Universität Stuttgart
Institut für Steuerungstechnik
der Werkzeugmaschinen und
Fertigungseinrichtungen (ISW)

Studienarbeit

Konzeption und Implementierung einer Diagnoseschnittstelle zwischen Industrieroboter und Anlagensteuerung

eingereicht von

Lukas Schlotter

aus Stuttgart

Studiengang	M. Sc. Mechatronik
Prüfer	Prof. Dr.-Ing. Alexander Verl
Betreuer	Dr.-Ing. Andreas Wolf, Dipl.-Ing. Daniel Knauss
Eingereicht am	30. Dezember 2023

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Motivation . . . 1
- 1.2 Anforderungsdefinition / Aufgabenstellung . . . 1
- 1.3 Methodik und Vorgehensweise . . . 1

2 Grundlagen und Stand der Technik **2**
- 2.1 Feldbusse . . . 2
  - 2.1.1 Standard-Feldbusse . . . 3
  - 2.1.2 Ethernet basierende Feldbusse . . . 4
- 2.2 TCP/IP . . . 7
- 2.3 OPC/UA . . . 10
- 2.4 Stäubli-Roboter . . . 11
  - 2.4.1 Controller CS9 . . . 12
  - 2.4.2 Programmiersprache VAL3 . . . 14
- 2.5 Anlagensteuerung . . . 16
- 2.6 WPF . . . 18
- 2.7 Bewertung . . . 18

3 Anforderungsdefinition **19**

4 Konzeptionierung und Systementwurf **21**
- 4.1 Feldbus-Verbindung . . . 21
- 4.2 TCP/IP-Verbindung . . . 23
- 4.3 Datenverwertung . . . 25
- 4.4 WPF und GUI . . . 30

5 Design und Implementierung **31**
- 5.1 Stäubli-Roboter in VAL3 . . . 31
  - 5.1.1 EtherCAT . . . 31
  - 5.1.2 TCP/IP . . . 31
- 5.2 .NET in C# . . . 32
  - 5.2.1 TCP/IP . . . 32
  - 5.2.2 Datenverwertung und Visualisierung . . . 32

6 Validierung **33**

7 Ausblick und Fazit **34**

Abbildungsverzeichnis **35**

Tabellenverzeichnis **36**

Literatur **37**

1 Einleitung

1.1 Motivation

Col1	Col2	Col2	Col3
1	6	87837	787
2	7	78	5415
3	545	778	7507
4	545	18744	7560
5	88	788	6344

Tabelle 1.1: Table to test captions and labels.

1.2 Anforderungsdefinition / Aufgabenstellung

1.3 Methodik und Vorgehensweise

Die Verwendung geeigneter Methoden und eines strukturierten Vorgehens sind entscheidend für eine erfolgreiche Umsetzung des Projektes. Durch ein Vorgehensmodell wird eine effiziente Arbeitsweise und eine zeitgerechte Fertigstellung gefördert. Da die Komplexität des Projektes und der Software im Projektverlauf zunehmend wächst, ist eine wissenschaftliche Methodik essentiell. Methoden und Werkzeuge erleichtern es neue Lösungsansätze zu finden und faktenbasierte Entscheidungen zu treffen. [1] [2]
Gant-Diagramm wie Seite 39 ISW-Leitfaden

2 Grundlagen und Stand der Technik

2.1 Feldbusse

Maschinen und Anlagen verfügen über eine Vielzahl an Sensoren und Aktoren. Diese werden klassisch über Parallelverdrahtung an die Ein-und Ausgänge der SPS angeschlossen. Ein Feldbus ermöglicht es dagegen die Sensoren und Aktoren an einzelne aktive Verteilerboxen anzuschließen. Die Verteilerboxen werden wiederum mit einem Feldbus untereinander und mit der SPS verbunden. Dies ermöglicht eine Dezentralisierung, was den Schaltschrank vereinfacht, die Energieeffizient steigert und die Flexibilität bezüglich Änderungen erhöht. Der Übergang von der Parallelverdrahtung zu der Fehlbusverdrahtung ist mit den Zwischenschritt von passiven Verteilerboxen nachfolgend dargestellt (vgl. Abbildung 2.1). Bei der Feldbus Anschaltung wird meist wie in der Abbildung dargestellt eine Linien-Struktur verwendet. Alternativen sind die Stern-, Ring- oder Baum-Struktur. [3]

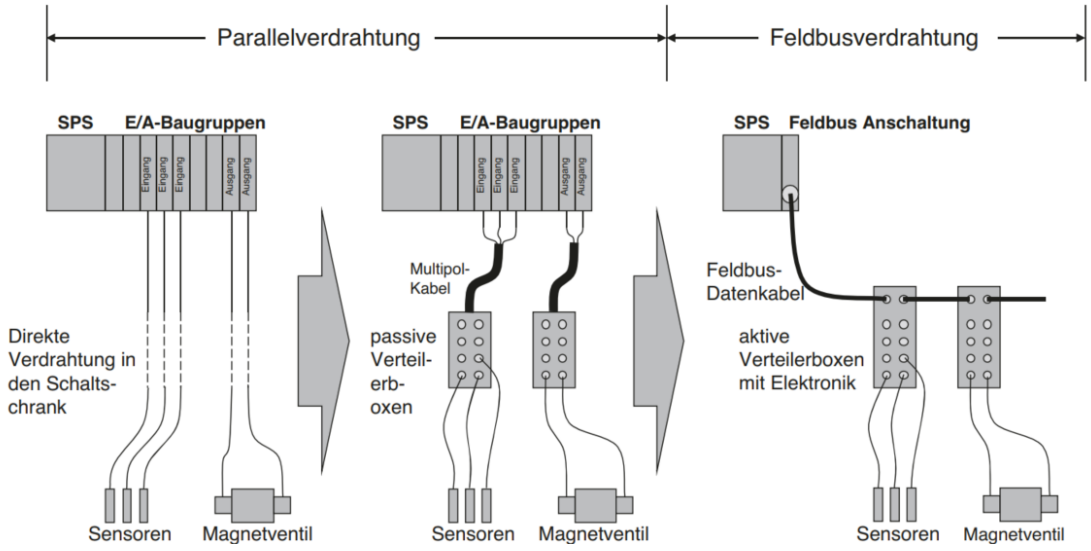


Abbildung 2.1: Übergang Parallelverdrahtung zu Feldbussen [3]

Es haben sich Feldbusse von vielen namhaften Herstellern etabliert. Mittlerweile werden diese Standard-Feldbusse immer weiter von Ethernet basierenden Feldbussen (auch Industrial Ethernet genannt) abgelöst, da diese vor allem Vorteile bezüglich der Übertragungsgeschwindigkeit aufweisen. Die Marktaufteilung aus dem Jahr 2021 ist nachfolgend

abgebildet, wobei eine weitere Zunahme der Marktanteile von den Ethernet basierenden Feldbussen zu erwarten ist (vgl. Abbildung 2.2). [3] [4]

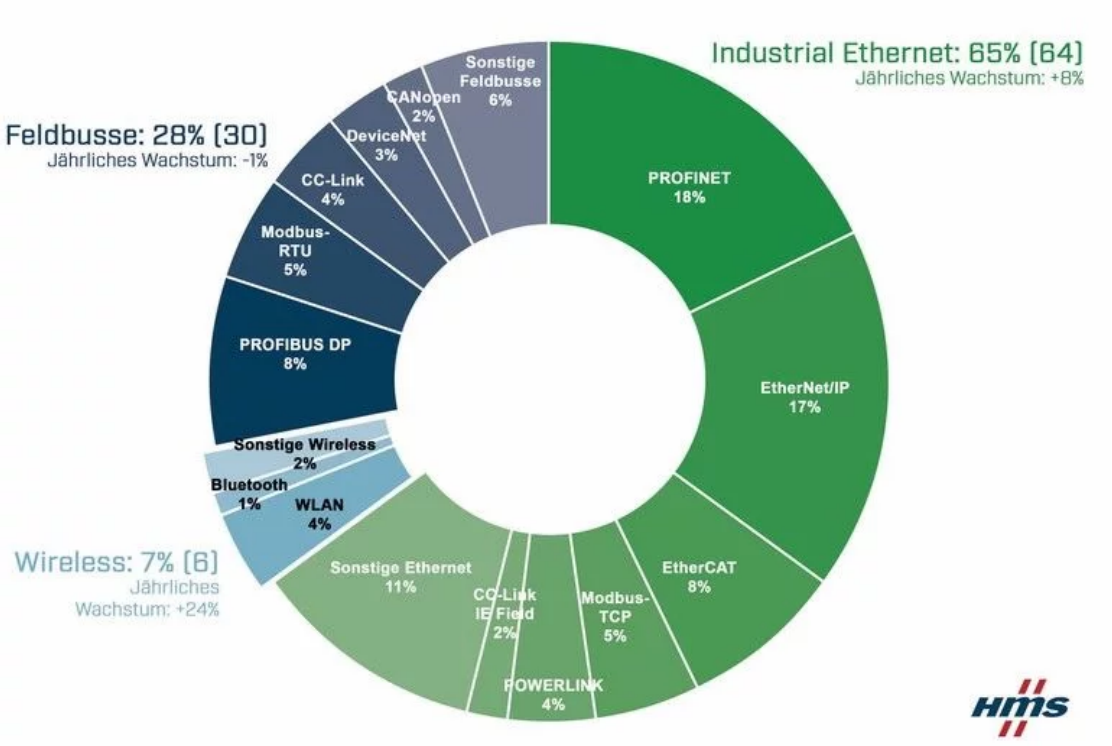


Abbildung 2.2: Marktanteile Feldbusse und Industrial Ethernet [4]

2.1.1 Standard-Feldbusse

In der Anlagentechnik und der Robotik ist die sofortige zuverlässige Datenübertragung unter anderem aus Gründen der Sicherheit essentiell. Feldbusse sind daher echtzeitfähig. Die Echtzeitfähigkeit besagt, dass die Daten innerhalb einer sehr kurzen festgelegten Zeitspanne unabhängig von äußeren Einflüssen übertragen werden müssen. Durch die harte Echtzeit, ist im Gegensatz zur weichen Echtzeit zusätzlich definiert, dass die Daten mit einem absoluten Determinismus übertragen werden müssen. Bei der weichen Echtzeit verlieren die Daten lediglich an Nutzen bei verspätetem Eintreffen, bei der harten Echtzeit werden sie komplett nutzlos. Aus diesem Grund müssen bei der harten Echtzeit 100 % der Daten innerhalb der definierten Zeit übertragen werden, was insbesondere bei sicherheitsrelevanten Funktionen von Relevanz ist. [5] [6]
Im folgenden sollen ein paar gängige Feldbusse kurz erwähnt und erläutert werden.

Profibus wird von dem Dachverband "Profibus &ProfinetÏnternational"verwaltet, um die Interessen der Nutzer einzubringen. Neben dem Profibus auf Feldbus-Ebene (auch Profibus-DP genannt) existiert eine weitere Variante auf Zellensteuerungs-Ebene und eine auf Prozess-Automatisierungs-Ebene. Als Übertragungstechnik dient der sogenannte RS485-Standard mit einer Zweidrahtleitung. Die Datentransferrate von Profibus beträgt bis zu 12 MBit/s. Speicherprogrammierbare Steuerungen von Siemens setzen häufig Profibus ein. [3]
CAN-Bus wurde ursprünglich für die Automobil-Industrie entwickelt und kommt dort heute noch zum Einsatz. Darüber hinaus, nahm die Verbreitung in der Anlagensteuerung zu. Möglich sind Datentransferraten von bis zu 1 MBit/s. Als Übertragungsmedium dient eine Zweidrahtleitung. CAN-Bus zeichnet sich durch eine besonders hohe Datensicherheit, also eine hohe Zuverlässigkeit bei der Datenübertragung aus, weshalb er in der Medizintechnik und Robotik hohen Zuspruch findet. [3]
DeviceNet basiert auf CAN und ist eine Entwicklung des nordamerikanischen Herstellers "Rockwell Automation". Die Datenübertragungsrate beträgt bis zu 500 kBit/s und die Spannungsversorgung, sowie die Datenkommunikation können über ein Kabel erfolgen. [3]
CC-Link stellt eine Entwicklung des Unternehmen Mitsubishi dar. Verwaltet wird das Protokoll von einer Anwenderorganisation ähnlich zu Profibus. Die maximale Übertragungsrate beträgt 10 MBit/s und als Übertragungsmedium dient eine dreiadrige Leitung. [3]

2.1.2 Ethernet basierende Feldbusse

Ethernet basierende Feldbusse, häufig auch als Ïndustrial Ethernet"bezeichnet, ermöglichen deutlich höhere Übertragungsraten als Standard-Feldbusse. Daher nimmt deren Verbreitung ständig zu und lösen in vielen Bereichen den Standard-Feldbus ab. [3]
Ethernet-Technologie
Ethernet ist einer von mehreren Standards für die lokale Netzwerkverbindung mittels LAN-Technologie. Sie ist im IEE 802-3- Standard (Institute of Electrical and Electronis Engineers) genormt. Der Aufbau eines Ethernet-Telegramms ist nachfolgend abgebildet (vgl. Abbildung 2.3). Die Adressierung erfolgt mittels MAC-Adresse oder IP-Adresse (mehr hierzu in Kapitel 2.2). [7]
Verschiedene Geräte sind an das Übertragungsmedium angeschlossen und teilen sich dieses. Man spricht von einem Multi-Master-Bus, da jedes Gerät selbständig die Initative ergreifen kann, um Nachrichten zu senden. Damit jedes Gerät kommunizieren kann und dabei Sendekonflikte vermieden werden kommt bei Ethernet der CSMA/CD-Mechanismus zum Einsatz. CSMA/CD steht für "Carrier Sense Multiple Access with Collision Detectionünd ermöglicht einen mehrfachen Zugriff (Muliple Access) auf das Übertragungsmedium. [3] [7]

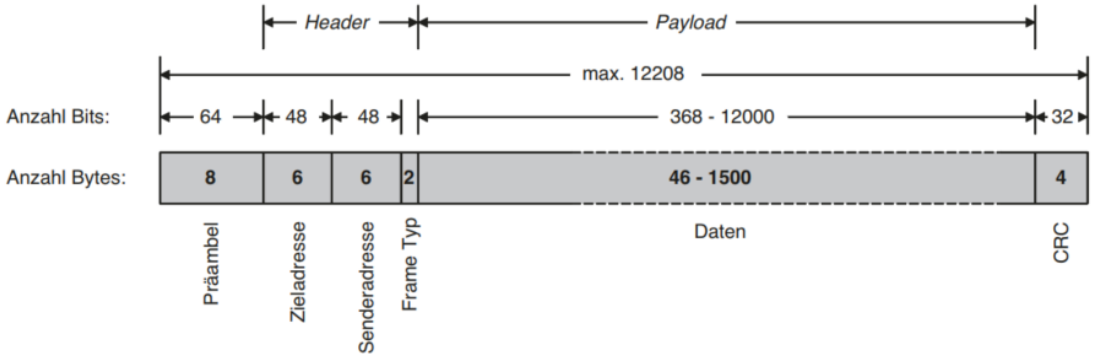


Abbildung 2.3: Telegramm-Aufbau Ethernet [3]

- **Carrier Sense:** Es überwacht, dass das Übertragungsmedium nicht durch ein anderes Gerät belegt ist. Erst nachdem das Medium als frei erkannt wird, erfolgt nach einer kurzen Wartezeit die Übertragung. Während des Sendens, wird das Medium weiterhin auf Kollisionen abgehört.
- **Multiple Access:** Es besagt, dass alle Geräte gleichberechtigt auf das Übertragungsmedium zugreifen können.
- **Collision Detection:** Wenn mehrere Geräte zur gleichen Zeit das Übertragungsmedium als frei erkennen kommt es zur Überlagerung, da mehrere Geräte gleichzeitig beginnen zu senden. Dies wird als Kollision bezeichnet und es kommt zur Signalverfälschung durch Überlagerung. Da jeder Sender das Übertragungsmedium während des Sendevorgangs überwacht wird die Kollision direkt erkannt. Dasjenige Gerät, welches die Kollision erkennt, informiert alle anderen Geräte mittels eines Signales über die aufgetretene Kollision und fordert diese zur Unterbrechung jeglicher Übertragung auf.
  Danach warten die Geräte eine zufällige Zeitspanne ab und versuchen das Senden erneut. [7]

Durch die Kollisionen kommt es zu nicht vorhersehbaren Wartezeiten im Sendeprozess. Daher handelt es sich um ein nicht deterministisches Zugriffsverfahren. Ethernet mit CSMA/CD ist nicht echtzeitfähig, was wie bereits in Kapitel 2.1.1 erwähnt, zu Problem bei Automatisierungsanlagen führen kann.
Um die Ethernet-Technologie dennoch in der Automatisierung zu nutzen, haben die Hersteller verschiedene Echtzeitprotokolle entwickelt, um Ethernet echtzeitfähig zu machen. Beispielsweise wird die Methodik CSMA/CD außer Kraft gesetzt und stattdessen durch sogenanntes Pooling oder ein Zeitscheibenverfahren ersetzt. Jeder Hersteller geht hier jedoch seinen eigenen Weg. Bekannte Ethernet basierende Feldbusse sind Profinet, Ethernet/IP, EtherCAT und Sercos III. EtherCAT ist eine Technologie des Herstellers "Beckhoff", dessen SPS im Rahmen dieses Projektes zum Einsatz kommt. Aus diesem

Grund ist EtherCAT die bevorzugte Technologie und wird im Rahmen dieser Arbeit näher erläutert. [8] [7]
Weiterhin ist zu beachten, dass die Netzwerktopologie sich bei den Ethernet basierenden Feldbussen häufig von den Standard-Feldbussen unterscheidet. Während bei den Standard-Feldbussen die Ansteuerung vorwiegend in der Linienstruktur erfolgt, wird bei der Ethernet Anschaltung oft eine Sternstruktur mit einem Switch eingesetzt (vgl. Abbildung 2.4).

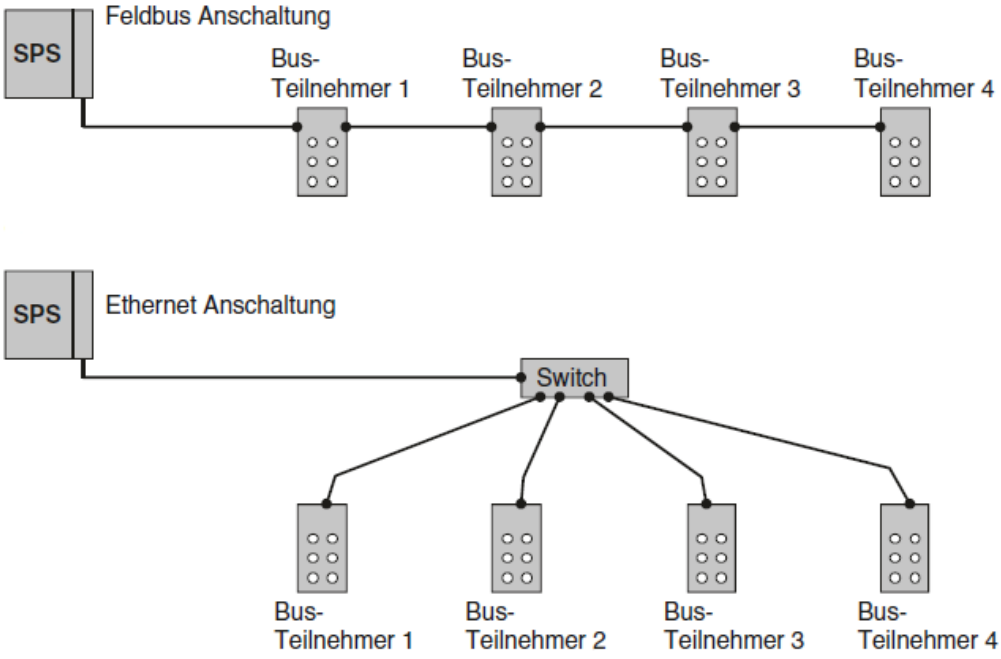


Abbildung 2.4: Topologie: oben Standard-Feldbus, unten Ethernet basierender Feldbus [3]

Der Ethernet Switching Hub, abgekürzt als Switch erlernt beim Einschalten an welchen Ports Teilnehmer angeschlossen sind. Der Switch leitet ein Datentelegramm eindeutig vom Sender zum Empfänger. Dadurch werden von dem Telegramm nicht betroffene Teilnehmer und Ethernet-Segmente nicht unnötig belastet und Kollisionen vermieden. Das Weiterschalten erfolgt auf Basis der Sende- und Ziel-Adresse, welche im Ethernet-Telegramm hinterlegt ist. [3]

EtherCAT

Wie bereits erwähnt setzen viele Hersteller auf das Zeitscheibenverfahren oder Pooling, um EtherNet echtzeitfähig zu machen. Der Zeitverzug ist hierbei jedoch implementierungsabhängig und kann durch erforderliche technische Komponenten am Bus weiter steigen. Um dies zu verhindern setzt EtherCAT auf einen anderen Lösungsansatz. Im Gegensatz zu anderen Verfahren sollen Daten nicht mehr empfangen, interpretiert und

dann wieder weiterversendet werden. Stattdessen setzt EtherCAT auf sogenannte FMMU (Fieldbus Memory Management Unit) in allen E/A-Klemmen. Diese FMMU entnimmt nur die relevanten Daten vom Bus, sodass die Telegramme mit einer Verzögerung von nur wenigen Nanosekunden weiterlaufen können. Zum Versenden werden Daten in die durchlaufenden Telegramme eingefügt, sodass es auch hier zu kaum einer Verzögerung kommt. Dieses Prinzip ist nachfolgend dargestellt (vgl. Abbildung 2.5).

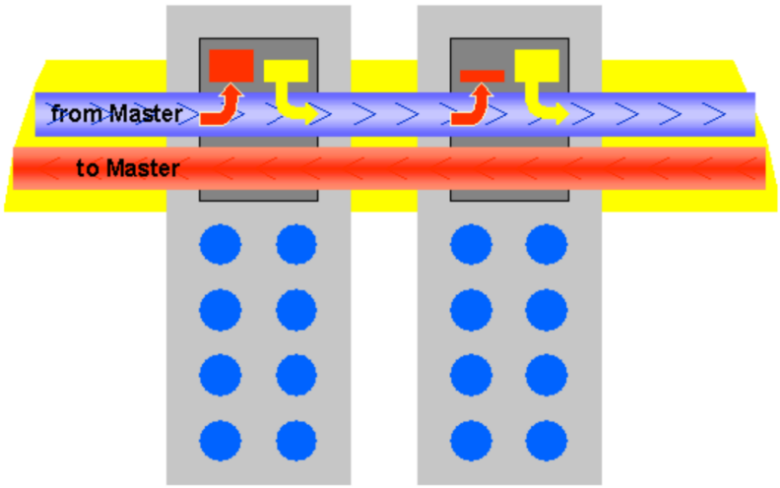


Abbildung 2.5: Telegrammbearbeitung [8]

Dabei wird auf dem gesamten Bus vom Ethernet-Protokoll nach IEEE 802.3 nicht abgewichen. Bit-Fehler werden so durch die CRC-Prüfsumme erkannt. Neben der Stern-Topologie unterstützt EtherCAT auch eine Linien- oder Baum-Struktur und viele weitere. EtherCAT zeichnet sich durch geringe Laufzeitverzögerungen und eine hohe Datenrate aus. Beispielsweise können 1000 E/as in nur 30 μs aktualisiert werden und auch eine Übertragungsrate GBit-Bereich ist mit Erweiterung möglich. [8]
Multi-Master Bussen (z.B. CAN oder TCP/IP) vs. Mono-Master

2.2 TCP/IP

TCP/IP (Transmission Control Protocol/Internet Protocol) ist das meist verwendete Netzwerkprotokoll weltweit und zudem frei zugänglich. Durch dieses Protokoll wird definiert, wie Daten durch Netzwerkkommunikationshardware versendet und empfangen werden kann. Welches Übertragungsmedium verwendet wird ist nicht definiert, sodass sich neben LAN z.B. auch WLAN einsetzen lässt.[9] [10]
TCP/IP ist nicht echtzeitfähig, stellt jedoch eine gute Ergänzung zu den Echtzeitprotokollen dar, um nicht zeitkritische Daten, wie z.B. zur Diagnose oder Visualisierung zu übertragen. Bei TCP/IP handelt es sich um eine sichere Datenübertragung die Nachrich-

ten in einen Bytestrom verpackt und entpackt. Vom Sender zum Empfänger durchlaufen die Daten vier Schichten (vgl. Abbildung 2.6). Für eine sichere fehlerfreie Übertragung werden die eigentlichen Daten ergänzt um die MAC-/IP- und TCP-Header. Das CRC-Feld stellt eine Art Prüfsumme, mit welcher die Korrektheit der übertragenen Daten überprüft wird. [3]

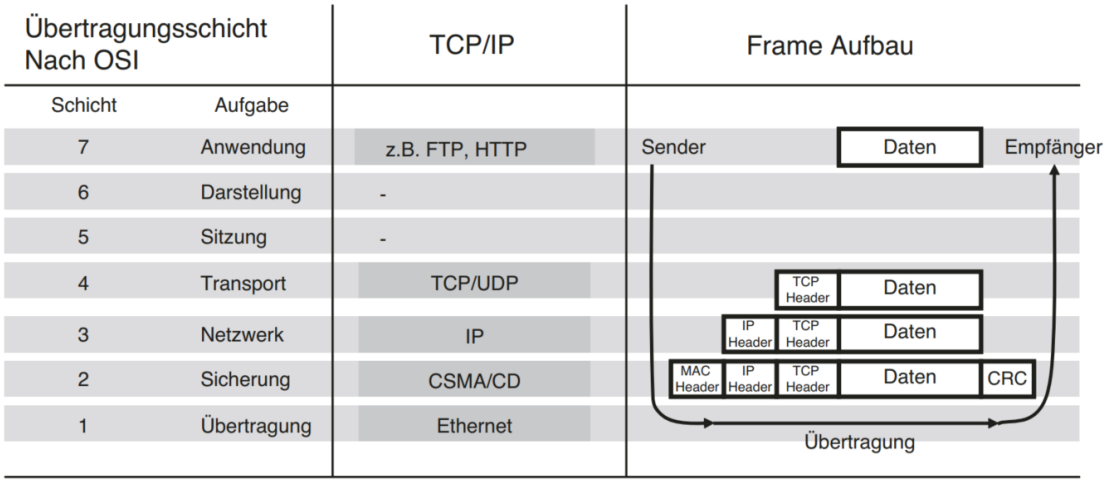


Abbildung 2.6: Schichtenmodell TCP/IP [3]

TCP/IP setzt sich aus den Protokollen TCP und IP zusammen.

TCP
Das Ziel von TCP ist eine fehlerfreie Datenübertragung. Hierzu muss der Empfänger die korrekt erhaltenen Daten, über eine Nachricht, die an den Sender zugesendet wird, bestätigen. Die Überprüfung der Korrektheit erfolgt mit dem CRC-Segment. Erhält dieser Sender diese Bestätigung nicht wird erneut versucht die Daten zu versenden. Hierdurch ist eine fehlerfreie und lückenlose Datenübertragung, selbst bei Netzwerkproblemen garantiert, was jedoch die Prozesse verlangsamt. Eine Alternative zu TCP ist UDP (User Datagram Protocol) . Hierbei erhält der Absender keine Bestätigung, dass die Daten korrekt empfangen wurde. Der Sender fährt direkt mit der Versendung der nächsten Pakete durch. Eine fehlerfreie Übertragung ist nicht garantiert, jedoch ist sie im Vergleich zu TCP schneller. Sowohl TCP, als auch UDP bauen auf dem Internetprotokoll IP auf. [9]

IP
Das IP-Protokoll arbeitet auf der Internet bzw. IP-Schicht des TCP/IP-Protokolls, was der Netzwerkschicht des OSI-Modells entspricht. Die Ausgabe des IP-Protokolls ist es Datenpakete vom Sender zum Empfänger zu übertragen. Eine Datenüberprüfung und Fehlerkorrektur erfolgt nicht, sodass Daten verloren gehen können oder fehlerbehaftet sind. Die Garantie für die korrekte Datenlieferung geschieht durch das TCP-Protokoll.

Das IP-Protokoll verwendet IP-Adressen, um Netzwerkknoten zu identifizieren. Mit Hilfe derer Adressen wird die Nachricht durch das Netz geroutet. Es wird ein idealer Weg zwischen Sender und Empfänger gesucht. [11] [7] Es gibt zwei Versionen des IP-Protokolls IPv4 und IPv6. Bei IPv4 besteht die IP-Adresse aus 32 Bit, bei IPv6 aus 128 Bit, was die Anzahl der eindeutigen Adressen erhöht. Aufgrund der zunehmenden Geräteanzahl wird in der Zukunft IPv6 der Standard werden. Heute dominiert jedoch IPv4. [7]
Eine MAC-Adresse (Medium Access Control) ermöglicht eine eindeutige Identifikation eines Gerätes im Netzwerk. Da allerdings meist herstellerübergreifende Netzwerke eingesetzt werden, ist die IP-Adresse für eine eindeutige Identifikation besser geeignet. Diese kann entweder manuell oder automatisch zugewiesen werden. Der Aufbau einer IP-Adresse nach IPv4 ist nachfolgend dargestellt (vgl. Abbildung 2.7). [3]

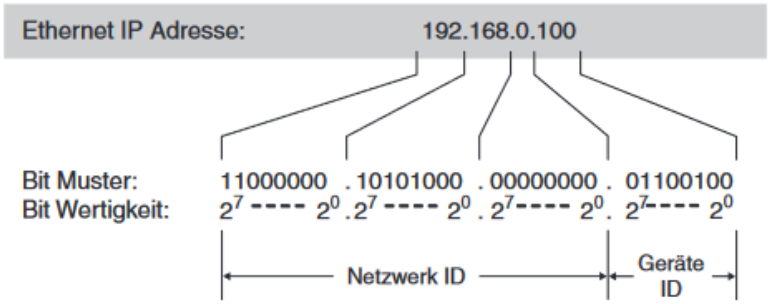


Abbildung 2.7: IP-Adresse Aufbau [3]

Netzwerkarchitektur
Während Netzwerke in ihrer Topologie, wie z.B. Bus, oder Stern-Topologie unterschieden werden können, kann auch eine Aufteilung nach Architekturtyp erfolgen. Neben der monolithischen Architektur und der Client-Server-Architektur gibt es Cloud-, Edge- und Fog-Computing. In der monolithischen Architektur existiert nur ein zentraler Rechner. An diesen werden externe Geräte angebunden, die selbst keine Rechenleistung aufweisen. Bei Cloud-, Edge- und Fog-Computing wird ein Teil der Netzwerktechnik an externe Organisationen ausgelagert. Eine externe Organisation kann ein Provider sein, der ein Server bereitstellt.
Im nachfolgenden soll die Client-Server-Architektur näher erläutert werden, da diese von Relevanz für diese Arbeit ist. Bei der Client-Server-Architektur stellt der Client Anfragen an den Server, welche dieser beantwortet. Dieser Ablauf ist als Sequenzdiagramm in Abbildung 2.8 dargestellt.

Charakteristisch für den Server ist dessen vergleichsweise hohe Rechenleistung, was es ermöglicht, dass mehrere Clients zeitgleich auf den Server zugreifen können. Veranschaulichen lässt sich dieses Prinzip mit dem Webbrowser, wie beispielsweise Google

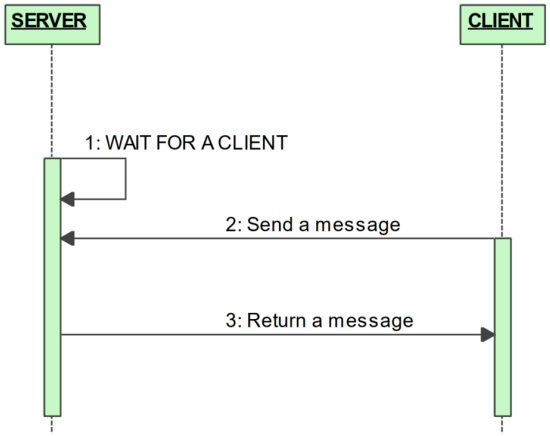


Abbildung 2.8: Sequenzdiagramm Server-Client-Architektur [9]

Chrome auf dem privaten Rechner. Dieser stellt den Client dar, welcher Anfragen an einen Webserver stellt, um aktuelle Daten passend zu seiner Anfrage zu erhalten. Auf dem Webserver wird beispielsweise ein Online-Shop verwaltet und mehrere Clients können zeitgleich auf diesen zugreifen. Das oben erwähnte Cloud-Computing basiert auf der Client-Server-Architektur, wurde jedoch nutzerfreundlicher gestaltet. [12]

2.3 OPC/UA

OPC UA (Open Platform Communication Unified Architecture) ist ein Protokoll zur Maschine-zu-Maschine-Kommunikation, aber auch zur Kommunikation zwischen SPS oder HMI (Human Interface) und Maschine. Es handelt sich hierbei um einen offenen und plattformunabhängigen Standard, welcher von der Open Platform Foundation (OPC) verwaltet wird. OPC UA basiert analog zu TCP/IP auf Ethernet und Internet. Es ist ebenso nicht echtzeitfähig. Das Kommunikationsprotokolls OPC UA ermöglicht eine vertikale und horizontale Kommunikation, da es die einzelnen Automatisierungsebenen verbindet. Somit kann von einem Roboter bis hin zur gesamten Fabriksteuerung alles verbunden werden, was OPC UA beliebt, um die Fabrik Ïndustrie 4.0"fähig zu machen. [13]
OPC UA basiert auf der Server-Client Architektur. Der OPC UA Server kann dann beispielsweise auf der SPS oder dem HMI (Human Interface) laufen und ein Roboter kann als OPC UA Client angebunden werden. Die Daten werden standardisiert und maschinenlesbar semantisch beschrieben, sodass herstellerübergreifende Kommunikation problemlos möglich ist. Sensordaten einer Maschine können dadurch an die Anlagensteuerung übertragen und visualisiert werden. OPC UA liefert dabei viele Vorteile: [14]

- **Sicher:** IT-Sicherheit wurde bei der Entwicklung von OPC UA direkt mit eingeplant. Die Kommunikation und der Transport ist sicher und kann über Zugriffsrechte gesteuert werden.
- **Plattformunabhängig:** Neben der Herstellerunabhängigkeit ist OPC UA unbhängig von der Programmiersprache, was es ermöglicht verschiedene Anlagen einfach zu verknüpfen.
- **Einfach:** OPC UA ist auf Nutzerfreundlichkeit ausgelegt und das Festlegen von Kommunikationsprotokollen und der Aufbau von Nachrichten entfällt, sodass auch ohne Programmierkenntnisse eine Umsetzung möglich ist.
- **Big Data:** OPC UA ermöglicht einfaches Sammeln von Daten aus diversen Maschinen und von unterschiedlichsten Automatisierungsebenen. Hierdurch können sich umfangreiche Potentiale durch die Analyse der Daten ergeben und neue Geschäftsmodelle erschlossen werden.

Aufgrund dieser unterschiedlichsten Vorteile, wird OPC UA immer mehr im Bereich IoT und Industrie 4.0 eingesetzt. [14] [13]

2.4 Stäubli-Roboter

Stäubli ist ein in der Schweiz ansässiges Industrieunternehmen, welches in den Bereichen Ëlektrische Verbindungen", "Fluidische Verbindungen","Textilünd "Robotik"tätig ist. Das Produktportfolio im Bereich Robotik umfasst Vier- und Sechsachs-Roboter, sowie kollaborative und mobile Roboter. [15]
Stäubli-Roboter sind nach IP 65 und IP 67 Staub- und Wasser- geschützt, wenn ein Arm mit Überdruckeinheit zum Einsatz kommt. Dies ermöglicht eine gründliche Reinigung aus Hygienezwecken und verhindert das Ansammeln von Staub im Roboter. Beispielsweise ist auch eine Reinigung mit Desinfektionsmittel oder Isopropylalkohol möglich. Zudem können die Roboter mit einem Lebensmittelverträglichem Öl eingesetzt werden. Stäubli-Roboter eignen sich daher insbesondere für Anwendungen in der Lebensmittel-, Pharma- und Medizin-Technik, in der viele andere Roboter die Hygieneanforderungen nicht erfüllen können. [16]
Roboter TX2 - 90
Im Rahmen dieser Arbeit kommt der 6 Achs-Roboter TX2-90 zum Einsatz (vgl. Abbildung 2.12). Dieser weist eine Tragkraft von 12 kg und eine Reichweite von 1000 mm auf. Die Wiederholgenauigkeit beträgt 0,03 mm und die maximale kartesische Geschwindigkeit 10,9 m/s. Der Roboter wiegt 114 kg und wird von der Robotersteuerung CS9 mit 2kVA gesteuert. Der Roboter ist für verschiedene, teils auch aggressive Reinigungsmittel geeignet, was für die Lebensmittelindustrie wichtig ist. Die Achsen können maximale Drehmomente zwischen 11 Nm (Achse 6) und 318 Nm (Achse 1) aufbringen.

Der Vorderarm des Roboters ist mit vier Magnetventilausgängen, von zwei 5/2-Wege-Magnetventilen versehen. Dies ermöglicht beispielsweise das Öffnen und Schließen eines Greifers.
Für den Roboter liegt ein Wartungsplan vor, welcher Wartungsarbeiten in unterschiedlichen Intervallen von monatlich bis 5-jährig vorsieht. Diese reichen von einfachen Sichtprüfungen und Bremstests über Ölwechsel und Dichtungstausch bis hin zum Auswechseln von Achsen. [17]

Abbildung 2.9: Stäubli TX2 - 90 [17]

2.4.1 Controller CS9

Der CS9 Controller übernimmt die Steuerung und Versorgung des TX2 - 90 (vgl. Abbildung 2.10). Der Computer-Einschub (1) beeinhaltet Karten, für die Sicherheitssteuerung, Bewegungssteuerung und die externe Kommunikation. Der Verstärker-Einschub (2) wandelt mit Hilfe von digitalen Achsverstärkern Bewegungsvorgaben in Motorströme um. Er beinhaltet zudem die Schnittstelle, an dem der Roboter angeschlossen wird. Der Stromversorgungs-Einschub (3) wandelt die Netzversorgungsspannung für die zuvor genannten Einschübe um und versorgt diese. Zudem existiert ein Anschluss für eine Antistatikmanschette (4), sowie Befestigungsmöglichkeiten (5). Der Computer-Einschub (vgl. Abbildung 2.11) besitzt eine Vielzahl an Kommunikationsschnittstellen, die in der Tabelle nachfolgend dargestellt sind (vgl. Tabelle 2.1).

Abbildung 2.10: Stäubli CS9 Controller [18]

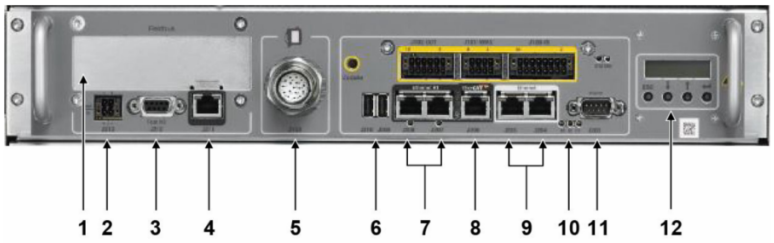


Abbildung 2.11: CS9 Controller Computer-Einschub [18]

Tabelle 2.1: CS9 Controller Computer-Einschub

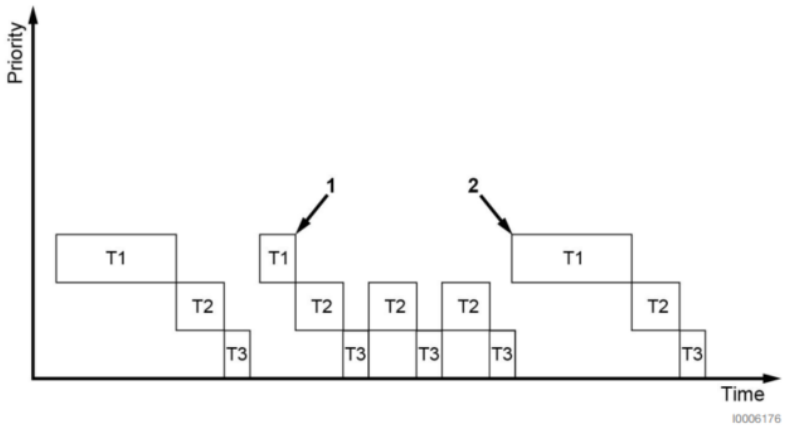
Nr.	Bezeichnung
1	Feldbus, Option für PCIe-Karte
2	Anschluss für optionale externe 24 V Stromversorgung
3	Schnelle Ein-/Ausgänge
4	EtherCAT Port (Reserviert)
5	Anschluss Handbediengerät
6	USB-Ports
7	Echtzeit-Ethernet-Slave (EtherCAT, Sercos III, EtherNet/IP, PROFINET
8	EtherCAT-Master
9	Ethernet-Ports (1000 MBits/s und 100 MBits/s)
10	Status LEDs
11	serielle Verbindung RS232
12	Benutzerschnittstelle

An den CS9 Controller kann kann ein Handbediengerät angeschlossen werden (5), was z.B. manuelles Verfahren oder einfachere Programmier- und Konfigurationsarbeiten ermöglicht. Hervorzuheben ist zudem der Echtzeit-Ethernet-Slave-Anschluss (7), durch welchen die Steuerung z.B. über EtherCAT mit einer SPS verbunden werden kann. Zusätzlich steht ein EtherCAT-Master-Anschluss (8) zur Verfügung. Die CS9 ermöglicht es somit den Roboter als EtherCAT-Slave oder als EtherCAT-Master zu betreiben.
Zudem stehen zwei Ethernet-Ports (9) zur Verfügung, was eine externe Kommunikation z.B. über TCP/IP ermöglicht. Die IP-Adresse der Ethernet-Ports kann frei konfiguriert oder automatisch vergeben werden. Als Dateiübertragungs-Serverprotokoll dient "FTPS+FTP". FTP wird als Standardeinstellung verwendet, während FTPS eine sichere Authentifizierung ermöglicht.
Auf der CS9-Steuerung lassen sich sogenannte Sockets konfigurieren, welche eine TCP/IP oder eine UDP/IP - Kommunikation ermöglichen. Die Konfiguration kann dabei im Client oder Server-Modus erfolgen. [18]

2.4.2 Programmiersprache VAL3

Die Programmierung des Roboters kann auf dem Handbediengerät erfolgen oder bevorzugt in der Stäubli Robotic Suite (SRS). In der SRS kann neben der Programmierung eine Simulation des Roboters inklusive 3D-Darstellung erfolgen, sodass die Programme direkt getestet werden können.
"VAL 3ïst eine höhere Programmiersprache und besitzt spezielle Steuerungsfunktionen für den Roboter zur Bewegungsteuerung und zur Ansteuerung von Ein-/Ausgängen. [20]
Programm
In einer Applikation können mehrere Programme angelegt werden. Diese verhalten sich wie Funktionen und lassen sich aufrufen. Zur Ausführung der VAL3-Applikation wird standardmäßig das Prgoramm ßtart()"gestartet, welches vergleichbar mit einer "main-Funktionïst. Von hier lassen sich wiederum andere Programme aufrufen. Bei Beendigung der Applikation wird das Programm ßtop()ëinmalig aufgerufen, um z.B. von dort Tasks zu beenden.
Programme lassen sich auch als Task ausführen. Bei einer Task handelt es sich um ein Programm welches zu einem gewissen Zeitpunkt ausgeführt wird. Üblicherweise wird beispielsweise die Bewegungssteuerung und die Kommunikation in eigene Tasks ausgelagert. Es wird ein möglichst gleichzeitiges Ausführen der Tasks angestrebt. Da die Steuerung jedoch nur einen Prozess besitzt, ist es jedoch nur möglich eine Task auszuführen. Die augenscheinliche Parallität, wird dadurch erreicht, dass die Tasks sequentiell sehr schnell hintereinander ausgeführt werden. Dabei wird nicht die gesamte Task abgearbeitet, sondern einige wenige Anweisungen. Eine Task erhält zudem eine Priorität, die angibt, wie viele Programmzeilen ausgeführt werden bis die nächste Task abgearbeitet wird. Eine hohe Priorität sorgt daher für ein schnellstmögliches Abarbeiten dieser Task. Wird in einer Task eine Wartezeit ("delay") ausgeführt, pausiert diese Task und der

Prozessor arbeitet nur die anderen Tasks ab. Neben den erwähnten asynchronen Tasks bietet VAL 3 ebenso synchrone Tasks.



Kennziffer	Bezeichnung
1	T1 führt Funktion delay() aus
2	Wartezeit

Abbildung 2.12: Sequentielle Anordnung von Tasks [20]

Datentypen
In VAL 3 gibt es verschiedene einfache Datentypen, die nachfolgend kurz erläutertet werden. Neben den einfachen Datentypen gibt es weitere Datentypen, die z.B. Punkte für die Bewegungsteuerung darstellen. Auf eine Erläuterung dieser wird verzichtet, da diese Kommunikation nicht benötigt werden.

- **BOOL:** Variablen und Konstanten von diesem Typ können true und false annehmen.
- **NUM:** Datentyp zur Darstellung von numerischen Werten mit 14 signifikanten Stellen. Im Gegensatz zu vielen gängien Programmiersprachen wird in VAL 3 nicht zwischen verschiedenen Zahlenformaten unterschieden. Dieser Datentyp kann von ganzzahligen Werten bis hin zu Kommazahlen alles abbilden.
- **BITFELD:** Dieser Datentyp ermöglicht es eine Bitfolge z.B. von booleschen Werten oder digitalen Ein- und Ausgängen kompakt abzuspeichern.
- **STRING:** In Variablen und Konstanten diesen Typs können Zeichenketten mit einer Länge von bis zu 256 Byte gespeichert werden. Die interne Zeichencodierung erfolgt mit Unicode UTF 8. Bei ASCII-Zeichen entspricht die Läge von 256 Bytes somit 234 Zeichen. Initialisiert werden Variablen diesen Datentyps standardmäßig mit einem leeren String ().

- **DIO:** Dieser Datentyp ermöglicht es aus dem Programm auf die digitalen Ein-/Ausgänge der Steuerung zuzugreifen. Hierzu speichert die dio-Variable einen Link zu der entsprechenden Hardware in Form einer physischen Adresse.
- **AIO:** Mit Hilfe diesen Datentyps können die analogen Ein- und Ausgänge der Steuerung verknüpft werden.
- **SIO:** Dieser Datentyp ermöglicht es einen seriellen Port oder einen Ethernet Socket Anschluss zu verknüpfen. Dadurch kann beispielsweise eine TCP/IP - Nachricht geschrieben oder gelesen werden, in dem der Variable des Datentyps ein Wert zugewiesen wird oder ein Wert abgelesen wird.

Robotersteuerung
Neben Funktionalitäten die klassische Programmiersprachen aufweisen, benötigt VAL 3 insbesondere Möglichkeiten, um den Roboterarm zu steuern. Exemplarisch werden einige Befehle kurz vorgestellt. Zum Einsatz

- **disable / enable Power:** Hiermit lässt sich die Armleistung ab bzw. einschalten. Für den lokalen, manuellen oder Testbetrieb hat diese Funktion keinen Einfluss.
- **getVersion:** Mit dieser Anweisung lassen sich die Version der Hardware- und Software - Komponenten des Robotercontrollers abfragen.
- **movej:** Dieser Befehl führt eine Bewegung zu einem Punkt aus. Angegeben werden muss die Winkelposition der Achsen für den Zielpunkt (vom Datentyp "joint"), die Geometrie des Werkzeuges (Datentyp "tool") und Bewegungsdaten, wie Geschwindigkeit und Beschleunigung (Datentyp "msdec").
- **movel:** Dieser Befehl ermöglicht im Gegensatz zum Vorangegangen eine lineare Bewegung. Anstelle der Winkelpositionen als Ziel wird eine kartesische Zielkoordinate (Datentyp "point") vorgegeben.

2.5 Anlagensteuerung

Ein Roboter ist meist nur ein Teil einer Automatisierungslösung. Dieser kann um Förderbänder, Sensoren und andere Aktoren ergänzt werden. Wie die Steuerung dieser Gesamtanlage umgesetzt wird unterscheidet sich von Anlagenautomatisierer zu Anlagenautomatisierer. Hier soll die bei robomotion gängige Umsetzung aufgezeigt werden, da diese relevant für das Gesamtverständnis und die Konzeptionierung der Komponenten ist.
SPS

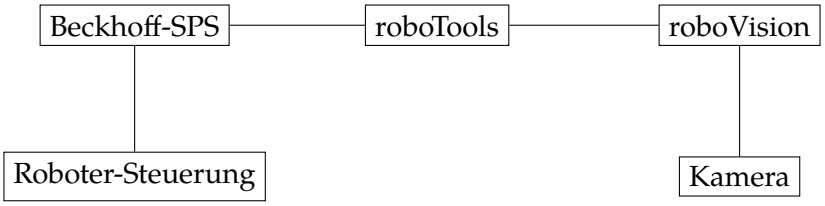


Abbildung 2.13: Anlagensteuerung Aufbau

Zentrales Element jeder Anlagenautomatisierung ist eine Beckhoff-SPS, welche auf einem IPC läuft. Diese ist für die Ablauflogik und İntelligenz"der Anlage zuständig. Auf anderen Bestandteilen, wie z.B. in einer Robotersteuerung ist hingegen nur eine geringe Komplexität abgebildet. In der Robotersteuerung werden meistens Kommondos mit Jobnummern angelegt, wie beispielsweise die Bewegung von Punkt A zu Punkt B. Durch den Aufruf der Jobnummer durch die SPS erhält der Roboter den Befehl zur Bewegungsausführung und meldet eine erfolgreich durchgeführte Bewegung der SPS zurück. Welche Bewegungen in welcher Reihenfolge und wann durchgeführt werden, ist dementsprechend nicht in der Robotersteuerung, sondern in der SPS-Steuerung programmiert.

roboTools

Weiterer Bestandteil der Automatisierung ist ein Programm namens "roboTools". Dieses läuft auf einem Windows-Betriebssystem in Form einer WPF (Windows Presentation Foundation), des ".NETFrameworks. Das Programm roboTools mit seiner grafischen Oberfläche enthält verschiedene Diagnosewerkzeuge und kann dem Anlagenbediener Informationen über bspw. Störungen oder der Anlagenauslastung ausgeben und visualisieren. Die Daten hierfür bezieht roboTools überwiegend von der SPS.

roboVision

Häufig werden in Anlagen Industriekameras eingesetzt, um die Position von Objekten zu ermitteln oder eine automatisierte Qualitätskontrolle vorzunehmen. Zur Weiterverarbeitung der Kameradaten, um die Position oder die Qualitätsgüte zu bestimmen kommt das Programm "roboVisionßum Einsatz. Wie roboTools handelt es sich um ein WPF-Anwendung, welche auf einem Windows-Betriebssystem läuft.

Hardware und Zusammenspiel

Sowohl roboTools, als auch roboVision laufen auf einem IPC. Dies kann der selbe IPC sein, auf dem bereits die SPS läuft. Insbesondere für roboVision wird jedoch meist ein eigenständiger leistungsfähiger IPC eingesetzt, da für die Bildverarbeitung eine hohe Rechenleistung zur Verfügung stehen muss. Für die Umsetzung der Kommunikationsschnittstelle, spielt dies jedoch keine große Rolle, da sich alle IPCs in einem Netzwerk befinden. Da es sich um getrennte Programme handelt können diese alle als eigenständige Einheiten behandelt werden, auch wenn sie auf dem selben IPC laufen. Das Zusammenspiel dieser Komponenten ist nachfolgend dargestellt (vgl. Abbildung 2.13).

2.6 WPF

Da bereits roboTools und roboVision auf WPF beruhen, ist WPF von großer Relevanz für diese Arbeit. WPF basiert auf dem .NET - Framework. Das .NET Framework ist eine Plattform für unterschiedliche Programmiersprachen, wie C# oder Visual Basic mit denen verschiedene Applikationstypen, wie Desktop, Web oder Mobile entwickelt werden können.
Das .NET-Framework ist eine Entwicklung von Microsoft. Die meisten Anwendungen können jedoch plattformübergreifend eingesetzt werden. Lediglich Desktopanwendungen, wie z.B. WPF sind an Windows gebunden.
C# Das .NET-Framework ist offen für verschiedenste Programmiersprachen, jedoch ist die Verwendung von C# am geläufigsten und kommt in den vorhandenen Anwendungen von robomotion zum Einsatz. C# ist eine Entwicklung speziell für .NET und kombinierte Konzepte und Bestandteile von Java, JavaSrcipt, C++ und Visual Basic. Dabei handelt es sich um eine vollwertige objektorientierte Sprache mit Eigenschaften, wie Vererbung und Kapselung.
Programmierumgebung
Visual Studio und DevOps
WPF

2.7 Bewertung

In der Literatur wird meist das TCP Protokoll zur Datenübertragung eingesetzt, um die Korrektheit der Daten sicherzustellen. In [19] kommt das Protokoll UDP zur Übertragung von Live-Kamera-Bildern zum Einsatz, da hier die Schnelligkeit über die Korrektheit zu stellen ist. Während in der Literatur z.B. in [19] VPN zur IT-Sicherheit eingesetzt wird, kann dies im Rahmen des Projektes entfallen.

3 Anforderungsdefinition

Die Aufgabenstellung und die grobe Anforderungsformulierung soll in diesem Kapitel mit Hilfe eines Lastenheftes in eine strukturierte Form gebracht werden. Das Lastenheft beschreibt, welche Funktionalitäten das zu entwerfende System aufweisen soll. Darüber hinaus kann das Lastenheft Rahmenbedingungen enthalten, wie z.B. zulässige Reaktionszeiten oder weitere Einschränkungen. Das Lastenheft wird in aller Regel vom Auftraggeber erstellt. Der Auftragnehmer beschreibt im Pflichtenheft durch welche Systemfunktionalitäten die Aufgaben erfüllt werden sollen und wie die Umsetzung technisch erfolgt. Das Lasten- und Pflichtenheft dient auch zur rechtlichen Absicherung der Vertragsparteien. Obwohl ein Lasten- und Pflichtenheft oft als Vertragsgrundlage entworfen wird und dies im Rahmen des Projektes nicht notwendig ist, soll ein Lastenheft eingesetzt werden. Dies ermöglicht es alle Anforderungen und Rahmenbedingungen zu berücksichtigen und diese im gesamten Projektverlauf im Blick zu behalten. Auf ein Pflichtenheft wird hingegen verzichtet. [1] Das Lastenheft wird in Muss- und Kann-Kriterien unterteilt. Während bei den Muss-Kriterien eine Erfüllung essentiell ist, stellen die Kann-Kriterien eine optionale Erweiterung dar, die dem System jedoch einen Mehrwert verschaffen.

Tabelle 3.1: Lastenheft

Nr.	Anforderung	Kann (K) / Muss (M)
1	Übertragung von Fehlermeldungen des Stäubli-Roboters an die Anlage	M
2	Übertragung von beliebigen Daten des Stäubli-Roboters an die Anlage	M
3	Übertragung von Daten z.B. Kameradaten von der Anlage an den Stäubli-Roboter	M
4	Ausgabe der Roboterfehler auf der Anlagensteuerung in verständlicher Form (ausformuliert, Fehlercode genügt nicht)	M
5	Funktionierende Feldbusverbindung zwischen Stäubli-Roboter und SPS	M
6	Feldbus Kommunikation zwischen Stäubli-Roboter und SPS	M
7	Abspeichern der Fehler und der Daten in log-File	M
8	Fehlerfreie und prozesssichere Kommunikation	M

Tabelle 3.1: Lastenheft

Nr.	Anforderung	Kann (K) / Muss (M)
9	Einsatz von mehreren Stäubli-Robotern zeitgleich soll möglich sein	K
10	Vorgabe von der Anlagensteuerung, wie häufig Daten versendet werden sollen	K
11	Visualierung der Daten	K
12	Verwertung der Daten, um Warnungen oder ähnliches auszugeben	K

4 Konzeptionierung und Systementwurf

UML-Diagramm Anwendungsfälle definieren, Rahmenbedingungen identifizieren, Anforderungen aus Lastenheft verfeinern

Softwarekomponenten definieren und im Komponentendiagramm darstellen gemäß UML

4.1 Feldbus-Verbindung

Aufgabe der Feldbusverbindung ist die Kommunikation zwischen dem Stäubli-Roboter und der Beckhoff-SPS sicherzustellen. Hierzu werden einerseits Befehle, wie zur Bewegungsdurchführung zum Roboter übertragen, andererseits Informationen, wie z.B. eine Bestätigung der vollendeten Bewegung zurück an die SPS gesandt.

Wahl des Feldbusses

In Kapitel 2.1 wurden die Feldbusse in Standard-Feldbusse und Ethernet basierende Feldbusse unterteilt. Die Wahl fällt hierbei auf einen Ethernet-basierenden Feldbus, weil dies der aktuelle Stand der Technik darstellt und zudem firmenintern ausnahmslos eingesetzt wird. Nach Kapitel 2.1.2 sind Profinet, Ethernet/IP, EtherCAT und Sercos III häufig eingesetzte Ethernet basierende Feldbusse. Eine Gegenüberstellung der Vor- und Nachteile der einzelnen Feldbusse und eine Auswahl anhand dieser Kriterien kann jedoch entfallen, da zwei Argumente für den Einsatz von EtherCAT, alle anderen Kriterien überwiegen.

Zum Einsatz kommt eine Beckhoff-SPS. EtherCAT ist eine Entwicklung von Beckhoff und daher bestmöglichst auf die SPS von Beckhoff abgestimmt.

Darüber hinaus, kommt unternehmensintern bereits in anderen Projekten EtherCAT zum Einsatz. Die Wahl von EtherCAT bringt daher folgende Vorteile mit sich

- **Know How:** Firmenintern hat sich durch den wiederholten Einsatz von EtherCAT bereits eine große Wissenbasis und Erfahrung aufgebaut. Bei schwerwiegenderen Problemen ermöglicht dies eine einfachere Fehlersuche.
- **Schulungszwecke:** Durch andere Projekte sind beispielsweise die Service-Techniker bereits in EtherCAT geschult. Die Buchung einer zeit- und kostenintensiven Schulung für einen anderen Feldbus kann entfallen
- **Standardisierung:** Der Einsatz von vielen unterschiedlichen Systemen erhöht den Aufwand in Dokumentation und Einkauf. Daher wird versucht die Varianz möglichst gering zu halten und auf einen Standard, welcher bereits eingesetzt und

verbreitet ist zu setzen.

Master-Master vs. Master-Slave
Standardmäßig wird jedoch sowohl die Beckhoff-SPS, als auch der Stäubli Roboter als Master eingesetzt. Jedoch ist EtherCAT rein für Master-Slave-Kommunikation ausgelegt, d.h. es kann nur ein Master existieren. [8] Beckhoff bietet jedoch eine spezielle Bridge-Klemme (EL6692) an, die es erlaubt EtherCAT-Stränge zu synchronisieren und ein Datenaustausch zwischen den einzenen Strängen zu ermöglichen. Da jeder EtherCAT-Strang einen eigenen Master besitzt ermöglicht die Bridge den Einsatz von mehreren Mastern.
Bridge Hierdurch entstehen jedoch zusätzliche Kosten und auch die Komplexität des Systems steigt. Da sowohl die Stäubli-Steuerung, als auch die SPS ebenso als Slave konfiguriert werden kann, entfällt die Entscheidung auf die Master-Slave-Kommunikation, um die zusätzliche Master-Master-Klemme einzusparen.
Wahl des Masters
Nachdem die Entscheidung für eine Master-Slave-Architektur gefallen ist, muss festgelegt werden, ob die Beckhoff-SPS oder der Stäubli-Roboter als Master konfiguriert wird. Da der Einsatz von mehreren Stäubli-Robotern in Automatisierungslösungen denkbar ist, könnte nicht jeder Stäubli-Roboter als Master eingesetzt werden. Folglich wäre ein Stäubli-Roboter ein Master, die anderen Stäubli-Roboter, wie auch die SPS Slaves. Um diese Unterschiede in den einzelnen Stäubli-Robotern zu verhindern, ist es sinnvoll die Beckhoff-SPS als Master zu konfigurieren, sodass alle Stäubli-Roboter einheitlich als Slaves eingesetzt werden. Darüber hinaus ist die SPS der zentrale Teil der Anlagenautomatisierung, von welcher weitere Komponenten, wie Roboter anderer Hersteller gesteuert werden können, weshalb die Wahl der SPS als Master sinnvoll ist.
Die Gesamtentscheidung zur Konfiguration ist nachfolgend mit Hilfe eines morphologischen Kastens dargestellt.

	Konfiguration			
Parameters	**Option 1**	**Option 2**	**Option 3**	**Option 4**
Bussystem	etherCAT	Profinet	Sercos III	Soft
Architektur	Master-Master	Master-Slave		
Master	SPS	Stäubli		

Merkmal	Ausprägung			
Faktor 1	etherCAT	profinet	Ethernet/IP	Sercos III
Faktor 2	Master-Master	Master-Slave		
Faktor 3	SPS als Master	Stäubli als Master		

4.2 TCP/IP-Verbindung

Nachdem die Feldbus-Verbindung konzeptioniert wurde, erfolgt die Auslegung der TCP/IP-Verbindung. Hierbei müssen viele Einzelentscheidungen getroffen werden, welche am Ende mit einem morphologischen Kasten zusammengefasst dargestellt werden.

Wahl der Architektur

Wie in Kapitel 2.2 erläutert kommt für eine TCP/IP-Verbindung die Client-Server-Architektur zum Einsatz. Dabei kann entweder der Stäubli-Roboter oder die Anlagensteuerung als Server dienen. In der Literatur wird meist empfohlen, die Seite mit der höheren Rechenleistung als Server einzusetzen (vgl. Kapitel 2.2). Dies stellt in diesem Fall die Anlagensteuerung dar. Wenn hingegen der Roboter als Server dienen würde und es zum Einsatz von mehreren Roboter kommen würde, wäre der Einsatz von mehreren Client-Instanzen auf der Anlagensteuerung notwendig oder die anderen Roboter müssten als Client konfiguriert werden. Daher eignet sich die Anlagensteuerung besser als Server und die einzelnen Roboter werden als Client konfiguriert.

Nachrichtentypen

Für folgende Daten, die übertragen werden müssen, muss festgelegt werden, ob und wie diese auf einzelne Nachrichten aufgeteilt werden.

- **Systemereignisse:** Erfasste Meldungen, wie z.B. Fehler müssen von dem Roboter zu der Anlagensteuerung übertragen werden.
- **Prozessdaten:** Vom Roboter erfasste Daten, wie Temperaturen oder digitale Werte, wie Fehler an den Ventilen müssen vom Roboter zur Anlagensteuerung übertragen werden.
- **Kameradaten:** Positionsdaten von zugreifenden Objekten, welche durch Kameradaten generiert werden, müssen von der Anlagensteuerung zum Roboter übertragen werden.

Da die Kameradaten in die entgegengesetzte Richtung, wie die anderen Daten übertragen werden, werden diese in eine eigene Nachricht verpackt. Die Systemereignisse und die Prozessdaten können in einer Nachricht oder in getrennten Nachrichten versandt werden. Da die Prozessdaten durchgängig anfallen, die Systemereignisse jedoch azyklisch lohnt sich eine Trennung. Die Prozessdaten wiederum können gebündelt gesendet werden oder getrennt werden in einzelne Nachrichten z.B. für die Temperaturen und die digitalen Werte. Da beispielsweise Temperaturen nicht besonders dynamisch sind könnte die Übertragungsfrequenz für diese Daten geringer sein, was die Auslastung der Verbindung reduzieren würde. Dies erhöht jedoch die Komplexität, da viele verschiedene Nachrichtentypen anfallen, was bei Wartungen oder Änderungen in späteren Jahren zum Problem werden kann. Da das zyklische Versenden aller Prozessdaten, die Übertragungsrate der TCP/IP-Verbindung nicht übersteigen wird, fällt die Entscheidung

auf das Zusammenfassen aller Prozessdaten in einer Nachricht. Folglich gibt es drei Nachrichtentypen: Kameradaten, Systemereignisse und Prozessdaten.

Auslöser für Versand der Systemereignisse Die Systemereignisse können entweder zyklisch z.B. einmal pro Sekunde zusammengefasst versendet werden oder direkt bei Auftreten. Die Entscheidung fällt auf ein direktes Versenden bei Auftreten, damit wichtige Fehlermeldungen schnellstmöglich ankommen. Zudem handelt es sich um ein azyklisches Ereignis, d.h. die Systemereignisse werden nur sporadisch versendet.

Auslöser für Versand der Prozessdaten Die Prozessdaten werden zyklisch benötigt und sind nicht abhängig von besonderen Ereignissen. Ein zyklischer Versand bietet sich hier an. Die Sendefrequenz muss dann in jedem Stäubli-Roboter einprogrammiert werden und ist dann fest eingestellt. Vorteilhaft wäre es wenn die Sendefrequenz von dem Server Anlagensteuerung vorgegeben werden kann. Dadurch kann die Sendefrequenz angepasst und z.B. zur Fehlerfindung erhöht werden. Darüber hinaus kann die Änderung an dieser zentralen Instanz erfolgen und muss nicht manuell bei jedem einzelnen Stäubli-Roboter durchgeführt werden. Dies ist insbesondere praktisch, wenn ein Zugriff über das Fernwartungssystem auf die Anlagensteuerung erfolgt. Daher soll ein Datenanforderungstelegramm zyklisch z.B. einmal pro Sekunde von der Anlagensteuerung an alle Clients (Stäubli-Roboter) gesandt werden. Diese sollen nach Erhalt der Anforderung ein Prozessdatentelegramm versenden. Der Einsatz von Datenanforderungstelegrammen bringt ein weiteren Vorteil mit sich. Durch die Zeitdifferenz zwischen Datenanforderungstelegramm und Erhalt des Datentelegramms kann überprüft werden, ob die Datenübertragung ausreichend stabil ist und ob der Client noch verbunden ist. Wenn innerhalb einer vorgegeben Zeitspanne von x Milisekunden keine Antwort in Form des Datentelegramms erfolgt kann eine Warnung ausgegeben werden, dass dieser Stäubli-Roboter nicht erreichbar ist.

Auslöser für Versand von Kameradaten

Beim Anfallen von Positionsdaten, welche anhand von Kameradaten berechnet werden, solllen diese schnellstmöglichst an den Roboter übertragen werden. Aus diesem Grund erfolgt der Versandt dieser Daten Ereignissgetriggert, d.h. direkt nach Erhalt der Positionsdaten wird ein Telegramm an den entsprechenden Roboter gesandt. Die Entscheidungsfindung bei den eben genannten Auswahlmöglichkeiten ist nachfolgend in einem morphologischen Kasten dargestellt.

Tabelle 4.1: Morphologischer Kasten TCP/IP-Verbindung

Merkmal	Option 1	Option 2	Option 3
Architekturwahl	Roboter = Server	Anlage = Server	
Nachrichtentypen	Prozessdaten unterteilt	Eine Nachricht	Prozessdaten und Systemereignisse getrennt
Auslöser Systemereignisse	Zyklisch	Ereignisbasiert	

Merkmal	Option 1	Option 2
Wahl der Architektur 1	Anlagensteuerung als Server	Roboter als Server
Nachrichtentypen 4	Eine Nachricht 5	Prozessdaten und Systemereigniss
Auslöser Versand Systemereignisse	Zyklisch	Ereignisbasiert

Tabelle 4.2: Beispiel für eine gefärbte Tabelle mit erster Spalte losgelöst.

Tabelle 4.1: Morphologischer Kasten TCP/IP-Verbindung

Merkmal	Option 1	Option 2	Option 3
Auslöser Prozess-daten	Anforderungs-telegramm	Zyklisch	
Auslöser Kamera-daten	Zyklisch	Ereignisbasiert	

Nachrichtenaufbau Datentelegramm// Nachfolgend soll der Nachrichtenaufbau des Datentelegramms festgelegt werden. Durch Brainstorming und durch Inspiration von anderen Telegram-Aufbauten wie Ethernet (vgl. Kapitel 2.1.2) werden verschiedene Möglichkeiten des Telegrammaufbaus entworfen:

- **Konzept 1:** <Nachrichten-ID>,<Sender-ID>,<Uhrzeit>,<Daten>
- **Konzept 2:** <Nachrichten-ID>,<Sender-Name>,<Daten>
- **Konzept 3:** <Einzigartiger-Code>,<Sender-ID> <Daten>
- **Konzept 4:** <Nachrichten-ID>,<Sender-ID>,<Daten>,<CRC>
- **Konzept 5:** <Nachrichten-ID>,<Sender-ID>,<Daten Gleitkomma, dynamisch>

In Konzept 1,2,4,5 kommt eine Nachrichten-ID am Anfang, damit der Empfänger weiß, dass z.B. ein Datentelegramm und keine Systemereignisse versendet werden. Anschließend folgt die Sender-ID, damit der Server weiß von welchem Roboter die Daten stammen. Bei Konzept 2 wird die Sender-ID, welche durch eine Zahl dargestellt wird, durch einen konfigurierbaren Namen ersetzt, was die Nutzerfreundlichkeit erhöht, da bei späterer Wartung ggf. unbekannt ist, welcher Roboter welche Nummer besitzt. Bei Konzept 1 wird darüber hinaus die Uhrzeit mit versendet, dies ist für Dokumentationszwecke sinnvoll und ebenso kann eine Zeitdifferenz oder unterschiedliche Einstellungen in der Uhrzeit zwischen Roboter und Anlagensteuerung detektiert werden. Das Datum kann durch die Anlagensteuerung bei Erhalt der Nachricht bestimmt werden und muss daher nicht mitversendet werden. Bei Konzept 4 wird zusätzlich ein CRC-Segment (Cyclic Redundancy Check) versendet. Im CRC-Segment wird eine Prüfsumme übermittelt, welche vom

Sender aus der zu übermitteltenden Nachricht berechnet. Der Empfänger berechnet die Prüfsumme aus der erhaltenen Nachricht und vergleicht sie mit dem CRC-Segment, um zu überprüfen, ob die Nachricht korrekt übermittelt wird. TCP/IP stellt jedoch bereits selbständig eine fehlerfreie Übertragung durch ein CRC-Segment sicher. Durch eine zusätzliche Prüfsumme kann zwar die Sicherheit bezüglich des korrekten Einlesens der Daten erhöht werden, für die Übertragung selbst ist ein CRC-Segment jedoch nicht mehr notwendig.

4.3 Datenverwertung

Tabelle 4.3: Verfügbare Daten

Daten	Zugriff	Verwendung
CPU battery test	D (CpuIO)	-
CPU overcurrent	D (CpuIO)	-
Fast memory state	D (CpuIO)	-
CPU temperature	A (CpuIO)	Alarm, PM
CPU board temperature	A (CpuIO)	Alarm, PM
CPU fan speed	A (CpuIO)	Alarm, PM
Free RAM	A (CpuIO)	-
CFast memory remaining lifetime	A (CpuIO)	-
System management CPU usage	A (CpuUsage)	-
Robot control CPU usage	A (CpuUsage)	-
Synchronous VAL 3 CPU usage	A (CpuUsage)	-
Available CPU usage for VAL 3 processing	A (CpuUsage)	-
User Fieldbuses CPU usage	A (CpuUsage)	-
HMI CPU usage	A (CpuUsage)	-
SRS connection CPU usage	A (CpuUsage)	-
OPC-UA CPU usage	A (CpuUsage)	-
CPU Load Score (higher is better)	A (CpuUsage)	-
CPU Load Score (min)	A (CpuUsage)	-
VAL 3 instructions per sequencing	A (CpuUsage)	-
VAL 3 synr inst. per cyle (high priority)	A (CpuUsage)	-
VAL 3 synr inst. per cyle (low priority)	A (CpuUsage)	-
Valve feedback (1.1, 1.2, 2.1, 2.2)	D (DsiIO)	-
Axis brake feedback (1-6)	D (DsiIO)	-
Error on valves outputs	D (DsiIO)	Alarm
Error on brakes outputs	D (DsiIO)	Alarm
Error on safe digital inputs	D (DsiIO)	Alarm

Tabelle 4.3: Verfügbare Daten

Daten	Zugriff	Verwendung
DSI non-reduced brake supply voltage undershoot	D (DsiIO)	Alarm
DSI reduced brake supply voltage undershoot	D (DsiIO)	Alarm
DSI logic supply voltage undershoot	D (DsiIO)	Alarm
DSI overtemperature	D (DsiIO)	Alarm
DSI board tempereature	A (DsiIO)	Alarm, PM
Axis motor temperature (1-6)	A (DsiIO)	Alarm, PM
Axis encoder temperature (1-6)	A (DsiIO)	Alarm, PM
DSI state	A (DsiIO)	-
DSI error code	A? (DsiIO)	? Grau? Alarm
Arm operation counter	A (DsiIO)	-
safe Input state (0-7)	D (DsiIoSafe)	-
Fast Input (1-2)	D (FastIO)	-
Fast Output (1-2)	D (FastIO)	-
Power unit identification (bit 1-3)	D (PSIO)	-
Main power state	D (PSIO)	-
Internal bus voltage state	D (PSIO)	-
Power unit internal temperature state	D (PSIO)	-
24V state	D (PSIO)	-
Buckfull mode feedback	D (PSIO)	-
Memorize a power dropout	D (PSIO)	-
Brake test warning	D (Rsi9IO)	Alarm
Brake test successful	D (Rsi9IO)	-
Temperature of RSI board	A (Rsi9IO)	Alarm, PM
Error Code of RSI board	A? (Rsi9IO)	?Grau? Alarm
STARC board temperature	A (StarcIO)	Alarm, PM
Axis drive case temperature (1-6)	A (StarcIO)	Alarm, PM
Motor Winding Temperature (1-6)	A (StarcIO)	Alarm, PM
Axis drive junction temperature (1-6)	A (StarcIO)	Alarm, PM
Geschwindigkeit Endmanipulator	getSpeed(...)	-
Schleppfehler Achsen 1-6	getPositionErr()	-
Drehmoment Achsen 1-6	getJointForce(...)	-
Bewegungsauftrag und Fortschritt	getMoveId()	-
Konfiguration des Roboters	getVersion(...)	-
Stromversorgung bei Stillstand abschalten	hibernateRobot()	-
Systemereignisse	getEvents(...)	-

Durch die Verfügbarkeit von Roboterdaten ergeben sich Nutzungspotentiale, wie z.B.:

- **Überwachung und Alarm:** Bei Überschreiten von Schwellwerten oder bei Fehlermeldungen Alarm auslösen
- **Predictive Maintenance:** Vorhersagen treffen, wann Wartung erfolgen soll, um Ausfälle vorbeugend zu verhindern.
- **Datenarchivierung und Compliance:** Datenspeicherung für Schadensfall oder gesetzliche Gewährleistung
- **Trendanalyse:** Muster und Tendenzen in den Daten erkennen

Prinzipiell lassen sich alle verfügbaren Daten sammeln, visualisieren und auswerten. Bei einigen dieser Daten wie z.B. des freien RAM-Speichers ist der daraus entstehende Nutzen beschränkt. Mittels Brainstorming wurden verschiedene Anwendungsszenarien ausgedacht, im nachfolgenden sollen jedoch nur die sinnvollsten hiervon vorgestellt werden.

Temperaturen

Neben der Temperatur von verschiedenen Computer-Chips und Platinen, wie z.B. CPU, CPU-Platine DSI-Platine, RSI-Platine und STARC-Platine sind verschiedene Temperaturwerte von den Antrieben abrufbar. Hier sind die Temperaturen der Motoren, Encoder, Antriebsgehäuse, Antriebswicklungen und Steuergeräte zu nennen. Diese Daten können sowohl zur Überwachung als auch zur Predictive Maintenance verwendet werden. Bei Überschreiten eines Grenzwertes kann ein Alarm bzw. eine Warnung ausgegeben werden. Da von Seiten des Herstellers keine zulässigen Grenzwerte vorgegeben sind, müssen die aufgezeichneten Messwerte unter Berücksichtigung von Schwankungen analysiert werden und Grenzwerte festgelegt werden, ab welchen Werten das Verhalten nicht mehr als "normalängesehen werden kann. Die Dynamik von Temperaturen ist im Allgemeinen relativ gering, da sich die Materialien aufgrund ihrer Wärmekapazität erst aufheizen müssen. Aus diesem Grund ist das Übertragen von den Temperaturwerten in vergleichsweise großen Abständen zulässig z.B. alle 15 Sekunden. Im Falle eines technischen Defektes wäre eine möglichst frühe Warnung wünschenswert, jedoch ist die Zeit bis ein Techniker den Alarm wahrnimmt und entsprechende Aktionen starten kann vergleichsweise groß, weshalb dieser Anwendungsfall kein höhere Übertragungsrate rechtfertigt. Da die Genauigkeit von vielen Temperatursensoren nur im Bereich von 0,5K liegt und sehr genaue Temperaturwerte keinen zusätzlichen Mehrwert bieten, ist eine Übertragung der Temperaturen ohne Nachkommastellen ausreichend. Dadurch lässt sich ein Temperaturwert problemlos mit einem einzelnen Byte (Werte zwischen 0 und 255°C) übertragen.

Error-Meldungen

Von dem Stäubli-Roboter werden einige Fehlermeldungen erfasst, wie von den Ventil und Brems-Ausgängen, den digitalen Sicherheits-Eingängen, des DSI-Chip, RSI-Chip und

Bremsentests. Ebenso werden "DSI non-reduced brake supply voltage undershoot", "DSI reduced brake supply voltage undershoot", "DSI logic supply voltage undershootünd "DSI overtemperatureërfasst. Error-Meldungen wie diese sollen als Alarm bzw. Fehlermeldung ausgegeben werden. Ebenso ist ein Dokumentieren sinnvoll, um bei Ausfällen oder Defekten die Ursachensuche zu erleichtern. Jeder Fehler kann als Wert 1 oder 0 dargestellt werden, weshalb je Fehlermeldung ein Bit genügt. Alle Error-Meldungen lassen sich somit über 2 Bytes abbilden.

Schleppfehler

Der Schleppfehler entspricht der Abweichung zwischen Soll-Position und Ist-Position jeder einzelnen Achse. Ein herausstechend großer Schleppfehler kann auf Probleme mit der Steuerung, den Motoren, den Encoder oder den Achsen selbst hinweisen. Eine genaue Eingrenzung ist auf Basis der gegebenen Daten nicht möglich, jedoch ist eine Warnung sinnvoll. Zur Ermittlung der Grenze, bei deren Überschreiten eine Warnung ausgegeben werden soll, können aufgezeichnete Daten unter Berücksichtigung von Schwankungen herangezogen werden. Der Schleppfehler muss sehr regelmäßig erfasst werden, um die kontinuierliche Bewegung best möglichst abzubilden.

Drehmoment Achsen

Das Drehmoment der einzelnen Achsen ändert sich kontinuerlich während der Bewegung. Durch das Vergleichen der Drehmomente mit einer Referenzfahrt können Auffälligkeiten entdeckt werden. Schleichend zunehmende Momente deuten z.B. auf fehlende Schmierung oder Lagerdefekte hin. Abrupt zunehmende Momente können durch ein Festhängen oder einen Crash verursacht werden. Eine Dokumentation der Abweichungen ist sinnvoll, um aus Gewährleistungsgründen unsachgemäße Verwendung z.B. durch einen Crash nachzuweisen. Analog zum Schleppfehler müssen auch die Drehmomente sehr kontinuierlich erfasst werden.

Sytemereignisse

Ereignisse, die von der Steuerung festgestellt werden, wie z.B. Fehlermeldungen oder das Einschalten der Energieversorgung für die Achsen werden in einer system-log-Datei aufgezeichnet. Sie lassen sich in die Schweregrade Ïnfo", "Warning", Ërroründ "Criticalünterteilen. Das Auslesen der Systemereignisse kann mit dem Befehl "getEventsërfolgen. Diese sollen beim Auftreten sofort über TCP/IP übertragen werden.

4.4 WPF und GUI

5 Design und Implementierung

Design: Vergleich von Lösungsbausteinen (z.B. Vergleich Softwarebibliotheken) und Auswahl) auf Basis von Anforderungen und Recherche. Wissenschaftliche Bewertungskriterien nötig. Harvey-Diagramm.

Neben grundlegenden Entwurfsentscheidungen wird die Beschreibung einzelner Komponenten detailiert. Für komplexe Komponenten UML-Klassendiagramm Definition Schnittstelle zwischen den Komponenten und Kommunikation in Form von Sequenzdiagramm

Implementierung - Programmierung erfolgt gemäß den in Softwarespezifikation und entwurf festgelegten Randbedingungen, Kommentierung von Source Code ist essentiell. Gemäß bestehenden Standards (z.B. xdoc) - Generierung einer HTML-Hilfe aus den Kommentaren des Source-Codes z.b. durch Doxygen - Entwicklung erfolgt unter Git - Einheitliche Konventionen für das Schreiben des Quellcodes: Coding Conventions!

5.1 Stäubli-Roboter in VAL3

5.1.1 EtherCAT

5.1.2 TCP/IP

Für die Implementierung der TCP/IP-Verbindung auf dem Controller des Stäubli-Roboters muss in der SRS eine Socket-Verbindung angelegt werden. Hierzu wird in der E/A-Verwaltung ein Client angelegt, welcher die IP-Adresse und den Port des Servers zugewiesen bekommt. Darüber hinaus wird ein sogenannter Timeout von 0 s gesetzt. Bei einem Timeout von 0 wird auf den Vorgang, welcher ein Lesen oder Schreiben sein kann gewartet. Bei einem Timeout kleiner 0 wird hingegen nicht bis zur Ausführung des Vorgangs gewartet. Be einem Timeout größer 0 wird hingegen eine gewisse Zeit gewährt, bis zu dieser der Timeout durchgeführt werden kann. Die Nachricht soll in diesem Fall jedoch direkt gelesen oder geschrieben werden, weshalb kein Spielraum im Rahmen des Timeouts gewährt wird. [20] Die Socket-Verbindung wird als E/A-Verbindung in VAL3 betrachtet, weshalb eine globale Variable mit dem Namen des Clients angelegt werden kann und hierüber auch gelesen und beschrieben werden kann. Die Socket-Verbindung wird nur dann erstellt, wenn sie ihm Rahmen des Programmablaufs z.B. durch die Befehle sioSet und sioGet benötigt wird. Der Client versucht dann eine Verbindung zum Server aufzubauen. usepackageffcode
num sioGet(sio siInput, num& nData[])

Diese Funktion schreibt ein gelesenes Zeichen oder einen gelesen Array von Zeichen von siInput in das Array nData. Als Rückgabewert dient die Anzahl der gelesenen Zeichen.
num sioSet(sio siOutput, num& nData[])
Mit dieser Funktion kann in VAL3 die zu übermittelnde Nachricht nData versendet werden, indem der E/A-Verbindung siOutput die Nachricht zugewiesen wird. Zurückgegeben wird die Anzahl der geschriebenen Zeichen oder 1ïm Falle des Timeouts.
Das Versenden von Nachrichten erfolgt über einen Byte-Array, das heißt durch die Aneinanderreihung mehrerer Bytes. Folglich muss die zu versendete Nachricht in einen Byte-Array umgewandelt werden und beim Empfangen muss der Byte-Array interpretiert werden.
num toBinary(num nValue[], num nValueSize, string sDataFormat, num& nDataByte[])
Diese Funktion wandelt einen numerischen Wert, welcher das Datenformat sDataFormat besitzt in einen Byte-Strom und speichert diesen im Array nDataByte. Über das Datenformat wird beispielsweise angegeben ob es sich um einen Gleitkommawert handelt, ob ein Vorzeichen vorliegt und ob das Little-Endian oder das Big-Endian-Format angewandt wird. Mit nDataSize kann die Anzahl der zu kodierenden Zeichen beschränkt werden.
num fromBinary(num nDataByte[], num nDataSize, string sDataFormat, num& nValue[])
Umgekehrt ermöglicht diese Funktion, einen empfangen Byte-Array in numerische Werte zu konvertieren. Das Ergebnis im Datenformat nDataFormat wird in nValue gespeichert. Die Anzahl der zu decodierenden Bytes wird festgelegt durch nDataSize, wenn nicht alle Bytes des Eingangs-Array nDataByte decodiert werden sollen.

5.2 .NET in C#

5.2.1 TCP/IP

5.2.2 Datenverwertung und Visualisierung

6 Validierung

7 Ausblick und Fazit

Abbildungsverzeichnis

2.1 Übergang Parallelverdrahtung zu Feldbussen [3] 2
2.2 Marktanteile Feldbusse und Industrial Ethernet [4] 3
2.3 Telegramm-Aufbau Ethernet [3] 5
2.4 Topologie: oben Standard-Feldbus, unten Ethernet basierender Feldbus [3] 6
2.5 Telegrammbearbeitung [8] 7
2.6 Schichtenmodell TCP/IP [3] 8
2.7 IP-Adresse Aufbau [3] 9
2.8 Sequenzdiagramm Server-Client-Architektur [9] 10
2.9 Stäubli TX2 - 90 [17] 12
2.10 Stäubli CS9 Controller [18] 13
2.11 CS9 Controller Computer-Einschub [18] 13
2.12 Sequentielle Anordnung von Tasks [20] 15
2.13 Anlagensteuerung Aufbau 17

Tabellenverzeichnis

1.1 Table to test captions and labels. 1

2.1 CS9 Controller Computer-Einschub 13

3.1 Lastenheft 19
3.1 Lastenheft 20

4.1 Morphologischer Kasten TCP/IP-Verbindung 24
4.2 Beispiel für eine gefärbte Tabelle mit erster Spalte losgelöst. 25
4.1 Morphologischer Kasten TCP/IP-Verbindung 25
4.3 Verfügbare Daten 26
4.3 Verfügbare Daten 27

Literatur

[1] M. Broy und M. Kuhrmann, *Einführung in die Softwaretechnik*. Springer, 2021.

[2] P. D.-I. A. Verl und P. D.-I. O. Riedel, *Leitfaden für die Anfertigung studentischer Arbeiten am ISW*. ISW Stuttgart, 2020.

[3] E. Hering, R. Martin, J. Gutekunst und J. Kempkes, *Elektrotechnik und Elektronik für Maschinenbauer*. Springer, 2017.

[4] H. Networks, *Profinet jetzt vor EtherNet/IP - Industrial Ethernet legt weiter zu*, [Online, aufgerufen am 01. Dezember 2023], 2021.

[5] F. Dopatka, „Ein Framework für echtzeitfähige Ethernet-Netzwerke in der Automatisierungstechnik mit variabler Kompatibilität zu Standard-Ethernet," 2008.

[6] K. GmbH, *Echtzeit-Ethernet-Kommunikation*, [Online, aufgerufen am 01. Dezember 2023], -.

[7] W. Riggert, „Rechnernetze," *Fachbuchverlag Leipzig*, 2002.

[8] Beckhoff, „EtherCAT System-Dokumentation Version 5.6," *Beckhoff Automation GmbH & Co. KG*, 2002.

[9] Stäubli, *socket TCP-IP Technical documentation*. Stäubli International AG, 2019.

[10] D. W. Kim, H.-D. Lee, C. W. de Silva und J.-W. Park, „Service-provider intelligent humanoid robot using TCP/IP and CORBA," *International Journal of Control, Automation and Systems*, Jg. 14, S. 608–615, 2016.

[11] C. Harnisch, *Netzwerktechnik*. mitp Verlags GmbH & Co. KG, 2009.

[12] S. Wendzel, „Einführung in die Netzwerksicherheit," in *IT-Sicherheit für TCP/IP- und IoT-Netzwerke: Grundlagen, Konzepte, Protokolle, Härtung*, Springer, 2018.

[13] W. Babel, „Internet of Things und Industrie 4.0," in *Internet of Things und Industrie 4.0*, Springer, 2023.

[14] H. Stefan und J. Schreier, *Was ist OPC UA? Definition, Architektur und Anwendung*, [Online, aufgerufen am 18. Dezember 2023], 2019.

[15] S. I. AG, *Stäubli - Über uns*, [Online, aufgerufen am 19. Dezember 2023].

[16] S. I. AG, *Stäubli - Produktübersicht*, [Online, aufgerufen am 19. Dezember 2023].

[17] Stäubli, *Roboterarm - Baureihe TX 90 Betriebsanleitung*. Stäubli International AG, 2019.

[18] Stäubli, *Controller CS9 Betriebsanleitung*. Stäubli International AG, 2022.

[19] B. Groza und T.-L. Dragomir, „Using a cryptographic authentication protocol for the secure control of a robot over TCP/IP," in *2008 IEEE International Conference on Automation, Quality and Testing, Robotics*, IEEE, Bd. 1, 2008, S. 184–189.

[20] Stäubli, *VAL 3 - Handbuch*. Stäubli International AG, 2022.