HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

# MASTERARBEIT

Herr
**Tim Sieber**

## Konzeptionierung eines industriell genutzten Hebezeugs für die Herausforderungen von Industrie 4.0

2020

Fakultät **Ingenieurwissenschaften**

# MASTERARBEIT

# Konzeptionierung eines industriell genutzten Hebezeugs für die Herausforderungen von Industrie 4.0


Autor:
**Tim Sieber**

Studiengang:
Elektro- und Informationstechnik

Seminargruppe:
EI18w1-M

Erstprüfer:
Prof. Dr. -Ing. T. Beierlein

Zweitprüfer:
M.Sc. Markus Süß


Mittweida, 2020

**Bibliografische Angaben**

Sieber, Tim: Konzeptionierung eines industriell genutzten Hebezeugs für die Herausforderungen von Industrie 4.0, 115 Seiten, 45 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Ingenieurwissenschaften

Masterarbeit, 2020

**Referat**

# I. Inhaltsverzeichnis

# II. Abbildungsverzeichnis

# III. Tabellenverzeichnis

# IV. Abkürzungsverzeichnis

| | |
|---|---|
| $\mu$C | Mikrocontroller |
| ADC | Analog-Digital-Converter |
| ALELO | Adress Latch Enable Low |
| CAN | Controller Area Network |
| COB ID | Connection Object Identifier |
| ECAT HDR | *EtherCAT Header* |
| EEPROM | Electrically Erasable programmable Read Only Memory |
| EPU | *EtherCAT* Processing Unit |
| ESC | *EtherCAT-Slave-Controller* |
| FIFO | First In First Out |
| FMMU | Fieldbus Memory Managment Unit |
| GÜ | Generalüberholung |
| HMI | Human Machine Interface |
| LAN | Lokal Area Network |
| LLC | Logical Link Control |
| MAC | Media Access Control |
| MDI | Medium Dependent Interface |
| MII | Media Independent Interface |
| NMT | Netzwerk Management |
| OD | Object Dictionary |
| OSI-Referenzmodell | Open Systems Interconnection Referenzmodell |
| PCS | Physical Coding Sublayer |
| PDO | Process Data Object |
| PLCP | Physical-Layer-Convergence-Procedur |
| PMA | Physical Medium Attachment |
| PMD | Physical Medium Dependent |
| PWM | pulsweitenmodeliert |
| RAM | Random Access Memory |
| ROM | Read Only Memory |
| RPDO | Receive PDO |
| RTC | Real-Time-Clock |
| RTOS | Real Time Operating System |

S ..... tatsächliche Nutzungsdauer
SAS ..... Sauer Automation Sachsen
SD ..... Secure Digital
SDIO ..... Secure Digital Input Output
SDK ..... Software Developmentkit
SDO ..... Service Data Object
SFL ..... Steuerflasche
SPI ..... Seriell Peripherie Interface
SWP ..... Save Working Period
TPDO ..... Transmit PDO
WLAN ..... Wireless Local Area Network

# 1 Einleitung

Die hier vorliegende Arbeit dient der Erstellung eines Konzeptes zur Weiterentwicklung von Hebezeugen der Firma *Sauer Automation Sachsen* (im weiteren Verlauf der Arbeit mit *SAS* abgekürzt). Diese ist notwendig, um die wachsenden Anforderungen an das Hebezeug zu erfüllen. Bei den Anforderungen handelt es sich um Flexibilität und Vernetzung. Eine solche Weiterentwicklung nutzt die Verfahren, welche durch *Industrie 4.0* in der Produktion Einzug gehalten haben.

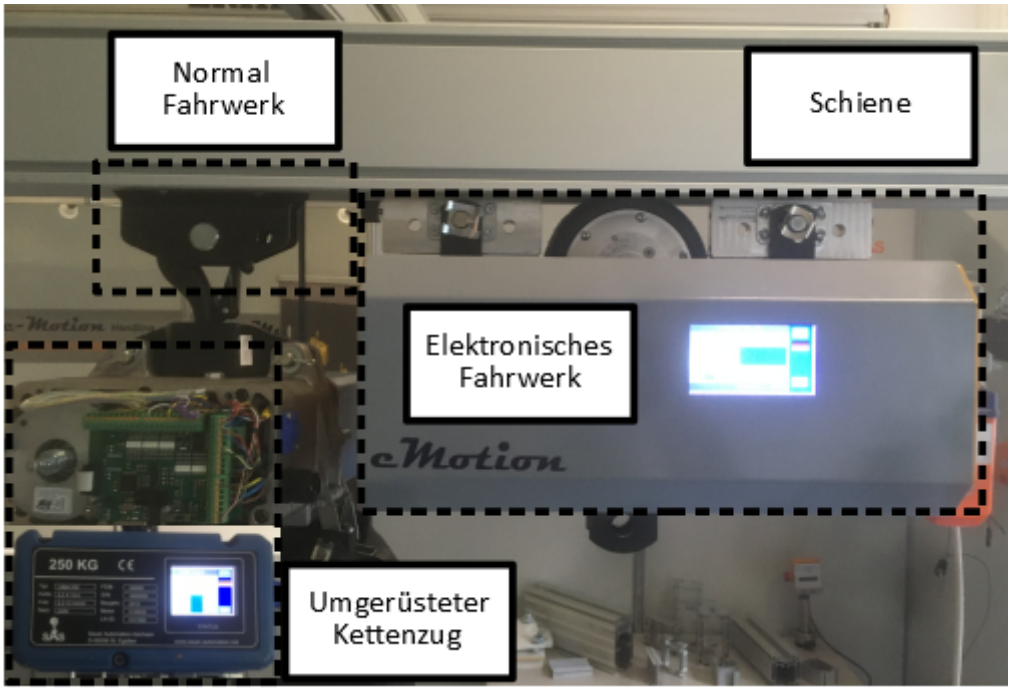



Abbildung 1.1: Fahrwerksvarianten

Die *SAS* betreibt auch Entwicklungsarbeit an den Komponenten, die sich im direkten Zusammenhang mit dem Hubprozess befinden. Dazu zählt das Fahrwerk. Fahrwerke sind nötig, wenn eine horizontale Bewegung ausgeführt werden muss. Durch die Elektrifizierung dieser Bewegung zeichnet sich der technische Fortschritt aus. Die angebotenen Hebezeuge besitzen als Tragmittel Seile oder Ketten. Bei Kettenhebezeugen handelt es sich um Serienhebezeuge. Diese werden zugekauft und elektronisch optimiert. Damit ist erkennbar, dass die Kompetenz im Bereich der Hebezeuge nicht auf der Herstellung, sondern auf der Weiterentwicklung von Serienhubgeräten liegt.
In der Abbildung 1.1 sind alle Komponenten dargestellt, die zu einer Krananwendung zählen. Bei dem aufgezeigten Hebezeug handelt es sich um einen Kettenzug. Dieser ist einmal mit und einmal ohne Abdeckkappe dargestellt.
Weiterentwickelte Produkte zeichnen sich durch Funktionen aus, wie stufenlose Bedienung, die Möglichkeit unterschiedliche Parameter projektbezogen anpassen zu können

und den Zugriff auf diese Parameter über ein integriertes Human Machine Interface (weiter mit HMI abgekürzt). Verwendet werden die HMIs der Firma *Nextion* [1]. Im weiteren Verlauf der Arbeit erfolgt daher die Nutzung der Begriffe HMI und *Nextion* synonym. Für die Realisierung der Funktionen ist ein Tausch der Steuerelektronik notwendig. Mit der eingesetzten *SAS*-Platine (weiter als *Logikboard* bezeichnet) mit integrierten Mikrocontroller (im Verlauf der Arbeit mit $\mu$C bezeichnet) sind die unterschiedlichen Funktionen und Parameter als Softwareroutinen verfügbar [2]. Diese Steuerelektronik wird auch beim elektronischen Fahrwerk *eDrive* eingesetzt.
Das *Logikboard* steuert einen Regler, welcher entsprechend der Situation einen Elektromotor regelt. Ein Einstellen des gewünschten Betriebszustandes erfolgt über vier Leitungen. Bei den darüber übertragenen Signalen handelt es sich um das Signal für die Richtung (je eine Leitung) und das Geschwindigkeitssignal als analoger Spannungswert von 0...10 V (Spannungs- und Masseleitung). Diese drei Signale werden vom Benutzer durch ein Bedienteil gesteuert. Zur anwendungsspezifischen Steuerung sind von Seiten der *SAS* drei unterschiedliche Bedienvarianten verfügbar.
Mit der Steuerflasche (in der weiteren Arbeit mit SFL abgekürzt) lässt sich die Last durch das Drücken der jeweiligen Taste anheben oder senken. Beide Tasten sind zweistufig ausgeführt, wodurch insgesamt vier unterschiedliche Fahrkennlinien verfügbar sind. Über das HMI können für jede dieser Kennlinien die Endgeschwindigkeit, die Beschleunigungszeit und die Verzögerungszeit konfiguriert werden. Die Kommunikation von SFL zum *Logikboard* findet mittels logischen Signalen statt. Diese Signale werden über jeweils eine Leitung für Heben, Senken und die Geschwindigkeitsstufe zwei übertragen.
Ist als Bedienelement der Drehgriff ausgewählt, so wird das Heben beziehungsweise Senken der Last über den vom Bediener vorgegebenen Drehwinkel gesteuert. Die Geschwindigkeit ist dabei ebenfalls vom jeweiligen Drehwinkel abhängig. Zum Betrieb des Drehgriffs sind drei Potentiometerleitungen nötig.
Das umfangreichste Steuerelement ist das TFT-Bedienteil oder auch *Balancer*. Die angehängte Nutzlast kann entweder durch den angebrachten blauen Sensorring am Griff manuell oder durch das Führen dieser vom Bediener automatisch positioniert werden. Für den Automatikbetrieb ist im Griff eine Lastmesszelle enthalten, welche zu jedem Zeitpunkt die anhängende Nutzlast erfasst. Das Ausführen von vertikalen Bewegungen resultiert aus der Laständerung. Die Referenzgröße zur Bestimmung der Änderung ist die Nutzlast zu Beginn des Automatikmodus. Im Bediengriff ist ebenfalls ein HMI enthalten. Über dieses sind die Einstellungen und Kalibrierungen für den *Balancer* ausführbar. Die Signalübertragung zwischen TFT-Bedienteil und *Logikboard* erfolgt durch zwei logische Signale für die Auf- beziehungsweise Abwärtsbewegung und einem pulsweitenmodulierten-Signals (im Weiteren mit PWM-Signal abgekürzt) für die Geschwindigkeit [1]. Die grafische Darstellung der Bedienelemente ist in Abbildung 1.2 gegeben.

[1] Darstellung mit Abdeckkappe in Abbildung 1.1
[2] obere Darstellung in Abbildung 1.1 ohne Abdeckkappe


Abbildung 1.2: Bedienteil für Hebezeuge

Auf der folgenden Seite ist in Abbildung 1.3 die gesamte Kommunikationsstruktur grafisch dargestellt. Durch die Anforderungen an das System muss diese Struktur überarbeitet werden. Diese Notwendigkeit resultiert aus dem Anwendungsgebiet der Hebezeuge. Sie sind in der Industrie Teil von Fertigungsprozessen. Sicherheits- und Ablaufsteuerungen der Prozesse benötigen Daten des Hubvorgangs.
Der Bezug dieser Daten, derer Bereitstellung und Verarbeitung macht die Weiterentwicklung der Hebezeuge unabdingbar.

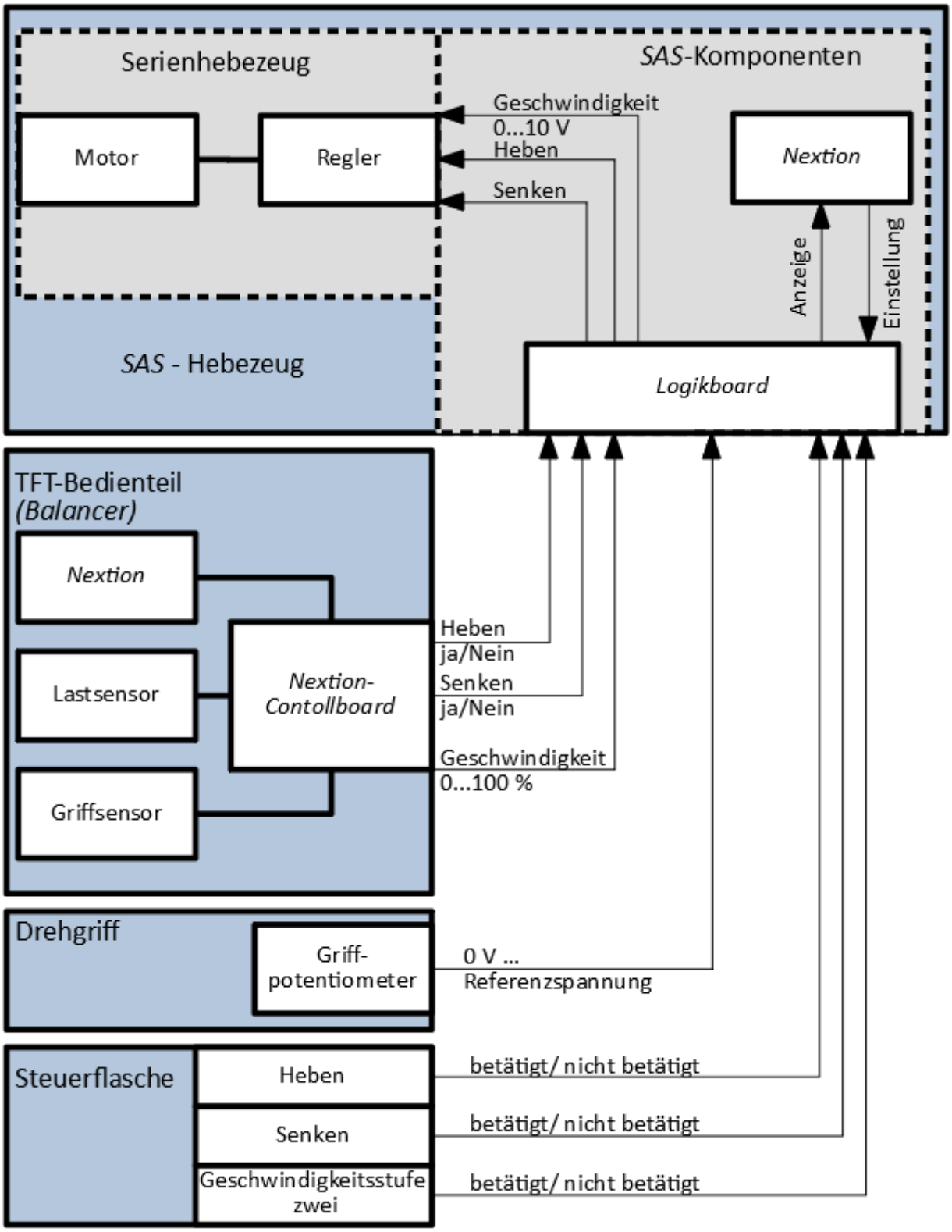



Abbildung 1.3: Schematische Darstellung der Umrüstung [1]

# 2 Grundlagen

Dieses Kapitel dient zur Beschreibung der Grundlagen, welche zum Verständnis der nachfolgenden Ausarbeitung und in der Einleitung getroffenen Aussagen nötig sind. Eingeteilt ist das Kapitel in die Abschnitte Arbeitsumfeld der Konzeptionierung, Mikroprozessortechnik, Kommunikation und SPS.

## 2.1 Arbeitsumfeld der Konzeptionierung

Im Unterpunkt 2.1.1 wird das Themengebiet der Hebetechnik erläutert. Dazu zählt die Ausführung von Begriffen und Spezifikation bezüglich dieses Gebietes.
Das Arbeitsumfeld, was sich durch die Aufnahme dieses Projektes mit der Hebetechnik überschneidet, ist die *Industrie 4.0*. Merkmale der *Industrie 4.0* und warum diese mit den Anforderungen aus Kapitel 1 identisch sind, erschließt sich aus Punkt 2.1.2.
In Kapitel 2.1.3 wird der aktuelle Stand der Technik aufgezeigt, in dem sich aktuelle Hebezeuge im Kontext zu *Industrie 4.0* derzeit befinden.

### 2.1.1 Hebetechnik

Hebezeuge zählen zu den Fördermaschinen. Diese Maschinengruppe bildet die technische Lösung für die Bewegung von stofflichen Gütern, ohne diese substantiell oder geometrisch zu verändern. Die Hauptaufgabe der Hebezeuge ist die senkrechte Förderung von Lasten. Eine vertikale Umpositionierung beschreibt ebenfalls die Aufgabe von Aufzügen. Der Unterschied zwischen Aufzügen und Hebezeugen ist die Anzahl der vorhandenen Freiheitsgrade während der Umpositionierung. Aufzüge sind in Schienen geführt, wo hingegen die Last bei Hebezeugen freischwebend ist. Dadurch entsteht ein räumlicher Arbeitsbereich. Ein Serien- oder Einzelhebezeug zeichnet sich zusätzlich durch seine horizontal feste Position aus. Kann das Gerät waagerecht verfahren werden, so handelt es sich um einen Kran [16, S.1-3].
Die gesamte Gruppe der Hebezeuge lässt sich noch auf Grund ihrer Antriebskraft und des verwendeten Tragmittels unterscheiden. Die üblichen Tragmittel sind Ketten, Seile oder Bänder. Der Antrieb kann entweder mittels Muskelkraft, durch den Einsatz von Elektro- oder Druckluftgeräten erfolgen [17, S.5].
Da im Projekt als Antriebskraft einzig Elektromotoren verwendet werden, handelt es sich bei den Hebezeugen um Maschinen. Diese Behauptung resultiert aus der Definition der Maschine. Aus eben dieser Definition ist auch erkenntlich, dass eine vollständige Maschine alle Komponenten beinhaltet, welche zum Betrieb dieser nötig sind. Die Komponenten, welche sie mit ihrem Einsatzort oder ihrer Energiequelle verbinden, sind nicht

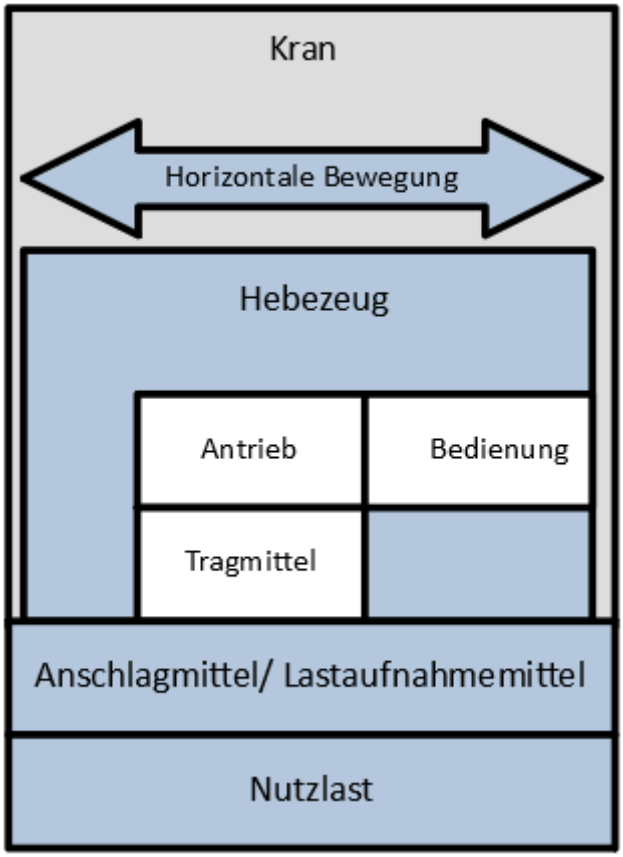



Abbildung 2.1: Komponenten und Zusammenhänge des Hebezeug

Bestandteil. [18, Artikel 2a].
Eine für den Betrieb des Hebezeugs nötige Komponente ist das Anschlag- oder auch Lastaufnahmemittel. Über dieses wird die Fördermaschine mit der Nutzlast verbunden [9, S.2].
Lastaufnahmemittel oder Anschlagmittel sind nicht zum Hebezeug gehörende Ausrüstungsteile, die zum Ergreifen der Last nötig sind und gesondert in Verkehr gebracht werden [18, Artikel 2d]. Die Zusammenhänge zwischen den Komponenten werden durch das Schema 2.1 zusammengefasst und grafisch dargestellt.

### 2.1.2 *Industrie 4.0*

In der vierten industriellen Revolution sollen moderne Produktions- und Automatisierungstechniken dazu beitragen, eine bessere Organisation und Steuerung der gesamten Wertschöpfungskette über den Lebenszyklus des Produktes zu gewährleisten [19, S.5].
Bei modernen Automatisierungstechniken handelt es sich um SPSen. Der Einsatz dieser industriellen Steuerung steht im Zusammenhang mit der immer weiter steigenden Anlagenkomplexität in Industriebetrieben. Durch den Einsatz solcher Steuerungen ist die individuelle Anpassung der Anlage an das jeweilige Projekt möglich [19, S.6] [20].
Die Verbesserung der Wertschöpfungskette soll mit der Vernetzung aller menschlichen und maschinellen Akteure, die am Prozess beteiligt sind, erreicht werden. Basis der Vernetzung ist die Digitalisierung und Echtzeitauswertung aller notwendigen Informationen [19, S.6].
Eine Vernetzung von Systemen bildet ein Netzwerk. Im industriellen Umfeld kommen verschiedene Netzwerktypen zum Einsatz. Aufteilen lassen sich die Netzwerke in Feldbusse, industrielles Ethernet und Wireless Verbindungen.
Die Arten und Verteilung der Netze in der Industrie ist in Abbildung 2.2 dargestellt.

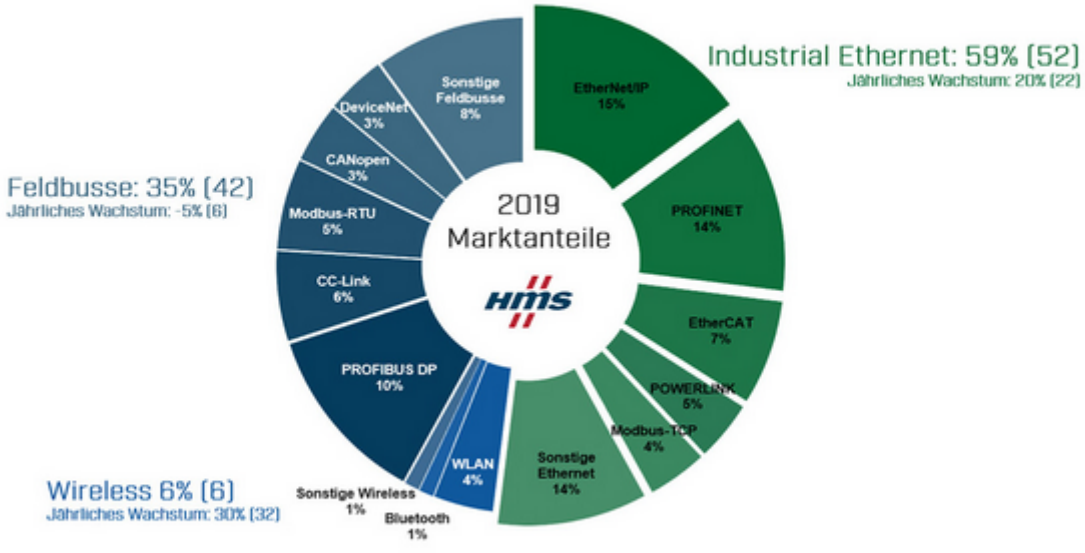



Abbildung 2.2: Marktanteile industrieller Netzwerke 2019 [2]

### 2.1.3 Stand der Technik

Die Hebetechnik hat ihr Aufgabenfeld an der Schnittstelle zur Automatisierungstechnik. Seil- oder Kettenhebezeuge werden eingesetzt, wenn das Umsetzen von Lasten eine starke Belastung für den Menschen darstellt aber ein Automatisieren unmöglich beziehungsweise unnötig ist [1].
Werden die Hebezeuge im industriellen Umfeld eingesetzt, stehen sie den Herausforderungen aus Kapitel 2.1.2 gegenüber. Aus der Vielzahl an Herstellern von industriellen Hebezeugen sind unterschiedliche Schritte in Richtung *Industrie 4.0* getätigt wurden.
Die Firma *Demag* realisiert die Übertragung der Echtzeitdaten über ihr *SafeControl*-System. Mittels dieser Sicherheitssteuerung und den darin beinhalteten Systemen *StatusControl* sowie *Smartcheck* sind Voraussetzungen gegeben, um moderne Produktions- und Logistikprozesse zu unterstützen. So können über die Sensortechnik *SmartCheck* alle Betriebsparameter kontinuierlich erfasst werden. Das System *StatusControl* ermöglicht über einen Fernzugriff die Betriebsdaten der Krananlage in Echtzeit zu liefern [21, S.5].
Die Sicherheitssteuerung *SafeControl* ist als Applikationsbaugruppe für den Schaltschrank verfügbar. Ergänzt wird diese durch das *SaftyKit* mit Antrieben, Frequenzumrichtern, Drehzahlgebern und einem Anwendungsprogramm [22].
Bei der Firma *Indeva* ist der Stand als *Industrie 4.0 ready* bezeichnet. Die Daten vom Hebezeug werden durch einen am Manipulator angebrachtes Gateway über das Wi-fi Netz an das IT-System gesendet. Es handelt sich bei den ausgelesenen Daten um Maschinen-Daten (digitale I/O, Alarme), Performance-Daten (Arbeitszyklus, zurückgelegte Strecke des Seils, Energieverbrauch, Arbeitstemperatur, gehobenes Gewicht) und Instandhaltungsdaten (präventive oder normale Wartung, Benachrichtigung wenn Wartung fällig ist). Es können auch Daten an das Hubgerät übermittelt werden. Die Daten, welche übertragen werden können, sind die Position und Höhe, in der die Last aufgenommen werden soll, Barcode-Daten und Lastaufnahmeszenarien [23].
Die Firma *Liftket* ermöglicht die Darstellung von relevanten Daten über ein im Hebezeug integriertes Display. Dabei werden durchgängig die Restnutzungsdauer, Betriebsstunden, Volllaststunden, Anzahl der Bremsspiele und Umrichtertemperatur angezeigt [24].
Mit dem *IQ2* der Firma *Gorbel* ist ein weiteres Hebezeug auf dem Markt, welches eine erkennbare Ausrichtung hin zur *Industrie 4.0* vorweist. Für die Vernetzung des Systems mit der Umgebung ist das Produkt mit WLAN ausgestattet. Eine Steigerung der Anpassungsfähigkeit an das jeweilige Projekt wird mit Hilfe frei konfigurierbarer Ein- und Ausgänge bewerkstelligt [25]. Die Konfiguration des Hebezeugs ist mit dem Tausch der Steuerelektronik verbunden. Das bedeutet, wenn eine Anpassung durchgeführt werden soll, muss dazu die gesamte Steuerelektronik *Gorbel* zur Verfügung gestellt werden. Anschließend wird eine entsprechend veränderte Version zurückgeliefert [26].

## 2.2 Mikroprozessortechnik

In diesem Abschnitt werden die Begriffe erläutert, die in der Arbeit im Zusammenhang mit dem Gebiet der Mikroprozessortechnik verwendet werden. Die Überschneidung der Themenbereiche Hebetechnik, *Industrie 4.0* und Mikroprozessortechnik ergibt sich aus den angewendeten Techniken und nötigen Komponenten.
Eine Komponente ist der $\mu$C. Er ist ein um Speicher und Peripheriekomponenten erweiterter Mikroprozessor.
Innerhalb eines $\mu$C sind die Daten in Registern abgelegt. Register sind in der Zentralrecheneinheit angelegte Speicherstellen [15, S.17-26].
Ein Speicher kann entweder als Festwertspeicher oder als Schreib-/Lesespeicher ausgeführt sein. Der Festwertspeicher oder auch Read Only Memory (ROM) kann vom Prozessor nur gelesen werden. Inhalte in den Speicherstellen werden bei der Herstellung oder in separaten Programmierprozessen eingefügt. Diese Inhalte bleiben auch ohne Energiezufuhr bestehen. Die Art des Zugriffs und der Datenerhalt unterscheiden sich beim Schreib-/Lesespeicher oder auch Random Access Memory (RAM) von denen des ROM. Er gestattet Lese- und Schreibzugriffe, verliert allerdings seine Inhalte, sobald die Energiezufuhr unterbrochen ist [15, S.30].
Bei den zu speichernden oder verarbeitenden Daten handelt es sich in der Mikroprozessortechnik um diskrete Signale. Ein binäres Signal kann genau zwei Werte annehmen. Jedes Zeichen aus diesen zwei Werten wird als Bit bezeichnet und kann den Wert eins oder null annehmen.
Die Anzahl n aneinander gereihter Bits ist die Wortlänge. Typische Wortlängen sind in der Tabelle 2.1 aufgelistet [15, S.46]. In einer größeren Anordnung von Bits werden die

| **Bezeichnung** | **Bitanzahl** |
|---|---|
| Nibble | 4 |
| Byte | 8 |
| Word | 16 |
| Doubleword | 32 |
| Quadword | 64 |

Tabelle 2.1: Bezeichnungen von Standardwortlängen [15, S.162]

Bereiche nach diesen Standardtypen aufgeteilt. Ein Byte beinhaltet die Bits null bis sieben. Dabei wird das Bit an Stelle null als Least-Significant-Bit und das auf dem Platz mit der Nummer sieben als Most Significant Bit bezeichnet. Bildet die Anzahl der Bits ein Word, teilt sich dieses in Lowbyte (Bit null bis sieben) und Highbyte (Bit acht bis fünfzehn) auf. Die weitere Einteilung erfolgt sinngemäß.
Beim Ablegen oder Übertragen von einem oder mehreren Bytes ist darauf zu achten, wie diese in den Speicher geschrieben werden. Dabei unterscheiden sich das Big- und Little-Endian-Format. Beim Little-Endian-Format wird das LSB auch an der niedrigsten Adresse abgelegt und das MSB an der höchsten. Für das Big-Endian-Format erfolgt dieser Vorgang umgekehrt [15, S.161/162].

Nachdem nun die Begriffe und Strukturen zu Daten bekannt sind, folgen wichtige Ausführungen zu ihrer Bearbeitung. Sind die zuerst abgelegten auch die zuerst entnommenen Daten, handelt es sich um ein First In First Out System (weiterhin mit FIFO bezeichnet) [15, S.161/162].
Zwischen der Eingabe und der Ausgabe findet die Datenverarbeitung statt. Werden diese Prozesse zyklisch vom Programm wiederholt, handelt es sich um Polling. Stellt eine Ein- oder Ausgabeeinheit eine Anforderung zur Bearbeitung, so handelt es sich um einen Interrupt [15, S.37/38].
Verschiedene Ein- und Ausgabeeinheiten sowie $\mu$C tauschen Daten miteinander aus. Eine Anordnung, in der sich diese Teilnehmer (Busknoten) befinden, wird als Bussystem bezeichnet [15, S.215]. Innerhalb eines solchen Systems besitzen Busknoten unterschiedliche Befugnisse. Der Master nimmt aktiv an den Abläufen teil, wo hingegen ein Slave auf Anweisungen wartet. Sind innerhalb eines Bussystems beide Komponenten vorhanden, handelt es sich um ein Master-Slave-System [15, S.25].
Den letzten zu betrachtenden Punkt zum Thema Mikroprozessortechnik bilden die Begriffe des deterministischen Verhaltens und der Echtzeit. Ein deterministisches Verhalten liegt vor, wenn auf eine Eingangsinformation stets die selbe Ausgangsinformation folgt.
Die Echtzeit gibt an, dass ein Prozess definierte Fristen aufweist. Kommt es zur Verletzung dieser Fristen, folgen ernsthafte Konsequenzen oder Fehlfunktionen. Anhand dieser Konsequenzen lassen sich harte und weiche Echtzeit unterscheiden.
Tritt nach einer Fristüberschreitung nur eine Reduzierung der Leistungsfähigkeit auf, ist es eine weiche Echtzeitanforderung. Hat hingegen das Verpassen der Frist einen Totalausfall oder ernsthafte Fehlfunktionen zur Folge, handelt es sich um eine harte Echtzeitanforderung.
Eine Frist beschreibt den Zeitpunkt, zu dem in einem System eine Wirkung auf spezifische Umstände resultieren muss [15, S.337/338].

## 2.3 Kommunikation

Wie in Abschnitt 2.1.2 ausgeführt, beruht *Industrie 4.0* auf der Vernetzung von Prozesskomponenten. Diese Vernetzung kann mit unterschiedlichen Kommunikationstechniken hergestellt werden. Mit den folgenden Inhalten werden die Grundlagen der in diesem Projekt verwendeten Techniken aufgeführt.

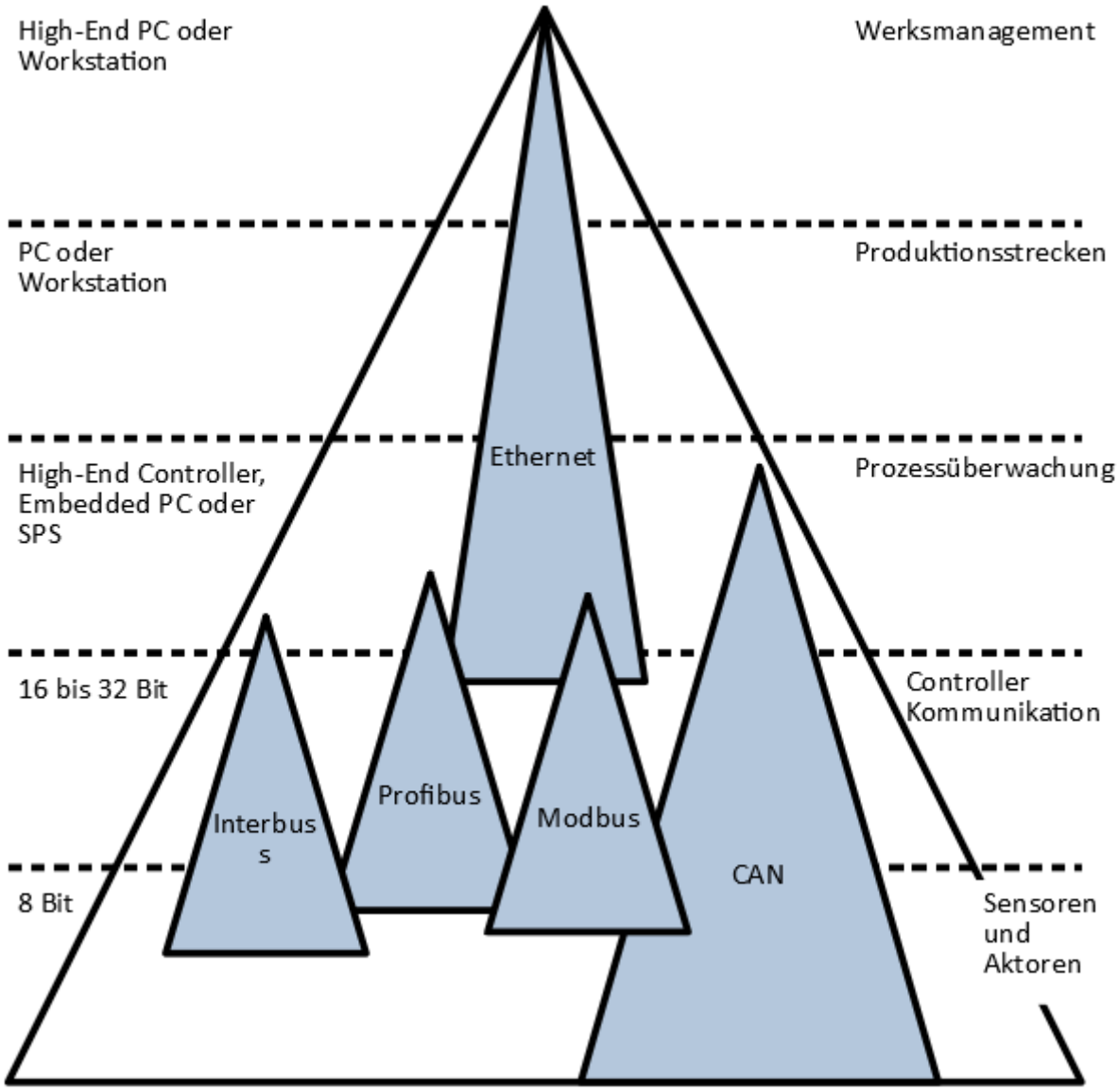



Abbildung 2.3: Automatisierungspyramide [3, S.5]

Bevor in den Kapiteln 2.3.1 bis 2.3.3 Angaben zu den Kommunikationstypen beschrieben werden, erfolgt eine Ausführung der Anwendungsebenen und des allgemeinen Aufbaus von Übertragungsprotokollen.
Netzwerke werden grundsätzlich auf Grund ihrer Ausdehnungsform eingeteilt. Eines ist das Lokal Area Network (LAN). Dieses Netzwerk zeichnet sich durch seine begrenzte

räumliche Ausdehnung aus. Der Einsatz von Hebezeugen und somit auch die Kommunikation diesbezüglich, findet in so einem begrenzten, industriellen Umfeld statt [4, S.10]. In diesem lokalen Raum befinden sich weiterhin unterschiedliche Prozessebenen. Eine Einteilung dieser ist in Abbildung 2.3 gegeben.
Die unterste Ebene dient der Ansteuerungen von Ausgängen und dem Einlesen von Eingängen. In der einfachsten Form werden in dieser Ebene Schalterzustände ausgelesen und Elektromotoren entsprechend angesteuert.
In der darüber liegenden Ebene steigt die Komplexität des Prozesses, indem die eingelesenen Signale mittels eines Programmalgorithmus ausgewertet werden. Die Ansteuerung der Ausgänge erfolgt dann auf Grundlage der resultierenden Ergebnisse.
Die Prozessüberwachungsebene kombiniert mehrere Komponenten aus der Controller-Ebene. Das bedeutet, dass ein Prozesskontrollsystem mehrere Untersysteme hat. Jedes dieser Systeme ist für die Bearbeitung einer Aufgabe zuständig.
Eine höher liegende Ebene umfasst immer eine Vielzahl von Systemen der darunter liegenden Schicht [3, S.8].
Aufgabe eines Netzwerks ist es, die Komponenten der unterschiedlichen Ebenen vertikal und gleicher Ebenen horizontal zu verbinden. Unabhängig vom Typ wird die Kommunikation des jeweiligen Netzwerks anhand eines Netzwerkmodells beschrieben.
Dem spezifischen Modell liegt das Open Systems Interconnection Referenzmodell (weiterhin mit OSI-Referenzmodel abgekürzt) zu Grunde. Das OSI-Referenzmodell beschreibt eine Standard-Netzwerkarchitektur, welche zur Vermittlung des wesentlichen Verständnisses der funktionellen Zusammenhänge beiträgt. Durch das Modell wird die Kommunikation in sieben Schichten unterteilt. Jede dieser Schichten stellt einen funktionellen Beitrag für den Ablauf der Kommunikation zur Verfügung [4, S. 23-26].
Durch die oberen drei Schichten wird die Anbindung zu den Netzwerkanwendungen und -anwendern hergestellt. Die Schichten eins bis vier sind für die Definition der Transportmechanismen und für die Datenübertragung verantwortlich [4, S.29].
Der Beitrag der Schichten zur Datenübertragung wird als Dienst bezeichnet. Ein Dienst einer oberen Schicht erweitert die Funktionalität eines Dienstes der darunterliegenden Schicht. Die Funktionen werden in jeder Schicht nach definierten Regeln ausgeführt. Diese Regeln werden als Protokolle bezeichnet. Für eine erfolgreiche Kommunikation müssen die Schicht und das Protokoll der Teilnehmer identisch sein.
Eine Kommunikation zwischen den selben Schichten, welche Peers genannt werden, gilt als virtuelle Kommunikation. Reell bewegen sich die Daten im Sender vertikal abwärts bis sie die Bitübertragungsschicht erreichen. Diese Schicht stellt einen Dienst bereit, durch den die Daten physikalisch zum anderen Teilnehmer übertragen werden. Über das physikalische Medium erreichen die Daten die Bitübertragungsschicht des Empfängers. In diesem steigt der Datenstrom, bis zum zugehörigen Peer vertikal auf [4, S. 23-26].
Der beschriebene Aufbau und der Datenfluss wird in Abbildung 2.4 visualisiert. Die Erläuterung der Schichten und Dienste wird in den folgenden drei Kapiteln spezifisch zu jeden Netzwerktyp ausgeführt.

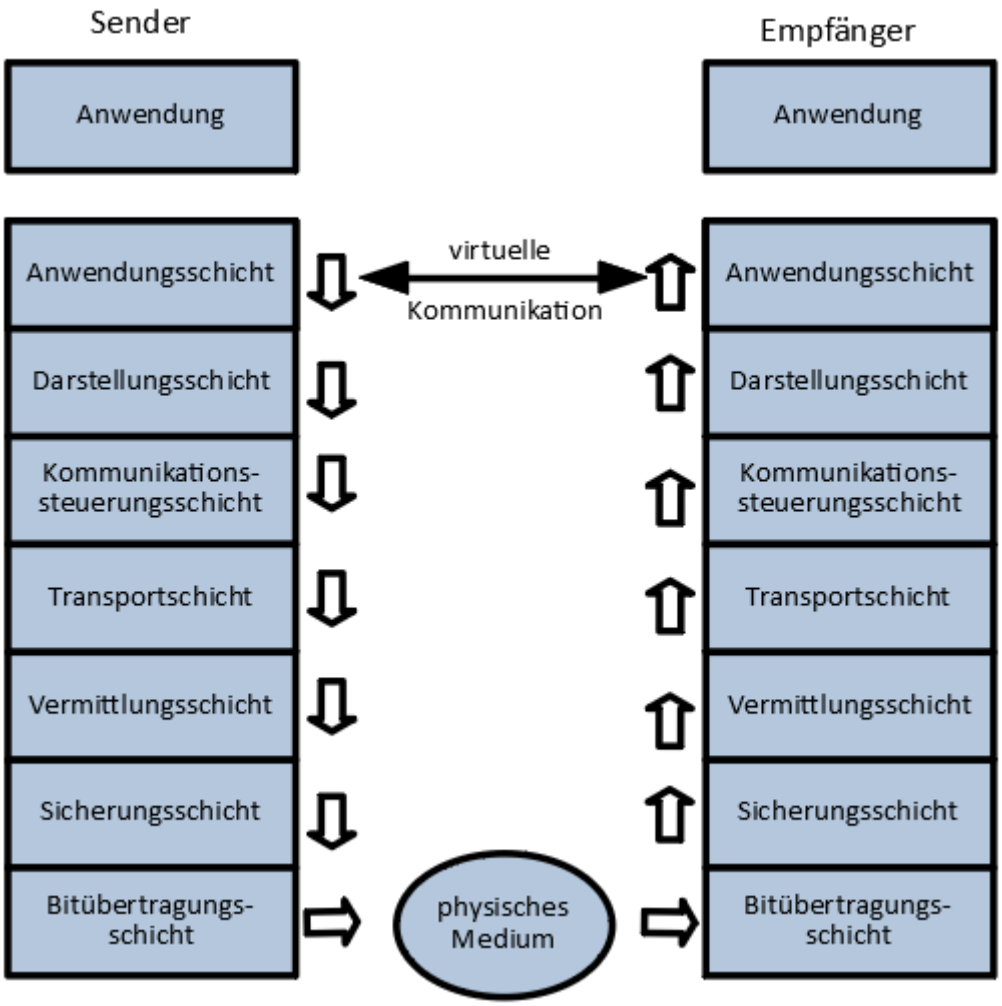



Abbildung 2.4: Kommunikationswege innerhalb des OSI-Referenzmodells [4, S.26]

### 2.3.1 CANopen

Ein Protokolltyp ist *CANopen*. Es ist für die Übertragungstechniken des Controller Area Network (im weiterführenden Verlauf mit CAN abgekürzt) entwickelt wurden. Über das CAN werden werden die ersten beiden Schichten im OSI-Referenzmodell beschrieben. [3, S. XV]. In diesem Kapitel sind die relevanten Funktionsprinzipien des Protokolls ausgeführt.
Die Grundlage für den *CANopen*-Standard bilden verschiedene Dokumente. Diese lassen sich unterscheiden in Geräteprofile, Anwendungsprofile und Frameworks. Allen zu Grunde liegt das Kommunikationsprofil *CiADS-301*. Der Aufbau und die Einordnung in das OSI-Referenzmodel ist in Abbildung 2.5 dargestellt.
Für die Kommunikation via *CANopen* bildet das Objektverzeichnis (im Weiteren als OD - Object Dictionary bezeichnet) die Basis. In ihm sind alle Parameter eines *CANopen*-Node abgebildet. Alle Daten in diesem Verzeichnis können von Peers gelesen und beschrieben werden.

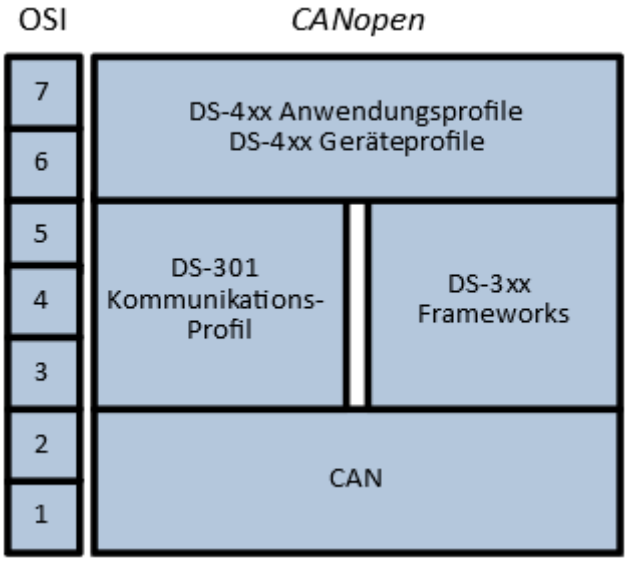



Abbildung 2.5: Einordnung *CANopen* in das OSI-Referenzmodel [3, S.18-20, 113-17]

Ein Zugriff auf die Daten kann entweder durch ein Service Data Object (Nachfolgend als SDO bezeichnet) oder ein Process Data Object (weiterführend mit PDO abgekürzt) erfolgen.
Ein Eintrag im OD besteht aus einem 16 Bit Index, einem 8 Bit Subindex, dem Datentyp und der Erläuterung des Eintrags. Die Einträge im OD werden hexadezimal, das heißt mit der Basis 16 angegeben. Jeder Index kann bis zu 256 Subindizes besitzen. Der grundsätzliche Aufbau eines OD ist in Tabelle 2.2 gegeben [3, S.42-44]. Die Einträge auf den Indizes 6000h bis 9FFFh sind die Parameter, welche zum ausgewählten Geräteprofil passen. Ein solcher Standard ist beispielsweise der *DSP402*. Dieser gilt für Geräte mit Aufgaben im Bereich der Bewegungssteuerung [3, S.114-117]. Der Austausch der Daten mittels SDO findet zwischen Client und Server statt. Ein Client fragt

| Index Range | Beschreibung |
|---|---|
| 0000h | reserviert |
| 0001h - 0FFFh | Datentypen |
| 1000h - 1FFFh | Einträge für die Kommunikation |
| 2000h - 5FFFh | Herstellerspezifisch |
| 6000h - 9FFFh | Spezifische Einträge je nach Geräteprofil |
| A000h - FFFFh | reserviert |

Tabelle 2.2: Organisation des OD [3, S.44/45]

einen Service an und ein Server liefert diesen. Im Falle der SDO-Kommunikation entspricht der Service einem OD Zugriff.
Für die präzise Adressierung eines Zugriffs erhält jede Nachricht eine Identifizierung (weiterführend mit COB ID abgekürzt). Die ID für das Versenden von Anfragen setzt sich standardmäßig aus 600h und der Kennung des Ziel-Nodes (Server) zusammen. Die auf den Zugriff folgende Antwort ist durch die Summe aus 580h und der entsprechenden Server ID gekennzeichnet. Für die Nummerierung der Teilnehmer steht der Bereich von 1 bis 127 zur Verfügung. Die Vernetzung des Client mit den Servern ist in Abbildung 2.6 gegeben.

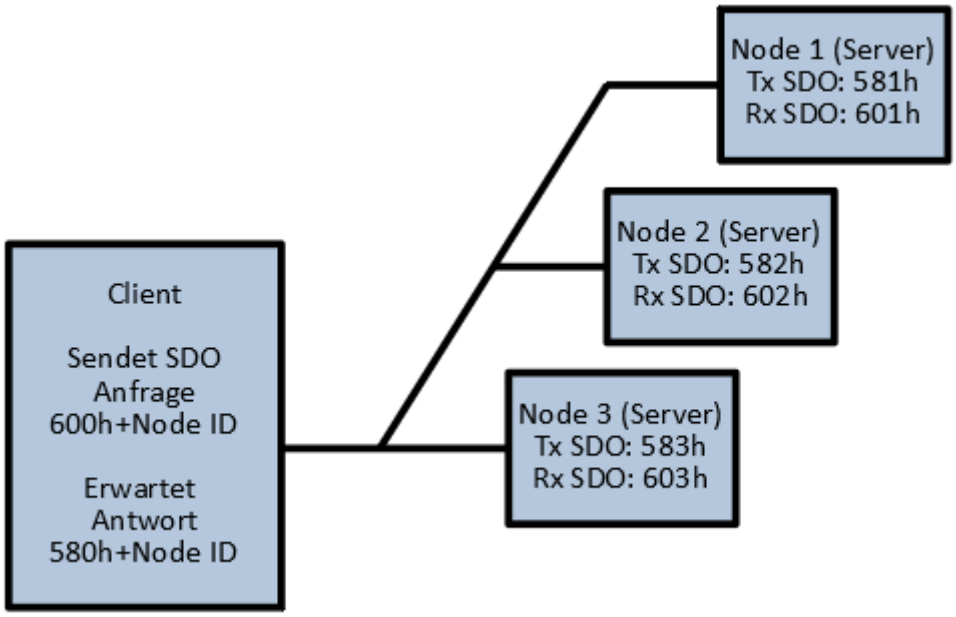



Abbildung 2.6: SDO Kommunikation [3, S.62]

Sowohl die Anfrage als auch die Antwort SDO umfassen die ID und acht Byte an Daten. Das erste Byte beinhaltet, um was für einen Zugriff es sich handelt und wie die Daten übertragen werden.
Die Datenübertragung ist abhängig von der Anzahl der zu übermittelnden Bytes. Ist derer Anzahl kleiner oder gleich vier kann der Transfer in einer SDO abgeschlossen werden. Handelt es sich um eine Übertragung mit mehr als vier Datenbytes, wird deren

Transfer auf mehrere Nachrichten aufgeteilt. Im typischen Fall enthalten die Bytes eins bis drei den Index und Subindex des Eintrages, auf den der Zugriff erfolgen soll. Auf den Zugriffseintrag folgen mit Byte vier bis sieben die Daten, welche übertragen werden sollen. Die Struktur des Datenpaketes ist in Abbildung 2.7 dargestellt.

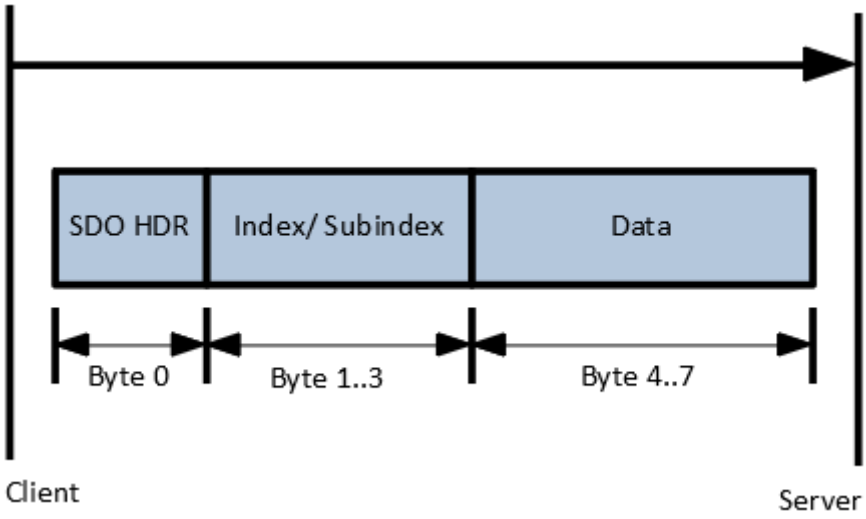



Abbildung 2.7: Datenpaket für einen SDO Zugriff [3, S.95-100]

Für das zyklische Übertragen von Daten in Prozessen werden PDOs verwendet. In diesen können mehrere Einträge aus dem OD in einer Nachricht versendet werden. Die PDOs werden in Transmit-PDOs(TPDO) und Receive-PDOs (RPDO) unterschieden. Jeder Teilnehmer kann bis zu vier dieser PDOs besitzen. Wie auch schon beim SDO setzt sich eine Nachricht aus der COB ID und acht Datenbytes zusammen.
Zur Übertragung der Nachricht mit einem PDO muss dieser initialisiert werden. Zu diesem Zweck ist es notwendig, im OD die Kommunikationsparameter und das Mapping festzulegen. In der Tabelle 2.3 ist dargestellt, auf welchen OD Index sich der jeweilige Initialisierungsparameter befindet und wie sich die jeweilige COB ID zusammensetzt.
Die Eingabe der Mappingparameter unterscheidet sich zwischen TPDO und RPDO. Durch die Zuweisung von Einträgen in einem TPDO werden diese anschließend gesendet. Beim Mapping von Einträgen des RPDO werden diese mit den Inhalten der empfangenen PDOs beschrieben. Die Auslösung einer TPDO kann durch vier verschiedene Ereignisse geschehen.
1. **Event**: Liegt eine Änderung eines Inputs vor und damit ein Update des OD, so wird ein PDO versendet. Durch diese Trigger-Methode erhält man sehr schnelle Antwortzeiten.
2. **Time**: Ein PDO wird nach dem Ablauf einer vorgegebenen Zeit versendet. Durch dieses Verfahren ist eine gute Kontrolle über die Busauslastung gegeben.
3. **Individual Polling**: Diese Möglichkeit ist eine Standard-CAN-Funktion. Die PDO wird nur nach Anfrage versendet.
4. **Sychronized**: Die PDO wird erst nach Erhalt des *Sync* Signals versendet. Sind mehrere PDOs von verschieden Nodes zu versenden, erfolgt die Übertragung seriell nach

Signaleingang [3, S.37/38].

| **Zusammen-stezung der COB ID** | **Kommunika-tionsobjekt** | **OD Index Kommunikations-parameter** | **OD Index Mapping-parameter** |
|---|---|---|---|
| 180h + Node ID | 1. TPDO | 1800h | 1A00h |
| 200h + Node ID | 1. RPDO | 1400h | 1600h |
| 280h + Node ID | 2. TPDO | 1801h | 1A01h |
| 300h + Node ID | 2. RPDO | 1401h | 1601h |
| 380h + Node ID | 3. TPDO | 1802h | 1A02h |
| 400h + Node ID | 3. RPDO | 1402h | 1602h |
| 480h + Node ID | 4. TPDO | 1803h | 1A03h |
| 500h + Node ID | 4. RPDO | 1403h | 1603h |

Tabelle 2.3: Zusammensetzung der PDO COB ID und Parameterindex [3, S. 51/75-78]

Eine Kommunikation über PDOs setzt voraus, dass sich der *CANopen*-Node in dem operativen Zustand befindet und verfügbar ist. Diese Voraussetzungen und die Überwachung sind Aufgaben des Netzwerk Management (weiterführend mit NMT abgekürzt). Innerhalb des NMT gibt es Services, die implementiert sein müssen. Dazu zählt die *CANopen*-Slave-Statemachine. Durch diese ist es dem Slave möglich, unterschiedliche Zustände einzunehmen. Der Übergang zwischen den Zuständen kann entweder automatisch erfolgen oder durch einen NMT-Master geregelt werden. Automatisch bedeutet, dass der Slave selbst seinen Zustand ändert.
Der Slave initialisiert nach dem Power-On sein *CANopen*-Interface, die zugehörige Kommunikation und sein OD. Nach diesem Vorgang und dem erfolgreichen Versenden seiner Boot-up Nachricht, wechselt er in den Zustand Pre-Operativ. In Systemen ohne NMT-Master wird der Wechsel, in den Zustand Operativ, automatisch ausgeführt. Die möglichen Zustände und wie der Slave in diese gelangt, wird auf der folgenden Seite durch Abbildung 2.8 dargestellt.
Die letzte NMT-Funktion ist das Versenden von Emergencies. Eine Emergency-Nachricht ist durch den CAN ID gekennzeichnet. Der Wert beträgt 80h zuzüglich der Node ID vom sendenden Slave. In den beinhalteten acht Datenbytes stellen die ersten zwei den *CANopen*-Errorcode dar. Das dritte Byte ist eine Kopie des Error-Registers 1001h, 00h. In den folgenden fünf Bytes können herstellerspezifische Fehlermeldungen eingetragen werden [3, S.83-88].

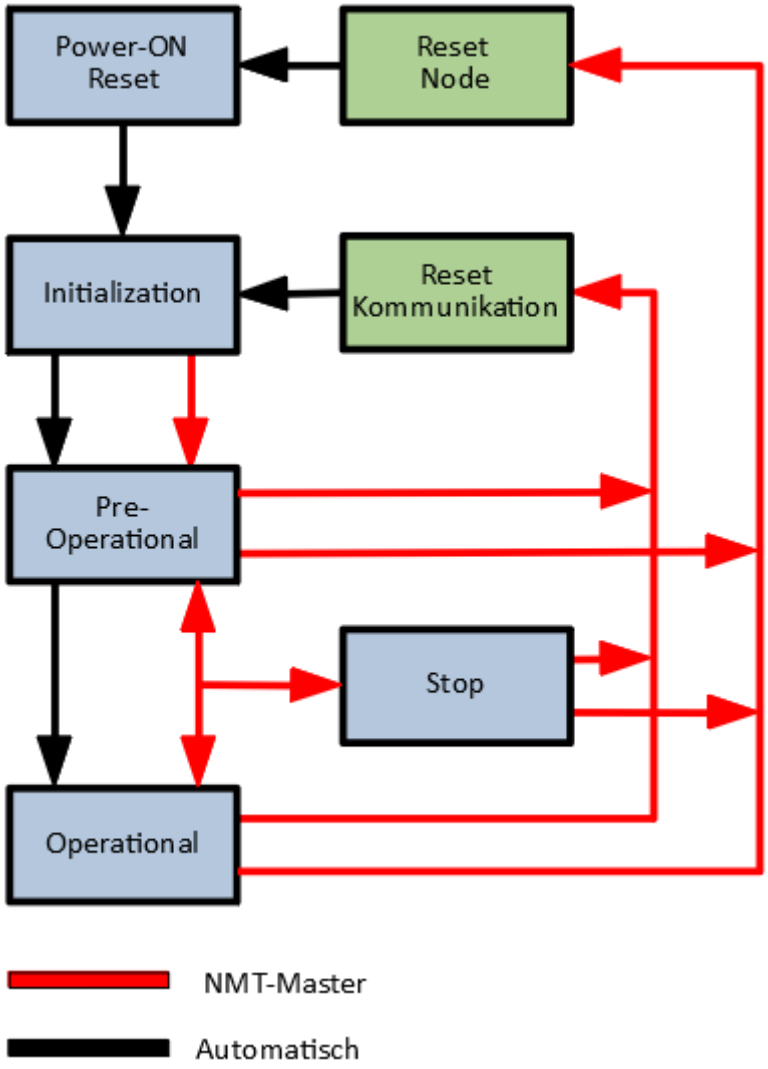



Abbildung 2.8: Standard State Machine nach *CiADS301* [3, S.84]

### 2.3.2 Ethernet-Standard

Über den Ethernet-Standard *IEEE 802.3* werden die Schichten eins und zwei des OSI-Referenzmodells beschrieben. Mit Hilfe dieser beiden Ebenen werden die Transportmechanismen definiert. Die Überführung des Standards in das OSI-Referenzmodell ist in Abbildung 2.9 dargestellt. Innerhalb des Standards sind unterschiedliche Normen vorhanden. Über diese werden die Geschwindigkeit, das Übertragungsmedium und das Übertragungsverfahren, der jeweiligen Ethernet-Verbindung beschrieben. Die *100BASE-TX*-Norm gibt beispielsweise an, dass eine Übertragung mit 100 Mbit/s im Basisbandübertragungsverfahren auf einem Twisted-Pair-Medium statt findet [4, S.29-37].
Unabhängig von der angewandten Norm durchlaufen die ausgetauschten Daten die verschiedenen Schichten des Standards. Den Kontakt zum Übertragungsmedium stellt das Medium Dependent Interface (MDI) her. Das MDI ist verantwortlich für die mechanische

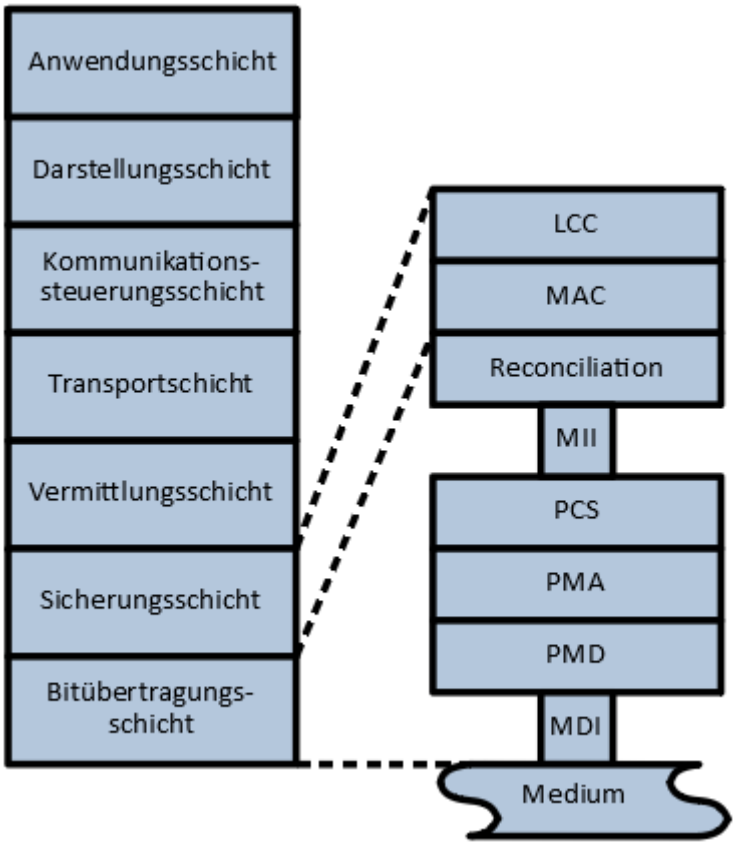



Abbildung 2.9: Einordnung Ethernet-Standard ins OSI-Referenzmodell [4, S.88]

Anbindung an das Übertragungsmedium und die physikalische Bereitstellung der Daten [4, S.41].
Über dem MDI folgt der Physical Medium Dependent Sublayer (PMD). Dieser ist die unterste, medienabhängige Teilschicht im Physical Layer. Der PMD-Layer kann innerhalb des Standards für das Medium Kupfer (*100BASE-TX*) oder Glasfaser (*100BASE-RX*) ausgeführt sein [4, S. 92]. Auf dieser Ebene werden die Informationen physikalisch an das Medium angepasst [27].
Das Physical Medium Attachment (PMA) bildet von oben betrachtet die erste Schicht, welche mit ihrer Funktion im Zusammenhang zum Medium steht. Die wesentlichen Funktionen dieser Einheit sind die Umwandlung der Daten in ein dem Übertragungsmedium entsprechendes Format, die Generierung der Kontrollsignale zur Anzeige der Verfügbarkeit des PMD und die Zurückgewinnung des Taktes aus dem empfangenen Signal [4, S.40/92].
Der weitere Weg in die übergeordneten Schichten führt in den Physical Coding Sublayer (PCS). Im PCS werden mehrere Aufgaben ausgeführt. Eine dieser Aufgaben ist die Codierung des Signals. Die Notwendigkeit der Codierung liegt im Übertragunsverfahren und dient zur Taktgewinnung. Des Weiteren werden in der PCS die Signale erzeugt, mit denen in der Media Access Control-Schicht (fortführend als MAC bezeichnet) das Zugriffsverfahren und die Kollisionsüberwachung gesteuert werden. Die letzte Aufga-

be, welche in diesem Layer realisiert wird, ist die Anpassung der Signale zwischen der PMA-Schicht und dem Media Independent Interface (MII).
Das MII stellt die Schnittstelle zum Medium dar. Die Platzierung des Interfaces ist entweder intern auf der Komponente oder nach außen geführt. Bei der Integration des MII in die Komponente fungiert es als Schnittstelle zu den unteren Schichten. Ist es nach außen geführt, kann ein Transceiver angeschlossen werden. Dieser kann dann direkt oder über ein genormtes 40-poliges MII-Kabel am MII angeschlossen werden.
Der Übergang von Physischer in die MAC-Schicht wird in der Reconciliation-Ebene durchgeführt. Bei diesem Übergang erfolgt eine Anpassung der Daten in zwei Richtungen. Daten, die vom MAC kommen, werden für die unteren Schichten aufbereitet. In der entgegengesetzten Richtung wird dafür Sorge getragen, dass die Daten für den MAC-Layer stehts die selbe Erscheinung haben [4, S.88-93].
Nach der Betrachtung des Ethernet-Standards auf physikalischer Ebene erfolgt die Definition und Beschreibung auf MAC-Ebene. Die Funktion dieser Schicht ist völlig unabhängig von der darunterliegenden Bitübertragungsschicht. Mittels der MAC-Ebene wird das zielgerichtete Senden und das Übertragen von Daten im Netzwerk definiert.
Zu den Inhalten gehören die Beschreibung des Zugriffsverfahrens mit der Kollisionserkennung sowie der Verfahrensweise beim Auftreten einer Kollision. Des Weiteren wird auf dieser Ebene definiert, in welchem Format (Frame-Format) die einzelnen Datenbytes angeordnet werden müssen, um eine zielgerichtete Datenübertragung zu gewährleisten. Ein weiterer wichtiger Bestandteil ist die Definition des Adressschemas. In diesem wird das Format für die Ethernet-Adressen festgelegt. [4, S.60/61].

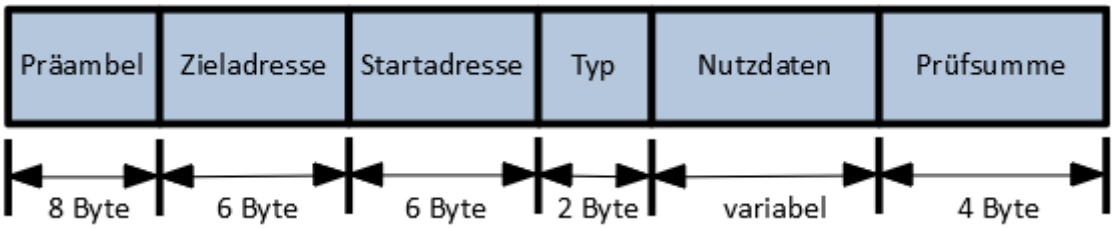



Abbildung 2.10: Standard Ethernet-Frame nach *IEEE 802.3* [5]

Ein Datenframe nach *802.3* besitzt sechs Komponenten. Diese Komponenten sind die Präambel, die Zieladresse, die Startadresse, der Protokolltyp, die Nutzdaten und die Prüfsequenz. Mit der Präambel wird die Übertragung synchronisiert. Die Adressen für den Start der Nachricht und das Ziel sind die Hardwareadressen der jeweiligen Komponenten. Der Inhalt des Feldes Typ gibt an, über welches Protokoll die höheren Schichten kommunizieren. Am Ende des Frames steht eine Prüfmenge, über die Fehler bei der Übertragung festgestellt werden [5]. Dargestellt ist der Aufbau in Abbildung 2.10.
Der Logical Link Control Layer (abgekürzt mit LLC) ist zuständig für die Auswertung des Protokolltyps. Aufbauend auf dem Ethernet-Standard ist das Protokoll zur virtuellen Kommunikation wählbar. Durch die Auswertung wird sichergestellt das nur Nachrichten der richtigen Sprache empfangen werden. [4, S.80/81].

### 2.3.3 Wireless Local Area Network

Durch das Wireless Local Area Network (weiterführend mit WLAN abgekürzt) werden die ersten zwei Schichten des OSI-Referenzmodells beschrieben. Grundlage für die Übertragung via *Luft* bildet die Norm *IEEE 802.11*. In dieser Norm werden auf Level der Bitübertragungsschicht die Funktionsprinzipien für das Frequenz-Hopping-Spred-Spectrum und Direct-Sequenz-Spred-Spectrum beschrieben [6, S. 10-13].
Mit dem Frequenz-Hopping-Spred-Spectrum wird der Zugriff auf die verschiedenen Frequenzbänder zeitlich eingeteilt. Das bedeutet, dass die Kommunikation nach einer definierten Zeit den Slot wechselt. Voraussetzung ist die Synchronisierung von Sender und Empfänger bei diesem Wechsel.
Für die Übertragung von größeren Datenraten wird das Direct-Sequenz-Spred-Spectrum verwendet. Über die Anwendung dieses Funktionsprinzips wird das zu übertragene Signal auf ein größeres Frequenzband aufgeteilt [6, S. 225-268].
Diese Standards wurden stetig weiterentwickelt und es entstanden neue Bitübertragungsschichten. Ein solcher Standard ist mit einem Buchstaben nach der Normkennzahl bezeichnet (zum Beispiel *IEEE 802.11g*). Um welche Änderung es sich im Detail handelt, kann spezifisch nachgelesen werden [6, S. 9-11].
Auf Höhe der Sicherungsschicht wird der MAC-Layer betrachtet.
Funktionell wird der physikalische Teil der Norm durch den Physical-Layer-Convergence-Procedur (mit PLCP abgekürzt) und dem PMD umgesetzt.
Durch den PLCP wird die Bitübertragungsschicht mit dem MAC verbunden. Außerdem ist er verantwortlich für das Erkennen von MAC-Frames auf dem Medium. Eine weitere Funktion ist das Hinzufügen von Feldern zu den vom MAC kommenden Frames. Diese hinzugefügten Felder enthalten Details, welche zum Senden und Empfangen der Nachricht notwendig sind. Das Senden ist Aufgabe des PMD. Die beschriebene Struktur ist in Abbildung 2.11 dargestellt [6, S. 10-14].

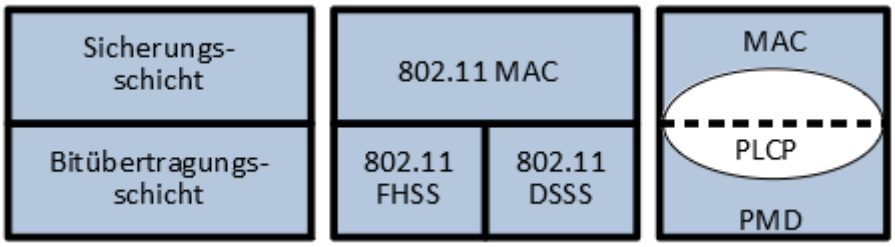



Abbildung 2.11: WLAN - Einordnung im OSI-Referenzmodell [6, S. 13/14]

Ein durch Anwendung der Norm entstandener Netzwerkknoten kann ein Einzelsystem sein oder sich in Verbindung zu anderen Netzwerken befinden. Handelt es sich um ein alleinstehendes System, wie zum Beispiel ein Laptop, so wird dieses als Station bezeichnet. Ein WLAN kann nur aus Stationen bestehen. Die Norm *IEEE 802.11* ist für die Kommunikation zwischen Stationen konzipiert wurden.
Ist an einer Station noch ein leitungsgebundenes Netzwerk angeschlossen, handelt es

sich um einen Access Point. Dieser konvertiert das wireless-Signal auf die Leitung. Solch eine Komponente stellt im Heimnetz beispielsweise der Router dar.
Das sich am Access Point befindliche Netzwerk wird als Distributionssystem bezeichnet. Dieses dient der Verteilung der mittels WLAN übertragenen Daten an ihrer Zieladresse. Eine Datenübertragung findet im WLAN-Netzwerk über ein leitungsungebundens Medium statt. Im Regelfall ist dieses Medium die uns umgebene Luft. Eine grafische Darstellung der Anordnung von Komponenten im WLAN ist in Abbildung 2.12 dargestellt [6, S. 15/16].

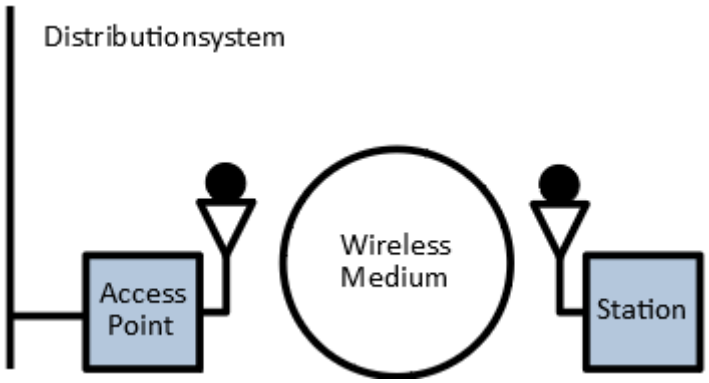



Abbildung 2.12: Netzwerkkomponenten eines WLAN [6, S. 15]

## 2.4 SPS

Der Funktionsablauf einer SPS ist in der Abbildung 2.13 gegeben. Zu Beginn eines Zyklus werden die Eingänge einmal eingelesen. Auf Basis des somit erhaltenen Prozessabbildes führt die SPS das gespeicherte Programm aus und aktualisiert die Ausgänge. Nach einer definierten Zeit wird der Ablauf wiederholt. Die Anzahl der Wiederholungen des gesamten Ablaufs ist unbegrenzt. Eine SPS folgt von seiner Wirkungsweise her dem EVA-Prinzip. Der **E**ingabe folgt die **V**erarbeitung und anschließend die **A**usgabe [7, S.9].
Um eine Steuerung für dieses Projekt auszuwählen ist es notwendig, die Komponenten zu kennen, welche einer SPS angehörig sind. Zunächst besitzt eine Steuerung eine gewisse Hardware. Diese gibt vor, wie viele digitale oder analoge Ein- beziehungsweise Ausgänge zur Verfügung stehen, welche Kommunikationsschnittstellen genutzt werden können und mit welchem $\mu$C der Programmablauf gesteuert wird.
Ein $\mu$C besteht allgemein aus einem Kern (Core), Arbeits- und Datenspeicher, Peripheriekomponenten und einem Interruptsystem. Die Komponenten sind über Systembusse im Chip miteinander verbunden [15, S.261]. Wird in den Kern eine Laufzeitumgebung implementiert entsteht im Zusammenhang mit der vorhandenen Peripherie ein Port. Die Laufzeitumgebung wird auch als Echtzeitbetriebsystem (im Englischen Real Time Operating System, was abgekürzt wird mit RTOS) bezeichnet.

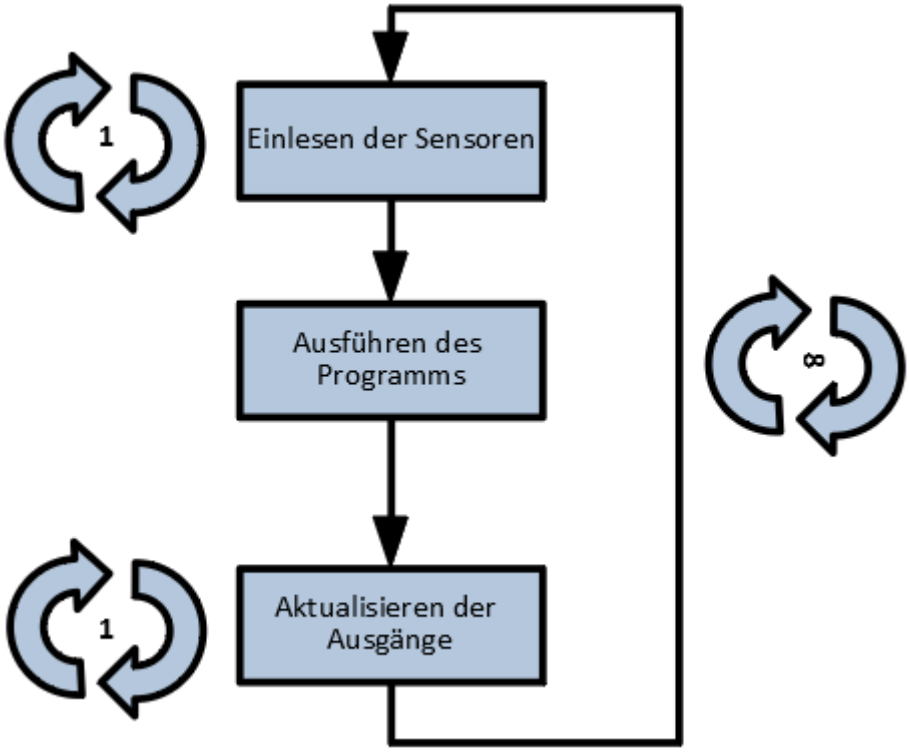



Abbildung 2.13: Ablaufschema SPS [7, S.9]

Das RTOS ist notwendig für die Realisierung eines Multitaskingsystems. Unter Multitasking ist die quasi-gleichzeitige Bearbeitung mehrerer Programmaufgaben auf dem Prozessor zu verstehen. Ein Beispiel für ein Multitaskingsystem ist Windows. Die Basis eines RTOS bildet ein Echtzeitkern. Eine Implementierung eines solchen Kernels mit der Erweiterung durch Dateisysteme, Netzwerkanbindungen oder komplexe Debugging-Werkzeuge ergibt ein Echtzeitbetriebssystem. Der Echtzeitkern hat die Aufgaben die Ressourcen des $\mu$C, in Abhängigkeit von Events, den einzelnen Tasks zuzuordnen. [15, S.342]. Das Funktionsprinzip ist auf der folgenden Seite in Abbildung 2.14 dargestellt. Passend zu dem jeweiligen Port ist ein verfügbares Software Developmentkit (kurz SDK) auszuwählen. Mit diesem werden Programmbibliotheken passend zum Systemumfeld erstellt und in den Prozessor integriert. Über die SDK wird entschieden, welche Programmiersprachen zur Verfügung stehen. Sind die anwendungsspezifischen Bibliotheken geschaffen, können diese in den Prozessor geladen werden.
In der Automatisierungstechnik weit verbreitet sind die Sprachen der *IEC 61131-3*. Dabei werden textuelle und grafische Sprachen unterschieden.

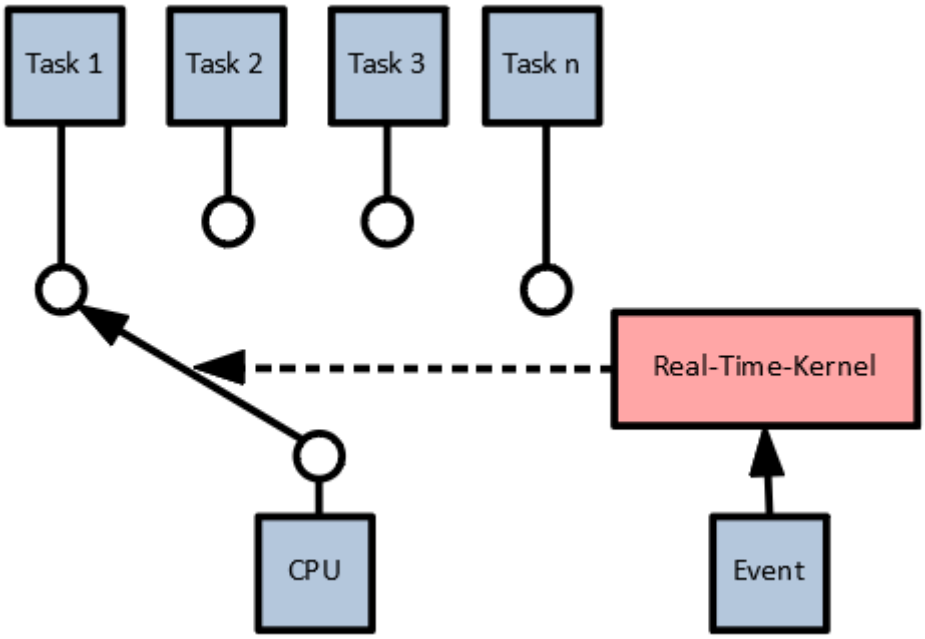



Abbildung 2.14: Funktionsschema eines *Real-Time-Kernel* [8]

# 3 Datenbezug

Für die Funktionalität eines Hebezeugs oder einer Krananwendung sind verschiedene Daten nötig. Die Menge der benötigten Daten und wie diese bezogen werden hängt von der spezifischen Anwendung ab. Dieses Kapitel bezieht sich auf Grundfunktionalitäten, die stets in einer Krananwendung beinhaltet sind.

## 3.1 Datenerhebung

Wie Eingangs im Kapitel 1 erwähnt wurde, werden die Inhalte dieses Konzeptes für den Betrieb eines Serienhebezeuges erarbeitet. Diese umfassen die notwendige Kommunikation zum Ausführen der vertikalen und horizontalen Bewegungen, sowie von Sonderfunktionen. Durch die nachfolgenden beiden Abschnitte wird erläutert, aus welcher Anforderung oder Funktion der jeweilige Datenbedarf resultiert.

### 3.1.1 Datenbedarfsanalyse

Eine Komponente des Hebezeus, als auch des *eDrives* ist der Elektromotor. Durch diesen ist die entsprechende horizontale oder vertikale Bewegung ausführbar. Zu diesem Zweck ist die Übertragung von Daten erforderlich. Außer der Bewegung von Nutzlasten sind noch andere Funktionen und Anforderungen zu beachten. Diese verursachen ebenfalls Daten oder einen Datenbedarf. Um welche Funktionen oder Anforderungen es sich im Detail handelt, wird im Folgenden betrachtet.
Hebezeuge sind eine Untergruppe von Fördermaschinen [3]. Diese haben die Aufgabe stoffliche Güter zu transportieren. Mit der Ausführung der Bewegung unterliegt das Hebezeug einer Beanspruchung. Für einen sicheren Betrieb des Hebezeuges ist es wichtig, alle Funktionselemente regelmäßig zu kontrollieren. Da trotz regelmäßiger Kontrolle nicht alle Schäden erkannt werden können, ist nach Ablauf einer bestimmten Beanspruchungszeit eine Generalüberholung (weiterführend mit GÜ abgekürzt) durchzuführen. Ziel einer GÜ ist das Erreichen einer sicheren Betriebsperiode (nachfolgend mit SWP abgekürzt). Eine SWP endet mit dem Erreichen der theoretischen Nutzungsdauer. Diese theoretische Nutzungsdauer ist die nach *FEM 9.511* ermittelte rechnerische Gesamtlaufzeit und bezieht sich auf die Nutzung der Maschine innerhalb eines Zeitraumes von zehn Jahren. Ist die Bedingung nach Formel 3.1 erfüllt, befindet sich das Hebezeug außerhalb seiner SWP. Daraus folgt, die Gesamtlaufzeit ist erreicht beziehungsweise der Abnutzungsvorrat ist aufgebraucht [10].

$$\frac{\text{tatsächliche Nutzung } \mathbf{S}}{\text{theoretische Nutzung } \mathbf{D}} >= 1 \tag{3.1}$$

[3] siehe 2.1.1

Die tatsächliche Nutzungsdauer (S) ergibt sich aus der Summe der Einzelprodukte, des tatsächlichen Belastungsfaktors **$km_i$** mit den dazugehörigen effektiven Betriebsstunden **$T_i$**. In Formel 3.2 ist dieser Zusammenhang dargestellt [10].

$$S = \sum_{i=1}^{n} (Km_i \cdot T_i) >= D \tag{3.2}$$

Der Zeitraum der theoretischen Nutzungsdauer ist abhängig von der Triebwerksgruppe und dem Lastkollektiv. Durch Kenntnis dieser beiden Parameter kann mittels der Tabelle 3.1 der vorhandene Abnutzungsvorrat ermittelt werden. Die Triebwerksgruppe wird vom Hersteller der Maschine angegeben. Der Belastungsfaktor ist abhängig von der Belastung, welche aus der Aufgabe resultiert [10].

| Triebwerksgruppe | $1D_m$ M1 | $1C_m$ M2 | $1B_m$ M3 | $1A_m$ M4 | $2_m$ M5 | $3_m$ M6 | $4_m$ M7 | $5_m$ M8 |
|---|---|---|---|---|---|---|---|---|
| Lastkollektive/ Faktor des Belastungsspektrums | theoretische Nutzungsdauer **D** [h] | | | | | | | |
| L1 ($Km_1$=0,125) | 800 | 1600 | 3200 | 6300 | 12500 | 25000 | 50000 | 100000 |
| L2 ($Km_2$=0,25) | 400 | 800 | 1600 | 3200 | 6300 | 12500 | 25000 | 50000 |
| L3 ($Km_3$=0,5) | 200 | 400 | 800 | 1600 | 3200 | 6300 | 12500 | 25000 |
| L4 ($Km_4$=1) | 100 | 200 | 400 | 800 | 1600 | 3200 | 6300 | 12500 |

Tabelle 3.1: Theoretische Nutzungsdauer [10]

Die Einstufung der Beanspruchung, die aus der Hubarbeit resultiert, basiert auf dem kubischen Mittel. Eine Bestimmung des kubischen Mittels unterscheidet sich je nach Gewicht des Tragemittels. Zum Tragmittel gehören alle Komponenten vom Seil oder der Kette bis zum Lasthaken. Ist die Bedingung des Verhältnisses 3.3 erfüllt, folgt Formel 3.4.

$$\frac{\text{Gewicht des Tragmittels}}{\text{Tragfähigkeit}} <= 0,05 \tag{3.3}$$

$$k = \sqrt[3]{(\beta_1 + \gamma)^3 \cdot t_1 + (\beta_2 + \gamma)^2 \cdot t_2 + ... + \gamma^3 \cdot t_\Delta} \tag{3.4}$$

Anderen Falls wird das kubische Mittel nach Formel 3.5 berechnet.

$$k = \delta \sqrt[3]{(\beta_1 + \gamma + \alpha)^3 \cdot t_1 + (\beta_2 + \gamma + \alpha)^2 \cdot t_2 + ... + (\gamma + \alpha)^3 \cdot t_\Delta} \tag{3.5}$$

$$t = \frac{\text{Laufzeit mit Nutz- oder Teillast und Totlast}}{\text{Gesamtlaufzeit}} \tag{3.6}$$

$$\beta = \frac{\text{Nutz- oder Teillast}}{\text{Tragfähigkeit}} \tag{3.7}$$

$$\gamma = \frac{\text{Totlast}}{\text{Tragfähigkeit}} \tag{3.8}$$

$$t_\Delta = \frac{\text{Laufzeit nur mit Totlast}}{\text{Gesamtlaufzeit}} \tag{3.9}$$

$$\alpha = \frac{\text{Gewicht des Tragmittels}}{\text{Tragfähigkeit}} \tag{3.10}$$

$$\delta = \frac{\text{Tragfähigkeit}}{\text{Hubkraft}} \tag{3.11}$$

Die Berechnungen, der Bestandteile vom kubischen Mittel, sind in den Formeln 3.6 bis 3.11 gegeben. In der Tabelle 3.2 sind die Definitionen der verwendeten Begriffe enthalten.

| | |
|---|---|
| Tragfähigkeit [kg] oder [t] | größte Last, welche gehoben werden kann |
| Nutzlast [kg] oder [t] | größte Nutzlast, die gehoben werden kann |
| Totlast [kg] oder [t] | Summe des Gewichts aus Lastaufnahme- und Anschlagmittel |
| Teillast [kg] oder [t] | Anteil der Nutzlast |
| Hubkraft [N] oder [kN] | Kraft, die der Masse aus größter Last und Tragmittel entspricht |

Tabelle 3.2: Definition der Begriffe für Errechnung des kubischen Mittels [9]

Die Anordnung der beteiligten Komponenten eines Hubwerks ist in Abbildung 3.1 dargestellt.

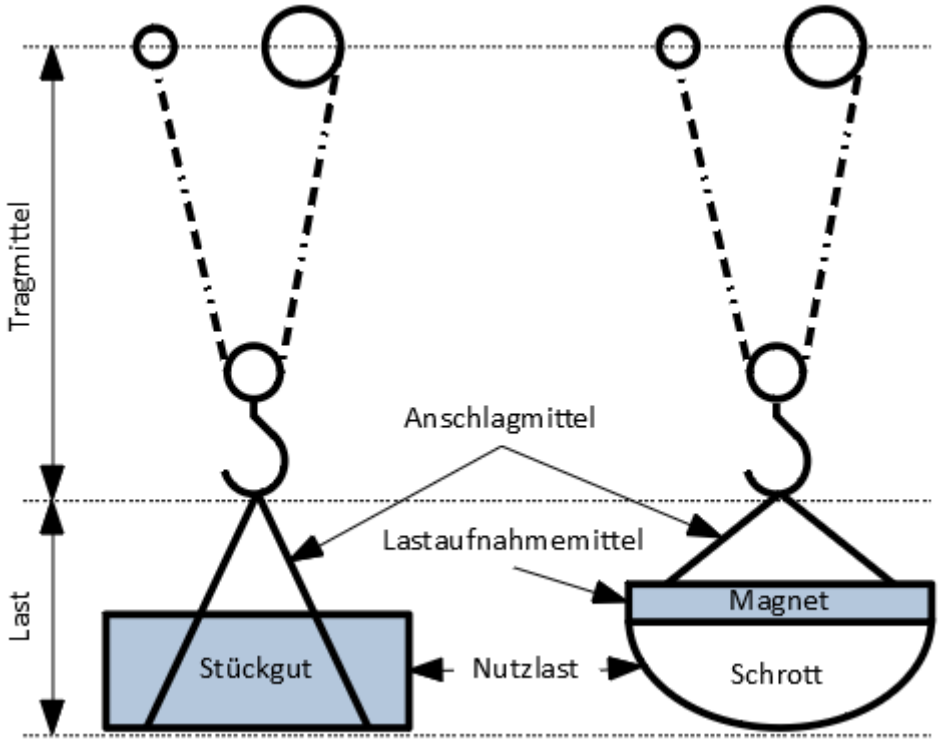



Abbildung 3.1: Schema der Komponenten eines Hubwerkes [9]

Mit dem erhaltenem kubischen Mittelwert kann der Betrieb einem Lastkollektiv in Tabelle 3.3 zugeordnet werden. Die Beanspruchung des Triebwerks wird in Totlast, geringe Last, mittlere Last und maximale Last unterschieden. Welchen Anteil der jeweilige Lastfall an den einzelnen Lastkollektiven besitzt, ergibt sich aus der zweiten Spalte der Tabelle 3.3 [9]. Ist die Beanspruchung ermittelt und die Triebwerksgruppe bekannt, kann der Abnutzungsvorrat aus Tabelle 3.1 abgelesen werden.

| **Lastkollektiv** | | **Begriffsbestimmung** | **kubischer Mittelwert** |
|---|---|---|---|
| **FEM** | **ISO** | | |
| 1 (leicht) | L1 | ausnahmsweise Höchstbeanspruchung; laufend sehr geringe Beanspruchung | k<=0,5 |
| 2 (mittel) | L2 | ziemlich oft Höchstbeanspruchung; laufend geringe Beanspruchung | 0,5<k<=0,63 |
| 3 (schwer) | L3 | häufig Höchstbeanspruchung; laufend mittlere Beanspruchung | 0,63<k<=0,80 |
| 4 (sehr schwer) | L4 | regelmäßig Höchstbeanspruchung | 0,80<k<=1 |

Tabelle 3.3: Lastkollektivklassen [9]

Durch die Erfassung der Echtzeitdaten kann die theoretische Nutzungsdauer dynamisch überprüft und angepasst werden. Eine Protokollierung mit Echtzeitdaten bietet den Vorteil, dass die GÜ entsprechend der Nutzung ausgeführt wird. Die Protokollierung mittels aktuellen Daten ist mit der Protokollierung 3.1 in der *9.755* beschrieben.
Mit der Abbildung 3.2 wird der Wartungsverlauf eines Hubwerkes über zwei SWPs dargestellt. Kurve eins zeigt die Nutzung nach der theoretischen Einstufung. Der Verlauf zwei stellt eine variable Nutzung dar [10] [4].
Zusätzlich zur Errechnung des restlichen Abnutzungsvorrates sollen auch die Wartungsverläufe und andere Ereignisse aufgezeichnet werden. Was im spezifischen Fall ein Ereignis von Interesse ist, wird vom Hersteller definiert. Dies kann beispielsweise die Überschreitung der maximalen Stromaufnahme des Motors, eine Parametrierung des Hebezeugs oder eine Wartung sein. Ein Auftreten eines solchen Ereignisses ist datiert zu hinterlegen.
Neben den Funktionen zur Aufzeichnung von Ausnahmen und Berechnung der Nutzungsdauer ist die Realisierung weiterer Funktionen vorgesehen, wofür Daten nötig sind. Eine dieser Funktionen ist die Einstellung von Benutzeranschlägen. Diese sind notwendig, wenn der Hakenweg innerhalb der maximalen und minimalen Länge extra begrenzt werden soll. Mit der Einstellung soll verhindert werden, dass zum Beispiel das Bedienelement oder die Last selbst außerhalb der Zugriffsreichweite abgestellt oder das Hebezeug auf den Boden gefahren wird. Zur Erfassung dieser Positionen wird der Hakenweg benötigt.

[4] Wie der Verbrauch des Abnutzungsvorrates spezifisch in diesen Konzept ermittelt wird, ist Inhalt des Abschnittes 3.2.1.

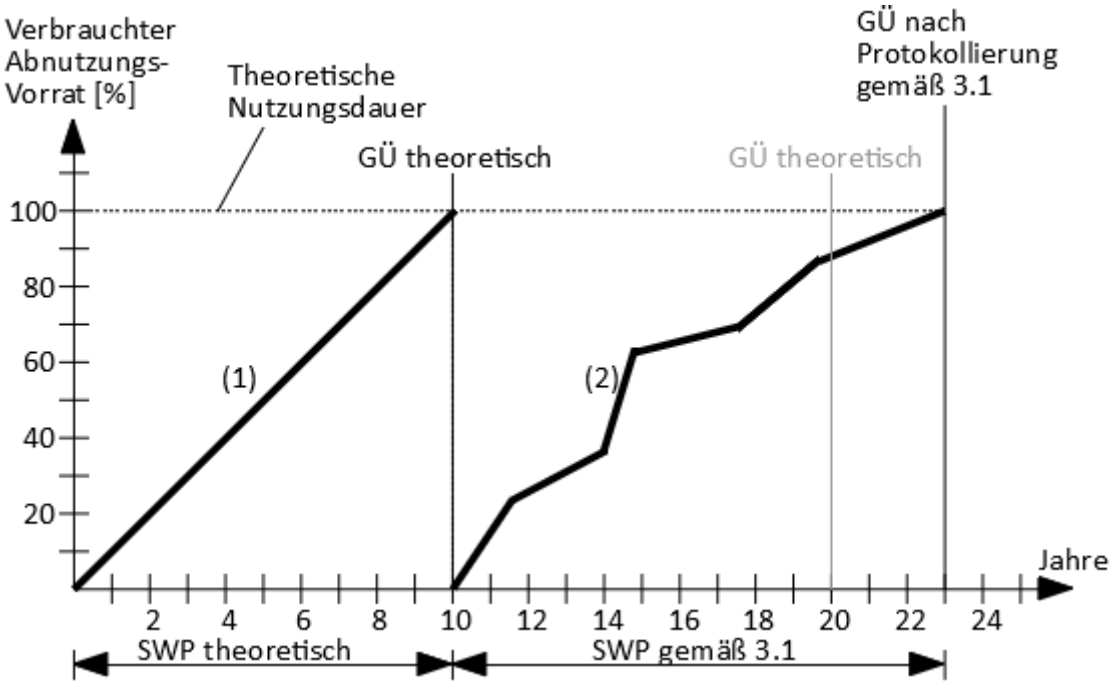



Abbildung 3.2: Wartungsverlauf innerhalb der Nutzungsdauer [10]

Mit der Kenntnis zur aktuellen Hakenposition sind auch das Tragemittel betreffende Funktionen realisierbar. Mit einer *Schlaffseilerkennung* ist es nicht mehr möglich, die Last weiter absenken zu können, wenn das Hebezeug diese bereits nicht mehr trägt. Für diese Funktion wird zuzüglich zum Hakenweg die Last benötigt. Durch die Vermeidung eines schlaffen Seils ist eine Reduzierung der Unfallgefahr gegeben.
Eine andere Funktionalität ist die Anpassung der Geschwindigkeit an gewisse Umweltgegebenheiten. Beispielsweise soll kurz vor dem Erreichen des Benutzeranschlages die Geschwindigkeit gesenkt werden. Für diese Erfassung wird die aktuelle Geschwindigkeit des Lasthakens benötigt.
Der letzte zu erfassende Parameter ist die Nutzlastposition. In einer Portalanlage kann die anhängende Nutzlast dreidimensional im Raum bewegt werden. Durch die damit verbundene horizontale Beweglichkeit, handelt es sich somit um einen Kran [5]. Auf der nächsten Seite in Abbildung 3.3 ist eine solche Krananlage dargestellt. Die Z-Koordinate entspricht dem aktuellen Hakenweg. Eine Bewegung in X- oder Y-Richtung kann durch manuelles Verschieben oder automatisch ausgeführt werden. Für eine automatische Positionierung sind zwei *eDrive*-Fahrwerke erforderlich.
In der Tabelle 3.4 auf der folgenden Seite sind alle benötigten Parameter in Bezug auf die Funktion, in welcher sie benötigt werden, aufgelistet.

[5] Definition aus Kapitel 2.1.1

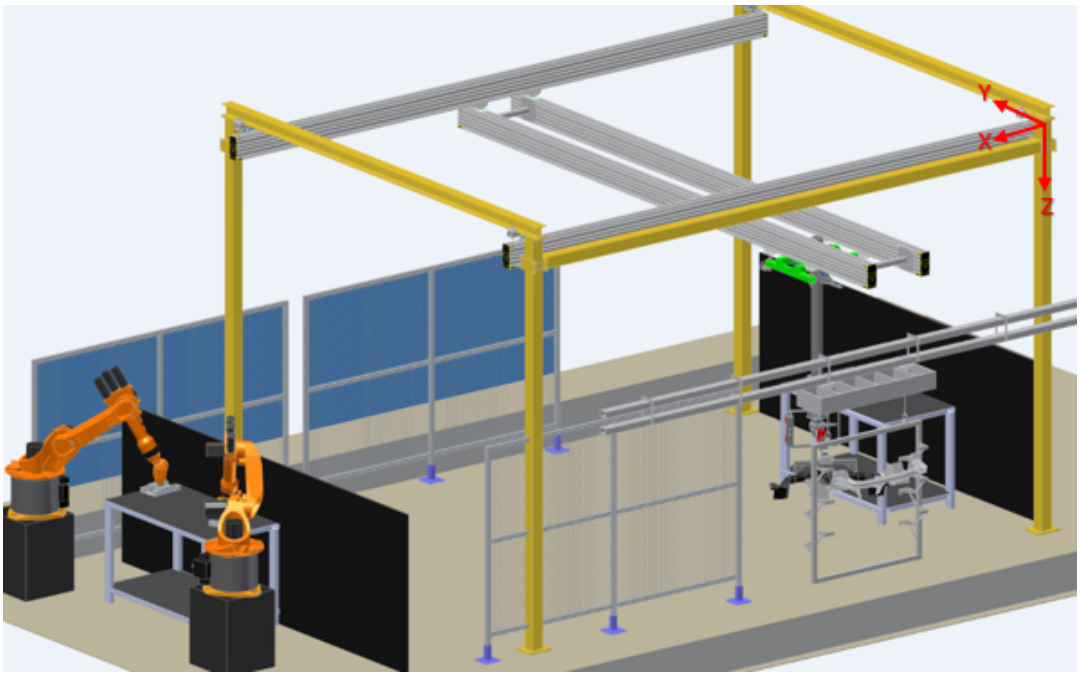



Abbildung 3.3: Einsatz einer Portalanlage

| **Funktion** | **benötigte(r) Parameter** |
|---|---|
| Berechnung der theoretischen und tatsächlichen Nutzungsdauer | mechanische Last, Zeit |
| Aufzeichnung der Ausnahmen | elektrische Daten des Antriebs, mechanische Last, Koordinaten, Zeitpunkt der Aufzeichnung |
| *Schlaffseilerkennung* | Hakenweg, mechanische Last |
| Positionserfassung | X, Y, Z-Koordinaten |
| Geschwindigkeits-regelung | Position, Geschwindigkeit |
| Benutzeranschläge | Hakenposition |

Tabelle 3.4: Benötigte Daten zum ausführen der Funktionen

### 3.1.2 Datengewinnung

In diesem Kapitel wird aufgezeigt, wie die im vorhergehenden Abschnitt aufgeführten Daten erfasst werden. Als erster Schritt wird geprüft, ob im aktuellen System der entsprechende Parameter schon erfasst wird. Wenn die Erfassung vorhanden ist, folgt die Analyse des derzeitigen Beschaffungsweges. Daraus resultiert, ob die Erfassung weiterhin so durchgeführt werden kann oder ob es einer Änderung bedarf.
Ist die Aufnahme des jeweiligen Wertes noch nicht vorgesehen, so werden Wege betrachtet, um dies zu realisieren. Das Verfahren ist in Abbildung 3.4 schematisch dargestellt.

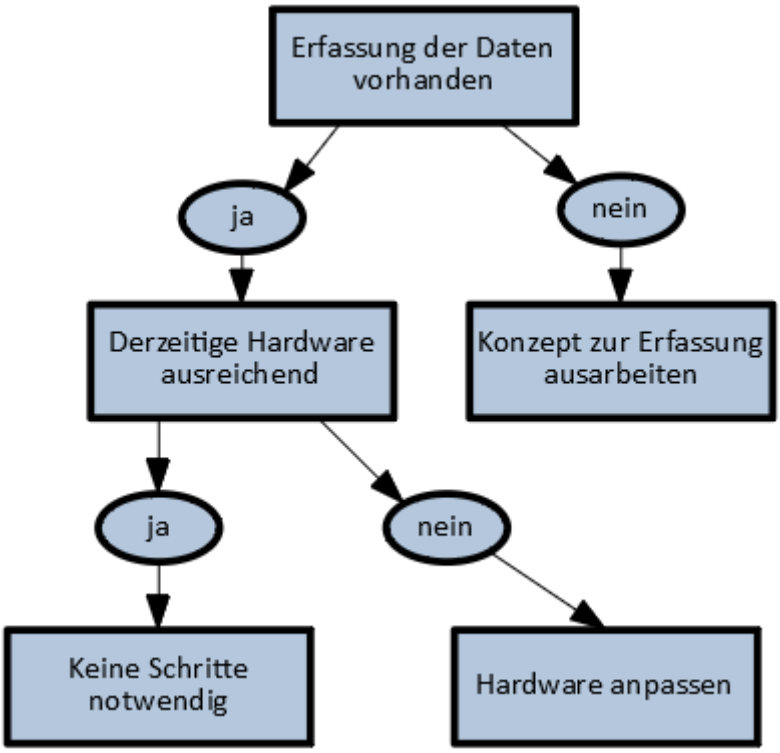



Abbildung 3.4: Schematisches Vorgehen zur Datenerhebung

Der erste betrachtete Parameter ist der Hakenweg. Im derzeitigen System wird die Seil- oder Kettenlänge nicht erfasst. Aus diesem Grund muss nach Schema 3.4 ein Konzept für die Erfassung erstellt werden.
Der Hakenweg ist die Entfernung, die zwischen der aktuellen Position und dem oberen Anschlag zurückgelegt wurde. Dieser ist durch direkte oder indirekte Methode erfassbar. Dabei ist das Messergebnis bei der direkten Messung, der absolute Abstand vom oberen Anschlag zur Last. Unter der indirekten Messung ist das Verhältnis von Umdrehung zum Hakenweg zu verstehen.
Dargestellt ist der Hakenweg in Abbildung 3.5.

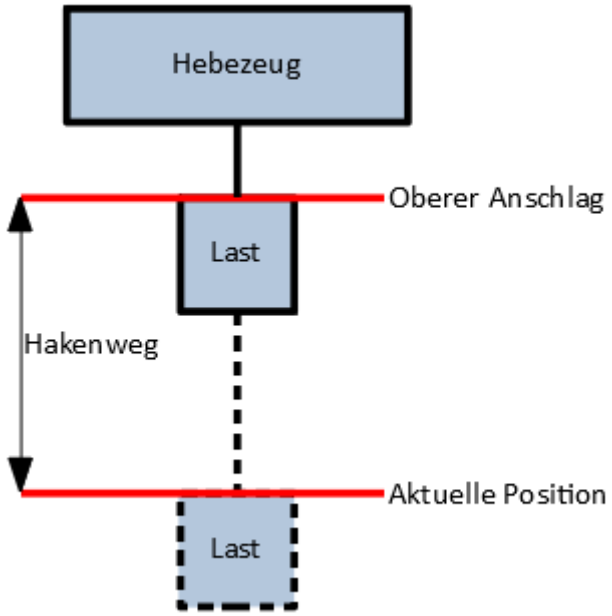



Abbildung 3.5: Darstellung der Definition Hakenweg

Auf Grund des Umfeldes ist eine direkte Messung nicht möglich. In der Messstrecke befindet sich mindestens das Tragmedium und eine elektrische Leitung. Diese könnten zur Verfälschung der Messung führen. Zusätzlich kann die Strecke durch menschliches Eingreifen und Schmutz gestört werden. Mit menschlichen Eingreifen lassen sich zwei unterschiedliche Szenarien umschreiben. Zum einen ist es möglich, während des Betriebes in die Strecke, also an Tragmedium oder Kabel zu greifen und somit die erfasste Reflexionsweite zu verfälschen, zum anderen ist die ausgeführte vertikale Bewegung nicht geführt. Dadurch kann beim Führen oder Positionieren der Last ein zu erfassender Punkt außer Sicht geraten und somit ein falscher Hakenweg erfasst werden.
Diese Einflüsse sind bei einer indirekten Messung nicht von Bedeutung. Zur Ermittlung des Hakenweges über dieses Verfahren wird zunächst der Antriebsstrang des Hebezeugs betrachtet. Mit einem Servomotor, einem Getriebe und einem Servoregler wird die Last vertikal bewegt. Ein Servomotor zeichnet sich dadurch aus, dass er seine Winkelposition zurückmelden kann. Über die zukünftige Vernetzung innerhalb des Hebezeugs, sind diese Daten für alle Komponenten zugänglich [6].
Mit dem ermittelbaren, zurückgelegten Winkel kann allerdings nicht direkt auf den Hakenweg geschlossen werden. Ein Grund dafür ist das eingebaute Getriebe. Einfluss auf den Hakenweg nimmt dieses Bauteil durch seine Übersetzung. Durch diese entspricht der Winkel des Motors nicht zwangsläufig dem Winkel des Seilträgers.
Das Medium worauf, das Seil aufgewickelt wird, bildet den zweiten Einflussfaktor. Dabei ist zwischen Seiltrommel und Seilscheibe zu unterscheiden. In der Abbildung 3.6 ist die Trommel links und die Scheibe rechts dargestellt. Auf der Seiltrommel wird das Seil parallel aufgewickelt. Dadurch bleibt der Wickelradius über die gesamte Seillänge

[6] Die detaillierte Vernetzung wird in Kapitel 4.1.1

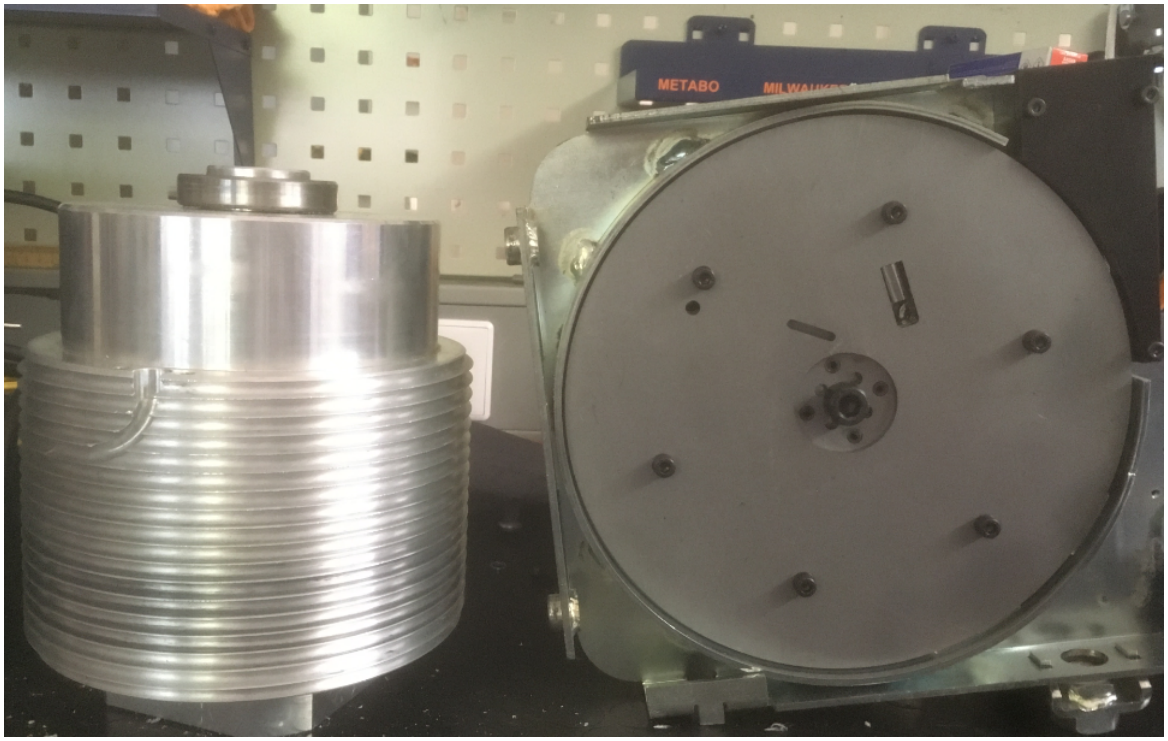



Abbildung 3.6: Seiltrommel und Seilscheibe

gleich. Aus der Formel 3.12 ist der Hakenweg bei bekannten Radius der Seiltrommel und Drehwinkel am Getriebeausgang ermittelbar. Die Bogenlänge b entspricht dabei dem Hakenweg.

$$b = \pi \cdot r_{Seiltrommel} \cdot \frac{\alpha_{Getriebeausgang}}{180^\circ} \tag{3.12}$$

Komplexer gestaltet sich die Berechnung bei der Verwendung einer Scheibe als Seilspeicher. Das Seil wird spiralförmig aufgewickelt. Dadurch verändert sich der Radius vom Scheibenmittelpunkt zum Seil mit jeder Winkeländerung. Das bedeutet, dass der Radius r eine Funktion von $\alpha$ ist. Bekannt sind der minimale und maximale Radius, sowie die Seilstärke. Der minimale Radius ergibt sich bei komplett abgewickelten Seil. Für den maximalen Radius muss der Hakenweg den kleinst möglichen Wert annehmen. Mit der Seilstärke ergibt sich der Abstand zwischen den einzelnen Lagen. Inwiefern sich dieser Abstand durch das Anhängen einer Last verändert und somit auch der resultierende Hakenweg, muss getestet werden. Für die weitere Betrachtung wird der Abstand als konstant angenommen.
Durch diese Annahme und der spiralförmigen Anordnung vom aufgewickeltem Seil ergibt sich eine Archimedische Spirale. Die Bogenlänge wird bei einer solchen Spirale nach Formel 3.13 berechnet.

$$b = \frac{a}{2}(\alpha\sqrt{\alpha^2+1} + \ln(\alpha + \sqrt{\alpha^2+1}) \tag{3.13}$$

Das in der Formel verwendete *a* ergibt sich aus der Berechnung 3.14. Diese ist aus der

Definition abgeleitet, dass der Abstand zwischen zwei Schnittpunkten zweier benachbarter Spiralwindungen und einer Geraden gleich $2\pi a$ ist [28, S. 106/107].

$$a = \frac{h_{Seil}}{2 \cdot \pi} \tag{3.14}$$

Bei der Seilscheibe ist zu beachten, dass das Seil auf einer Grundscheibe mit einem gewissen Radius aufgewickelt ist. Daraus und aus der Definition des Hakenwegs folgt, dass der Hakenweg Null die Summe der maximalen Anzahl von Windungen zuzüglich des Grundradius ist.
Weiterführend ist für die Berechnung des Hakenwegs Null die Anzahl der in den maximalen Radius passenden Windungen notwendig. Diese ergibt sich aus Formel 3.15.

$$Windungsanzahl = \frac{Maximalradius}{Seildurchmesser} \tag{3.15}$$

Mit der Kenntnis darüber, dass eine Seilstärke einem Drehwinkel von 360° entspricht, ist $\alpha$ für den Nullpunkt errechenbar. Mit diesem Wert kann eine Bogenlänge der Spirale mit maximalen Drehwinkel errechnet werden. Damit ergibt sich der aktuelle Hakenweg bei Nutzung einer Seilscheibe aus der Differenz zwischen der Bogenlänge des maximalen Drehwinkels und der des aktuellen. Eine schematische Darstellung der einzelnen Größen befindet sich in Abbildung 3.7.

$$Hakenweg = b_{maximalerDrehwinkel} - b_{aktuellerDrehwinkel} \tag{3.16}$$

Die Erfassung der aktuellen Position in X- und Y- Koordinate erfolgt nach dem Prinzip der Hakenwegmessung mit Seiltrommel. Im Fahrwerk *eDrive* lässt sich der Drehwinkel des Motors wie beim Hebezeug auslesen. Die Berechnung der zurückgelegten Strecke erfolgt nach der Formel 3.12. Im Falle des *eDrive* ist der Radius die Strecke zwischen Achsenmittelpunkt des Reibrades und der Kontaktstelle zur Schiene.
Für die Realisierung der Funktionen *Aufzeichnung von Ausnahmen* und *Berechnung des Abnutzungsvorrates* ist eine Zeiterfassung nötig.
Bei der Aufzeichnung von Situationen ist es erforderlich, den Zeitpunkt des Auftretens zu hinterlegen. Dieser ist durch Integrieren einer Echtzeituhr (weiterführend mit RTC abgekürzt) erfassbar. Eine RTC kann auf zwei Arten implementiert werden. Entweder enthält der $\mu$C diese Peripherie bereits oder eine externe Komponente wird integriert.
In diesem Projekt wird mit einem *Microchip SAME70* und einem *Microchip PIC24FJ-128GB204* gearbeitet. Diese beinhalten eine RTC. Damit ist die Erfassung der Ausnahmezeitpunkte realisierbar.
Für den Abnutzungsvorrat ist die Zeitspanne, in der die Last bewegt wird, relevant. Diese kann bestimmt werden über das Bilden der Differenz von Anfangs- und Endzeitpunkt der Positionierung. Der Start einer Positionierung ist an der Stromaufnahme des Motors erkennbar. Dem Elektromotor vorgeschallten befindet sich ein Servoregler. Unabhängig von dem verwendeten Typ unterstützen die meisten das *CANopen*-Protokoll [7].

[7] genaue Beschreibung des Protokolltyps in Kapitel 2.3.1

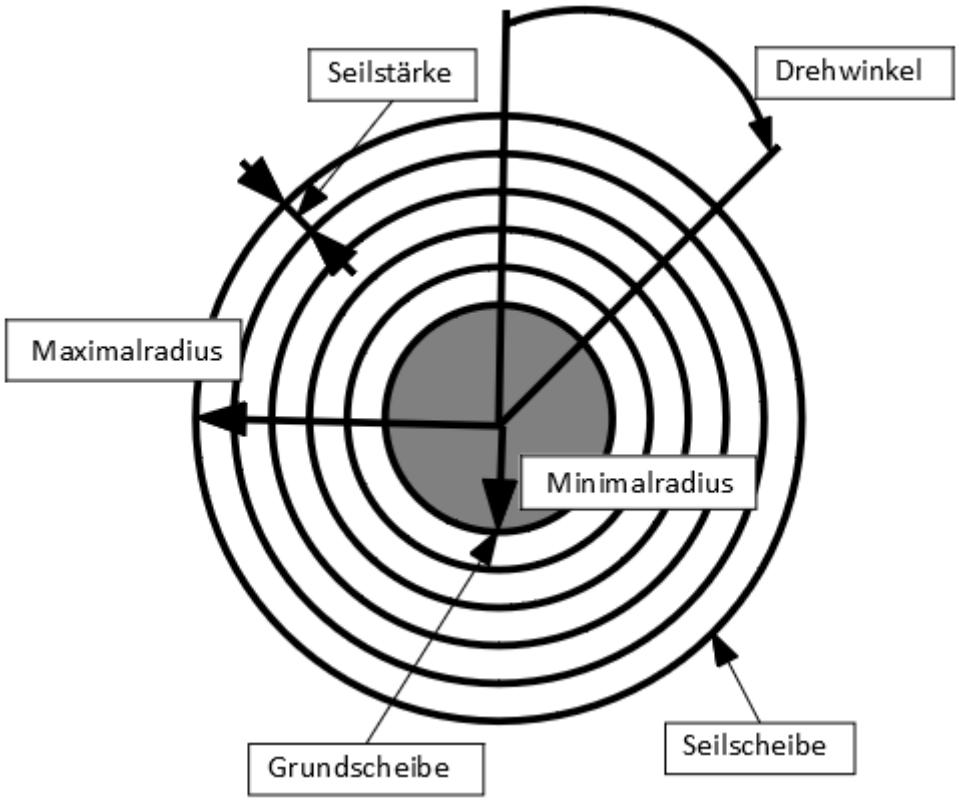



Abbildung 3.7: Schema Bemessungsgrößen der Seilscheibe

Über das Geräteprofil *cia 402* sind verschiedene Strom-, Spannungs- und weitere Werte über die jeweiligen Objektnummern auslesbar. Somit ist durch eine Kommunikation via *CANopen* ein Zugriff auf alle Parameter möglich.
Die Messung der anhängenden Last erfolgt über eine Lastmesszelle im Bedienteil *Balancer*. Das bisher verwendete Lastmessverfahren funktioniert zuverlässig. Die Komponenten der Lasterfassung sind auf der Folgeseite in Abbildung 3.8 dargestellt. Über die Lastzelle *Zemic H3G* wird die Last als Analogsignal an einen Analog-Digital-Wandler (weiterführend mit ADC abgekürzt) übertragen. Der angeschlossene ADC *AD7191* gibt diesen Wert in 24-Bit via SPI an den $\mu$C weiter. Mittels der entsprechenden Routinen und einer kalibrierten Referenzlast wird dem Nutzer die angehängte Last in kg angezeigt. Dieser Wert ist nur im Bauteil verfügbar. Das *Logikboard* erhält als Daten vom *Nextion-Controlboard* die Fahrtrichtung per Logiksignal und die Fahrtgeschwindigkeit als PWM-Signal [1].
Für eine Verwendung der Lastdaten in anderen Zusammenhängen und ohne *Balancer* ist es notwendig, die Erfassung von der Komponente zu entkoppeln. Es resultiert die Konzeptionierung einer selbstständigen Komponente zur Lasterfassung. Diese soll mittels CAN die Lastdaten zur Verfügung stellen. Zusätzlich ist vorgesehen, die Lastmessung im neuen System aus dem Laststrang zu entfernen. Durch diesen Schritt kann das Bedienteil auch außerhalb des Laststranges verbaut werden. [26].

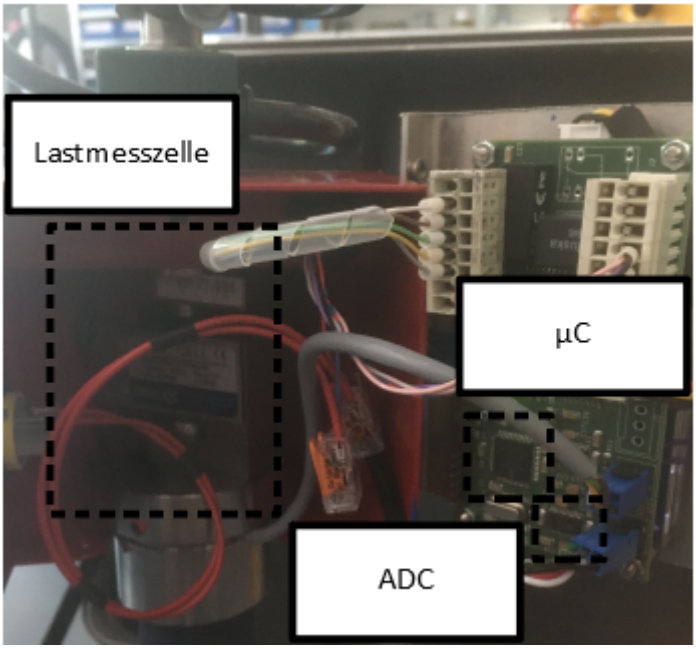



Abbildung 3.8: Komponenten der Lasterfassung

## 3.2 Datenspeicherung

Die erfassten Daten lassen sich unterteilen in Prozessdaten und Speicherdaten. Der Inhalt dieses Kapitels bezieht sich auf Speicherdaten. Sie dienen einer Auswertung zum späteren Zeitpunkt. Unterschieden wird in Daten die, für den Wartungsverlauf relevant sind und jene, die für die Historie der Maschine benötigt werden. In den beiden nachfolgenden Kapiteln sind die jeweiligen Anforderungen erläutert.

### 3.2.1 Lastkollektivspeicher

Über den Lastkollektivspeicher soll die Belastung des Hebezeugs erfasst werden. Zu diesem Zweck ist es notwendig Daten des Hubprozesses zu sammeln. Mittels dieser Daten, kann die Wartung an die reelle Belastung der Maschine angepasst werden. Zur Durchführung der Wartungsanpassung ist die Berechnung des Abnutzungsvorrates notwendig. Ziel ist es die dafür benötigten Daten zu erfassen.
Aktuell sind die Normen *FEM 9.511* und *FEM 9.755* Grundlage für die Berechnung des Abnutzungsvorrates. Diese unterscheiden vier Lastkollektive. Mit der Norm *FEM 9.511* werden die Triebwerke anhand ihrer theoretischen Laufzeit und Lastkollektivklasse in Triebwerksgruppen eingeteilt. Durch die Norm *FEM 9.755* ist die Protokollierung der tatsächlichen Nutzung definiert.
Der Abnutzungsvorrat ist die Differenz aus tatsächlicher und theoretischer Nutzung. Die theoretische Nutzung ist ein Tabellenwert. Für die tatsächliche Nutzung sind der Be-

lastungsfaktor und die Betriebszeit des jeweiligen Lastkollektives entscheidend [8]. Bei den vier Lastkollektiven handelt es sich um ideelle Belastungsverläufe [9]. Durch die in Formel 3.17 für den Belastungsfaktor gegebene Abhängigkeit, sind auch die vier Belastungsspektrumsfaktoren **Km** ideell [9] [10].

$$Km = k^3 \tag{3.17}$$

Im Anschluss wird durch ein Rechenbeispiel der Unterschied aufgezeigt, welcher durch die Berechnung mit ideellen und mit reellen Belastungsfaktoren entsteht. Anhand dessen ist auch erkennbar, welche Fehler durch die normgerechte Berechnung eintreten. Der Berechnung dient der Beispielaufbau, welcher in Abbildung 3.9 dargestellt ist. In diesem Bild ist ein Elektrokettenzug *KITO ER2* abgebildet, welcher mit einem *Balancer* ausgestattet ist. Die zur Ermittlung nötigen Parameter sind auf der Folgeseite in Tabelle 3.5 aufgelistet.

| **Parameter** | **Wert** |
|---|---|
| Triebwerksgruppe | $2_m$ |
| Gewicht Tragmittel | 5,2 kg |
| Gewicht Lastaufnahmemittel | 0,6 kg |
| Gewicht Nutzlast | 20 kg |
| Tragfähigkeit | 125 kg |

Tabelle 3.5: Werte des Beispielaufbaus

Die Formeln 3.18 bis 3.22 enthalten die durchgeführten Rechenschritte. Ob das Gewicht des Tragmittels relevant ist, ergibt sich aus dem Verhältnis zur maximalen Traglast des Hebezeugs. Der Anteil des Tragmittels an der Traglast beträgt 4 $\%$. Nach *FEM 9.755* ist das Gewicht des Tragmittels somit nicht zu berücksichtigen [9]. Die nötigen Zeiten der einzelnen Lastkollektive für die Berechnung des kubischen Mittels basieren auf Annahmen. Diese Annahmen folgen aus dem Arbeitsablauf, in dem der dargestellte Aufbau verwendet wird. In diesem Arbeitsablauf wird ein Akku vom Tisch in den Transportkoffer umgelagert. Daraus folgt, dass die Zeiten für Nutzung mit Teillast und Totlast identisch sind. Es folgt also $t_1 = t_\Delta = 0,5$.
Durch die Tatsache, dass beim gegebenen Arbeitsablauf stets die selbe Last umgelagert wird, ergibt sich ein Nutzlastfall. Die Zeit, in der das Hebezeug ohne Teillast betrieben wird, stellt den Totlastfall dar.

$$\alpha = \frac{5,2kg}{125,6} = 0,04 \tag{3.18}$$

$$\beta = \frac{20,6kg}{125,6} = 0,16 \tag{3.19}$$

[8] siehe Kapitel 3.1.1
[9] siehe Kapitel 3.1.1

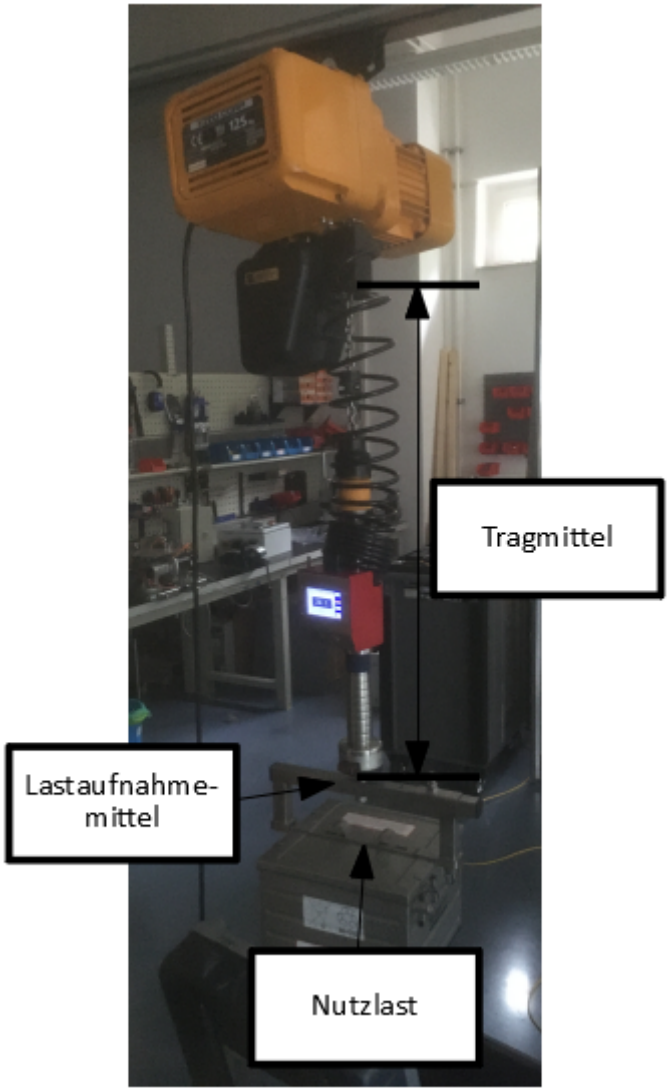



Abbildung 3.9: Beispielaufbau zur Errechnung des Belastungsfaktors

$$\gamma = \frac{0,6kg}{125,6} = 0,00 \tag{3.20}$$

$$k = \sqrt[3]{(0,16+0,00)^3 \cdot 0,5 + 0,00^3 \cdot 0,5} = 0,13 \tag{3.21}$$

$$Km = 0,13^3 = 0,0022 \tag{3.22}$$

Das Ergebnis der Berechnung zeigt, dass der Belastungsfaktor $K_m$ mit 0,002 deutlich unter dem zum Lastkollektiv eins passenden Wert von 0,125 liegt.
Die theoretische Nutzungsdauer für den Versuchsaufbau beträgt 12500 h [10]. Dieser Wert wird im Anschluss zur Berechnung der tatsächlichen Nutzungsdauer als Gesamtbetriebszeit angenommen.

$$S = (0,002 \cdot 12500h) = 25h \tag{3.23}$$

[10] abgelesen aus Tabelle 3.1 Kapitel 3.1.1

$$S = (0,125 \cdot 12500h) = 1.562h \tag{3.24}$$

Es resultiert eine Differenz von 1.537 h. Mit einer feineren Einteilung der Lastkollektive würde sich somit die GÜ verzögern. Aus der größeren Zeitspanne können jedoch Schäden am oder Gefährdungen vom Hebezeug resultieren. Mit dem Auftreten dieser Tatsachen würde eine SWP ausgeschlossen und der Zweck des Lastkollektivspeichers widerlegt. Aus diesem Grund wird die Ablagestruktur der Daten, für eine normgerechte Ermittlung des Abnutzungsvorrates, konzipiert.
Für die spätere Ausgabe werden Last- und Zeitwerte benötigt. Die Zeitspanne des abgelaufenen Nutzungsvorgangs wird an zwei Stellen abgelegt. Zum einem wird sie auf die Gesamtbetriebsdauer aufsummiert und zum anderen der entsprechenden Last zugeordnet.
Die *FEM 9.511* teilt die Tragfähigkeit des Hebezeugs in Totlast, 1/3 Nutzlast, 2/3 Nutzlast und volle Nutzlast auf. Diese vier Bereiche bilden die entsprechenden Lastklassen. Die Nutzlast ist die Differenz aus Tragfähigkeit und Totlast. Durch Zuordnung der Betriebszeit in diese Bereiche ist eine spätere Errechnung der tatsächlichen Nutzungsdauer nach Norm möglich.
Soll der Abnutzungsvorrat gemäß Norm berechnet werden, sind die einzelnen Nutzungen den Lastklassen zu zuordnen. Das bedeutet, es werden vier Speicherstellen nötig. Der maximale Wert für die theoretische Nutzungsdauer beträgt 100.000 h [11]. Zur Ablage eines solchen Wertes bedarf es 17 Bit oder drei Byte. Auf Grund dessen, dass ein einzelner Hubvorgang jedoch keine Stunde, sondern Minuten beziehungsweise Sekunden dauert, sind jeweils noch weitere sieben Bit nötig. Diese sieben Bit dienen zur Darstellung der Werte 0 - 59. Daraus resultiert ein Datenbedarf pro Lastklasse von 31 Bit oder vier Byte. Mit der Gesamtbetriebsdauer sind insgesamt 20 Byte Datenraum nötig, um eine normgerechte Aufzeichnung durchführen zu können.
Um den Gesamtbedarf an Speicher zu ermitteln ist es notwendig, die Anzahl an Hubzyklen abzuschätzen, welche in einer SWP getätigt werden. Über deren Menge und den vier Byte zum Speichern einer Nutzung, ergibt sich der Gesamtbedarf. Dazu dient erneut das Beispiel aus Abbildung 3.9. Die Nutzung des Hebezeugs zur Positionierung der Last findet im Bereich von ca. 30 Sekunden statt. Bis zum Ende einer SWP kann das verwendete Hebezeug der Triebwerksgruppe $2_m$, 12500 h im Lastkollektiv eins genutzt werden. Das ergibt eine Anzahl von 45.000.000 Einzeleinträgen. Durch die Multiplikation dieser Menge mit dem Datenbedarf eines Einzeleintrags von 32 Bit entsteht ein benötigtes Speichervolumen von 1,44 GBit. Der Vergleich des Ergebnisses mit dem maximal verfügbaren Speicher eines EEPROM (Electrically Erasable programmable Read Only Memory) der Firma *Microchip* entsteht die Notwendigkeit, den Bedarf an Datenvolumen zu reduzieren. Der größte Speicher besitzt 2 MBit Kapazität.
Das bedeutet, eine Aufzeichnung jedes Hubzyklus ist nicht möglich. Da nur die Berechnung des Abnutzungsvorrates nach Norm relevant ist, müssen auch nur diesbezüglich Daten gespeichert werden. Daraus folgt, dass ein Datenrahmen von 20 Byte oder 160 Bit zur Verfügung gestellt werden muss. Die spezifische Last wird nicht erfasst, sondern

[11] siehe Tabelle 3.1

die Betriebszeit der jeweiligen Lastklasse zugeordnet. Damit ist in der Produktauswahl des Herstellers der kleinste Speicher mit 1 kbit ausreichend. In der Abbildung 3.10 ist der Aufbau des Speichers schematisch dargestellt.

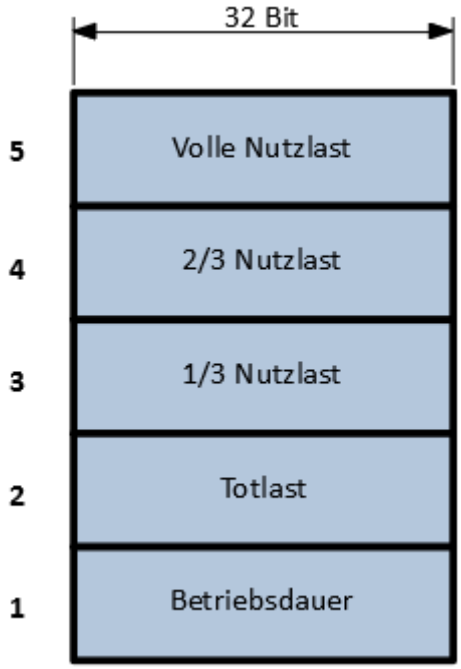



Abbildung 3.10: Speicheraufbau des Lastkollektivspeichers

Der Speicher ist der *AT25010B*, mit dessen Einsatz sich die nächste Anforderung zur Aufnahme der Last ergibt. Bei dieser Anforderung handelt es sich um die Zugriffsanzahl auf den Speicher. Der ausgewählte EEPROM ist laut Datenblatt für 1.000.000 Schreibzyklen ausgelegt [29]. Durch die vorherige Kalkulation der Hebezyklen ist mit einer höheren Anzahl an Zugriffen zu rechnen.
Die Anzahl der Zugriffe lässt sich reduzieren, in dem Werte länger im RAM des Prozessors verbleiben, bevor sie im EEPROM gespeichert werden. Eine Erhöhung der Verweilzeit wird durch Bedingungen zur Datenablage erreicht.
Als Referenz für die Erstellung der nötigen Bedingungen dienen die Angaben zu den Nutzungsdauern. Als Maßeinheit für die Zeitspanne der Nutzungen werden Stunden verwendet. Daraus folgt, dass es wichtig ist, die Betriebsdauer im Stundenzyklus zu aktualisieren.
Ein weiterer Faktor zur Speicherung der Nutzungszeiten ist die Last selbst. Bei einer Umpositionierung wird ein Gut von A nach B bewegt. Ist die Last größer null, so ist anzunehmen, dass die Bewegung noch fortgesetzt wird. Somit wird die Ablage der Daten noch verzögert, um die Betriebszeit der Umpositionierung vollständig erfassen zu können.
Bei einer unregelmäßigen Nutzung des Hebezeugs kann es auf Grund der zuvor geschaffenen Bedingungen aber zu einer niedrigen Aktualisierungsrate kommen. Diese wird begrenzt, in dem auch kleinere Betriebszeiten nach einer maximalen Sicherungs-

periode im EEPROM hinterlegt werden.
Der gesamte Ablagealgorithmus ist in Abbildung 3.11 nochmals grafisch dargestellt.

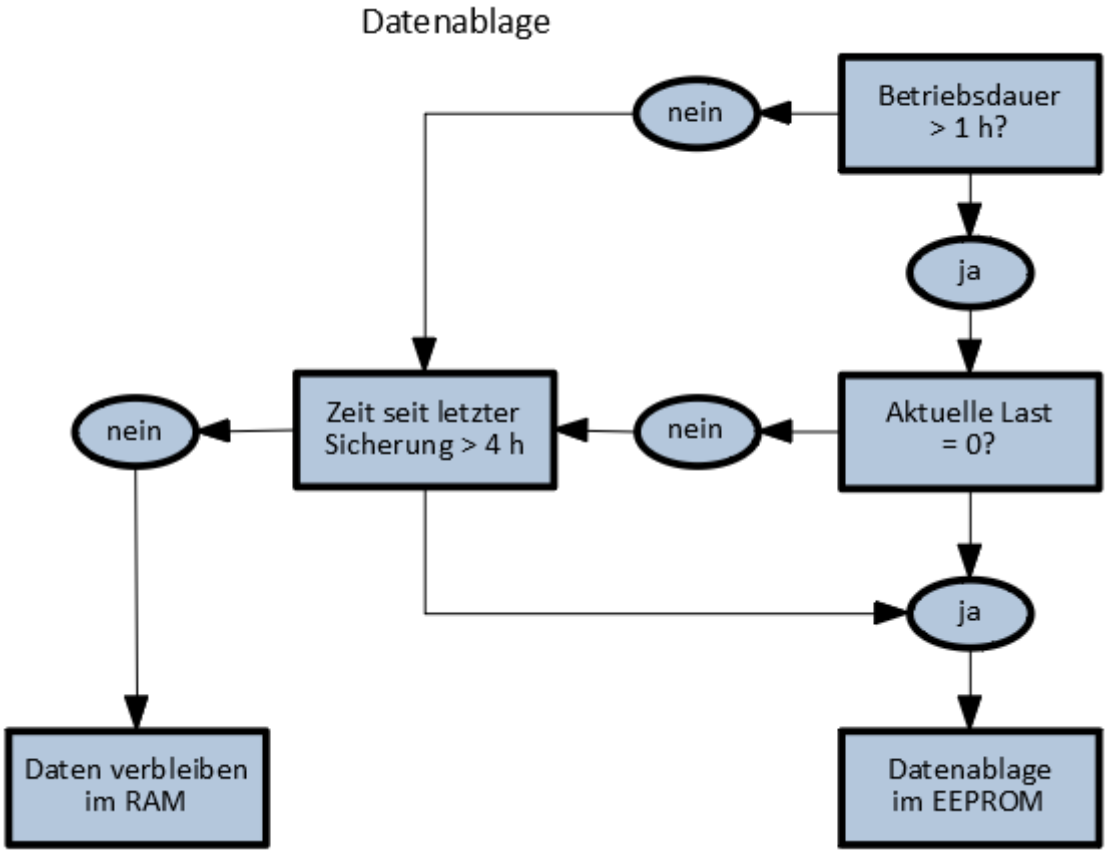



Abbildung 3.11: Algorithmus zum Minimierung der EEPROM Zugriffe

Für die spätere Berechnung des Abnutzungsvorrates sind noch Daten aufzunehemen, welche unabhängig von den Betriebszeiten sind. Bei diesen handelt es sich um die zutreffende Triebwerksklasse mit den entsprechenden theoretischen Nutzungszeiten, die Tragfähigkeit des Triebwerks und die einzelnen Gewichte.
Alle diese Werte müssen vor der ersten Nutzung initialisiert werden. Die Initialisierung der Werte wird durch den Bediener realisiert. Der Zugriff ist auf unterschiedliche Arten möglich. Eine Variante ist die direkte Eingabe der Werte und Befehle über das HMI. Durch die Eingabe der Werte an einem entfernten Zugriffsort und das anschließende Senden dieser via WLAN, *CANopen* oder *EtherCat* wird eine weitere zur Verfügung gestellt.
Unabhängig vom Zugriffsort folgt die Initialisierung der selben Ablaufroutine. Zu Beginn wird vom Nutzer die Eingabe der Triebwerksparameter erwartet. Die Triebwerksgruppe dient der späteren Berechnung des Abnutzungsvorrates. Anschließend erwartet das System die Eingabe der Last, welche dem Tragmittel entspricht.
Darauf folgend muss das Gewicht des Lastaufnahmemittels hinterlegt werden. Dieser Schritt kann auf zwei Arten ausgeführt werden. Mit der ersten Variante erfolgt die Initialisierung mittels einer Benutzereingabe des absoluten Gewichtes. In der zweiten wird

dem System mitgeteilt, dass das Gewicht des Lastaufnahmemittels dem derzeit anhängenden entspricht. Der Absolutwert wird anschließend vom System selbst ermittelt. Mit dem Abschluss der Initialisierung sind alle notwendigen Daten hinterlegt, um Berechnungen für den Abnutzungsvorrat auszuführen. Nach der Initialisierung des Systems kann die Betriebszeiterfassung beginnen. Der Algorithmus für die Initialisierung ist in der Abbildung 3.12 dargestellt.

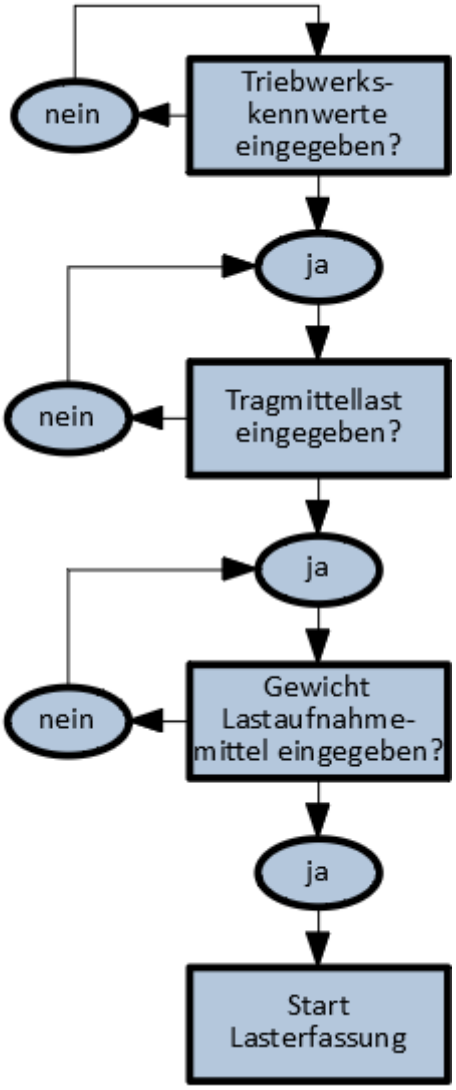



Abbildung 3.12: Ablaufschema Initialisierung Lastkollektivspeicher

Wird ein Motorstrom registriert, beginnt die Datenerfassung. Der zugehörige Algorithmus ist in Abbildung 3.13 dargestellt. Der aktuell aufgenommene Strom des Motors ist Inhalt der vom Servoregler gesendeten PDO-Nachricht [12].
Der folgende Schritt beschreibt die Lasterfassung. Bei der Ausführung dieses Schrittes sind unterschiedliche Szenarien zu beachten. Ein Szenario ist, dass die Last bereits vom Hebezeug getragen wird. Mit dieser Ausgangssituation kann die Betriebszeit der

[12] Genauere Details zum Nachrichtenaustausch via *CANopen* sind in Kapitel 4.1.1

folgenden Nutzung dem Lastkollektiv sofort zugeordnet werden.
Findet die Lastaufnahme in der Art statt, dass das zu befördernde Gut erst aufgenommen werden muss, ergibt sich ein anderes Bild. In diesem Fall ist zu ermitteln, wann die Last frei hängt. Erst durch diese Erkenntnis kann die Nutzungsdauer dem passenden Lastkollektiv zugeordnet werden.

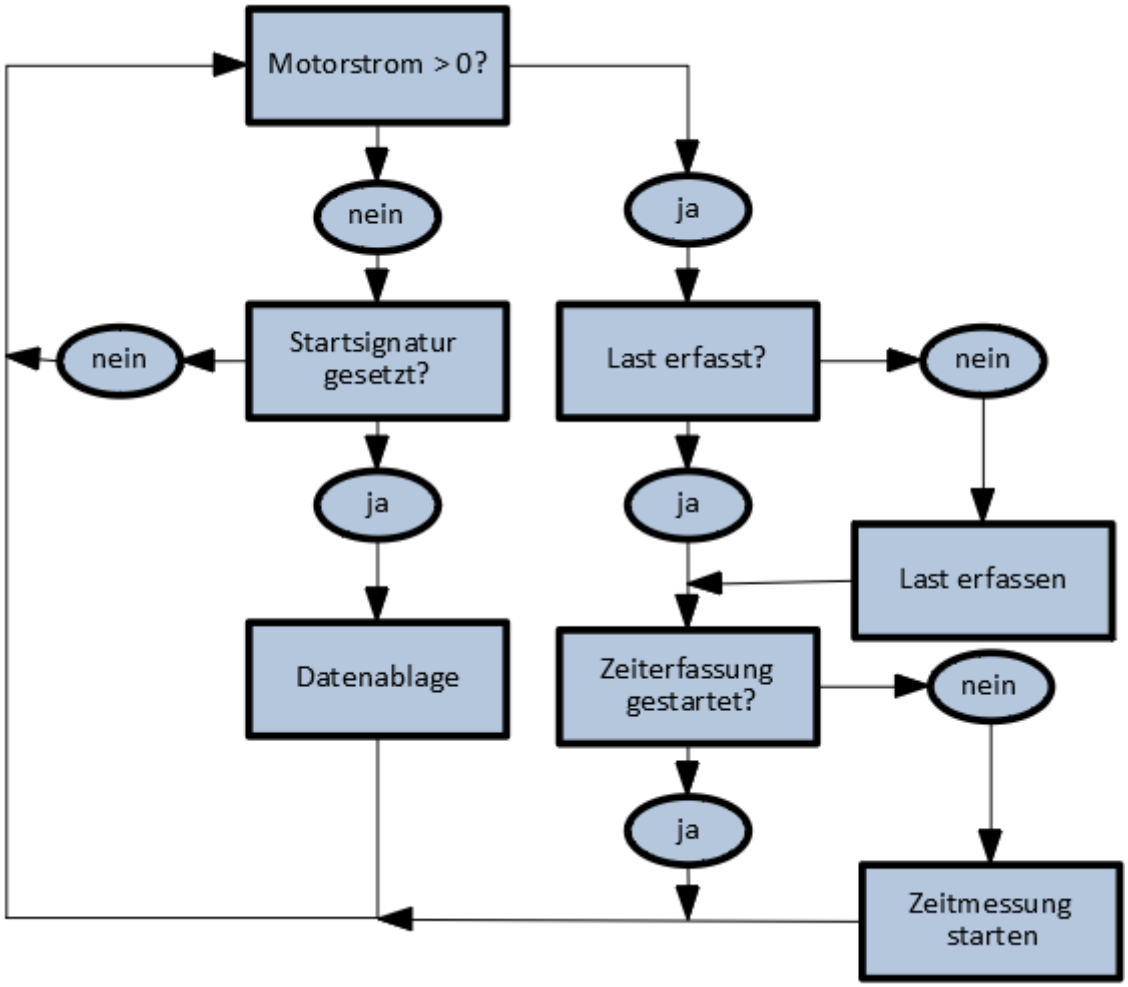



Abbildung 3.13: Prozessschema Lastkollektivdatenerfassung

Nach der Lasterfassung wird die Nutzungsdauer aufgenommen. Dazu ist es nötig, den Beginn der Nutzung zu kennen. Dieser wird in der Signatur hinterlegt. Ist die Signatur gesetzt, erfolgt die erneute Messung des Motorstromes und damit der nächste Durchlauf. Wenn nicht, wird diese Signatur gesetzt.
Eine Wiederholung dieser Routine erfolgt, solang ein Motorstrom gemessen wird. Dieser ist das Indiz für die Nutzung der Maschine. Befindet sich das Hebezeug im Stillstand, nimmt der Motor auch keinen Strom auf. In diesem Moment, erfolgt der Durchlauf des zweiten Zweiges der Routine. Zu Beginn ist zu prüfen, ob eine Signatur gesetzt ist. Sollte das nicht der Fall sein, kann daraus geschlussfolgert werden, dass keine Nutzung stattgefunden hat.
Anderenfalls lag eine Nutzung vor. In diesem Szenario wird die Routine in Abbildung

3.11 durchlaufen. Am Ende erfolgt die Aufsummierung der Zeitdauer des Vorgangs auf die entsprechenden Speicherstellen.
Die somit erhaltenen Daten und alle die zur Berechnung der Nutzdauer nötig sind, sollen anderen Komponenten zur Verfügung gestellt werden. Der Lastkollektivspeicher ist eine Einzelkomponente, welche über das *CANopen*-Protokoll mit anderen Teilnehmern kommuniziert. Die mit dem Lastkollektivspeicher verwalteten Daten entsprechen keinem *cia*-Geräteprofil. Daraus folgt, dass es sich um herstellerspezifische Daten handelt. Diese sind im OD-Bereich von 2000h bis 5FFFh zu finden sind [13].
Mit der Tabelle 3.6 werden alle Parameter, die im Lastkollektivspeicher enthalten sind, aufgelistet. Jedem Parameter ist ein OD-Eintrag zugewiesen. Die verwendeten und gesammelten Daten sind im OD-Bereich entsprechend ihrer Verwendung aufgeteilt wurden. So erfolgte eine Reservierung vom Eintrag 2000h bis 20FFh für Parameter, welche zur Initialisierung verwendet werden.
In den Einträgen mit den Index-Nummern 2200h bis 220FFh stehen die Parameter, welche für eine normgerechte Analyse des Abnutzungsvorrates benötigt werden.

| **Parameter** | **Index** | **Subindex** |
|---|---|---|
| Initialisierungsparameter (2000h-20FFh) | | |
| Triebwerksgruppe | 2000h | 00h |
| Gewicht Totlast | 2001h | 00h |
| Gewicht Lastaufnahmemittel | 2002h | 00h |
| Tragfähigkeit | 2003h | 00h |
| allgemeingültige Parameter (2100h-21FFh) | | |
| Gesamtbetriebsdauer | 2100h | 00h |
| aktuelle Last | 2101h | 00h |
| normbezogene Parameter (2200h-22FFh) | | |
| theoretische Nutzung | 2200h | 00h |
| Nutzungsdauer Norm | 2201h | |
| Totlast | | 01h |
| drittel Nutzlast | | 02h |
| zweidrittel Nutzlast | | 03h |
| Nutzlast | | 04h |
| tatsächliche Nutzung | 2202h | 00h |
| Restnutzung | 2203h | 00h |

Tabelle 3.6: Herstellerspezifische Einträge Lastkollektivspeicher

In diesem Kapitel soll abschließend noch betrachtet werden, welche Hardwarekomponenten benötigt werden, um die Funktion des Lastkollektivspeichers ausführen zu können. Dabei erfolgt der Bezug auf den Schaltplan in Anhang C.
Der Lastkollektivspeicher ist als Einzelkomponente ausgeführt. Das heißt, er ist nicht Bestandteil des *Logikboards*. Für das Ausführen der Kernfunktion muss die Last und die Zeit erfasst werden.

[13] siehe Kapitel 2.3.1

Die Erfassung der Last erfolgt durch eine Lastmesszelle [14]. Diese wird aus dem *Balancer* ausgelagert. Dieser Schritt ist notwendig, weil auch bei Hebezeugen ein Lastkollektivspeicher eingesetzt wird, die ein anderes Bedienteil verwenden.
Über die im $\mu$C integrierte Echtzeituhr erfolgt die Erfassung der Zeit. Zur Visualisierung von definierten Ereignissen sind zwei LEDs integriert. Mit ihnen ist es zum Beispiel möglich, eine anstehende oder überfällige GÜ anzuzeigen.
Ebenfalls zur Visualisierung dient die Anschlussmöglichkeit eines HMI. Über dieses ist es zu dem noch möglich, die Initialisierung auszuführen und alle verwendeten Parameter anzuzeigen.

### 3.2.2 Flugschreiber

Durch den im Prozessor des *Logikboard* integrierten Flugschreiber werden definierte Ereignisse protokolliert. Mit dem erstellten Protokoll kann die Historie des Hebezeugs nachvollzogen werden. Auf Basis dieser Daten lassen sich zum Beispiel Fehlerquellen zu einem späteren Zeitpunkt ermitteln oder der Wartungsverlauf einer Maschine nachvollziehen.
Welche Ereignisse detailliert erfasst werden, korreliert mit der jeweiligen Anwendung. Inhalt dieses Abschnitts ist der allgemeine Aufbau und die Funktion eines Flugschreibers.
Für die mit der Protokollierung verbundene Speicherung von Daten ist es notwendig, diese Daten und deren Format zu kennen. Zur Erstellung einer Historie ist es unabdingbar, das spezifische Ereignis mit einem Zeitstempel zu markieren. Dieser muss das Datum und den Zeitpunkt enthalten, an dem die entsprechenden Bedingungen vorlagen.
Wie schon beim Lastkollektivspeicher bildet die auf dem $\mu$C implementierte RTC die Zeitquelle. Demnach ist das Datenformat aus dem Datenblatt des Prozessors zu entnehmen. Beim Flugschreiber handelt es sich nicht um eine eigenständige Komponente. Er ist im $\mu$C des Logikboards implementiert. Als $\mu$C wird ein *SAME70* von *Microchip* verwendet.
Bei diesem wird zur Ablage der Datierung ein Datenraum von sieben Byte benötigt. Mit dieser Menge an Bits lässt sich die Zeit (hh:mm:ss) und das Datum (dd.mm.jjjj) darstellen [30, S.229].
Zusätzlich zum Zeitpunkt ist eine Kennung zu hinterlegen, welche auf das zum jeweiligen Zeitpunkt aufgetretene Ereignis verweist. Für die Datengröße dieses Informationsteils wird Bezug auf die Emergency-Nachricht des *CANopen*-Protokolls genommen. In Abbildung 3.14 ist der Aufbau einer solchen Nachricht noch einmal grafisch dargestellt. Die acht Bytes der Emergency-Nachricht gilt es zusätzlich zum Datum und der Uhrzeit zu hinterlegen. Damit ergibt sich ein Speicherbedarf von 15 Bytes oder 120 Bit. Eine grafische Darstellung der Ereignisstruktur ist in Abbildung 3.15 gegeben.

[14] siehe Abschnitt Lasterfassung in Abschnitt 3.1.1

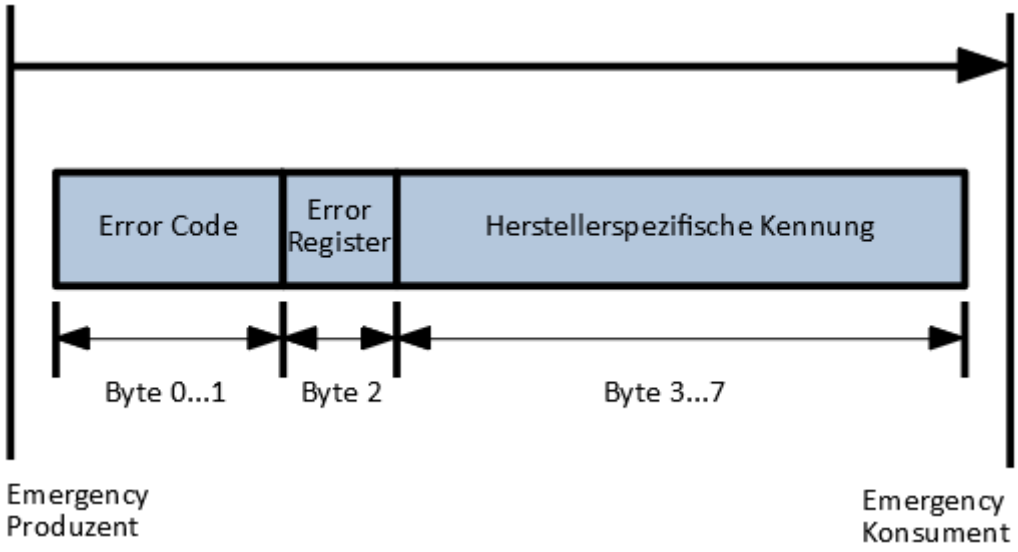



Abbildung 3.14: Aufbau Emergency-Nachricht *CANopen* [3, S. 102]

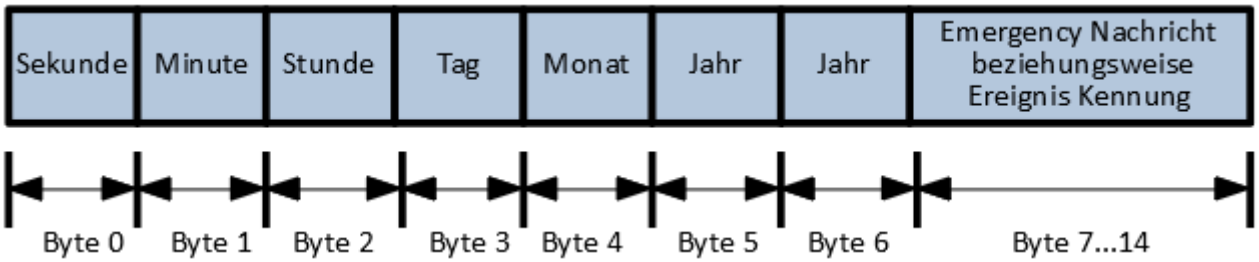



Abbildung 3.15: Aufbau eines Eintrags im Flugschreiber

Die Anzahl der Nachrichten, welche angelegt werden können, hängt von der Speicherkapazität des jeweiligen Mediums ab. Wie auch beim Lastkollektivspeicher soll dieses ein EEPROM sein. Aus diesem Grund dient zur Auswahl des Bausteins die Bauteilliste von *Microchip*. Verfügbar sind diese Speicher mit einer Kapazität von einem kBit bis zwei MBit. Mit der Anzahl an Ereignissen ausgedrückt bedeutet die Kapazitäten ein Ablagevolumen von 8 bis ca. 16.600 Ereignissen. Der Raum zwischen diesen beiden Grenzen ist abgestuft.
Ereignisse, die von Interesse sind, werden nicht ausschließlich über eine Emergency-Nachricht gesendet. Beispielsweise sollen auch Änderungen an den Parametern und durchgeführte Wartungen hinterlegt werden.

# 4 Datenbereitstellung und Datenverarbeitung

Der für die Funktionen resultierende Datenbedarf setzt eine Vernetzung der Komponenten voraus [15]. Eine solche Vernetzung ist Inhalt der Industrie 4.0.
Im Abschnitt 4.1 wird aufgezeigt, wie die Kommunikation innerhalb und außerhalb des Hebezeugs beziehungsweise Kranes realisiert werden kann.
Das Ziel des zweiten Abschnittes ist die Steigerung der Flexibilität des Systems. Damit verbunden ist in Kapitel 4.2 aufgeführt, welche Anforderungen an die Komponenten gestellt werden und aus welchen Gegebenheiten diese resultieren.
Zuletzt wird in Abschnitt 4.3 das Gesamtsystem betrachtet und herausgearbeitet, welche Anforderungen an die Hardwarestrukturen gestellt werden.

## 4.1 Kommunikation

Das folgende Kapitel teilt sich inhaltlich in zwei Bereiche. Der erste Bereich umfasst die Vernetzung im Hebezeug, das heißt all jene Komponenten, welche notwendig sind, um eine Last umpositionieren zu können.
Im zweiten Teil wird die Anbindung an das industrielle Umfeld betrachtet. Dabei wird darauf eingegangen, was für Komponenten für eine Anbindung vorhanden sein müssen und wie die Kommunikation zwischen ihnen und dem *Logikboard* abläuft.

### 4.1.1 Vernetzung im Hebezeug

Auf Grund des erhöhten Datenverkehrs zur Erfüllung der Anforderungen ist die Kommunikationsstruktur des Hebezeugs zu überarbeiten. Zum jetzigen Stand sind Signale immer durch eine direkte Verdrahtung der Komponenten übertragen wurden [16]. Soll ein zusätzlicher Parameter integriert werden, muss in die Hardware eingegriffen werden. Durch die nachfolgende Betrachtung ist dies im neuen Hebezeug nicht mehr nötig.
Der in diesem Kapitel betrachteten Kommunikationsstruktur liegt ein Projekt mit Nutzung einer Portalanlage zu Grunde [17]. Bei diesem ist das Umpositionieren im dreidimensionalen Raum möglich. Die Bewegungen in x- und y-Richtung werden jeweils von einem *eDrive*-Fahrwerk ausgeführt. Als Protokoll zur Kommunikation zwischen den einzelnen Komponenten ist das *CANopen* ausgewählt wurden. Dieses eignet sich für die Vernetzung des Systems, da einzelne Prozessoren, Sensoren und Aktoren miteinander

[15] der detaillierte Bedarf ist in Kapitel 3 beschrieben
[16] siehe Kapitel 1
[17] Abbildung 3.3 Kapitel 3.1.1

| Komponente | Datenbedarf | Datenbereitstellung |
|---|---|---|
| **Logikboard** | -aktuelle Geschwindigkeit Fahrwerk x<br>-aktuelle Geschwindigkeit Fahrwerk y<br>-aktuelle Geschwindigkeit Hebezeug<br>-aktuelle Position x<br>-aktuelle Position y<br>-aktuelle Position z<br>-Bedienteilidentität<br>-Soll-Benutzervorgaben | -Soll-Geschwindigkeit Fahrwerk x<br>-Soll-Geschwindigkeit Fahrwerk y<br>-Soll-Geschwindigkeit Hebezeug<br>-Soll-Position x<br>-Soll-Position y<br>-Soll-Position z |
| **Regler Antrieb Hebezeug** | -Soll-Position z<br>-Soll-Geschwindigkeit Hebezeug | -aktuelle Position z<br>-aktuelle Geschwindigkeit Hebezeug<br>-Strom-Istwert |
| **Regler Antrieb Fahrwerk y** | -Soll-Position y<br>-Soll-Geschwindigkeit Fahrwerk y | -aktuelle Position y<br>-aktuelle Geschwindigkeit Fahrwerk y |
| **Regler Antrieb Fahrwerk x** | -Soll-Position x<br>-Soll-Geschwindigkeit Fahrwerk x | -aktuelle Position x<br>-aktuelle Geschwindigkeit Fahrwerk x |
| **Lasterfassung** | -Strom-Istwert | -aktuelle Last |
| **Bedienteil** | -aktuelle Last (nur Balancer) | -Soll-Benutzervorgaben<br>-Bedienmodus (nur Balancer)<br>-Bedienteilidentität |

Tabelle 4.1: Datenbedarf und -bereitstellung der *CANopen* Nodes für zyklischen Austausch

verbunden werden [18]. Mit den Reglern der Motoren, für die Antriebe von Hebezeug und Fahrwerk, sind zudem Komponenten im System vorhanden, die über ein implementiertes OD verfügen [19]. Für die übrigen Teilnehmer muss dieses erstellt werden. Dieses wird mittels EDS im Node implementiert. Zur Erstellung eines solchen EDS sind Softwarelösungen vorhanden. Eine solche ist das *CANopen Architect* der Firma *EmSA*. Mit dieser ist es möglich, dem Gerät und der Applikation entsprechende Profile vorzuladen. Mit dem ausgewählten Profil stehen eine Reihe von Einträgen zur Verfügung.
Der Fokus wird zunächst auf die Kommunikation zum Ausführen der Grundprozesse gelegt. Unter diesen verstehen sich alle Vorgänge, die der Positionierung von Lasten dienen. Dabei wird im neuen System zwischen der Positionierung mittels Koordinatenvorgabe und Geschwindigkeitsvorgabe unterschieden.
Da es sich bei beiden Arten um die Übertragung zyklischer Prozessdaten handelt, er-

[18] siehe Automationpyramide in Abbildung2.3 Kapitel 2.3
[19] das OD bildet die Basis für die *CANopen*-Kommunikation, siehe 2.3.1

folgt der Austausch via PDOs. Um diese erstellen zu können, müssen der Datenbedarf und die Datenbereitstellung der Komponenten ermittelt werden. In Tabelle 4.1 ist die Auflistung der Daten dargestellt.
Auf Basis dieser Auflistung lässt sich die Anzahl der notwendigen PDOs ermitteln. Da jeder Node Daten bereithält, die für einen anderen Teilnehmer von Bedeutung sind, ergibt sich die Anzahl von mindestens sechs TPOs.
Für die weitere Betrachtung ist der Wertebereich und -typ der einzelnen Parameter signifikant. Da ein PDO maximal acht Byte beinhalten kann, muss überprüft werden, wie viele Bytes die jeweiligen Daten benötigen. Diese Angaben sind in der Tabelle 4.2 enthalten.

| **Parameter** | **Datentyp** | **Anzahl Bytes** |
|---|---|---|
| Positionsangaben | Integer 32 | 4 |
| Geschwindigkeiten | Integer 32 | 4 |
| Strom | Integer 32 | 4 |
| Lasten | Integer 16 | 2 |
| Bedienteilparameter | | |
| ID | Integer 8 | 1 |
| Bedienmodus | Integer 8 | 1 |
| Drehrichtung | Integer 8 | 1 |
| Benutzervorgabe | Integer 16 | 2 |

Tabelle 4.2: Wertetypen der PDO-Parameter

Für die Analyse der Datenlängen von den Positions- und Geschwindigkeitsangaben wurde das OD des zur Zeit verwendeten Reglers für die Fahrwerke herangezogen. Dabei handelt es sich um den *CL4-E-X-12* der Firma *Nanotec*. In dessen OD befinden sich die aktuelle Position auf Index 6064h und die derzeitige Geschwindigkeit auf 606Ch. Diese Einträge sind geräteprofilspezifisch [20]. Das heißt, sie sind herstellerunabhängig und auch für den Regler des Hebezeugantriebs zu verwenden.
Der aktuelle Stromwert befindet sich im herstellerspezifischen Bereich des OD. Dadurch ist bei jedem Hersteller die passende Adresse aus dem Handbuch auszuwählen. Im Projekt lässt sich dieser Wert über die Adresse 2077h am verwendeten *AKD*-Regler der Firma *Kollmorgen* auslesen.
Bei der Übertragung von Lastwerten ergibt sich das Datenvolumen aus zwei Anforderungen heraus. Die erste ist die maximal tragbare Last von 500 kg. Über die Auflösung von 100 g ergibt sich ein Bedarf von 13 Bit zur Datenübertragung [21]. Die Bit-Anzahl ist nötig, um den maximalen Wert von 5000 darstellen zu können. Auf Grund dessen, dass der Umfang des Datenvolumens auf Standardwortlängen bezogen wird, ist ein Datenbereich von zwei Byte für die Lastwerte zu reservieren.
Über die Informationen aus den Geräteprofilen und der Lasterfassung können alle zum

[20] siehe Tabelle 2.2 in Kapitel 2.3.1
[21] Auflösung wird für den Automatikbetrieb des *Balancers* benötigt, siehe 1

Betrieb notwendigen Parameter, mit Ausnahme der Bedienteildaten, übertragen werden. Als erste Bedienteilinformation wird dessen Identifikation benötigt. Die Steuerung des Hebezeugs ist mit drei Varianten möglich [22]. Der Wechsel des Bedienmodus ist dem *Balancer* vorbehalten. Nur bei diesem Bedienteil sind zwei unterschiedliche Modi verfügbar. Das eine Byte, welches für die Angabe der Drehrichtung bereitgestellt wird, ergibt sich aus den möglichen Zuständen. Die möglichen Zustände sind links, rechts oder Stillstand. Aus der Verwendung des Drehgriffs ergibt sich die Datengröße für Benutzervorgaben. Unter Benutzervorgaben ist das Geschwindigkeitslevel zu verstehen, welches der Maschinenbediener mit Hilfe Bedienelements vorgibt. Beim Drehgriff ist im derzeitigen System ein Wert im Bereich von 0 bis 1024 zu erwarten. Zur Darstellung des Wertes 1024 werden zwei Bytes benötigt.
Durch die so ermittelten Datengrößen und des maximalen Fassungsvolumens eines PDOs ist ersichtlich, dass für alle Nodes mit Ausnahme des *Logikboards* und dem Regler des Hebezeugs ein TPDO ausreicht. Durch zwei TPDO kann der Antriebsregler seine Daten an die anderen Nodes senden. Das Logikboard benötigt drei solcher Objekte, um alle benötigten Daten zur Verfügung zu stellen.
Der Dateninhalt dieser drei PDOs ist so angeordnet, dass jedes Objekt einen Zielteilnehmer Besitzt. So wird beispielsweise über das Objekt mit der Kennung 181h der Zustand des *eDrive* zur Positionierung der X-Achse aktualisiert. Diese und alle übrigen TPDOs sind in der Tabelle 4.3 aufgeführt.

| **Komponente** | **COB ID** | **Datenbereitstellung** |
|---|---|---|
| **Logikboard** | 181h | -Soll-Position Fahrwerk x |
| | | -Soll-Geschwindigkeit Fahrwerk x |
| | 281h | -Soll-Position Fahrwerk y |
| | | -Soll-Geschwindigkeit Fahrwerk y |
| | 381h | -Soll-Position Hebezeug |
| | | -Soll-Geschwindigkeit Hebezeug |
| **Regler Antrieb Hebezeug** | 182h | -aktuelle Position z |
| | | -aktuelle Geschwindigkeit Hebezeug |
| | 282h | -Strom Ist-Wert |
| **Regler Antrieb Fahrwerk y** | 184h | -aktuelle Position y |
| | | -aktuelle Geschwindigkeit Fahrwerk y |
| **Regler Antrieb Fahrwerk x** | 183h | -aktuelle Position x |
| | | -aktuelle Geschwindigkeit Fahrwerk x |
| **Lasterfassung** | 186h | -aktuelle Last |
| **Bedienteil** | 187h | -Soll-Benutzervorgaben |
| | | -Bedienmodus (nur Balancer) |
| | | -Bedienteilidentität |

Tabelle 4.3: TPDO - Zuweisung

[22] siehe Kapitel 1

Eine schematische Darstellung der, durch diese Objekte und der Tabelle 4.1, entstandenen Verknüpfungen ist auf der folgenden Seite in Abbildung 4.2 gegeben. Mit Hilfe der Abbildung wird verdeutlicht, dass der Hauptfokus auf dem *Logikboard* liegt. Auf diesem werden mittels Ist und Soll-Daten die Berechnungen ausgeführt, welche für das Durchführen der gewünschten Positionierung notwendig sind.
Eine weitere Funktionalität dieser Komponente betrifft die Parametrisierung der übrigen Komponenten. Soll zum Beispiel die maximale Geschwindigkeit geändert werden, welche in z-Richtung höchstens gefahren werden kann, so ist ein Zugriff auf das OD des Antriebsreglers nötig. Der gewünschte Wert wird dem *Logikboard* mitgeteilt und dieses führt die Operation mittels SDO aus.
Über die Anzahl der Komponenten ist zu erkennen, dass fünf solcher Objekte benötigt werden. Die grafische Darstellung erfolgt durch Abbildung 4.1. Mittels dieser Struktur ist es möglich, jeden beliebigen Eintrag in einem der ODs zu ändern oder auszulesen.

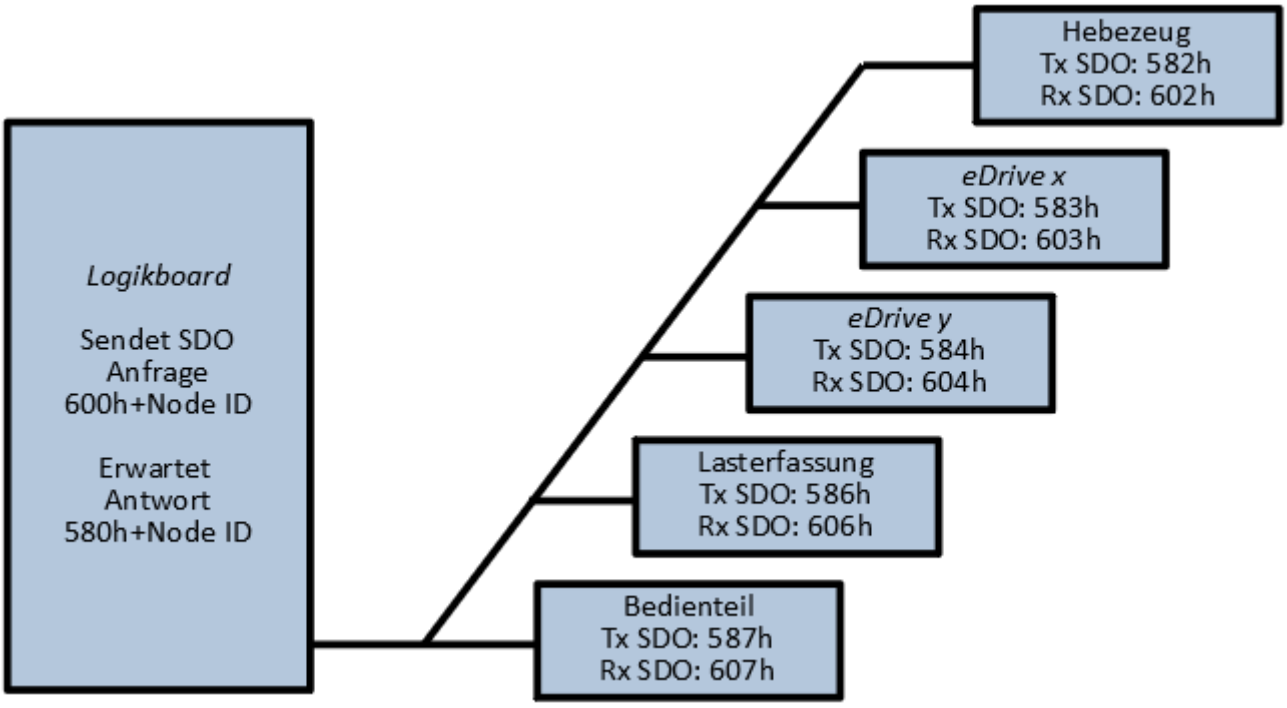



Abbildung 4.1: Struktur für Parametrisierungszugriffe via SDO

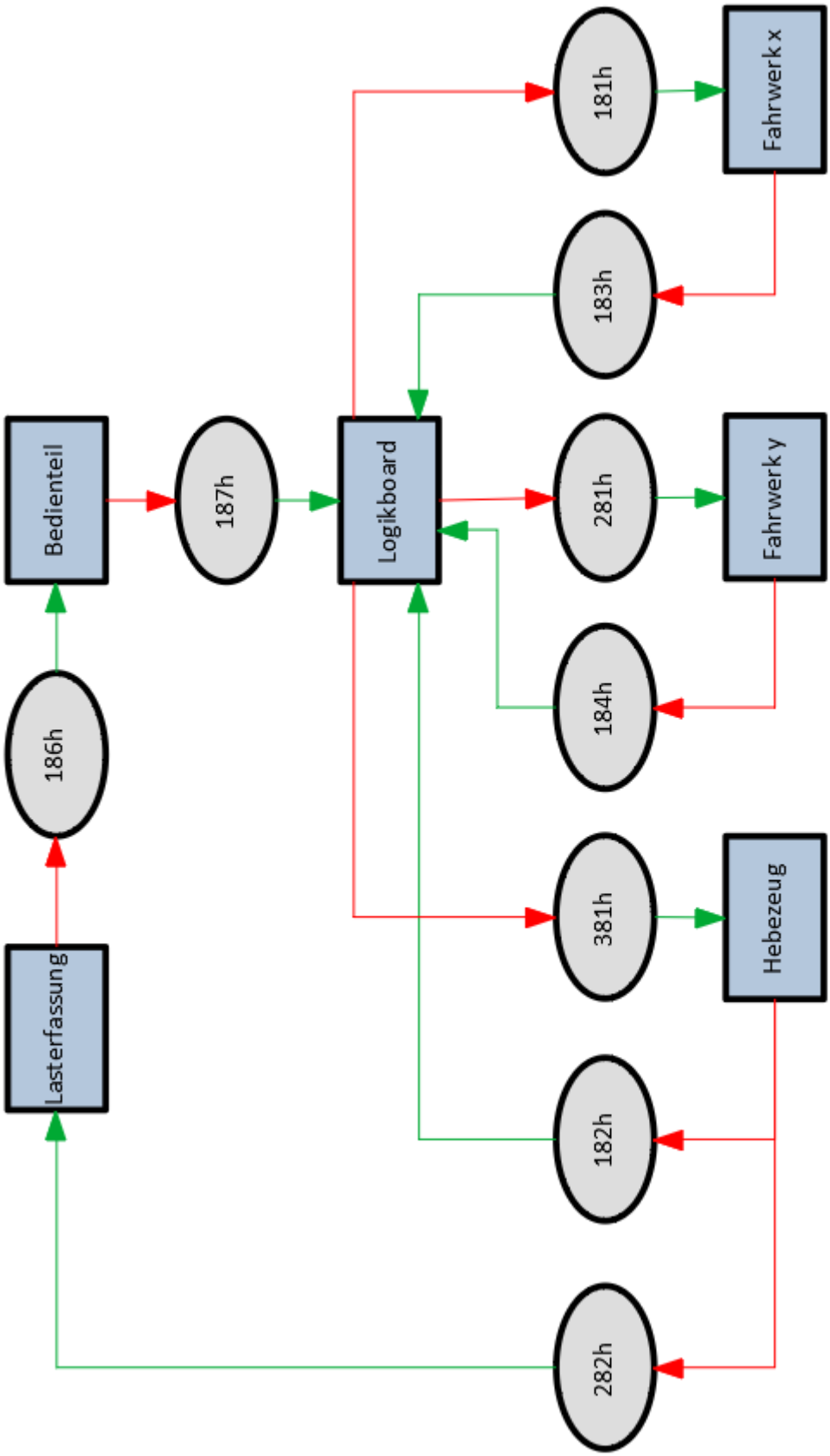



Abbildung 4.2: Schema des *CANopen*-Netzwerks

### 4.1.2 Anbindung an das industrielle Kommunikationsnetzwerk

Für die Kommunikation mit dem Umfeld ist es erforderlich, die Implementierung von Protokollen vorzusehen, welche in Fertigungsprozessen zum Einsatz kommen. Dabei handelt es ich um eine leitungsgebundene und eine leitungsungebundene Möglichkeit. Die folgenden zwei Kapitel sollen aufzeigen, welche Hardwarekomponenten vorgesehen werden müssen, um eine spätere Kommunikation zu ermöglichen. Zusätzlich erfolgt die Beschreibung, wie die Vernetzung mit den gewählten Komponenten funktioniert.

#### 4.1.2.1 *EtherCAT*

In der Automatisierungstechnik gibt es andere Anforderungen an Ethernet-basierte Kommunikationssysteme als in der Welt der Informationstechnologie. Diese Anforderungen betreffen das Regeln vieler Teilnehmer mit wenigen Prozessdaten und die deterministischen Antwortzeiten. Um die Antwortzeiten sicher bestimmen zu können, müssen harte Echzeitanforderungen an das System gestellt werden [11]. Aus der Reihe der Kommunikationsvarianten wurde für dieses Projekt das *EtherCAT*-Protokoll ausgewählt. In diesem Abschnitt wird erläutert, was das *EtherCAT* Protokoll ist, warum es verwendet und wie es integriert wird.
Ein Grund für die Verwendung von *EtherCAT* ist das deterministische Verhalten der Übertragung. Dadurch ist die Echtzeitfähigkeit gewährleistet.
Für zukünftige Tests im industriellen Umfeld steht eine *Beckhoff*-Steuerung zur Verfügung. Diese Tatsache ist ein zweiter Grund für die Verwendung des Protokolls. Das *EtherCAT* ist im Hause *Beckhoff* entwickelt wurden und bietet mit den zugehörigen Komponenten auch eine Vielzahl an Schnittstellen-Gateways zu anderen Kommunikationsprotokollen.
Dieser Ansatz ist ein dritter Auswahlpunkt. Um eine Schnittstelle zu anderen Protokollen zu schaffen, muss nicht zwangsläufig eine *Beckhoff*-Komponente eingesetzt werden. Auch andere Hersteller kommunizieren über *EtherCAT* und bieten folglich Systemteile an.
Das letzte Auswahlkriterium betrifft die Integration im Konzept. Diese ist mittels einzelner Hardwarebaugruppen durchgeführt. Für die *EtherCAT*-Kommunikation bietet die Firma *Microchip* spezifische Komponenten an.
*EtherCAT* baut auf das Standard Ethernet-Protokoll auf. Eine Kommunikation findet auf Höhe der Anwendungsschicht statt. Die zwischen dieser und dem LLC-Layer liegenden Schichten werden für eine *EtherCAT*-Kommunikation nicht verwendet. Dass es sich um ein *EtherCAT*-Frame handelt, wird durch den Eintrag der Kennung 0x88A4 im Ethernet-Typen-Feld angezeigt. Auf das Typenfeld folgen die *EtherCAT* spezifischen Frame-Inhalte. Bei diesen Inhalten handelt es sich um den *EtherCAT*-Header (mit ECAT HDR abgekürzt) und das *EtherCAT*-Telegramm, in welchem die Nutzdaten enthalten sind. In der Abbildung 4.3 ist das Frame dargestellt.
Die Unterhaltung basiert auf das Master-Slave-Prinzip. Nur der *EtherCAT*-Master darf

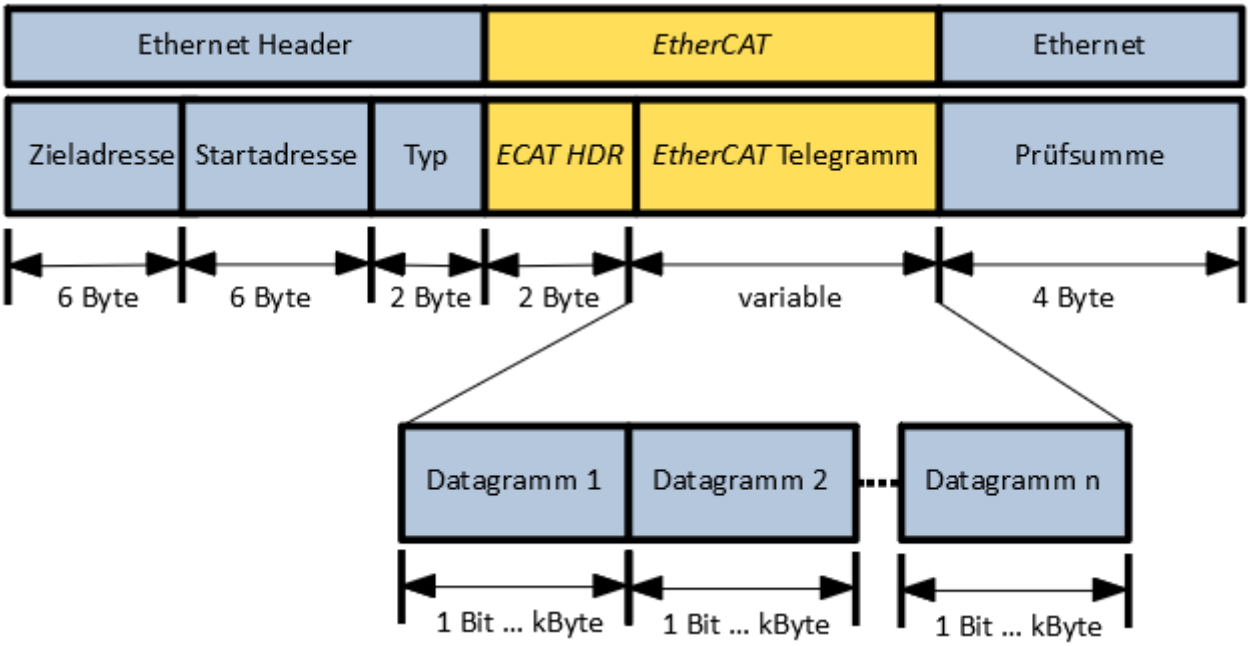



Abbildung 4.3: *EtherCAT*-Frame [11]

ein Frame senden. Über den *EtherCAT*-Header legt er fest, ob es sich um einen Lese-, Schreib- oder Lese- und Schreibzugriff handelt.
Beim Hochlauf des Netzwerkes bekommen die Slaves eine oder mehrere Adressen zugewiesen. Über diese kann der Master auf einen bestimmten (direkte Adressierung) oder mehrere Slaves (logische Adressierung) zugreifen. Mit der logischen Adressierung werden zyklische Prozessdaten ausgetauscht. Für diesen Austausch ist das Telegramm in einzelne Datagramme unterteilt. Jedes dieser Datagramme adressiert einen bestimmten Teil des Prozessabbildes. Damit kann der Master entscheiden, wann er auf welche Daten zugreift.
Das vom *EtherCAT*-Master gesendete Frame durchläuft alle Slaves. Diese nutzen einen *EtherCAT*-Slave-Controller (im weiteren Verlauf mit ESC abgekürzt) um die Daten im Durchlauf (*on-the-fly*) zu verarbeiten. Da die Verarbeitung nur in Hardware ausgeführt wird, ist die gesamte Durchlaufzeit berechenbar und hängt nicht von der Implementierung einzelner Slaves ab.
Die einzelnen *EtherCAT*-Slaves entnehmen und schreiben die Prozessdaten in die entsprechenden Datagramme. Der Kommunikationsprozess ist in Abbildung 4.4 dargestellt.
Ein durch dieses Konzept beschriebenes Hebezeug soll eine Slavekomponente im *EtherCAT*-Segment sein. Ein *EtherCAT*-Slave benötigt einen ESC. Diesen ESC stellt der *LAN9252* von *Microchip* dar. Der *LAN9252* bildet alle Schichten vom LLC bis zum PMA des *EtherCAT*-Protokolls ab [23]. Ergänzt werden muss das Bauteil um einen Ethernet-Stecker. Dieser ergänzt dann den fehlenden PMD-Layer und das MDI. Im Projekt wird

[23] für die Erläuterung der einzelnen Schichten siehe 2.3.2

ein *EtherCAT* Slave nach Norm *100BASE-TX* erstellt. Daraus folgt, dass es sich beim Übertragungsmedium um Kupfer handelt. Mit dem Medium wird der ESC auf Grundlage der Norm mittels einer RJ45 Buchse verbunden. Aus der Kombination von *LAN9252* und RJ45 entsteht ein physisch vollständiger *EtherCAT*-Controller [12, S.8].

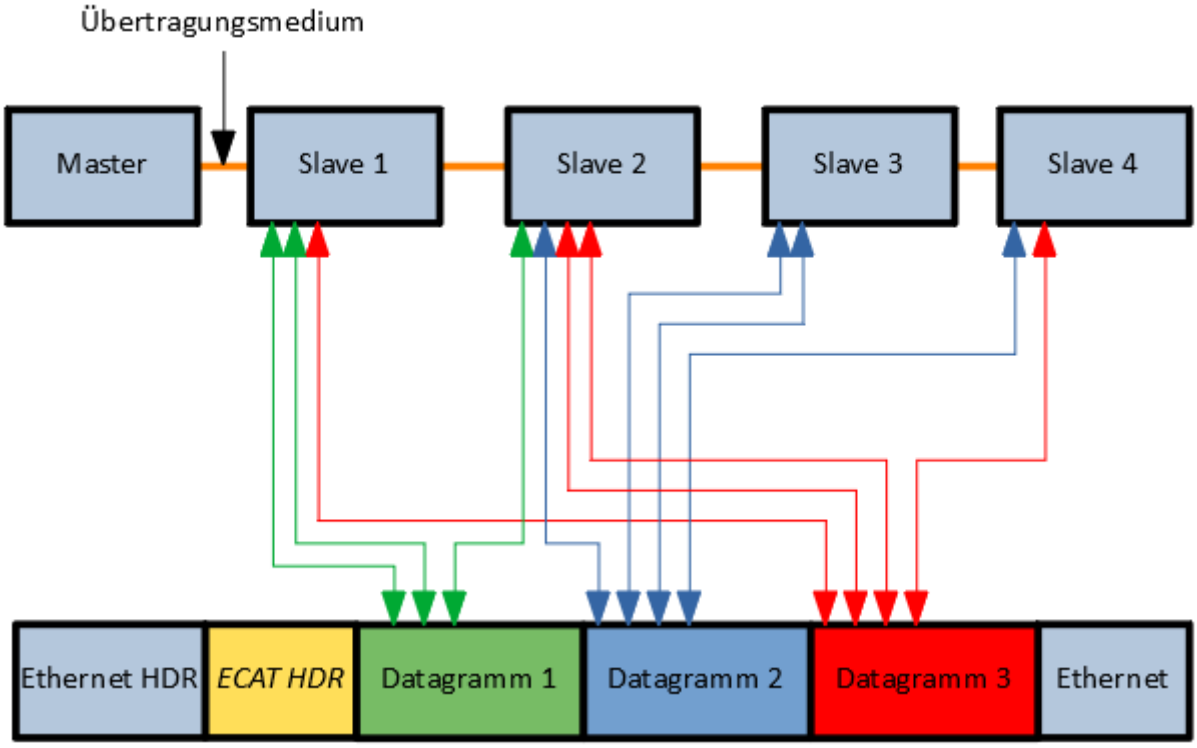



Abbildung 4.4: Kommunikationsprinzip *EtherCAT* [11]

Für den Betrieb des ESC sind noch ein Oszillator und ein EEPROM notwendig. Der Oszillator gibt die nötige Taktfrequenz von 25 MHz vor. Im EEPROM sind aller Konfigurationen und Prozessparameter enthalten, welche der spezifischen Anwendung entsprechen. Diese Konfigurationen können vom $\mu$C beschrieben und ausgelesen werden. Über die Verbindung via Ethernet zu einem PC können alle diese Zugriffe ebenfalls mittels des *TwinCAT*-Programms ausgeführt werden. Dazu wird das Programm als *EtherCAT*-Master deklariert. Vor der Erstinbetriebnahme des ESC muss die Konfiguration durch das *TwinCAT* oder einen I2C-Master durchgeführt werden. Dies ist notwendig, um Prozessparameter zu erstellen und eine Grundkonfiguration vorzugeben [31].
Der Betrieb eines *EtherCAT*-Slaves mit einem $\mu$C wird entsprechend als *Microcontroller Mode* bezeichnet. Die Kommunikation zwischen $\mu$C und ESC findet im Projekt über ein Host Bus Interface (weiterführend mit HBI abgekürzt) statt. Durch diese Konfiguration wird der $\mu$C zum HBI-Master und der ESC zum HBI-Slave.
Mit dem Vernetzen aller oben genannten Komponenten entsteht der betriebsbereite *EtherCAT*-Slave. Dieser ist in Abbildung 4.5 als Blockschaltbild dargestellt und wird anschließend hinsichtlich seiner Funktionsweise betrachtet. [12, S.62-101].
Zu Beginn ist die Betrachtung der Kommunikationsstruktur zwischen Host-$\mu$C und ESC

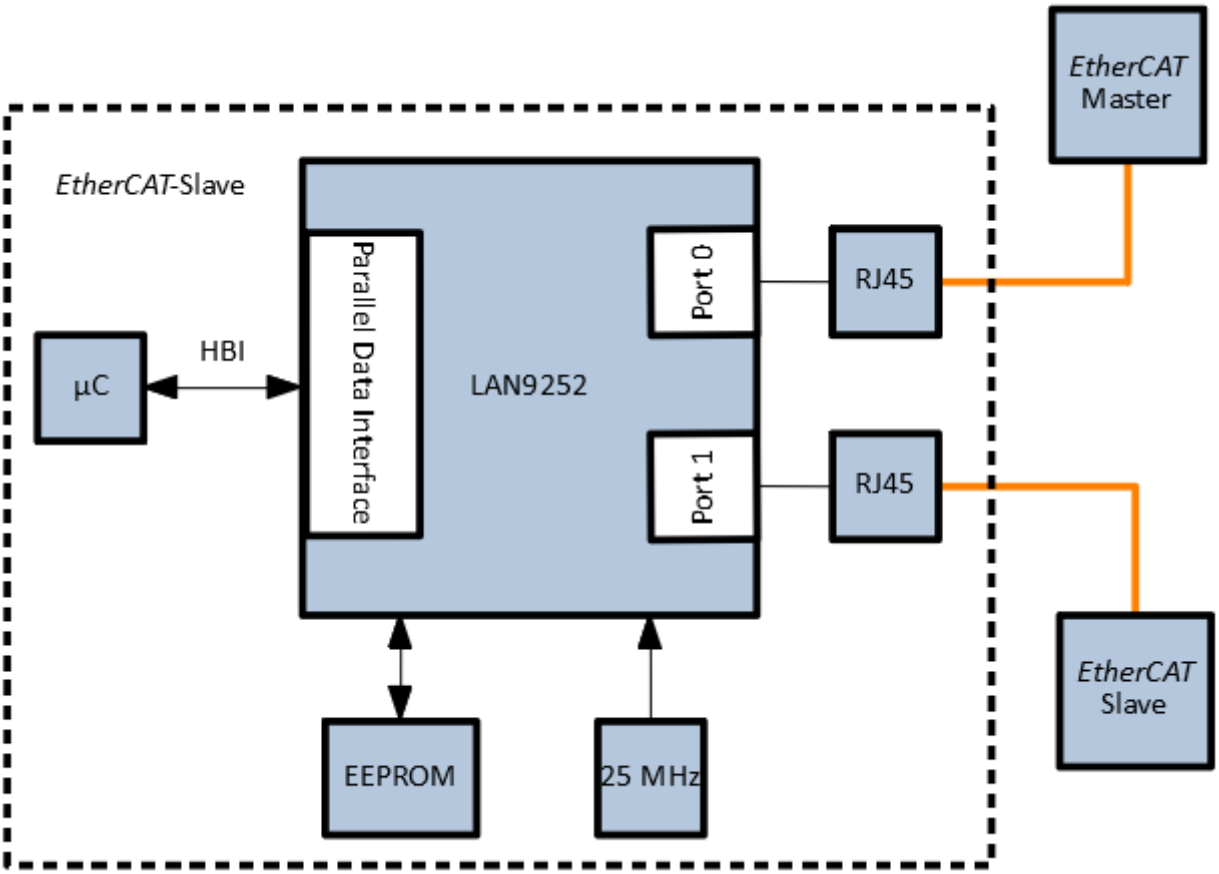



Abbildung 4.5: Blockschaltbild *EthernCAT*-Slave [12, S.9]

ausgeführt. Diese HBI kann verschieden ausgeführt werden. Die aufwendigste Methode ist von der Hardwareseite betrachtet, die des *16 Bit Multiplexed Addressing With Singel Phase Latching*. Um für eine Verwendung dieser Methode die notwendige Hardwarekapazität einzuplanen, ist die Konzeptionierung darauf ausgelegt.
Mit der Verwendung dieser Struktur ist eine Datenbusbreite von 16-Bit verfügbar. Das bedeutet, dass 16 Bit parallel über 16 Verbindungspunkte zwischen dem ESC und dem Host übertragen werden. Diese Verbindungspunkte werden als Adress und Data-Pins, kurz AD bezeichnet.
Nach der Datenbreite des Datenbusses ist die Verwendung der Datenleitungen angegeben. Beim *Multiplexed Adressing* werden die 16 Leitungen der Datenverbindung sowohl für Adressierungen als auch für Datenübertragungen verwendet.
Mit der *Single Phase Latching*-Methode geht einher, dass alle zuvor genannten Zugriffsdaten zeitgleich übertragen werden. Das *LAN9252*-Modul liest die Adressdaten in dem Moment ein, in dem eine Zustandsänderung auf der Adressierungssteuerleitung, *Adress Latch Enable Low* (bei wiederverwenden mit ALELO abgekürzt), stattfindet.
Zusätzlich zu dieser werden drei weitere Steuerleitungen benötigt. Bei diesen handelt es sich um die Anzeige eines Lese- (RD-Read) beziehungsweise Schreibzugriffs (WR-Write) und dem Indikator (CS-Chip Select), dass dieses Bauteil für den folgenden Trans-

fer ausgewählt wurde [12, S.62-101].
Der Ablauf einer Kommunikation ist in der Abbildung 4.6 visualisiert.

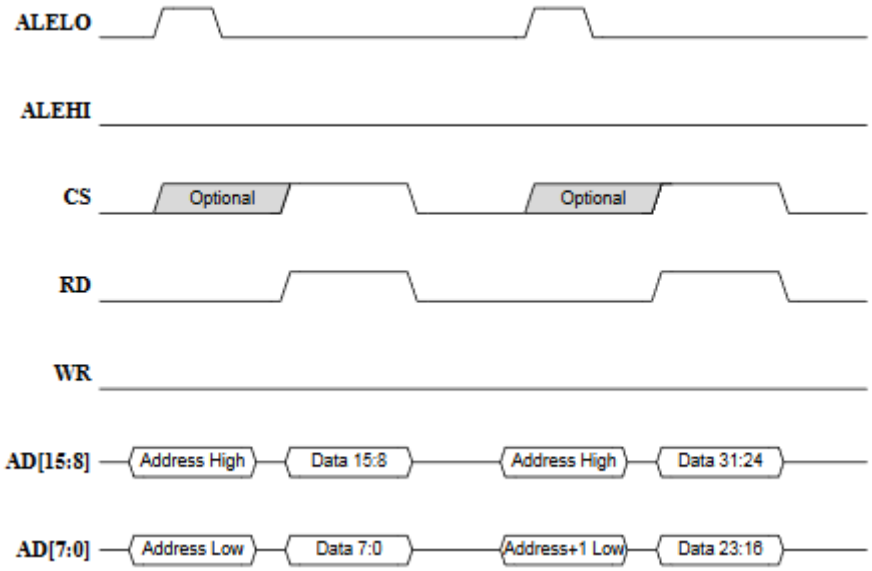



Abbildung 4.6: Ablauf des HBI Lesezugriffs [12, S.74]

Mit der steigenden Flanke des ALELO-Signals liest der ESC die Pegel an den AD[15:0] ein. Optional ist es möglich, diesen Vorgang noch in Abhängigkeit des CS-Signals zu setzen. Ist das der Fall, muss zum Zeitpunkt der steigenden Flanke der CS-Pegel bereits aktiv sein. In den eingelesenen 16 Bit befindet sich an der Stelle von Bit eins bis neun die Adresse, auf die zugegriffen werden soll, auf Bit zehn die Information, ob es sich um einen Zugriff auf den FIFO-Buffer im Prozessor RAM des ESC handelt und auf Bit elf, in welchem Datenformat die Übertragung ausgeführt wird.
Sind die Adressdaten in den ESC eingelesen, muss unabhängig von der Konfiguration das CS-Signal auf den aktiven Pegel gelegt werden. Darauf folgt beim Lesezugriff das Schalten des Pegel von RD auf eben diesen Zustand. Durch Erkennen der somit entstehenden steigenden Flanke, stellt der ESC die ab der Adresse stehenden 16 Bit auf den Datenleitungen bereit. Um ein Register mit dem DWort-Format auszulesen, muss der Vorgang mit einem um eins erhöhten Adresswert wiederholt werden.
Der Vorgang beim Schreibzugriff ist bei der Adressierung identisch zum Lesezugriff. Die Datenbereitstellung erfolgt jedoch vom $\mu$C und die steigende Flanke muss auf der WR-Leitung geschallten werden. Wird dieses Signal vom ESC erkannt, speichert er die enthaltenen Daten auf der jeweiligen Adresse [12, S.62-101].
Außerhalb der Kommunikation über das HBI ist es möglich, das Auftreten bestimmter Ereignisse zu signalisieren. Diese Möglichkeit wird durch drei Steuerleitungen zwischen Host und ESC zur Verfügung gestellt. Nummer eins ist der *IRQ*-Pin des ESC. Mit diesem Pin kann der *LAN9252* dem $\mu$C anzeigen, dass ein Interrupt vorliegt. Die Quellen, welche Ursache für ein derartiges Ereignis sind, werden über das Interrupt Configurations Register festgelegt. Wird ein Interrupt über das Aktivsetzen des *IRQ*-Pins angezeigt,

kann der Master die Ursache über das Interrupt Status Register auslesen [12, S.53].
Die Signalleitungen zwei und drei haben jeweils zwei Funktionalitäten. Ihre Funktion als Slave-Input ist das Setzen eines Zeitstempels, wenn bestimmte Events stattfinden. In dieser Anwendungsweise werden sie als *Latch*-Pins bezeichnet. In die entgegengesetzte Richtung kann der Slave dem Master das Auftreten von zeitlich abhängigen Ereignissen mitteilen. Die Bezeichnung der Pins in dieser Output-Funktion ist *Sync* [12, S.197].
Nach den Betrachtungen bezüglich der Kommunikation im *EtherCAT*-Slave und dessen Aufbau erfolgt nun die Struktur, wie *EtherCAT*-Protokolle bearbeitet werden. Der Ethernet-Frame wird vom Port 0 empfangen. Im Port wird das Frame von der *EtherCAT*-Processing-Unit (abgekürzt mit EPU) verarbeitet. Über diese EPU läuft sowohl die Kommunikation mit dem *EtherCAT*-Master als auch mit der zum Slave gehörenden Anwendung. In der projektspezifischen Betrachtung ist diese Anwendung der $\mu$C. Die EPU koordiniert alle Zugriffe auf die internen Register und den Speicherraum.
Die in der *EtherCAT*-Nachricht enthaltenen logischen Adressen werden von der Fieldbus Memory Managment Unit (FMMU) den passenden Adressen im ESC zugeordnet.
Wie der Zugriff auf die Daten erfolgt, wird vom *EtherCAT*-Master konfiguriert. Über diesen Zugriff wird geregelt, ob Daten vom Master überschrieben werden können, ohne das der $\mu$C diese zuvor ausgelesen hat oder ob das vorherige Lesen Voraussetzung zum Schreiben neuer Daten ist.
Nach der Verarbeitung erfolgt das weiterleiten an Port eins. Ist der zweite Port nicht belegt, wird das Protokoll über Port null zurückgesendet [12, S.196].

#### 4.1.2.2 Industrielles WLAN

Der Datenaustausch mit industriellen Netzwerken soll auch über WLAN möglich sein. In diesem Kapitel wird aufgezeigt, wie eine Anbindung an ein solches Netzwerk realisiert werden kann.
Zunächst muss die Hardware implementiert werden, um das Hebezeug WLAN-fähig zu gestalten. Eine Anforderung an die Hardware ergibt sich auf Basis des Einsatzgebietes. Beim WLAN handelt es sich um ein nicht leitungsgebundenes Netzwerk [24]. Durch die Tatsache, dass bei der Übertragung via Luft ein offener Zugang zu den Daten vorliegt, ist es nicht in jedem Industriebetrieb gestattet, WLAN-fähige Maschinen einzusetzen [26] [6, S. 2/3].
Daraus folgt, dass es sich um eine optional verwendete Komponente handelt. Die Flexibilität im Funktionsumfang des Hebezeugs, welche durch diese Auswahlmöglichkeit resultiert, soll auch nachträglich erhalten bleiben. Das heißt, ein Entfernen oder Nachrüsten der WLAN-Funktion soll auch zu einem späteren Zeitpunkt durchführbar sein.
Auf Grundlage dessen ist eine steckbare WLAN-Hardware ausgewählt wurden. Die Auswahl fiel auf den *ATWILC1000-SD* von *Microchip*. Bei dem in Abbildung 4.7 dargestellten Board handelt es sich um ein Secure Digital Card Interface Board (weiterführend

[24] siehe Kapitel 2.3.3

mit SD-Card bezeichnet). Die Hauptkomponente ist der WLAN-Controller *ATWILC1000-MR110PB*. Mit dessen Einsatz sind Kommunikationen über die Normen *IEEE 802.11 b/g/n* möglich. Mit dem Einsatz dieses Moduls ist die Implementierung der MAC-Schicht und der physischen Schicht abgeschlossen.


Abbildung 4.7: Hardware für WLAN [13]

Das Modul wird physisch implementiert, in dem es in den SD-Card-Connector eingesteckt wird. Die Kommunikation zwischen dem $\mu$C auf der Steuerplatine und dem WLAN-Modul ist auf zwei Arten durchführbar.
Zum einen ist die Übertragung von Daten via Seriell Peripherie Interface (abgekürzt mit SPI) möglich. Für die Anwendung im Projekt stellt bei der Verwendung einer SPI die Steuerelektronik den Master und das WLAN-Board den Slave dar. Zum anderen ist die Übertragung via Secure Digital Input Output (weiterführend mit SDIO bezeichnet).
Die gewünschte Kommunikation lässt sich auf dem *ATWILC1000-SD* über eine entsprechende Hardwarekonfiguration einstellen. Erreicht wird die jeweilige Konfiguration durch das Einlöten einer definierten Anordung von Widerständen. Im Detail sind die verschiedenen Möglichkeiten in der Anleitung zu finden [13].
Für die weitere Betrachtung wird Bezug auf die SPI-Kommunikation. Der Kommunikationspartner des WLAN-Moduls ist der $\mu$C *ATSAME70N20B*. Zusätzlich zu den Leitungen für den Datenaustausch sind eine für die Steuerung des Reset (Resetn) und eine Interrupt-Leitung vorgesehen. Nach der Betrachtung der zur Kommunikation notwendigen Hardware folgt die softwareseitige Beschreibung der Implementierung. Für die Steuerung des *ATWILC1000* steht ein Software-Treiber zur Verfügung. Dieser unterteilt sich in eine plattformabhängige und eine -unabhängige Komponente. Auf dem Modul läuft ein Linux-Betriebssystem. Um über die SPI-Schnittstelle Zugriff auf die Funktionalitäten zu erhalten ist es nötig, die plattformabhängigen Treiberteile in den Prozessor

der Steuerplatine zu implementieren. Mittels dieses Treibers entsteht eine Schnittstelle zum WLAN-Board. Das Schema zum Datenaustausch ist in Abbildung 4.8 dargestellt.

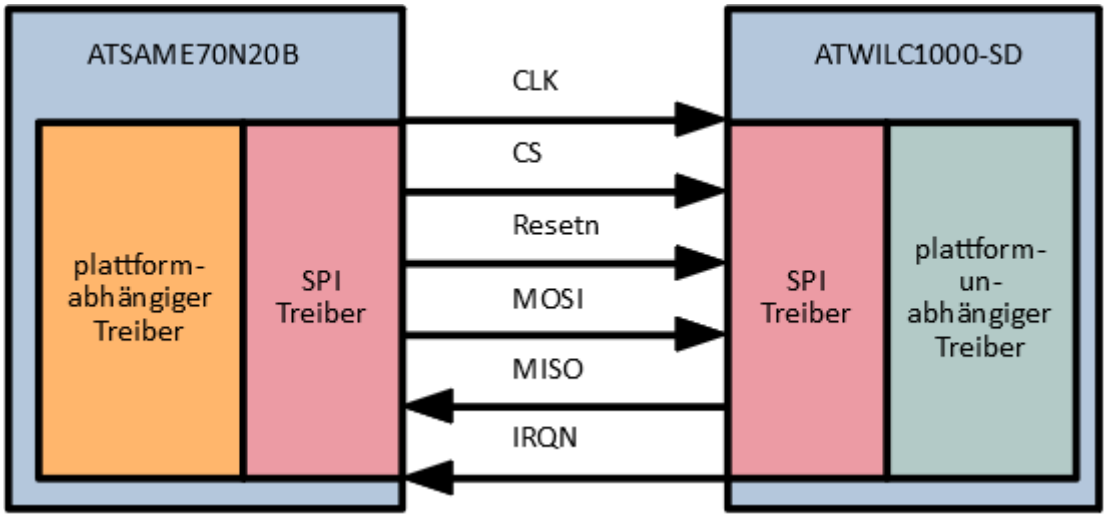



Abbildung 4.8: SPI-Kommunikation mit ATWILC1000

Diese stellt drei Routinen bereit. Eine zur Initialisierung der Schnittstelle, eine zur Rücknahme der Initialisierung und eine zur Steuerung. Mit der Funktion zur Steuerung wird auf bestimmte Merkmale der angesprochenen Hardware zugegriffen. Das heißt, es können außer Parameterabfragen auch Betriebsmodi geändert und Register geschrieben werden. Eine Kommunikation setzt sich aus dem der Operation entsprechenden Kommando und dem zugehörigen Parameter zusammen. Welche Operationen verfügbar sind und die entsprechenden Kennungen dazu sind in der Codierung (Abgebildet in Anlage F) zu finden [32] [33].

## 4.2 Flexibilisierung des Hebezeugs

Das konzipierte Hebezeug soll im Umfeld der *Industrie 4.0* eingesetzt werden. Durch den Einsatz in dieser Umgebung resultiert die Anforderung, das Hebezeug projektspezifisch anpassen zu können. Wie diese Flexibilisierung der Maschinen umgesetzt werden kann, ist Inhalt des folgenden Kapitels.
Für die Umsetzung muss zunächst erörtert werden, was das Ziel einer projektspezifischen Anpassung im thematischen Bereich der industriellen Hebetechnik ist. Die Grundaufgabe des Hebezeugs ist es, ein stoffliches Gut vertikal zu bewegen. Eine Erweiterung dieser Funktion gilt als flexible Anpassung.
Diese Anpassungen werden auch zum jetzigen Zeitpunkt vorgenommen. Mit einer solchen Maßnahme verbunden sind Änderungen an der Hardware und den Grundroutinen des Hebezeugs. Als Beispiel sei die Lastverriegelung genannt. Durch das Überschreiten einer Lastgrenze wird ein Relais geschallten. Das mittels Relais produzierte Signal verhindert ein Lösen des Lastaufnahmemittels von der Last.

Der damit verbundene Eingriff in die Hubroutinen soll vermieden und das Problem durch die Vernetzung der Komponenten gelöst werden. Für das genannte Beispiel bedeutet dies, es wird nicht die erweiterte Funktion gegeben, sondern die dafür notwendigen Parameter. Prozessparameter werden im neu konzipierten Hebezeug über TPDOs ausgetauscht [25]. Daraus resultiert, dass mit dem Zugang zum Netzwerk auch die entsprechenden Prozessdaten zur Verfügung stehen. Mit dieser Schnittstelle ist es nun möglich, Prozessdaten der Grundroutine für erweiterte Anwendungen zu nutzen, ohne in die Hard- oder Software des Hebezeugs eingreifen zu müssen.
Das projektspezifische Auswerten der Daten wird von einer externen Komponente ausgeführt. Das Programmieren dieser soll dann außerhalb der *SAS* durchgeführt werden können. Aus dieser Anforderung wurden weitere Bedingungen abgeleitet. Zur Erläuterung der Bedingungen ist es nötig, die Voraussetzungen zu definieren, die zum Programmieren der Anlage vorhanden sein müssen.

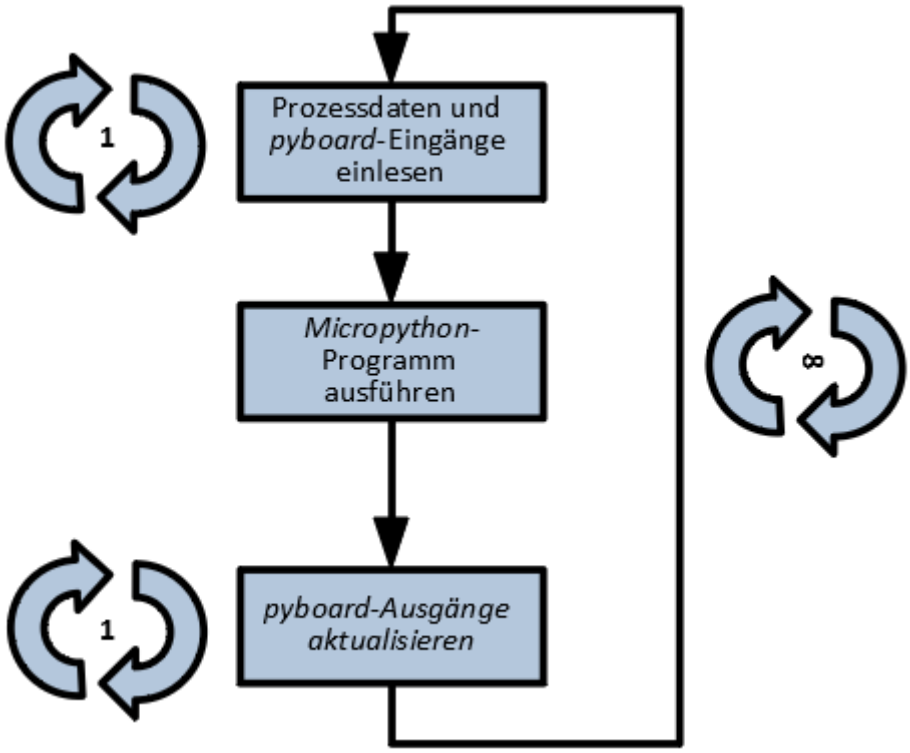



Abbildung 4.9: Ablaufschema SPS bezogen auf das *Micropython pyboard*

Es sollte ein grundlegendes technisches Verständnis vorhanden sein, ohne das dabei tiefe Programmierkenntnisse vorausgesetzt werden können. Eine Anforderung an die Schnittstelle ist, dass die Installation oder Kosten von Programmiersoftware vermieden werden soll. Dazu ist Sorge zu tragen, dass ein gewisses Minimum an Hardware verfügbar ist. Das heißt, steuerbare Ein- und Ausgänge sollen vorhanden sein [26].
Zudem soll die Schnittstellenkomponente nicht extern in einem Schaltschrank verbaut

[25] Aufgezeigt wurden die detaillierten Nachrichten in Abschnitt 4.1.1

werden, sondern sich direkt im Hebezeug befinden. Diese Anforderung beschränkt den verfügbaren Bauraum.
Wegen dieser Anforderungen fiel die Wahl auf die Integration eines *MicroPython pyboard*. Bei der Bezeichnung sind zwei Komponenten zu trennen. Die erste ist *MicroPython*. Dabei handelt es sich gleichermaßen um eine Programmiersprache und eine Laufzeitumgebung. Mit der Programmiersprache *MicroPython* ist die höhere Sprache *Python* durch Kürzung des Funktionsumfangs an die beschränkten Bedingungen im $\mu$C-Bereich angepasst wurden.
Durch Laufzeitumgebung *MicroPython* können Ports mit dem entsprechenden Kernel Echtzeitapplikationen ausführen. Ein solcher Port ist das *pyboard*. *MicroPython* selbst ist unabhängig vom Microcontroller und der zugehörigen Hardware. Im angegeben Port ist der *MicroPython*-Kernel bereits implementiert und mit der Hardware verbunden [34]. Dieser Kernel nimmt die Rolle des RTOS ein. Das gesamte *pyboard* erfüllt den Zweck einer SPS. Damit ergibt sich die Routine nach Abbildung 4.9.
Zwar ist mit der zur Verfügung gestellten Schnittstelle keine Programmierung nach *IEC 61331-3* möglich, jedoch ist die Sprache hardwarebezogen und benötigt keine Programmiersoftware [26]. Die Eingabe von Routinen erfolgt hardwarenah und portspezifisch. Dadurch ist zum Stellen von Ausgängen wenig Programmieraufwand notwendig. Zudem handelt es sich um ein offenes System, wofür viel Unterstützung im Internet angeboten wird. Neue Programme können mittels Texteditor erstellt und über eine SD-Karte aufgespielt werden.
Von der Hardware betrachtet sind alle benötigten Komponenten verfügbar. Das *pyboard* bietet einen CAN-Controller und eine Anzahl an weiterer Peripherie. Auf dem *pyboard* nicht enthalten sind der CAN-Transceiver und die entsprechende galvanische Trennung der Ein- und Ausgänge.

## 4.3 Verschaltung Gesamtsystem

In diesem Abschnitt erfolgt die Betrachtung des Gesamtsystems. Dabei wird zunächst in Abschnitt 4.3.1 die Vernetzung der Einzelkomponenten zum Hebezeug betrachtet.
Auf die funktionelle Betrachtung folgen Besonderheiten beim Hardwaredesign und Schaltplanentwurf.

### 4.3.1 Vernetzungsplan

Inhalt dieses Kapitels ist die Verschaltung aller Komponenten [27]. Zur Visualisierung dient die Abbildung 4.10, auf Seite 64. In dieser sind alle Komponenten des Systems mit den verwendeten Kommunikationsstrukturen dargestellt.
Es ist nicht notwendig, alle Komponenten in jedes Projekt einzubinden. Das Durchfüh-

[26] die *IEC 61331-3* beschreibt textuelle und grafische Programmierung
[27] die Komponenten wurden in den Abschnitten 3.1 bis 4.2 betrachtet

ren der vertikalen Bewegung wird durch die im Bereich I enthaltenen Baugruppen realisiert. Der Lastkollektivspeicher gehört deshalb in diesen Sektor, weil die Berechnung des Abnutzungsvorrats für das Hebezeug verpflichtend wird [26].
Die Bauteile in den mit II, III und IV gekennzeichneten Feldern sind abhängig vom jeweiligen Projekt. Handelt es sich um eine elektrische Krananwendung, so ist der Bereich II zu integrieren. Je nach Freiheitsgrad sind eine oder zwei *Nanotec*-Steuerungen zu verwenden.
Der Bereich, welcher mit III markiert wurde, ist abhängig vom ausgerüsteten Bedienelement.
Über den Einsatz der *pyboard*-Komponente IV können projektspezifische Steuerungsroutinen integriert werden [28]. Auch die notwendige Hardware für eine Kommunikation über das *EtherCat*-Protokoll V oder WLAN VI ist optional. Durch den modularen Aufbau des Gesamtsystems kann das Hebezeug oder der Kran spezifisch an jedes Projekt angepasst werden.
Die Aufgaben, die mit dem jeweiligen Modul bewerkstelligt werden können, sind in der Tabelle 4.4 aufgeführt. Im nachfolgenden Kapitel 4.3.2 sind die Besonderheiten ausgeführt, die beim Design zu beachten sind.

| **Nummer** | **Komponente** | **Funktion** |
|---|---|---|
| I | Hebezeug | -Vertikale Bewegung (Z-Achse)<br>-Abnutzungsvorrat berechnen<br>-Parametrisierung<br>-Zustand visualisieren |
| II | Elektrisches Fahrwerk | -Horizontale Bewegung (X- oder Y-Achse) |
| III | *Balancer* | -Bedienersteuerung<br>-Zustandsvisualisierung<br>-Parametrisierung |
| IV | SPS | -Projektspezifische Funktionen |
| V | Ethernet | -Anbindung an *EtherCat*-Netzwerk |
| VI | WLAN | -Anbindung an WLAN-Netzwerk |

Tabelle 4.4: Funktionen der Module

[28] siehe Kapitel 4.2

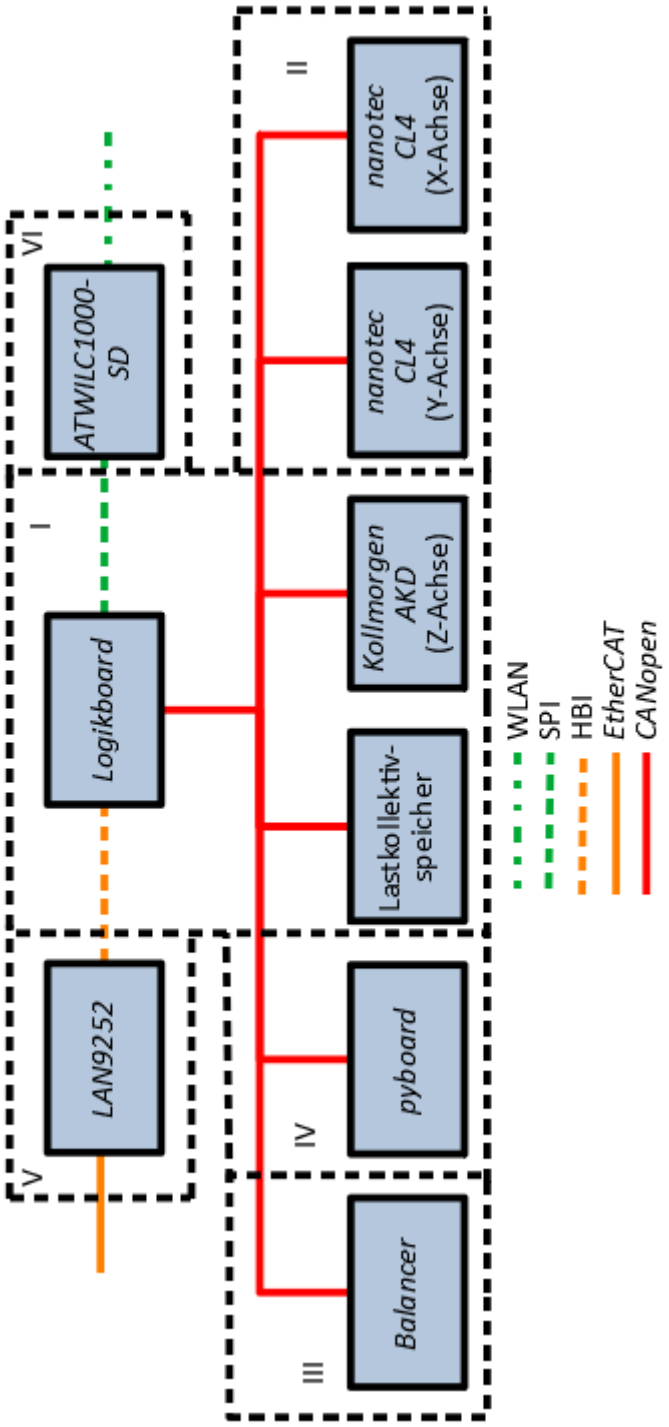



Abbildung 4.10: Vernetzung im Hebezeug

### 4.3.2 Hardwaredesign und Schaltungslayout

Mit diesem Kapitel sollen die Bedingungen aufgezeigt werden, welche die Auslegung der benötigten Hardware beeinflussen. Für das Hardwaredesign sind drei Faktoren maßgebend. Der zur Verfügung stehende physische Raum, die Funktionen der Komponenten und die Spannungsversorgung. Unter Beachtung dieser Maßgaben ist die Anordnung der Hardwarebestandteile vorzunehmen.
Der Raum, welcher zum Platzieren der Hardware zur Verfügung steht, ergibt sich aus den Dimensionen des Gehäuses. Das Gehäuse besteht aus Ober- und Unterteil. In beiden Komponenten kann eine Leiterkarte eingefügt werden.
In das Unterteil ist eine Montage der jeweiligen Leiterkarten für das *Logikboard*, die SPS und den Lastkollektivspeicher vorgesehen. Jede dieser Komponenten sitzt in einem eigenen Gehäuse. Das Oberteil des Gehäuses vom *Logikboard* wird mit der Platine bestückt, welche die nötige Hardware für die *EtherCat* und WLAN Kommunikation umfasst. Eine Kommunikation über eine dieser Schnittstellen erfolgt durch den $\mu$C des *Logikboard*. Die Verlagerung der Bauteile auf eine externe Platine ist Folge des eingeschränkten physischen Raumes. Eine Darstellung des verwendeten Gehäuses ist in Abbildung 4.11 gegeben. Die Erläuterung der darin enthaltenen Bezeichnungen erfolgt durch Tabelle 4.5.

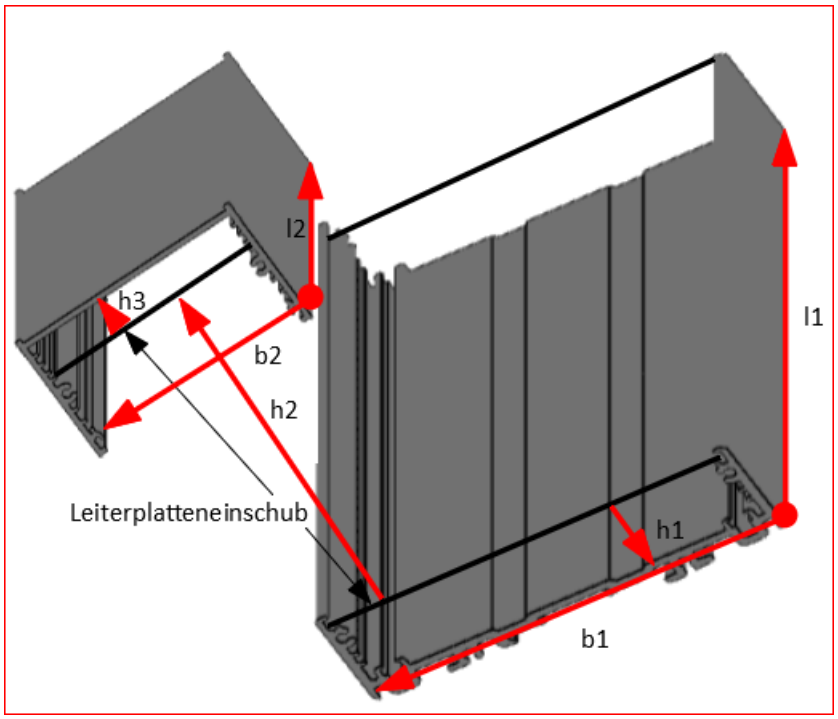



Abbildung 4.11: Hutschienengehäuse

| Bezeichnung | Bedeutung |
|---|---|
| l1 | Länge des Gehäuseunterteils |
| b1 | Breite des Gehäuseunterteils |
| h1 | Höhe Leiterplatte bis Boden Gehäuse |
| h2 | Höhe untere bis obere Leiterplatte |
| h3 | Höhe Leiterkarte bis Decke Gehäuse |
| l2 | Länge des Gehäuseoberteils |
| b2 | Breite des Gehäuseoberteils |

Tabelle 4.5: Abmessungen Gehäuse

Das Ermitteln der aktuell anhängenden Last ist, Aufgabe des Lastkollektivspeichers [29]. Die bisherige Erfassung im *Balancer* kann somit entfallen.
Die Übertragung der Prozessdaten wird zwischen den Modulen des Kranes oder Hebezeugs über *CANopen* realisiert. Dazu ist es nötig, die entsprechende Hardware zu integrieren. Die Kommunikation findet über den Baustein *MCP25625* statt. Dabei handelt es sich um einen CAN-Controller. Er empfängt die Nachrichten und wandelt sie in eine über SPI auslesbare Struktur um. Ein Transceiver ist nicht ausreichend, da der verwendete *PIC24FJ128GB204* keine CAN-Peripherie umfasst.
Die übrigen Komponenten sind im Projekt konzipiert wurden. Einen Überblick über die jeweils benötigte Hardware liefern die Schaltpläne in Anlage A bis E.
Grundlage dieser Aufteilung ist, außer der verfügbare Platz im Gehäuse, noch die Funktion der Module. Das *Logikboard* fungiert als zentrale Steuereinheit und ist das Bindeglied zwischen Hubsystem und industriellen Umfeld. Aus diesem Grund muss die *Kommunikationsplatine* nur mit dem*Logikboard verbunden werden*. Diese wird über die Stecker *HBI*, *ETHER_CONTR*, *WLAN* und *POWER_KOMMU* realisiert.
Das Vernetzungsschema nach Abbildung 4.10 kann auf zwei Arten umgesetzt werden. Eine Möglichkeit ist durch die 2x8 - poligen *Phoenix*-Stecker gegeben. Diese werden am Rand der Platinen aufgesetzt und ermöglichen das Zusammenfügen der Module.
Mittels der Verkabelung der Module ergibt sich Variante zwei. Jeder CAN-Anschluss wird mit denen der übrigen Komponenten über Leitungen verbunden.

| Modul | Stromaufnahme [mA] |
|---|---|
| Kommunikation | 410 |
| *pyboard* | 445 |
| Lastkollektivspeicher | 490 |
| Balancer | 580 |
| *Logikboard* | 590 |
| *Kollmorgen AKD* | 250 |
| Fahrwerk Y-Achse | 590/ 6000 |

Tabelle 4.6: Stromaufnahme der Module

[29] Verfahren wird in Abschnitt 3.2.1 aufgezeigt

Der dritte Faktor, welchen es zu beachten gilt, ist die Stromversorgung. Um eine entsprechende Struktur zu schaffen, wurden die Stromaufnahmewerte der einzelnen Komponenten ermittelt. Aufgeführt sind die Werte in Tabelle 4.6. Für das elektrische Fahrwerk zur Positionierung in der Y-Koordinate sind zwei Werte eingetragen. Der erste betrifft die Stromaufnahme der Logik. Die 6 A werden zum Betrieb des Motors benötigt. Da das System mit Netzspannung gespeist wird, ist eine Spannungswandlung notwendig. Über ein Netzteil wird diese Spannung zunächst in 24 V Gleichspannung umgesetzt. Mit den 24 V werden Signalübertragungen durchgeführt und die einzelnen Module versorgt. Innerhalb der Komponenten wird mit 5 V, 3,3 V und 1,2 V Spannungsniveau gearbeitet.
Die Wandlung von 24 V auf 5 V realisiert ein *SPU03N-05*. Bei diesem Bauteil handelt es sich um einen DC-DC-Wandler. Dieser kann bis zu 600 mA bei 155 mA Stromaufnahme liefern. Anhand der Abbildung 4.12 ist zu erkennen, dass sieben solcher Module im Hebezeug vorhanden sind. Daraus resultiert eine Gesamtstromaufnahme von 1085 mA für die Logikkomponenten. Zusätzlich muss das Netzteil die bis zu 6 A Betriebsstrom des Fahrwerkmotors leisten können. Auf Grundlage der verfügbaren Module und derer Abstufungen ist dem System ein 10 A Netzteil vorzuschalten.
Das Fahrwerk für die Positionierung auf der X-Achse wird extra versorgt. Dies ergibt sich aus der Einbauposition des Elements. Ein Anschluss an den 24 V Kreis des Kranes würde das Verlegen von externen Leitungen inkludieren, da dieser Spannungspegel nicht über die in den Profilen liegenden Stromschienen anliegt.
In der Abbildung 4.12 ist das 24 V Spannungsnetz visualisiert.

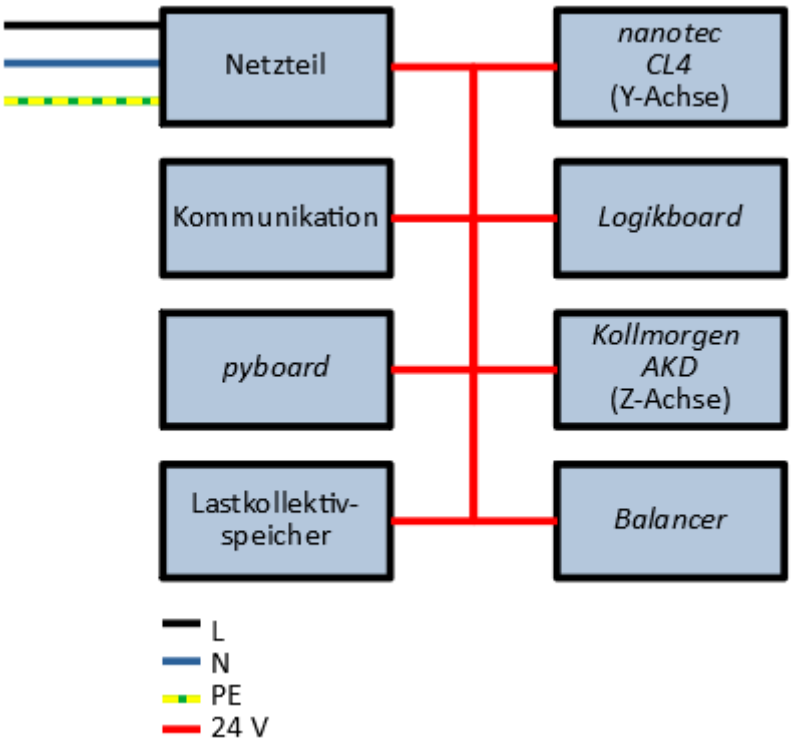



Abbildung 4.12: Vernetzung der Spannungsversorgung

# 5 Fazit

Mit diesem abschließenden Kapitel wird das bearbeitete Projekt zusammengefasst. Die Zusammenfassung ist in drei Teile gegliedert. Im ersten werden die Neuerungen aufgezählt, welche im Gegensatz zum jetzigen System und dem Stand der Technik erreicht wurden.
Durch Abschnitt 5.2 werden offene Themenkomplexe und die nächsten Schritte angegeben. Im letzten Teil des Kapitels erfolgt eine Bewertung des erzielten Ergebnisses.

## 5.1 Vergleich zum aktuellen technischen Stand

Mit diesem Kapitel wird aufgezeigt, wie sich das konzipierte Hebezeug zum einen vom bisherigen *SAS*-Hebezeug und zum anderen von den Produkten der anderer Hersteller unterscheidet [30].
Die systeminternen Signale werden über das *CANopen*-Protokoll übertragen. Durch diese Vernetzung wird der Verkabelungsaufwand reduziert und die Übertragung der Daten ist flexibler. Eine Einbindung neuer Komponenten setzt nur voraus, dass die Maximalanzahl der *CANopen*-Nodes noch nicht erreicht und der neue Teilnehmer *CAN*-fähig ist.
Optionale Teilnehmer im konzipierten Hebezeug sind unter anderen die elektronischen Fahrwerke der *SAS*. Je nach Anwendung sind bis zu zwei *eDrive* im System verbaut. Daraus resultiert die Möglichkeit der automatischen, dreidimensionalen Positionierung.
Das SPS-Modul bietet die Möglichkeit, systemnahe Funktionen projektspezifisch zu erstellen und zu integrieren. Mit der Einbindung der SPS in das *CANopen*-Netzwerk stehen den Routinen alle Prozessdaten zur Verfügung. Die Erstellung der Routinen erfolgt soft- als auch hardwareseitig unabhängig von den Grundfunktionen des Hebezeug. Wie in Abschnitt 1 beschrieben, war dieser Eingriff im aktuellen System nur durch Veränderung der Soft- und Hardware möglich.
Eine weitere Veränderung gegenüber herkömmlichen Hebezeugen stellt die Kommunikation zum industriellen Umfeld dar. Dieser Datenaustausch war bisher nicht vorgesehen. Im Konzept sind die Grundlagen erarbeitet wurden, um die Kommunikation mit dem Umfeld herzustellen. Diese ist via *EtherCAT*, *CANopen* oder WLAN möglich.
Die Wettbewerber *Gorbel*, *Indeva* und *Demag* setzen auf eine WLAN-Übertragung der Daten. Die Firma *Liftket* visualisiert alle Prozessdaten einzig über ein integriertes Display.
Eine Kommunikation zur Umgebung über WLAN ist, ebenso wie die Visualisierung des Systemzustands über ein HMI, Systembestandteil. Durch die zusätzlichen Varianten der Kommunikation über Ethernet oder Feldbus ist das System flexibler.
Flexibilität ist auch das folgende Unterscheidungsmerkmal. Bei einer projektbezoge-

[30] der aktuelle technische Stand ist in Kapitel 2.1.3 gegeben

nen Anpassung der Funktionen wird bei *Gorbel* die komplette Steuereinheit getauscht. In den anderen Systemen ist eine Anpassung, zum jetzigen Zeitpunkt, nicht vorgesehen.

## 5.2 Weiterführende Aufgaben

Für die Umsetzung der zuvor beschriebenen Funktionen sind weitere Schritte notwendig. Diese sind in fünf Ausbaustufen eingeteilt. Die Abbildung 5.1 visualisiert den Gesamtprozess zur Entwicklung des *Hebezeugs 4.0*. Jede der Ausbaustufen umfasst das Design und die Umsetzung aller notwendigen Hard- und Softwarekomponenten. Mit dieser Konzeptionierung sind die Anforderungen, welche solch einer Entwicklung zu Grunde liegen, aufgezeigt.
Im Anschluss an die Konzeptionierung folgt die Umsetzung der Ausbaustufe I. Diese erweitert die bislang verfügbaren Funktionalitäten des Hebezeugs mit der Verwendung des *CANopen*-Protokolls. Durch die *CANopen*-Kommunikation findet der Austausch der Prozessdaten statt, die für die Grundfunktionen von Bedeutung sind [31]. Nach dem Aufbau der Kommunikation sind noch offene Festlegungen zu treffen. Dazu zählt beisielsweise, ob ein NMT-Master integriert werden soll.
Die Implementierung von Flugschreiber und Lastkollektivspeicher bildet die Ausbaustufe II. Bei der Ablage der Daten ist zu prüfen, ob die optimale Struktur verwendet wird. Eine Verbesserung der Speicherausnutzung kann beim Flugschreiber eventuell durch Nutzung eines UNIX-Zeitstempels erreicht werden.
Sind diese Funktionen des Systems erstellt, folgt die Einbindung der SPS. Dabei ist die Priorität zwischen Grundfunktionen und spezifischen Routinen zu bewerten. Das bedeutet die Vergabe der Zugriffsrechte der SPS auf Grundfunktionen und Prozessdaten. Beispielsweise ist zu definieren, wann eine SPS-Routine eine Positionierung auslösen oder verhindern darf.
Nach erfolgreicher Integration der Ausbaustufe III erfolgt die Integration der *EtherCAT*-Kommunikation. Die dabei erforderlichen Betrachtungen betreffen die Qualität des Datenaustauschs. Das bedeutet, welche Daten können ausgelesen und welche Befehle können von extern ausgeführt werden.
Die letzte Ausbaustufe umfasst die leitungsungebundene Kommunikation mit industriellen Netzwerken. Für die Übertragung von Prozessdaten über WLAN gilt es unter anderem das Ziel zu definieren, auf das die Daten übermittelt werden sollen. Als Optionen stehen die Übertragung auf andere Stationen oder das Hochladen der Daten in eine Cloud bereit. Die Varianten sind anschließend zu testen.
Des Weiteren ist im Design der WLAN-Verbindung die Sicherheit der Daten mittels Expertenrat sicherzustellen.

[31] siehe Kapitel 4.1.1

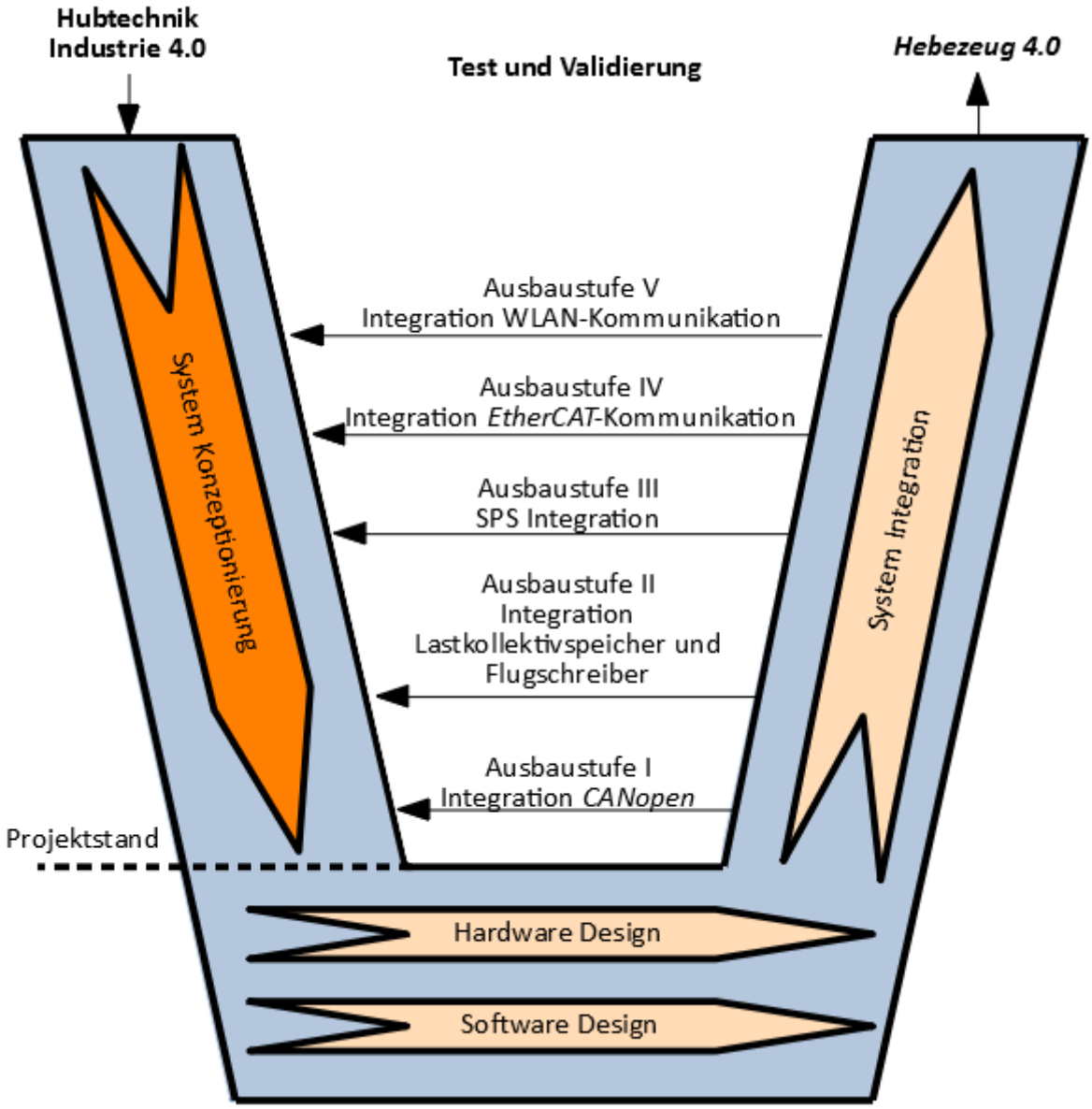


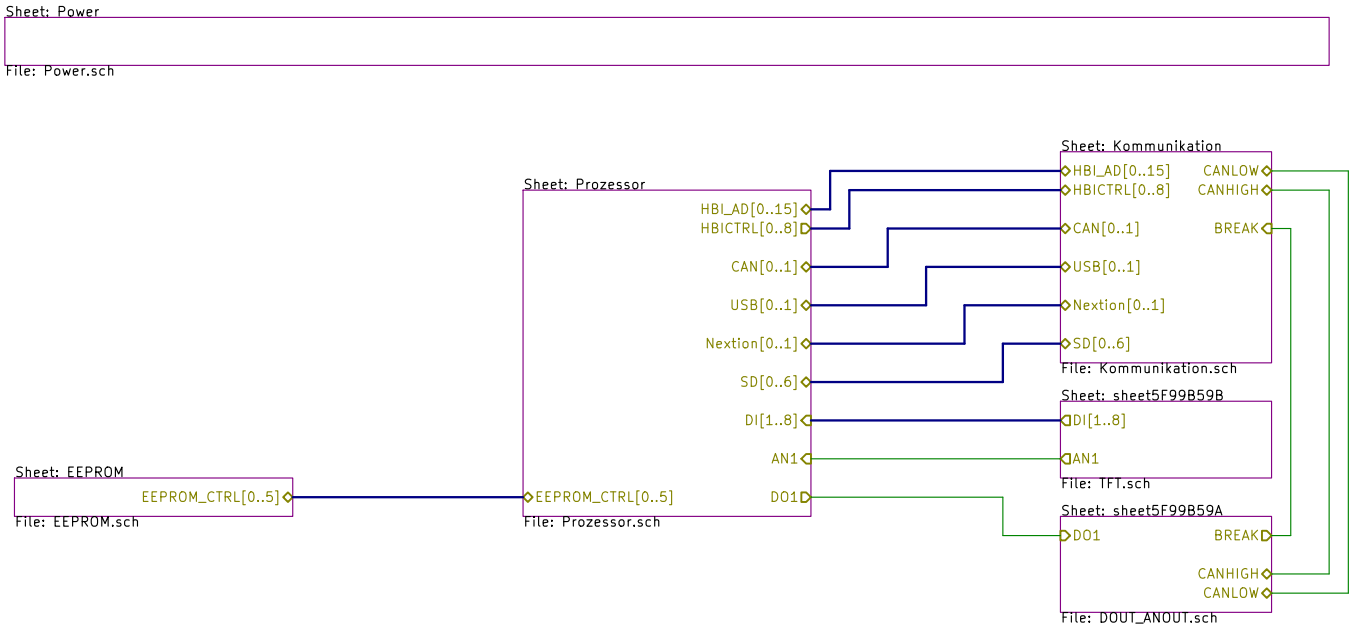

Abbildung 5.1: Darstellung der Folgearbeiten im V-Modell [14]

## 5.3 Bewertung des erzielten Resultates

Die Bewertung des aufgestellten Konzeptes ist Inhalt dieses Kapitels. Ziel der Arbeit war es, ein Konzept zu entwickeln, mit dem die Hebezeuge der *SAS* die Techniken der *Industrie 4.0* unterstützen. Die Anforderungen, die es dabei zu bewältigen gilt, stammen aus dem Schnittfeld der beiden Themengebiete. Die *Industrie 4.0* beschreibt die Verbesserung des Gesamtprozesses durch Vernetzung und Flexibilisierung aller Beteiligten.
Für die Erfüllung dieser Aufgaben wurden verschiedene Ansätze und Strukturen vorgegeben. Zur Vernetzung des Hebezeugs stehen in der Ausbaustufe V drei Möglichkeiten zur Auswahl [32]. Es kann an Netzwerke angeschlossen werden, welche über das *CANopen* oder *EtherCat* Protokoll kommunizieren. Zusätzlich besteht die Möglichkeit einer drahtlosen Verbindung.
Der Bezug und die Verteilung der notwendigen Daten ist sichergestellt. Außerdem können relevante Daten für Wartung und Historie gesichert werden. Die Speicherung der Wartungsdaten wird durch einen Lastkollektivspeicher ausgeführt. Daten, die für eine spätere Analyse der Nutzung hinterlegt werden sollen, stehen im Flugschreiber.
Zur Umsetzung des Konzeptes in die Praxis sind noch Aufgaben offen. Eine Anbindung an Netzwerke, welche beispielsweise über *ProfiNet* oder *Modbus* kommunizieren, ist nicht betrachtet wurden. Eine Realisierung der Anbindung an solch ein Netzwerk erfordert eine anwendungsspezifische Lösung.
Weiterführend ist das Vernetzungsschema im Bereich der leitungsungebunden Übertragung noch offen. Das bedeutet, es müssen noch das Ziel und Protokoll der Übertragung bestimmt werden.
Für eine erhöhte Flexibilität ist die *pyboard*-Komponente konzipiert wurden. Um dieses Modul nutzen zu können, sind grundlegende Programmierkenntnisse beim Anwender vorausgesetzt. Es muss durch eine Testphase festgestellt werden inwieweit Änderungs- oder Erweiterungsbedarf der SPS besteht. Erweiterungsbedarf umschreibt dabei die Erstellung von Handbüchern oder Softwareunterstützung durch Mitgabe von extra Programmen.
Insgesamt erfüllt das erstellte Konzept theoretisch alle Anforderungen aus den Bereichen Hebetechnik und *Industrie 4.0*. Auftretende Fehler oder Lücken im Konzept müssen während der Entwicklung der einzelnen Ausbaustufen mittels Anpassungen oder Ergänzungen behoben werden.

[32] für die Inhalte der Ausbaustufen siehe Kapitel 5.2

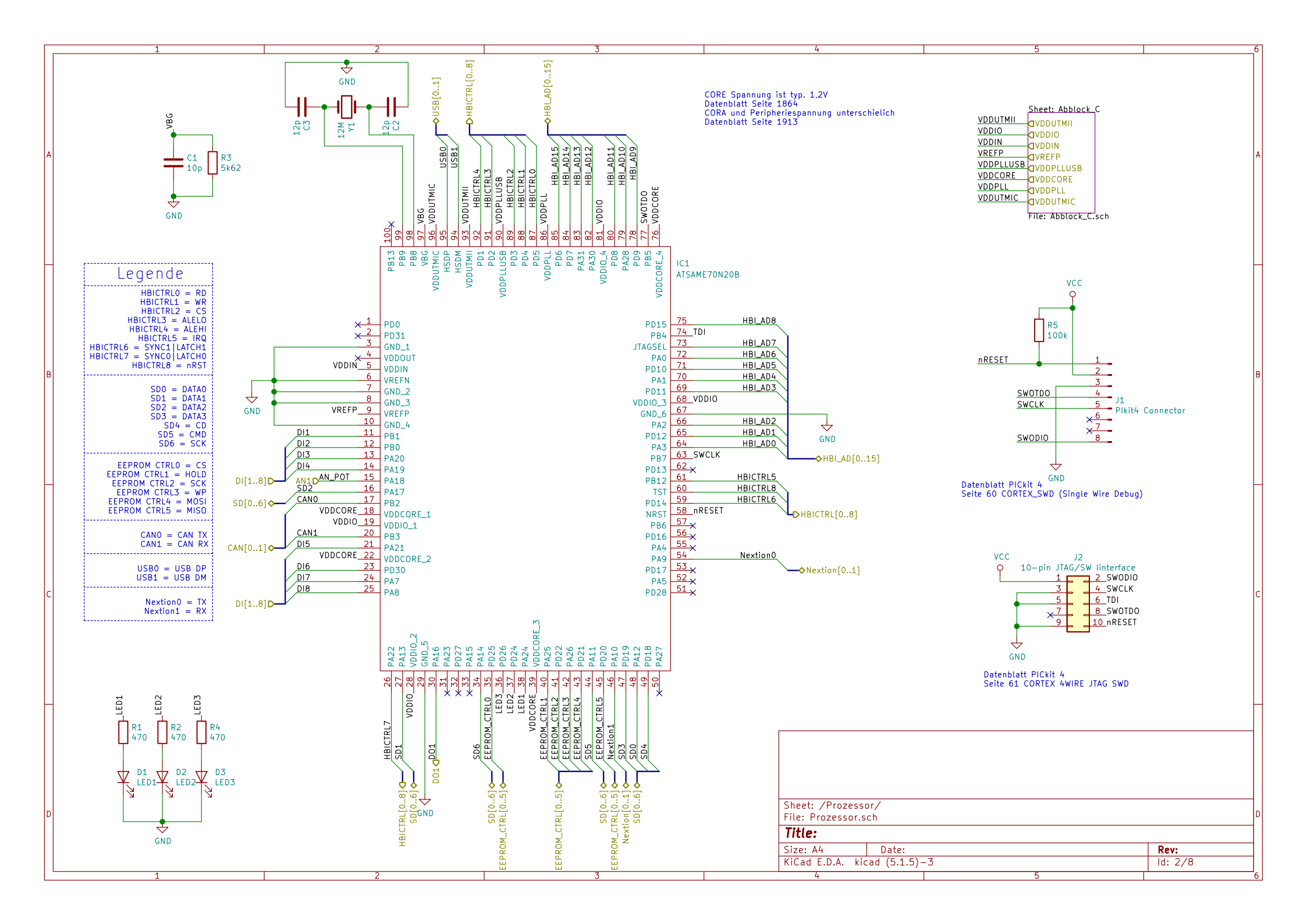
# Anhang A: Schaltplan Logikboard

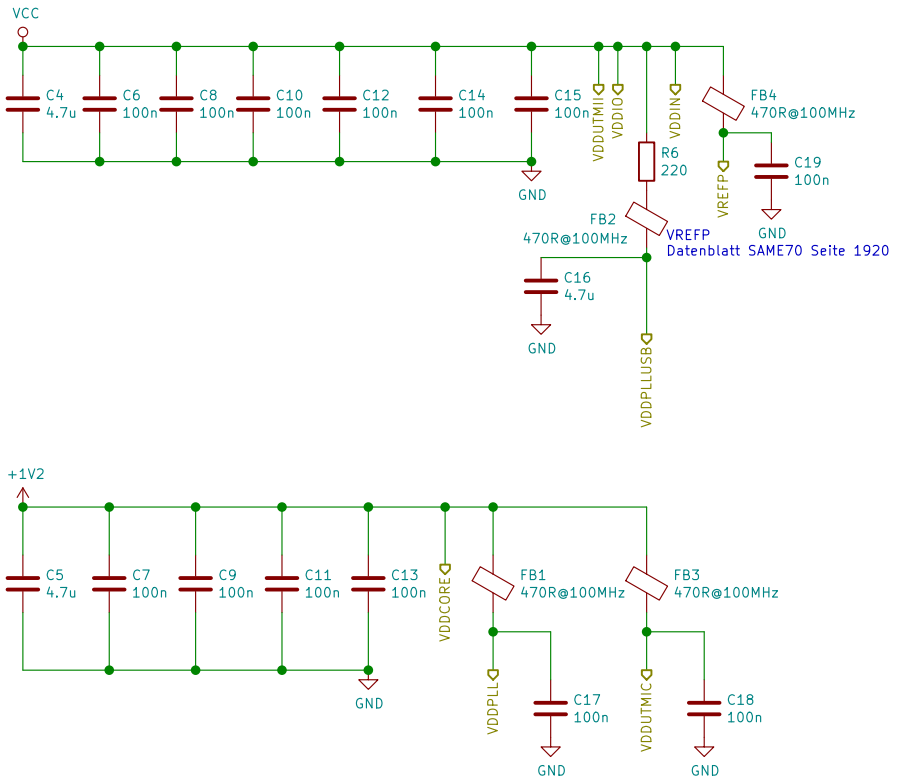



Sheet: /
File: SAS_4.0_Platine.sch

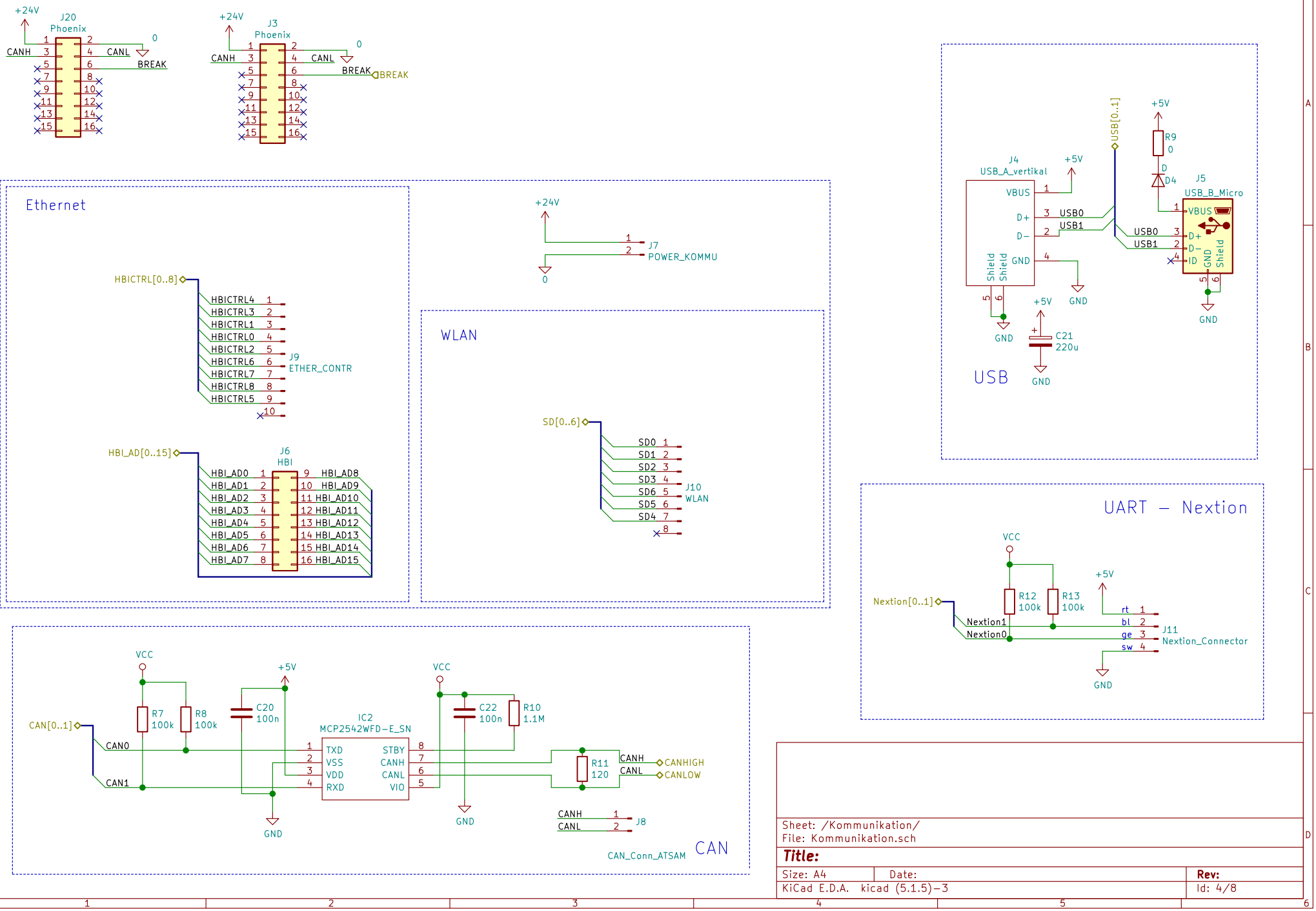
**Title:**

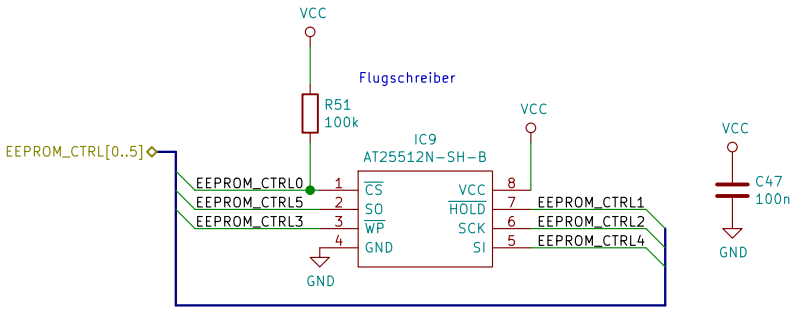
| Size: A4 | Date: | **Rev:** |
|---|---|---|
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 1/8 |


CORE Spannung ist typ. 1,2V
Datenblatt Seite 1864
CORA und Peripheriespannung unterschiedlich
Datenblatt Seite 1913
IC1
ATSAME70N20B
Legende
HBICTRL0 = RD
HBICTRL1 = WR
HBICTRL2 = CS
HBICTRL3 = ALELO
HBICTRL4 = ALEHI
HBICTRL5 = IRQ
HBICTRL6 = SYNC1|LATCH1
HBICTRL7 = SYNC0|LATCH0
HBICTRL8 = nRST
SD0 = DATA0
SD1 = DATA1
SD2 = DATA2
SD3 = DATA3
SD4 = CD
SD5 = CMD
SD6 = SCK
EEPROM CTRL0 = CS
EEPROM CTRL1 = HOLD
EEPROM CTRL2 = SCK
EEPROM CTRL3 = WP
EEPROM CTRL4 = MOSI
EEPROM CTRL5 = MISO
CAN0 = CAN TX
CAN1 = CAN RX
USB0 = USB DP
USB1 = USB DM
Nextion0 = TX
Nextion1 = RX
Sheet: Abblock_C
File: Abblock_C.sch
Datenblatt PICkit 4
Seite 60 CORTEX_SWD (Single Wire Debug)
Datenblatt PICkit 4
Seite 61 CORTEX 4WIRE JTAG SWD
Sheet: /Prozessor/
File: Prozessor.sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad (5.1.5)-3
Id: 2/8



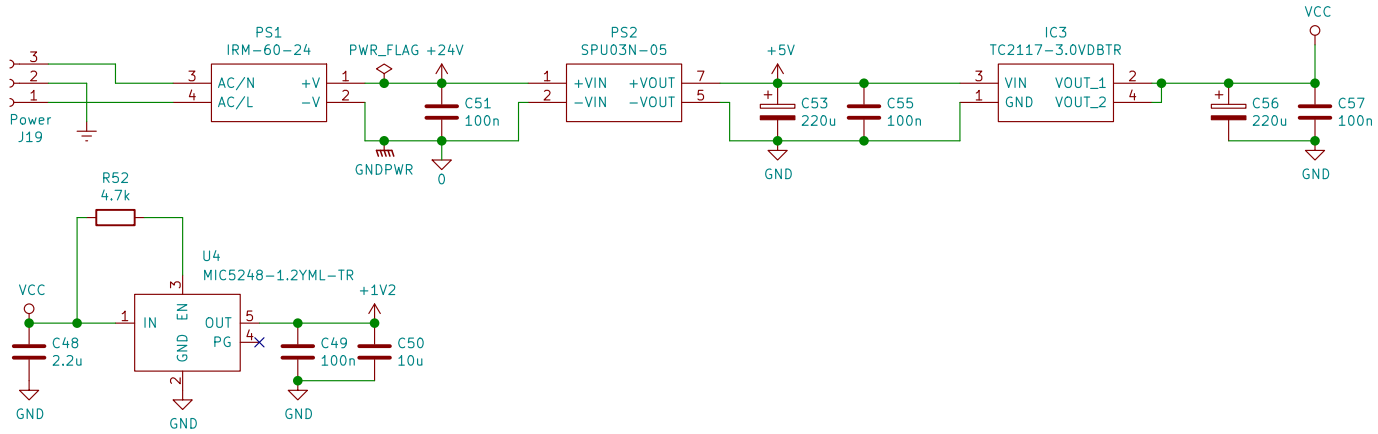

| | | |
|---|---|---|
| Sheet: /Prozessor/Abblock_C/ | | |
| File: Abblock_C.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 3/8 |


+24V
J20
Phoenix
CANH
CANL
BREAK
0
+24V
J3
Phoenix
CANH
CANL
BREAK
BREAK
0
Ethernet
HBICTRL[0..8]
HBICTRL4
HBICTRL3
HBICTRL1
HBICTRL0
HBICTRL2
HBICTRL6
HBICTRL7
HBICTRL8
HBICTRL5
J9
ETHER_CONTR
HBI_AD[0..15]
J6
HBI
HBI_AD0
HBI_AD1
HBI_AD2
HBI_AD3
HBI_AD4
HBI_AD5
HBI_AD6
HBI_AD7
HBI_AD8
HBI_AD9
HBI_AD10
HBI_AD11
HBI_AD12
HBI_AD13
HBI_AD14
HBI_AD15
+24V
J7
POWER_KOMMU
0
WLAN
SD[0..6]
SD0
SD1
SD2
SD3
SD6
SD5
SD4
J10
WLAN
USB[0..1]
+5V
R9
0
D
D4
J4
USB_A_vertikal
VBUS
D+
D−
GND
Shield
Shield
USB0
USB1
J5
USB_B_Micro
VBUS
D+
D−
ID
GND
Shield
+5V
C21
220u
GND
USB
UART – Nextion
VCC
R12
100k
R13
100k
+5V
Nextion[0..1]
Nextion1
Nextion0
rt
bl
ge
sw
J11
Nextion_Connector
GND
CAN[0..1]
CAN0
CAN1
VCC
R7
100k
R8
100k
+5V
C20
100n
IC2
MCP2542WFD-E_SN
TXD
VSS
VDD
RXD
STBY
CANH
CANL
VIO
VCC
C22
100n
R10
1.1M
R11
120
CANH
CANL
CANHIGH
CANLOW
GND
CANH
CANL
J8
CAN_Conn_ATSAM
CAN
Sheet: /Kommunikation/
File: Kommunikation.sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad (5.1.5)-3
Id: 4/8

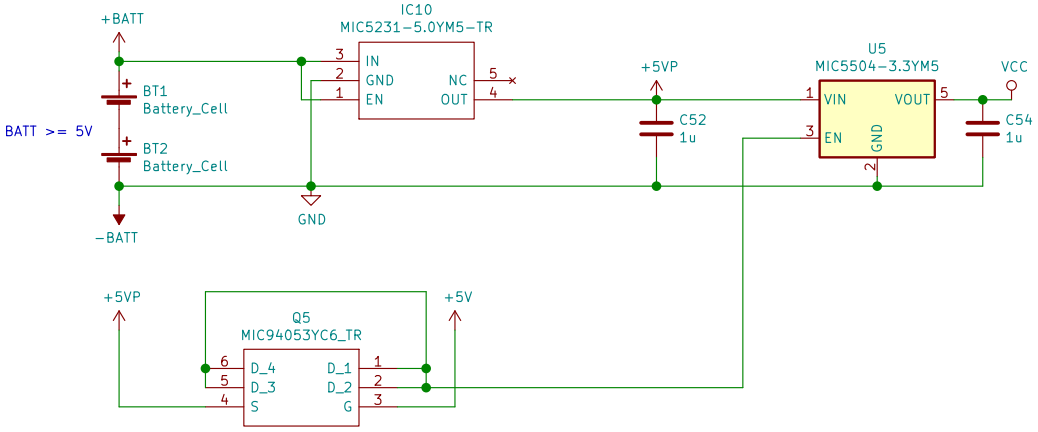



| | | |
|---|---|---|
| Sheet: /EEPROM/ | | |
| File: EEPROM.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 5/8 |

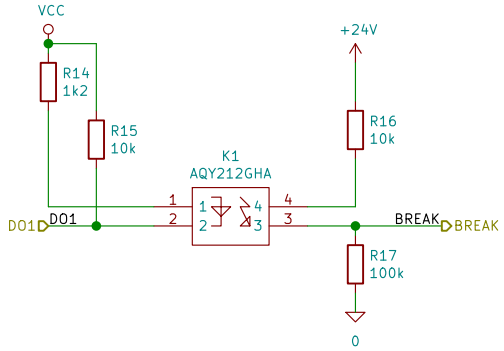
## Netzbetrieb



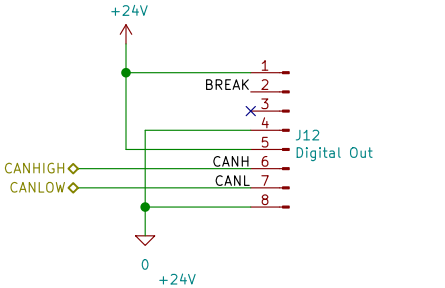

## Batterie

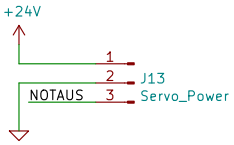


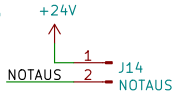

| | | |
|---|---|---|
| Sheet: /Power/<br>File: Power.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 6/8 |

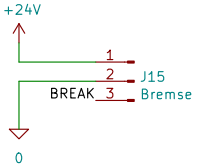



Belegung
DCOM X7
DCOM X8
EN
müssen für Servo geschalten werden



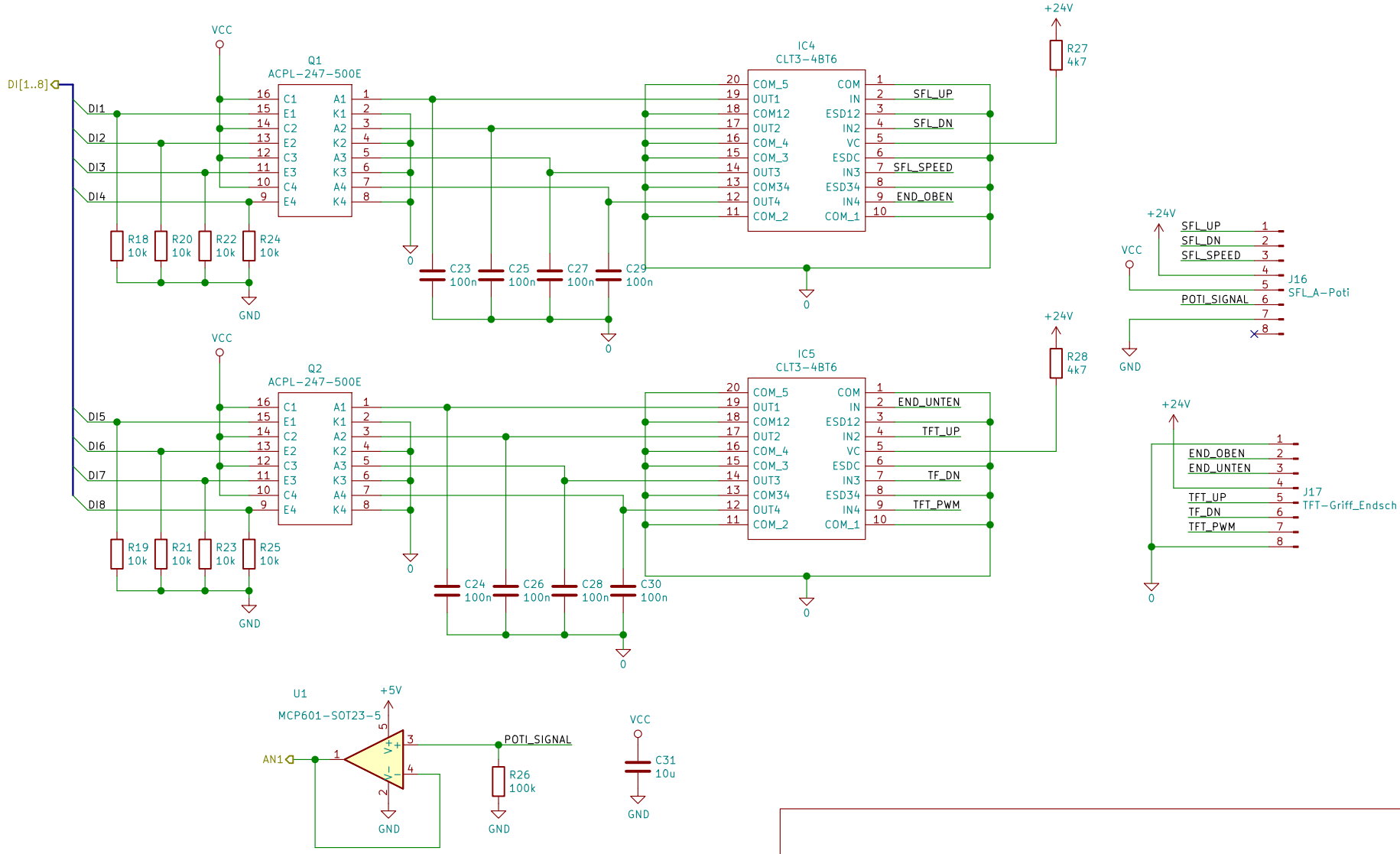



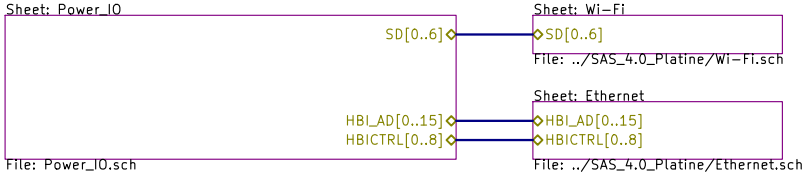

0

| | | |
|---|---|---|
| Sheet: /sheet5F99B59A/ | | |
| File: DOUT_ANOUT.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 7/8 |

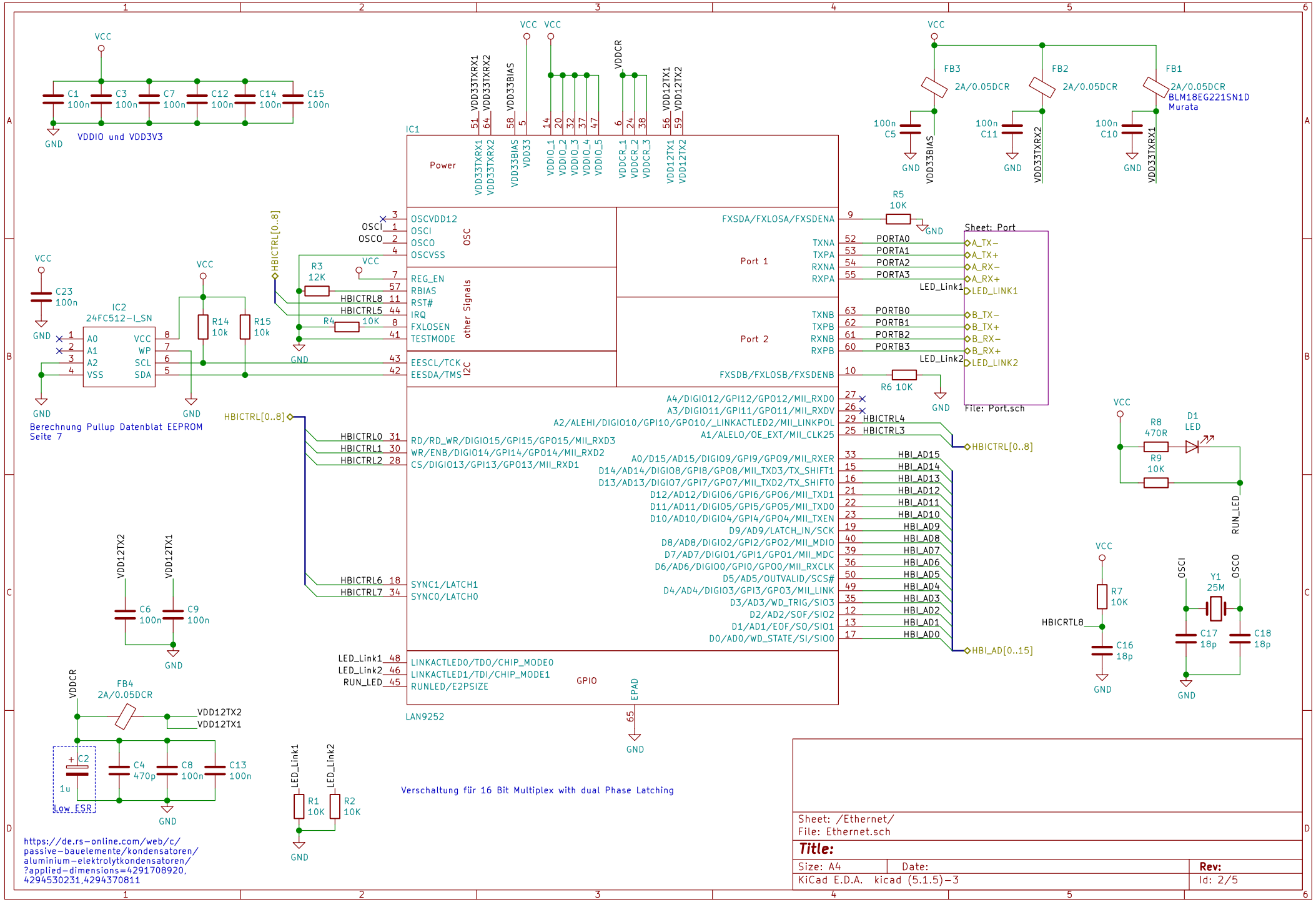



| | |
|---|---|
| Sheet: /sheet5F99B59B/ | |
| File: TFT.sch | |
| **Title:** | |
| Size: A4 | Date: |
| KiCad E.D.A. kicad (5.1.5)-3 | Id: 8/8 |

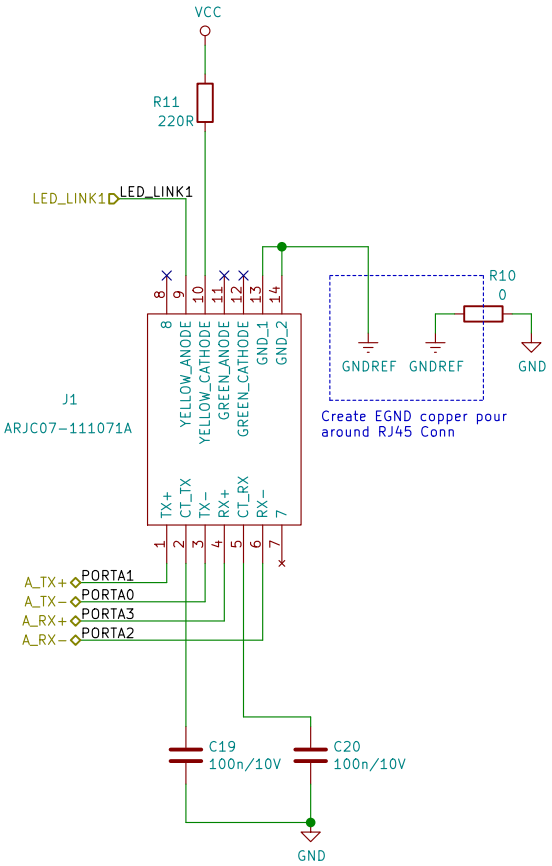
# Anhang B: Schaltplan Kommunikationsplatine



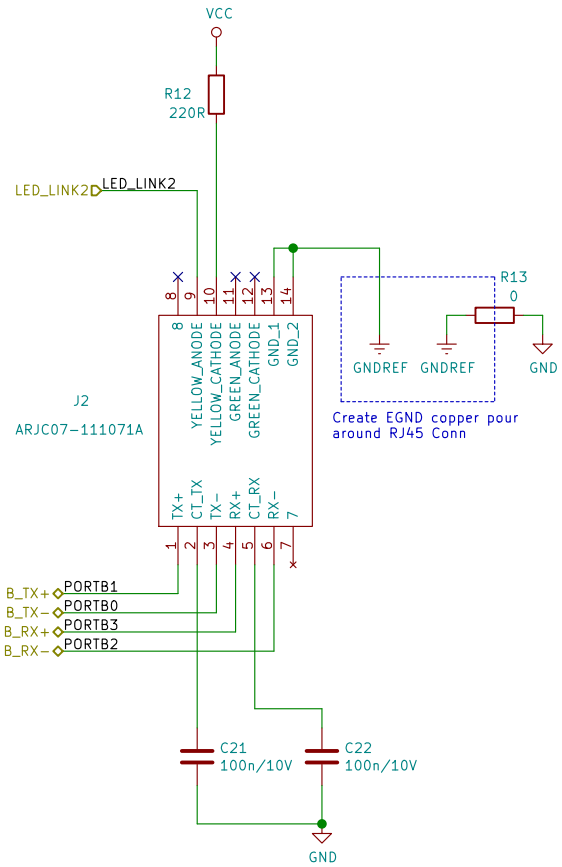

| | | |
|---|---|---|
| Sheet: /<br>File: SAS_Kommunikationsplatine.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 1/5 |


VDDIO und VDD3V3
C1 100n
C3 100n
C7 100n
C12 100n
C14 100n
C15 100n
GND
VCC
IC1
LAN9252
Power
OSC
Port 1
Port 2
other Signals
I2C
GPIO
EPAD
FB3 2A/0.05DCR
FB2 2A/0.05DCR
FB1 2A/0.05DCR
BLM18EG221SN1D
Murata
C5 100n
C11 100n
C10 100n
VDD33BIAS
VDD33TXRX2
VDD33TXRX1
R5 10K
R6 10K
Sheet: Port
File: Port.sch
PORTA0
PORTA1
PORTA2
PORTA3
PORTB0
PORTB1
PORTB2
PORTB3
LED_Link1
LED_Link2
IC2
24FC512-I_SN
C23 100n
R14 10k
R15 10k
R3 12K
R4 10K
HBICTRL[0..8]
HBICTRL8
HBICTRL5
Berechnung Pullup Datenblat EEPROM
Seite 7
R8 470R
R9 10K
D1 LED
RUN_LED
R7 10K
HBICRTL8
C16 18p
Y1 25M
C17 18p
C18 18p
OSCI
OSCO
C6 100n
C9 100n
VDD12TX2
VDD12TX1
FB4 2A/0.05DCR
C2 1u
Low ESR
C4 470p
C8 100n
C13 100n
R1 10K
R2 10K
HBI_AD[0..15]
Verschaltung für 16 Bit Multiplex with dual Phase Latching
https://de.rs-online.com/web/c/passive-bauelemente/kondensatoren/aluminium-elektrolytkondensatoren/?applied-dimensions=4291708920,4294530231,4294370811
Sheet: /Ethernet/
File: Ethernet.sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad (5.1.5)-3
Id: 2/5

Anschluss nach KSZ8061 Daughter Board Sheet



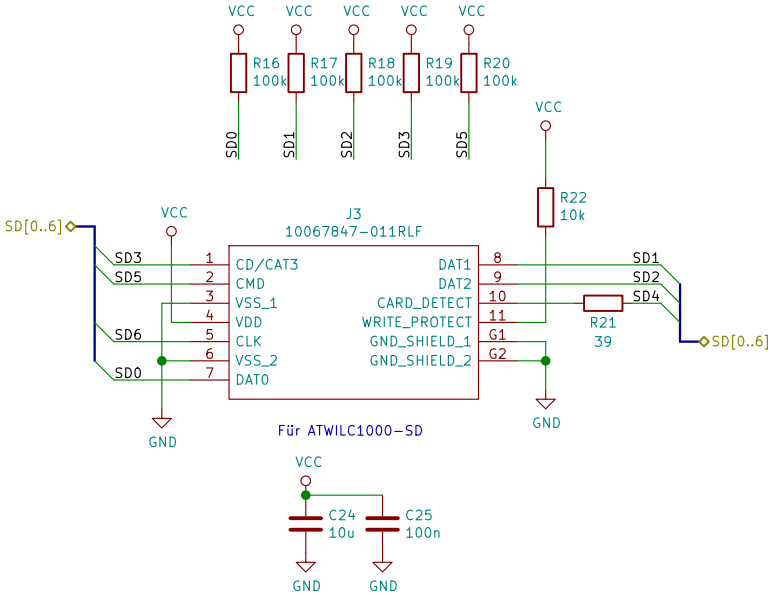

Anschluss nach KSZ8061 Daughter Board Sheet



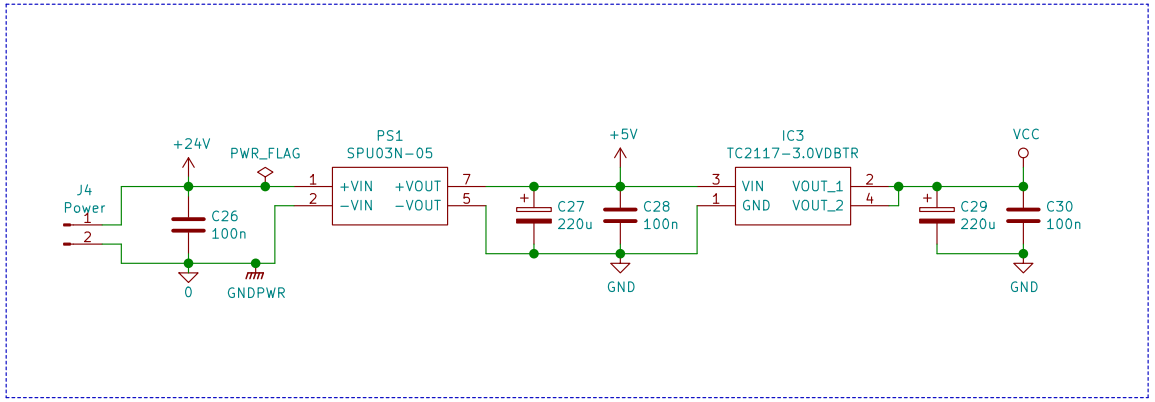

Sheet: /Ethernet/Port/
File: Port.sch

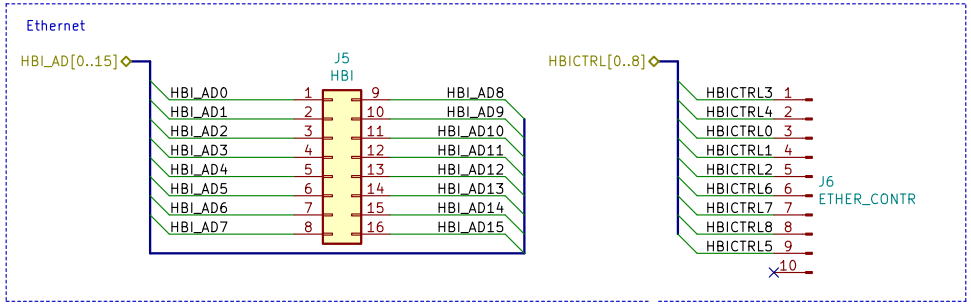
**Title:**

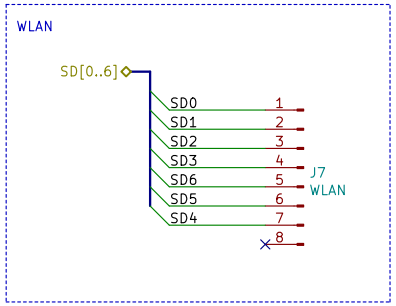
| Size: A4 | Date: | **Rev:** |
|---|---|---|
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 3/5 |




| | | |
|---|---|---|
| Sheet: /Wi-Fi/ | | |
| File: Wi-Fi.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 4/5 |

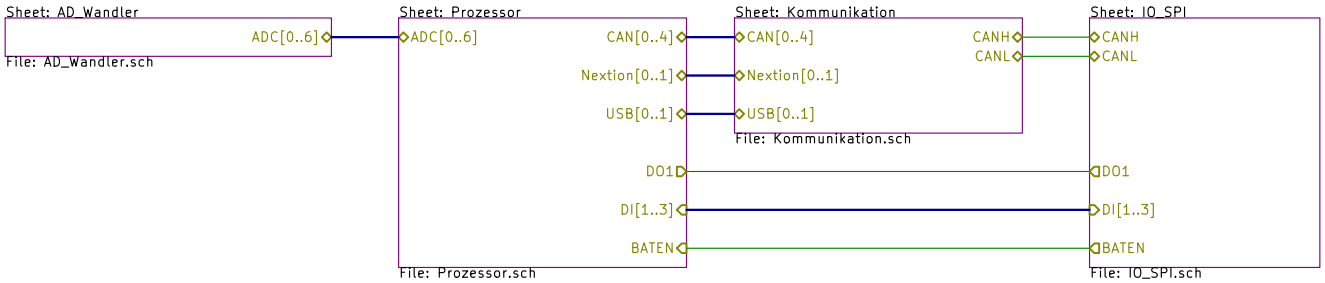



| | | |
|---|---|---|
| Sheet: /Power_IO/<br>File: Power_IO.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 5/5 |

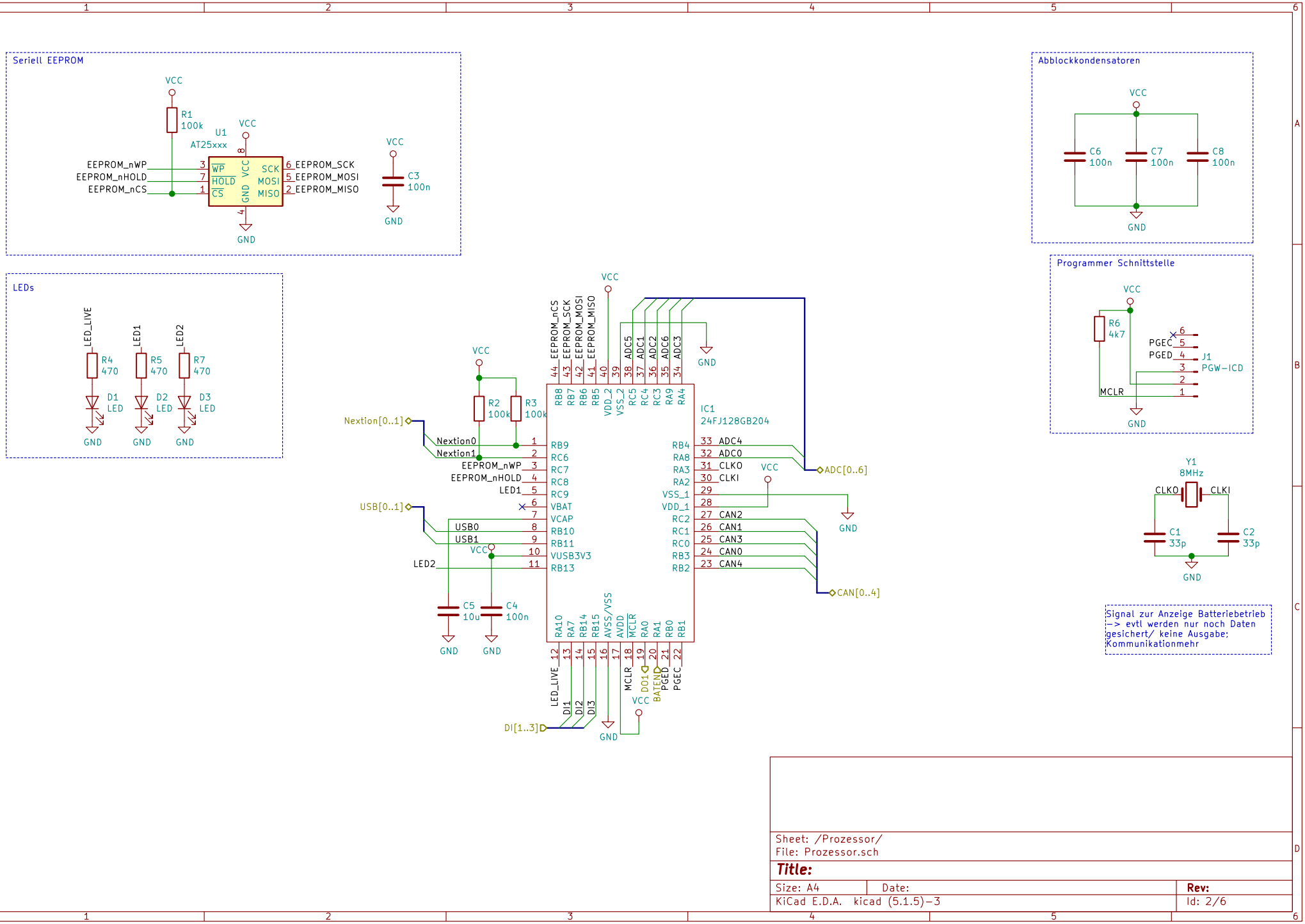
# Anhang C: Schaltplan Lastkollektivspeicher

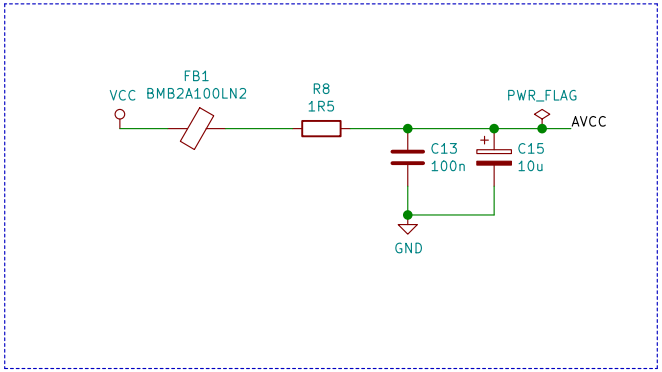


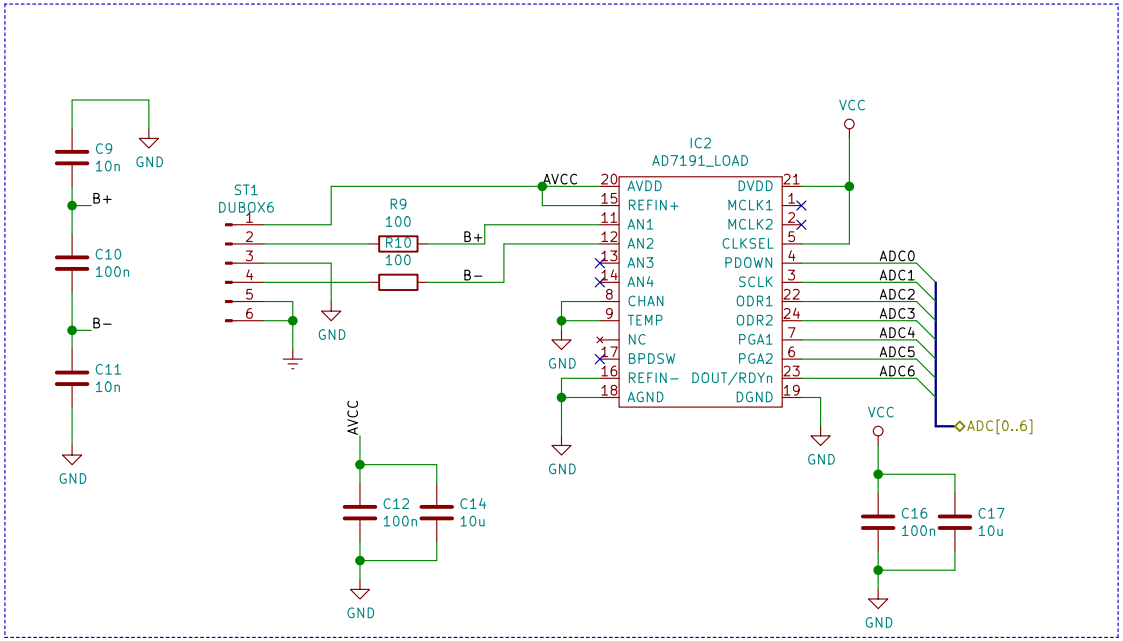

Sheet: /
File: Lastkollektivspeicher.sch

**Title:**

| Size: A4 | Date: | **Rev:** |
|---|---|---|
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 1/6 |


Seriell EEPROM
VCC
R1
100k
U1
AT25xxx
EEPROM_nWP
EEPROM_nHOLD
EEPROM_nCS
WP
HOLD
CS
SCK
MOSI
MISO
EEPROM_SCK
EEPROM_MOSI
EEPROM_MISO
GND
C3
100n
LEDs
LED_LIVE
LED1
LED2
R4
470
R5
470
R7
470
D1
LED
D2
LED
D3
LED
Abblockkondensatoren
C6
100n
C7
100n
C8
100n
Programmer Schnittstelle
R6
4k7
PGEC
PGED
MCLR
J1
PGW-ICD
Y1
8MHz
CLKO
CLKI
C1
33p
C2
33p
IC1
24FJ128GB204
Nextion[0..1]
Nextion0
Nextion1
R2
100k
R3
100k
USB[0..1]
USB0
USB1
LED2
C5
10u
C4
100n
ADC[0..6]
CAN[0..4]
DI[1..3]
Signal zur Anzeige Batteriebetrieb
-> evtl werden nur noch Daten
gesichert/ keine Ausgabe;
Kommunikationmehr
Sheet: /Prozessor/
File: Prozessor.sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad (5.1.5)-3
Id: 2/6



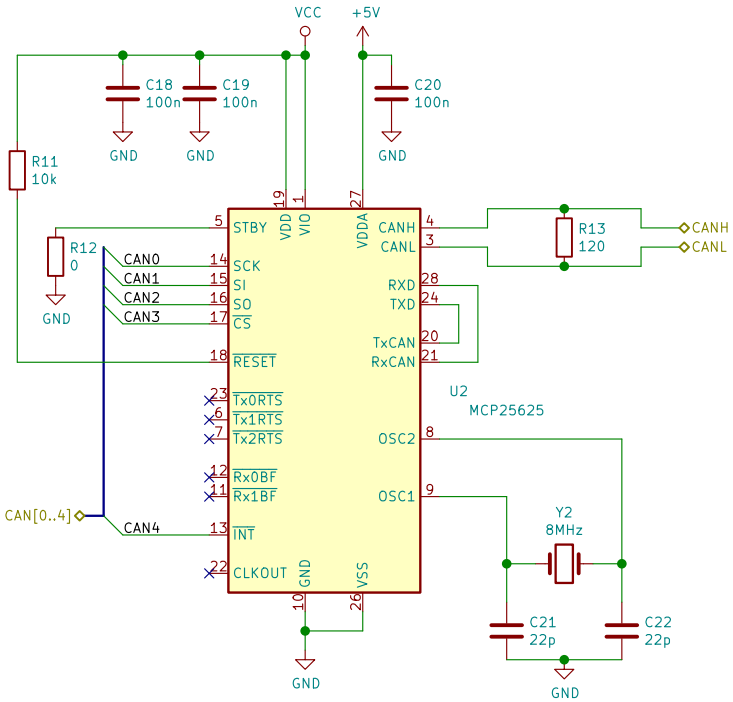




Sheet: /AD_Wandler/
File: AD_Wandler.sch

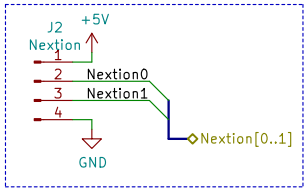
**Title:**

| Size: A4 | Date: | **Rev:** |
|---|---|---|
| KiCad E.D.A. kicad (5.1.5)−3 | | Id: 3/6 |

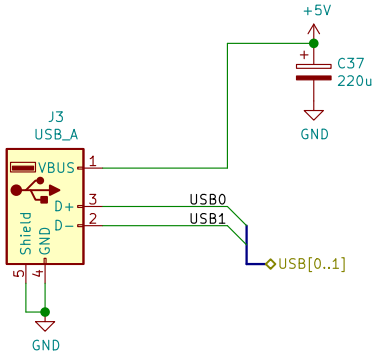
## CAN



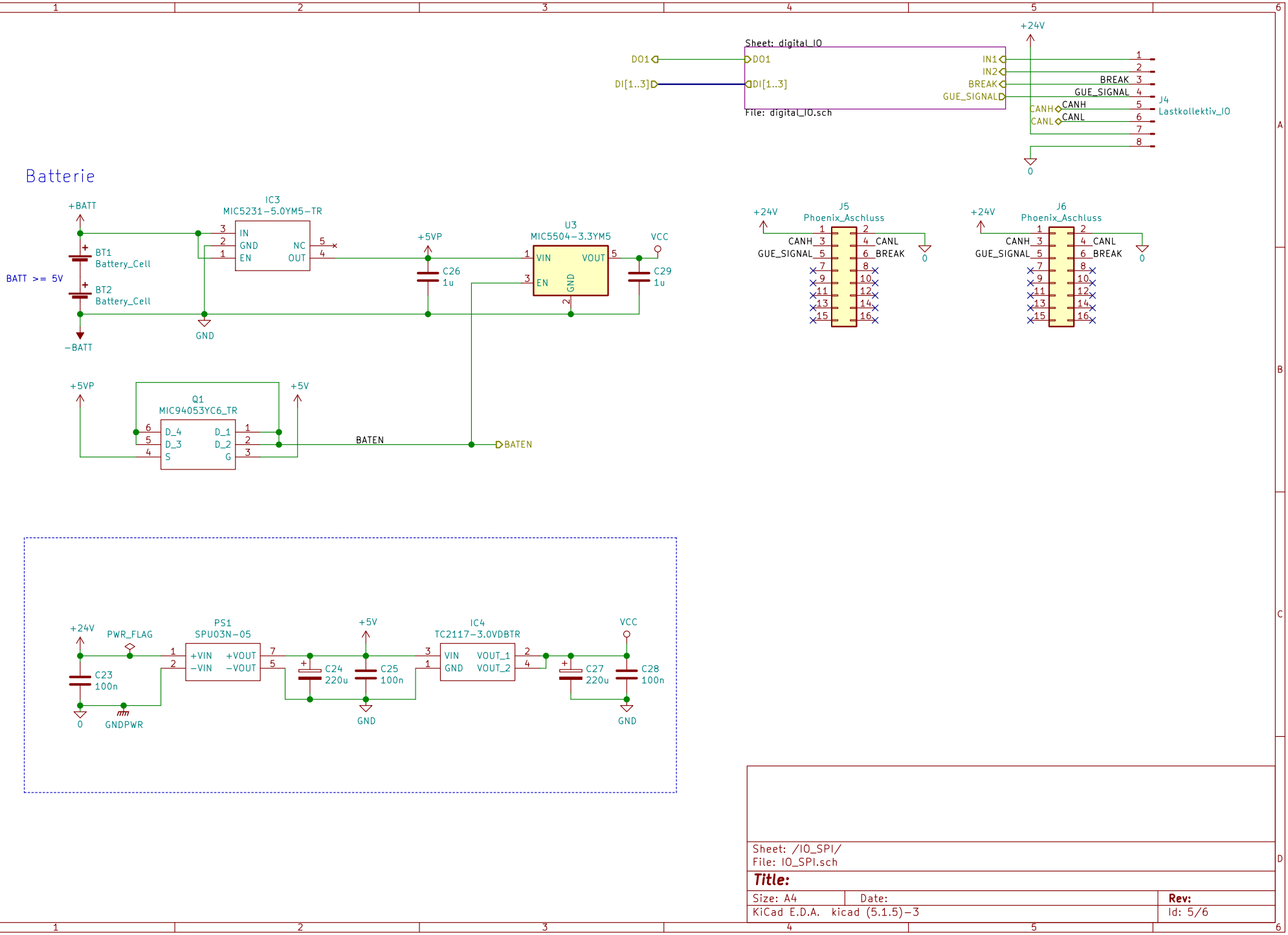

## UART




## USB

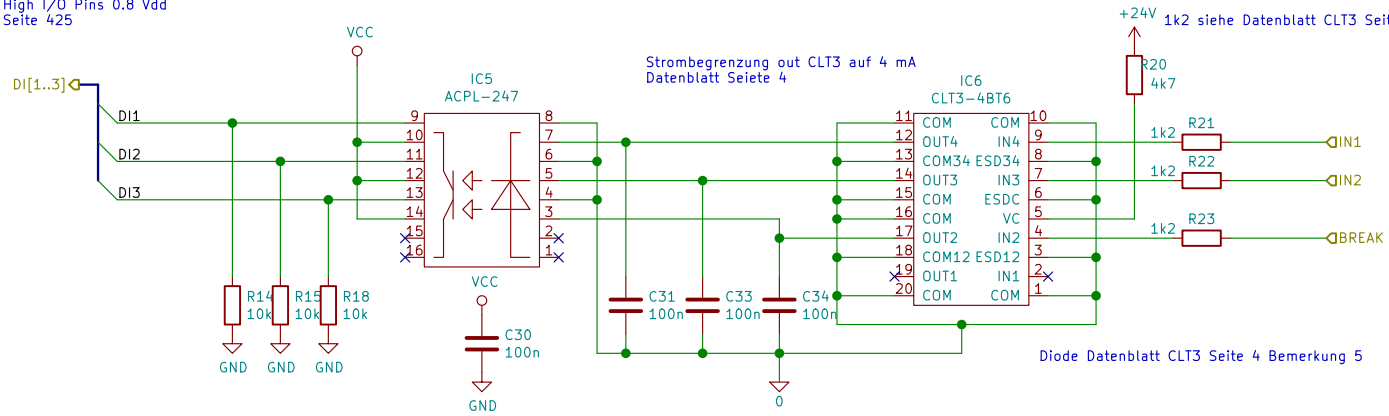


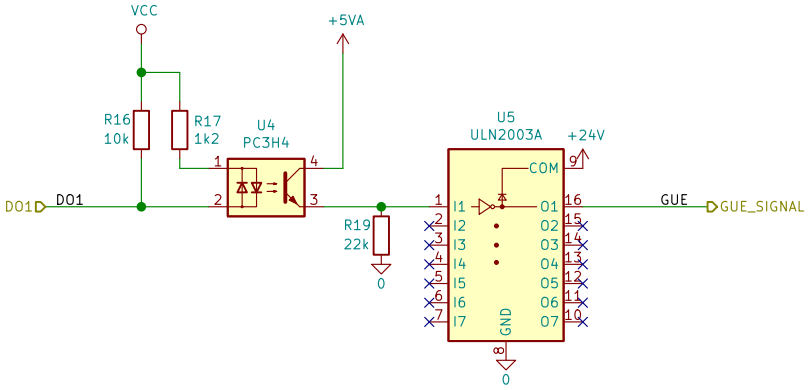

| | |
|---|---|
| Sheet: /Kommunikation/ | |
| File: Kommunikation.sch | |
| **Title:** | |
| Size: A4 | Date: |
| KiCad E.D.A. kicad (5.1.5)-3 | Id: 4/6 |

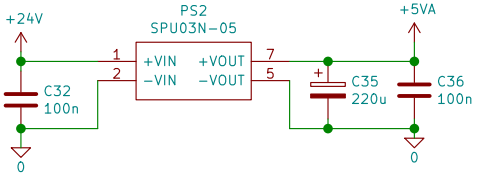



| | | |
|---|---|---|
| Sheet: /IO_SPI/ | | |
| File: IO_SPI.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.5)−3 | | Id: 5/6 |

Eingangsstrom muss auf Grund der Begrenzung im ACPL nicht über R ins µC
max Strom 7mA
Strom ins µC ca. 1mA Datenblatt ACPL Seite Seite 7 Figure 8
High I/O Pins 0.8 Vdd
Seite 425

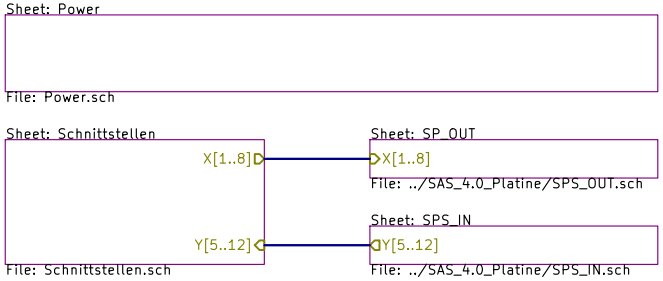


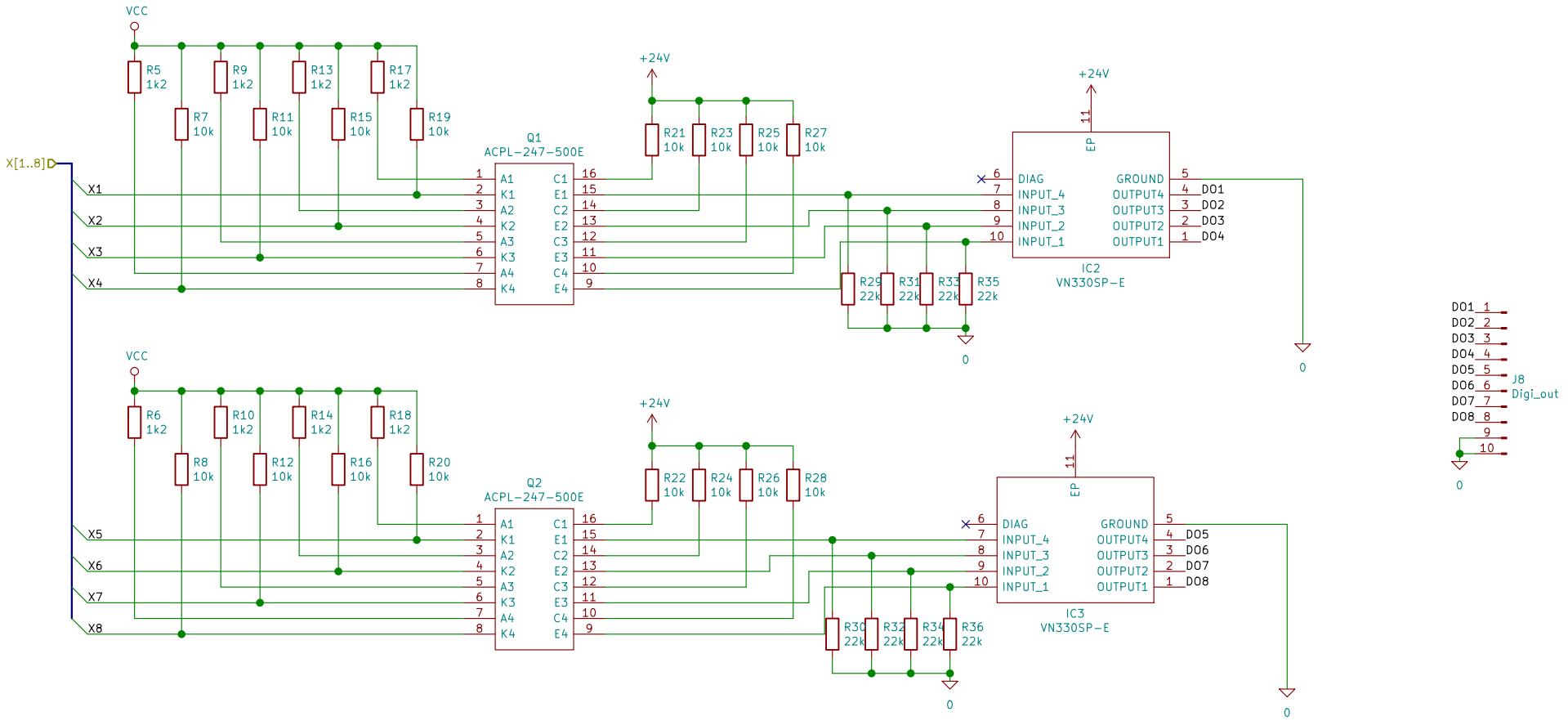




| | | |
|---|---|---|
| Sheet: /IO_SPI/digital_IO/ | | |
| File: digital_IO.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 6/6 |

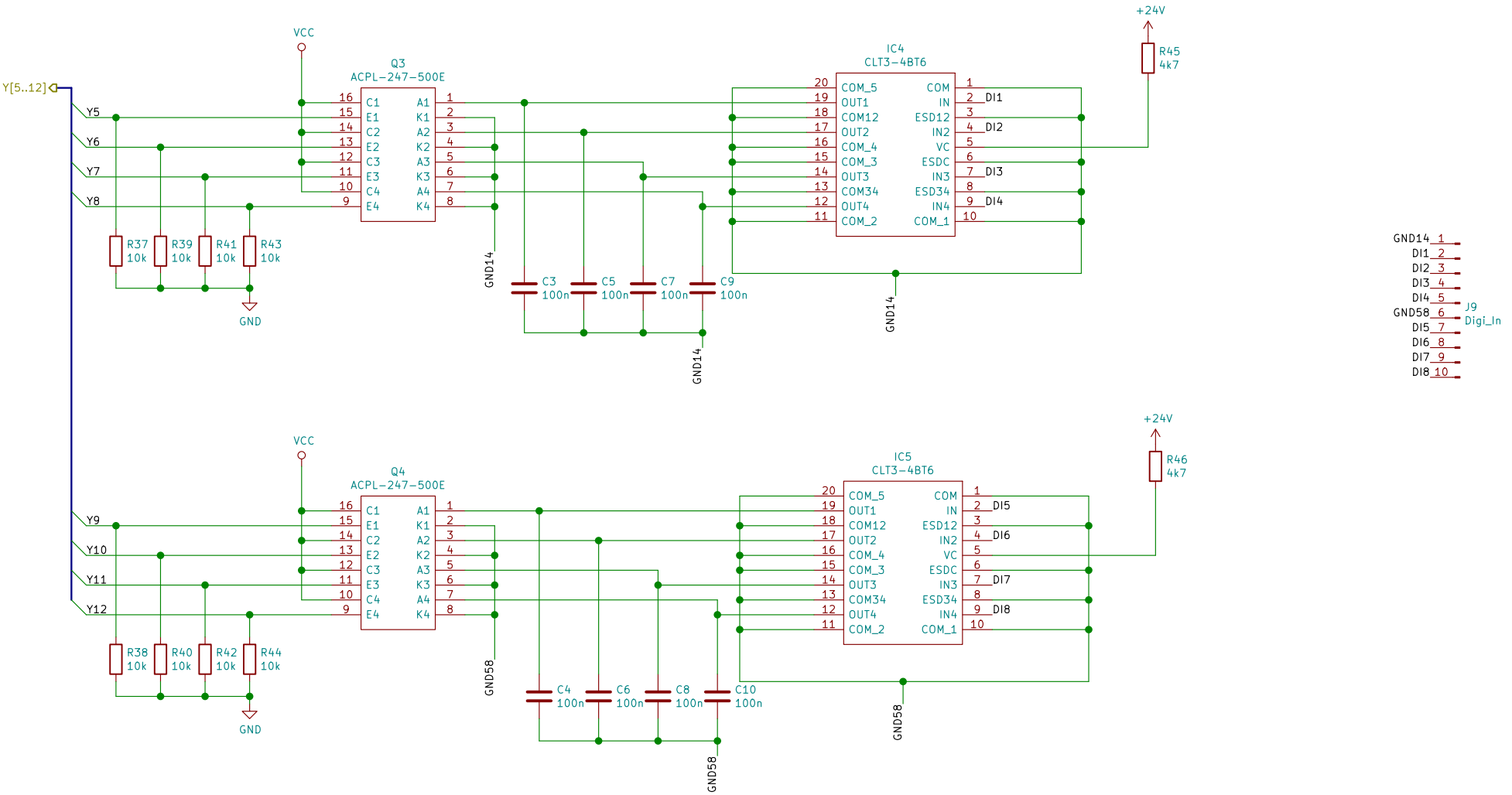
# Anhang D: Schaltplan *pyboard*



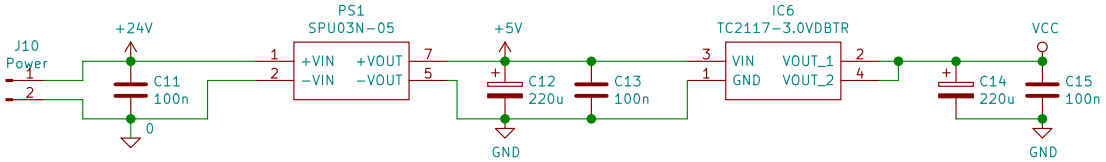

| | | |
|---|---|---|
| Sheet: /<br>File: SAS_SPS_pyboard.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 1/5 |



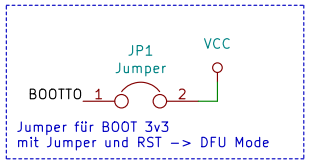

| | | |
|---|---|---|
| Sheet: /SP_OUT/ | | |
| File: SPS_OUT.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 2/5 |

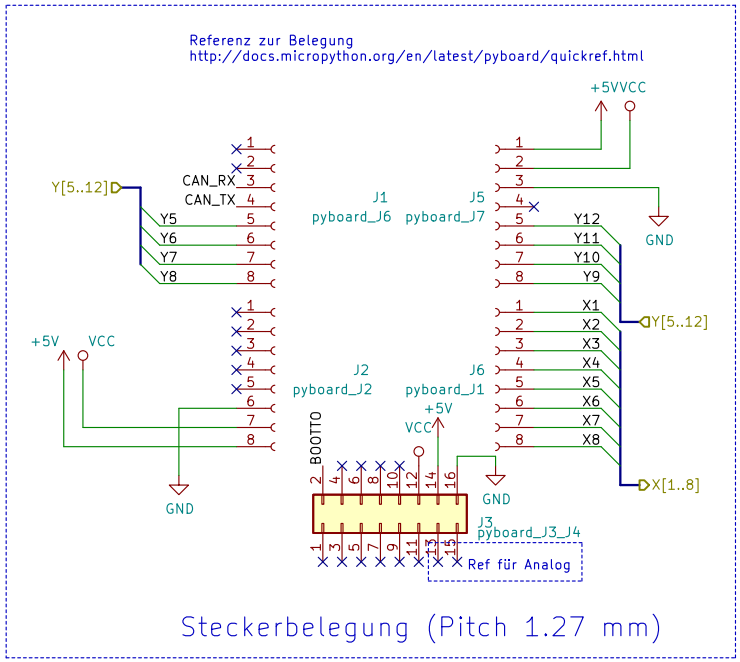


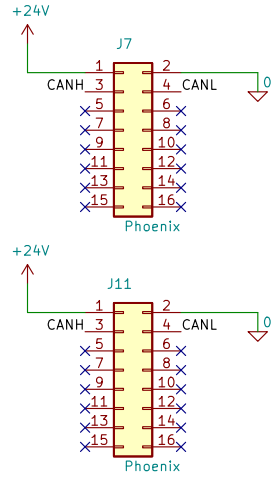

| | | |
|---|---|---|
| Sheet: /SPS_IN/ | | |
| File: SPS_IN.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 3/5 |

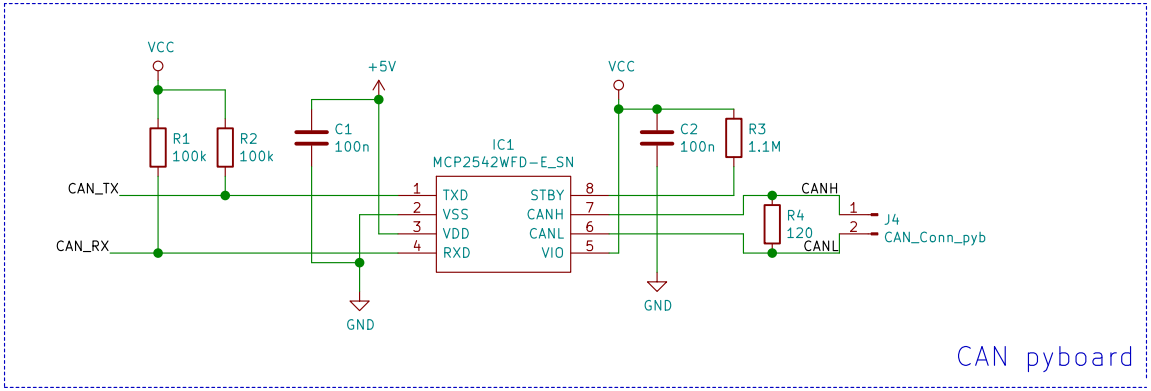



| | | |
|---|---|---|
| Sheet: /Power/<br>File: Power.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 4/5 |



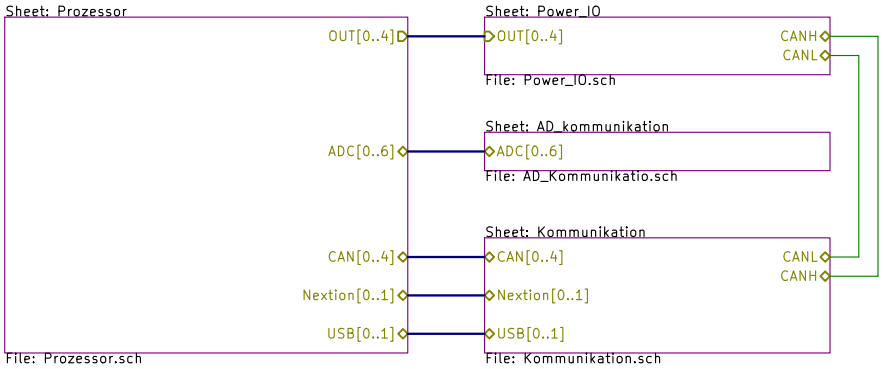




| | | |
|---|---|---|
| Sheet: /Schnittstellen/<br>File: Schnittstellen.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 5/5 |

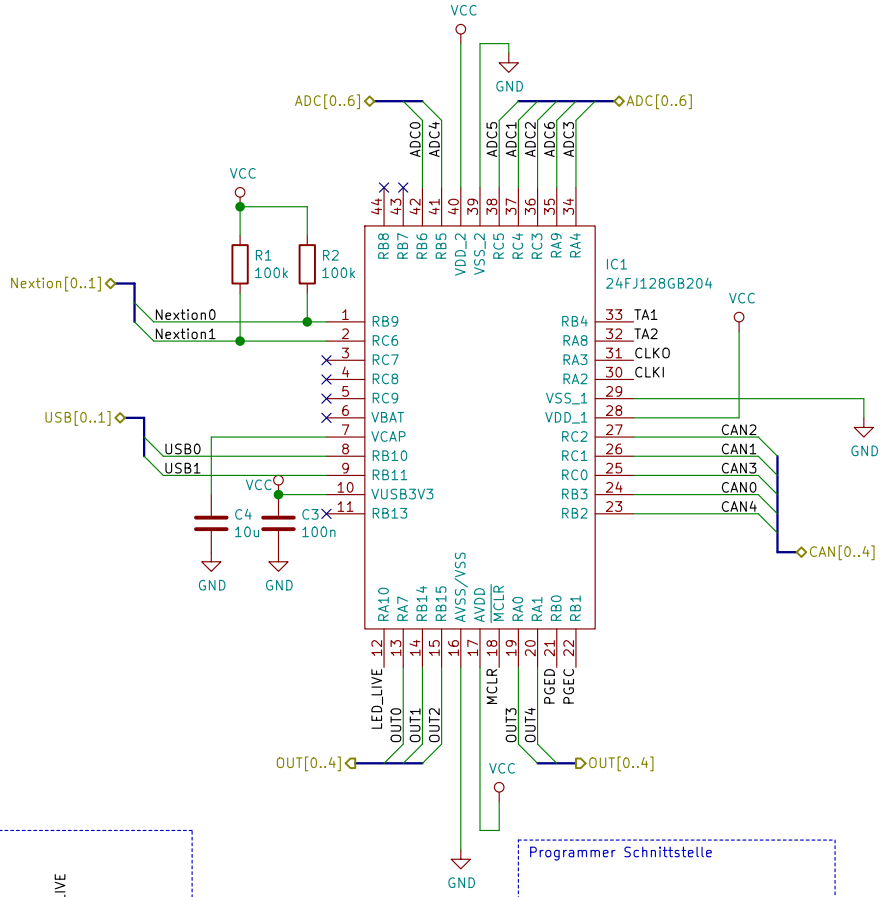
# Anhang E: Schaltplan *Balancer* Elektronik

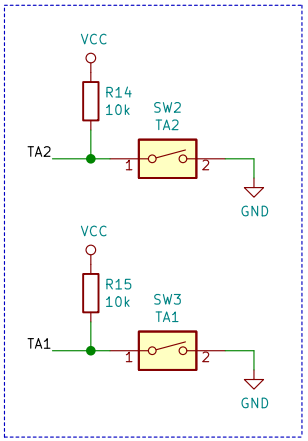


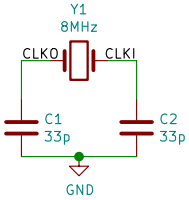

Sheet: /
File: 10.60.70.80_Nextion_Controllboard.sch

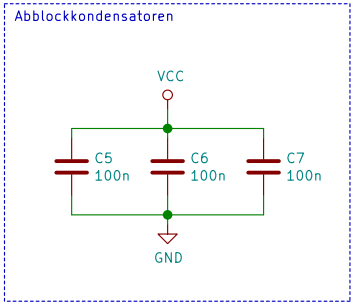
**Title:**

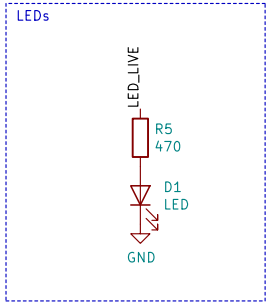
| Size: A4 | Date: | **Rev:** |
|---|---|---|
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 1/5 |

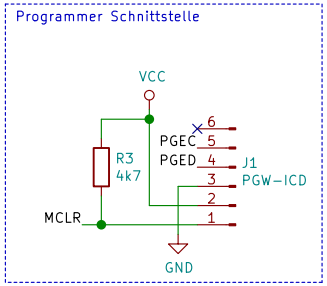


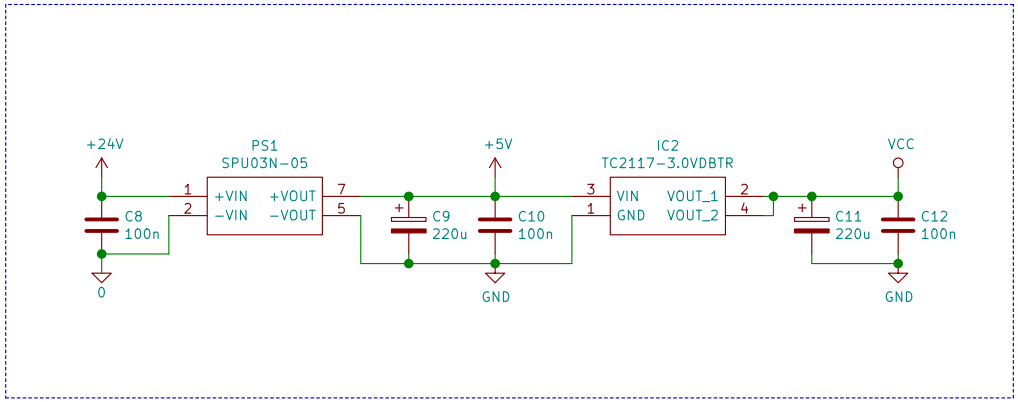



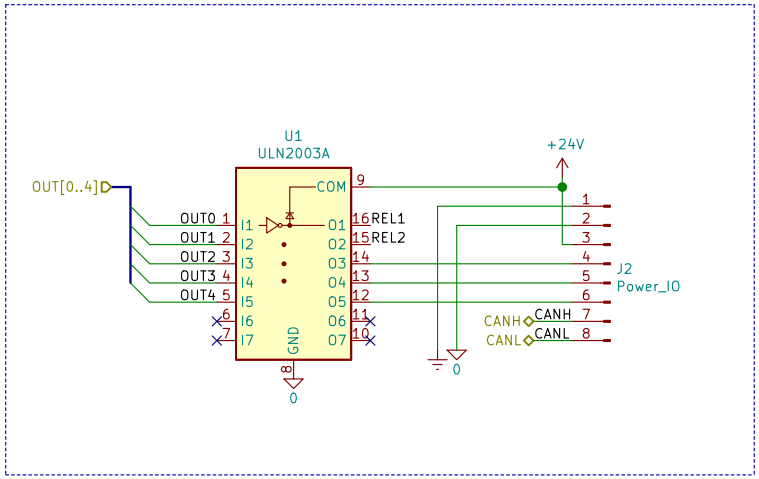



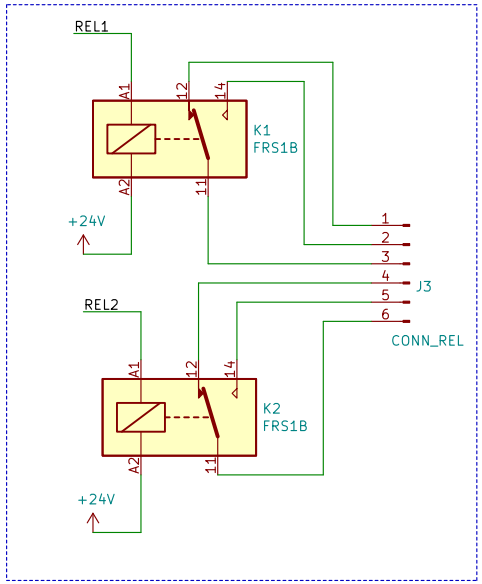



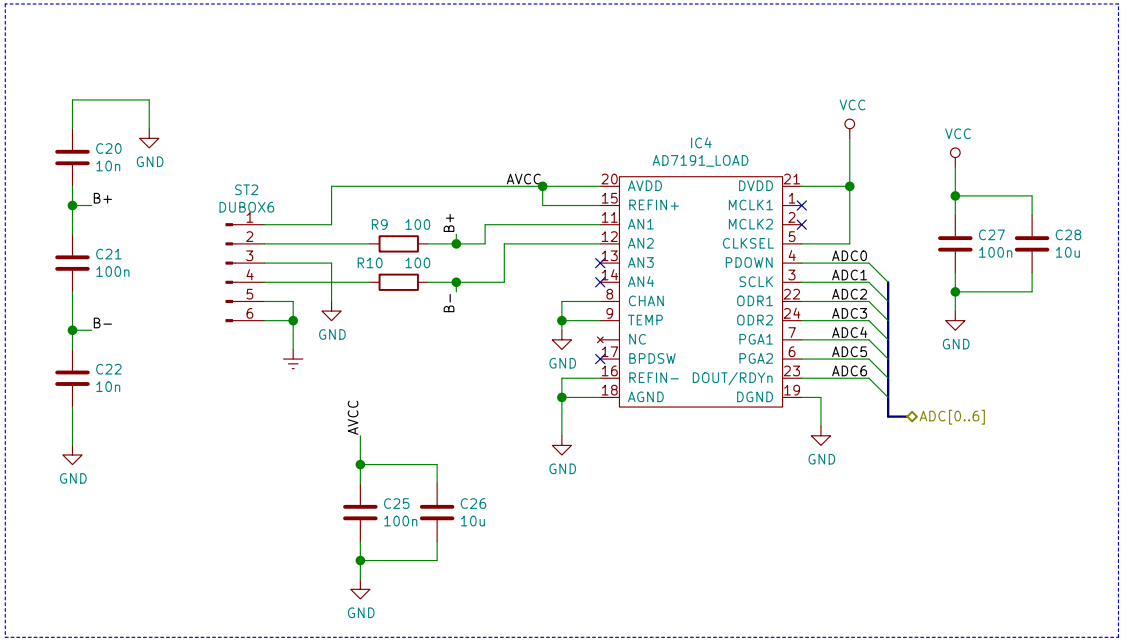

| | | |
|---|---|---|
| Sheet: /Prozessor/<br>File: Prozessor.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 2/5 |

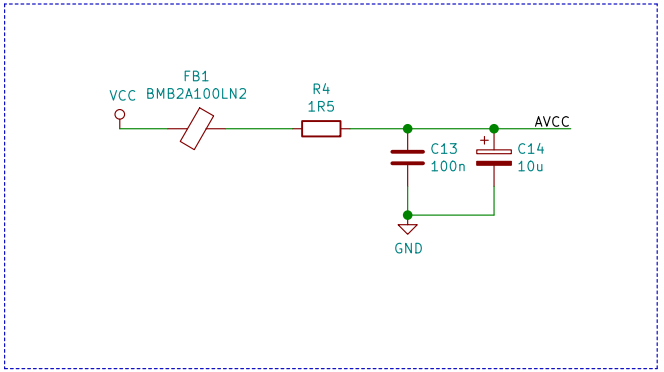


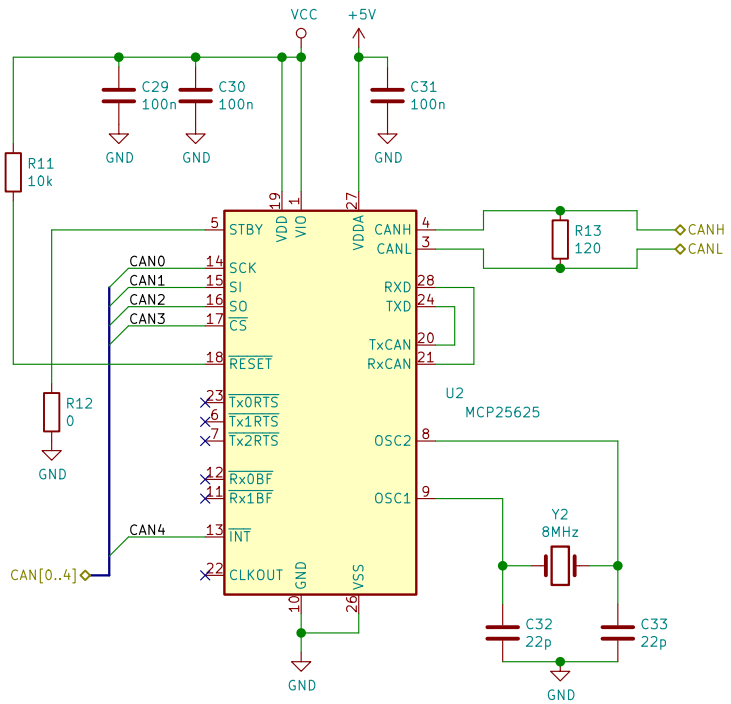

| | | |
|---|---|---|
| Sheet: /Power_IO/<br>File: Power_IO.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.5)-3 | | Id: 3/5 |



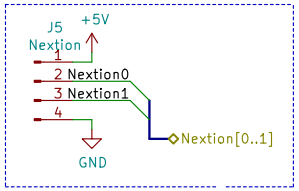




| | | |
|---|---|---|
| Sheet: /AD_kommunikation/<br>File: AD_Kommunikatio.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.5)−3 | | Id: 4/5 |

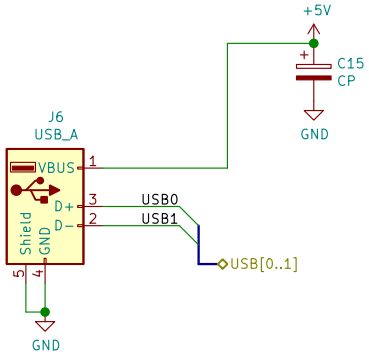
## CAN




## UART




## USB




| | |
|---|---|
| Sheet: /Kommunikation/ | |
| File: Kommunikation.sch | |
| **Title:** | |
| Size: A4 | Date: |
| KiCad E.D.A. kicad (5.1.5)-3 | Id: 5/5 |

# Anhang F: SPI-Funktionen

```
/*********************************************

*

*                Spi protocol Function

*

*********************************************/

# define CMD_DMA_WRITE                  0xc1
# define CMD_DMA_READ                   0xc2
# define CMD_INTERNAL_WRITE             0xc3
# define CMD_INTERNAL_READ              0xc4
# define CMD_TERMINATE                  0xc5
# define CMD_REPEAT                     0xc6
# define CMD_DMA_EXT_WRITE              0xc7
# define CMD_DMA_EXT_READ               0xc8
# define CMD_SINGLE_WRITE               0xc7
# define CMD_SINGLE_READ                0xca
# define CMD_RESET                      0xcf

# define N_OK                           1
# define N_FAIL                         0
# define N_RESET                        -1
# define N_RETRY                        -2

# define SPI_RESP_RETRY_COUNT           (10)
# define SPI_RETRY_COUNT                (10)
# define DATA_PKT_SZ_256                256
# define DATA_PKT_SZ_512                512
# define DATA_PKT_SZ_1K                 1024
# define DATA_PKT_SZ_2K                 (2 * 1024)
# define DATA_PKT_SZ_4K                 (4 * 1024)
# define DATA_PKT_SZ_8K                 (8 * 1024)
# define DATA_PKT_SZ                    DATA_PKT_SZ_8K

# define USE_SPI_DMA                    0
```

# Literaturverzeichnis

[1] Tim Sieber. Nachbildung von funktionsabläufen zur steuerung von hubmodulen mittels verschiedener bedienelemente in software und integration eines nextion-displays zur zustandsanzeige und parametrierung des moduls. Beleg zum Forschungsmodul Wintersemester 2019/2020.

[2] OA Redaktion. Hms: Marktanteile industrieller netzwerke 2019. *open automation*, 2019.

[3] Christian Keydel Olaf Pfeiffer, Andrew Ayre. *Embedded Networking with CAN and CANopen*. Embedded Systems Academy Inc., 2016.

[4] Jörg Rech. *Ethernet : Technologien und Protokolle für die Computervernetzung*. Heise Zeitschriften Verlag GmbH & Co. KG, Hannove, 2014.

[5] Prof. Dr.-Ing. Heinz Frank. *Einführung in Ethernet*. Reinhold-Würth-Hochschule der Hochschule Heilbronn in Künzelsau, April 2010.

[6] Matthew S. Gast. *802.11 Wireless Networks - The Definitive Guide*. O'Reilly Media Inc., 2005.

[7] Jens Bathelt. *Entwicklungsmethodik für SPS-gesteuerte mechatronische Systeme*. PhD thesis, Eidgenössische Technische Hochschule Zürich, 2006.

[8] J. Labrosse. Hardware-accelerated rtos. Technical report, Micrium Embedded Software, July 2014.

[9] FEDERATION EUROPEENNE DE LA MANUTENTION. *Einstufung der Triebwerke*. FEDERATION EUROPEENNE DE LA MANUTENTION, 1986.

[10] FEDERATION EUROPEENNE DE LA MANUTENTION. *Maßnahmen zum Erreichen sicherer Betriebsperioden von motorisch angetriebenen Serienhubwerken (S.W.P.)*. FEDERATION EUROPEENNE DE LA MANUTENTION, 1993.

[11] EtherCAT Group. Ethercat - der ethernet feldbus. Technical report, Beckhoff Automation GmbH, 2018.

[12] Microchip. Lan9252 - 2/3-port ethercat slave controller with integrated erthernet phys. Technical report, Microchip Technology Incorporated, 2015.

[13] Microchip Technology Inc. Atwilc1000-sd user guide. Technical report, Microchip Technology Inc., 2017.

[14] Stefan Moehringer Jürgen Gausemeier. New guideline vdi 2206 - a flexible procedure model for the design of mechatronic systems. In *INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN ICED 03 STOCKHOLM*, 2003.

[15] Olaf Hagenbruch Thomas Beierlein. *Taschenbuch Mikroprozessortechnik*. Fachbuchverlag Leipzig im Carl Hanser Verlag, 2004.

[16] Martin Scheffler, Klaus Feyrer, and Karl Matthias. *Fördermaschinen*. Deutscher Universitätsvlg, 2014.

[17] Deutsche Gesetliche Unfallversicherung. *Winden, Hub- und Zuggeräte*. Deutsche Gesetliche Unfallversicherung, 1996.

[18] europäisches Parlament und Rat. *Maschinenrichtline 2006/42/EG*. europäisches Parlament und Rat, 2006.

[19] Armin Roth. *Einführung und Umsetzung von Industrie 4.0*. Springer-Verlag GmbH, 2016.

[20] Prof. Dr. Wolfgang Bock. *Grundlagen der SPS-Programmierung / Prozessinformatik*. OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG FAKULTÄT MASCHINENBAU, January 2018. Skriptum zum weiterbildenden Kurs im Rahmen des Projekts OTHmind.

[21] DEMAG. Der modulare demag seilzug dmr, 2020.

[22] Christoph Kreutzenbeck. Demag sicherheitssteuerung für verfahrwagen und krane. Technical report, DEMAG GmbH, 2018.

[23] INDEVA GROUP. Indeva gateway, 2020.

[24] LIFTKET. Produktvorstellung. Technical report, liftket.de, 2020.

[25] Gorbel Inc. Intelligent lifting. Technical report, Gorbel, 2020.

[26] Klaus Marchewka. Besprechung zur anforderung an künftige hebezeuge. Technical report, Sauer Automation Sachsen, 2020.

[27] Bernhard Klotz. *Übertragungstechnik Ethernet*. Duale Hochschule Baden-Würtenberg, 2016.

[28] I.N. Bronstein; K.A. Semendjajew; G. Musiol; H. Mühlig. *Taschenbuch der Mathematik*. Verlag Europa Lehrmittel, 2013.

[29] Microchip Technology Inc. At25320b/at25640b. Technical report, Microchip Technology Inc., 2018.

[30] Microchip Technology Incorporated. Sam e70/s70/v70/v71 family. Technical report, Microchip Technology Incorporated, 2017.

[31] Microchip. Evb-lan9252-hbi+quick start guide. Technical report, Microchip Technology Inc., 2015.

[32] Jonathan Corbet Alessandro Rubini. *LINUX Gerätetreiber*. OReilly, 2002.

[33] Microchip Technology Inc. Atwilc1000 rtos driver porting guide. Technical report, Microchip Technology Inc., 2018.

[34] Jacob Beningo. Schnelle entwicklung von echtzeitanwendungen auf mikrocontroller-basis mithilfe von micropython. *Artikel-Bibiliothek Digi-Key*, September 2017.

# Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 7. August 2020