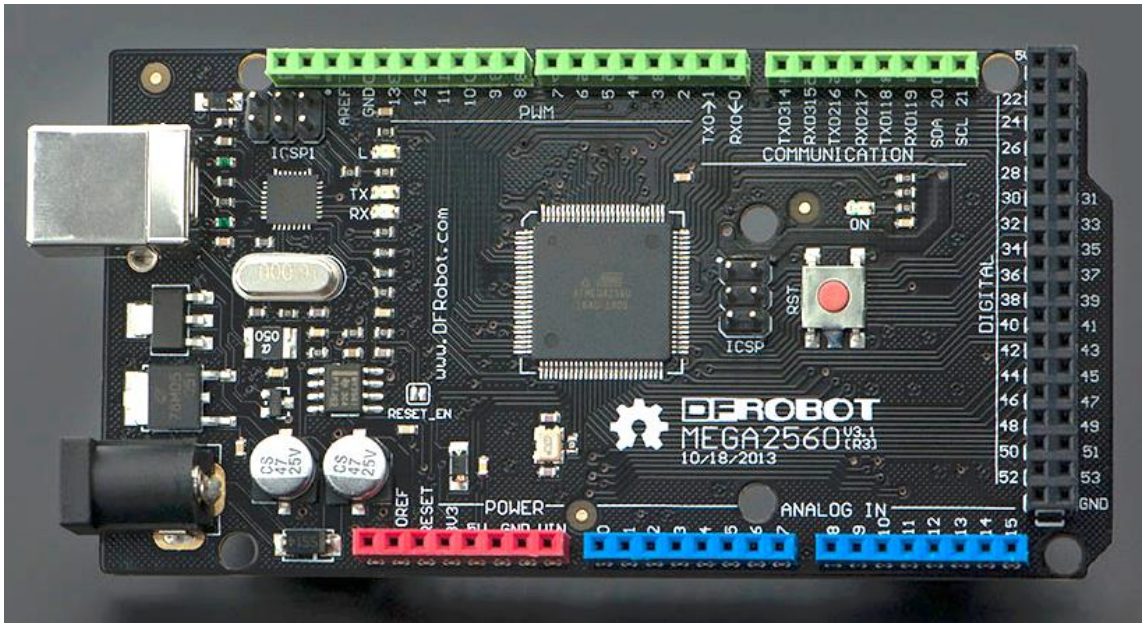


Abbildung 40: Layout des Mikrocontrollers Arduino Mega 2560[17]

Aufgebaut wird das Fahrzeug gemäß der Anleitung aus dem Lieferumfang. Erst nachdem das Fahrzeug erfolgreich aufgebaut ist, wird die Verbindung mit dem Arduino hergestellt. Zu Beginn ist eine wichtige Voreinstellung vorzunehmen, die für die gleichzeitige Bewegung beider Motoren an den Fahrseiten notwendig ist. Mittels des Einsatzes von Jumpern ist ein Kurzschluss an den Motor Select Pins durchzuführen. Dadurch wird erreicht, dass wenn der „Motor 1“ (rechte Fahrzeugseite) angesteuert ist, auch der hintere „Motor 3“ mitgesteuert wird. Auf dieselbe Art und Weise ist dies mit dem „Motor 2“ (linke Fahrzeugseite) und „Motor 4“ auszuführen. Der korrekte Einbau der Jumper ist in

Abbildung 41 dargestellt.

Abbildung 41: Einstellung der Motor Select Pins mit Jumpern

[17] (DFRobot, o. J.-b)

Nachfolgend ein Überblick der Hauptplatine des Fahrzeug-Bausatzes (Abbildung 42), welches mit dem Arduino-Board verbunden wird. Für die Erläuterung sind die einzelnen Stellen markiert und werden in mit Verweis auf die Kennzeichnung auf der Abbildung in kurzer Form erklärt.

- Der **Bereich 1** beinhaltet einen „XBee“ und einen Bluetooth socket, welche in dieser Arbeit nicht eingesetzt werden. Die Kommunikation zwischen den Entitäten wird nicht über diese Module, sondern mittels einer Internetanbindung über das MQTT-Protokoll ermöglicht (vgl. 4.4.1).

- Anschließend wird der **Bereich 2** in zwei Teile (a/b) unterteilt. In diesem Bereich sind die oben erwähnten Pins für den Motor Select enthalten. Außerdem werden die Kabel der Motoren, mit Beachtung der Vorzeichen, an die jeweiligen Eingänge eingeklemmt. Zusätzlich befinden sich die Stromversorgerpins für Encoder, womit eine sehr genaue Steuerung der Radumdrehungen erreicht werden können. Encoder kommen in dieser Arbeit nicht zum Einsatz, da eine einfachere Art der Motorsteuerung über die Pulsweitenmodulation (PWM) ausreichend ist.

- Die Verbindung zwischen der Hauptplatine des Fahrzeuges und dem Arduino Mikrocontroller erfolgt im **Bereich 3**. Besonders gut eignen sich hier Jumper Kabel zu verwenden (female/male), um Wackelkontakte zu vermeiden. Die Pins RX und TX ermöglichen die serielle Kommunikation zwischen dem Mikrocontroller und der Hauptplatine. Beide haben einen fest definierten Pin auf dem µC mit denen sie angeschlossen werden müssen. **RX** wird mit dem digitalen **Pin „0“** und **TX** mit dem digitalen **Pin „1“** verbunden. Hier sollte darauf geachtet werden nicht die analogen Pins zu verwenden. Weiterhin wird der Motortreiber L298P mit dem Arduino verbunden.

  Aufgrund dessen, dass je Fahrseite der vordere und der hintere Motor gleichzeitig angesteuert werden, wird während der Programmierung nur der vordere Motor berücksichtigt. Bei der Programmierung gibt es jeweils zwei Pins je Motor, die berücksichtigt werden müssen. Zum einen ein Pin, welcher für die Richtung „EN“ (vorwärts/rückwärts), und zum anderen ein Pin, welcher für die Geschwindigkeit durch die Pulsweitenmodulation zuständig ist. Die seitens des Herstellers vorgegebenen Pins (digitale) für den Arduino Mega sind für die rechte Fahrseite mit **Motor 1 EN = Pin 4** und **PWM = Pin 5**. Für die linke Fahrseite **Motor 3 EN = Pin 6** und **PWM = Pin 7**.

- Die Stromversorgung der Hauptplatine und des µC erfolgt mittels eines Batteriepacks im **Bereich 4** (angeschlossen in 4a), welcher oben auf der Erweiterungsplatine angebracht ist. Das Batteriepack nimmt insgesamt 5 Batterien mit je 1,5 V auf. Um den µC mit Strom zu versorgen wird der eingebaute Netzstecker (4b) an vorgesehener Stelle angeschlossen.

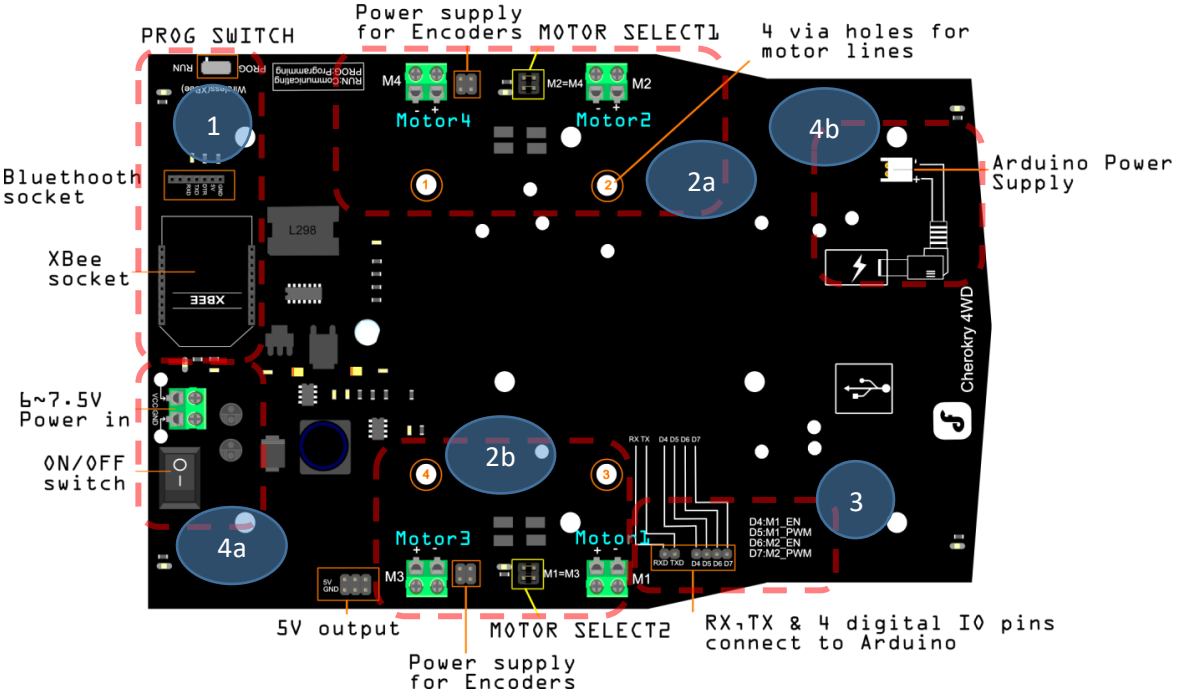


Abbildung 42: Überblick der Hauptplatine

Nachdem das Fahrzeug erfolgreich aufgebaut und angeschlossen ist, müssen die Fahrbewegungen hergestellt werden. Das Fahrzeug fährt in allen 4 Richtungen, wobei die Bestimmung der Winkel, also der Grad der rechts/links-Bewegung, eine nicht sehr einfache Aufgabe ist. Aufgrund dessen, dass keine Encoder eingesetzt und die Radumdrehungen nicht ermittelt werden können, werden die Motoren über die PWM-Methode gesteuert. Dabei geht es um das wechselseitige an und ausschalten der Stromzufuhr an die jeweiligen Motoren und der aus der durchschnittlichen Stromzufuhr resultierenden Geschwindigkeit.

Der allererste Schritt ist das Bewegen des Fahrzeuges mittels eines Programmcodes, welcher in der Arduino-Programmierumgebung (IDE) geschrieben und anschließend auf den µC hochgeladen wird. Das Fahrzeug soll zu Beginn eine Abfolge von Fahrbewegungen durchführen (vorwärts, rückwärts, rechts, links). Die Richtung der Motoren wird im Programmcode über die Pins mit den Bezeichnungen **„richtungPin_M1“** und **„richtungPin_M2“** und die Geschwindigkeit anhand eines Wertes der Pulsweitenmodulation über die Pins mit den Bezeichnungen **„speedPin_M1“** und **„speedPin_M2“** bestimmt.

Ob ein Motor vorwärts oder rückwärts drehen soll, hängt von dem Status HIGH (vorwärts) oder LOW (rückwärts) ab. Hierbei sei zu erwähnen, dass aufgrund der Gleichheit der Motoren bei korrekter Verkabelung (positiv/negativ), der Zustand High auf beiden Pins eine Vorwärtsbewegung der rechten Seite und eine Rückwärtsbewegung der linken Seite für das Fahrzeug bedeutet, da der Motor spiegelverkehrt angebracht wird. Um dieses Problem zu lösen gibt es zwei Möglichkeiten:

- Die Kabelverbindungen zu den Motoren der linken Seite vertauschen (positiv → negativ).
- Im Programmcode einmalig innerhalb der Bewegungsfunktionen für eine Vorwärtsbewegung der linken Motoren ausnahmsweise LOW nutzen.

Um für keine Verwirrung beim Aufbau des Fahrzeuges zu sorgen, fällt die Entscheidung für die zweite Lösungsvariante. Exemplarisch befindet sich nachfolgend die Funktion für die Vorwärtsbewegung, unter Beachtung der erwähnten Veränderung, wobei die Funktion **„vorwaerts()“** in Abhängigkeit zu den bereits erwähnten zwei Variablen steht. Diese Variablen werden bei dem Aufrufen der Funktion vergeben. Soll das Fahrzeug nun mit voller Geschwindigkeit vorwärts fahren, so wird die Funktion mit **„vorwaerts(255,255)“** aufgerufen. Die Dauer der Vorwärtsbewegung wird unmittelbar nach der Bewegungsfunktion mit **„delay(t)“**, als Funktion der Zeit in Millisekunden, angegeben.

```
void vorwaerts(int linksSpeed, int rechtsSpeed) {
  digitalWrite(richtungPin_M1, HIGH);
  analogWrite (speedPin_M1, rechtsSpeed);
  digitalWrite(richtungPin_M2, LOW);
  analogWrite (speedPin_M2, linksSpeed);
}
```

Abbildung 43: Bewegungsfunktionen

Insgesamt gibt es fünf Bewegungsfunktionen, welche in Abbildung 43 aufgeführt sind. Jedem dieser Funktionen wird beim Aufrufen der gewünschte PWM-Wert, wie in dem Beispiel oben, als Geschwindigkeitswert übermittelt. Hiermit ist der erste MVP-Schritt abgeschlossen und die Fahrbewegungen des FTF sind sichergestellt. Diese Bewegungsfunktionen werden für die Entwicklung der Line-Follower-Funktion im nächsten Abschnitt verwendet.

5.2.2 Bewegungssteuerung als Line-Follower

MVP-Schritt 2: Der zweite Entwicklungsschritt befasst sich mit der Entwicklung einer Funktion, mit der sich das FTF steuern lässt. Im Grundlagenteil sind mögliche Verfahren zur Bewegungssteuerung von FTF aufgelistet (vgl. 2.1.2). Grundlegend lassen sie sich in zwei Arten unterteilen:

- Spurunabhängige und freie Steuerung im gesamten Raum
- Spurgebundene Steuerung mit festen Wegen

Die erste Variante erfordert eine komplexere Technik als die zweite. Um eine völlig unabhängige Steuerung im gesamten Raum zu ermöglichen, muss sich das Fahrzeug beispielsweise mittels GPS an jeder Position lokalisieren können. Außerdem sind Erkennungsmarken (definierte Objekte im Raum) ein Mittel zur Lokalisieren, um eine noch genaue Positions- und Ausrichtungserkennung zu ermöglichen. Diese Variante ist selbstverständlich die weitaus bessere und anspruchsvollere Lösung. Da es sich in dieser Arbeit lediglich um eine Prototypenentwicklung und Modellierung eines dezentralen Produktionssystems für fahrerlose Transportfahrzeuge handelt, genügt die Entwicklung einer spurgebundenen Steuerung, welches vom Entwicklungsaufwand her weitaus geringer ist und trotz dessen die Anforderungen vollkommen erfüllt.

Dazu bewegt sich das FTF auf dem Testfeld auf schwarzen Linienspuren, welche mit einer Gitterstruktur die komplette Produktionsumgebung durchlaufen (vgl. 4.3). Daher bedingt es eine technische Lösung für den FTF zu implementieren, welches diese Linien erkennt und seine Fahrbewegungen dahingehend steuern kann. Diese Art der Technik wird als „Line-Tracking“ und diese Fahrzeuge als „Line-Follower“ bezeichnet. Dabei wird eine bestimmte Anzahl an Line-Tracking-Sensoren auf die untere Seite des Fahrzeuges angebracht, welche durchgehend dem µC Rückgabewerte liefern. Dabei melden die Sensoren den Wert „0“ bei schwarzen Oberflächen und den Wert „1“ bei hellen Oberflächen zurück (vgl. 2.2.2).

Bei dem zu entwickelnden FTF kommen drei Line-Tracking-Sensoren zum Einsatz, welche zentral im vorderen Drittel der Unterseite des FTF mit einem Abstand von ungefähr 2 cm zueinander angebracht werden. Dabei müssen die Sensoren exakt parallel und gerade positioniert sein, denn ohne die korrekte Anbringung der Sensoren können keine zuverlässigen Rückgabewerte erhalten werden, welche für eine richtige Bewegungssteuerung notwendig sind. Die Line-Tracking-Sensoren stammen, wie die anderen Komponenten bisher auch, von dem Hersteller DFRobot, wobei eines der drei Sensoren ein unterschiedliches Modell ist, welcher im Analogmodus auch Grauwerte lesen kann. Für

die Anwendung in diesem Falle macht es keinen Unterschied, da alle Sensoren im Digitalmodus betrieben werden und diese Funktion nicht genutzt wird. So werden alle drei Sensoren eingesetzt, um lediglich zwischen zwei Zuständen, helle oder dunkle Oberfläche, zu unterscheiden. In Abbildung 44 und Abbildung 45 sind die Sensoren mit den jeweiligen Modellbezeichnungen dargestellt. Außerdem gibt es von diversen anderen Herstellern Line-Tracking-Arrays zu erwerben, welche mehrere Sensoren als ein einziges Bauteil enthalten. Diese Art von Modulen sorgen für eine noch genauere Steuerung von Fahrzeugen, da je mehr Sensoren im Einsatz sind, desto höher die Kombinationsmöglichkeiten der Rückgabewerte werden und eine damit zusammenhängende Erhöhung der Freiheitsgrade bei der Steuerung entsteht.

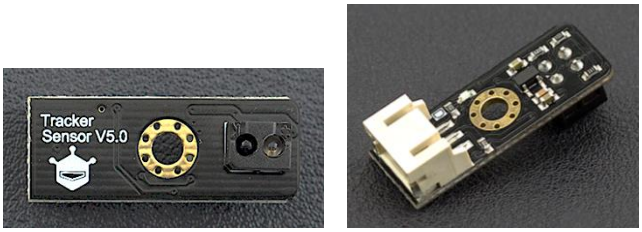


Abbildung 44: DFRobot - Tracker Sensor SEN0017[18]

Abbildung 45: DFRobot - Smart Grayscale Sensor SEN0147[19]

In Tabelle 7 sind alle möglichen Kombinationen der Sensor-Rückgabewerte, mit Nennung der Variablennamen im Programmcode, und der daraus resultierenden Fahrbewegungen aufgelistet. Zusätzlich werden noch die einzelnen Motorbewegungen mit entsprechender Motorbezeichnung aufgeführt. Wie bereits erwähnt, kann das FTF insgesamt fünf Bewegungsfunktionen ausführen. Die Rückwärtsbewegung wird lediglich als Teil des Korrekturprozesses verwendet. Sobald das FTF aus der schwarzen Linie komplett rausfährt und alle drei Sensoren eine helle Oberfläche erkennen, muss sie sich erneut auf die Spur begeben. In diesem Falle wird der Korrekturprozess aufgerufen, indem das Fahrzeug vorerst ein Stück zurückfährt und seine letzte seitliche Bewegung (rechts oder links) wiederholt. Diese Information wird fortlaufend in der Variable **„letzte_wende“** gespeichert.

[18] Links und rechts angebracht
[19] Mittig angebracht

Bei einem Line-Follower wird die Rückwärtsbewegung in der Regel nicht benötigt, da ausgehend einer Linie keine Information für einen Rückwärtsfahrbefehl erzeugt werden kann. Sollte sich das FTF unter bestimmten Bedingungen um 180° drehen müssen, wird dies mit einer längeren Rechtsbewegung und anschließender Positionierung auf der schwarzen Linie realisiert.

Tabelle 7: Sensor-Rückgabewerte mit resultierender Fahrbewegung

Sensor-Rückgabewerte			**Motorbewegung**				**Bewegungsfunktion**
Links	**Mitte**	**Rechts**	**Linke Seite**		**Rechte Seite**		
sen_l	**sen_m**	**sen_r**	**M2**	**M4**	**M1**	**M3**	
1	0	1	Vorwärts		Vorwärts		vorwaerts()
0	0	0	Vorwärts		Vorwärts		vorwaerts()
0	0	1	Rückwärts		Vorwärts		links()
0	1	1	Rückwärts		Vorwärts		links()
1	1	0	Vorwärts		Rückwärts		rechts()
1	0	0	Vorwärts		Rückwärts		rechts()
0	1	0	in Ruhe		in Ruhe		Stopp()
1	1	1	# # #		# # #		Korrekturprozess

Grundlegend geht es daher um das Aufrufen der Bewegungsfunktionen „vorwaerts()“, „links()“ und „rechts()“. Mit

Abbildung 46 sind drei der möglichen Kombinationen der Sensor-Rückgabewerte illustriert. Wenn der mittlere Sensor eine Schwarze Oberfläche erkennt, fährt das FTF geradeaus weiter. Bei den beiden anderen Varianten muss sich das FTF dementsprechend wieder einpendeln, sodass das FTF erneut geradeaus auf der Spur weiterfahren kann.

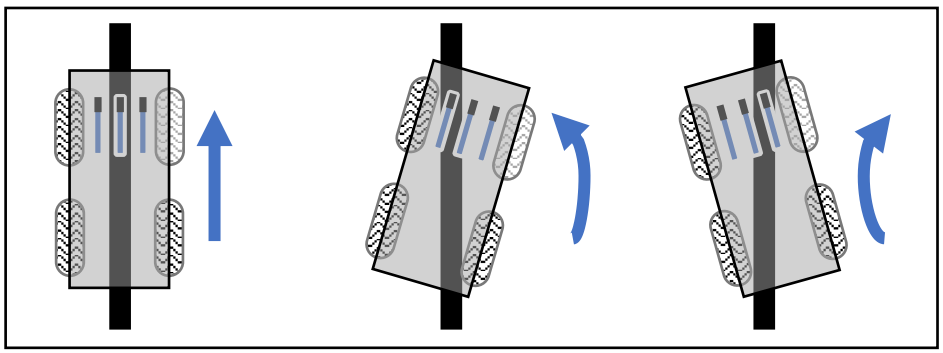


Abbildung 46: Kombination der Sensor-Rückgabewerte für Fahrbewegungen

Mit der Entwicklung der Line-Follower-Funktion ist der zweite MVP-Schritt abgeschlossen. Im nächsten Schritt muss eine Methode zur Lokalisation des FTF entwickelt werden.

5.2.3 Lokalisation: Positions- und Ausrichtungserkennung

MVP-Schritt 3: Für die Positionserkennung kommen Positionsmarker in Form von RFID-Tags zum Einsatz. Das FTF erkennt durch das Lesen des jeweiligen RFID-Tags mittels eines RFID-Lesegerätes, welcher zentral auf der Unterseite des FTF angebracht wird, zu jeder Stelle auf dem Testfeld, wo es sich exakt befindet. Diese RFID-Tags sind an jeder Position angebracht und beinhalten zwei Informationen. Zum einen die Positionskoordinaten und zum anderen die Positionsklasse.

Jede Position befindet sich in einem von drei gegebenen Klassentypen (Abbildung 47). Dem FTF wird ein bestimmter Halteplatz (Ladestation) an der Position **„x0y0“** zugewiesen, welcher mit dem allgemeinen Klassentypen **„H“** klassifiziert wird. Die Zielpositionen auf dem Testfeld des Konzeptes sind die fest definierten Arbeitsplätze mit dem Klassentyp **„A“**. Alle restlichen Positionsmarken auf dem Testfeld besitzen den Klassentypen **„W“**, auf denen abhängig der Routenplanung entschieden wird, ob sie geradeaus weiterfahren oder eine Wende machen müssen.

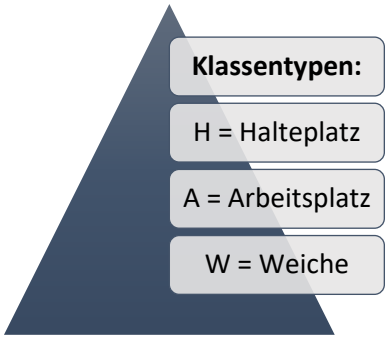


Abbildung 47: Klassentypen der Positionsmarken

Wie bereits erwähnt ist ein RFID-Lesegerät notwendig, um die Positionserkennung zu ermöglichen. Verwendet werden hierfür das RFID-Lesegerät des Herstellers „Funduino“. mit der Bezeichnung **„MFRC522“**. Hierfür wird die bereits erwähnte Bibliothek des Entwicklers Miguel Balboa verwendet (vgl. 2.2.3). Die Beispielcodes dienen dabei als Basis für die Entwicklung der Funktionen zur Positionserkennung. Dieses Lesegerät ist sowohl für das Lesen von Informationen aus RFID-Tags, als auch für das Beschreiben dieser RFID-Tags fähig.

Nach dem Anschließen des RFID-Lesegerätes an den Arduino, werden die Funktionen zu dem Lese- und dem Schreibprozess an das Konzept angepasst. So werden mit der entwickelten Funktion **„rfid_lesen()“** jeweils der **Block 4 (Klasse)** und **Block 5 (Position)** eines RFID-Tags gelesen und dabei in die zugehörigen Variablen (Abbildung 49) gespeichert. Die Positionsmarken müssen daher für den Aufbau des Testfeldes vorbe-

reitet werden. Hierfür wird jedes einzelne RFID-Tag mit den Informationen zu der Position gemäß dem (vereinfachten) Testfeld sowohl mit den Positionskoordinaten als auch der Positionsklasse beschrieben.

Verwendet werden Tags in Form von runden Stickern, welche mit einem Durchmesser von 25 mm, einer sehr geringen Dicke und einer weißen Papieroberfläche optimal für das Vorhaben geeignet sind. Auf die Oberfläche wird die Beschriftung der jeweiligen Positionskoordinate notiert, um beim Auf- und Abbauen des Testfeldes die Positionsmarker richtig positionieren zu können. Dabei handelt es sich um RFID-Tags der Firma NXP mit der Bezeichnung MIFARE Classic 1K. [20]

Für eine genaue Lokalisation muss das FTF zusätzlich zu seiner Position auch in seiner Ausrichtung exakt kennen. In der Entwicklung dieses Fahrzeuges wird keine technische Lösung für die Ausrichtungserkennung genutzt, da diese ohne Mehrwert die Komplexität der Entwicklung steigern würde. Daher wird diese Problemstellung mittels der Programmierung in Form eines zu entwickelnden Algorithmus gelöst (vgl. 5.2.5). Zusammengefasst geht es bei dem Algorithmus um die laufende Aktualisierung einer Variable, welches laufend die aktuelle Ausrichtung beschreibt.

Nachdem die Lokalisation des FTF möglich ist, sind die benötigten Funktionen hinsichtlich des Fahrens vollständig entwickelt. Die weitere Betrachtung in den nächsten Schritten folgt in der Auftragsausführung.

5.2.4 Auftragsannahme mittels Auftragskarten

MVP-Schritt 4: Gemäß dem aufgestellten Konzept hat eine Auftragsannahme des FTF stets über das Auftragsvergabemodell zu erfolgen. Da es im Rahmen dieser Bearbeitungsdauer nicht möglich ist, das gesamte Vorhaben zu realisieren, wird an dieser Stelle ein Abstrich gemacht und ein vereinfachtes Vorgehen zur Auftragsübermittlung eingeführt, um die Funktionen des FTF validieren zu können, denn wie ein Auftrag letztendlich von dem FTF angenommen wird, spielt für das Ausführen der Funktionen keine Rolle. Somit wird die Auftragsverhandlung übersprungen und dem FTF der auszuführende Auftrag direkt zugeführt.

Das FTF benötigt unter der im weiteren Entwicklungsprozess zu entwickelnden Aufnahmevorrichtung ein zusätzliches RFID-Lesegerät, wodurch die Informationen aus dem RFID-Tag der entgegengenommenen Transportkiste eingelesen werden können. Für die alternative Auftragszuweisung wird eine Auftragskarte in Form eines separaten RFID-

[20] Über Amazon.de zu erwerben mit der ASIN-Nr: B01HEU96C6

Tags dem Lesegerät zugeführt, welches die Informationen zu der Positionsklasse und den Positionskoordinaten beinhaltet, wobei die Positionsklasse „A“ sein muss, da die Fahrzeuge nur Arbeitsplätze als Ziele anfahren.. Somit erhält das FTF einen Transportauftrag mit den Zielkoordinaten, welches nach der Berechnung der kürzesten Route mit dem A*-Algorithmus (vgl. 5.2.5) umgehend ausgeführt.

Wichtig ist, dass beide Lesegeräte simultan lesen und schreiben können. In diesem Sinne gibt es innerhalb der Funktion **„Rfid_lesen()“** eine Schleife in der bei jedem Aufruf beide RFID-Lesegeräte nacheinander einen Lesevorgang durchführen. Begonnen wird immer mit dem unteren Lesegerät. Sofern beiden Lesegeräten also zeitgleich RFID-Tags zugeführt werden, erhält der untere immer zuerst die Information. Damit wird zudem eine Fehlermöglichkeit vermieden, dass ein Auftrag nie eintreffen kann, ohne dass das FTF seine aktuelle Position kennt.

5.2.5 Routenfindung mit dem A*-Algorithmus

MVP-Schritt 5: Für die Berechnung einer Route werden zwei Informationen benötigt. Zum einen die Startposition und zum anderen die zu erreichende Zielposition. Das FTF kennt mittels des unteren RFID-Lesegerätes stets seine aktuelle Position. Die Zielposition wird aus der dem oberen RFID-Lesegerät zugeführten Auftragskarte entnommen. Bei jedem Aufrufen der Funktion „Rfid_lesen()“ werden die unten aufgeführten vier Variablen im RFID-Leseprozess mit Informationen befüllt. Diese Variablen sind mit dem Datentyp **„char“** deklariert, welches ein Textformat ist. Anhand dieser Variablen werden in der Funktion zur Routenberechnung die Start- und Zielpositionen festgelegt.

```
char rfid_klasse_unten[18];
char rfid_klasse_oben[18];
char rfid_pos_unten[18];
char rfid_pos_oben[18];
```

Wie bereits erwähnt, wird zur Routenberechnung der A*-Algorithmus verwendet (vgl. 2.3). Dazu wurde ein fehlerhafter Ansatz eines Forenmitgliedes aus dem Internet aufgegriffen, welcher nicht funktionsfähig war und von Grund auf verändert und sehr stark mit Erweiterungen ergänzt wurde (M_Jahangeer_Qureshi, 2016). Der originale Ursprung des Programmcodes ist aus dem Foreneintrag nicht eindeutig ersichtlich.

Nach dem Eingang der Start- und Zielkoordinaten findet vorerst ein Initialisierungsprozess für den A*-Algorithmus statt. In diesem finden die Vorbereitungen für die Ausführung des Rechenvorganges statt. Zu Beginn werden die gelesenen Informationen aus

den RFID-Tags auf die Variablen zugewiesen. Sowohl die Start- auch als die Zielkoordinaten werden in die x- und y-Komponenten unterteilt. Daher gibt es insgesamt vier Variablen, wobei „s“ für Start und „z“ für Ziel steht. Da die gelesenen Informationen aus den RFID-Tags im Textformat mit dem Datentyp **„char“** gespeichert sind, wird vorerst die Variable in den Datentyp **„String“** konvertiert und in Verbindung mit der Kombination der **„Substring-Methode“** und der **„toInt-Methode“** in ein Zahlenformat gebracht. Mit diesem Vorgehen werden ausgewählte Bereiche eines Textes selektiert und einer Variable im Zahlenformat (hier: byte) zugeordnet:

```
s_x = String(rfid_unten).substring(1, 2).toInt();
s_y = String(rfid_unten).substring(3, 4).toInt();
z_x = String(rfid_oben).substring(1, 2).toInt();
z_y = String(rfid_oben).substring(3, 4).toInt();
```

Die anfangs zugewiesene Halteposition des FTF ist mit der Position „x0y0“ festgelegt. Für den Fall, dass das FTF von seiner Ladestation aus startet, hat die Variable **„s_x“** den Wert „0“ und die Variable **„s_y“** ebenso den Wert „0“. Berücksichtigt werden nur die Zahlenbereiche der gelesenen Positionskoordinaten.

Das vereinfachte Testfeld muss für die Routenberechnung in einer Matrix abgebildet werden. Gemäß der Beschreibung des A*-Algorithmus werden in der Matrix die Hindernisse mit dem Wert 0 und befahrbare Positionen mit dem Wert „1“ belegt. Sollte eine Zielposition beauftragt werden, welches ein Hindernis darstellt, so bricht die Initialisierung an der Stelle ab. Sofern das FTF an einem PC angeschlossen betrieben wird, erscheint im seriellen Monitor eine Fehlermeldung mit der Information über die Fehlerursache. Das FTF geht wieder in Wartezustand und wartet auf einen neuen Auftrag. Die Eingabe dieser Matrix als Programmcode, welches auch als Karte bezeichnet wird, erfolgt wie nachstehend:

```
byte matrix[Zeilen][Spalten] = {
  {1, 0, 0},
  {1, 1, 1},
  {1, 1, 1},
  {1, 1, 1},
};
```

Ausgehend der erhaltenen Start- und Zielpunkte wird gemäß der Ausgangsmatrix eine weitere Matrix mit der Anwendung einer Heuristik generiert. Hierzu wird, wie in den Grundlagen bereits beschrieben, die Manhattan-Distance-Methode verwendet. In dieser ist aufgeführt, wie weit sich ein beliebiger Punkt auf der Matrix von dem zu erreichenden Zielpunkt entfernt befindet. Zu den Schätzkosten der Positionen, welche ein Hindernis sind, werden ein im Vergleich zu der Matrixdimension höherer Wert dazu addiert, um diese im Rechenvorgang des Algorithmus ausscheiden zu lassen.

Im Berechnungsvorgang der Route wird gemäß der beschriebenen Vorgehensweise zunächst die mit dem geringsten f-Wert erreichbaren Position betrachtet. Im Laufe der Routenermittlung werden die Positionen, die in die Route aufgenommen werden, schrittweise in zwei Arrays abgespeichert. Dazu werden die x-Koordinaten der Positionen in die Variable **„weg_x[anzufahrendeKnoten]“** und die y-Koordinaten in die Variable **„weg_y[anzufahrendeKnoten]“** geschrieben. Diese Arrays haben eine dynamische Größe, da sie abhängig der Fahrstrecke unterschiedlich groß sein können. Demnach steht die Array-Dimension in Abhängigkeit zu der Variable **„anzufahrendeKnoten“**. Diese beiden Arrays sind das Ergebnis aus der Routenberechnung mit dem A*-Algorithmus. Nachdem die Routenermittlung abgeschlossen ist, werden die für die Berechnung verwendeten Variablen zurückgesetzt.

5.2.6 Finaler Funktionsumfang – Zusammenspiel aller Funktionen

MVP-Schritt 6: Mit dem letzten MVP-Schritt wird die Teilentwicklung des Konzeptes abgeschlossen. Darin werden alle entwickelten Funktionen gemeinsam angewendet, sodass mit dem vereinfachten Testfeld die Entwicklung validiert werden kann. Der Programmcode besteht aus insgesamt vier einzelnen Seiten (Dateien) - einer Hauptprogrammseite und drei Unterprogrammseiten, in denen die benötigten Funktionen enthalten sind und je nach Ereignis und Bedingung aufgerufen werden. Mit dem Öffnen der Hauptdatei in der Arduino-IDE werden automatisch die Unterdateien des Projektes geöffnet. Die Struktur mit allen enthaltenen Funktionen ist in

Abbildung 48 dargestellt. Innerhalb der Hauptprogrammseite wird der grundlegende Prozess abgebildet und je nach Bedingung die Funktionen aus den Unterseiten aufgerufen. Dabei werden Variablen übergeben und entgegengenommen. Zu Beginn wird das FTF in seine Ausgangslage mit der Richtung „S“ auf seiner festen Ladeposition (Halteplatz) positioniert. Mit der Betätigung des Stromschalters des FTF startet das sich auf dem Arduino hochgeladene Programm.

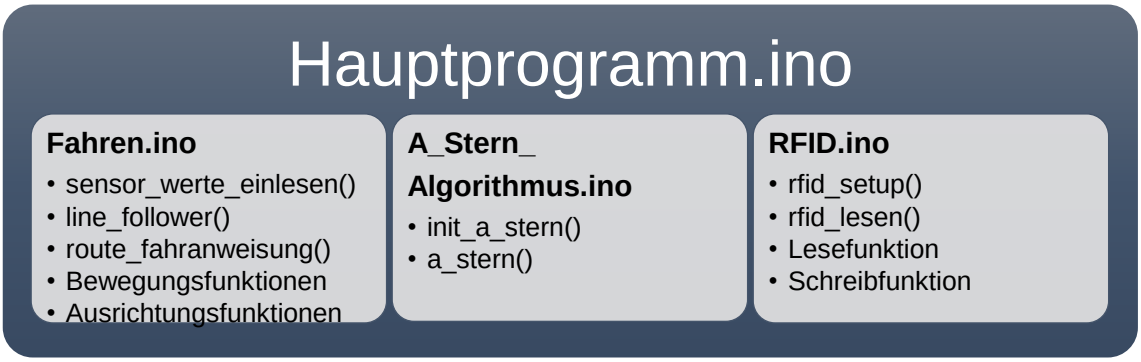


Abbildung 48: Struktur des Programmcodes

Wie bereits erwähnt, besteht ein Projekt im Arduino immer aus einem Setup- und einem Loop-Teil. Nachdem die Konfiguration im Setup-Teil abgeschlossen ist, läuft der Arduino in einer Dauerschleife im Loop-Teil, indem der Code ständig wiederholt wird. Daher befindet sich das FTF so lange im Wartezustand, bis ein neuer Transportauftrag dem oberen RFID-Lesegerät zugeführt wird. Dazu wird am Anfang des Loop-Teils mit einer Schleife durchgehend die Funktion **„rfid_lesen()“** aufgerufen, mit dem der Leseprozess für beide RFID-Lesegeräte ausgeführt und nach neuen Tags geprüft wird. Der gelesene Inhalt wird dann auf zwei Variablen je RFID-Lesegerät gespeichert (Abbildung 49). Am Ende jeder Schleife wird anschließend geprüft, ob die Variable **„rfid_klasse_oben“** den Inhalt **„A“** aufweist. Sobald eine Auftragskarte eingelesen wird, trifft die Bedingung für das Verlassen der Schleife ein. Damit ist sichergestellt, dass nur bei einer zugeführten Auftragskarte das Fahrzeug anfängt die weiteren Teile des Programmablaufes auszuführen.

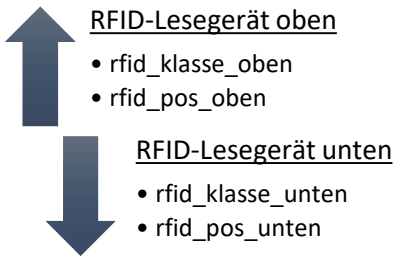


Abbildung 49: RFID-Variablen

```
do {
    rfid_lesen();
} while (String(rfid_klasse_oben) != "A");
```

Nachdem ein Auftrag eingegangen ist, startet die Initialisierung des A*-Algorithmus für die Routenberechnung. Start- und Zielkoordinaten werden übergeben und dabei überprüft, ob das Ziel angefahren werden kann oder nicht. Sollte ein Zielpunkt beauftragt werden, welches auf der Matrix als ein Hindernis klassifiziert ist, so erscheint eine Fehlermeldung und das FTF begibt sich wieder zu der obigen Schleife zurück, um nach einem neuen Transportauftrag zu warten. Sofern die Voraussetzungen stimmen, wird die Initialisierung abgeschlossen und der A*-Algorithmus ausgeführt, um den kürzesten Weg zum Zielpunkt zu berechnen. Wie im vorigen Abschnitt bereits beschrieben, werden am Ende der Routenberechnung zwei Arrays ausgegeben, welche schrittweise die einzelnen x- und y-Koordinaten beinhalten. Jeder Arrayblock entspricht zugleich einem Zeitslot, in der das FTF sich zur nächsten Position bewegen wird. Diese beiden Arrays werden der Funktion **„route_fahranweisung()“** übergeben, welches das FTF mit den auszuführenden Fahrbewegungen steuert.

Für diese Steuerung des FTF mit den übergebenen Koordinaten sind vorerst einige weitere Funktionen zu erläutern. Das Fahren des FTF auf dem Testfeld erfordert eine Vorgehensweise, mit der die Ausrichtung dynamisch bestimmt und gesteuert werden kann, da keine selbstbestimmende Lokalisierung und Ausrichtungserkennung mit anderweitigen Technologien realisiert wurde. Daher wurde ein Algorithmus entwickelt, welches in Abhängigkeit der nächsten anzufahrenden Position am Anfang der Funktion

„route_fahranweisung()“ eine Neuausrichtung vornimmt. Ohne eine Neuausrichtung kann die gewünschte Position nicht angefahren werden. Hierfür wird zu Beginn betrachtet, in welcher Fahrtrichtung sich das FTF aktuell befindet und welche Fahrtbewegung vorgenommen werden muss, um den nächsten anzufahrenden Punkt zu erreichen. Die Ausrichtung wird mittels Richtungsangaben beschrieben. Dabei gilt das Prinzip der Kennzeichnung nach „W-A-S-D“ (Abbildung 50), welche in der Computerumgebung als das Pendant zu den Pfeiltasten gelten, so wie es beispielsweise in Ego-Shooter-Spielen der Fall ist. Durch das Speichern der Richtungsinformation in die Variable mit der Bezeichnung **„ausrichtung“** kann die aktuelle Ausrichtung festgehalten werden. Zu Beginn muss dieser Variable muss einmalig eine feste Ausrichtung zugewiesen werden, wovon ausgehend das Programm die zu tätigenden Neuausrichtungen während des Fahrens ermittelt. Die festgelegte Ausgangslage für das FTF ist die Richtung **„S“**, wobei dies keine feste Vorgabe ist und die Ausgangslage jederzeit im Programmcode verändert werden kann. Sofern das Fahrzeug am Anfang falsch positioniert wird, funktioniert die gesamte Prozedur nicht mehr korrekt. Daher ist diese Vorbereitung essentiell für die Funktionserfüllung.

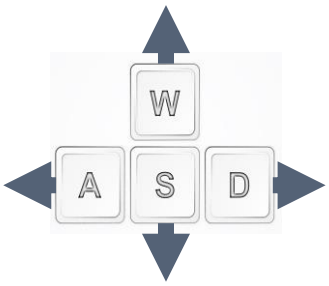


Abbildung 50: Ausrichtungsnotation

Dabei gibt es drei Ausrichtungsfunktionen (Abbildung 51), die je nach der zutreffenden Bedingung (Abbildung 52) des entwickelten Algorithmus aufgerufen werden. Die Funktion **„route_fahranweisung()“** arbeitet die Route in Schleifen ab. Die Anzahl der Schleifendurchläufe ist dabei abhängig von der Größe der übergebenen Koordinaten-Arrays. Anhand der nachfolgend aufgeführten Arrays zu einem Beispiel mit Startposition „x0y0“ und Zielposition „x3y1“ soll der Algorithmus zu der Ausrichtungskorrektur näher beschrieben werden.

```
weg_x = [0; 1; 1; 2; 3]
weg_y = [0; 0; 1; 1; 1]
```

Die mit dem A*-Algorithmus berechnete Route gibt eine Fahrstrecke von vier Positionen aus. Demnach wird die Schleife in der Funktion **„route_fahranweisung()“** genauso oft durchlaufen. In jeder Schleife wird die Veränderung des zu betrachtenden nächsten Array-Blocks überprüft. In der ersten Bewegung findet Veränderung in x-Richtung statt, da sich der Wert von „0“ auf „1“ verändert. In der zweiten Bewegung hingegen muss das FTF sich in y-Richtung bewegen, da der Wert in x-Richtung gleich bleibt und die y-Richtung den Wert „1“ aufweist. In Abbildung 52 wird die Anwendung des Algorithmus in

Abhängigkeit zu zwei Bedingungen aufgezeigt. Zunächst wird geprüft wie die Veränderung der Arrays zu dem nächsten Zeitslot aussieht. Anschließend wird die momentane Ausrichtung anhand der Variable „ausrichtung“ abgefragt. Aus den beiden Informationen entscheidet der Algorithmus, ob eine Neuausrichtung vorgenommen werden muss.

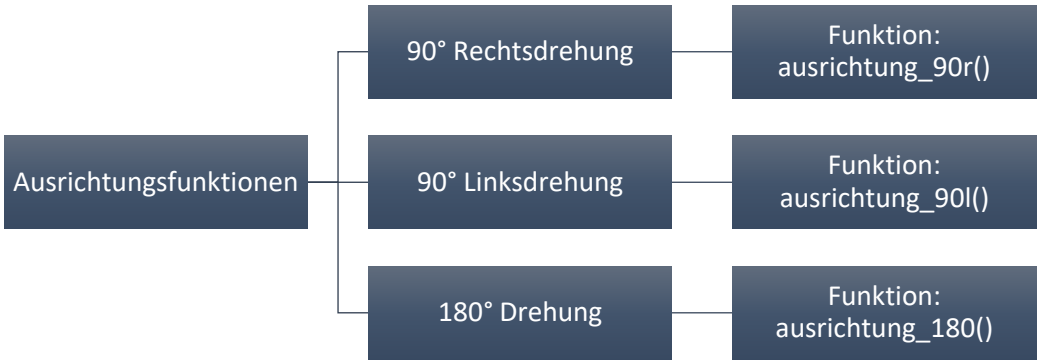


Abbildung 51: Übersicht der Ausrichtungsfunktionen

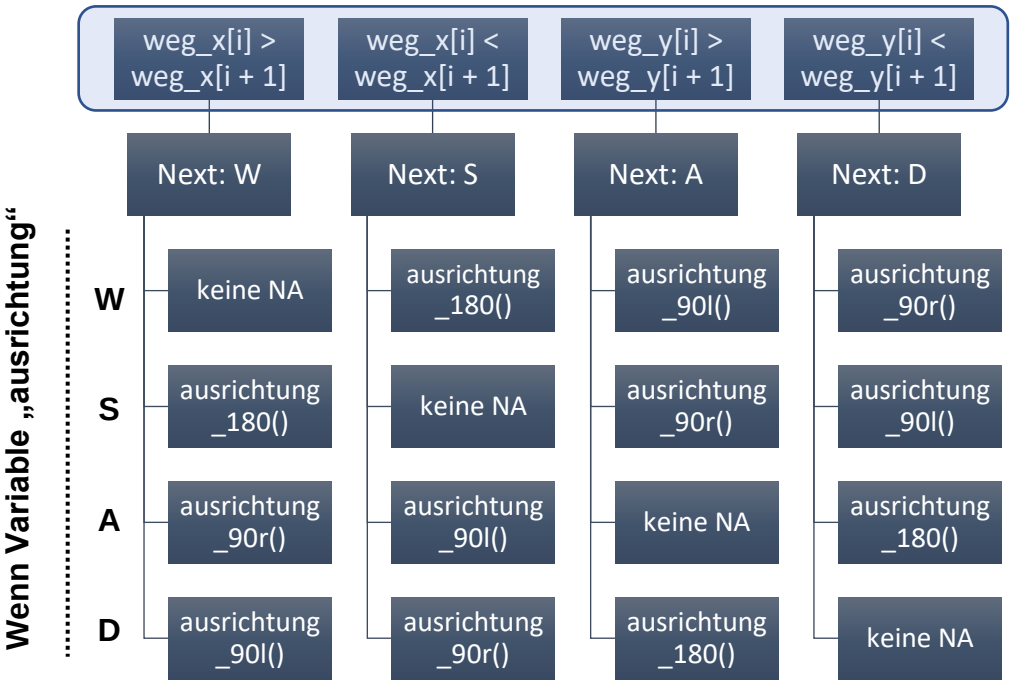


Abbildung 52: Algorithmus zur Ausrichtungserkennung und -korrektur

Nachdem die Neuausrichtung erfolgt ist, fängt das FTF an mittels der Funktion **„line_follower()“** geradeaus auf der Spur bis zur nächsten Positionsmarke weiterzufahren. Das Erreichen einer Position wird durch das untere RFID-Lesegerät ermöglicht, da während des Fahrens ein kontinuierlicher Lesevorgang des RFID-Lesegerätes stattfindet, in der die Koordinaten der als nächstes anzufahrenden mit der laufend aktualisierten Variable **„rfid_pos_unten“** verglichen wird. Sobald übereinstimmen, wird die Schleife beendet. In der nächsten Schleife findet dann evtl. eine Neuausrichtung für die Fahrt zur nächsten Position statt. Sobald der Zielpunkt der Route erreicht ist, fängt eine am

„Pin13“ angeschlossene grüne LED für 10 Sekunden an zu blinken, welches die erfolgreiche Zielerreichung signalisiert.

Abschließend wird mit Abbildung 53 der gesamte Entwicklungsverlauf dargestellt. In dem MVP-Bild ist eindeutig ersichtlich, in welchem Maße sich das FTF von der ersten Teilentwicklung bis hin zum finalen Funktionsumfang verändert hat.

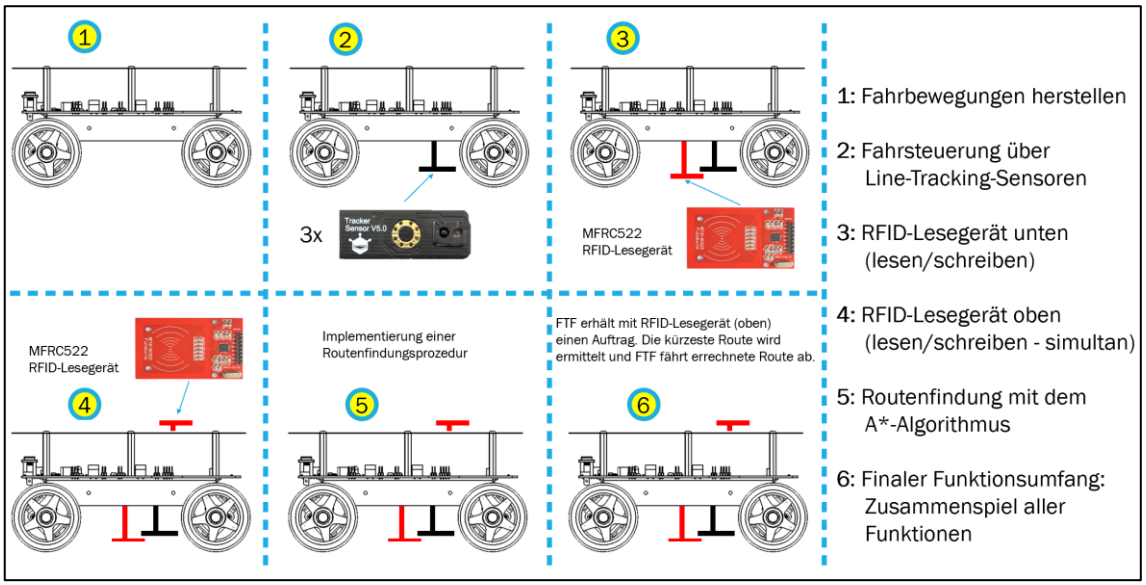


Abbildung 53: MVP-Bild zum Entwicklungsverlauf des FTF

6 Kritische Würdigung der Ergebnisse

Um die Ergebnisse und Erkenntnisse dieser Arbeit abschließend zu bewerten, werden nachstehend drei zu beantwortende Fragen aufgestellt. Durch die Beantwortung dieser Fragen wird zudem der Erfüllungsgrad der Anforderungen deutlich gemacht.

Eignet sich eine dezentrale Steuerung für fahrerlose Transportsysteme?

Zu Beginn der Arbeit wurde ein Vergleich zu den Arten der Produktionssteuerung – die zentrale und die dezentrale Steuerung – durchgeführt. In dem Vergleich wurden aus allgemeiner Sicht, ohne die Betrachtung eines expliziten Referenzsystems, die Vorzüge einer dezentralen Steuerung deutlich gemacht. Mit der Einführung einer dezentralen Steuerung wird eine Produktion flexibler und robuster gegenüber unerwarteten Ereignissen. Dabei ist das Ganze nicht allein aus Sicht der FTS zu betrachten, denn ein von dem gesamten Produktionssystem losgelöstes FTS ist nicht vorstellbar, sodass es auch nicht möglich ist, in einer Produktionsstätte zentral und dezentral zugleich zu steuern, weshalb eine Kohärenz zwischen den Systemen herrschen muss. Wie in Abschnitt 3.4 beschrieben, sind die heutigen FTS mit einer zentralen Steuerung sehr unflexibel, da sie keine eigenen Entscheidungsräume zur Verfügung gestellt bekommen und in einer sehr starken Abhängigkeit zu weiteren Instanzen stehen.

Vor allem der Aspekt des „Single-Point-of-Failure“ ist Grund genug, um von einer zentralen Steuerung der FTS wegzukommen, da sie für die gesamte Produktion ein zu großes Risikopotential besitzt. Ein Ausfall des Transportsystems bedeutet zugleich eine nahezu stillstehende Produktion, welcher wohlmöglich einer der Gründe ist, weshalb sich produzierende Unternehmen der Einführung eines solchen Systems noch nicht gewagt haben.

Wurden die genannten Anforderungen an die Konzeptionierung des Systems erfüllt? In wie weit ist das aufgestellte Konzept geeignet, um ein fahrerloses Transportsystem in der Realität damit zu steuern?

Die Erfüllung der Anforderungen liegt grundlegend darin, eine durchgängig konsistente Steuerungslogik zu schaffen. Mit der Fokussierung auf ein zentrales Element in Form einer Transportkiste, welches die eigenständig Auftragssteuerung übernimmt, ist quasi die heutige zentrale Steuerung überführt worden, in einzelne dezentrale Elemente, die die gleiche Aufgabe übernehmen, wie die zentrale Produktionssteuerung. Aufgrund des-

sen, dass jede auftragssteuernde Transportkiste fähig ist den Produktionsplan zu beeinflussen, ist ein hoher Grad an Flexibilisierung erreicht, da beispielsweise bei einem Ausfall eines FTF, welches die Transportkiste zu einem Arbeitsplatz transportiert, umgehend die gebuchten Kapazitäten freigesetzt werden, für die die anderen Aufträge zugreifen können.

Aus dem Kommunikationsmodell ist ersichtlich, dass eine komplette Auftragsabwicklung innerhalb des Systems zwischen den einzelnen Entitäten reibungslos und planmäßig kommuniziert werden kann, denn in einem dezentralen System ist eine funktionierende Kommunikation die zwingende Grundlage für die Auftragssteuerung. Mit dem Einsatz des Kommunikationsprotokolls MQTT wurde dieser Anforderung nachgekommen. Für die Auswahl eines Kommunikationsprotokolls war ein sehr wichtiges Kriterium, die einfache Implementierung in das Arduino-System zu ermöglichen. Zudem können im weiteren Entwicklungsprozess, ohne eine Veränderung des Hauptprogramms, weitere Teilnehmer in unbegrenzter Dimension hinzugefügt werden, welches zu einer sehr hohen Variabilität aus Sicht der Erweiterbarkeit der Modellierung führt. Nicht nur die Erweiterbarkeit, sondern auch die Ersetzbarkeit der Komponenten ist auf eine einfache Art und Weise möglich, da alle Komponenten entkoppelt vom System existieren. Daher müsste nur die Kennung der zu ersetzenden Entität übertragen werden, wie beispielsweise die FTF-Nummer.

Mit dem Auftragsvergabemodell ist die aktive Teilnahme der benötigten Entitäten an der Produktionssteuerung ermöglicht. Sie können dadurch in Verhandlungen eintreten, um benötigte Ressourcen zu buchen, die für die Auftragserfüllung notwendig sind. Anhand der Annahme über die zeitliche Betrachtung durch ein Zeitslot-basiertes System ist zudem eine Feinplanung der gesamten Ressourcen und insbesondere der kollisionsfreien Routenplanung der FTF sichergestellt. Außerdem sind alle zeitlichen Abweichungen der einzelnen Entitäten, ausgestattet mit Arduino-Systemen, kompensiert, wodurch eine quantitative Bewertung der Ergebnisse erreicht werden kann.

Um dem aufgestellten Fallbeispiel zu der „ABC-Spielzeuge GmbH“ eine abschließende Antwort liefern zu können, ist zunächst einmal das gesamte Konzept zu realisieren. Nach der Realisierung des Konzeptes kann das aufgestellte Produktionsszenario erprobt und bewertet werden, welches kein Betrachtungsgegenstand dieser Arbeit mehr ist. Unabhängig der Ergebnisse aus der Modellierung ist dennoch die Übertragbarkeit in die Realität zu bewerten. Die in dem Konzept eingebundenen Technologien sind bereits aus dem realen Produktionsumfeld und mit ihrem erfolgreichen Einsatz bekannte Technologien. Es wird daher keine Behauptung aufgestellt Neuentwicklungen zu bewerkstelligen. So ist beispielsweise die RFID-Technologie seit sehr langer Zeit ein treuer Unterstützer

der Produktion. In diesem Konzept wurden diese und weitere Technologien hinsichtlich einer dezentralen Steuerungslogik zusammengeführt. Das Ziel dabei ist es damit ausgewählte Komponenten aus dem Produktionsprozess mit Hilfe der Möglichkeiten der Digitalisierung „intelligent“ zu machen, sodass sie in die Entscheidungswege der Produktionssteuerung einbezogen werden. Demnach steht einer Übertragung in das reale Produktionsgeschehen kein Hindernis im Wege. Lediglich die Entwicklung eines fahrerlosen Transportfahrzeuges sollte nicht spurgebundener Art sein, da der heutige Entwicklungsstand für FTF hinsichtlich ihrer freien Bewegungsmöglichkeiten durch hochgenaue Positionierungssysteme und ihrer implementierten Routenberechnungsalgorithmen sehr weitgeschritten ist. In dieser Arbeit fiel aufgrund des zeitlichen Rahmens und der Möglichkeiten zu einer einfachen Modellierung die Entscheidung für eine spurgebundene Steuerung.

Sind die entwickelten Funktionsumfänge des fahrerlosen Transportfahrzeuges für die abschließende Realisierung des Konzeptes ausreichend?

Gemäß den aus den ersten Projektgesprächen erarbeiteten Anforderungen und der Anwendung der MVP-Entwicklungsmethode wurde das vorgestellte fahrerlose Transportfahrzeug auf Basis eines Arduino-Systems entwickelt. Das FTF ist fähig sich zu lokalisieren und mit dem kürzesten Weg den Zielpunkt seines Auftrages zu erreichen. Diese Teilentwicklung hatte das Ziel, als eine Entwicklungsgrundlage in Form eines Prototyps für die Realisierung des gesamten Konzeptes zu dienen. Die Funktionen wurden in dem geforderten Rahmen zufriedenstellend erreicht, da durch die Anwendung der MVP-Entwicklungsmethode die Entwicklungsschritte und die Funktionen stets abgestimmt und validiert wurden. Das FTF wurde erfolgreich in einer Veranstaltung[21] an der HAW Hamburg am 12.06.2018 der Öffentlichkeit vorgeführt.

Die Anforderung zur Routenermittlung wurde zudem übertroffen, da die Anforderung nach einer einfachen Prozedur mittels einer Adjazenzmatrix war. Es wurde trotz dessen, unter Berücksichtigung der Möglichkeiten, ein weit bekannter Algorithmus zur Routenermittlung implementiert – der A*Algorithmus. Obwohl die Ressourcen eines Arduino-Systems ziemlich eingeschränkt sind, ist es dennoch gelungen diese Funktion zu entwickeln.

Abschließend ist mit großem Erfreuen die Erreichung aller Ziele bekannt zu geben.

[21] Veranstaltung: „Produktionsmanagement stellt sich vor“
durch das Institut für Produkt- und Produktionsmanagement (IPP) in BT21, F213

7 Weiterfolgende Handlungsempfehlungen für die Realisierung des Konzeptes

Für die Realisierung des aufgestellten Konzeptes ist noch weitere Entwicklungsarbeit notwendig. Um erste Anreize hinsichtlich der Lösungsfindung zu ermöglichen, werden in diesem Abschnitt in kurzer Form die offenen Punkte und mögliche Vorgehensweisen zur Erreichung beschrieben. Dafür wird für jede Entität einzeln und in Stichpunkten zuerst die fehlenden Komponenten und anschließend die offenen Punkte beschrieben. Diese Art der Darstellung ist gewählt worden, um den Effekt einer Abarbeitungsliste zu erzeugen, bei der sich anschließend entweder dafür oder dagegen entschieden werden kann.

Allgemeingültig:

- *Kommunikationsfähigkeit mittels MQTT:* Die Kommunikation mit dem MQTT-Protokoll erfordert eine Anbindung in das Internet, weshalb alle Komponenten mit einem Netzwerkmodul ausgestattet werden müssen. Dadurch können sich die Entitäten mit dem Server des MQTT-Brokers verbinden und anfangen zu kommunizieren, indem sie Nachrichten senden und empfangen. Hierfür eignet sich der Einsatz des Moduls „ESP8266“, für den es diverse Bibliotheken hinsichtlich des Einsatzes mit MQTT gibt. Als Beispiel gilt hierfür die Bibliothek des Entwicklers Tuan (Tuan, 2014/2018).

- *Anlegen einer MySQL-Datenbank:* Im Konzeptteil wurde die Anwendung von MySQL-Datenbanken einbezogen. Eine solche Datenbank ist daher gemäß dem aufgestellten Layout aufzubauen. Solche Datenbanken eignen sich sehr gut, um darin Daten abzuspeichern oder abzurufen. Um eine Datenbank anzulegen, muss zunächst ein MySQL-Server entweder gemietet oder von einem eigenständigen PC aus eingerichtet werden.

- *Zugriff auf Daten aus einer MySQL-Datenbank:* Der Zugriff auf eine MySQL-Datenbank mit einem Arduino kann auf einfache Art über eine öffentlich verfügbare Bibliothek des Entwickler Bell mit der Bezeichnung „MySQL Connector Arduino“ realisiert werden, wodurch sowohl das Speichern als auch das Abrufen von Informationen auf einer MySQL-Datenbank ermöglicht wird (Bell, 2016/2018).

- *LED-Tafel zur Anzeige des aktuellen Zeitslots:* Damit beobachtet werden kann, in welchem Zeitslot das System ist, könnte an separater Stelle ein zusätzlicher Arduino mit einer LED-Tafel und Internetanbindung aufgebaut werden.

- *Stromversorgung:* Genauso wie bei den FTF kann die Stromversorgung für die nachfolgend aufgeführten Komponenten über ein Batteriepack realisiert werden. Siehe dazu Abschnitt 5.2.1.

Fahrerloses Transportfahrzeug:

- *Fehlende Bauteile/Module:*
  - ESP8266 für Internetanbindung
- *Aufnahmevorrichtung für Transportkisten:* Die Aufnahmevorrichtung dient zum einen für den sicheren Transport der Transportkiste und zum anderen für die Entgegennahme der auftragsanfragenden Transportkiste. Hierzu könnte eine Vorrichtung aus Plexi-Glas gefertigt werden. Dieses Material ist leicht in der Bearbeitung und verleiht zudem ein schönes Aussehen. Die Maße müssen dabei an die noch auszuwählenden Transportkisten angepasst werden, da die Transportkiste ziemlich fest auf der Vorrichtung fixiert sein muss, um während des Fahrens nicht zu ruckeln. Die Vorrichtung ist zudem maßlich so zu konzipieren, dass sowohl der auf der Unterseite der Transportkisten angebrachte RFID-Tag, als auch das RFID-Lesegerät sich genau treffen, um die Lesesicherheit zu gewährleisten.
- *Abschließen der Kollisionsvermeidung:* Die Funktionen zur Bewegungssteuerung und Routenberechnung sind erfolgreich entwickelt worden. Ein möglicher Ansatz zur Erweiterung des Algorithmus zur Routenberechnung mit Kollisionsvermeidung wurde in 4.5.2 angesprochen, welches aufgrund der Anforderungen im praktischen Teil dieser Arbeit nicht realisiert worden ist, da aktuell nur ein FTF als Prototyp existiert. Die Routenberechnung der FTF erfolgt gemäß definierter Umgebungskarten auf Basis einer Matrix im Arduino. In dieser Matrix ist fest definiert, welche Positionen befahrbar oder nicht befahrbar sind. Bei nur einem einzigen FTF ist der aktuelle Ansatz genügend, doch bei dem Hinzufügen eines weiteren FTF im weiteren Entwicklungsprozess muss bei der Routenberechnung auch die Position des anderen FTF berücksichtigt werden. Demnach ist eine kollisionsfreie Routenplanung notwendig.

  Dies kann ermöglicht werden, wenn die oben erwähnte Matrix während des Prozessablaufes dynamisch angepasst werden kann. Demnach würde eine Position, welches den Wert „1“ für befahrbar aufweist, zu einem bestimmten Zeitslot(t) den Wert „0“ annehmen. Da es nicht möglich ist, mit einer einzigen Matrix den Zustand unterschiedlicher Momente abzubilden, kann als Ansatz, bei der Annahme

einer Simulationsdauer von einem ganzen Produktionstag mit einer Dauer von 3600 ZS (1 Minute = 1 Zeitslot), exakt 3600 Matrizen erstellt werden, die zu den jeweiligen Zeitslots die Positionszustände bereitstellen. In Abbildung 54 ist der inhaltliche Aufbau einer Matrix, anlehnend auf das im Konzeptteil vorgestellte Testfeld, abgebildet. Arbeitsplätze als feste Hindernisse sind mit einer „0“ beschrieben und alle weiteren Positionen, die als Fahrwege gelten, sind in der Regel frei befahrbar, können aber in einer zeitlichen Abhängigkeit aufgrund einer momentanen Belegung durch einen FTF den Wert „0“ annehmen. Diese Aufgabe müsste von einem separaten Arduino übernommen werden, welches lediglich eine Internetanbindung und eine Stromversorgung benötigt.

$$\begin{bmatrix} 1(t) & 1(t) & 1(t) & 1(t) & 1(t) & 1(t) & 1(t) \\ 1(t) & 1(t) & 1(t) & 1(t) & 0 & 0 & 1(t) \\ 1(t) & 0 & 0 & 1(t) & 1(t) & 1(t) & 1(t) \\ 1(t) & 1(t) & 1(t) & 1(t) & 0 & 0 & 1(t) \\ 1(t) & 0 & 0 & 1(t) & 1(t) & 1(t) & 1(t) \\ 1(t) & 1(t) & 1(t) & 1(t) & 0 & 0 & 1(t) \\ 1(t) & 1(t) & 1(t) & 1(t) & 1(t) & 1(t) & 1(t) \end{bmatrix}$$

Abbildung 54: Inhalt einer Matrix als Umgebungskarte Momente ZS(t)

Transportkisten:

- *Fehlende Bauteile/Module:*
  - Transportkisten
  - Arduino-Mikrocontroller
  - ESP8266 für Internetanbindung
  - RFID-Lesegerät
  - RFID-Tags
  - Stromversorgung
- *Ausstattung mit Intelligenz:* Durch die Anbringung von Arduinos, welches in der Anschaffung keine sehr hohen Kosten verursachen, können die Transportkisten im Produktionsgeschehen aktiv teilnehmen. Da sie kostengünstig sind, ist die Ausstattung mehrerer Transportkisten mit den Arduinos kein Problem. Diese sind so anzubringen, dass im Produktionsprozess keine Beschädigungen geschehen können. Ein möglicher Ansatz ist in
- Abbildung 55 vorgestellt. Durch eine Einlegeplatte kann ein Unterraum der Transportkiste für die Technik genutzt werden. Ein großer Vorteil durch eine Einlegeplatte ist zudem die problemlose Wartbarkeit bei Störungen.

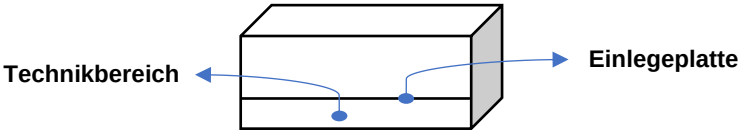


Abbildung 55: Möglicher Aufbau einer Transportkiste

- *RFID-Lesegerät innerhalb der Transportkisten:* Das entwickelte FTF beinhaltet bereits einen oben angebrachten RFID-Lesegerät, welches im Rahmen dieser Arbeit für die temporäre Auftragsannahme mit Hilfe von Auftragskarten genutzt wurde. Da die Arduinos nur eine Aufgabe zur selben Zeit vornehmen können, wäre ein zwischenzeitliches Beschreiben des RFID-Tags der Transportkisten mit dem Lesegerät des FTF nicht möglich, wenn beispielsweise die Priorität des Auftrages verändert werden müsste. Daher muss ein RFID-Lesegerät in dem Technikbereich genau über der Position des RFID-Tags angebracht werden. Damit kann bei Bedarf umgehend das RFID-Lesegerät eingesetzt werden.

Arbeitsplätze:

- *Fehlende Bauteile/Module:*
  - Attrappe eines Arbeitsplatzes
  - Arduino-Mikrocontroller
  - ESP8266 für Internetanbindung
  - RFID-Lesegerät
  - Rote und grüne LED
  - Stromversorgung
- *Attrappe für die Darstellung eines Arbeitsplatzes:* Aufgrund dessen, dass keine aktive Werkstückbearbeitung in der Modellierung stattfindet, werden die Bearbeitungen lediglich durch das Verweilen der Transportkiste an dem Arbeitsplatz dargestellt. Auf dem Testfeld sind hierfür beispielsweise kleine Attrappen aus Lego oder einer anderen Art aufzustellen. Dadurch wird die Veranschaulichung stark verbessert.
- *RFID-Lesegerät für Aktualisierung des Bearbeitungsstandes:* Um den Empfang und die erfolgte Bearbeitung der Transportkisten zu bestätigen, werden auch an einem Arbeitsplatz RFID-Lesegeräte benötigt. Sobald die vorgesehene Bearbeitungszeit beendet ist, wird automatisch der Bearbeitungsstand auf dem RFID-Tag der Transportkiste aktualisiert, wodurch die Transportkiste umgehend beginnt sich für den nächsten Bearbeitungsschritt vorzubereiten.

- *Rote und grüne LED:* Um den aktuellen Zustand eines Arbeitsplatzes zu signalisieren werden LEDs eingesetzt. Eine rote LED entspricht dabei einer aktiven Bearbeitung eines Auftrages und eine grüne LED für die momentane Verfügbarkeit.

Kistenregal:

- *Fehlende Bauteile/Module:*
  - Attrappe eines Kistenregales
  - Arduino-Mikrocontroller
  - ESP8266 für Internetanbindung
  - RFID-Lesegerät
  - Grüne LED
  - Stromversorgung
- *Attrappe für die Darstellung eines Kistenregals:* Um den Prozessschritt der Arbeitsvorbereitung besser kenntlich zu machen, ist auch eine Attrappe für den Kistenregal erforderlich. Die Transportkisten können hinter dieser Attrappe gestapelt werden, da sie bei der Auftragszuweisung nur manuell zugeführt werden.
- *RFID-Lesegerät für Auftragszuweisung:* Sofern ein offener Auftrag in der Warteliste existiert, leuchtet eine grüne LED auf. Es signalisiert die Aufforderung eine leere Transportkiste auf den RFID-Lesegerät zuzuführen, um den eingegangenen Auftrag zuzuweisen. Während der Zuweisung erlischt die LED und blinkt nach Abschluss der Aufgabe für einige Sekunden, damit die Transportkiste entnommen und auf eine andere Position gestellt. Die Transportkiste bucht umgehend einen Arbeitsplatz und einen FTF zur Abholung. Sobald das FTF da ist, wird die Transportkiste manuell dem FTF zugeführt.

Verschiedenes:

- *Auftragsmaske als Eintragsassistent:* Eine größere Erweiterung wäre es eine Auftragsmaske als ein eigenständiges Tool zu programmieren, welcher als ein Eintragsassistent für neue Kundenaufträge dient. Diese Software müsste eine Verbindung zum MQTT-Broker aufbauen können und in einem Fenster für den Eintrag eines neuen Auftrages folgende Felder zum Ausfüllen bereitstellen:
  - **Artikel:** Auswahlmenü der Produkte mit Bezug zur MySQL-Datenbank
  - **Priorität:** Auswahl zwischen NP und HP

  Weitere Punkte können je nach Entwicklungsverlauf und -erweiterung hinzukommen.

- *Wechsel der RFID-Tags:* In der Entwicklung des Fahrzeuges und des Testfeldes wurden RFID-Tags in Form von Stickern genutzt. Diese Sticker weisen einen Durchmesser von 25mm auf. Während den Testfahrten wurde beobachtet, dass diese Größe hinsichtlich der Lesesicherheit ein Problem darstellen kann, da die genutzten RFID-Lesegeräte einen unwesentlich größeren Lesebereich besitzen. Daher ist es besser, andere RFID-Tags derselben Klasse mit einem größeren Durchmesser einzusetzen.

8 Schlussteil

8.1 Zusammenfassung

In dieser Arbeit wurde erfolgreich ein Konzept erstellt, mit dem eine dezentrale Steuerung von fahrerlosen Transportsystemen ermöglicht wird. Angefangen mit der Erläuterung von Grundlagen zu den angewendeten Themenfeldern bis hin zu einer Gegenüberstellung der zentralen und dezentralen Produktionssteuerung, wurde der theoretische Teil erarbeitet. Ein geeigneter Übergang zu dem Konzeptteil fand mit den Argumenten für eine dezentrale Steuerung von fahrerlosen Transportsystemen statt.

Der konzeptionelle Teil der Arbeit beinhaltet viele verschiedene Themenfelder, die zur Anwendung gekommen sind. Aktuelle Technologien aus der Informationstechnik wurden vereint, um eine Steuerung dieser Art aufstellen zu können. Hierzu gehören unter anderem der Einsatz von Mikrocontrollern wie Arduinos, die RFID-Technologie, die Einbindung von Datenbanken und noch weitere. Die dezentrale Steuerung des Konzeptes wurde darauf basierend anhand einer Modellierung aufgezeigt, in der die einzelnen Komponenten, ausgestattet durch IT-Module, aktiv am digitalen Produktionsgeschehen teilnehmen können. Das zentrale Element hierbei sind die Aufträge in Form von Transportkisten, die in der Hauptrolle für die dezentrale Steuerung stehen. Nur von den intelligenten Transportkisten gehen die Entscheidungen aus, die sie dazu befähigen. selbststeuernd die Auftragserfüllung in der Produktion zu steuern.

Als praktische Leistung der Arbeit galt es zudem eine Teilvalidierung des aufgestellten Konzeptes durchzuführen. Hierfür sollte ein Fahrzeugroboter mit einem Arduino-System und ein vereinfachtes Testfeld mit einer temporären Auftragsvergabe entwickelt werden, auf dem das Fahrzeug errechnete Transportrouten abfährt. Der Fahrzeugroboter stellt hierbei ein fahrerloses Transportfahrzeug in Form eines Prototyps dar. Aufgrund technischer Restriktionen des genutzten Arduino-Systems und der anfänglich besprochenen Anforderungen, wurde die Bewegungssteuerung des Fahrzeuges auf eine spurgebundene Art realisiert, in der es sich anhand Line-Tracking-Sensoren über die schwarze Spurfarbe orientiert. Das Fahrzeug wurde gemäß den aufgestellten Anforderungen erfolgreich entwickelt und in einer Veranstaltung an der HAW Hamburg der Öffentlichkeit präsentiert.

8.2 Ausblick

In dem Konzeptteil wurde zu Beginn ein Fallbeispiel aufgestellt, welches mit einer offenen Frage nach dieser Arbeit stehen bleibt. Es sollte eine Aussage über die erfolgte Modellierung getätigt werden, welches noch nicht möglich ist, da das Konzept zuerst vollständig realisiert werden muss. Die weitere Bearbeitung dieses Konzeptes resultiert somit in der Möglichkeit studentische Arbeiten daraus abzuleiten. Mit den Handlungsempfehlungen aus dem vorigen Kapitel wurden hierfür die Voraussetzungen geschaffen.

Das aufgestellte Konzept ist umfassend beschrieben. Der einzige Themenbereich, welches noch optimiert werden muss, ist die kollisionsfreie Routenplanung, welches in die Funktionen zur Routenberechnung mit dem A*-Algorithmus eingebaut werden muss. Empfehlungen zum Vorgehen wurden an gegebener Stelle erläutert.

Nach der Realisierung des Konzeptes und der erfolgreichen Testphase in der Modellebene, kann die dezentrale Steuerung mit den vorhandenen Mitteln der HAW Hamburg realitätsnah erprobt werden. Die Fertigungsschritte wurden daher vorausschauend so ausgewählt, dass alle Arbeitsgänge an der Hochschule durchgeführt werden können. Sowohl die spanende Bearbeitung, das Umformen, als auch der Montagearbeitsplatz können genutzt werden. Darüber hinaus besitzt die HAW Hamburg zwei fahrerlose Transportfahrzeuge des Herstellers „Mobile Industrial Roboters (MIR)“, die mit der Unterstützung der Departments aus „Fakultät Technik und Informatik“ für eine dezentrale Steuerung gemäß dem aufgestellten Konzept angepasst werden können.

Der Fokus in dieser Arbeit lag lediglich in der Aufstellung einer dezentralen Steuerung. Unabhängig davon können im weiteren Entwicklungsverlauf die Erfassung von logistischen Kennzielen näher betrachtet werden. Somit können Durchlaufzeiten der einzelnen Produkte oder die Transportzeiten hinsichtlich einer dezentralen Steuerung beobachtet und analysiert werden.

Bei der Übergabe des fahrerlosen Transportfahrzeuges in Form werden sämtliche Programmcodes auf einer CD bereitgestellt. Zusätzlich sind auf der CD Schaltpläne für den Aufbau des FTF enthalten, worin der Aufbau des Fahrzeug-Roboters und das Hinzufügen der verwendeten Module beschrieben werden. Durch die Verwendung der Software „Fritzing“ wird die gesamte Verkabelung zwischen dem Arduino und den Modulen übersichtlich dargestellt. Damit ist alles Notwendige für den Aufbau weiterer FTF gewährleistet. Für den Aufbau des Testfeldes werden vorbereitete RFID-Tags gemäß der vereinfachten Version des Testfeldes mitgegeben. Diese müssen nur noch auf einen hellen Unterboden korrekt positioniert werden. Anschließend werden die Linienspuren mit einem PVC-Band aufgebaut.

Literaturverzeichnis

Balboa, M. (2018). *Arduino RFID Library for MFRC522*. C++. Abgerufen von https://github.com/miguelbalboa/rfid (Original work published 2012)

Beaton, W. (2018, Mai 2). Eclipse Mosquitto [Project]. Abgerufen 15. Juli 2018, von https://projects.eclipse.org/projects/technology.mosquitto

Bell, D. C. (2018). *MySQL_Connector_Arduino: Database connector library for using MySQL with your Arduino projects*. C++. Abgerufen von https://github.com/ChuckBell/MySQL_Connector_Arduino (Original work published 2016)

BMBF (Hrsg.). (2013). Zukunftsbild „Industrie 4.0“, *Hightech-Strategie*, 36.

BMWi. (2016). *Netzkommunikation für Industrie 4.0* (Diskussionspapier) (S. 11). Berlin: Bundesministerium für Wirtschaft und Energie (BMWi). Abgerufen von https://www.plattform-i40.de/I40/Redaktion/DE/Downloads/Publikation/netzkommunikation-i40.pdf

Bolt, E. (2013, April 1). Pulse Width Modulation with analogWrite | Robotic Controls. Abgerufen 11. Juli 2018, von http://robotic-controls.com/learn/arduino/pulse-width-modulation-analogwrite

Botthof, A., & Hartmann, E. A. (Hrsg.). (2015). *Zukunft der Arbeit in Industrie 4.0*. Berlin: Springer Vieweg.

Dangelmaier, W. (2009). *Theorie der Produktionsplanung und -steuerung: im Sommer keine Kirschpralinen?* Berlin: Springer.

DFRobot. (2017a, Juni 8). Smart Grayscale sensor SKU:SEN0147 - DFRobot Electronic Product Wiki and Tutorial. Abgerufen 8. April 2018, von https://www.dfrobot.com/wiki/index.php/Smart_Grayscale_sensor_SKU:SEN0147

DFRobot. (2017b, Juni 29). Cherokey 4WD Mobile Platform (SKU:ROB0102) - DFRobot Electronic Product Wiki and Tutorial. Abgerufen 2. April 2018, von https://www.dfrobot.com/wiki/index.php/Cherokey_4WD_Mobile_Platform_(SKU:ROB0102)

DFRobot. (o. J.-a). Cherokey: 4WD Arduino Mobile Robot-DFRobot. Abgerufen 2. April 2018, von https://www.dfrobot.com/product-896.html

DFRobot. (o. J.-b). DFRobot Mega 2560 V3.0 (Arduino Mega 2560 R3 Compatible). Abgerufen 3. Mai 2018, von https://www.dfrobot.com/product-655.html

Dressler, J. (2013, November 7). Interview mit Joachim Dressler, Sierra Wireless: M2M als Voraussetzung für Industrie 4.0. Abgerufen von https://www.elektroniknet.de/markt-technik/industrie-40-iot/m2m-als-voraussetzung-fuer-industrie-4-0-102757.html

Drolshagen, R. (2015). *Evaluation von M2M-Kommunikationsansätzen für Industrie 4.0 und Herleitung von Entscheidungskriterien für deren Einsatz*. Hochschule RheinMain - Fachbereich Design Informatik Medien Studiengang Informatik, Wiesbaden. Abgerufen von https://wwwvs.cs.hs-rm.de/downloads/extern/pubs/thesis/drolshagen15.pdf

Eclipse Mosquitto - Download. (2018, Januar 7). Abgerufen 15. Juli 2018, von https://mosquitto.org/download/

Fay, A. (2016). *Industrie 4.0: Smart Factory*. Abgerufen von https://www.youtube.com/watch?v=z7R8jg4Texw

Feldmann, K., & Wolfgang, W. (2006). Autonom navigierende Fahrerlose Transportsysteme in der Produktion. In P. Levi (Hrsg.), *Autonome mobile Systeme 2005: 19. Fachgespräch Stuttgart, 8./9. Dezember 2005*. Berlin: Springer.

FIPA English Auction Interaction Protocol Specification. (2011). *XC00031F*, 6.

Fischer, J. (Hrsg.). (1997). *Dezentrale controllinggestützte (Auftrags-)Steuerungskonzepte für mittelständische Unternehmen* (Als Ms. gedr). Düsseldorf: VDI-Verl.

Fleisch, E., Weinberger, M., & Wortmann, F. (2017). Geschäftsmodelle im Internet der Dinge. In S. Reinheimer (Hrsg.), *Industrie 4.0: Herausforderungen, Konzepte und Praxisbeispiele* (S. 1–16). Wiesbaden: Springer Vieweg.

Furmans, K. (2015, September). *Industrie 4.0 und cyberphysische Systeme: Kurzlebiger Trend oder Arbeitswelt von morgen?* Gehalten auf der Fachtagung Sicherheit und Gesundheitsschutz in der Warenlogistik 2015, BGHW, Dresden. Abgerufen von https://www.bghw.de/arbeitsschuetzer/praevention-von-a-z/f-l/fachvortraege-auf-bghw-veranstaltungen/fachtagung-sicherheit-und-gesundheit-in-der-warenlogistik-2015/vortraege/industrie-4-0-und-cyberphysische-systeme-kurzlebiger-trend-oder-arbeitswelt-von-morgen

Gärtner, H. (2018, April). *Vorlesungsreihe Digitale Produktion: Produktionssteuerung Digital Teil 1*. Gehalten auf der Vorlesungsreihe Digitale Produktion, Hamburg.

Grinninger, J. (2012). *Schlanke Produktionssteuerung zur Stabilisierung von Auftragsfolgen in der Automobilproduktion*. München: Fml, Lehrstuhl für Fördertechnk Materialfluss Logistik.

Grote, B. (2013, Oktober 8). Produktionssteuerung - Produktion in Netzwerken [Wiki]. Abgerufen 28. Mai 2018, von https://wikis.fu-berlin.de/display/pin/Produktionssteuerung

Günther, M. (2016, März 10). Wie viel dezentrale Planung verträgt die Produktionssteuerung? Abgerufen 15. Februar 2018, von https://www.inform-software.de/blog/post/wie-viel-dezentrale-planung-vertraegt-die-produktionssteuerung

Günther, W. A. (2012). *Algorithmen und Kommunikationssysteme für die Zellularе Fördertechnik*. Garching b. München: Lehrstuhl für Fördertechnik Materialfluß Logistik (fml) TU München.

Institut für Integrierte Produktion, H. (2013). *Dezentrale, agentenbasierte Selbststeuerung von Fahrerlosen Transportsystemen (FTS) : Schlussbericht ; (Bewilligungszeitraum: 01.07.2011 - 30.06.2013)*. Hannover [u.a.].

Jahn, M. (2017). *Industrie 4.0 konkret: ein Wegweiser in die Praxis*. Wiesbaden: Springer Gabler.

Jena, S., & Liebelt, N. (2015). *Heuristische Algorithmen im Beispiel des A*-Algorithmus / 8-Puzzle* (Ausarbeitung für das Fach “Algorithmische Anwendungen”). FH Köln, Köln. Abgerufen von http://www.gm.fh-koeln.de/~hk/lehre/ala/ws0506/Praktikum/Projekt/E_gelb/ALA-HeuristischeAlgorithmen-Jena-Liebelt.pdf

Jodlbauer, H. (2008). *Produktionsoptimierung: wertschaffende sowie kundenorientierte Planung und Steuerung* (2., erw. Aufl). Wien: Springer.

Knauer, R. (2016, September 19). CoAP – Zukünftiges Standard-Protokoll für das Internet of Things? Abgerufen 11. Juli 2018, von https://blog.doubleslash.de/coap-das-zukuenftige-standard-protokoll-fuer-das-internet-of-things/

Kniberg, H. (2016, Januar 25). Making sense of MVP (Minimum Viable Product) – and why I prefer Earliest Testable/Usable/Lovable. Abgerufen 12. Juli 2018, von https://blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp

Kurbel, K. (2016). MRP II - Enzyklopaedie der Wirtschaftsinformatik. In *Enzyklopädie der Wirtschaftsinformatik Online-Lexikon*. Abgerufen von http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/informationssysteme/Sektorspezifische-Anwendungssysteme/MRP-II/index.html

Lass, S., & Theuer, H. (o. J.). Hybride Simulation Den besten Grad an dezentraler Produktionssteuerung bestimmen. Abgerufen 8. Juli 2018, von http://www.productivity.de/node/585

Lester, P. (2005, Juli 18). A* Pathfinding for Beginners. Abgerufen 14. Juli 2018, von http://homepages.abdn.ac.uk/f.guerin/pages/teaching/CS1013/practicals/aStarTutorial.htm

Lödding, H. (2016). *Verfahren der Fertigungssteuerung: Grundlagen, Beschreibung, Konfiguration* (3. Auflage). Berlin Heidelberg: Springer Vieweg.

Loos, P., & Allweyer, T. (2013). Dezentrale Planung und Steuerung in der Fertigung - quo vadis? In A. W. Scheer (Hrsg.), *Organisationsstrukturen und Informationssysteme auf dem Prüfstand: 18. Saarbrücker Arbeitstagung 1997 für Industrie, Dienstleistung und Verwaltung* (S. 83–99). Physica-Verlag HD. Abgerufen von https://books.google.de/books?id=VLUcBgAAQBAJ

Maritsch, M., Kittl, C., & Ebner, T. (2015). Sichere Vernetzung von Geräten in Smart Factories mit MQTT. In A. Weisbecker, M. Burmester, & A. Schmidt (Hrsg.), *Mensch und Computer 2015 - Workshopband*. Berlin, München, Boston: DE GRUYTER. https://doi.org/10.1515/9783110443905-032

Metz, B. (2014). Was ist ein Mikrocontroller?, 11.

M_Jahangeer_Qureshi. (2016, November 13). A* Pathfinding in the Arduino. *Arduino.cc - Forum*. Abgerufen 2. Mai 2018, von https://forum.arduino.cc/index.php?topic=435203.0

Mors, A., Zutt, J., & Witteveen, C. (2007). Context-Aware Logistic Routing and Scheduling. In *ICAPS 2007, 17th International Conference on Automated Planning and Scheduling* (S. 328–335).

Nebel, M., & Wild, S. (2018). *Entwurf und Analyse von Algorithmen: eine Einführung in die Algorithmik mit Java* (2., vollständig überarbeitete Auflage). Wiesbaden: Springer Fachmedien Wiesbaden GmbH.

Neubauer, R. (2014, Dezember 10). Cyber Physical Systems: Zentrale Bausteine von Industrie 4.0. Abgerufen 15. Februar 2018, von https://www.computerwoche.de/a/zentrale-bausteine-von-industrie-4-0,3090293

Niemann, B., Baum, M., Fricke, D.-H., & Overmeyer, L. (2006). Aufbau von Fahrerlosen Transportsystemen (FTS) durch eine dezentrale Datenstruktur. *Logistics Journal : nicht-referierte Veröffentlichungen*, *2006*(Oktober). https://doi.org/10.2195/LJ_Not_Ref_Overmeyer_102006

Nyhuis, P., & Wiendahl, H.-P. (2012). *Logistische Kennlinien: Grundlagen, Werkzeuge und Anwendungen* (3. Auflage). Berlin Heidelberg Dordrecht London New York: Springer Vieweg.

Obermaier, D. (2014, September). *Pub/Sub for the masses- Ein Einführungsworkshop in MQTT [GERMAN]*. Technologie. Abgerufen von https://de.slideshare.net/dobermai/pubsub-for-the-masses-ein-einfhrungsworkshop-in-mqtt-german

Obermaier, D. (2017, Dezember 8). Schlankes IoT-Protokoll: So funktioniert MQTT. Abgerufen 1. Juni 2018, von https://www.elektroniknet.de/elektronik/kommunikation/so-funktioniert-mqtt-148672.html

Piller, F. T. (1998). Das Produktivitätsparadoxon der Informationstechnologie. *WIST*, 257–262.

Ries, E. (2009a, März 23). What is the minimum viable product? Abgerufen von http://venturehacks.com/articles/minimum-viable-product

Ries, E. (2009b, August 3). Minimum Viable Product: a guide. Abgerufen 12. Juli 2018, von http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html

Sanjeev, A. (2014, Mai 16). How to Make a Line Follower Robot in 10 Minutes | Arduino. Abgerufen 12. März 2018, von https://maker.pro/arduino/projects/make-line-follower-robot

Sarshar, K. (2015, September 9). Das Produktivitätsparadoxon der IT - Antrittsvorlesung von Prof. Dr. Kamyar Sarshar. Abgerufen 7. Juli 2018, von https://idw-online.de/de/news637231

Schuh, G., & Stich, V. (Hrsg.). (2012). *Produktionsplanung und -steuerung. 1: Grundlagen der PPS* (4., überarbeitete Auflage). Berlin Heidelberg: Springer Vieweg.

Schwarz, C. (2014). *Untersuchung zur Steigerbarkeit von Flexibilität, Performanz und Erweiterbarkeit von Fahrerlosen Transportsystemen durch den Einsatz dezentraler Steuerungstechniken*. Universität Oldenburg, Oldenburg. Abgerufen von https://www.uni-oldenburg.de/fileadmin/user_upload/informatik/Dissertation_-_final_.pdf

Sendler, U. (2013). Industrie 4.0 – Beherrschung der industriellen Komplexität mit SysLM (Systems Lifecycle Management). In U. Sendler (Hrsg.), *Industrie 4.0: Beherrschung der industriellen Komplexität mit SysLM* (S. 1–19). Berlin: Springer Vieweg.

Skyetek Inc. (Hrsg.). (2017). USING MIFARE CLASSIC TAGS. Abgerufen von https://www.jadaktech.com/skyetekfiles/docs/m2/mifareclassic.pdf

Studytonight.com. (2018). Datatypes in C Language | C Language Tutorial | Studytonight. Abgerufen 14. Juli 2018, von https://www.studytonight.com/c/datatype-in-c.php

Tuan. (2018). *esp_mqtt: MQTT client library for ESP8266*. C. Abgerufen von https://github.com/tuanpmt/esp_mqtt (Original work published 2014)

TUM - Mathematik - M9. (o. J.). Abgerufen 15. Mai 2018, von http://www-m9.ma.tum.de/Allgemeines/GraphAlgorithmen

Ullrich, G. (2014). *Fahrerlose Transportsysteme: eine Fibel - mit Praxisanwendungen - zur Technik - für die Planung ; mit zahlreichen Tabellen* (2., überarb. und erw. Aufl). Wiesbaden: Springer Vieweg.

VDI-Richtlinie: VDI 2510 Fahrerlose Transportsysteme (FTS). (2005, Oktober). Abgerufen 14. Juli 2018, von https://www.vdi.de/nc/richtlinie/vdi_2510-fahrerlose_transportsysteme_fts_/

VDI-Richtlinie: VDI 4451 Blatt 7 - Leitsteuerung für FTS. (2015). Abgerufen 7. Juli 2018, von https://www.vdi.de/richtlinie/vdi_4451_blatt_7-kompatibilitaet_von_fahrerlosen_transportsystemen_fts_leitsteuerung_fuer_fts/

Velden, L. (2014). Der A*-Algorithmus. Abgerufen 15. Mai 2018, von https://www-m9.ma.tum.de/graph-algorithms/spp-a-star/index_de.html

Walenta, R., Schellekens, T., Ferrein, A., & Schiffer, S. (2017). A decentralised system approach for controlling AGVs with ROS. IEEE. https://doi.org/10.1109/AFRCON.2017.8095693

Westkämper, E., & Bauernhansl, T. (2014). *Enterprise-Integration: auf dem Weg zum kollaborativen Unternehmen*. (G. Schuh & V. Stich, Hrsg.). Berlin: Springer Vieweg.

Selbstständigkeitserklärung

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

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

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

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

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: Varal

Vorname: Ali

dass ich die vorliegende Masterarbeit bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Konzeptionierung und Aufbau eines Testfeldes zur dezentralen Steuerung eines fahrerlosen Transportsystems

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

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

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

Hamburg	16.07.2018	Ali Varal
Ort	Datum	Unterschrift im Original