

Potenzialanalyse von benutzergesteuerten Anpassungs- und Analysewerkzeugen für Wertschöpfungsketten

Potential analysis of low-end user-controlled adjustment and analysis tools for value chains

Bachelorarbeit von Robin Lucas Garbe

Tag der Einreichung: 21.01.2022

1. Gutachten: Prof. Dr.-Ing. Reiner Anderl

2. Gutachten: Prof. Dr. Peter Buxmann

Darmstadt



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Maschinenbau

Fachgebiet
Datenverarbeitung in der
Konstruktion

Robin Lucas Garbe
Matrikelnummer: 2634021
Studiengang: Informatik

Bachelor-Thesis
Thema: Potenzialanalyse von benutzergesteuerten Anpassungs- und Analysewerkzeugen für Wertschöpfungsketten

Eingereicht: 21.01.2022

Betreuer: Yübo Wang, M.Sc. M.A. & Timo Koppe, M.Sc.

Prof. Dr.-Ing. Reiner Anderl, Prof. Dr. Peter Buxmann
Fachgebiet Datenverarbeitung in der Konstruktion
Fachbereich Maschinenbau
Technische Universität Darmstadt
Otto-Berndt-Straße 2
D-64287 Darmstadt



Erklärungen

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Bachelor-Thesis mit dem Titel „Potenzialanalyse von benutzergesteuerten Anpassungs- und Analysewerkzeugen für Wertschöpfungsketten“ selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, 21.01.2022

Robin Lucas Garbe

Robin Lucas Garbe

Ich bin damit einverstanden, dass die TU Darmstadt das Urheberrecht an meiner Bachelor-Thesis zu wissenschaftlichen Zwecken nutzen kann.

Darmstadt, 21.01.2022

Robin Lucas Garbe

Robin Lucas Garbe

Bachelor-Thesis

für

Herr Robin Lucas Garbe (Matrikelnr.: 2634021)



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Potenzialanalyse von benutzergesteuerten Anpassungs- und Analysewerkzeugen für Wertschöpfungsketten

Potential analysis of low-end user-controlled adjustment and
analysis tools for value chains

- Die Planung und Steuerung von Fertigungsprozessen finden bei konventionellen Fertigungsstraßen meist auf spezieller, proprietärer Hardware und direkt bei der Maschine statt. Dies erschwert die dynamische Steuerung des Systems, vor allem aus der Ferne. Um eine erhöhte Interoperabilität und eine bessere Kontrolle über das Gesamtsystem sicherstellen zu können, kann der Aspekt der Steuerung von der proprietären Hardware gelöst und in eine externe Komponente abstrahiert werden. Im Rahmen dieser Arbeit soll untersucht werden, inwiefern die Prozessplanung und -steuerung vereinfacht und individualisiert werden kann, und welches Potenzial solche externen Werkzeuge für die Flexibilität von Fertigungsprozessen haben. Hierzu sollen Komponenten entwickelt werden, welche eine individualisierte Prozessplanung und -steuerung ermöglichen.
- **Kernaufgaben:**

- **Umfassende Literaturrecherche zu Industrie 4.0, Automatisierungstechnologien, und Produktionsplanung sowie Produktionskontrolle**
- **Identifikation der Probleme von konventionellen Planungs- und Steuerungssystemen von Produktionsstraßen**
- **Potenzialanalyse und Konzeption** von endbenutzergesteuerten und interoperablen Software-Werkzeugen zur Prozessplanung und -steuerung
- **Entwicklung und Einbindung eines Werkzeuges zur benutzerfreundlichen und webbasierten Prozesssteuerung in ein bestehendes Web-Programm**
- **Entwicklung** eines Werkzeugs zur automatisierten Produktionsplanung auf Basis von mathematischen und informatischen Modellen
- **Verifikation und Validierung** der Konzeption mittels Tests und Simulationen

Studiengang: B.Sc. Informatik
Betreuer: Yübo Wang, M.Sc. M.A.

Fachgebiet
Datenverarbeitung in der
Konstruktion

Department of Computer
Integrated Design



Prof. Dr.-Ing. Reiner Anderl

Otto-Berndt-Straße 2
64287 Darmstadt

Tel. +49 6151 16-21791
Fax +49 6151 16-21793

Kurzfassung

Seit nunmehr einigen Jahren ist ein zunehmendes Aufkommen von allgegenwärtigen Verknüpfungen mit digitalen Objekten unter dem Namen „Internet of Things“ (IoT; auch bekannt als Internet of Objects) zu bemerken. Historisch waren Industrial Automation and Control Systems (IACS, dt.: Industrielle Automations- und Kontrollsysteme) jedoch größtenteils von einer solchen Verknüpfung ausgenommen. Um diese Lücke zu schließen, ist in jüngerer Vergangenheit das Konzept des Industrial Internet of Things (IIoT) aufgekommen. Dessen Ziel ist es, eine industrielle Infrastruktur über ähnliche Methoden wie beim IoT anzubinden.

Diese Thesis beschäftigt sich mit dem daraus hervorgehenden Potenzial der endbenutzergesteuerten Prozesssteuerung, -analyse, und -planung, sowie mit den Herausforderungen, die dadurch entstehen. Dabei wird untersucht, inwiefern diese Aspekte vereinfacht und individualisiert werden können. Des Weiteren werden webbasierte und interoperable Software-Komponenten entwickelt, in eine Demonstrationsplattform eingebunden und analysiert.

Schlüsselwörter: Industrie 4.0, Industrial Internet of Things (IIoT), Prozesssteuerung, Produktionsplanung, REST, OPC-UA, Manufacturing Bill of Materials (MBOM)

Abstract

For some years now, one has noticed an increasing emergence of ubiquitous connections with digital objects under the name „Internet of Things“ (IoT; also known as Internet of Objects). Historically, Industrial Automation and Control Systems (IACS) were largely excluded from such a link. To close this gap, the concept of the Industrial Internet of Things (IIoT) has recently emerged. It aims to connect an industrial infrastructure using methods similar to those used for the IoT.

This thesis deals with the resulting potential of end-user-controlled process control, analysis and planning, as well as with the challenges that arise from this. It is examined to what extent these aspects can be simplified and individualized. Furthermore, web-based and interoperable software components are developed, integrated into a demonstration platform, and analyzed.

Keywords: Industrie 4.0, Industrial Internet of Things (IIoT), Process Control, Production Planning, REST, OPC-UA, Manufacturing Bill of Materials (MBOM)

Inhaltsverzeichnis

| | |
|--|-----------|
| 1. Einleitung | 1 |
| 1.1. Motivation | 1 |
| 1.2. Zielsetzung der Thesis | 3 |
| 1.3. Aufbau dieser Arbeit | 3 |
| 2. Aktueller Stand | 5 |
| 2.1. Internet of Things | 5 |
| 2.1.1. Verbreitung und Wachstum des Internet of Things | 7 |
| 2.1.2. Industrial Internet of Things | 8 |
| 2.2. Industrie 4.0 | 10 |
| 2.2.1. Kategorisierung von Industrie 4.0 | 12 |
| 2.2.2. Design-Prinzipien von Industrie 4.0 | 14 |
| 3. Grundlagen | 18 |
| 3.1. HTTP | 18 |
| 3.1.1. HTTPS | 19 |
| 3.1.2. URI/URL | 19 |
| 3.2. REST APIs | 21 |
| 3.2.1. REST API Kommunikation | 22 |
| 3.2.2. Eigenschaften von REST APIs | 22 |
| 3.3. JSON | 24 |
| 3.4. OPC-UA | 25 |
| 3.5. WebSockets | 30 |
| 3.6. UUID | 31 |

| | |
|--|------------|
| 4. Endbenutzergesteuerte Prozessplanung und -steuerung & automatisierte und anpassbare Produktionsplanung | 34 |
| 4.1. Auswertung der OPC Factory Plattform | 35 |
| 4.2. Prozessplanung | 37 |
| 4.3. Prozesssteuerung | 41 |
| 4.4. Automatisierte und anpassbare Produktionsplanung | 44 |
| 5. Implementierung, Verifikation und Validierung | 45 |
| 5.1. Implementierung | 45 |
| 5.1.1. Implementierung der Prozessplanung | 45 |
| 5.1.2. Implementierung der Prozesssteuerung | 50 |
| 5.1.3. Implementierung der Produktionsplanung | 53 |
| 5.2. Potenzialanalyse | 54 |
| 5.3. Verifikation und Validierung | 55 |
| 5.3.1. Einhaltung von Standards | 55 |
| 5.3.2. Gleichzeitige Ausführung mehrerer Prozesse | 55 |
| 5.3.3. Latenzen und Latenz-Varianzen | 56 |
| 5.3.4. Tests | 59 |
| 5.3.5. Simulation | 60 |
| 6. Ausblick und Zusammenfassung | 61 |
| 6.1. Ausblick | 61 |
| 6.2. Zusammenfassung der wichtigsten Ergebnisse | 62 |
| A. Anhang | vii |

Abbildungsverzeichnis

| | |
|---|----|
| 1.1. Festo MPS System 403-1 am DiK | 2 |
| 1.2. Inhalte, Schwerpunkte und Erkenntnisse der Kapitel dieser Arbeit | 4 |
| 2.1. IoT Heimnetzwerk mit Heimserver | 6 |
| 2.2. Wachstum weltweiter Ausgaben für IoT-Lösungen [Vailshery, 2021] | 8 |
| 2.3. Vergleich zwischen industriellem und Massenmarkt-IoT | 9 |
| 2.4. Weg von Industrie 1.0 zu Industrie 4.0 [Inray, 2021] | 10 |
| 2.5. Von der Automatisierungspyramide zur industriellen Transformation mit Industrie 4.0 [eigene Abbildung nach i-SOOP, 2022] | 12 |
| 3.1. HTTP Request Over TCP + TLS | 20 |
| 3.2. Extended URL Example | 21 |
| 3.3. REST API Kommunikation | 22 |
| 3.4. Definition eines Objektes in JSON [Internet Engineering Task Force, 2022] | 25 |
| 3.5. OPC-UA Netzwerk [inray Industriesoftware GmbH, 2022] | 26 |
| 3.6. Aufbau einer WebSocket Verbindung | 31 |
| 3.7. UUID Version 1 Komponenten | 32 |
| 4.1. Benutzeroberfläche der OPC Factory | 35 |
| 4.2. Kommunikations-Infrastruktur der OPC Factory [Garbe u. a., 2019] | 36 |
| 4.3. Desktop-Browser-Marktanteil weltweit [StatCounter, 2021] | 39 |
| 4.4. Übersichtsskizze der Kommunikations-Infrastruktur mit Zwischenschicht | 42 |
| 4.5. Kommunikations-Infrastruktur mit Prozessplanungs und -steuerungs Komponenten | 43 |
| 5.1. Googles Blockly Editor | 46 |
| 5.2. Implementierter Editor mit ausgeklappter Toolbar | 47 |
| 5.3. Beispiel eines Logik-Blocks | 48 |

| | |
|--|----|
| 5.4. Aufbau der Prozessplanungs-Seite | 49 |
| 5.5. Prozessplanungs-Übersichtsseite | 49 |
| 5.6. Ablaufplan einer Prozessplanung und -steuerung | 52 |
| 5.7. Benutzeroberfläche der Produktionsplanung | 53 |
| 5.8. Ereignisprotokoll und Warteschlange | 56 |
| 5.9. Potenziell problematischer Prozess | 57 |
| 5.10. Verbesserte Kommunikations-Infrastruktur | 59 |
| | |
| A.1. Ganzseitiger Screenshot der Prozessplanungs und -steuerungs Seite | ix |
| A.2. Ganzseitiger Screenshot der Übersichts-Seite | x |
| A.3. Ganzseitiger Screenshot der Produktionsplanungs-Seite | xi |

Tabellenverzeichnis

| | |
|--|----|
| 2.1. Regierungsausgaben für IoT-Endgeräte und -Kommunikationsinfrastruktur in Milliarden US-Dollar [Goasduff, 2021] | 7 |
| 2.2. Design-Prinzipien von Industrie 4.0 Komponenten [Hermann, Pentek und Otto, 2016] | 15 |
| 3.1. Vergleich zwischen OPC-UA und REST APIs | 28 |
| 3.2. HTTP Methoden und äquivalente OPC-UA Dienste [Schiekofer, Scholz und Weyrich, 2018] | 29 |
| 3.3. Vergleich der UUID Versionen [Ryan, 2021] | 33 |
| 4.1. Zu unterstützende Browser-Versionen | 39 |
| 5.1. Testreihen zur Messung der Latenz-Varianz über die Cloud | 58 |
| 5.2. Testreihen zur Messung der Latenz-Varianz über das lokale Netzwerk . . . | 58 |

Abkürzungen

| | |
|---------|---|
| API | Application Programming Interface |
| App | Application |
| AWS | Amazon Web Services |
| CPS | Cyber-Physische Systeme |
| ERP | Enterprise Resource Planning |
| HATEOAS | Hypermedia as the Engine of Application State |
| HMI | Human-Machine-Interface |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| IACS | Industrial Automation and Control Systems |
| IIC | Industrial Internet Consortium |
| IIoT | Industrial Internet of Things |
| IoS | Internet of Services |
| IoT | Internet of Things |
| IP | Internet Protocol |
| JSON | JavaScript Object Notation |
| JWT | JavaScript Web Tokens |
| KI | Künstliche Intelligenz |
| MaaS | Manufacturing-as-a-Service |
| MAC | Media Access Control |
| MBOM | Manufacturing bill of materials |
| MEMS | Microelectromechanical Systems |
| MQTT | Message Queuing Telemetry Transport |
| OPC | Open Platform Communications |
| OPC-UA | OPC Unified Architecture |
| PLC | Programmable Logic Controller |
| Pub/Sub | Publish/Subscribe |

| | |
|-------|--|
| REST | Representational State Transfer |
| RFID | Radio-frequency identification |
| SCADA | Supervisory Control and Data Acquisition |
| SPS | Speicherprogrammierbare Steuerung |
| TCP | Transmission Control Protocol |
| TLS | Transport Layer Security |
| TSN | Time Sensitive Network |
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator |
| UUID | Universally Unique Identifiers |
| WAMP | Web Application Messaging Protocol |
| WASM | WebAssembly |
| WWW | World Wide Web |
| XML | Extensible Markup Language |

1. Einleitung

Das Internet of Things ist mittlerweile ein gut etabliertes Konzept, welches nicht nur den Einzug in das tägliche Leben vieler Menschen bewältigt hat, sondern auch in Großprojekten wie etwa beim Speditions- und Logistikunternehmen UPS [Roby, 2020] oder dem Gerätehersteller John Deere [Cambridge Innovation Institute, 2020] Anwendung findet. Allerdings finden die Planung und Steuerung von Fertigungsprozessen bei konventionellen Fertigungsstraßen immer noch meist auf spezieller, proprietärer Hardware und vor Ort bei der Maschine statt. Um dies an moderne Gegebenheiten wie Heimarbeit oder Fernwartung anzupassen und um die Prozess-/Produktionsplanung und -steuerung zugänglicher zu machen, ist eine Modernisierung der Schnittstellen und Benutzeroberflächen nötig.

Das Institut Datenverarbeitung in der Konstruktion (DiK) beschäftigt sich unter anderem eingehend mit den Themen Industrie 4.0 und Industrial Internet of Things (IIoT) und hat eine Vielzahl von Kooperationen mit nationalen und internationalen Partnern. In der Forschungseinrichtung wird eine moderne und für die Industrie repräsentative Fertigungsstraße (abgebildet in Grafik 1.1) eingesetzt, um das Potenzial von Industrie 4.0 und IIoT Applikationen zu analysieren. Als didaktische Plattform dient hierfür die Webseite Digital Twin Academy (digitaltwinacademy.de).

1.1. Motivation

Einer der Hauptzwecke des Internets of Things (IoT) besteht darin, physische Gegenstände über das Internet zu verbinden. Ein Hauptmerkmal des Industrial Internet of Things (IIoT) ist dann im Weiterführenden, verschiedene industrielle Geräte auf breiter Ebene miteinander zu verknüpfen. Zusammen mit intelligenter Software ist damit die Entwicklung immer fortschrittlicherer technologischer Fertigungssysteme, die über Softwareprogramme aus der Ferne überwacht und gesteuert werden können, möglich. Die



Abbildung 1.1.: Festo MPS System 403-1 am DiK

Adaption solcher Technologien ist noch recht niedrig, was dadurch verdeutlicht wird, dass zum Stand von 2020 weniger als 30 % der Hersteller von einer umfassenden Adaption berichten [Wopata, 2020]. Ein Grund für diese langsame Umstellung liegt an der hohen Komplexität der eingesetzten Softwarekomponenten. Oft werden spezialisierte Einzellösungen genutzt, welche in der Regel bereits während des Planungs- und Aufbauprozesses der Anlage bedacht und integriert werden müssen. [Gonzalez, 2018]

Diese Arbeit beschäftigt sich mit interoperablen, modularen und benutzerfreundlichen Softwarelösungen, deren Ziel es ist, eine an Industrie 4.0 und IIoT angepasste Prozessplanung und -steuerung sowie Produktionsplanung zu ermöglichen. Dabei werden Applikationen in den genannten Bereichen konzeptuell implementiert und ihr Potenzial für die Industrie wird analysiert.

1.2. Zielsetzung der Thesis

Im Verlauf dieser Thesis soll aufgezeigt werden, welches Potenzial Applikationen zur Verknüpfung von Anlagen an das Industrial Internet of Things sowie zur Verbindung mit Endnutzern haben können. Hierfür soll in Kapitel 4.2 und 4.3 zunächst eine Applikation zur endbenutzergesteuerten Prozessplanung und -steuerung konzipiert werden. Als Zweites soll in Kapitel 4.4 eine konzeptionelle Benutzeroberfläche für eine automatisierte und anpassbare Produktionsplanungs-Applikation entworfen werden, bevor abschließend diese Applikationen in Kapitel 5 implementiert, auf ihr Potenzial analysiert, verifiziert und validiert werden sollen.

1.3. Aufbau dieser Arbeit

Um eine Übersicht über die Inhalte, Schwerpunkte und Erkenntnisse der folgenden Kapitel zu geben, sei in Grafik 1.2 ein Überblick über diese Arbeit abgebildet.

| Kapitel | Inhalte | Erkenntnisse |
|---|---|--|
| 1. Einleitung | <ul style="list-style-type: none"> Motivation Zielsetzung Aufbau | <ul style="list-style-type: none"> Nutzen von endbenutzergesteuerten und interoperablen Planungs-, Steuerungs- und Analysewerkzeugen |
| 2. Aktueller Stand | <ul style="list-style-type: none"> Einführung in IIoT Kategorisierung und Design Prinzipien von Industrie4.0 | <ul style="list-style-type: none"> Übersicht über IIoT (Industrial Internet of Things) Entwurf von Industrie 4.0 Applikationen Stärken von IIoT und Industrie 4.0 |
| 3. Grundlagen | <ul style="list-style-type: none"> Übersicht über grundlegende und verwendete Technologien | <ul style="list-style-type: none"> Näherer Einblick auf aktuell etablierte Technologien Erläuterung relevanter und verwendeter Technologien |
| 4. Prozessplanung und -steuerung & Produktionsplanung | <ul style="list-style-type: none"> Auswertung bisheriger Arbeit Konzeption der Applikationen | <ul style="list-style-type: none"> Bestimmung der Zielgruppe und Platform Kriterien, welche die Applikationen erfüllen müssen Nutzen einer Ablaufplanung als Teil der vorgestellten Infrastruktur |
| 5. Implementierung, Verifikation und Validierung | <ul style="list-style-type: none"> Implementierung Potenzialanalyse Betrachtung von Problemfällen Messungen | <ul style="list-style-type: none"> Reflexion auf die verwendeten Standards und Kriterien Identifizierung von Problemfällen und Vorschlag von Lösungen Analyse des Potenzials |
| 6. Ausblick | <ul style="list-style-type: none"> Reflexion Ausblick Zusammenfassung | <ul style="list-style-type: none"> Ausblick auf zukünftige Arbeiten und Forschungsansätze Überblick über die wesentlichen Inhalte und wichtigsten Ergebnisse |

Abbildung 1.2.: Inhalte, Schwerpunkte und Erkenntnisse der Kapitel dieser Arbeit

2. Aktueller Stand

Im Folgenden wird ein Überblick über die theoretischen Grundlagen und bisherigen Fortschritte der im Hauptteil aufgeführten Konzepte dargelegt. Viele der hier angeschnittenen Konzepte und Technologien werden in Kapitel 3 näher beschrieben, bevor sie dann in den darauffolgenden Kapiteln zur Anwendung kommen.

2.1. Internet of Things

Das Internet of Things (IoT) beschreibt im Generellen Geräte, die mit dem Internet verbunden sind; doch im engeren bezieht sich der Begriff vielmehr auf Geräte, welche zusätzlich dazu auch miteinander kommunizieren. Mit „Geräten“ sind in diesem Kontext sowohl simple Sensoren gemeint, als auch PCs, Smartphones oder tragbare Geräte wie Smartwatches.

Die Fähigkeiten und Möglichkeiten von IoT gehen allerdings über eine simple Verknüpfung hinaus. So ist es möglich, die Informationen, welche von dem Netzwerk an IoT Geräten gesammelt werden, zu analysieren und auszuwerten. Daraus können dann weiterführende Informationen gewonnen werden, oder diese Daten können Automatisierungssystemen als Eingabe gegeben werden.

Ein Beispiel für eine solche Verknüpfung und Automatisierung wäre ein Thermostat, welches die Temperatur anhand der Jahreszeit, der aktuellen Temperatur im Zimmer (gemessen über einen separaten Sensor) und der Präsenz einer Person (gemessen über die WLAN-Verbindung eines Smartphones) regelt. Die Stärke liegt hier also vorwiegend in der Verknüpfung verschiedener Datenquellen und Geräte.

Die Geräte in einem solchen Netzwerk kommunizieren oft über den Router des lokalen Netzes miteinander und gegebenenfalls übers Internet zu einem anderen Standort oder

in die Cloud. Viele IoT Geräte benötigen dafür jedoch einen zusätzlichen Gateway. Ein typisches IoT Heimnetzwerk ist in Grafik 2.1 abgebildet. In diesem konkreten Beispiel wird neben der Cloud ein Heimserver im lokalen Netzwerk benutzt, um einige Anwendungen lokal auszuführen. So können unter anderem die Sicherheitskameraaufzeichnungen vollständig lokal gespeichert und verarbeitet werden, während die anderen Geräte ihre Daten an die Cloud weitergeben und auch von außerhalb des Netzwerkes gesteuert werden können.

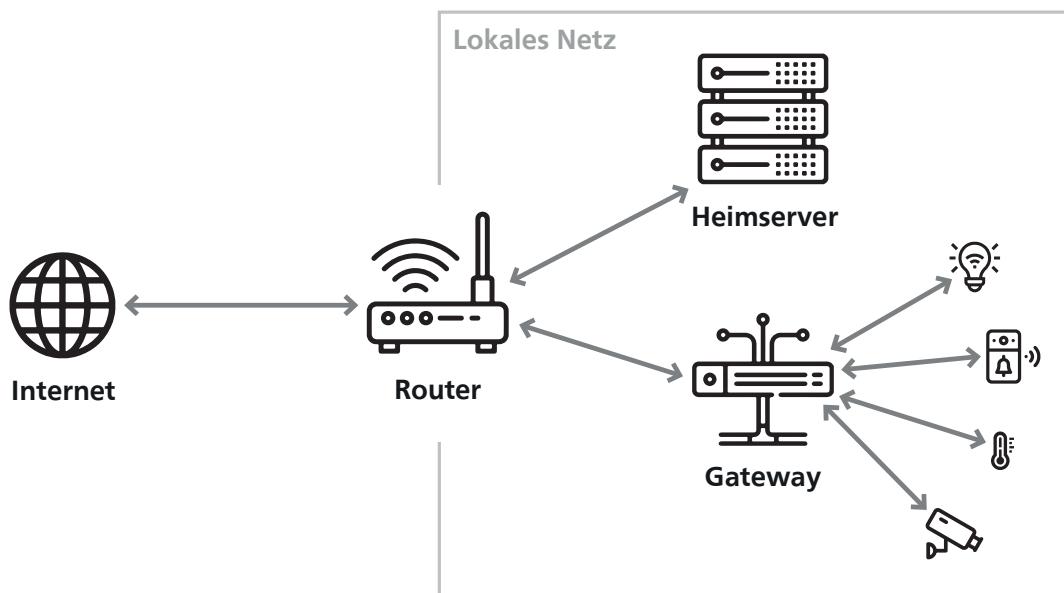


Abbildung 2.1.: IoT Heimnetzwerk mit Heimserver

Ermöglicht wurde diese durch IoT bedingte Trendwende neben der allgegenwärtigen Präsenz von Elektronik durch die weite Verbreitung von robusten, drahtlosen Netzwerken, sowie den immer weiter wachsenden Cloud-Infrastrukturen. Bei der Elektronik war eine wesentliche Neuerung die Verfügbarkeit von günstigen Allzweck-Prozessoren. Während zuvor noch oft spezialisiertere Hardware benutzt wurde, ist es mittlerweile vielmals günstiger, einen allgemeineren, massenproduzierten Chip zu verbauen [Moore, 2017]. Ein weiterer Faktor ist IPv6 (Internet Protocol Version 6), wodurch es möglich wurde, nicht nur dem gesamten Netzwerk eines Hauses oder einer Wohnung eine IP-Adresse zuzuordnen, sondern jedem einzelnen Gerät.

Um es mit den Worten von Rakesh Kumar, Associate Professor für Elektro- und Computertechnik an der University of Illinois zu sagen: „Prozessoren sind für die meisten Anwendungen überdimensioniert“ [Moore, 2017]. Im Falle von IoT ist dies aber von großem Vorteil, denn so ist es nicht nur für die Hersteller wesentlich einfacher und günstiger

geworden, nicht funktionsnotwendige Technologien wie eine Programmschnittstelle (z.B. eine REST API, siehe Kapitel 3.2) in ihre Geräte einzubetten, sondern es ist auch für die Nutzer einfacher geworden solche Funktionalitäten im Nachhinein hinzuzufügen. So hat sich eine wachsende Gemeinschaft aufgebaut, welche sich damit beschäftigt, Geräte „smart“ zu machen und sie an eine Verbindung an ein IoT Netzwerk vorzubereiten.

2.1.1. Verbreitung und Wachstum des Internet of Things

Um zu verdeutlichen, wie weitverbreitet, das Internet of Things und die dazugehörigen Geräte sind, zeigt Tabelle 2.1 die Regierungsausgaben für IoT-Endgeräte und -Kommunikationsinfrastruktur, geordnet nach Anwendungsgebiet und in Milliarden US-Dollar. Die Tabelle beinhaltet dabei neben den historischen Daten für 2020 auch eine Vorhersage für die Jahre 2021 und 2022. Diese Daten beziehen sich zwar nur auf Regierungsausgaben, doch sie geben auch einen Hinweis auf die allgemeine Akzeptanz und Verbreitung von IoT Technologien.

| Anwendungsgebiet | 2020 | 2021* | 2022* |
|------------------------------|-------------|-------------|-------------|
| Außenüberwachung | 9,3 | 9,7 | 12,0 |
| Maut- und Verkehrsmanagement | 1,8 | 2,1 | 2,6 |
| Verfolgung von Stadt-Anlagen | 1,5 | 1,9 | 2,2 |
| Spurensicherung der Polizei | 0,9 | 1,2 | 1,5 |
| Parkraumverwaltung | 0,5 | 0,6 | 0,7 |
| Feuerwehrüberwachung | 0,8 | 1,0 | 1,1 |
| Andere | 0,8 | 1,0 | 1,2 |
| Gesamtmarkt | 15,6 | 17,5 | 21,3 |

Tabelle 2.1.: Regierungsausgaben für IoT-Endgeräte und -Kommunikationsinfrastruktur in Milliarden US-Dollar [Goasdouf, 2021]

Wie in Tabelle 2.1 abzulesen ist, wird ein starkes Wachstum für die Regierungsausgaben für IoT prognostiziert. Um dieses Wachstum mehr in den Fokus zu rücken, werden im Diagramm 2.2 die weltweiten Endbenutzerausgaben für IoT-Lösungen für die Jahre 2017 bis 2025 abgebildet. Die mit * markierten Jahre symbolisieren hier Jahre, für die die

Daten auf Grundlage des historischen Trends abgeschätzt wurden. Wie in dieser Grafik zu sehen ist, wird ein enormes und an Geschwindigkeit zunehmendes Wachstum im IoT Bereich prognostiziert.

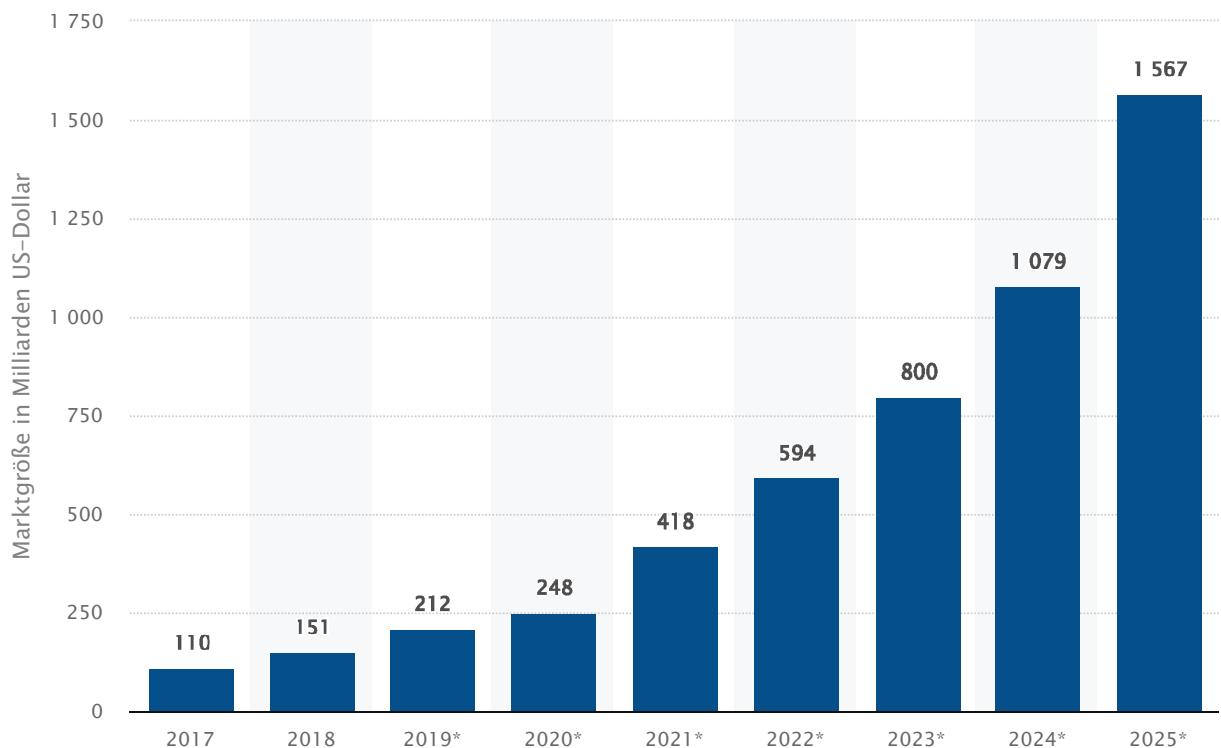


Abbildung 2.2.: Wachstum weltweiter Ausgaben für IoT-Lösungen [Vailshery, 2021]

2.1.2. Industrial Internet of Things

Ein weiterer Einsatzpunkt für IoT-Anwendungen ist in Fabrik- und Industrieanlagen. Hier werden Geräte mit Sensoren ausgestattet, wodurch Metriken wie Auslastung, Materialverbrauch, Durchsatz, etc. genauer und automatisiert ausgelesen und analysiert werden können. Auch kann der Maschinenverschleiß besser im Auge behalten werden, was zu einer proaktiven, statt einer reaktiven Wartung und einer längeren Lebensdauer der Geräte führt. Diese Eigenschaften führen zu einer zunehmenden Automatisierung von Fabriken [Kern und Anderl, 2019].

Da sich diese industriell orientierten Anwendungen in einigen Punkten von Anwendungen in anderen Gebieten unterscheidet, wird diese Unterkategorie des IoT auch als Industrial Internet of Things (IIoT) bezeichnet. In Grafik 2.3 sind einige dieser Unterschiede zwischen IIoT und Massenmarkt-IoT aufgezeigt.

2.2. Industrial Internet of Things

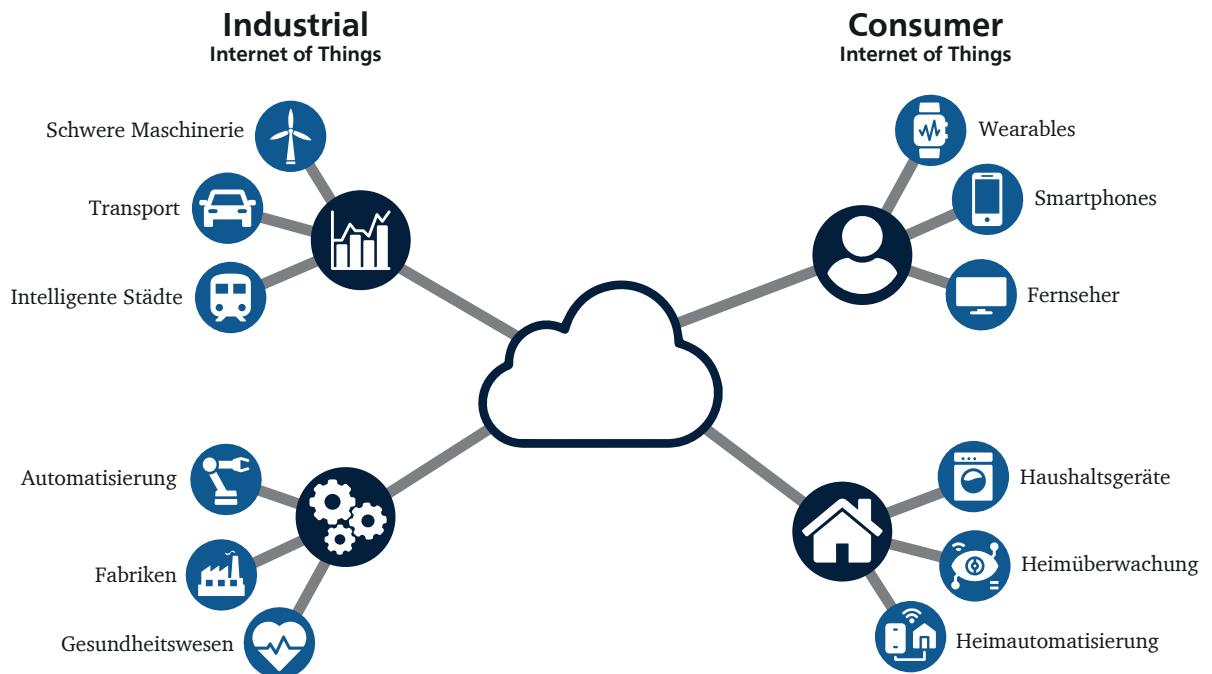


Abbildung 2.3.: Vergleich zwischen industriellem und Massenmarkt-IoT

IIoT verfügt über eine Referenzarchitektur vom Industrial Internet Consortium (IIC), welche in die folgenden drei Ebenen unterteilt ist [Bhattarai u. a., 2019]:

1. **Edge** – Die Edge-Schicht liegt direkt an der Anlage. Sie ist dafür verantwortlich, Daten von den Edge-Geräten zu sammeln und sie stromaufwärts zur Platform-Schicht zu übertragen.
2. **Platform** – In der Platform-Schicht sind Verwaltungsfunktionen sowie Datenanalyse und -auswertungen angesiedelt.
3. **Enterprise** – Die Enterprise-Schicht stellt eine Schnittstelle zum Endbenutzer dar und implementiert verschiedene Entscheidungsunterstützungssysteme und Anwendungen.

Neben dieser Referenzarchitektur wurde von verschiedenen Seiten weitere Architekturen vorgeschlagen, doch in dieser Arbeit wird der Fokus vorwiegend auf einem Aufbau liegen, welcher nah an der Referenzarchitektur liegt. Dies geht unter anderem darauf zurück, dass sich die drei Schichten gut auf die typische Softwarestruktur mit einem Backend, einer Middleware, und einem Frontend übertragen lässt.

Durch die Integration verschiedener Softwarealgorithmen, welche die Systeme autonom optimieren können, verspricht das IIoT-Paradigma, die Betriebseigenschaften technischer

Systeme zu verbessern. Zu diesen Eigenschaften zählen unter anderem Kostenreduktionen, Energieverbrauch, Durchsatz und Zeiteffizienz. [Ożadowicz u. a., 2018]

2.2. Industrie 4.0

Industrie 4.0, oft auch synonym als „Die vierte industrielle Revolution“ oder „Smart Industry“ bezeichnet, ist eine Vision und der Name des aktuellen Trends hin zu einer vermehrten Automatisierung und zunehmenden Datenaustauschs in der Fertigungstechnik [Anderl, 2015]. Der Ursprung von Industrie 4.0 geht auf eine Initiative der Bundesregierung und der deutschen Fertigungsindustrie zurück.

Die wesentlichen Charakteristiken der verschiedenen „Stufen“ von Industrie 1.0 bis Industrie 4.0 sind in Grafik 2.4 abgebildet. Dabei wird die Entwicklung von den Grundzügen der Automatisierung und Mechanisierung bis hin zur aktuellen vernetzten, selbst gesteuerten und selbst optimierenden Produktion dargestellt.

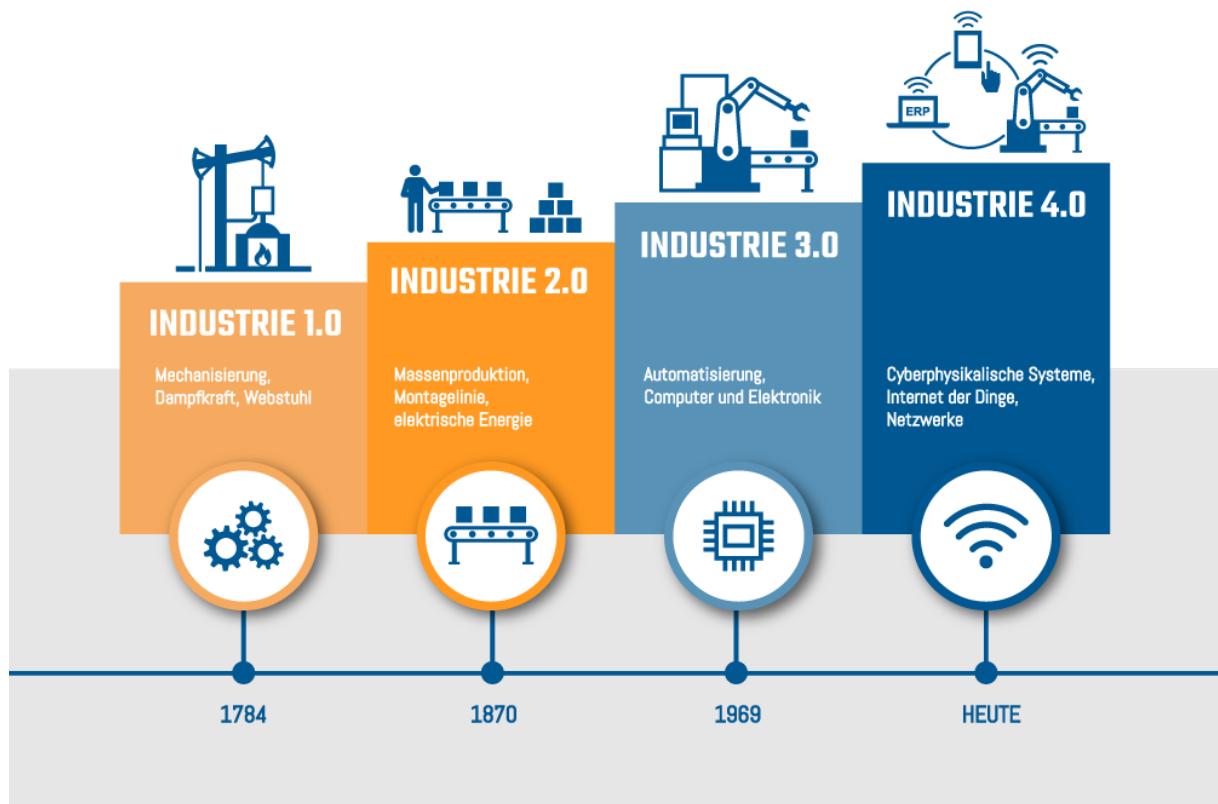


Abbildung 2.4.: Weg von Industrie 1.0 zu Industrie 4.0 [Inray, 2021]

Um weiter auf die konkreten Charakteristiken von Industrie 4.0 einzugehen und wie sie sich von dem vorhergehenden Status quo absetzt, seien hier einige dieser Punkte aufgeführt [i-Scoop, 2022]:

- Überbrückung der physischen und digitalen Welt durch Cyber-Physische Systeme (CPS), ermöglicht durch IIoT (siehe Kapitel 2.1.2)
- Mehr und weitreichendere Automatisierung als in der dritten industriellen Revolution
- Selbstgesteuerte Fabriken (Smart Factories)
- Closed-Loop-Regelsysteme und -Datenmodelle bei denen der Output wieder in das System zurückgegeben wird, um die zukünftige Funktion anzupassen
- Steuerung von Produktionsschritten durch intelligente Produkte und eine damit einhergehende Verlagerung weg von einem zentralen Steuerungssystem und hin zu einer verteilten und adaptiven Steuerung
- Benutzergesteuerte Personalisierung/Anpassung von Produkten und die damit verknüpfte und tiefgreifende Flexibilität in der Fertigung
- Wandelbare und anpassungsfähige Fabriken

Dabei sei ein besonderer Fokus auf die Relevanz von Industrial IoT (siehe Kapitel 2.1.2) gelegt, welches viele der anderen Punkte überhaupt erst ermöglicht, sowie auf die weitreichende Automatisierung und benutzergesteuerte Anpassung, welche beide eine vorherrschende Rolle in dieser Arbeit haben.

Doch natürlich ist IIoT nicht die einzige Grundlage, welche diese neuartigen Möglichkeiten begünstigt. Daneben sind auch Technologien wie Cloud-Computing und -Plattformen (wie Amazon Web Services (AWS), Microsoft Azure und Google Cloud), Big Data (Advanced Data Analytics, Data Lakes, Edge Intelligence) mit künstlicher Intelligenz (KI), Datenanalyse, Datenspeicherung und Rechenleistung an den Endpunkten von Netzwerken (Edge Computing), Mobile Endgeräte, Datenkommunikation und Netzwerktechnologien, Veränderungen unter anderem auf der Ebene von Mensch-Maschine-Schnittstellen (engl.: Human-Machine-Interface, kurz: HMI) und Supervisory Control and Data Acquisition (SCADA), Manufacturing Execution Systems (MES), Enterprise Resource Planning (ERP, wird i-ERP), speicherprogrammierbare Steuerungen (SPS), Sensoren und Aktoren, Microelectromechanical Systems (MEMS) und Transducer (dt.: Wandler, Umsetzer, Umrichter und Konverter) und innovative Datenaustauschmodelle eine wichtige Grundlage,

auf der Industrie 4.0 aufgebaut ist [i-SOOP, 2022]. Neben diesen Grundsteinen gibt es noch zahlreiche Weitere, von denen die für diese Arbeit besonders relevanten in Kapitel 3 näher beschrieben werden. Die meisten dieser Begriffe sollen hier jedoch nur der Vollständigkeit halber und zur grundlegenden Einordnung erwähnt werden.

Wie diese Technologien in eine Hierarchie einzuordnen sind und wie sie zusammenarbeiten, um zu einer industriellen Transformation zu führen, sei in Grafik 2.5 kurz angeschnitten. Wie dort zu sehen ist, decken die für Industrie 4.0 relevanten Neuerungen und Aspekte die gesamte Automationspyramide ab.

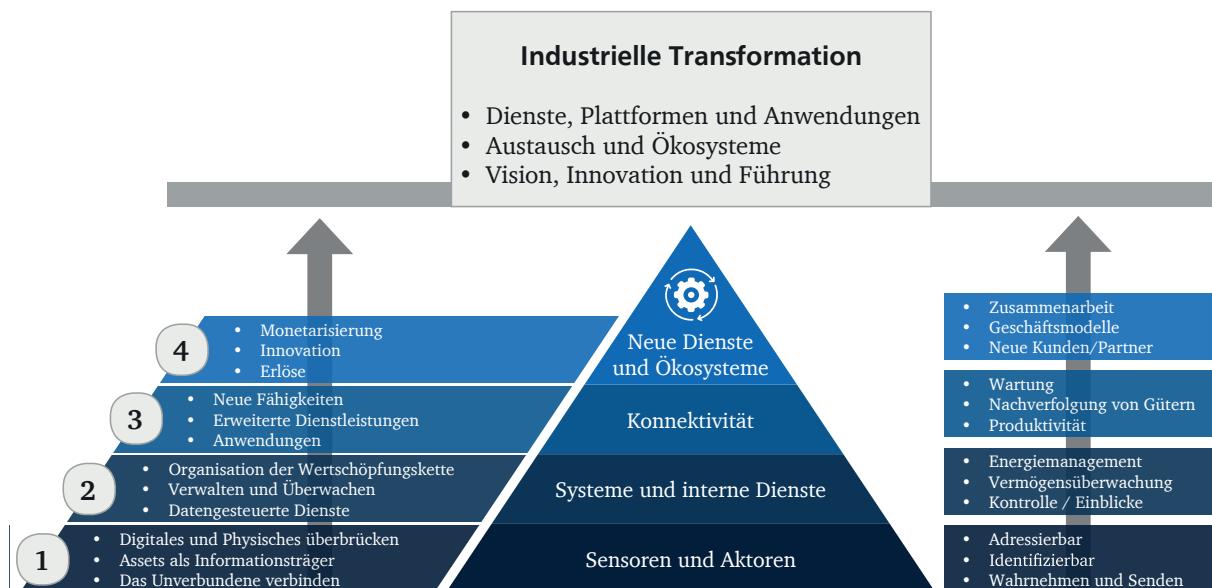


Abbildung 2.5.: Von der Automatisierungspyramide zur industriellen Transformation mit Industrie 4.0 [eigene Abbildung nach i-SOOP, 2022]

2.2.1. Kategorisierung von Industrie 4.0

Diese Technologien lassen sich grob in neun Kategorien einteilen, welche alle neuerlich größere Fortschritte erlebt haben. Industrie 4.0 lässt sich daher als Konvergenz dieser neun Teilbereiche von digitalen Industrietechnologien bezeichnen [i-SOOP, 2022]. Diese Teilbereiche sind im Folgenden ausgeführt:

Big Data und Datenanalyse

- Unterstützung bei der Entscheidungsfindung in Echtzeit

-
-
- Maschinenausfälle vorhersagen und Ausfallzeiten reduzieren
 - Vollständige Auswertung der verfügbaren Daten
 - Optimierung von Prozessen

Cloud

- Auslagerung von Berechnungen
- Verlagerung von Infrastruktur nach Außen
- Rasante und flexible Skalierung
- Verwaltung großer Datenmengen in offenen Systemen

Virtuelle/Erweiterte Realität

- Geführte Wartung, Support und Logistik
- Risikofreie Mitarbeiterschulung
- Verbesserung der Kundenerfahrungen
- Realitätsnahe Simulation und digitaler Zwilling

Horizontale und vertikale Integration

- Datenaustausch zwischen allen Gebieten bis hin zum Endbenutzer und Kunden
- Unternehmensübergreifende Datenintegration auf Basis von Datenübertragungsstandards
- Voraussetzung für eine voll automatisierte Wertschöpfungskette

Cybersicherheit

- Größere Netze und horizontale Integration bieten mehr Angriffsfläche
- Betrieb in Netzwerken und offenen Systemen
- Hohe Vernetzung zwischen Maschinen, Systemen und Produkten

Simulation

- Simulation von Wertschöpfungsnetzwerken
- Optimierung auf Basis von Echtzeitdaten intelligenter Systeme

-
-
- Reduzierung der Verschwendungen von Zeit und Ressourcen und Erhöhung der Effizienz in der Fertigung

Industrielles Internet

- Verbesserte Sicherheit und Fehleranfälligkeit
- Multidirektionale Kommunikation zwischen vernetzten Objekten
- Verbesserte und intelligente Konnektivität zwischen Geräten, Maschinen und Produkten

Additive Fertigung

- 3D Druck
- Schnelle Prototypentwicklung
- Einfache und schnelle Herstellung von Einzel- und Ersatzteilen
- Dezentrale 3D-Druckanlagen zur Reduzierung von Transportwegen und Lagerbeständen
- Konsolidieren einer Baugruppe zu einem einzigen Teil
- Reduzierung der Kosten und Erhöhung der Geschwindigkeit kleiner Produktionsserien

Fortschrittliche Robotik

- Autonome Industrieroboter
- Kooperierende und kommunizierende Roboter
- Zahlreiche integrierte Sensoren
- Standardisierte Schnittstellen

2.2.2. Design-Prinzipien von Industrie 4.0

Um nun Software (oder auch Hardware) zu entwickeln, welche in diesen Kontext passt, ist es hilfreich, Prinzipien zu identifizieren, welche aus den Charakteristiken von Industrie 4.0 hervorgehen und welche dann als Richtlinien genutzt werden können [Hermann, Pentek und Otto, 2016]. Neben den oben genannten Teilbereichen lässt sich Industrie 4.0 auch in verschiedene Komponenten einteilen. Im Gegensatz zu den Teilbereichen orientieren diese sich eher an den zuvor genannten Charakteristiken von Industrie 4.0. Hinzu kommt hier das Internet of Services (IoS), welches das Prinzip abbildet, dass alles

als ein abgekapselter, standardisierter und interoperabler Service verpackt und über das Internet zugänglich gemacht wird.

Welche Design-Prinzipien sich von diesen Komponenten ableiten, wird in Tabelle 2.2 aufgezeigt und darauffolgend wird auf jede dieser Prinzipien eingegangen.

| | Cyber-Physische Systeme | Smart Factory | Internet of Things | Internet of Services |
|---------------------|-------------------------|---------------|--------------------|----------------------|
| Dezentralisierung | ✓ | ✓ | ✗ | ✗ |
| Virtualisierung | ✓ | ✓ | ✗ | ✗ |
| Interoperabilität | ✓ | ✓ | ✓ | ✓ |
| Serviceorientierung | ✗ | ✗ | ✗ | ✓ |
| Modularität | ✗ | ✗ | ✗ | ✓ |
| Echtzeitfähigkeit | ✗ | ✓ | ✗ | ✗ |

Tabelle 2.2.: Design-Prinzipien von Industrie 4.0 Komponenten [Hermann, Pentek und Otto, 2016]

Dezentralisierung

Eine zentralisierte Überwachung und Steuerung wird aufgrund zunehmend diversifizierter Produktangebote erschwert; ebenso über mehrere Fertigungsstandorte hinweg. Eine Dezentralisierung dieser Systeme kann mithilfe von eingebetteten Computern erreicht werden. Diese können dann eigenständig Entscheidungen treffen, sodass nur im Fehlerfall die Verantwortlichkeiten auf eine übergeordnete Ebene übertragen werden müssen [Marques u. a., 2017]. Um auch dezentral eine Verwaltung und Qualitätssicherung sicherzustellen, können beispielsweise Radio-frequency identification (RFID) Chips verwendet werden [Hermann, Pentek und Otto, 2016].

Virtualisierung

Durch die Kopplung von virtuellen Anlagen- und Simulationsmodellen an die Sensordaten physischer Systeme, kann eine Anlage über Virtualisierung überwacht werden. Es wird also eine virtuelle Nachbildung der realen Welt konstruiert, in welcher der Zustand

der physischen Anlagen gespiegelt wird. So kann nicht nur eine bessere Überwachung stattfinden, sondern auf dem digitalen Zwilling können auch Simulationen und Analysen vollzogen werden und der Benutzer wird im Umgang mit der Maschine unterstützt [Kugler, Christ und Anderl, 2017].

Interoperabilität

Ein zentraler Bestandteil von Industrie 4.0 ist Interoperabilität, denn bei einem hohen Grad an Vernetztheit ist es erforderlich, dass unterschiedliche Geräte von verschiedenen Herstellern alle miteinander kommunizieren können. Da eine einheitliche Kommunikation im Wesentlichen von verwendeten Standards abhängt, wurde von der *Deutschen Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE* schon 2013 die „Deutsche Normungsroadmap Industrie 4.0“ [Standardization Council Industrie 4.0, 2020] herausgegeben, welche Normen für das Themengebiet Industrie 4.0 definiert. In den vergangenen Jahren wurde dieses Dokument mehrfach überarbeitet und liegt nun in der vierten Version vor.

Serviceorientierung

Für die Serviceorientierung werden die einzelnen Funktionen einer Anlage oder Anlagenkomponente zu einzelnen „Services“ abgekapselt. So wird ein hohes Maß an Modularität gewährleistet, da die Nutzer dieser Services sich so besser entscheiden können, welche Funktionen genutzt werden und welche nicht [Reis und Gonçalves, 2018]. Aufgrund der isolierten Natur dieser Services ist es außerdem möglich, die Fähigkeiten der Anlage schnell an einen geänderten Bedarf anzupassen.

Modularität

Um sich flexibel an die Veränderung von Anforderungen anpassen zu können, ist ein modulares System nötig. Die einzelnen Module können dann verändert, erweitert, oder gänzlich ausgetauscht werden, um die Fähigkeiten des Gesamtsystems anzupassen. Dies ist sowohl wichtig, um sich kurzfristigen Veränderungen wie Kundenwünschen anzupassen, als auch um regelmäßige oder geplante Veränderungen zu erleichtern. Wichtige Bestandteile hierfür sind standardisierte Soft- und Hardwareschnittstellen sowie ein

automatisierter Findungs- und Kopplungs-Mechanismus. Im Idealfall sollte damit eine Modularität nach dem Plug&Play-Prinzip möglich sein [Gupta, 2019]. Die zuvor erwähnten Punkte der Interoperabilität und Serviceorientierung spielen eine zentrale Rolle bei der Umsetzung eines modularen Systems.

Echtzeitfähigkeit

Um Daten in Echtzeit zu sammeln und zu analysieren, sowie um kurzzeitig auf Probleme und Ausfälle zu reagieren, muss der Zustand der Maschinen durchgehend überwacht werden. Die Daten werden direkt an entsprechende Programme weitergegeben und verarbeitet.

3. Grundlagen

Die hier aufgeführten und beschriebenen Technologien bilden die Grundlage der in dieser Arbeit entworfenen Anwendungen. Sie werden daher in den folgenden (Unter-)Kapiteln referenziert und hier nur grob in ihren Anwendungskontext eingeordnet.

Nach einem Überblick über HTTP und HTTPS (Kapitel 3.1) und damit der fundamentalen und traditionellen Kommunikation im Internet folgt eine Zusammenfassung zu REST und REST APIs (Kapitel 3.2), der vorherrschenden Technologie für Programmschnittstellen im Internet, und JSON (Kapitel 3.3), dem prävalent Datenformat für REST APIs. Darauf folgt ein kurzer Einblick in OPC-UA (Kapitel 3.4) mit einem besonderen Fokus auf die Unterschiede zu REST, und ein Einblick in das Web Socket Protokoll (Kapitel 3.5), welches eine mächtige Erweiterung zu üblichen REST APIs darstellen kann. Abschließend wird noch auf UUIDs (Kapitel 3.6) eingegangen, welche in vielen Fällen als Identifikatoren benutzt werden.

Es sei auch erwähnt, dass in den folgenden Kapiteln die Begriffe „Internet“ und „World Wide Web“ (WWW, oder kurz „Web“) synonym verwendet werden.

3.1. HTTP

Das Hypertext Transfer Protocol (HTTP) ist ein Protokoll für die Kommunikation zwischen einem Web-Browser und einem Web-Server [Berners-Lee, 1996]. Dabei folgt HTTP dem üblichen Client-Server Modell: Der Web-Browser, hier der Client (dt: Benutzer), eröffnet also die Verbindung und stellt eine Anfrage, der Server empfängt diese, und sendet daraufhin eine Antwort an den wartenden Client. Da der Server in der Kommunikation via HTTP keine Daten zwischen zwei Anfragen speichert, gilt dieses Protokoll als zustandslos. HTTP bildet die Grundlage von jedem Datenaustausch über das Web [Mozilla Corporation, 2021].

Es werden jedoch verschiedene Standards dieses Protokolls benutzt. Zum Stand von November 2021 wird HTTP/2 zwar von etwa 97,59 % der Browser unterstützt (gewichtet nach Marktanteil) [Deveria, 2021a], aber nur von etwa 46,8 % der 10 Millionen meistbesuchten Webseiten [Q-Success, 2021a]. HTTP/3 ist die neuste aufkommende Version des Standards. Als wesentliche Veränderung zu den Vorgängern benutzt HTTP/3 das Netzwerkprotokoll QUIC statt dem Transmission Control Protocol (TCP) für die tieferliegende Kommunikation [Bishop, 2021]. Die dritte Version von HTTP wird zum Stand vom November 2021 von 73,51 % der Browser unterstützt [Deveria, 2021b] und von 23,4 % der meistbenutzten Webseiten [Q-Success, 2021b].

3.1.1. HTTPS

HTTPS ist eine Erweiterung von HTTP, wobei die gesendeten Daten mittels Transport Layer Security (TLS) beziehungsweise früher mittels Secure Sockets Layer (SSL) verschlüsselt werden [Rescorla, 2000]. Daraus folgt ein Schutz der Integrität und Vertraulichkeit der Daten zwischen dem Computer des Benutzers und dem angefragten Server. Die Telemetriedaten des Firefox Browsers zeigen an, dass im November 2021 etwa 82,62 % der von Nutzern aufgerufenen Webseiten über HTTPS geladen wurden [Internet Security Research Group, 2021].

In Grafik 3.1 wird der übliche Nachrichtenverkehr einer HTTPS Verbindung zwischen einem Client (insbesondere einem Web-Browser) und einem Server dargestellt. Dabei ist ersichtlich, dass die Kommunikation im Wesentlichen in drei Schritte geteilt ist:

- Schritt 1: TCP Verbindung aufbauen
- Schritt 2: TLS Verschlüsselung herstellen
- Schritt 3: Anfrage senden und Antwort erhalten

3.1.2. URI/URL

Ein Uniform Resource Identifier (URI) ist ein Bezeichner, welcher aus einer eindeutigen Reihenfolge von Zeichen besteht. Der Zweck eines URI ist es, eine Ressource eindeutig zu identifizieren [Berners-Lee, Roy T. Fielding und Masinter, 2005].

Eine Unterform des URI ist der Uniform Resource Locator (URL), welcher eine Ressource nicht nur eindeutig identifiziert, sondern auch als eine Adresse zu dieser Ressource dient.

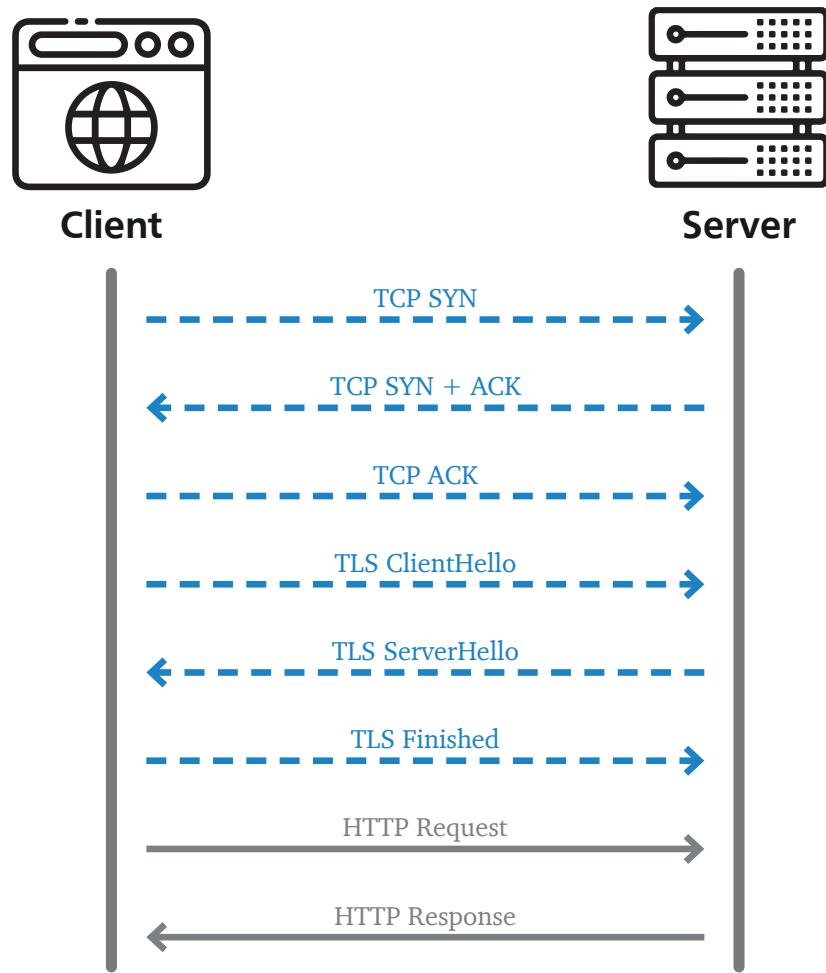


Abbildung 3.1.: HTTP Request Over TCP + TLS

Im Kontext des Internets werden in der Regel URLs von der Form `https://robin-garbe.de/a/uni/BA-Thesis.pdf` genutzt. Eine URL kann jedoch erweitert werden. Dies wird oft genutzt, um auf eine untergeordnete Ressource oder auf einen spezifischen Teil der Ressource direkt zu verweisen. Die Grafik 3.2 zeigt ein Beispiel für eine solche Erweiterung.

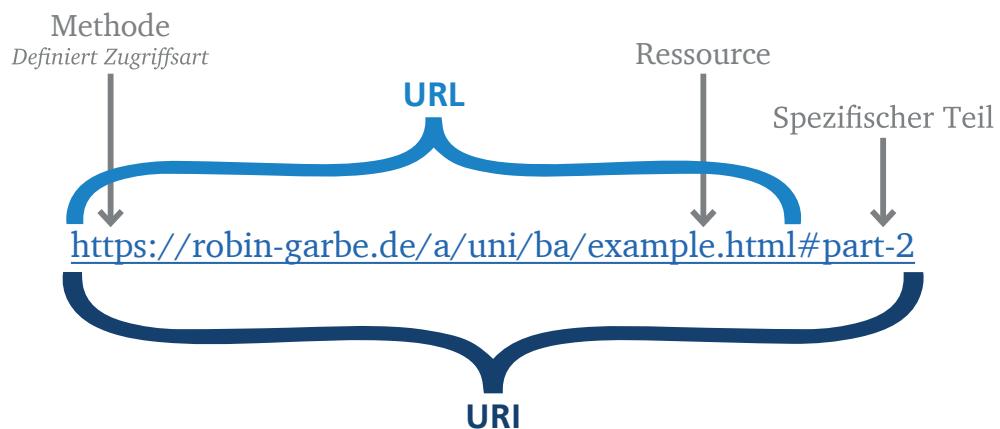


Abbildung 3.2.: Extended URL Example

3.2. REST APIs

Representational State Transfer (REST) ist der Name des Softwarearchitekturstils, der im Jahr 2000 von Roy Fielding in dessen Dissertation geprägt wurde [Roy Thomas Fielding, 2000]. REST benutzt dabei die zugrundeliegenden Protokolle und Technologien des World Wide Webs, um Richtlinien und Einschränkungen zu definieren, wie eine verteilte Softwarearchitektur sich verhalten sollte.

Ein häufiges Anwendungsgebiet für die von REST definierten Paradigmen sind Application Programming Interfaces (APIs). Eine API ist Software, welche es Anwendungen ermöglicht, über vordefinierte Methoden und Spezifikationen miteinander zu kommunizieren [Biehl, 2016]. Es dient dabei als die Schnittstelle zwischen verschiedenen Softwareprogrammen und erleichtert deren Interaktion.

3.2.1. REST API Kommunikation

Eine REST API (auch „RESTful API“ genannt) definiert eine Reihe von Funktionen, mit denen Entwickler Anfragen über HTTP-Protokolle wie GET, POST, PUT, PATCH, und DELETE ausführen und Antworten empfangen können [Richardson u. a., 2013]. Wie in Grafik 3.3 zu sehen ist, bildet die REST API dabei üblicherweise eine Schnittstelle zwischen einem Client (welcher auch ein Softwareprogramm sein kann) und einer Datenbank.

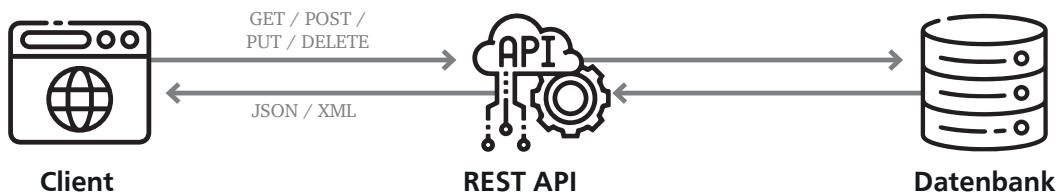


Abbildung 3.3.: REST API Kommunikation

3.2.2. Eigenschaften von REST APIs

Nicht jede API ist jedoch auch eine REST API. Dafür müssen sechs Kriterien und Eigenschaften erfüllt sein [Roy Thomas Fielding, 2000], welche sich vom REST Architekturstil ableiten. Es sollte jedoch erwähnt werden, dass die strenge Einhaltung dieser Attribute nicht immer gewünscht oder möglich ist. Genau genommen wäre eine solche API dann nicht mehr „truly RESTful“, doch in der Praxis hat dies wenig Relevanz und es geht vielmehr darum, diese Eigenschaften dort so gut wie angemessen möglich umzusetzen, wo es auch sinnvoll ist.

Einheitlichkeit

Die Schnittstelle einer REST API sollte so konsistent wie möglich sein. Es sollte dabei nicht zwei Endpunkte für eine Ressource geben und in der Ressource sollten (möglichst relative) Links zu zusammengehörigen, relevanten, oder weiterführenden Daten enthalten sein. Diese weiterführenden Links werden auch HATEOAS (Hypermedia as the Engine of Application State) genannt.

Alle Ressourcen sollten über konsistente Methoden zugänglich sein und in ähnlicher Weise

modifiziert werden können. Des Weiteren sollte das gesamte System sich an einheitliche Richtlinien bezüglich URL-Formaten, Namenskonventionen und Datenformaten halten.

Zustandslosigkeit

Ein „Zustand“ ist in diesem Kontext das Stadium, in dem der Client sich zu einem gegebenen Zeitpunkt befindet. Dieser Zustand kann dann relevant sein, wenn das Ergebnis einer Anfrage von dem aktuellen Zustand abhängt. Ein Beispiel hierfür wäre, dass in einer Suchmaschine auf der fünften Seite der Knopf „Nächste Seite“ geklickt wird.

Bei einer REST API ist der Client dafür verantwortlich den Zustand zu verwalten, und bei abhängigen Anfragen wie in dem genannten Beispiel, muss der Client den aktuellen Zustand in der Anfrage an den Server senden. Üblicherweise würde dann der Zustand in die URL der Anfrage eingebettet werden.

Zwischenspeicherbar

Im Kontext einer REST API bezieht sich das Zwischenspeichern (engl.: „cachen“) auf das Abspeichern der Serverantwort. Aufgrund der Zustandslosigkeit des Servers geschieht dieses Speichern auf der Seite des Clients, also in der Regel in dessen Web-Browser. Mit dieser Methode ist es möglich, dass ein Client eine Anfrage, zu der er eine Antwort bereits gespeichert hat, gegebenenfalls nicht noch einmal senden muss.

Die Antwort des Servers sollte hierfür Informationen darüber enthalten, ob und wie sie von einem Empfänger zwischengespeichert werden sollte.

Client-Server-Orientierung

Eine REST API sollte – wie auch der ihr zugrundeliegende HTTP Standard – dem Client-Server Modell folgen (siehe Kapitel 3.1). Der Client muss also die Kommunikation initiieren und der Server antwortet auf diese Anfrage.

Schichtsystem

Bei einer mehrschichtigen Systemarchitektur soll es möglich sein, dass eine Anfrage in mehreren Schritten von verschiedenen Servern bearbeitet werden kann. Beispielsweise

sollte ein Server die API-Schnittstelle bereitstellen, ein anderer eine Authentifizierung ermöglichen, und ein Dritter mit einer Datenbank verbunden sein. Für den Client sollte es dabei üblicherweise nicht ersichtlich sein, ob er direkt mit dem Endserver oder mit einem Zwischenhändler kommuniziert.

Code auf Anfrage

Dieser letzte Punkt ist weniger eine Anforderung, sondern eher ein besonderer Fall, welcher explizit erlaubt ist.

In der Regel sendet der Server seine Antwort an den Client als statische Datenrepräsentation in Form von JSON (JavaScript Object Notation) oder XML (Extensible Markup Language). Allerdings ist es auch möglich, ausführbaren Programmcode als Antwort zu senden. Dieser Code kann dann vom Client direkt ausgeführt werden. Dies kann in Situationen praktisch sein, in denen es vermieden werden soll oder es nicht möglich ist, zusätzliche Logik auf der Clientseite auszuführen, welche die Serverantwort parst, evaluiert, und dann ausführt.

3.3. JSON

JavaScript Object Notation (JSON) ist eine Syntax zum Definieren von Datenaustauschformaten. Diese ist textbasiert, simpel, und vor allem sprachunabhängig, was sie zu einem sehr verbreiteten Format für den Austausch strukturierter Daten im Internet macht. JSON wird von 99,71 % der Browser unterstützt (gewichtet nach Marktanteil) [Deveria, 2021c]. Ursprünglich wurde JSON aus der Syntax für Objektnotationen von der Programmiersprache ECMAScript (weithin bekannt als JavaScript) abgeleitet, doch der Standard für JSON [Bray, 2017] hat dessen Strukturierungsregeln abstrahiert, sodass nun die meisten Sprachen in der Lage sind, JSON zu parsen.

Der Satz dieser Regeln besteht lediglich aus den rekursiven Definitionen von `object`, `array`, `value`, `string`, `number`, und `whitespace`. In Grafik 3.4 ist beispielhaft die Definition eines Objektes dargestellt.

Ein wesentlicher Nachteil von JSON als Format für die Datenübertragung im Internet ist jedoch, dass JSON aufgrund seiner Struktur nicht streambar ist. Dies ist vorergründig im Kontext des Internets ein Problem, denn auch mit modernen Upload-

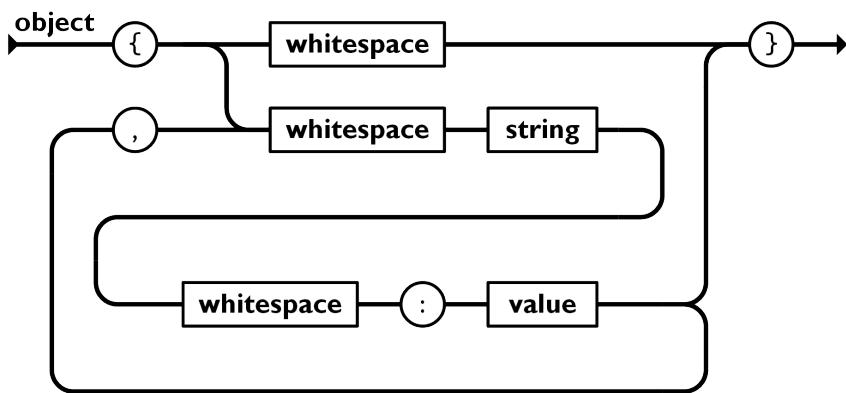


Abbildung 3.4.: Definition eines Objektes in JSON [Internet Engineering Task Force, 2022]

/Downloadgeschwindigkeiten liegt der rechentechnische Engpass bei der Datenübertragung. Andere Datenformate wie XML können gestreamt werden, das heißt, sie können geparsst und verarbeitet werden, noch während sie übertragen werden. JSON dagegen muss vollständig übertragen werden, bevor es von einem Programm verarbeitet werden kann. Dies ist jedoch in der Regel nur bei Echtzeitsystemen oder sehr großen übertragenen Datenmengen relevant.

3.4. OPC-UA

Eine Kommunikation mittels HTTPS und REST, mit Daten in Form von JSON oder XML funktioniert gut für eine Umgebung wie das Internet und mit Endgeräten wie Web-Browsern, sowie Web-Servers auf der Serverseite. Doch während diese Art der Kommunikation der de-facto Standard für das Internet ist, hat sie sich nicht für die Kommunikation für Technologien im Umfeld von Industrie 4.0 etabliert. In diesem Umfeld ist Interoperabilität zwischen Hardware, Software und Diensten verschiedener Hersteller nicht nur außerordentlich wichtig, sondern bisherige Technologien wie Web-Browser oder Web-Server sind auch unnötig komplex.

Open Platform Communications (OPC), beziehungsweise spezifischer OPC Unified Architecture (OPC-UA), ist daher ein offener Standard, welcher speziell für solche Anwendungen entworfen wurde und von verschiedenen Branchenanbietern weiterentwickelt und erweitert wird [TC 65/SC 65E, 2020]. Neben der Offenheit und der kollaborativen Natur des Standards bietet OPC-UA auch noch andere Vorteile, wie ein erweiterbares, umfangreiches Datenmodell. Dieses Datenmodell lässt sich leicht in neue Technologien

und Methoden integrieren, während es gleichzeitig mit älteren Produkten kompatibel bleibt. Ein weiterer zentraler Punkt und ein wichtiges Anliegen von OPC-UA ist Sicherheit, welche tief in den Standard integriert ist. Neben offensichtlichen Eigenschaften wie Verschlüsselung sind daher auch Aspekte wie Authentifizierung, Autorisierung, und Zugriffskontrollen eingebettet.

OPC-UA ist dabei ein neuerer Standard aus der OPC Familie. Wie der Name andeutet, ist der wesentliche Fokus von OPC-UA, eine einheitliche Architektur zu ermöglichen, welche eine größtmögliche Kompatibilität besitzt. In dieser Arbeit liegt der Fokus daher vollkommen auf OPC-UA, weshalb OPC und OPG-UA im folgenden als austauschbar zu verstehen sind.

In der OPC Architektur gibt es – wie auch im herkömmlichen Internet üblich – immer Server und Clients. Der OPC-Server ist dabei ein Softwareprogramm, welches eine Schnittstelle zur verwendeten Hardware bildet. Es wird also meist ein von einer Speicherprogrammierbaren Steuerung (SPS, engl.: Programmable Logic Controller, PLC) verwendetes Hardware-Kommunikationsprotokoll in das OPC-Protokoll umgewandelt. In der Regel hat jede Steuerungseinheit einen OPC-Server, und üblicherweise ist dieser auch direkt Teil der SPS. Grafik 3.5 bildet ein solches OPC-UA Netzwerk ab.

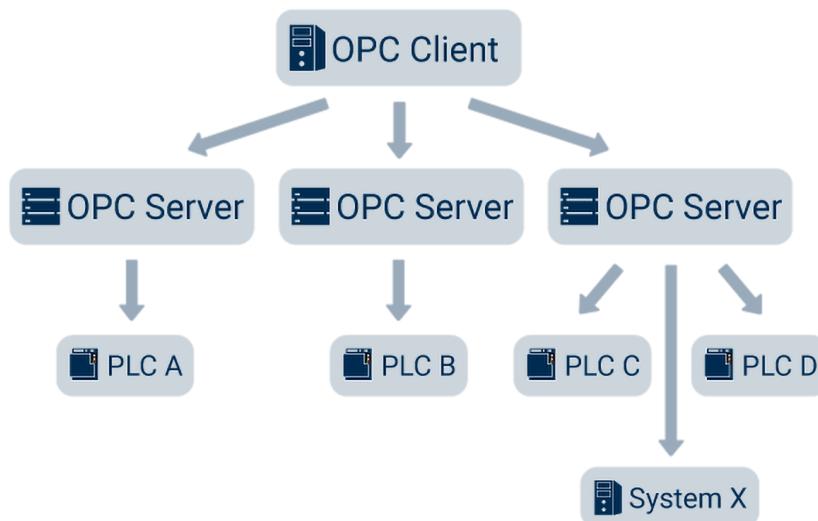


Abbildung 3.5.: OPC-UA Netzwerk [inray Industriesoftware GmbH, 2022]

OPC-UA Datenmodelle basieren auf Graphen, welche wiederum aus Knoten und Referenzen bestehen. Es gibt acht standardisierte Knotenklassen, welche die Attribute sowie die Semantik des Knotens definieren. Innerhalb eines Graphen können dabei sowohl die struk-

turellen Aspekte als auch die dynamischen Zusammenhänge eines Systems dargestellt werden [Tantik und Anderl, 2019].

Vergleich zu REST APIs

Zunächst sei gesagt, dass OPC-UA und REST APIs nicht direkt verglichen werden können, da sie einen leicht unterschiedlichen Umfang haben. OPC-UA ist wesentlich spezieller als eine REST API, bei welcher es darum geht, wie Daten im weiteren Sinne ausgetauscht werden. Bei OPC-UA dagegen sind Mechanismen wie Pub/Sub (Publish/Subscribe), Filterung, Ereignisse, Alarme, historische Daten, usw. schon eingebunden und müssen nicht separat implementiert werden. Es sollte bei einem Vergleich also berücksichtigt werden, dass solche Methoden – sofern benötigt – bei einem Softwaresystem, welches auf einer REST API basiert, noch auf anderem Wege hinzugefügt werden müssen.

Streng genommen ist OPC-UA nicht zustandslos, da zum Aufbau eines sicheren Kanals zur Kommunikation eine Art Zustand gespeichert wird, aber wenn über dieses Implementierungsdetail hinweggesehen wird, kann OPC-UA als praktisch zustandslos bezeichnet werden. Dazu sollte auch gesagt sein, dass eine REST API auf das gleiche Problem stoßen kann, wenn es darum geht, eine sichere Kommunikation zwischen dem Server und Client aufzubauen, allerdings ist dies dort nicht in den Standard eingebettet und kann daher auch anders gelöst werden.

Tabelle 3.1 vergleicht OPC-UA und REST APIs in einigen Punkten grob miteinander, um Gemeinsamkeiten und Unterschiede aufzuzeigen. Manche dieser Punkte sind dabei bewusst verallgemeinert, wie etwa die Tatsache, dass nicht absolut alle PLCs einen integrierten OPC-Server haben. Der Punkt „TCP-basiert“ verweist hier darauf, dass beide Arten der Kommunikation über TCP/IP funktionieren. Das bedeutet, dass zwar der Mehraufwand von TCP dazu kommt, aber das wird in den meisten Fällen von den Vorteilen aufgewogen, an erster Stelle der Absicherung vor verlorenen oder korrumptierten Datenpaketen. Bei einer REST API ist eine Kommunikation über User Datagram Protocol (UDP) daher nicht möglich, da die üblichen HTTP Protokolle (z.B. GET, POST, PATCH, etc.) schlichtweg über TCP funktionieren. Bei OPC-UA allerdings ist diese Einschränkung nicht gegeben, und es kann auch über UDP kommuniziert werden. Dies ist üblicherweise nicht der Standard, aber es kann zu Verbesserungen führen, wenn exemplarisch große Datenmengen gestreamt werden müssen und Datenverlust nicht allzu relevant ist. Zur Verschlüsselung sei gesagt, dass auf beide Arten eine Verschlüsselung möglich und zum aktuellen Stand

der Technik auch der Standard ist, aber es ist dennoch nicht ausgeschlossen, Daten auch unverschlüsselt zu übertragen (im Falle einer REST API etwa via HTTP statt über HTTPS). Bei beiden Kommunikationsarten findet die Verschlüsselung gleichwertig statt. Im Gegensatz zu ASCII-Dateien, die über RESTful-Architekturen gesendet werden, bietet OPC-UA ein umfassenderes Gerätedatenmodell und Datenwerte, die ihren ursprünglichen Datentyp, Genauigkeit und Korrektheit beibehalten [Rinaldi, 2019].

| | OPC-UA | REST API |
|----------------------------|--------|----------|
| Benutzbar über Web-Browser | ✗ | ✓ |
| Integriert in PLCs | ✓ | selten |
| Pub/Sub | ✓ | ✗ |
| TCP-basiert | (✓) | ✓ |
| Verschlüsselte Übertragung | (✓) | (✓) |
| Betriebssystemübergreifend | ✓ | ✓ |

Tabelle 3.1.: Vergleich zwischen OPC-UA und REST APIs

In Tabelle 3.2 werden die HTTP Methoden, welche von einer REST API benutzt werden, zu ihren äquivalenten OPC-UA Diensten und den dazugehörigen MIME-Typ eingeordnet. Die Tabelle umfasst dabei natürlich nur einige der abstrakten Dienste von OPC-UA, doch sie gibt eine Vorstellung darüber, wie Entsprechungen zwischen OPC-UA und einer REST API aussehen.

Abschließend lässt sich also zusammenfassen, dass OPC-UA eingebaute Sicherheitsfunktionen, flexiblere Transportmechanismen, Codierungen, Dienste und Datenmodellierungsfunktionen zur Verfügung stellt, während REST APIs zwar unkompliziert und einfach sind, dies aber auf Kosten der Funktionalität geht.

Es sollte sich also vor Augen geführt werden, dass beide Architekturen ihre Stärken in verschiedenen Richtungen haben und sie meist nicht in direkter Konkurrenz zueinander stehen. In vielen Fällen ist es sogar wünschenswert oder erforderlich, beides zu kombinieren. So ist es etwa möglich, eine über den Browser erreichbare REST API zu entwickeln, welche über eine Middleware mit einem OPC-Server direkt auf einer PLC kommuniziert.

| HTTP Methode | OPC-UA Dienste | MIME-Typ Repräsentation |
|--------------|-----------------|---|
| GET | Read | app/opcua.Boolean+json app/pdf ... |
| GET | HistoryRead | app/opcua.HistoryReadResult+json |
| PUT | Write | app/opcua.Boolean+json app/pdf ... |
| PATCH | HistoryUpdate | app/opcua.HistoryUpdateResult+json app/json-patch+json |
| GET | Browse | app/opcua.NodeRepresentation+json |
| GET | BrowseNext | app/opcua.NodeRepresentation+json |
| GET | TranslateBrowse | app/opcua.BrowsePathResult+json PathsToNodeIds |
| POST | Call | app/opcua.CallRequest+json app/opcua.CallResult+json |
| POST | AddNode | app/opcua.CallRequest+json app/opcua.CallResult+json |
| DELETE | DeleteNode | app/opcua.StatusCode+json |
| POST | Query | app/opcua.CallRequest+json app/opcua.CallResult+json |
| GET | QueryNext | app/opcua.QueryNextResponse+json |

Tabelle 3.2.: HTTP Methoden und äquivalente OPC-UA Dienste [Schiekofer, Scholz und Weyrich, 2018]

3.5. WebSockets

Traditionelle Client-Server Kommunikation folgt dem Prinzip von kurzlebigen Austauschen. Der Client initiiert den Dialog mit einer Anfrage, daraufhin folgen gegebenenfalls einige Nachrichten, um eine Verschlüsselung oder Ähnliches aufzubauen, und abschließend antwortet der Server. Damit endet in der Regel der Austausch und wenn der Client weitere Informationen will, ist dafür eine weitere Anfrage nötig. Auch bei längerlebigen Kommunikationen wie beispielsweise beim Streamen von Daten läuft dieser Prozess ähnlich ab. Dort hält der Server die Verbindung lediglich offen und sendet kontinuierlich neue Pakete an den Client. Was diese Art von Verbindungen jedoch nicht ermöglicht, ist ein fortlaufender Dialog zwischen beiden Seiten, während dem Nachrichten in beide Richtungen über eine bestehende Verbindung gesendet werden können.

Hier kommt das neuere Web Socket Protokoll ins Bild, welches eine Vollduplex Kommunikation spezifiziert. Eine große Stärke von WebSockets ist dabei, dass das dazugehörige Protokoll in allen größeren Browsern implementiert ist. Zum Stand von Dezember 2020 wird das Web Sockets Protokoll von etwa 98,36 % der Browser (gewichtet nach Marktanteil) unterstützt [Deveria, 2021d].

Grafik 3.6 zeigt, wie ein WebSocket aufgebaut wird. Zunächst wird vom Client der Handshake initiiert, worauf der Server nach erfolgreicher Authentifizierung mit dem Status Code **101 Switching Protocols** und einer neuen Adresse antwortet. Der Client benutzt dann diese neue Adresse für die bidirektionale Kommunikation, bis eine von beiden Parteien den Kanal schließt.

WebSockets sind eine dünne Transportschicht, welche auf TCP/IP aufbaut [Melnikov und Fette, 2011]. Es wird dabei allerdings nur definiert, wie der Nachrichtenaustausch stattfinden soll, nicht jedoch, wie die eigentlichen Daten innerhalb der Nachrichten strukturiert sind. Dafür werden sogenannte Unterprotokolle genutzt, welche definieren, wie die übertragenen Daten interpretiert werden sollen. Beispiele für Unterprotokolle sind JSON, XML, MQTT (Message Queuing Telemetry Transport) und WAMP (Web Application Messaging Protocol).

Nicht definiert wird eine Methode, mit der der Server den Client während des WebSocket-Handshakes authentifizieren kann. Dafür kann jeder clientseitiger Authentifizierungsmechanismus verwendet werden, der einem HTTP-Server zur Verfügung steht, wie etwa HTTP-Basic-Authentifizierung, JavaScript Web Tokens (JWT), oder Cookies.

Da WebSockets wie erwähnt direkt auf TCP/IP aufbauen statt auf HTTP, haben die Adres-

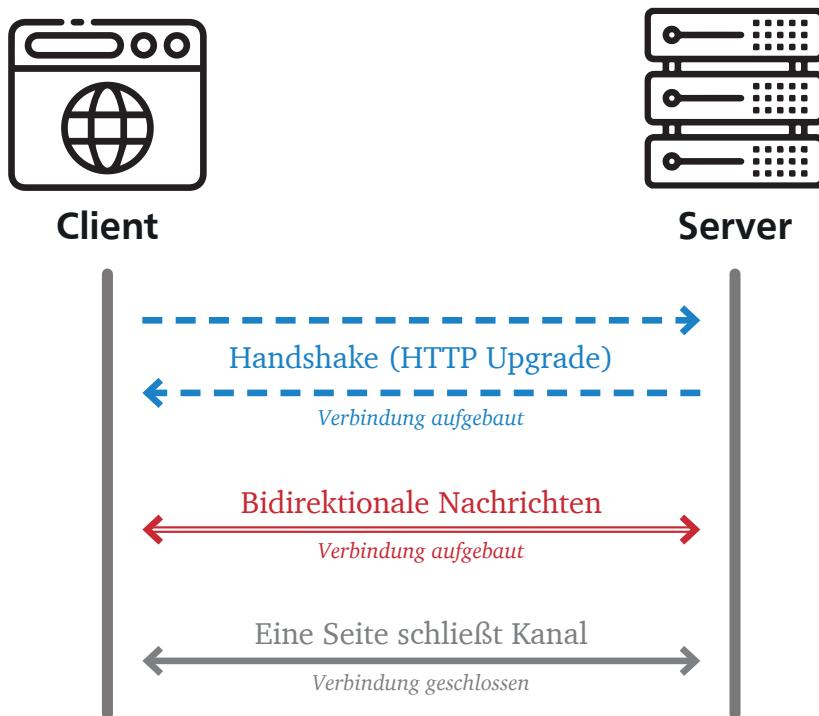


Abbildung 3.6.: Aufbau einer WebSocket Verbindung

sen für die bidirektionale Kommunikation eine Form wie `ws://example.com/socket`.

3.6. UUID

Universally Unique Identifiers (UUIDs) sind 128 Bit lange Zeichenfolgen bestehend aus 32 Base-16 encodierten Zeichen. Diese Länge ist ausreichend, sodass zwei zufällig generierte UUIDs mit annähernder Sicherheit nicht identisch sind. Da von den 128 Bits einer UUID 122 Bits zufällig gewählt werden, kann die Wahrscheinlichkeit für eine Überschneidung zweier zufällig generierter UUIDs mittels der Abschätzung $p(n) \approx 1 - e^{-n^2/(2 \cdot 2^{122})}$ approximiert werden. Eine Überschneidung bei der Generierung von einer Milliarde UUIDs tritt also beispielsweise mit einer Wahrscheinlichkeit von weniger als $1 \cdot 10^{-17} \%$ auf. Diese Berechnung setzt allerdings voraus, dass die UUIDs völlig zufällig und mit ausreichender Entropie generiert wurden.

Es gibt aktuell fünf Versionen von UUIDs, wobei die wesentlichen Unterschiede darin bestehen, wie die UUID generiert wird und dadurch wiederum verschiedene Vor- und Nachteile haben. Grafik 3.7 zeigt als ein Beispiel eine UUID der Version 1 und kennzeichnet

dessen unterschiedliche Bestandteile mit den dazugehörigen Komponenten-Namen. Die Komponenten `time_low`, `time_mid` und `time_hi_and_version` sind dabei etwa vom aktuellen Zeitpunkt abhängig, während die letzte Komponente von der MAC-Adresse (Media Access Control Adresse) des Gerätes kommt [Leach, Salz und Mealling, 2005].

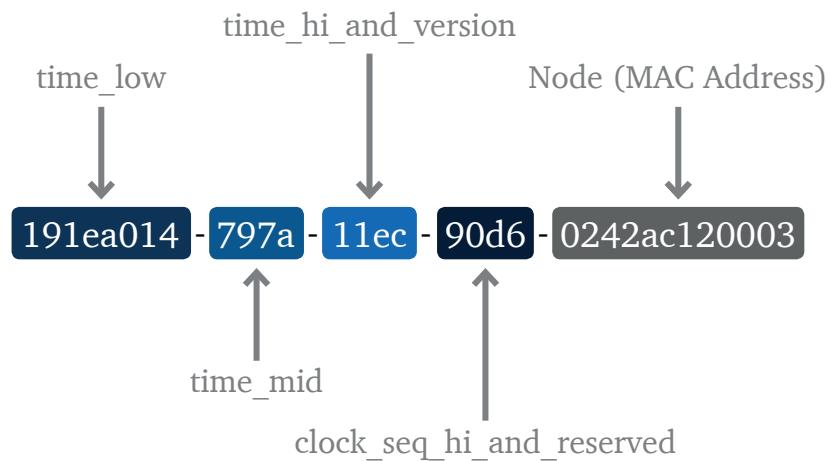


Abbildung 3.7.: UUID Version 1 Komponenten

Alle Versionen werden aktuell noch für unterschiedliche Anwendungsfälle genutzt. In Tabelle 3.3 werden die Attribute, Stärken und Schwächen der verschiedenen Versionen verglichen.

| Ver. | Attribute | Stärken | Schwächen |
|------|---|--|---|
| 1 | <ul style="list-style-type: none"> • Zeitstempel abgeleitet von der internen Uhr des Geräts • Node ist einzigartig für das Erzeugungsgerät | <ul style="list-style-type: none"> • Einzigartigkeit: Eine Überschneidung ist aufgrund von Geräte- und zeitbezogene Informationen praktisch unmöglich | <ul style="list-style-type: none"> • Anonymität: MAC-Adresse und Zeitstempel verraten Uhrzeit und Gerät der Generierung |
| 2 | <ul style="list-style-type: none"> • Zusätzliche Kennnummer wird Generierung einbezogen | <i>Identisch zu Version 1</i> | <i>Identisch zu Version 1</i> |
| 3 | <ul style="list-style-type: none"> • Verwendet einen Hashing-Algorithmus, um eine zufällige Zeichenfolge basierend auf dem Namespace zu generieren, in dem die UUID generiert wurde. • Keine Geräte- oder Zeitkomponenten | <ul style="list-style-type: none"> • Anonymität: Gehashte Namespace-Identifikatoren verbergen den Ursprung der UUID • Reproduzierbarkeit: Mit den gleichen Eingaben kann eine gleich UUID generiert werden | <ul style="list-style-type: none"> • Einzigartigkeit: Verlässt sich auf einen vom Benutzer eingegebenen Namen, um zufällige Komponente zu generieren • Reproduzierbarkeit: Mit den gleichen Eingaben kann eine gleich UUID generiert werden |
| 4 | <ul style="list-style-type: none"> • Zufällig generiert ohne zeit-, geräte- oder umfeldbezogene Komponenten | <ul style="list-style-type: none"> • Am zufälligsten • Keine Möglichkeit Herkunft zu bestimmen | <ul style="list-style-type: none"> • Keine Möglichkeit Herkunft zu bestimmen |
| 5 | <ul style="list-style-type: none"> • Ähnlich zu Version 3, jedoch mit einem sichereren Hashing-Algorithmus | <i>Identisch zu Version 3</i> | <i>Identisch zu Version 3</i> |

Tabelle 3.3.: Vergleich der UUID Versionen [Ryan, 2021]

4. Endbenutzergesteuerte Prozessplanung und -steuerung & automatisierte und anpassbare Produktionsplanung

Im Hinblick auf die Zielsetzung dieser Arbeit wird sich in diesem Kapitel der Entwicklung einer prototypischen Softwarelösung zur (end-)benutzergesteuerten Prozessplanung sowie -steuerung angenommen. Diese Aufgabe lässt sich in zwei offensichtliche Teilaufgaben aufteilen:

1. **Prozessplanung** – Ein benutzerfreundliches, simples und dennoch mächtiges Werkzeug zur feingranularen Planung von Fertigungsprozessen
2. **Prozesssteuerung** – Ein an die Prozessplanung angebundenes System zur Übermittlung und Steuerung der Maschinen einer verbundenen Fertigungsanlage

Die konkrete Software, welche hier entwickelt, betrachtet und analysiert wird, ist Teil der Digital Twin Platform des DiK. Genauer soll sie über die Digital Twin Academy Webseite unter dem Namen „Flexible MBOM Generation“ bereitgestellt werden. Bei dieser Webseite handelt es sich um eine didaktische Plattform, welche den Besuchern die Implementierung von IIoT näher bringt und ihnen ermöglicht, direkt und auf verschiedenen Ebenen mit Maschinen in verteilten Standorten zu interagieren.

Bevor im Folgenden dann auf die Konzeption und Entwicklung sowie das weitere Potenzial dieser Softwarelösung(en) eingegangen wird, wird zunächst ein Blick auf die „OPC Factory“ Platform geworfen. Kapitel 4.2 beschäftigt sich dann zunächst mit der Benutzeroberfläche und dem Interaktionsaspekt der Applikation, während das darauffolgende

Kapitel 4.3 sich näher mit der Kommunikations-Infrastruktur beschäftigt. Zuletzt wird in Kapitel 4.4 ein Konzept für eine automatisierte und anpassbare Produktionsplanungs-Applikation entworfen.

Die Implementierung dieser Anwendungen findet dann in Kapitel 5.1 statt.

4.1. Auswertung der OPC Factory Plattform

Während eines Informatik-Bachelorpraktikums „Mojito, Caipirinha, Sex on the Beach – Die Rezepte des BAR-Robots müssen auf eine Plattform“ wurde bereits 2019 die „OPC Factory“ Platform entwickelt, dessen Anwendungsfall dem in dieser Arbeit präsentierten nicht unähnlich ist [Garbe u. a., 2019]. In Grafik 4.1 ist die Benutzeroberfläche dieser Anwendung abgebildet.

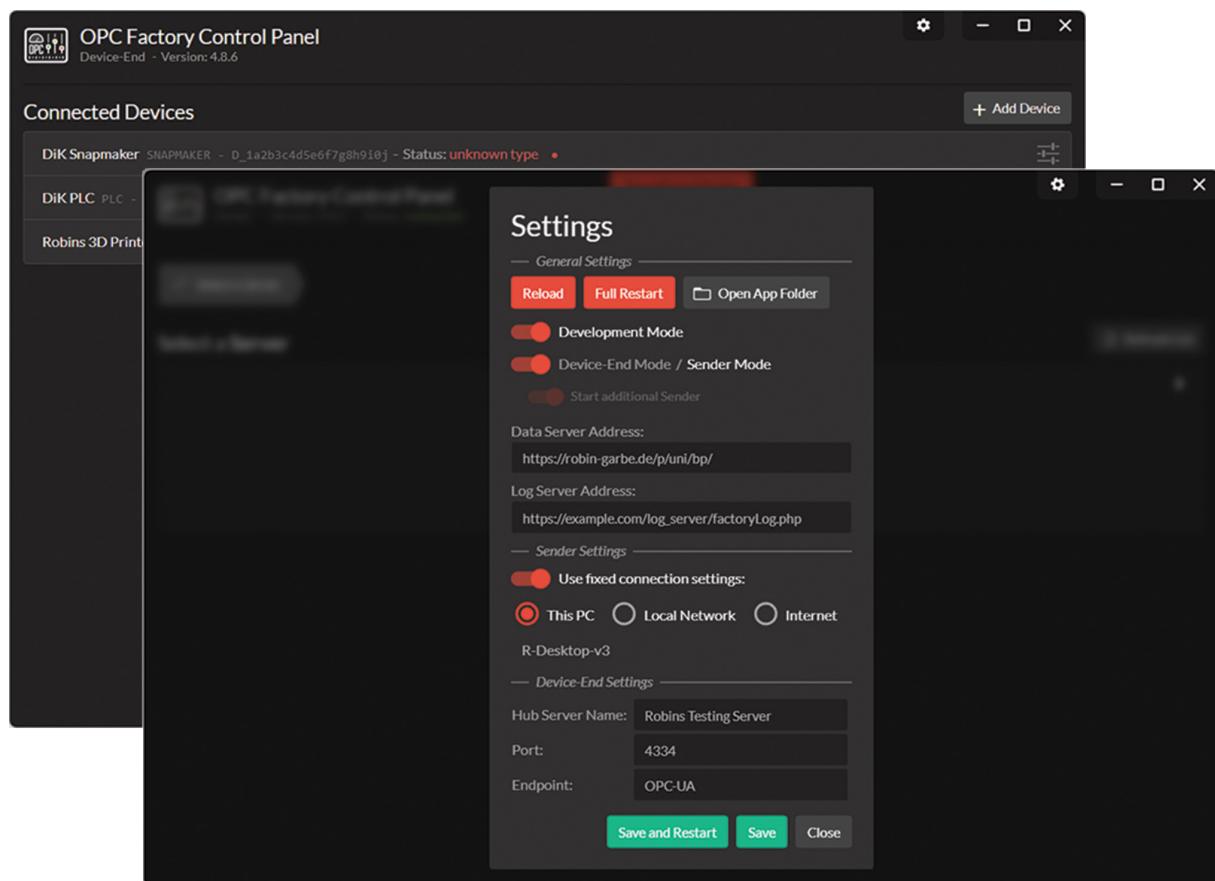


Abbildung 4.1.: Benutzeroberfläche der OPC Factory

Die OPC Factory ist zusammenfassend ausgedrückt eine Applikation, welche aus einem

Server („Device-End“) und einem Client („Sender“) besteht, und welche genutzt werden kann, um auf einer Vielzahl von verbundenen Maschinen vordefinierte Aufträge auszuführen. Das System war dabei serviceorientiert und flexibel gestaltet, sodass ein Server sich mit beliebig vielen Endgeräten verbinden kann und ein Client alle Endgeräten von beliebig vielen Servern ansprechen kann. Eine Besonderheit ist hier, dass die Verbindung zu den Endgeräten modular gestaltet ist. Das heißt, dass ursprünglich PLCs über OPC-UA sowie über USB verbundene Geräte mittels einer seriellen Verbindung angesprochen werden konnten, dies jedoch um beliebige weitere Gerätetypen erweitert werden kann. Die Kommunikations-Infrastruktur der OPC Factory ist in Grafik 4.2 abgebildet. Der Datenaustausch zwischen dem Client und dem Server läuft in dieser Applikation mittels OPC-UA.

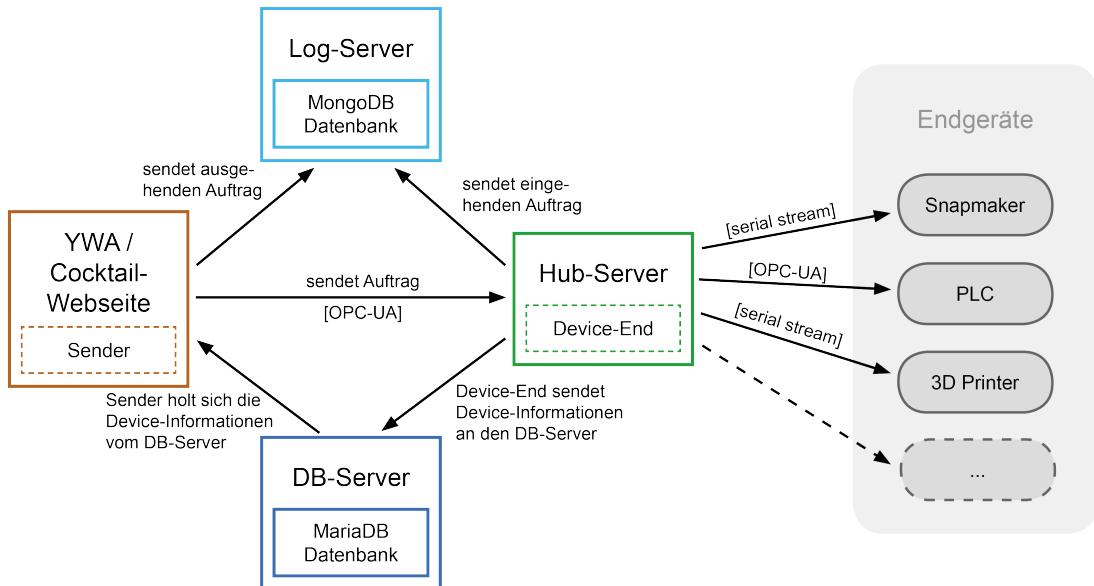


Abbildung 4.2.: Kommunikations-Infrastruktur der OPC Factory [Garbe u. a., 2019]

Einen Unterschied zu der in diesem Kapitel 4 behandelten Applikation ist, dass bei der OPC Factory nur vordefinierte Aufträge ausgewählt und abgeschickt werden können. Um dies so zu erweitern, dass auch individuelle Aufträge erstellt werden können, muss ein großer Fokus auf einem benutzerfreundlichen und dennoch mächtigen Interface liegen.

4.2. Prozessplanung

Wie beschrieben, ist der erste Schritt, eine Frontendapplikation zu entwickeln, welche einer Liste von Ansprüchen genügt. Hauptaspekt dieser Software soll die feingranulare Planung von Fertigungsprozessen sein.

Im folgenden Kapitel werden zunächst die erforderlichen Kriterien herausgearbeitet und diskutiert und auch mögliche Schwachstellen untersucht, während sich das dar-auffolgende Kapitel mit der Implementierung befasst. Kapitel 4.3 baut dann weiter auf den hier erarbeiteten Konzepten auf und erweitert diese. Die hier als „Prozessplanung“ beschriebene Applikation versteht sich also als ein reines Frontend-Programm.

Konzeption

Um die Ansprüche nun näher zu definieren, ist es zunächst erforderlich, die Zielgruppe dieser Applikation zu identifizieren. Trotz dieses konkreten Anwendungsfalls – als Teil der Digital Twin Platform – soll aber nicht der Aspekt der Potenzialanalyse dieser Arbeit vernachlässigt werden. Daher ist es nötig, dass die Zielgruppe nicht ausschließlich durch die Gruppe der Benutzer dieser Webseite eingegrenzt wird, sondern dass auch eine allgemeinere Zielgruppe in Betracht gezogen wird. Diese Gruppe reicht von unerfahrenen Laien, welche etwa die Digital Twin Academy Webseite nutzen, um erste Erfahrungen mit IIoT und der Planung von Prozessabläufen zu sammeln, bis hin zu erfahrenem Personal, welches das Prozessplanungs-Interface nutzt, um komplexe Prozesse an einer vertrauten Anlage zu planen. Letztere Teilgruppe hat evidenterweise bereits Werkzeuge für die Prozessplanung, doch diese sind in der Regel nicht einfach von der erstenen Teilgruppe nutzbar oder es fehlt an IIoT Fähigkeiten.

Das zu entwickelnde Programm sollte also für Laien einfach benutzbar sein, aber dennoch Experten nicht einschränken, sondern ihnen Funktionen zur Hand geben, um ihre Arbeit mit dem Programm zu erleichtern und zu beschleunigen. Außerdem sollte das Programm leicht zu lernen und intuitiv sein; es soll gewährleistet werden, dass Nutzer sich nicht erst in komplexe Dokumentationen einlesen müssen, sondern nach dem „learning by doing“ Prinzip agieren können.

Zusammenfassend lassen sich also folgende aus der Zielgruppe abgeleitete Kriterien für die angestrebte Softwarelösung festhalten:

-
-
1. **Simplizität** – Die Benutzeroberfläche soll leicht zu benutzen sein, auch für Benutzer, welche nicht oder kaum mit dem Anwendungsraum vertraut sind
 2. **Vielschichtigkeit** – Die Simplizität soll Expertennutzer nicht einschränken, sondern es soll Funktionalitäten geben, welche die Benutzung für diese Gruppe beschleunigt
 3. **Intuitivität** – Die Lernkurve soll möglichst flach sein, sodass ein Nutzer sich nicht erst einlesen muss

Als Nächstes ist es relevant, die Platform zu identifizieren. Auch unabhängig von der konkreten Implementierungsplatform (Digital Twin Platform) sollte ein allgemein zugängliches Programm möglichst einfach zu initialisieren sein. Bei Weitem am simpelsten ist hier offensichtlich ein Web-Interface, da damit – abgesehen von einem Web-Browser welcher ohnehin vorinstalliert ist – keine Software installiert werden muss. Um eine hohe Kompatibilität zu gewährleisten, muss sichergestellt werden, dass das Programm in einem möglichst hohen Anteil an Web-Browser funktioniert und von möglichst vielen potenziellen Nutzern benutzbar ist.

Als eine Richtlinie soll der „Europa Web Guide“ [Mauri und Bourrouet, 2021c] genutzt werden, welcher ein Regelbuch für die Webpräsenz der Europäischen Kommission darstellt. Da diese Richtlinien jedoch speziell für die Internetpräsenz der Europäischen Kommission angefertigt wurden, müssen einige Punkte für diesen Anwendungsfall abgeändert werden, wobei es dennoch eine nützliche Richtlinie bleibt.

Da keine konkreten Nutzeranalysedaten für die Webseite der Digital Twin Academy vorliegen, ist das Ziel also eine Abdeckung aller Desktop-Browser, welche einen Marktanteil von 3 % oder mehr haben. Wie in Grafik 4.3 zu sehen ist, betrifft dies mit Stand vom Dezember 2021 die fünf Browser Chrome, Safari, Edge, Firefox und Opera [StatCounter, 2021].

Die Browsersversionen betreffend sollten alle Versionen der letzten zwölf Monate unterstützt werden. Diese Zahl ist grob basierend auf der Annahme von neuen Browsersversionen, da in der Regel über 98 % der Nutzer eine Version benutzen, welche jünger als zwölf Monate alt ist [Babski, 2011].

Zum Stand vom Dezember 2021 ergeben sich also die in Tabelle 4.1 zu unterstützenden Versionen.

Aus dem Kriterium, dass der Browser als Platform dienen sollte, kann zudem abgeleitet werden, dass das Programm in JavaScript geschrieben werden soll; konkreter in

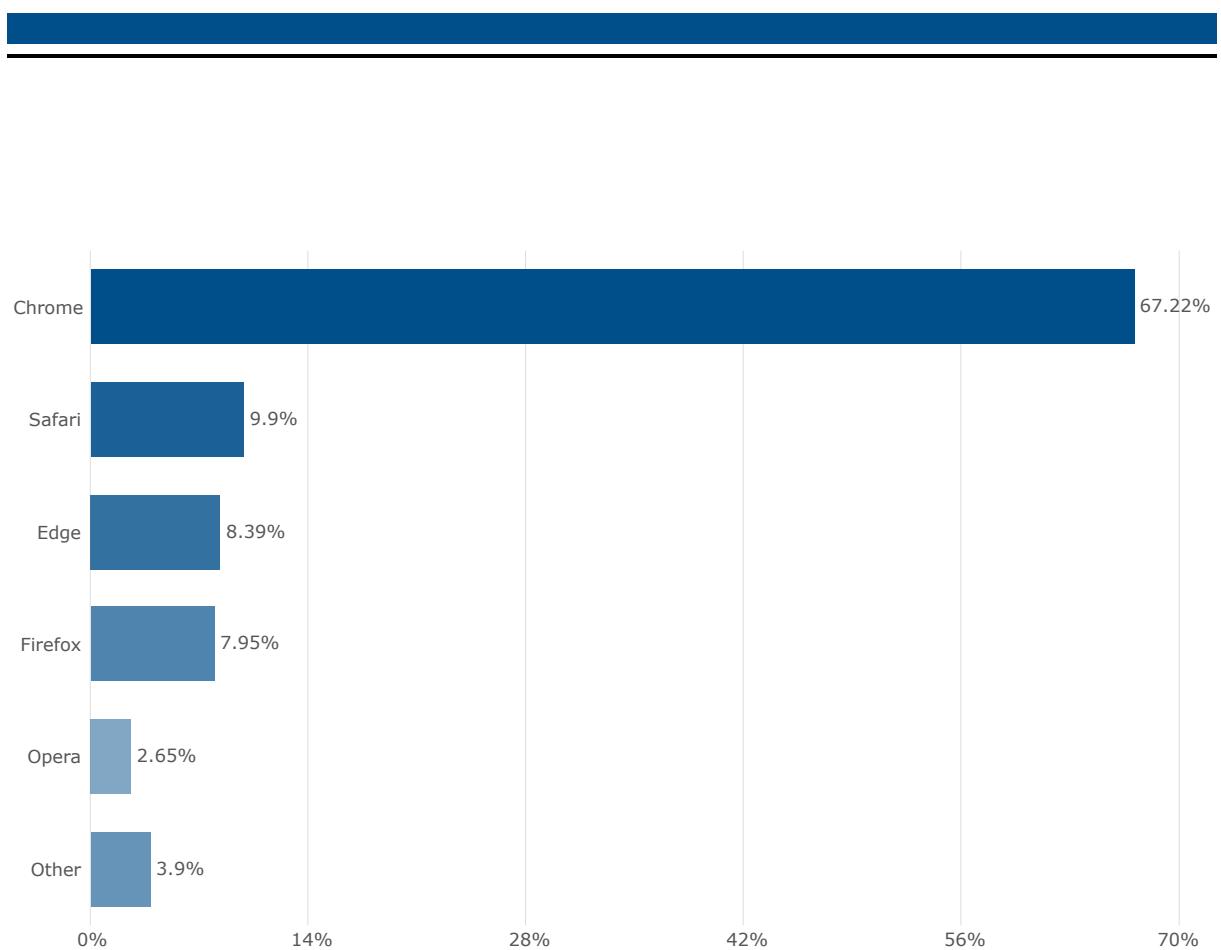


Abbildung 4.3.: Desktop-Browser-Marktanteil weltweit [StatCounter, 2021]

| Browser | Versionen |
|-----------------|-----------|
| Google Chrome | 88 – 97 |
| Apple Safari | 14 – 15 |
| Microsoft Edge | 88 – 96 |
| Mozilla Firefox | 78 – 95 |
| Opera | 74 – 76 |

Tabelle 4.1.: Zu unterstützende Browser-Versionen

ECMAScript 2020 (ES11). Über Kompatibilitätsschichten sind theoretisch auch andere Sprachen denkbar, allerdings ist ECMAScript 2020 die einzige (Frontend-)Programmiersprache, welche von den allen aktuellen Browsern unterstützt wird. Die einzige Ausnahme hierzu bildet das relativ neue WebAssembly (WASM), doch WASM wäre für eine solche Frontendapplikation exzessiv.

Aus den Erkenntnissen über die Platform lassen sich also folgende weitere Kriterien gewinnen:

4. **Zugängliche Platform** – Das Programm soll im Web-Browser laufen, sodass es nicht installiert werden muss und (praktisch) sofort benutzt werden kann
5. **JavaScript** – Als Programmiersprache soll ECMAScript 2020 genutzt werden
6. **Touchoptimiert** – Das Interface sollte auch auf größeren Touch-Geräten wie Tablets oder Laptops mit Touch-Funktion benutzbar sein

Außerdem lassen sich weitere Kriterien aus der Design-Prinzipien von Industrie 4.0 (siehe Kapitel 2.2.2) ziehen:

7. **Dezentralisierung** – Es sollen Prozessabläufe für Anlagen an beliebigen Standorten geplant werden können
8. **Virtualisierung** – Die geplanten Prozesse sollen virtualisierbar sein
9. **Interoperabilität** – Es soll vom Hersteller abstrahiert werden, sodass Anlagen von beliebigen Herstellern planbar sind
10. **Serviceorientierung** – Die Applikation soll als Service aufgebaut werden, um sie in Zukunft gegen einen anderen Service austauschen zu können
11. **Modularität** – Das Gesamtkonstrukt soll in Module eingegliedert werden, sodass es nach Belieben an veränderte Ansprüche angepasst werden kann
12. **Echtzeitfähigkeit** – Es sollen Prozesse geplant werden können, welche vom aktuellen Zustand der Anlage abhängig sind; außerdem sollen Latenzen und Latenzvarianzen (engl.: Jitter) in einem für Echtzeitsysteme akzeptablen Rahmen bleiben

4.3. Prozesssteuerung

Nachdem in Kapitel 4.2 eine Benutzeroberfläche zur Prozessplanung entwickelt wurde, muss nun der dahinterliegende Programmcode für die Weitergabe des geplanten Prozesses an die Maschinen und damit der Prozesssteuerung entworfen und implementiert werden. Es gilt nun zunächst, eine geeignete Kommunikationsstruktur auszuarbeiten, welche das Prozessplanungs-Interface an die Fertigungsanlagen anbindet und anschließend müssen die relevanten Teile dieser Infrastruktur implementiert werden.

Architektur und Konzeption

Die in Kapitel 4.2 genannten Kriterien gelten grundsätzlich auch hier, jedoch muss beachtet werden, dass Nutzer in der Regel nicht direkt mit der in diesem Kapitel behandelten Software interagieren. Einige der nutzerorientierten Kriterien entfallen daher. Da eine Übertragung von Daten hier eine zentrale Rolle spielt, ist die Datensicherheit von erheblicher Bedeutung. Konkret soll neben der Einhaltung europäischer Regelungen wie der Regulation 2018/1725 [European Parliament and Council, 2018] auch der industrielle Cyber-Sicherheitsstandard IEC 62443 [TC 65 - Industrial-process measurement, control and automation, 2009] befolgt werden. Selbstverständlich sollen auch Regulationen wie die europäische Regulation zur Verarbeitung personenbezogener Daten [European Parliament and Council, 2018] eng befolgt werden.

Um nun eine Infrastruktur zur Kommunikation und Steuerung von Fertigungsanlagen an verschiedenen Standorten über einen Web-Browser aufzubauen, sind einige Überlegungen zu beachten:

- Da die Kommunikation von einem Web-Browser – spezifischer von der in Kapitel 4.2 implementierten Applikation – ausgeht, muss JavaScript oder ein HTML Form genutzt werden.
 - Um die zuvor genannten Kriterien zu erfüllen – konkret die Kriterien der Serviceorientierung und der Modularität – sollte eine REST API genutzt werden, welche über ECMAScript 2020 angesprochen wird.
- Da es möglich sein soll, Anlagen direkt anzusteuern, ist es nötig, mit den PLCs dieser Anlagen zu kommunizieren. Einige PLCs haben einen integrierten Web-Server, welcher als eine REST API konfiguriert werden kann [Kinzig, 2018], jedoch ist dies

bei vielen älteren Modellen nicht der Fall und die Interoperabilität ist nicht ideal, da diese Konfiguration stark herstellerabhängig ist. Eine wesentlich bessere Lösung ist also die Kommunikation über OPC-UA, da dieser Standard von den meisten PLCs unterstützt wird und herstellerunabhängig ist. OPC-UA wird allerdings nicht von einem Web-Browser unterstützt.

- Es muss also sowohl über JavaScript als auch über OPC-UA kommuniziert werden, was es erforderlich macht, eine Zwischenschicht einzusetzen. Das heißt, dass es zwischen dem Browser des Nutzers und der Anlage vor Ort zwingendermaßen eine Zwischenschicht geben muss. Um die gesetzten Sicherheitsstandards einzuhalten, ist dies jedoch ohnehin vorteilhaft, da so die Nutzer nicht direkt mit der Anlage kommunizieren und potenzielle Angriffe abgehalten werden können.
 - Die Zwischenschicht muss sowohl eine REST API beinhalten, um mit dem Browser des Nutzers zu kommunizieren, als auch einen OPC-UA Client, um die Daten an die Anlage weiterzugeben.
- Um es zu ermöglichen, dass verschiedene Nutzer verschiedene Anlagen steuern können, muss es eine n-zu-1 Beziehung zwischen dem Nutzer und der Zwischenschicht sowie eine 1-zu-m Beziehung von der Zwischenschicht zu den Anlagen geben.

Aus diesen Überlegungen formt sich die in Grafik 4.4 skizzierte Infrastruktur-Übersicht.

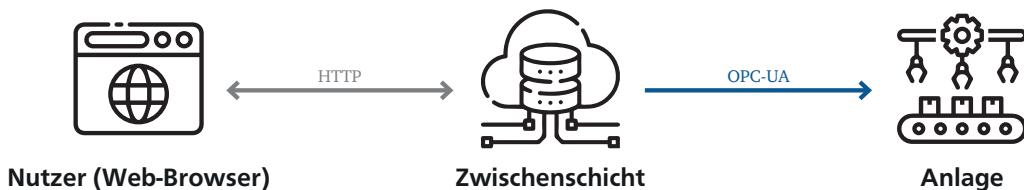


Abbildung 4.4.: Übersichtsskizze der Kommunikations-Infrastruktur mit Zwischenschicht

Diese Skizze ist selbstverständlich noch weit von einer vollständigen Kommunikations-Infrastruktur entfernt. Die Zwischenschicht sollte in die Cloud verlegt werden, um eine nahtlose Kommunikation von jedem beliebigen Ort zu jedweder Anlage an jedem Standort zu ermöglichen. Die „Zwischenschicht“ kann also als „Cloud-Schicht“ bezeichnet werden. Um eine Gesamtschau über die verfügbaren Anlagen und deren ansteuerbare PLCs und Aktoren zu haben und zu speichern, muss die Cloud-Schicht zudem eine Datenbank mit diesen Informationen beinhalten. Diese PLC- und Aktor-Informationen können dann über

die REST API von der Prozessplanungs-Software abgefragt werden, womit diese wie in Kapitel 4.2 beschrieben dynamisch Blöcke erzeugen kann.

Nun lässt sich also ein Blick auf die größere Kommunikationsstruktur werfen, um einzuordnen, wo die Prozessplanung und -steuerung in dieser Infrastruktur zu verorten sind. Grafik 4.5 bildet diese Infrastruktur detaillierter ab.

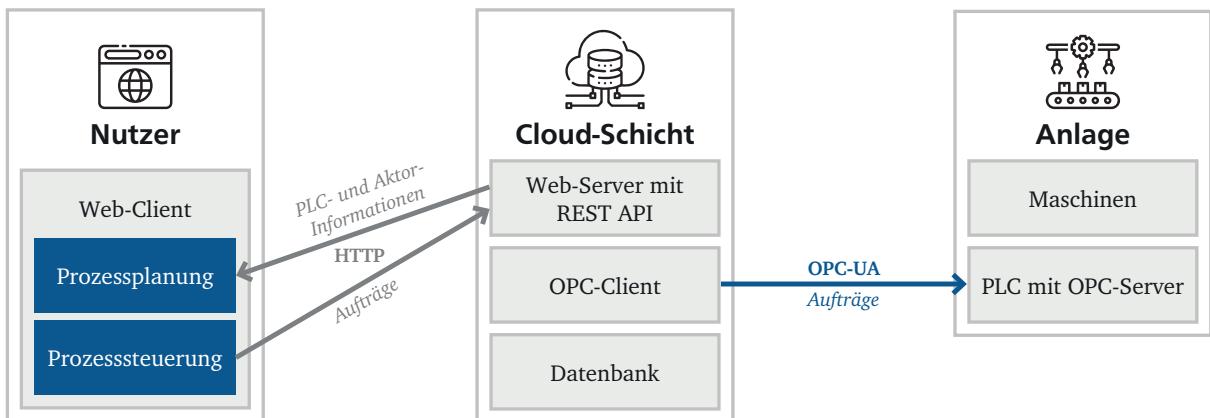


Abbildung 4.5.: Kommunikations-Infrastruktur mit Prozessplanungs und -steuerungs Komponenten

Zuletzt bleibt nur noch, eine Authentifizierung und Verschlüsselung hinzuzufügen. Letzteres ist mit wenig Aufwand getan, indem für die Übertragung zwischen dem Browser des Benutzers und der REST API über HTTPS stattfindet und damit wie in Kapitel 3.1.1 über TLS verschlüsselt ist; eine unverschlüsselte Übertragung über HTTP wird über die Web-Server Konfiguration verboten. Die Nachrichten, welche über OPC-UA von der Cloud-Schicht zur PLC gesendet werden, sind außerdem über die OPC-UA interne Verschlüsselung gesichert.

Die Authentifizierung dagegen wird über die API in der Cloud-Schicht gehandhabt. So muss jeder Nutzer erst einen gültigen API-Key von der REST API erhalten, und diesen dann bei jeder folgenden Nachricht mitsenden. Die API gibt einen solchen Key nur nach einer erfolgreichen Authentifizierungs-Anfrage zurück. Jeder Key ist nutzerspezifisch – und damit auf den Nutzer zurückführbar, falls er verloren gehen sollte – und nur für eine begrenzte Zeit gültig. Die Liste von registrierten Nutzern mit ihren aktuellen gültigen Keys werden dabei innerhalb der Datenbank in der Cloud-Schicht gespeichert.

4.4. Automatisierte und anpassbare Produktionsplanung

Dieser Abschnitt beschäftigt sich mit einer weiteren Applikation, welche der Produktionsplanung dient. Der wesentliche Unterschied zur Prozessplanung besteht darin, dass hier ein Ablaufplan für mehrere Prozesse und mehrere Maschinen erstellt wird. Die bei der Prozessplanung entworfenen Prozesse werden also möglichst effizient aneinander gereiht und über die verfügbaren Maschinen verteilt. Dies ist vorwiegend dann relevant, wenn eine so große Menge an Prozessen zu erledigen sind, dass eine möglichst effiziente Ausführungsreihenfolge und -verteilung notwendig ist. Dies steht im Kontrast zur aktuellen Implementation der in Kapitel 5.1.2 erdachten Prozesssteuerung, da dort die Prozesse sofort ausgeführt werden und eine Maschine immer direkt angesprochen wird.

Neben der automatisierten Produktionsplanung soll es allerdings weiterhin für einen Nutzer möglich sein, direkt in den Plan einzugreifen. Ein Nutzer soll also über eine grafische Benutzeroberfläche den erstellten Produktionsplan einsehen und über intuitive Methoden (drag&drop) anpassen können.

Die Applikation soll also eine Liste an Prozessen erhalten sowie eine Liste an verfügbaren Maschinen, und soll diese Prozesse dann möglichst effizient auf die Maschinen aufteilen. Der wesentliche Fokus soll hier an der Benutzeroberfläche liegen. Eine „Maschine“ ist in diesem Kontext eine Fertigungseinheit, welche ein Produkt vollständig herstellen kann.

Die konkretere Implementierung dieser Applikation sowie auch der zuvor genannten Anwendungen findet in Kapitel 5.1 statt.

5. Implementierung, Verifikation und Validierung

Dieses Kapitel stellt eine Fortführung der Entwicklung der in Kapitel 4.2 bis 4.4 konzipierten Softwarelösungen dar. Die dort vorgestellten Konzepte werden hier zunächst implementiert, dann auf ihr Potenzial analysiert und zuletzt verifiziert und validiert.

5.1. Implementierung

5.1.1. Implementierung der Prozessplanung

Für die Implementierung gilt es nun zunächst eine Grundlage zu finden, welche die zuvor genannten Kriterien erfüllt. Der Vollständigkeit halber seien hier noch einmal alle Kriterien aufgelistet:

1. Simplizität
2. Vielschichtigkeit
3. Intuitivität
4. Zugängliche Plattform
5. JavaScript
6. Touchoptimiert
7. Dezentralisierung
8. Virtualisierung
9. Interoperabilität
10. Serviceorientierung
11. Modularität

12. Echtzeitfähigkeit

Um komplexe Prozesse auch für Laien zugänglich zu machen, bietet sich ein visuelles Interface an. Ein textbasiertes Interface wäre hierfür zu komplex und eine auf Knöpfen und Formularelementen basierende Oberfläche wäre nicht skalierbar genug.

Eine Benutzeroberfläche ähnlich der von Scratch kommt in den Sinn [Scratch Wiki, 2022], und tatsächlich gibt es von Google die clientseitige Bibliothek „Blockly“, welche auch in Applikationen wie Google Coding School, Code.org, oder Hour of Code genutzt wird. Weitere wesentliche Gründe für die Verwendung von Blockly sind die Design-Kriterien des Frameworks. So erfüllt Blockly viele der genannten Kriterien wie Simplizität, Intuitivität, Touchoptimierung, etc.

Blockly bietet einen Editor, welcher benutzt werden kann, um mithilfe von ineinander greifenden, Puzzle-artigen Blöcken über das drag&drop-Prinzip Code-Konzepte darzustellen. Dieser Editor ist in Grafik 5.1 abgebildet.

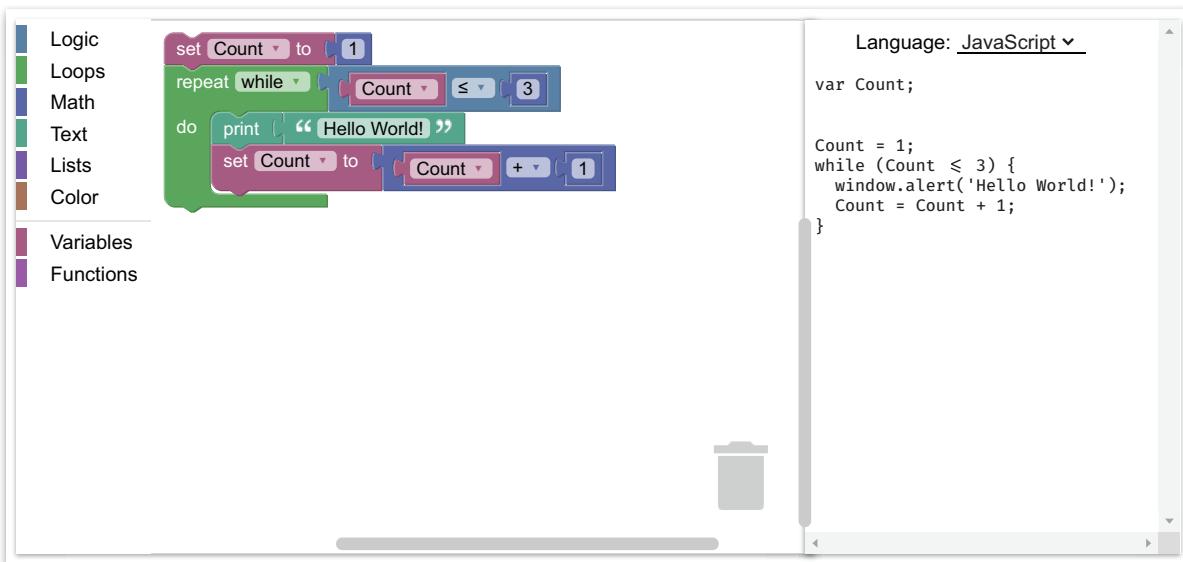


Abbildung 5.1.: Googles Blockly Editor

Die so abgebildete Logik kann dann in Form einer beliebigen Programmiersprache ausgegeben werden. Das Wesentliche ist jedoch Blocklys Erweiterbarkeit und Anpassungsfähigkeit. So ist es möglich, neue „Blöcke“ zu definieren und ihnen eine selbst definierte Logik zuweisen. Da diese Blöcke während der Initialisierung des Editors als Konfiguration übergeben werden können, ist es des Weiteren möglich, eine Konfiguration automatisiert zu generieren. So kann eine Sammlung an Blöcken (und damit Funktionsbausteinen) dynamisch für die verfügbaren Anlagen und deren Funktionen generiert werden.

So lassen sich also Blöcke für die Aktivierung und Deaktivierung aller Aktoren aller registrierter PLCs automatisch generieren. Zusätzlich dazu können noch einige generische und logische Blöcke wie Schleifen hinzugefügt werden. Daraus resultiert ein Editor, welcher wie in Grafik 5.2 auf der linken Seite eine Kategorie für jede PLC hat und darin die dazugehörigen Blöcke.

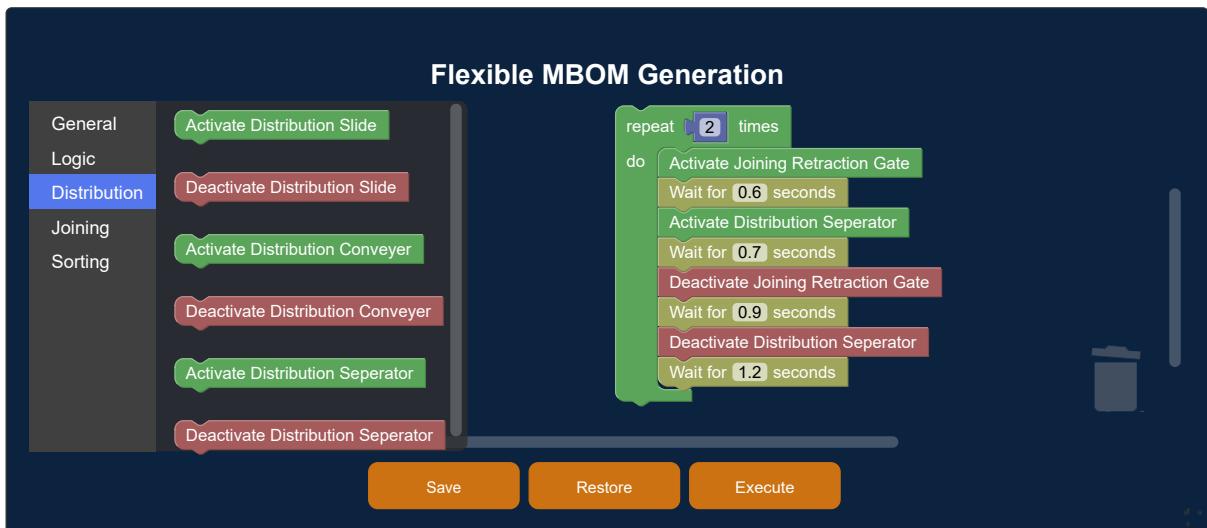


Abbildung 5.2.: Implementierter Editor mit ausgeklappter Toolbar

Wie ebenfalls in Grafik 5.2 zu sehen ist, lassen sich diese Blöcke über drag&drop aus der Toolbar (der Sammlung von Blöcken auf der linken Seite) in den Arbeitsbereich (dem scrollbaren Bereich auf der rechten Seite) ziehen. Dort greifen die Blöcke ineinander, was mit dem Nutzer in Form von visuellen und akustischen Hinweisen kommuniziert wird. Welche Blöcke wie ineinander greifen können, ist außerdem direkt anhand der Zähne zu erkennen, während die Farbe der Blöcke über den Typ Blockes aufklärt. So sind etwa Aktivierungsblöcke grün und Deaktivierungsblöcke rot.

Neben diesen Blöcken zur Steuerung der Aktoren gibt es auch Blöcke für allgemeine Funktionen wie etwa zum Warten oder Loggen, elementare Blöcke für Zahlen oder Strings, logische Blöcke für Schleifen oder (konzeptuelle) Bedingungen, und boolesche Blöcke für den Status eines Aktuators. Ein solcher logischer Block ist etwa in Grafik 5.3 abgebildet.

Zudem wurde eine Speicherungs- und Wiederherstellungs-Funktion hinzugefügt, welche es jedem Nutzer der Webseite ermöglicht, seinen aktuellen Arbeitsbereich exakt abzuspeichern und ihn jederzeit wiederherzustellen. Um einen Prozess zu speichern, wird

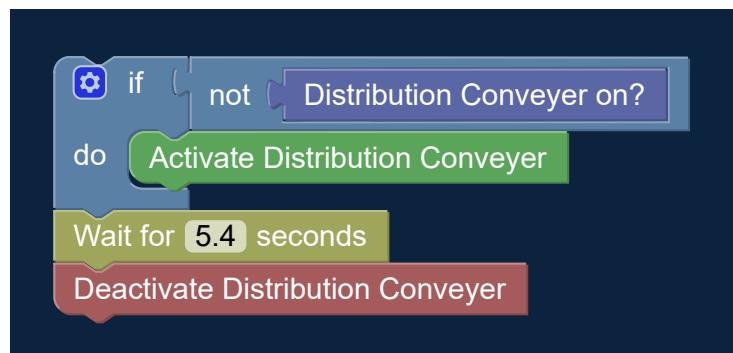


Abbildung 5.3.: Beispiel eines Logik-Blocks

dieser zu XML konvertiert und dieser XML Code wird dann als String abgespeichert. Wo genau diese Daten abgespeichert und übermittelt werden und wie die Authentifizierung funktioniert, wird im Folgenden in Kapitel 4.3 behandelt.

Neben dem Prozess-Editor wurde zudem noch ein Fenster für einen Live-Videostream der Anlage eingefügt, sodass ein Nutzer in nahezu Echtzeit die Ausführung des Prozesses auf der echten Anlage beobachten kann. Außerdem wurde ein Graph eingefügt, welcher den aktuellen Status aller Aktuatoren dieser Anlage anzeigt. Diese beiden Blöcke sind in Grafik 5.4 abgebildet. Im Anhang dieser Arbeit unter A.1 lässt sich zudem der XML Code eines simplen Prozesses finden.

Übersichtsseite

Neben der „Flexible MBOM Generation“ Seite wurde auch eine „MBOM Overview“ Seite angelegt, welche es ermöglicht, die gespeicherten Prozesse aller Nutzer anzuzeigen und Metriken zu ihnen zu berechnen. Dies erklärt auch die Benennung der Seiten: „MBOM“ steht für „Manufacturing bill of materials“ und repräsentiert eine Liste, welche alle erforderlichen Teile und Baugruppen enthält, die benötigt werden, um ein vollständiges Produkt anzufertigen. Der geplante Prozess enthält dabei theoretisch alle benötigten Informationen, wie etwa die verbrauchten Ressourcen, jedoch ist es recht umständlich, diese Daten aus den Blöcken abzulesen. Die Übersichtsseite erleichtert diese Aufgabe, indem der Prozessplan analysiert und alle relevanten Informationen übersichtlich präsentiert werden. In Grafik 5.5 ist diese Seite abgebildet. Wie dort zu sehen ist, kann oben ein Prozessplan ausgewählt werden, welcher dann links angezeigt wird; auf der rechten Seite werden dann die relevanten extrahierten Daten zu diesem Prozessplan angezeigt.

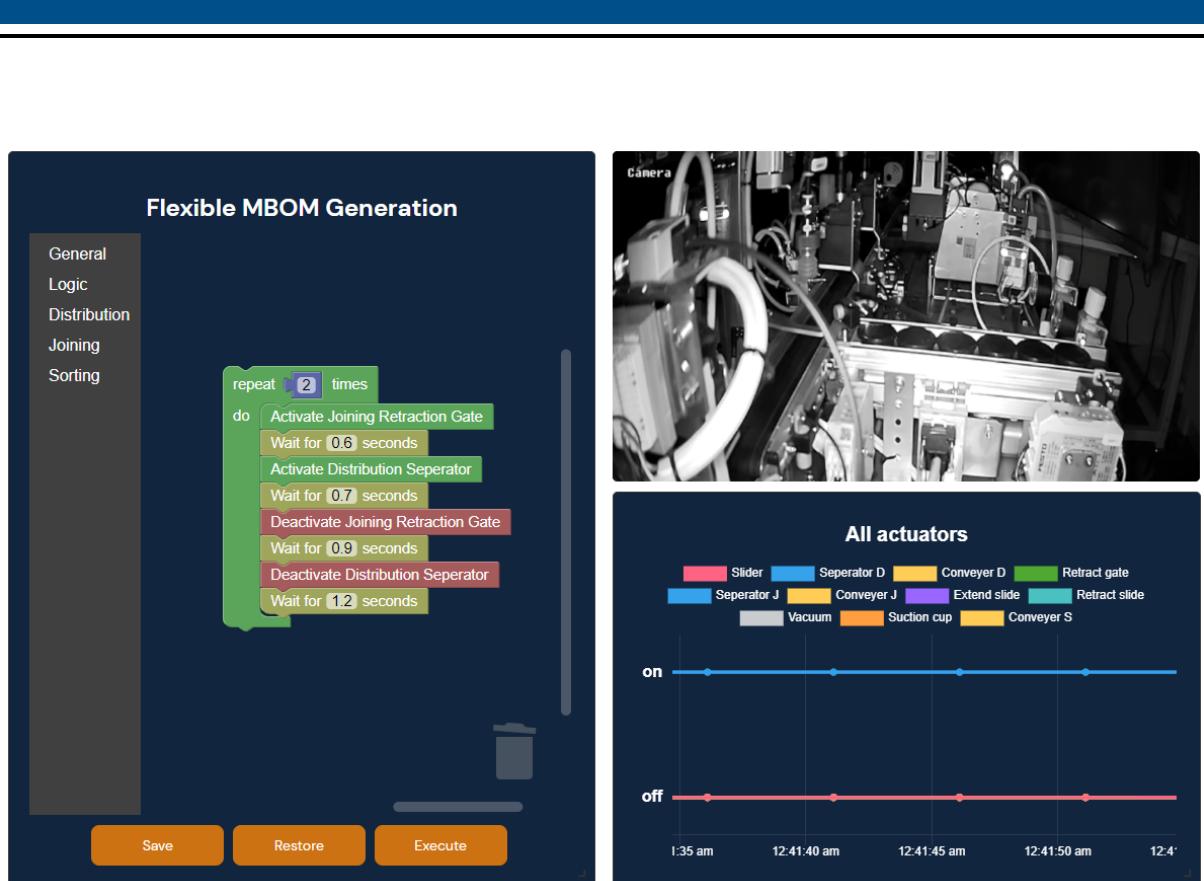


Abbildung 5.4.: Aufbau der Prozessplanungs-Seite

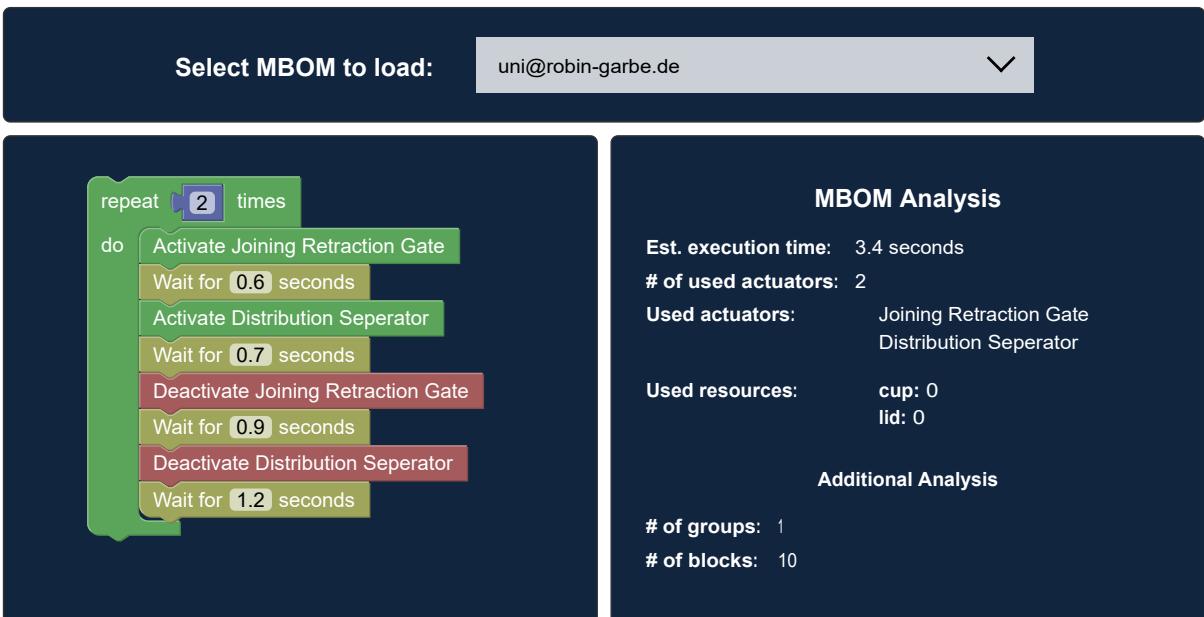


Abbildung 5.5.: Prozessplanungs-Übersichtsseite

Um diese Informationen zu extrahieren, wird der XML String des Prozesses untersucht und ein JavaScript Objekt mit den relevanten Daten wird erstellt. Dieses Objekt wird dann genutzt, um die erwähnten Informationen anzuzeigen. Der Code, um dieses Objekt zu erstellen, kann im Anhang unter A.2 gefunden werden.

5.1.2. Implementierung der Prozesssteuerung

Die Cloud-Schicht mit simpler REST API und Datenbank besteht bereits und wird bei AWS gehostet. Die API ist mit C# geschrieben und für die Datenbank wird eine manuell installierte MariaDB Datenbank genutzt. Das C# Programm umfasst noch eine Vielzahl zusätzlicher Module, wie etwa für MQTT, OPC, Rest Connection, VPN-Server, DNS-Server und einige weitere.

Die Authentifizierung wird auf der Digital Twin Academy Webseite zentral geregelt und der API-Key wird in dem lokalen Speicher des Web-Browsers – konkreter in dessen „SessionStorage“ – abgelegt und nach dem Schließen des Browsertabs werden diese Daten automatisiert gelöscht. So wird eine clientseitige Datenspeicherung konform zu den EU Regulationen und den aufgeführten Sicherheitsstandards erreicht.

ES11 hat Zugriff auf die `fetch` Browser-API, welche genutzt werden kann, um mithilfe von JavaScript eine Anfrage an die REST API zu senden. Des Weiteren unterstützt ES11 `async` Funktionen, welche genutzt werden können, um die `fetch` Anfragen asynchron abzuhandeln. In Blockly kann jedem Block Code in String-Form zugeordnet werden. Wenn dann das Blockly „Programm“ zu Code exportiert werden soll, werden diese String einfach entsprechend der definierten Logik konkateniert. Das Resultat ist ein Skript als String, welches dann vom Browser geparsst und ausgeführt werden kann. Jeder Aktivierungs- oder Deaktivierungs-Block führt also letztlich eine `fetch` Anfrage an die API aus. Die API prüft die Validität des API-Keys und ob die Aktion gültig ist; so dies der Fall ist, wird der geänderte Status des Aktuators in der Datenbank vermerkt und über den integrierten OPC-UA Client wird die Anfrage direkt an die entsprechende PLC weitergegeben.

Im Folgenden ist das JavaScript Object angegeben, welches einen Block konfiguriert. Hier sei beispielhaft der häufig verwendetet `instruction_wait` angeben, welcher die Ausführung für die angegebene Anzahl an Sekunden anhält.

```
1 {
2   "type": "instruction_wait",
3   "message0": "Wait for %1 seconds",
```

```

4 "args0": [
5   {
6     "type": "field_number",
7     "name": "wait_amount",
8     "value": 1,
9     "min": 0,
10    "max": 600,
11    "precision": 0.1
12  }
13 ],
14 "previousStatement": null,
15 "nextStatement": null,
16 "colour": 65,
17 "tooltip": "",
18 "helpUrl": ""
19 }

```

Diesem Block kann nun mit dem folgenden Code Funktionalität zugewiesen werden. Diese Funktion wird auch „Block Generator“ genannt. Wie bereits erwähnt, wird der zugewiesene Code in Form eines Strings übergeben.

```

1 Blockly.JavaScript['instruction_wait'] = function(block) {
2   var number_wait_amount = block.getFieldValue('wait_amount');
3   var code = `await new Promise(resolve => setTimeout(resolve, ${Number(
4     number_wait_amount) * 1000}));\n`;
5   return code;
5 };

```

Das Programm nimmt sich also den API-Key aus der SessionStorage, stellt damit eine `fetch` Anfrage an die API in der Cloud, um alle nötigen Informationen zu den verfügbaren PLCs und Aktoren einzusammeln, um damit dann den Editor für die Prozessplanung zu erstellen. Wird eine Prozessplanung ausgeführt, so wird das Prozesssteuerungs-Skript generiert, geparsst und ausgeführt.

Die HTTP POST Methode wird verwendet, um Daten ohne Rückgabewert an die API zu senden und die GET Methode wird genutzt, um Daten abzurufen. Im Folgenden sei etwa der Code der selbst-aufrufenden, asynchronen Pfeilfunktion für POST Anfragen aufgeführt. `endpoint` stellt dabei die (Teil-)Adresse für den anzusteuerten Aktuator dar und wird übergeben. `api_key` ist der aus der SessionStorage ausgelesene API-Key.

```

1 (async () => {
2   const url = "https://digitaltwinservice.de/api/Database/" + endpoint;

```

```

3 const response = await fetch(url, {
4   method: "POST",
5   headers: {
6     "X-API-KEY": api_key,
7     "accept": "/"
8   }
9 })
10 .then(resp => resp.status);
11 console.log("%cResponse status: " + response, "background:#5C4084bb;color
12 :white;padding:1px 4px 1px 16px; border-radius:2px;");
13 return response;
14 })()

```

Der beschriebene Ausführungsmodus wird als „direct“ bezeichnet. Neben diesem Ausführungsmodus wurde auch der „virtual“ Modus angelegt, welcher dafür gedacht ist, anstelle der echten Anlage eine virtualisierte Anlage anzusteuern. Hierauf wird in Kapitel 5 näher eingegangen.

Nun sei noch ein Ablaufplan für die Kommunikation zwischen den drei Schichten für eine typische Prozessplanung und -steuerung in Grafik 5.6 abgebildet.

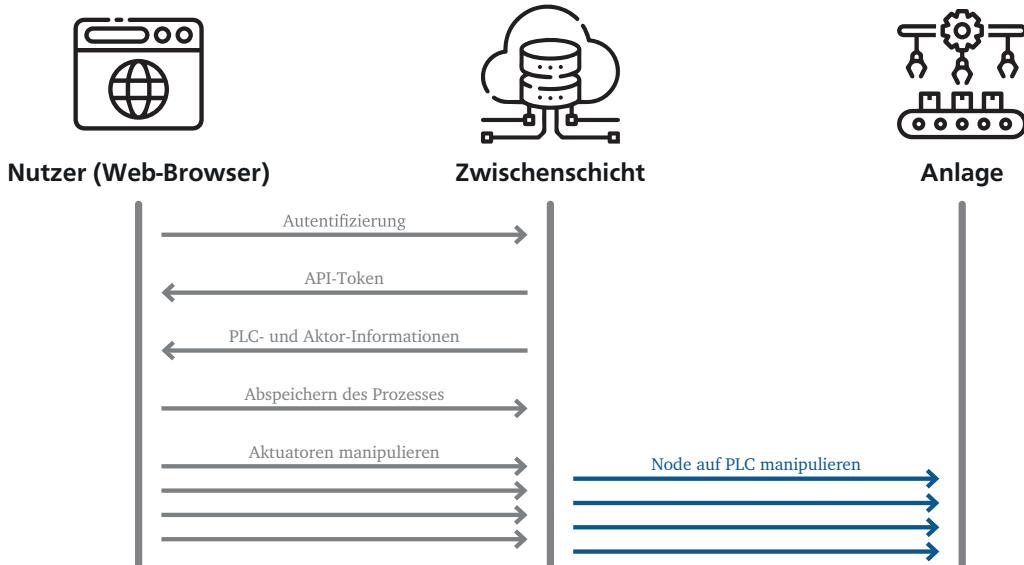


Abbildung 5.6.: Ablaufplan einer Prozessplanung und -steuerung

5.1.3. Implementierung der Produktionsplanung

Die Implementierung der Produktionsplanung läuft in großen Teilen analog zur Prozessplanung und sei hier nur kurz angeschnitten. In Grafik 5.7 ist die erstellte Benutzeroberfläche für die Produktionsplanung abgebildet. Jede Maschine wird als eigene Zeile dargestellt und jeder Auftrag wird als Block abgebildet. Die Breite eines Blockes repräsentiert dessen Ausführungsduer, während die Farbe für die Art des Blockes steht. Die „Art“ bezeichnet hier eine Produktkonfiguration, wobei vorgesehen ist, dass jede Maschine nur eine Art von Produktkonfigurationen anfertigen kann. Jedoch soll eine Maschine umgerüstet werden können, sodass um andere Konfigurationen anfertigen zu können.



Abbildung 5.7.: Benutzeroberfläche der Produktionsplanung

Es sei jedoch erwähnt, dass diese Anwendung hier nur konzeptionell umgesetzt wird, da es zum aktuellen Zeitpunkt noch keine Möglichkeit gibt, diese Produktionsplanung an den bestehenden Produktionsablauf anzubinden. Dies liegt neben dem Fehlen von mehreren und unterschiedlichen Anlagen vorwiegend an der Art der Planung. Für eine Demo- und Bildungs-Webseite wie Digital Twin Academy wäre es wenig sinnvoll, wenn alle geplanten Prozesse erst in den kommenden Tagen ausgeführt werden; stattdessen sollten die Prozesse möglichst zeitnah an die Anlage weitergeben werden. Zudem werden bei der aktuellen Architektur die PLCs der Anlagen direkt angesteuert, was eine Lastenverteilung über mehrere Anlagen hinweg wenig sinnvoll macht. Zuletzt rechtfertigt die aktuelle und auch die auf kurze Sicht vorhergesehene Menge an auszuführenden Prozessen eine so komplexe Produktionsplanung nicht. Anders als die in Kapitel 4 betrachtete endbenutzergesteuerte Prozessplanung und -steuerung wird hier also das theoretische Potenzial einer automatisierten und anpassbaren Produktionsplanung im Kontext von Industrie 4.0 im Allgemeinen untersucht.

5.2. Potenzialanalyse

Wie bereits in Kapitel 1.1 angesprochen wurde, ist die Verbreitung von Industrie 4.0 und IIoT Methoden in der Industrie verbesserungsfähig. Die vorgestellten Softwarelösungen haben zum einen dank ihrer Eigenschaften wie Interoperabilität, Serviceorientierung und Modularität die Möglichkeit, auf breiter Fläche eingesetzt werden zu können und so die Vorteile und Potenziale von Industrie 4.0 und IIoT zu weiteren Anbietern zu bringen; zum anderen haben diese Applikationen als Teil der Digital Twin Academy ein didaktisches Potenzial, da die Plattform diese Themenbereiche Benutzern aus der Industrie und dem akademischen Bildungsbereich näher bringt und so versucht, eine größere Adaption und Wahrnehmung dieser Technologien zu erreichen.

Potenzialanalyse der Prozessplanung und -steuerung

Nun soll ein näherer Blick auf die endbenutzergesteuerte Prozessplanung und -steuerung geworfen werden. Diese Anwendung erweitert die für Industrie 4.0 und IIoT typischen Potenzialen, da sie bestehende Anlagen und Fertigungsstätten auf moderne Gegebenheiten wie Heimarbeit oder Fernwartung vorbereitet.

Ein weiterer Anwendungsfall einer solchen Applikation ist aufgrund ihrer niedrigen Eintrittsbarriere das Manufacturing-as-a-Service (MaaS). MaaS bezeichnet die gemeinsame Nutzung einer vernetzten Fertigungsinfrastruktur zur Herstellung von Produkten, wobei diese Nutzung als Service angeboten wird.

Zuletzt sei noch die Gesamtarchitektur in den Fokus gehoben. REST APIs und OPC-UA sind beide vorherrschende Kommunikationsprotokolle, welche sich in verschiedenen Bereichen etabliert haben, wobei in der Fertigungs-Industrie in den meisten Fällen OPC-UA für die Kommunikation genutzt wird, insbesondere zu Maschinen. IoT-Geräte nutzen jedoch üblicherweise REST APIs für ihre Kommunikation. Eine Verbindung beider Protokolle ist hier nicht nur sinnvoll, sondern es hilft auch dabei, diese beiden genannten Bereiche einander näherzubringen – eine Aufgabe, welche zentraler Bestandteil von Industrie 4.0 und IIoT ist.

Potenzialanalyse der Produktionsplanung

Das Potenzial betreffend ist diese Applikation wie zu erwarten analog zu der Softwarelösung zur Prozessplanung und -steuerung (siehe Kapitel 5.2). Aufgrund des ähnlichen Anwendungsfalles gelten die dort genannten Punkte also auch hier.

5.3. Verifikation und Validierung

In diesem Abschnitt soll es um die Verifikation und Validierung der zuvor konzipierten und implementierten Applikationen gehen. Es wird also geprüft, ob die Software alle erwarteten Anforderungen vollständig erfüllt, den Bedürfnissen der Benutzer nachkommt und entsprechend der bestimmungsgemäßen Verwendung funktioniert.

5.3.1. Einhaltung von Standards

Der erste und simpelste Punkt in diesem Kapitel soll eine Reflexion auf die Standards sein, welche für die Softwareentwicklung genutzt wurden. Neben der angemessenen Befolgung der Richtlinien des Europa Web Guide [Mauri und Bourrouet, 2021c], gilt es Standards wie den industriellen Cyber-Sicherheitsstandard IEC 62443 [TC 65 - Industrial-process measurement, control and automation, 2009] und Regulationen wie die europäische Regulation IEC 62443 [European Parliament and Council, 2018] zu befolgen. Dadurch, dass all diese Richtlinien, Standards und Regulationen bereits in die Konzeption der Applikationen integriert und bei der Implementierung befolgt wurden, muss nun keine Anpassung mehr vollzogen werden.

5.3.2. Gleichzeitige Ausführung mehrerer Prozesse

Bei der aktuellen Architektur kann es zu Problemen führen, wenn mehrere Nutzer versuchen, gleichzeitig einen Prozess auszuführen oder wenn ein Nutzer sehr kurz nacheinander zwei langwierige Prozesse ausführt. In diesen Fällen würden die Befehle verschachtelt bei der API eingehen und damit auch verschachtelt an die PLC weitergeleitet werden, wo nicht mehr zwischen den beiden Prozessen unterschieden werden könnte. Falls die gleiche Anlage angesprochen wird, so kann es offensichtlich zu massiven Problemen

führen, wenn zwei Prozesse gleichzeitig ausgeführt werden.

Um dieses Problem zu mildern, wurde die API in der Cloud um ein Nachrichten-System erweitert, welches sowohl benutzt werden kann, um alle Nutzer der Prozessplanungs-Seite darüber zu informieren, wenn und wann ein Benutzer einen Auftrag abgeschickt hat und wie lange dieser in etwa dauern wird, als auch um ein Warteschlangen-System einzurichten. Diese Warteschlange sorgt dann dafür, dass ein Nutzer einen Auftrag erst abschicken kann, wenn gerade kein anderer Auftrag ausgeführt wird.

In Grafik 5.8 sind die beiden Blöcke für das Ereignisprotokoll und die Warteschlange abgebildet. Diese Blöcke wurden unten auf die Prozessplanungs-Seite eingefügt, sodass sie von allen Nutzern während dem Benutzen des Prozess-Editors gesehen werden.

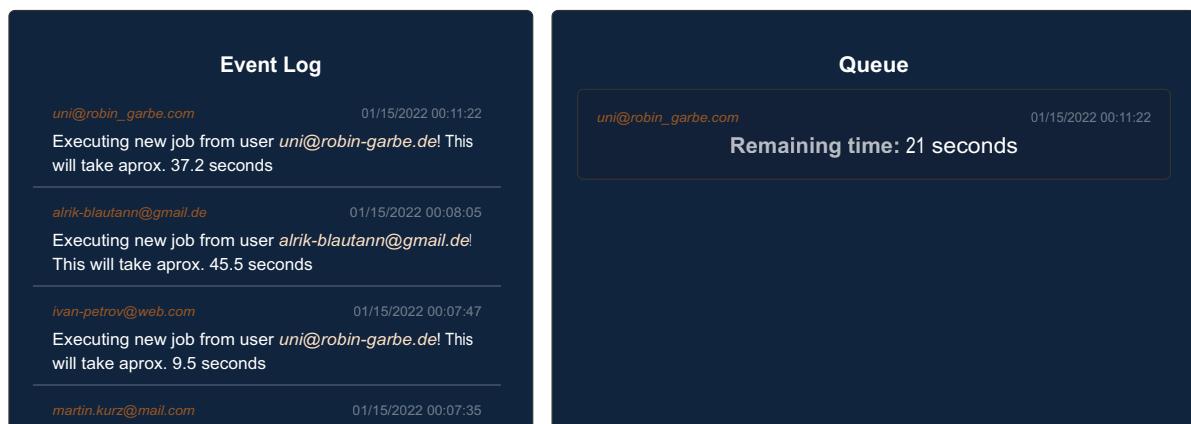


Abbildung 5.8.: Ereignisprotokoll und Warteschlange

5.3.3. Latenzen und Latenz-Varianzen

In Grafik 5.6 wurde der Ablaufplan einer typischen Prozessplanung und -steuerung dargestellt. Wie dort zu sehen und in Kapitel 4.3 beschrieben ist, wird für jede Aktivierung oder Deaktivierung eines Aktuators eine einzelne Anfrage an die Cloud und von dort an die entsprechende PLC gesendet. Dies macht es nötig, einen näheren Blick auf die Latenzen und vor allem die Latenz-Varianzen zu werfen, da eine Schwankung in den Ankunftszeiten der einzelnen Befehle den Programmfluss massiv beeinträchtigen könnte. Um dieses Problem näher zu verdeutlichen, wird der in Grafik 5.9 dargestellte Prozess betrachtet. Bei diesem simplen Prozess wird zunächst ein Bauteil herausgegeben und daraufhin wird das Förderband aktiviert bevor es nach exakt 4,3 Sekunden wieder deaktiviert wird.

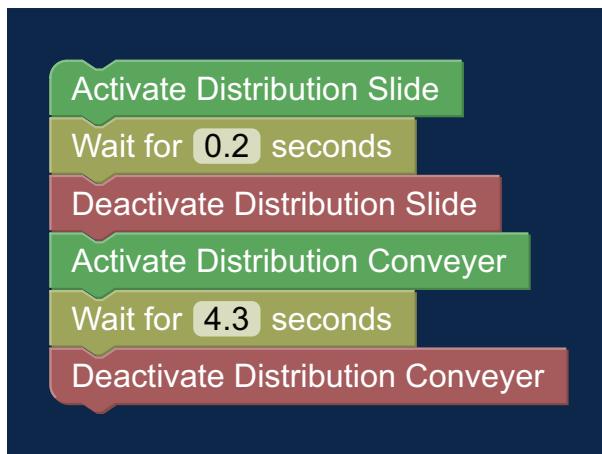


Abbildung 5.9.: Potenziell problematischer Prozess

In diesem Beispiel sei die Dauer von 4,3 Sekunden dadurch relevant, weil das Bauteil nach exakt dieser Zeit unter einem weiteren Element wie einem RFID Lesegerät steht. Wenn das Förderband also nicht nach präzise dieser Dauer wieder deaktiviert wird, ist das Bauteil nicht genau unter dem Lesegerät und der Prozess kann nicht fortgesetzt werden. Die Latenz ist hier nicht hochgradig relevant, da es keine entscheidende Rolle spielt, nach wie vielen Millisekunden der Prozess gestartet wird. Die Latenz-Varianz ist allerdings von massiver Wichtigkeit, da eine hohe Varianz in diesem Beispiel bedeuten könnte, dass das Förderband kürzer oder länger aktiv ist. Ein Beispiel hierfür wäre, wenn der Aktivierungsbefehl für das Förderband nach 0,1 Sekunden bei der PLC eingeht, der Deaktivierungsbefehl aber 0,3 Sekunden benötigt; das Förderband wäre dann für 4,5 Sekunden aktiv.

Um zu überprüfen, ob die Latenz tatsächlich in einem problematischen Maß schwanken, wurden einige Testreihen am echten Gesamtsystem durchgeführt. Bei diesen drei Testreihen wurde je 20 Mal mit einer Verzögerung von je zehn Sekunden ein Aktuator über die API in der Cloud angesteuert. Es wurde die Differenzen der Zeitstempel zwischen dem Senden des Befehls und dessen Eingang auf der PLC protokolliert und nach den 20 Durchläufen wurde die höchste und niedrigste Zeitstempel-Differenz sowie die Differenz aus diesen beiden notiert. Es sei hier angemerkt, dass die Differenzen der Zeitstempel zwischen dem Senden und Empfangen nicht äquivalent sind zu der Latenz, da die internen Uhren der Testgeräte nicht exakt aufeinander abgestimmt sind.

In Tabelle 5.1 stehen die Ergebnisse zu diesen drei Testläufen. Wie dort zu sehen ist, ist die Schwankung der Latenzen problematisch hoch. Die Latenz-Varianz hat bei diesen Testreihen etwa bei drei bis sechs Sekunden gelegen, was noch viel bedenklicher ist als

zunächst vermutet. Bei einer Varianz von fünf Sekunden könnte es so passieren, dass im obigen Beispiel das Förderband statt 4,3 Sekunden sogar 9,3 Sekunden aktiviert sein könnte.

| | Minimale Zeist.-Diff. | Maximale Zeist.-Diff. | Varianz |
|--------------|-----------------------|-----------------------|---------|
| Testreihe C1 | 3597897ms | 3603408ms | 5511ms |
| Testreihe C2 | 3597918ms | 3601218ms | 3300ms |
| Testreihe C3 | 3597972ms | 3600962ms | 2990ms |

Tabelle 5.1.: Testreihen zur Messung der Latenz-Varianz über die Cloud

Um dieses Problem zu lösen, könnte der Steuerungsablauf so angepasst werden, dass die Befehle nicht einzeln vom Browser gesendet werden, sondern dass der Browser den gesamten Prozess in einer Anfrage sendet. Die Applikation in der Cloud würde den Prozess dann an einen Computer – welcher sich im gleichen lokalen Netz wie die anzusteuерnde PLC befindet – senden. Dieser Computer würde den erhaltenen Prozess dann parsen und ausführen, wobei er die einzelnen Befehle sowohl als Update an die Datenbank weitergibt, sodass diese stets ein Abbild des aktuellen Zustandes hat, als auch an die PLC sendet. Die Latenz-Varianzen bei einer Übertragung im lokalen Netz an die PLC sind verschwindend gering. Um dies zu zeigen, wurden drei weitere Testreihen durchgeführt, bei welchen je 100 Befehle in einem zufällig gewählten Abstand von 1 bis 2 Sekunden gesendet werden. Wie im letzten Testlauf wurden die Ergebnisse notiert und sind in Tabelle 5.2 aufgeführt. Die Latenz-Varianzen sind hier wie vermutet mit unter 20 Millisekunden vernachlässigbar niedrig.

| | Minimale Zeist.-Diff. | Maximale Zeist.-Diff. | Varianz |
|--------------|-----------------------|-----------------------|---------|
| Testreihe L1 | 3558532ms | 3558547ms | 15ms |
| Testreihe L2 | 3558538ms | 3558542ms | 4ms |
| Testreihe L3 | 3558539ms | 3558551ms | 12ms |

Tabelle 5.2.: Testreihen zur Messung der Latenz-Varianz über das lokale Netzwerk

Die beschriebene neue Kommunikations-Infrastruktur ist in Grafik 5.10 abgebildet. Der wesentliche Unterschied zur vorherigen Infrastruktur (abgebildet in Grafik 4.4) ist das

Hinzunehmen eines Computers im lokalen Netz der Anlage. Dies ist jedoch wenig problematisch, da ein solcher Computer in der aktuell eingesetzten Infrastruktur ohnehin bereits eingesetzt wird, um den Live-Videostream der Anlage aufzunehmen und zu senden.

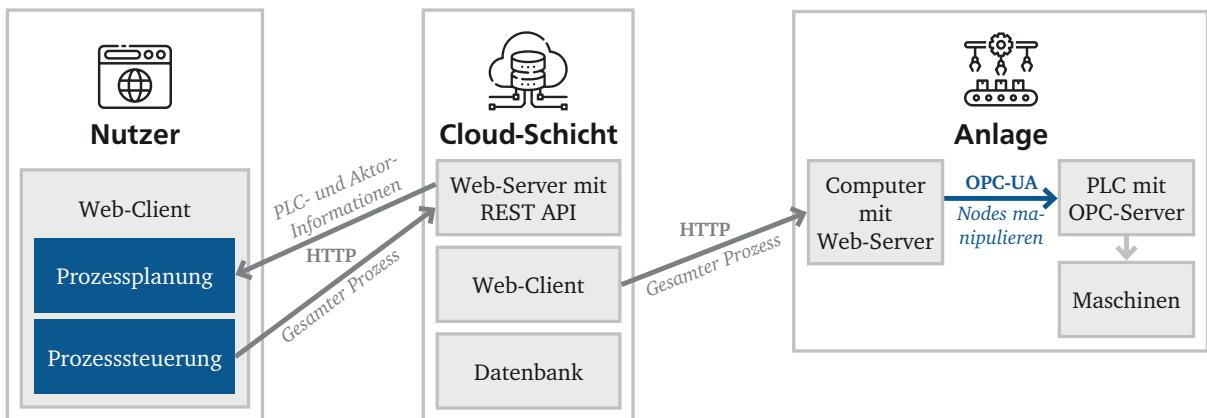


Abbildung 5.10.: Verbesserte Kommunikations-Infrastruktur

5.3.4. Tests

Da die aktuelle Webseite der Digital Twin Academy aufgrund ihrer Authentifizierung und ihres ständig ändernden Aufbaus ein automatisiertes Testen praktisch unmöglich macht, wurden manuelle Tests verwendet. Im Folgenden werden die getesteten Aspekte der Prozessplanungs und -steuerungs sowie der Produktionsplanungs-Applikation aufgeführt:

Endbenutzergesteuerte Prozessplanung und -steuerung

- Kompatibilität mit den in Kapitel 4.2 spezifizierten Browsern wurde getestet
- Uneingeschränkte Benutzbarkeit in diesen Browsern wurde überprüft
- Korrektheit der gesendeten API-Anfragen wurde sichergestellt
 - Sowohl bei der Generierung des Editors als auch bei der Ausführung eines Prozesses sind alle Aufrufe der API an eine korrekte Adresse, mit der korrekten Methode und mit einem eventuellen Rückgabewert wird korrekt umgegangen
- Fehlerfälle wie etwa eine Benutzung ohne gültige Autorisierung wurden überprüft

- Angriffsmöglichkeiten wie das Senden von ungültigen oder zu vielen Anfragen wurden erprobt

Automatisierte und anpassbare Produktionsplanung

Da es sich bei dieser Applikation lediglich um eine konzeptuelle Umsetzung in Form einer Benutzeroberfläche handelt, beschränken sich die überprüften Punkte auf die Kompatibilität und Benutzbarkeit in verschiedenen Browsern.

5.3.5. Simulation

Statt einen Prozess direkt an eine Anlage zu senden, sollte es möglich sein, ihn stattdessen an eine Simulation zu übergeben. Der Zustand dieser Simulation würde dann in Echtzeit als 3D Animation mittels Unity-Applet angezeigt werden, ähnlich wie auch der Zustand der echten Anlage in nahezu Echtzeit als Videostream angezeigt wird. Eine grundlegende Virtualisierung der Anlage sowie eine grobe Interaktion mit dieser Simulation besteht zwar bereits, allerdings ist dieses System zum Zeitpunkt dieser Arbeit noch bei Weitem nicht ausgereift genug, um es als Simulationsumgebung nutzen zu können. Es wurden allerdings die nötigen Weichen in der Prozesssteuerung angelegt, um eine solche Simulation in Zukunft problemlos zu ermöglichen, sobald eine Umgebung dafür besteht.

Ergebnisse der Verifikation und Validierung

Als Ergebnis der oben beschriebenen Punkte wurde die Funktion, Korrektheit und Sicherheit der Applikationen für den Kontext als konzeptuelle Umsetzung zufriedenstellend verifiziert und validiert. Mit Blick auf die Gesamtarchitektur sind weitere Schritte in den Bereichen der Cyber-Sicherheit möglich und für eine industrielle Anwendung nötig, jedoch liegen die hier zu verändernden Aspekte außerhalb des Umfangs dieser Arbeit und sind primär bei der Weiterentwicklung des C# Programmes und spezifischer der REST API zu finden.

6. Ausblick und Zusammenfassung

In diesem abschließenden Kapitel soll zunächst ein Ausblick auf zukünftige Arbeiten und Forschungsansätze geworfen werden, welche auf dieser Thesis aufbauen oder an sie anknüpfen. Im Anschluss daran werden die wesentlichen Inhalte und wichtigsten Ergebnisse dieser Arbeit noch einmal zusammengefasst.

6.1. Ausblick

Zunächst sei natürlich auf die in Kapitel 5.3.3 entwickelte und in Grafik 5.10 abgebildete verbesserte Kommunikations-Infrastruktur hingewiesen. Ein zentraler Ansatzpunkt für eine weitere Arbeit wäre demnach, diese Infrastruktur zu implementieren, um so die Latenz-Varianzen signifikant zu senken. Dieses Gesamtsystem kann jedoch auch noch an vielen Ecken zusätzlich erweitert und ausgebaut werden. So kann die Cloud-Schicht um weitere Funktionen wie etwa WebSockets (siehe Kapitel 3.5) oder Webhooks erweitert werden. So könnte etwa der Nachrichten-Service seine Daten immer sofort vom Server erhalten, statt ihn periodisch nach Änderungen abzufragen. Zudem wäre so eine tiefgehendere Kommunikation zwischen den Schichten der vorgeschlagenen Infrastruktur möglich, da so auch Statusmeldungen von der Cloud direkt zum Nutzer durchgegeben werden könnten. Zu einem ähnlichen Zweck kann der Computer, welcher in Grafik 5.10 hinzugefügt wurde und welcher das Parsen und Weitergeben der Befehle an die PLC übernimmt, um einen Web-Client erweitert werden. So könnte auch vonseiten der Maschine her eine umfangreichere Kommunikation an die Cloud ermöglicht werden.

Die Anbindung einer Simulation mittels digitalem Zwilling bietet ein großes Potenzial, da so zum einen eine bessere Planung möglich ist, aber zum anderen auch ein umfangreicheres und agileres Testen ermöglicht wird. Außerdem könnten – sobald die Webseite der

Digital Twin Academy ausgereifter ist – automatisierte Integrationstests genutzt werden, um die ständige und fortlaufende Funktionsfähigkeit der vorgestellten Applikationen zu gewährleisten.

Die in Kapitel 4.4 vorgestellte Applikation zur automatisierten und anpassbaren Produktionsplanung sei noch gesondert hervorgehoben, da diese Softwarelösung bisher als ein alleinstehendes Konzept entwickelt wurde. Mit einer weiter ausgearbeiteten Cloud-Schicht, mehr Anlagen ähnlichen Typs und mehr Nutzern der Seite würde es sich lohnen, diese Anwendung in den Arbeitsablauf zu integrieren.

Neben diesen Erweiterungsmöglichkeiten besteht natürlich auch das Potenzial des trivialen Ausbaus der aktuell angedachten Struktur. Das implementierte System kann automatisiert mit neuen Anlagen zureckkommen, sodass weitere PLCs einfach hinzugefügt werden und direkt benutzbar sind.

Als Teil der Digital Twin Platform haben die in dieser Thesis vorgestellten Softwarelösungen ein großes Wachstums- und Erweiterungspotenzial. Zum aktuellen Zeitpunkt sind bereits mehrere Bachelor- und Masterarbeiten rund um die Digital Twin Platform in Arbeit.

6.2. Zusammenfassung der wichtigsten Ergebnisse

Industrie 4.0 und IIoT sind Schlüssel-Technologien zur individuellen Produktion mittels Manufacturing-on-demand und Manufacturing-as-a-Service (MaaS). Hierfür werden interoperable, serviceorientierte und modulare Anwendungen benötigt, welche es ermöglichen, Endnutzer auf eine simple und benutzerfreundliche Art direkt mit den Maschinen zu verbinden. Neben der Integration der Verbraucher in die Produktion mittels MaaS sind auch moderne Anwendungsfälle wie Heimarbeit oder Fernwartung bedeutend. Eine allgemeinere Architektur mit einer flexiblen Benutzeroberfläche ist hierbei vorteilhafter als eine Vielzahl von Einzellösungen.

Es bieten sich also webbasierte und benutzerfreundliche Applikationen an, welche vom Benutzer für verschiedene Anlagen genutzt werden können und eine besonders niedrige Eintrittsbarriere haben. Die Modularität der vorgestellten Lösungen steht dabei im Vordergrund, sodass sie auf breiter Ebene einsetzbar sind.

In dieser Thesis wurden demnach Softwarelösungen zur endbenutzergesteuerten Prozessplanung und -steuerung sowie zur Produktionsplanung konzipiert, implementiert und auf ihr Potenzial analysiert. Beide Applikationen wurden in die Digital Twin Academy Webseite integriert. Es wurde das Potenzial einer Fertigung nach den Maßstäben von Manufacturing-as-a-Service aufgezeigt und die entwickelten Anwendungen wurden verifiziert und validiert.

Literatur

- [1] Reiner Anderl. „Industrie 4.0 – technological approaches, use cases, and implementation“. In: *at - Automatisierungstechnik* 63.10 (2015), S. 753–765. doi: doi: 10.1515/auto-2015-0025. URL: <https://doi.org/10.1515/auto-2015-0025>.
- [2] Christian Babski. *The developer's guide to browser adoption rates*. 18. Okt. 2011. URL: <http://blog.computedby.com/archives/100-The-developers-guide-to-browser-adoption-rates.html> (besucht am 10.01.2022).
- [3] Tim Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.0*. RFC 1945. Internet Engineering Task Force, Mai 1996. doi: 10.17487/RFC1945. URL: <https://rfc-editor.org/rfc/rfc1945.txt>.
- [4] Tim Berners-Lee, Roy T. Fielding und Larry M. Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986. Internet Engineering Task Force, Jan. 2005. doi: 10.17487/RFC3986. URL: <https://rfc-editor.org/rfc/rfc3986.txt>.
- [5] Sam Bhattacharai u. a. *The Industrial Internet of Things Volume G1: Reference Architecture*. IIRA-v1.9. Industry IoT Consortium, 19. Juni 2019. URL: <https://www.iiconsortium.org/pdf/IIRA-v1.9.pdf>.
- [6] Matthias Biehl. *RESTful API Design*. 1. Aufl. API-University Press, 2016.
- [7] Mike Bishop. *Hypertext Transfer Protocol Version 3 (HTTP/3)*. Internet-Draft. Work in Progress. Internet Engineering Task Force, 2. Feb. 2021. URL: <https://datatracker.ietf.org/doc/html/draft-ietf-quic-http-34>.
- [8] Tim Bray. *The JavaScript Object Notation (JSON) Data Interchange Format*. RFC 8259. Internet Engineering Task Force, Dez. 2017. doi: 10.17487/RFC8259. URL: <https://rfc-editor.org/rfc/rfc8259.txt>.

-
-
- [9] Cambridge Innovation Institute. *John Deere turns to IoT to make smart farming a reality*. 2020. URL: <https://internetofbusiness.com/john-deere-turns-iot-smart-farming/> (besucht am 22.11.2021).
 - [10] Alexis Deveria. *HTTP/2 protocol*. Nov. 2021. URL: <https://caniuse.com/http2> (besucht am 27.11.2021).
 - [11] Alexis Deveria. *HTTP/3 protocol*. Nov. 2021. URL: <https://caniuse.com/http3> (besucht am 27.11.2021).
 - [12] Alexis Deveria. *JSON parsing*. Dez. 2021. URL: <https://caniuse.com/json> (besucht am 11.12.2021).
 - [13] Alexis Deveria. *Web Sockets*. Dez. 2021. URL: <https://caniuse.com/websockets> (besucht am 17.12.2021).
 - [14] European Parliament and Council. *Regulation (EU) 2018/1725*. IEC/TS 62443-1-1. Version 1.0. 21. Okt. 2018. URL: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32018R1725> (besucht am 11.01.2022).
 - [15] Roy Thomas Fielding. „Architectural Styles and the Design of Network-based Software Architectures“. Diss. University of California, 2000. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.
 - [16] Robin Garbe u. a. *Mojito, Caipirinha, Sex on the Beach, Die Rezepte des BAR-Robots müssen auf eine Plattform*. Qualitätssicherungsdokument. DiK, 31. März 2019.
 - [17] Laurence Goasdouf. *Gartner Says Global Government IoT Revenue for Endpoint Electronics and Communications to Total \$21 Billion in 2022*. Gartner. 30. Juni 2021. URL: <https://www.gartner.com/en/newsroom/press-releases/2021-06-30-gartner-global-government-iot-revenue-for-endpoint-electronics-and-communications-to-total-us-dollars-21-billion-in-2022> (besucht am 23.12.2021).
 - [18] Mara Gonzalez. *Manufacturing Industry has Been Slow to Adopt New Technology*. MavenEcommerce. 5. Juni 2018. URL: <https://fullstack.net/manufacturing-industry-has-been-slow-to-adopt-new-technology/> (besucht am 13.01.2022).

- [19] Piyush Gupta. „Modularity enablers: a tool for Industry 4.0“. In: *Life Cycle Reliability and Safety Engineering* 8.2 (Juni 2019), S. 157–163. ISSN: 2520-1360. doi: 10.1007/s41872-018-0067-3. URL: <https://doi.org/10.1007/s41872-018-0067-3>.
- [20] Mario Hermann, Tobias Pentek und Boris Otto. „Design Principles for Industrie 4.0 Scenarios“. In: *2016 49th Hawaii International Conference on System Sciences (HICSS)*. 2016, S. 3928–3937. doi: 10.1109/HICSS.2016.488.
- [21] i-SCOOP. *Industry 4.0 and the fourth industrial revolution explained*. 6. Jan. 2022. URL: <https://www.i-scoop.eu/industry-4-0/> (besucht am 07.01.2022).
- [22] Inray. *Der Weg von Industrie 1.0 zu Industrie 4.0*. 6. Aug. 2021. URL: <https://www.inray.de/aktuelles/der-weg-von-industrie-1-0-nach-industrie-4-0> (besucht am 02.01.2022).
- [23] inray Industriesoftware GmbH. *What is OPC UA? A practical introduction*. 2022. URL: <https://www.opc-router.com/what-is-opc-ua/> (besucht am 08.01.2022).
- [24] Internet Engineering Task Force. *Introducing JSON*. 5. Jan. 2022. URL: <https://www.json.org/json-en.html> (besucht am 07.01.2022).
- [25] Internet Security Research Group. *Let's Encrypt Stats*. Nov. 2021. URL: <https://letsencrypt.org/stats/> (besucht am 27.11.2021).
- [26] Alexander Kern und Reiner Anderl. „Attribute-based network and system access control architecture for industrial machines“. en. In: (22. Okt. 2019). Event Title: 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IoTSMS19). Granada, Spain: IEEE, Nov. 2019. ISBN: 978-1-7281-2949-5. doi: <https://doi.org/10.1109/IOTSMS48152.2019.8939227>. URL: <http://tubiblio.ulb.tu-darmstadt.de/116467/>.
- [27] Johannes Kinzig. *RESTful API for Simatic S7-1200 PLC & Python Client (Part 1)*. 26. März 2018. URL: <https://johanneskinzig.de/index.php/software-development/18-restful-api-for-simatic-s7-1200-plc-python-client> (besucht am 12.01.2022).
- [28] Stefan Kugler, Alexander Christ und Reiner Anderl. „Der digitale Zwilling - Ein Zwischenbericht“. de. In: (5. Juli 2017). Event Title: SIEMENS PLM Connection Deutschland 2017. Seeheim - Jugenheim, Juli 2017. URL: <http://tubiblio.ulb.tu-darmstadt.de/113759/>.

-
-
- [29] Paul J. Leach, Rich Salz und Michael H. Mealling. *A Universally Unique IDentifier (UUID) URN Namespace*. RFC 4122. Internet Engineering Task Force, Juli 2005. doi: 10.17487/RFC4122. URL: <https://rfc-editor.org/rfc/rfc4122.txt>.
 - [30] Mariaa Marques u. a. „Decentralized decision support for intelligent manufacturing in Industry 4.0“. In: *Journal of Ambient Intelligence and Smart Environments* 9.3 (2017), S. 299–313. doi: 10.3233/AIS-170436.
 - [31] Fabio Mauri und Alice Bourrouet. *01. Browser support - WEBGUIDE - EC Public Wiki*. 4. Feb. 2021. URL: <https://wikis.ec.europa.eu/display/WEBGUIDE/01.+Browser+support> (besucht am 10.01.2022).
 - [32] Fabio Mauri und Alice Bourrouet. *01. Data Protection - WEBGUIDE - EC Public Wiki*. 7. Mai 2021. URL: <https://wikis.ec.europa.eu/display/WEBGUIDE/01.+Data+Protection> (besucht am 10.01.2022).
 - [33] Fabio Mauri und Alice Bourrouet. *Europa Web Guide*. 22. Okt. 2021. URL: <https://wikis.ec.europa.eu/display/WEBGUIDE/Europa+Web+Guide> (besucht am 10.01.2022).
 - [34] Alexey Melnikov und Ian Fette. *The WebSocket Protocol*. RFC 6455. Internet Engineering Task Force, Dez. 2011. doi: 10.17487/RFC6455. URL: <https://rfc-editor.org/rfc/rfc6455.txt>.
 - [35] Samuel K. Moore. „Bespoke processors: A new path to cheap chips“. In: *IEEE Spectrum* 54.8 (2017), S. 11–12. doi: 10.1109/MSPEC.2017.8000275.
 - [36] Mozilla Corporation. *An overview of HTTP*. 23. Nov. 2021. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview> (besucht am 27.11.2021).
 - [37] Andrzej Ożadowicz u. a. „Application of the Internet of Things (IoT) Technology in Consumer Electronics - Case Study“. In: *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*. Bd. 1. 2018, S. 1037–1042. doi: 10.1109/ETFA.2018.8502468.
 - [38] Q-Success. *Usage statistics of HTTP/2 for websites*. Nov. 2021. URL: <https://w3techs.com/technologies/details/ce-http2> (besucht am 27.11.2021).
 - [39] Q-Success. *Usage statistics of HTTP/3 for websites*. Nov. 2021. URL: <https://w3techs.com/technologies/details/ce-http3> (besucht am 27.11.2021).

-
-
- [40] Jacqueline Zonichenn Reis und Rodrigo Franco Gonçalves. „The Role of Internet of Services (IoS) on Industry 4.0 Through the Service Oriented Architecture (SOA)“. In: *Advances in Production Management Systems. Smart Manufacturing for Industry 4.0*. Hrsg. von Ilkyeong Moon u. a. Cham: Springer International Publishing, 2018, S. 20–26.
 - [41] Eric Rescorla. *HTTP Over TLS*. RFC 2818. Internet Engineering Task Force, Mai 2000. doi: 10.17487/RFC2818. url: <https://rfc-editor.org/rfc/rfc2818.txt>.
 - [42] Leonard Richardson u. a. *RESTful Web APIs: Services for a Changing World*. 1. Aufl. O'Reilly Media, Inc., 2013.
 - [43] John S. Rinaldi. *Why Use OPC UA Instead of RESTful Interface*. Real Time Automation. 25. Mai 2019. url: <https://www.rtautomation.com/rtas-blog/why-use-opc-ua-instead-of-restful-interface/> (besucht am 21.12.2021).
 - [44] Karen Roby. *UPS: How IoT devices are transforming supply chain logistics*. TechnologyAdvice. 11. Mai 2020. url: <https://www.techrepublic.com/article/ups-how-iot-devices-are-transforming-supply-chain-logistics/> (besucht am 22.11.2021).
 - [45] Sean Ryan. *What is a UUID?* mParticle. 12. Juli 2021. url: <https://www.mparticle.com/blog/what-is-a-uuid> (besucht am 18.12.2021).
 - [46] Rainer Schiekofer, Andreas Scholz und Michael Weyrich. *REST based OPC UA for the IIoT*. Siemens AG und University of Stuttgart, 2018. url: https://www.ias.uni-stuttgart.de/dokumente/publikationen/2018_rest_based_opc_ua_for_the_iiot.pdf.
 - [47] Scratch Wiki. *User Interface - Scratch Wiki*. 18. Jan. 2022. url: https://en.scratch-wiki.info/wiki/User_Interface (besucht am 18.01.2022).
 - [48] Standardization Council Industrie 4.0. *Deutsche Normungsroadmap Industrie 4.0*. DIN und DKE, März 2020. url: <https://www.din.de/resource/blob/95954/fef3e0c46a3b5d042f25078c50547f0d/aktualisierte-roadmap-i40-data.pdf>.
 - [49] StatCounter. *Desktop Browser Market Share Worldwide*. Dez. 2021. url: <https://gs.statcounter.com/browser-market-share/desktop/worldwide/#monthly-202012-202112-bar> (besucht am 10.01.2022).

-
-
- [50] Erdal Tantik und Reiner Anderl. „Concept for Improved Automation of Distributed Systems with a Declarative Control based on OPC UA and REST“. In: *2019 7th International Conference on Control, Mechatronics and Automation (ICCMA)*. 2019, S. 260–265. doi: [10.1109/ICCMA46720.2019.8988777](https://doi.org/10.1109/ICCMA46720.2019.8988777).
 - [51] TC 65 - Industrial-process measurement, control and automation. *Industrial communication networks - Network and system security - Part 1-1: Terminology, concepts and models*. IEC/TS 62443-1-1. Technical Specification. Version 1.0. 30. Juli 2009. URL: <https://webstore.iec.ch/publication/7029> (besucht am 11.01.2022).
 - [52] TC 65/SC 65E. *IEC TR 62541-1:2020 RLV*. Edition 3.0. OPC Foundation, 18. Nov. 2020. URL: <https://webstore.iec.ch/publication/68039>.
 - [53] Lionel Sujay Vailshery. *Global IoT end-user spending worldwide 2017-2025*. Statista. 22. Jan. 2021. URL: <https://www.statista.com/statistics/976313/global-iot-market-size/> (besucht am 23.12.2021).
 - [54] Matthew Wopata. *Industry 4.0 Adoption 2020 – who is ahead?* IoT Analytics GmbH, 4. Feb. 2020. URL: <https://iot-analytics.com/industry-4-0-adoption-2020-who-is-ahead/> (besucht am 13.01.2022).

A. Anhang

```
1 <xml xmlns="https://developers.google.com/blockly/xml">
2   <block type="PLC_Distribute_ExtendSlide__ON" id="w[ , ^t0V?1A2pYgamWghr"
3     x="15" y="88">
4     <next>
5       <block type="instruction_wait" id="FFAUMYks/WAmp_I=ASlj">
6         <field name="wait_amount">0.2</field>
7         <next>
8           <block type="PLC_Distribute_ExtendSlide__OFF" id=". [Hx1gsNm1`I_?Epql~0">
9             <next>
10            <block type="PLC_Distribute_ConveyerForward__ON"
11              " id="6k8D(GT3vcFL96uzt3-+)">
12              <next>
13                <block type="instruction_wait" id="qi(*Iu1XH*3u9LH3,j`z">
14                  <field name="wait_amount">4.3</
15                    field>
16                    <next>
17                      <block type="PLC_Distribute_ConveyerForwa
18                        rd__OFF" id="1jdIM*0Nm(ulS29B3{LF"/>
19                      </next>
20                    </block>
21                    </next>
22                  </block>
23                  </next>
24    </block>
```

25 </xml>

Listing A.1: XML Code für simplen Prozess

```
1 const allUsedActuatorsRaw = [...$(xml).find('block')]
2     .map((el) => dic(el.getAttribute("type")))
3     .filter((el) => el !== "instruction_wait");
4 const usedActuators = [...new Set([...$(xml).find('block')])
5     .map((el) => dic(el.getAttribute("type")).substring(0, el.getAttribute("type").indexOf("__"))))
6     .filter((el) => el !== ""));
7
8 const mbomAnalysis = {
9     executionTime: Math.round([...$(xml).find('field[name="wait_amount"]')])
10        .map((el) => Number(el.textContent))
11        .reduce((sum, step) => sum + step)*100)/100,
12     usedActuators: usedActuators,
13     usedResources: [
14         {
15             name: "cup",
16             amount: allUsedActuatorsRaw.filter((el) => el === "
17                 PLC_Distribute_ExtendSlide__ON").length
18         },
19         {
20             name: "lid",
21             amount: allUsedActuatorsRaw.filter((el) => el === "
22                 PLC_Distribute_Vacuum__ON").length
23         }
24     ],
25     nbrOfGroups: $(xml).children('block').length,
26     nbrOfBlocks: $(xml).find('block').length
27 };
```

Listing A.2: Code zur Erstellung des Prozess-Randdaten-Objektes

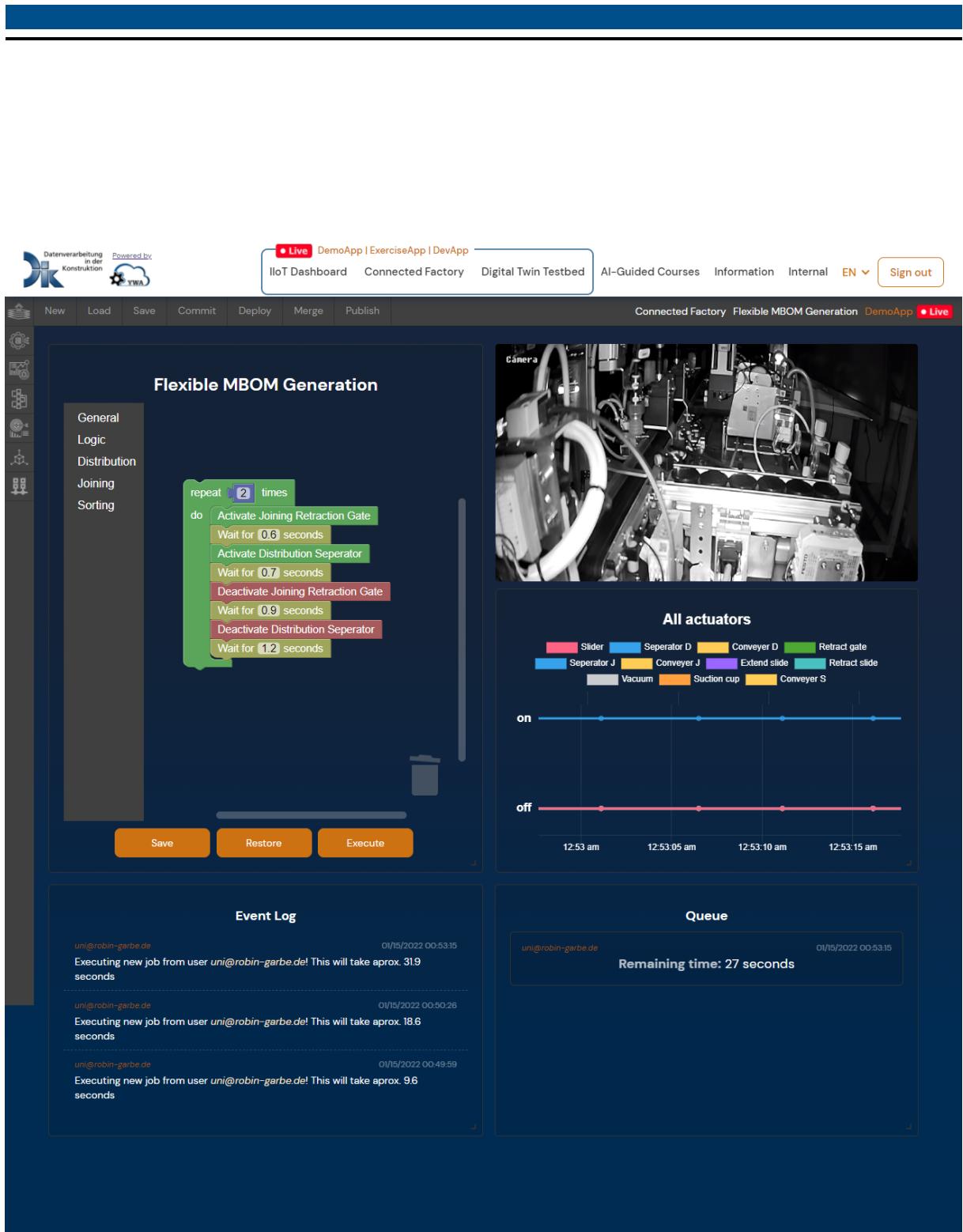


Abbildung A.1.: Ganzseitiger Screenshot der Prozessplanungs und -steuerungs Seite

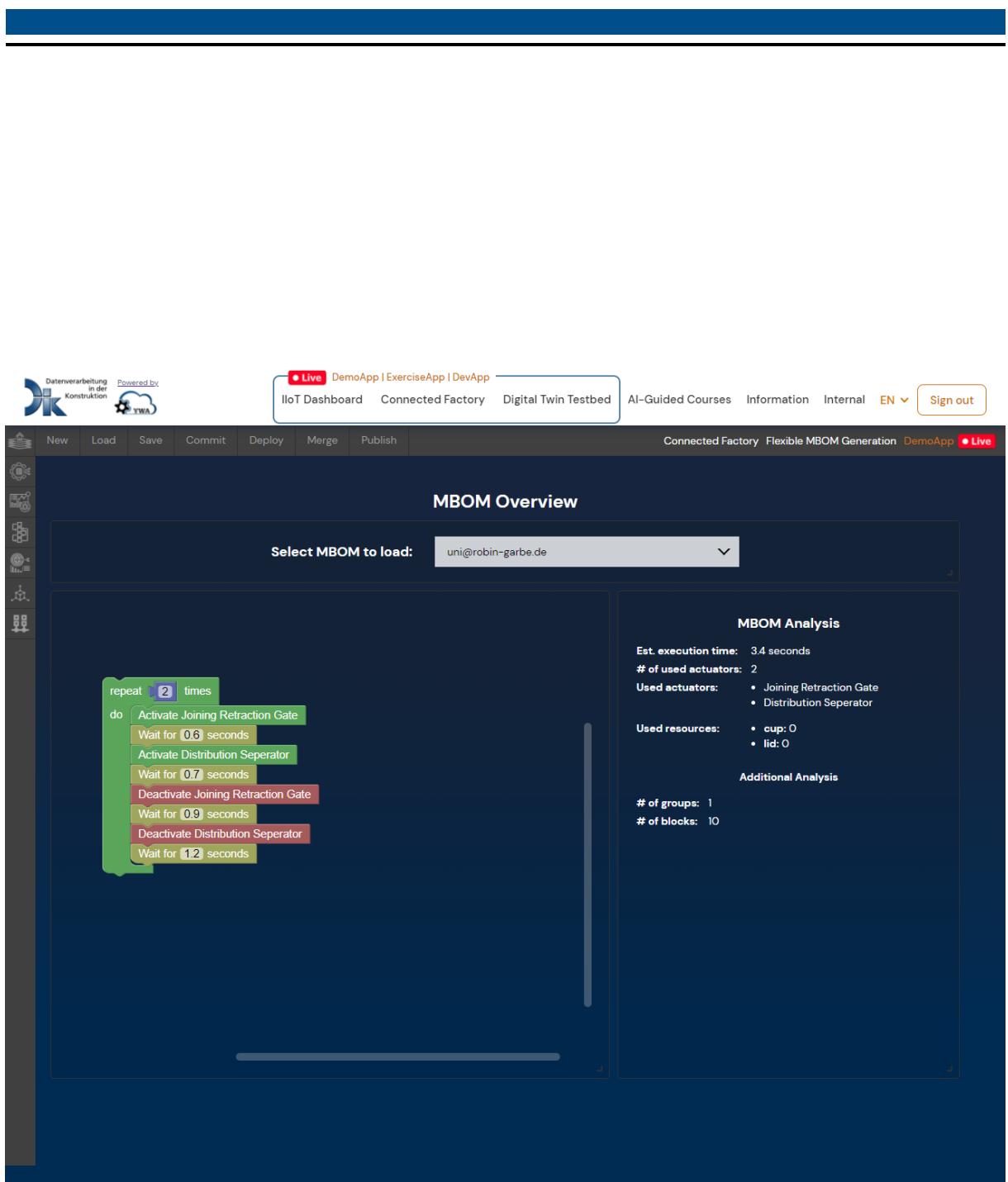


Abbildung A.2.: Ganzseitiger Screenshot der Übersichts-Seite

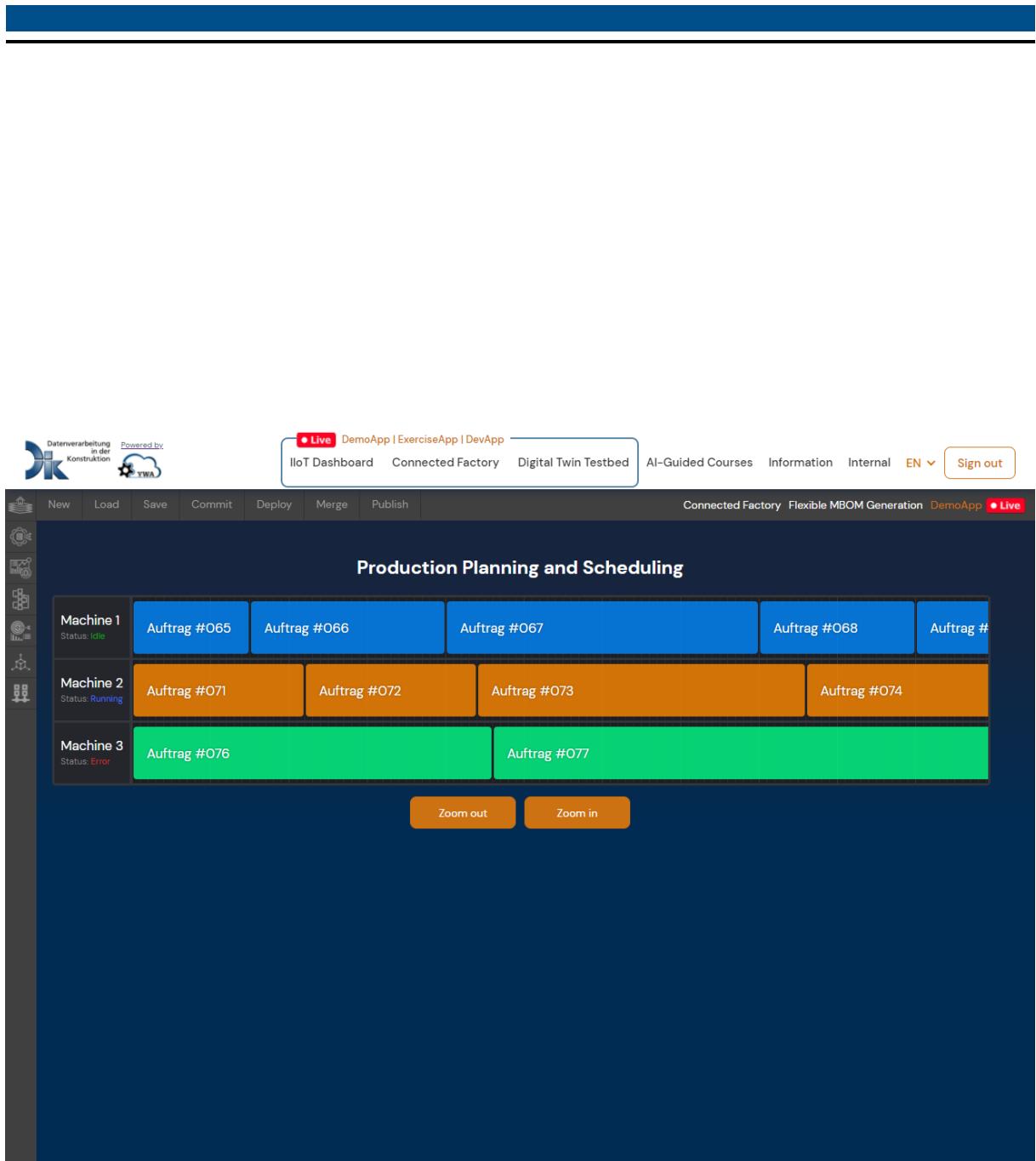


Abbildung A.3.: Ganzseitiger Screenshot der Produktionsplanungs-Seite