



**Universität Stuttgart**

Institut für Steuerungstechnik  
der Werkzeugmaschinen und  
Fertigungseinrichtungen (ISW)



# Studienarbeit

## jbjhkbkj

eingereicht von

*Lukas Schlotter*

aus Stuttgart

Studiengang

Prüfer

Betreuer

Eingereicht am

M. Sc. Mechatronik

Prof. Dr.-Ing. Oliver Riedel

My supervisor, M.Sc.

6. November 2023

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Motivation . . . . .	2
1.2	Anforderungsdefinition . . . . .	2
1.3	Methodik . . . . .	2
<b>2</b>	<b>Grundlagen und Stand der Technik</b>	<b>3</b>
2.1	Feldbusse . . . . .	3
2.2	TCP/IP . . . . .	4
2.3	Stäubli-Roboter . . . . .	4
2.4	Anlagensteuerung . . . . .	4
<b>3</b>	<b>Konzeptionierung und Systementwurf</b>	<b>5</b>
3.1	Feldbus-Verbindung . . . . .	5
3.2	TCP/IP-Verbindung . . . . .	5
3.3	Datenverwertung . . . . .	5
<b>4</b>	<b>Implementierung</b>	<b>8</b>
4.1	Stäubli-Roboter in VAL3 . . . . .	8
4.1.1	EtherCAT . . . . .	8
4.1.2	TCP/IP . . . . .	8
4.2	.NET in C# . . . . .	9
4.2.1	TCP/IP . . . . .	9
4.2.2	Datenverwertung und Visualisierung . . . . .	9
<b>5</b>	<b>Validierung</b>	<b>10</b>
<b>6</b>	<b>Ausblick und Fazit</b>	<b>11</b>
	<b>Abbildungsverzeichnis</b>	<b>12</b>
	<b>Tabellenverzeichnis</b>	<b>13</b>
	<b>Literatur</b>	<b>14</b>

# 1 Einleitung

Warum startet das hier mit ner 0? aTex allows you to manage citations within your document through the use of a separate bibtex file (filename.bib).

## 1 Einleitung

Bibtex files follow a standard syntax that allow you to easily reference the citations included in that file through the use of a bibliography management package. There are multiple bibliography management packages that you can use to manage citations. This guide will demonstrate how to use biblatex which allows for the most customization.

### 1.1 Motivation

Dieses Bild zeigt blabla bla von dem Buch [1] und auch [2]

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

### 1.3 Methodik

## 2 Grundlagen und Stand der Technik

### 2.1 Feldbusse

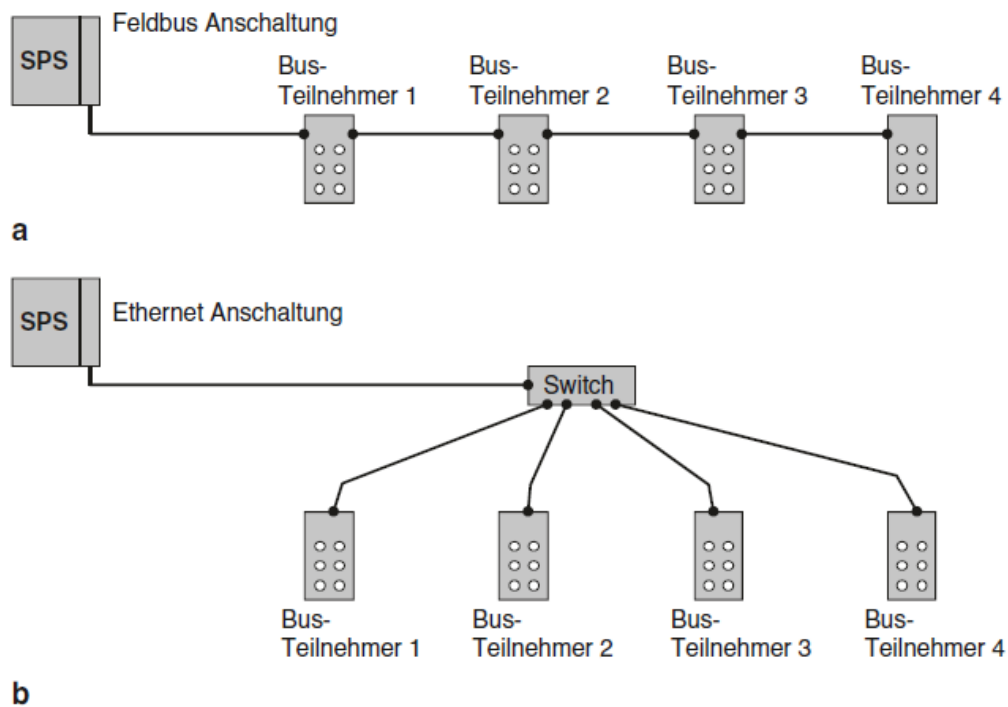
Feldbusse: Elektrotechnik für Maschinenbauer ab S.485

Feldbusse: Profibus, CAN, Sercos

Ethernet basierte Feldbusse: Profinet, Ethernet/IP, EtherCAT, Sercos III

Ethernetbasierte Systeme sind bereit Feldbusse abzulösen

Ethernet: deutlich mehr Daten als klassisch



**Bild H-19** Übergang von der Linienstruktur in die Sternstruktur bei Ethernet. (a) Linienstruktur bei Standard Feldbussystemen, (b) Sterntopologie bei Ethernet Feldbussystemen

Abbildung 2.1: Anschaltung Feldbus und Ethernet [3]

Multi-Master Bussen (z.B. CAN oder TCP/IP) vs. Mono-Master

## 2 Grundlagen und Stand der Technik

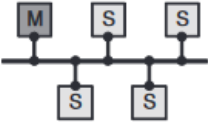
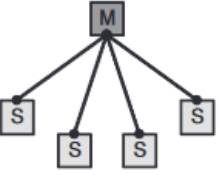
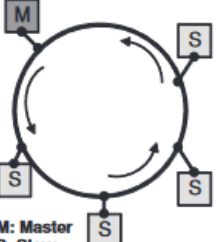
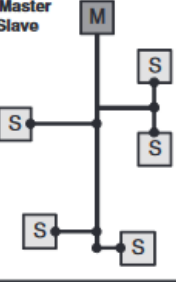
Topologie	Bus-/ Linien-Struktur	Stern-Struktur	Ring-Struktur	Baum-Struktur
Aufbau	 <p>M: Master S: Slave</p>	 <p>M: Master S: Slave</p>	 <p>M: Master S: Slave</p>	 <p>M: Master S: Slave</p>
Beispiele	Profibus CAN-Bus Bit-Bus P-Net	DIO-Bus Ethernet	Interbus S Ethernet Sercos	AS-Interface LON

Abbildung 2.2: Topologien

## 2.2 TCP/IP

MAC-Adresse eindeutig von Gerät. Für bessere Identifiuierung aber IP-Adresse

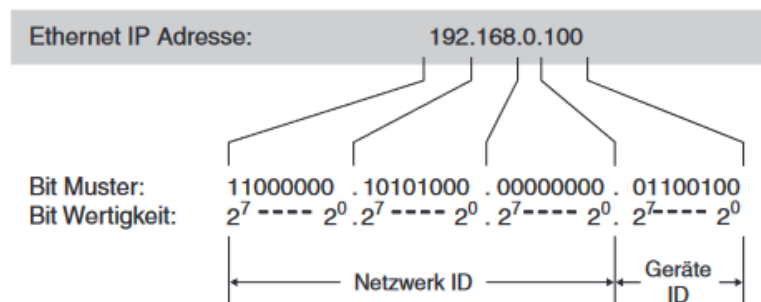


Abbildung 2.3: Topologien

Verschiedene Klassen an IP-Adressen. Meist Klasse C verwendet

Weil Multi-Master-Bus braucht man CSMA/CD-Verfahren -> nicht echtzeitfähig. Lösung: Echtzeitprotokolle

## 2.3 Stäubli-Roboter

## 2.4 Anlagensteuerung

# 3 Konzeptionierung und Systementwurf

## 3.1 Feldbus-Verbindung

## 3.2 TCP/IP-Verbindung

## 3.3 Datenverwertung

Tabelle 3.1: Verfügbare Daten

Daten	Zugriff	Verwendungsszenario
CPU battery test	E/A (CpuIO)	87837
CPU overcurrent	E/A (CpuIO)	78
Fast memory state	E/A (CpuIO)	778
CPU temperature	E/A (CpuIO)	Überlastungswarnung
CPU board temperature	E/A (CpuIO)	778
CPU fan speed	E/A (CpuIO)	Verntialtordefekt, Überlastung
Free RAM	E/A (CpuIO)	778
CFast memory remaining lifetime	E/A (CpuIO)	778
System management CPU usage	E/A (CpuUsage)	778
Robot control CPU usage	E/A (CpuUsage)	778
Synchronous VAL 3 CPU usage	E/A (CpuUsage)	778
Available CPU usage for VAL 3 processing	E/A (CpuUsage)	778
User Fieldbuses CPU usage	E/A (CpuUsage)	778
HMI CPU usage	E/A (CpuUsage)	778
SRS connection CPU usage	E/A (CpuUsage)	778
OPC-UA CPU usage	E/A (CpuUsage)	778
CPU Load Score (higher is better)	E/A (CpuUsage)	778
CPU Load Score (min)	E/A (CpuUsage)	778
VAL 3 instructions per sequencing	E/A (CpuUsage)	778
VAL 3 synr inst. per cyle (high priority)	E/A (CpuUsage)	778

### 3 Konzeptionierung und Systementwurf

Tabelle 3.1: Verfügbare Daten

Daten	Zugriff	Verwendungsszenario
VAL 3 synr inst. per cyle (low priority)	E/A (CpuUsage)	778
Valve feedback (1.1, 1.2, 2.1, 2.2)	E/A (DsiIO)	778
Axis brake feedback (1-6)	E/A (DsiIO)	778
Error on valves outputs	E/A (DsiIO)	778
Error on brakes outputs	E/A (DsiIO)	778
Error on safe digital inputs	E/A (DsiIO)	778
DSI non-reduced brake supply voltage undershoot	E/A (DsiIO)	778
DSI reduced brake supply voltage undershoot	E/A (DsiIO)	778
DSI logic supply voltage undershoot	E/A (DsiIO)	778
DSI overtemperature	E/A (DsiIO)	778
DSI board temperature	E/A (DsiIO)	778
Axis motor temperature (1-6)	E/A (DsiIO)	778
Axis encoder temperature (1-6)	E/A (DsiIO)	778
DSI state	E/A (DsiIO)	778
DSI error code	E/A (DsiIO)	778
Arm operation counter	E/A (DsiIO)	778
safe Input state (0-7)	E/A (DsiIoSafe)	778
Fast Input (1-2)	E/A (FastIO)	778
Fast Output (1-2)	E/A (FastIO)	778
Power unit identification (bit 1-3)	E/A (PowerSupplyIO)	778
Main power state	E/A (PowerSupplyIO)	778
Internal bus voltage state	E/A (PowerSupplyIO)	778
Power unit internal temperature state	E/A (PowerSupplyIO)	778
24V state	E/A (PowerSupplyIO)	778
Buckfull mode feedback	E/A (PowerSupplyIO)	778
Memorize a power dropout	E/A (PowerSupplyIO)	778
Brake test warning	E/A (Rsi9IO)	778
Brake test successful	E/A (Rsi9IO)	778
Temperature of RSI board	E/A (Rsi9IO)	778
Error Code of RSI board	E/A (Rsi9IO)	778
STARC board temperature	E/A (StarclIO)	778
Axis drive case temperature (1-6)	E/A (StarclIO)	778
Motor Winding Temperature (1-6)	E/A (StarclIO)	778



Tabelle 3.1: Verfügbare Daten

Daten	Zugriff	Verwendungsszenario
Axis drive junction temperature (1-6)	E/A (StarCIO)	778
Geschwindigkeit Endmanipulator	num getSpeed(tool tTool)	778
Schleppfehler Achsen 1-6	joint getPositionErr()	778
Drehmoment Achsen 1-6	void getJointForce(num& nForce))	778
Bewegungsauftrag und Fortschritt	num getMoveld()	778
Konfiguration des Roboters	string getVersion(string sComponent)	778
Stromversorgung bei Stillstand abschalten	num hibernateRobot()	778
Systemereignisse	num getEvents(...)	778

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

- **Überwachung und Alarm:** Bei Überschreiten von Schwellwerten 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 "normal" angesehen werden kann.

## 4 Implementierung

### 4.1 Stäubli-Roboter in VAL3

#### 4.1.1 EtherCAT

#### 4.1.2 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. Bei 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. [4] 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 im 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 gelesenen 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 1 im 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

## 4 Implementierung

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.

## 4.2 .NET in C#

### 4.2.1 TCP/IP

### 4.2.2 Datenverwertung und Visualisierung

## **5 Validierung**

## **6 Ausblick und Fazit**

# Abbildungsverzeichnis

2.1	Anschaltung Feldbus und Ethernet [3] . . . . .	3
2.2	Topologien . . . . .	4
2.3	Topologien . . . . .	4

# Tabellenverzeichnis

1.1	Table to test captions and labels. . . . .	2
3.1	Verfügbare Daten . . . . .	5
3.1	Verfügbare Daten . . . . .	6
3.1	Verfügbare Daten . . . . .	7

# Literatur

- [1] T. Tantau, *Tikz & pgf*, 2013.
- [2] M. Kohm und J.-U. Morawski, *KOMA-Script – ein wandelbares LaTeX-2-Paket*, 2013.
- [3] E. Hering, R. Martin, J. Gutekunst und J. Kempkes, *Elektrotechnik und Elektronik für Maschinenbauer*. Springer, 2017.
- [4] Staubli, *VAL 3-Handbuch*. Staubli International A, 2022.