UNIVERSITÄT KARLSRUHE (TH) - FORSCHUNGSUNIVERSITÄT

FAKULTÄT FÜR ELEKTRO- UND INFORMATIONSTECHNIK

INSTITUT FÜR BIOMEDIZINISCHE TECHNIK

DIPLOMARBEIT

Konzeption und Entwicklung eines intelligenten, mobilen medizinischen Ad-hoc Netzwerks zur Überwachung von Personen bei Massenanfällen von Verletzten (MANV)

vorgelegt von	cand. el. Marco Bauer
Betreuer	Prof. Dr. rer. nat. Armin Bolz
	Dr.-Ing. Marc Jäger
Abgabetermin	08.06.2009

Eidesstattliche Erklärung

Hiermit erkläre ich eidesstattlich, dass ich die vorliegende Diplomarbeit selbständig und ohne unzulässige fremde Hilfsmittel angefertigt habe. Die verwendeten Literaturquellen sind im Literaturverzeichnis vollständig angegeben. Die Arbeit wurde in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde zur Erlangung eines akademischen Grades vorgelegt.

Karlsruhe, 08.06.2009

Danksagung

An dieser Stelle möchte ich den Personen danken, ohne die diese Arbeit so nicht entstanden und auch nicht möglich gewesen wäre.

Ein herzliches Dankeschön gilt Herrn Prof. Dr. rer. nat. Armin Bolz für die Betreuung meiner Arbeit. Außerdem möchte ich meinem Betreuer Dr.-Ing. Marc Jäger sowie M. Eng. Daniel Wettach für die zahlreichen nützlichen Diskussionen und Ratschläge, die sie mir mit auf den Weg gegeben haben, danken.

Ein besonderes Dankeschön geht auch an Herrn cand. el. Michael Weiß, der mir bei Softwareproblemen immer beratend zur Seite stand.

Zuletzt möchte ich mich noch ganz besonders bei meiner Freundin Katja Löschner, bei meiner Familie und all meinen Freunden bedanken, die mich in schwierigen Stunden beim Verfassen dieser Arbeit immer motiviert und unterstützt haben.

Inhaltsverzeichnis

1. Einleitung

1.1. Motivation

Bei Naturkatastrophen oder Terroranschlägen gibt es häufig sehr viele verletzte Personen auf engem Raum, die möglichst schnell und zielgerichtet medizinische Hilfe benötigen. Das behandelnde medizinische Personal ist in solchen Fällen, die im Allgemeinen als „Massenanfälle von Verletzten“ (MANV) bezeichnet werden, häufig überfordert von jedem einzelnen Patienten den Gesundheitszustand genau zu bestimmen und zu überwachen.

In dieser Situation ist es wünschenswert ein mobiles, vernetztes System zur Hand zu haben, das mittels nichtinvasiven Monitorings[1] den Gesundheitszustand jedes Patienten überwacht und bei kritischen Kreislaufzuständen (z.B. Herz- oder Atemstillstand) dem behandelnden medizinischen Personal Rückmeldung über den Gesundheitszustand des oder der betroffenen Patienten liefert. Das medizinische Personal wäre dadurch in der Lage in Extremsituationen wertvolle Zeit einzusparen, die evtl. über Leben und Tod eines Patienten entscheidet.

Zusätzlich zu dem gerade beschriebenen Einsatzbereich bei „Massenanfällen von Verletzten“ wäre dieses System eine wertvolle Ergänzung in der Intensivmedizin, da die personellen Resourcen, die seither für die Überwachung des Gesundheitszustandes von Patienten benötigt wurden nun anderweitig zur Verfügung stehen würden. Weitere Einsatzbereiche des mobilen vernetzten Systems wären die Therapiekontrolle und Rehabilitation, sowie der Leistungssport und das Fitnesstraining.

1.2. Stand der Technik

In der Studienarbeit von Stefan Fernsner (siehe [Fer09]) wurde bereits damit begonnen ein Sensor-/Aktor-Netzwerk für den Einsatz in einem medizinischen Umfeld zu konzipieren. Dieses Sensor-/Aktor-Netzwerk wird auf Basis einer modifizierten UART[2]-Schnittstelle betrieben.

Abbildung 1.1 zeigt die in [Fer09] vorgeschlagene Struktur des medizinischen Netzwerks.

[1] In diesem Zusammenhang ist hier die fortlaufende Überwachung der Vitalparameter eines Patienten mittels eines Sensorsystems gemeint ohne den Patienten dabei zu verletzen.

[2] Universal Asynchronous Receiver and Transmitter

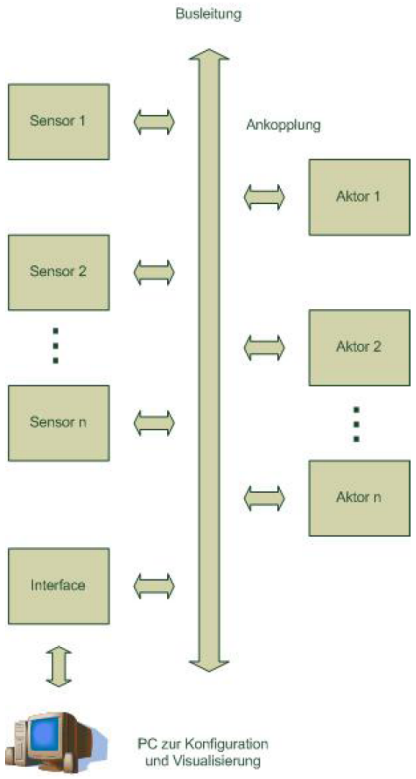

Abbildung 1.1.: Mögliche Struktur eines medizinischen Netzwerks [Fer09]

Die Netzteilnhmer des in Abbildung 1.1 dargestellten Netzwerks sind Sensoren, Aktoren und ein Interface.

Die Sensoren sind in der Lage, per Tastendruck einen kritischen Kreislaufzustand (z.B. Herz- oder Atemstillstand) einer verletzten Person zu simulieren. Der simulierte kritische Kreislaufzustand löst das Versenden einer Nachricht an einen Aktor oder an das Interface aus. Diese Nachricht gibt dem behandelnden medizinischen Personal Auskunft über den Gesundheitszustand der verletzten Person.

Die Aktoren können Nachrichten, die von einem oder mehreren Sensoren versendet wurden empfangen und bei Bedarf eine Aktion auslösen. Ein Aktor kann beispielsweise einen kritischen Kreislaufzustand einer verletzten Person über eine LED[3] visualisieren.

Das Interface empfängt ebenfalls wie ein Aktor Nachrichten, die von einem oder mehreren Sensoren versendet wurden. Es ist dabei in der Lage die empfangenen Nachrichten an einen PC weiterzuleiten, der eine aufwändigere Visualisierung der Daten als ein Aktor zulässt. An einem PC können z.B. Messwerte in einem Diagramm visualisiert werden.

Wie bereits oben angesprochen wurde, verwendet das Sensor-/Aktor-Netzwerk aus Abbil-

[3] Light Emitting Diode

dung 1.1 zur Kommunikation zwischen Sensoren, Aktoren und dem Interface eine modifizierte UART-Schnittstelle. UART wurde ursprünglich zur Verbindung von *genau* zwei Kommunikationspartnern entwickelt. Sollen mehrere Netzteilnehmer, wie in Abbildung 1.1 über ein Bussystem miteinander verbunden werden, so muss eine Möglichkeit geschaffen werden, dass niemals zwei oder mehr Netzteilnehmer zur gleichen Zeit Daten senden, da es sonst zu Konflikten auf der Busleitung kommen kann. Durch die Einführung zweier Arbitrierungsleitungen, die in [Fer09] *Semaphoren* genannt werden, können Konflikte beim Buszugriff verhindert werden.

In Abbildung 1.2 ist die Belegung der Steckverbinder für die Verbindung zwischen Sensoren, Aktoren und dem Interface dargestellt.

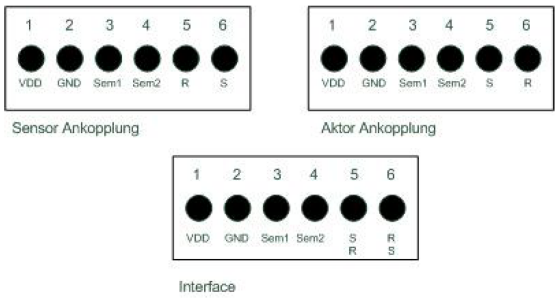

Abbildung 1.2.: Belegung der Steckverbinder von Sensoren, Aktoren und dem Interface [Fer09]

Alle Busteilnehmer werden mit einer 1:1-Verkabelung miteinander verbunden. D.h. alle Pins, die jeweils dieselbe Nummer tragen sind miteinander verbunden. Pin 1 und Pin 2 dient der gemeinsamen Spannungsversorgung des medizinischen Netzwerks. Pin 3 und Pin 4 sind die beiden Leitungen, die zur Busarbitrierung benötigt werden (Semaphoren). Pin 5 ist bei Sensoren der Pin für eingehende Daten, bei Aktoren der Pin für ausgehende Daten. Beim Interface kann dieser Pin zwar beide Funktionen erfüllen, aber nicht gleichzeitig. Pin 6 ist bei Sensoren der Pin für ausgehende Daten und entsprechend bei Aktoren der Pin für eingehende Daten. Pin 6 erfüllt auch beim Interface beide Funktionen, aber auch hier nicht gleichzeitig. Das Interface muss manuell umgeschaltet werden (vertauschen der Pins 5 und 6), je nach dem, ob es mit einem Sensor oder mit einem Aktor kommunizieren möchte.

Anhand der Belegung der Steckverbinder aus Abbildung 1.2 lässt sich der „UART-Charakter“ des Netzwerks leicht erkennen. Zur Kommunikation werden zwei unidirektionale[4] Leitungen verwendet. Aus Sicht eines Sensors wird die Leitung, die an Pin 5 angekoppelt ist nur zum Empfangen verwendet. Die Leitung die an Pin 6 angekoppelt ist, wird aus Sicht des Sensors nur zum Senden verwendet. D.h. Sensoren können untereinander nicht miteinander kommunizieren. Ebenso wenig können Aktoren untereinander kommunizieren. Bei dieser Art der physikalischen Verbindung können nur Sensoren mit Aktoren und umgekehrt Aktoren mit Sensoren kommunizieren. Das Interface muss die Belegung der Pins 5 und 6 tauschen, je nach dem, ob das Interface mit einem Sensor oder mit einem Aktor kommunizieren möchte.

[4] „in eine Richtung“

Das oben angesprochene Problem, dass nicht jeder Busteilnehmer mit jedem beliebigen anderen Busteilnehmer kommunizieren kann ist die größte Schwachstelle des in [Fer09] vorgestellten Netzwerkkonzeptes. Des weiteren verwendet das medizinische Netzwerk aus Abbildung 1.1 für Sensoren und Aktoren unterschiedliche Hardware, was nicht sehr kosteneffizient ist.

1.3. Ziele der Arbeit

Dem in Abschnitt 1.1 angesprochenem Problem, nämlich die Schwierigkeit der Erfassung des Gesundheitszustandes dutzender verletzter Personen soll nun mit dieser Diplomarbeit durch die Konzipierung und Entwicklung eines intelligenten, mobilen medizinischen Ad-hoc[5] Netzwerks Abhilfe geschaffen werden. Zusätzlich soll das in [Fer09] beschriebene Netzwerkkonzept so erweitert und verbessert werden, um die in Abschnitt 1.2 genannten Schwachstellen zu beheben.

Das kabelgebundene vernetzte System soll in der Lage sein den Gesundheitszustand verletzter Personen über Sensoren zu erfassen und bei Bedarf, also bei Auftreten eines kritischen Kreislaufzustandes (z.B. Herz- oder Atemstillstand), eine geeignete Aktion auslösen (z.B. Verschicken einer Nachricht an den behandelnden Arzt). Des weiteren soll das System in der Lage sein aktuelle Messwerte der Sensoren zu übertragen, die dann vom medizinischen Personal z.B. an einem PC oder an einem PDA[6] visualisiert und ausgewertet werden können.

Um die Praxistauglichkeit und die Zuverlässigkeit eines solchen medizinischen Netzwerkes zu gewährleisten, muss dieses zusätzlich zu den beiden oben genannten funktionalen Anforderungen noch den folgenden weiteren technischen Anforderungen gerecht werden:

1. Das vernetzte System soll Multi-Master fähig sein. Das bedeutet, dass jeder beliebige Netzwerkteilnehmer in der Lage ist selbständig Daten zu versenden ohne sich vorher die Sendeerlaubnis bei einem anderen Netzteilnehmer einholen zu müssen[7].

2. Das vernetzte System soll echtzeitfähig sein. D.h. eine durch einen Sensor gesendete Nachricht *muss* einnerhalb einer definierten Zeitspanne bei einem Empfänger (z.B. einem Aktor) angekommen sein.

3. Das vernetzte System soll ausfallsicher sein, d.h. der Ausfall bzw. das Fehlen eines oder mehrerer Netzteilnehmer darf keine funktionalen Konsequenzen auf das verbleibende System haben.

[5] „für den Moment gemacht“
[6] Personal Digital Assistant
[7] Dem gegenüber steht das Master-Slave-Prinzip. Ein spezieller Netzteilnehmer, der Busmaster, erteilt allen anderen Netzteilnehmern der Reihe nach die Sendeerlaubnis.

4. Das vernetzte System soll übertragungssicher sein. Durch geeignete Maßnahmen muss eine hohe Datenintegrität[8] übertragener Nachrichtenpakete sichergestellt werden.

5. Das vernetzte System soll gut skalierbar sein. D.h. das vernetzte System soll auch während des Betriebs mit möglichst geringem Aufwand um beliebige weitere Netzteilnehmer erweitert werden können. Dies trägt zur Steigerung der Benutzerfreundlichkeit des Systems bei.

6. Die Netzteilnehmer sollen mittels eines Terminals (z.B. PC oder PDA) remote[9] konfigurierbar sein. Das bedeutet, dass jedem Netzteilnehmer zu jeder Zeit über das Terminal eine andere Funktion zugewiesen werden kann. Es wäre z.B. denkbar einen Sensor, der im Moment nur Daten, die den Kreislaufzustand des Patienten betreffen, überträgt so umzukonfigurieren, dass dieser Sensor zusätzlich auch Pulsmesswerte überträgt.

7. Es soll die Möglichkeit geschaffen werden Ereignisse (z.B. Herzstillstand) und auch Messwerte (z.B. Pulsmesswerte) in einer Datenbank abzuspeichern, um die Messwerte jederzeit zur Verfügung zu haben.

1.4. Vorgehensweise und Gliederung der Arbeit

Zunächst werden in Kapitel 2 einige theoretische Grundlagen erläutert, die für das weitere Verständnis der darauf folgenden Kapitel nützlich sein könnten. Es werden u.a. Signalübertragungsverfahren für elektrische Leiter besprochen. Ebenso wird auch auf die Grundlagen der Leitungstheorie eingegangen.

In Kapitel 3 werden alle Schritte beschrieben, die zur Konzipierung eines intelligenten, mobilen medizinischen Ad-hoc Netzwerks nötig sind. Darunter fällt die Auswahl einer geeigneten Netztopologie[10] sowie die Auswahl eines geeigneten Bussystems für das medizinische Netzwerk. Des weiteren wird ein Kommunikationsprotokoll entworfen, das für die Datenkommunikation innerhalb des medizinischen Netzwerks zuständig ist.

In Kapitel 4 wird die Hardware der am medizinischen Netzwerk beteiligten Komponenten erläutert. Zu den Hardwarekomponenten zählen der Sensor-/Aktor-Knoten, der Interface-Knoten, die Bustreiber-Adapter sowie die Stromversorgung des medizinischen Netzwerks. Außerdem wird in Kapitel 4 die Firmware, die für den Betrieb der Sensor-/Aktor-Knoten sowie des Interface-Knotens nötig ist, beschrieben.

In Kapitel 5 werden Testverfahren erläutert mit, denen das medizinische Netzwerk hinsichtlich der maximal möglichen Buslänge, der elektromagnetischer Verträglichkeit und

[8] Datenintegrität bedeutet Schutz vor einer ungewollten Veränderung der übertragenen Information.
[9] „aus der Entfernung“
[10] Unter Netztopologie wird die physikalische Struktur der Verbindungen von Komponenten verstanden, um einen gemeinsamen Datenaustausch zu ermöglichen.

des Stromverbrauchs überprüft wird.

In Kapitel 6 werden die Ergebnisse diskutiert und diese Arbeit in Bezug zum Stand der Technik gesetzt.

In Kapitel 7 folgt schließlich die Zusammenfassung der Arbeit. Darüber hinaus wird ein Ausblick über sinnvolle Erweiterungen des Systems gegeben.

2. Theoretische Grundlagen

2.1. Überblick

In Abschnitt 2.2 werden Signalübertragungsverfahren für elektrische Leiter beschrieben. Zu den erläuterten Verfahren zählen die asymmetrische Signalübertragung, die pseudo-differenzielle Signalübertragung sowie die vollständig differenzielle Signalübertragung.

In Abschnitt 2.3 werden Endstufentypen, die in integrierten digitalen Schaltkreisen häufig angetroffen werden, erläutert. Die Open-Collector-Endstufe sowie die Push-Pull-Endstufe mit Tri-State-Ausgang wird oft in Verbindung mit Bussystemen eingesetzt. Die Push-Pull-Endstufe ohne Tri-State-Ausgang befindet sich in jedem Microcontroller zur Ansteuerung der I/O-Pins[1].

Abschnitt 2.4 erläutert die Leitungstheorie homogener Leitungen. Bei Kommunikationssystemen, deren Ausdehnung nicht vernachlässigbar ist, z.B. wenn Strecken über mehrere hundert Meter überbrückt werden sollen, ist es wichtig zu verstehen wie sich elektrische Signale auf Leitungen ausbreiten. In Abschnitt 2.4 werden zunächst einige physikalische Grundlagen der Leitungstheorie erläutert. Der Abschnitt endet mit einem Beispiel, das die Signalübertragung auf einem Twisted-Pair-Kabel[2] verdeutlicht.

2.2. Signalübertragungsverfahren für elektrische Leiter

2.2.1. Überblick

In den Abschnitten 2.2.2 sowie 2.2.3.1 und 2.2.3.2 werden Verfahren zur Signalübertragung auf elektrischen Leitern vorgestellt.

Bei den Signalübertragungsverfahren handelt es sich in Abschnitt 2.2.2 um die asymmetrische Signalübertragung. In den Abschnitten 2.2.3.1 und 2.2.3.2 werden zwei Verfahren zur symmetrischen Signalübertragung, nämlich zum einen die pseudo-differenzielle Signalübertragung und zum anderen die vollständig differenzielle Signalübertragung vorgestellt.

[1] Input/Output-Pins
[2] Ein Twisted-Pair-Kabel ist eine verdrillte Zweidrahtleitung.

Vor allem bei der Datenübertragung auf langen elektrischen Leitern hat die Wahl des Signalübertragungsverfahrens einen entscheidenden Einfluss auf die am Leitungsende zu erwartende Signalqualität[3].

2.2.2. Asymmetrische Signalübertragung

Bei der asymmetrischen Signalübertragung wird das zu übertragende Signal nur auf einer Leitung von einem Sender zu einem Empfänger transportiert. Abbildung 2.1 veranschaulicht die Signalübertragung zwischen einem Sender T1 und einem Empfänger R1.

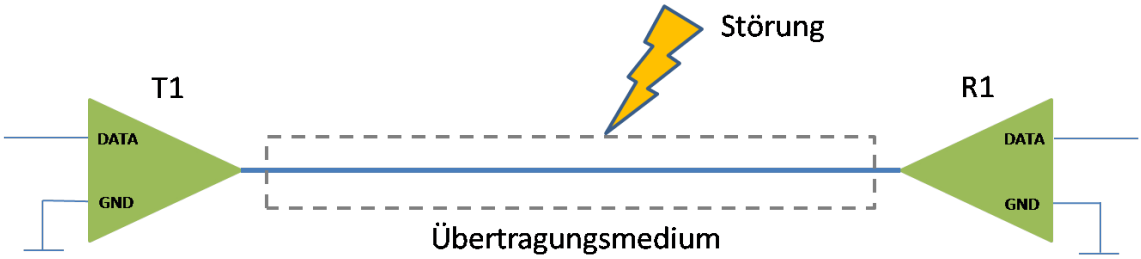

Abbildung 2.1.: Asymmetrische Signalübertragung zwischen Sender und Empfänger

Der Sender T1 gibt die zu übertragenden Daten auf das Übertragungsmedium. Auf der Übertragunsstrecke verfälschen eingekoppelte kapazitive und induktive Störungen das übertragene Datensignal. Der Empfänger R1 interpretiert das empfangene Datensignal bezüglich GND (Masse). Die beiden Massepotentiale des Senders T1 und des Empfängers R1 müssen dabei nicht zwangsläufig übereinstimmen.

Diese Art der Datenübertragung ist am einfachsten und kostengünstigsten zu realisieren, aber auch am störanfälligsten. Der Empfänger R1 hat keine Möglichkeit die eingekoppelten Störungen zu eliminieren, da kein Referenzsignal, das dem Empfänger Auskunft über die Störungen gibt, mit übertragen wird.

Im Abschnitt 2.2.3 werden zwei weitere Verfahren beschrieben, mit denen eingekoppelte Störungen teilweise bzw. ganz eliminiert werden können.

2.2.3. Symmetrische Signalübertragung

2.2.3.1. Pseudo-differenzielle Signalübertragung

Bei der pseudo-differenziellen Signalübertragung werden zusätzlich zur Datenleitung noch eine oder meherere Masseleitungen mitgeführt. Abbildung 2.2 zeigt das Prinzip der pseudo-

[3] Unter Signalqualität wird in diesem Zusammenhang ein für den Empfänger ausreichendes Signal-zu-Rausch-Verhältnis verstanden.

differenziellen Signalübertragung zwischen einem Sender und einem Empfänger.

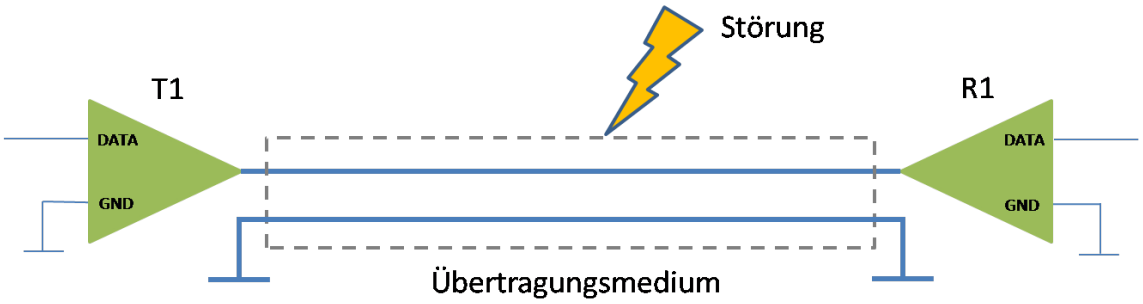

Abbildung 2.2.: Pseudo-differenzielle Signalübertragung zwischen Sender und Empfänger

Bei der pseudo-differenziellen Signalübertragung ist das Massepotential des Senders T1 über eine oder mehrere im Übertragungsmedium mitgeführten Leitungen mit dem Massepotential des Empfängers R1 verbunden. Auf der Übertragungsstrecke eingekoppelte kapazititve und induktive Störungen wirken sich sowohl auf die Signalleitung als auch auf die im Übertragungsmedium mitgeführten Masseleitungen aus. Der Empfänger R1 ist dadurch in der Lage zumindest die auf der Übertragungsstrecke hinzugekommenen Störungen zu eliminieren. Störungen, die nicht auf der Übertragungsstrecke hinzugekommen sind, sondern beispielsweise erst am Empfänger einwirken, können nicht eliminiert werden, da sich diese Art von Störungen nicht gleichermaßen auf die Daten- und Masseleitungen auswriken.

Im Vergleich zur asymmetrischen Datenübertragung ist dieses Verfahren zur Signalübertragung nur unwesentlich aufwendiger zu realisiern, da lediglich eine zusätzliche Masseleitung im Übertragungsmedium mitgeführt werden muss. Der zusätzliche Nutzen (die bessere Signalqualität) dieses Signalübertragungsverfahrens im Vergleich zur asymmetrischen Signalübertragung steht in einem sehr guten Verhältnis zu den zusätzlichen Kosten, die durch eine zusätzliche Ader im Übertragungsmedium entstehen.

2.2.3.2. Vollständig differenzielle Signalübertragung

Bei der vollständig differenziellen Signalübertragung wird auf einer Leitung das Datensignal und auf einer zweiten Leitung eine invertierte[4] Kopie des Datensignals der ersten Leitung übertragen. Abbildung 2.3 zeigt das Prinzip der vollständig differenziellen Signalübertragung zwischen Sender und Empfänger.

Die beiden Datenleitungen D+ und D- werden miteinander verdrillt. Dieser Kabeltyp wird Twisted-Pair-Kabel genannt. Wie auch bei der pseudo-differenziellen Signalübertragung wird durch die räumlich nahe beeinander liegenden Datenleitungen D+ und D- sicher gestellt, dass sich eingekoppelte kapazitive und induktive Störungen auf beide Datenleitungen gleichermaßen auswirken. Damit wird erreicht, dass sich die eingekoppelten Störungen eliminieren lassen. Der Empfänger R1 kombiniert die beiden Datenleitungen D+ und D-

[4] Das invertierte Signal entsteht durch Multiplikation des Originalsignals mit minus eins.

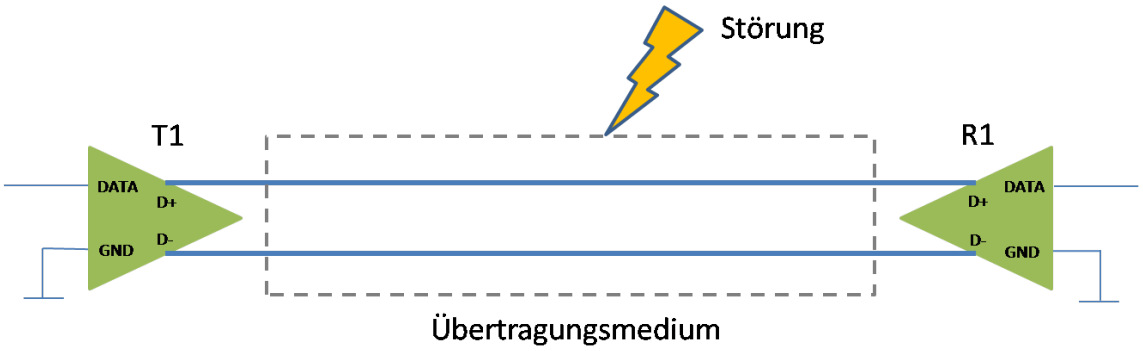

Abbildung 2.3.: Vollständig differenzielle Signalübertragung zwischen Sender und Empfänger

durch Subtraktion wieder zu einem einzigen Signal, das der Empfänger gegenüber Masse interpretiert. Das Massepotential des Senders T1 muss bei diesem Signalübertragungsverfahren nicht mit dem Massepotential des Empfängers R1 übereinstimmen.

Im Folgenden sei ein zu übertragendes Datensignal $s(t)$ und ein durch eine Störung eingekoppeltes Rauschen $n(t)$ gegeben.

Auf der D+ Leitung liegt das Signal $d_+(t) = s(t) + n(t)$ an. Auf der D- Leitung liegt entsprechend das Signal $d_-(t) = -s(t) + n(t)$ an.

Der Empfänger R1 bildet das Differenzsignal $\Delta(t)$ durch Subtraktion von D+ und D-. Das Ergebnis ist:

$$\Delta(t) = d_+(t) - d_-(t) = s(t) + n(t) - (-s(t) + n(t)) = 2 \cdot s(t)$$

Dieses Ergebnis zeigt, dass mittels der vollständig differenziellen Signalübertragung nicht nur die auf der Übertragungsstrecke hinzugekommenen Störungen eliminiert werden können, sondern zusätzlich auch noch das SNR[5] um $20 \cdot log\ 2 = 6\ dB$ gegenüber der asymmetrischen bzw. der pseudo-differenziellen Signalübertragung verbessert werden kann.

2.3. Verwendete Endstufentypen in integrierten digitalen Schaltkreisen

2.3.1. Überblick

In Abschnitt 2.3.2 wird die Open-Collector-Endstufe erläutert. Dieser Endstufentyp wird aufgrund seiner einfachen Struktur häufig bei kostengünstigen Bussystemen eingesetzt.

[5] Signal-to-Noise Ratio

Integrierte Schaltkreise, die eine Open-Collector-Endstufe besitzen können an ihren Ausgängen zusammengeschaltet werden, ohne dass Kurzschlüsse auftreten.

Die in Abschnitt 2.3.3 beschriebene Push-Pull-Endstufe ohne Tri-State-Ausgang befindet sich in jedem Microcontroller zur Ansteuerung der I/O-Pins. Des weiteren wird dieser Endstufentyp bei Schnittstellen zur Datenkommunikation, wie beispielsweise bei der EIA[6]-232-Schnittstelle eingesetzt.

In Abschnitt 2.3.4 wird die Push-Pull-Endstufe mit Tri-State-Ausgang beschrieben. Sie findet hauptsächlich bei differenziell betriebenen Bussystemen nach dem EIA-485 Standard Anwendung. Bussysteme mit Push-Pull-Endstufe mit Tri-State-Ausgang sind im Vergleich zu Bussystemen mit Open-Collector-Endstufe in Hardware aufwändiger zu realisieren und daher aus Kostengesichtspunkten Bussystemen mit Open-Collector-Endstufen unterlegen.

2.3.2. Die Open-Collector-Endstufe

Abbildung 2.4 zeigt einen integrierten Schaltkreis mit einer Open-Collector-Endstufe[7].

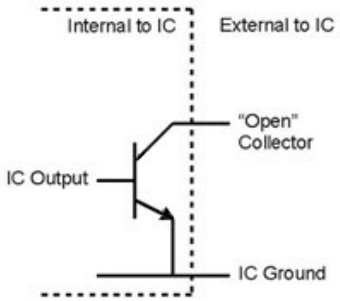

Abbildung 2.4.: Schaltbild einer Open-Collector-Endstufe [Wik09]

Wird der Transistor in Abbildung 2.4 geschlossen, so erscheint am Open-Collector-Ausgang eine logische Null. Um eine logische Eins am Open-Collector-Ausgang zu erzeugen wird der Transistor geöffnet. Zusätzlich muss der Open-Collector-Ausgang über einen Pull-up-Widerstand mit dem HIGH-Pegel verbunden werden.

Abbildung 2.5 zeigt die Zusammenschaltung mehrerer integrierter Schaltkreise mit Open-Collector-Ausgang und einem Pull-up-Widerstand.

Im Ruhezustand sind alle Transistoren der integrierten Schaltkreise G_1 bis G_n geöffnet. Die gemeinsame Leitung führt eine logische Eins ($U_a = U_b$). Schließt nun ein IC[8] seinen

[6] Electronic Industries Alliance
[7] Bei integrierten Schaltkreisen, die in CMOS-Technologie gefertigt sind wird dieser Endstufentyp als Open-Drain-Endstufe bezeichnet.
[8] Integrated Circuit

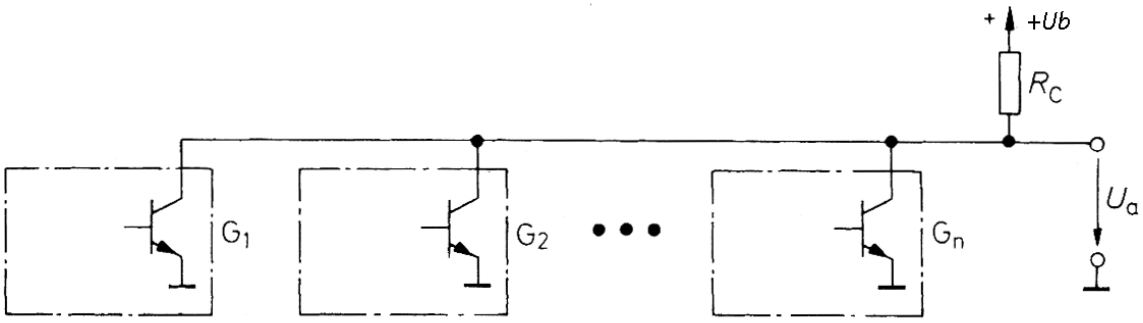

Abbildung 2.5.: Zusammenschaltung mehrerer Open-Collector-Ausgänge [Tie02]

Transistor, so fließt ein Strom von U_b über R_c durch den geschlossenen Transistor nach Masse. Die gemeinsame Leitung führt eine logische Null ($U_a = GND$).

Wie Abbildung 2.5 zeigt, können mehrere ICs an der gleichen Leitung betrieben werden, ohne miteinander in Konflikt zu geraten. Sobald ein IC seinen Transistor schließt, ist unabhängig vom Schaltzustand der anderen Transistoren, der logische Zustand der Leitung null. Diese Art der Verbindung wird *Wired-AND* genannt, da sich die Verbindung der ICs untereinander wie eine logische UND-Verknüpfung verhält.

Abbildung 2.6 vedeutlicht das Wired-AND-Verhalten der Anordnung aus Abbildung 2.5 anhand von zwei Gattern G_1 und G_2. Schaltet ein Transistor eines Gatters durch, so wird eine logische Null auf die Leitung gegeben. Öffnet ein Transistor eines Gatters so wird eine logische Eins auf die Leitung gegeben. Der logische Zustand der gemeinsamen Leitung ergibt sich aus einer logischen UND-Verknüpfung der Gatterzustände G_1 und G_2.

G_1	G_2	Logischer Zustand der Leitung
0	0	0
0	1	0
1	0	0
1	1	1

Abbildung 2.6.: Wahrheitstabelle der Wired-AND-Verknüpfung

Der Pegel, der durch einen Transistor niederohmig erzeugt wird, wird *dominanter Pegel* genannt (in obigem Beispiel der LOW-Pegel). Der Pegel, der durch den Pull-up-Widerstand erzeugt wird, wird *rezessiver Pegel* genannt (in obigem Beispiel der HIGH-Pegel). Wie Abbildung 2.6 zeigt, überschreibt ein dominanter Pegel einen rezessiven Pegel auf der gemeinsamen Leitung. Egal in welchen Schaltzuständen sich die Transistoren der ICs G_1 bis G_n befinden kann es nie zu Kurzschlüssen kommen.

2.3.3. Die Push-Pull-Endstufe ohne Tri-State-Ausgang

Die Push-Pull-Endstufe[9] ohne Tri-State-Ausgang[10] ist der gängigste Endstufentyp in digitalen integrierten Schaltkreisen. Abbildung 2.7 zeigt das Schaltbild einer Push-Pull-Endstufe ohne Tri-State-Ausgang.

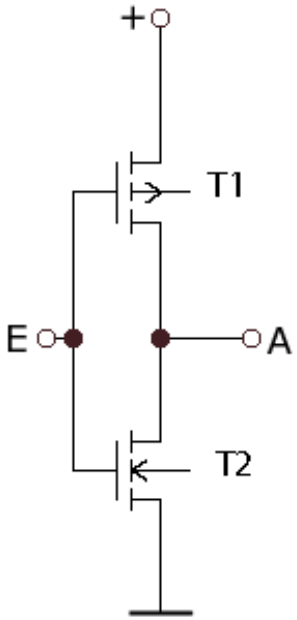

Abbildung 2.7.: Schaltbild einer Push-Pull-Endstufe ohne Tri-State-Ausgang [Wik09]

Mit dem Eingang E ist schaltkreisintern eine Logik verbunden, die darüber entscheidet ob am Ausgang A eine logische Null oder eine logische Eins erscheinen soll. Legt diese Logik an den Eingang E eine logische Null an, so leitet der Transistor T1 während der Transistor T2 sperrt. Am Ausgang A erscheint eine logische Eins. Legt die interne Logik an den Eingang E eine logische Eins an, so sperrt der Transistor T1, während der Transistor T2 leitet. Am Ausgang A erscheint eine logische Null. Die beiden Transistoren T1 und T2 arbeiten also immer im Gegentakt (deshalb wird diese Endstufe oft auch als Gegentaktendstufe bezeichnet). Es kommt niemals vor, dass beide Transistoren T1 und T2 gleichzeitig leiten. Es existiert jedoch eine modifizierte Form dieser Endstufe, bei der beide Transistoren T1 und T2 gleichzeitig speren können. In diesem Fall führt der Ausgang A kein definiertes Potential, er ist hochohmig. Dieser Endstufentyp wird Push-Pull-Endstufe mit Tri-State-Ausgang genannt (siehe Abschnitt 2.3.4).

Bei der Push-Pull-Endstufe ohne Tri-State-Ausgang ist es nicht möglich meherere Endstufen an ihren Ausgängen miteinander zu verbinden, da jede Endstufe einen definierten Pegel (entweder LOW oder HIGH) ausgibt. Dieses Verhalten kann beim Zusammenschalten von

[9] in der Literatur häufig auch als Gegentaktendstufe oder Totem-Pole-Ausgang bezeichnet

[10] Ein Tri-State-Ausgang ist ein Ausgang in der Digitaltechnik, der in einen hochohmigen Zustand versetzt werden kann.

Ausgängen zu Kurzschlüssen führen, nämlich dann, wenn eine Endstufe einen LOW-Pegel ausgibt während eine andere Endstufe einen HIGH-Pegel ausgibt oder umgekehrt.

2.3.4. Die Push-Pull-Endstufe mit Tri-State-Ausgang

Das Schaltverhalten einer Push-Pull-Endstufe mit Tri-State-Ausgang ist mit dem Schaltverhalten einer Push-Pull-Endstufe ohne Tri-State-Ausgang identisch (siehe Abschnitt 2.3.3). Jedoch gibt es bei der Push-Pull-Endstufe mit Tri-State-Ausgang zusätzlich die Möglichkeit den Ausgang der Endstufe in einen hochohmigen Zustand zu versetzen. Abbildung 2.8 zeigt den prinzipiellen Aufbau einer Push-Pull-Endstufe mit Tri-State-Ausgang.

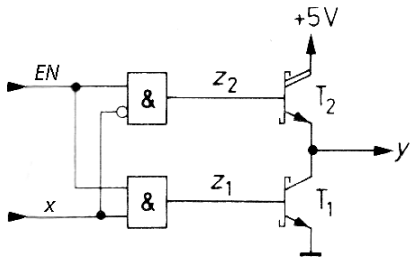

Abbildung 2.8.: Schaltbild einer Push-Pull-Endstufe mit Tri-State-Ausgang [Tie02]

Im Normalbetrieb, d.h. bei $EN = 1$, arbeiten die Transistoren T1 und T2 im Gegentaktbetrieb. Der Ausgang y führt in diesem Fall einen logischen Pegel (entweder HIGH oder LOW) in Abhängigkeit des Eingangs x. Die Endstufe kann mit $EN = 0$ in einen hochohmigen Zustand versetzt werden (Tri-State). Bei $EN = 0$ sperren beide Transistoren T1 und T2, damit führt der Ausgang y keinen definierten Pegel.

Der Vorteil einer Push-Pull-Endstufe mit Tri-State-Ausgang gegenüber einer Push-Pull-Endstufe ohne Tri-State-Ausgang ist, dass die Ausgänge y mehrerer Endstufen zusammen geschaltet werden können, sofern sicher gestellt wird, dass immer nur *eine* Endstufe zur selben Zeit aktiv ist ($EN = 1$). Zu diesem Zeitpunkt müssen alle anderen Endstufen Tri-State sein ($EN = 0$).

2.4. Leitungstheorie homogener Leitungen

Abbildung 2.9 zeigt das Ersatzschaltbild eines infinitesimal kleinen Stücks dx einer unendlichen langen homogenen Leitung[11].

Durch unendlich wiederholte Serienschaltung des Vierpols aus Abbildung 2.9 kann jede beliebig lange Leitung modelliert werden.

[11] Homogene Leitungen sind Leitungen, deren physikalische Eigenschaften sich in Ausdehnungsrichtung nicht ändern.

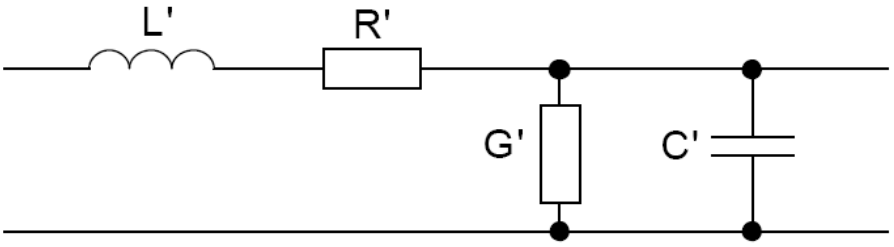

Abbildung 2.9.: Ersatzschaltbild eines infinitesimal langen Leitungsstücks [Wik09]

Die Größen L', R', G' und C' werden Leitungsbeläge genannt. Die Werte der Leitungsbeläge sind proportional zur Leitungslänge, sie hängen hauptsächlich vom verwendeten Isolationsmaterial des Leiters sowie der geometrischen Anordnung der Adern innerhalb des Leiters ab (Koaxialkabel unterscheiden sich beispielsweise deutlich in der geometrischen Anordnung der Adern von Twisted-Pair-Kabeln). Im Folgenden sollen die Leitungsbeläge kurz beschrieben werden.

Induktivitätsbelag L' Der Induktivitätsbelag L' hat die Einheit Henry pro Meter $[L'] = \frac{H}{m}$. Er stellt den Induktivitätswert der Leitung pro Längeneinheit dar. Bei Leitungen, die eng beieinander liegende Adern haben, ist der Induktivitätsbelag sehr gering.

Widerstandsbelag R' Der Widerstandsbelag R' hat die Einheit Ohm pro Meter $[R'] = \frac{\Omega}{m}$. Er stellt den ohmschen Widerstand der Leitung pro Längeneinheit dar. Bei langen Leitungen (über 1000 Meter) darf der Widerstandsbelag im Allgemeinen nicht vernachlässigt werden, da die ohmschen Verluste zu Spannungseinbrüchen entlang der Leitung führen. Dies kann zu Problemen bei der Datenübertragung führen.

Ableitungsbelag G' Der Ableitungsbetrag G' hat die Einheit Siemens pro Meter. $[G'] = \frac{S}{m}$. Er stellt die Isolationsverluste der Leitung pro Längeneinheit dar. Die Isolationsverluste sind typischerweise deutlich geringer als die ohmschen Verluste der Leitung. Der Ableitungsbetrag muss allerdings bei Datenübertragungen mit sehr hohen Frequenzen (über 10 MHz) berücksichtigt werden.

Kapazitätsbelag C' Der Kapazitätsbelag C' hat die Einheit Farad pro Meter. $[C'] = \frac{F}{m}$. Er stellt den Kapazitätswert der Leitung pro Längeneinheit dar. Der Kapazitätsbelag hängt hauptsächlich von der geometrischen Anordnung der Adern innerhalb eines Leiters ab. Er verhält sich reziprok zum Induktivitätsbelag der Leitung.

Eine wichtige Kenngröße zur charakterisierung elektrischer Leiter ist die sog. *charakteristische Impedanz* der Leitung, allgemein auch als *Wellenwiderstand* der Leitung bezeichnet. Der Wellenwiderstand beschreibt das Verhältnis zwischen Strom und Spannung einer sich in einem elektrischen Leiter ausbreitenden elektromagnetischen Welle. Nach [Bos96] ist der Wellenwiderstand einer elektrischen Leitung allgemein

$$Z_w = \sqrt{\frac{R' + j\omega L'}{G' + j\omega C'}}.$$

Der Wellenwiderstand ist, wie obige Formel zeigt, komplexwertig. Bei der Datenübertragung wird aus Kostengründen häufig rechteckige Impulsformung eingesetzt, d.h. die zu übertragenden Daten werden physikalisch auf der Leitung als Rechteckimpulse dargestellt. Bei einem Pegelwechsel eines Rechtecksignals kommt es zu einer steilen Flanke, die spektral betrachtet hohe Frequenzen erzeugt. Im Falle einer rechteckigen Impulsformung lässt sich obige Formel vereinfachen, da $\omega L' >> R'$ und $\omega C' >> G'$ gilt. Daraus ergibt sich ein rein reeller Wellenwiderstand

$$Z_w = \sqrt{\frac{L'}{C'}}. \tag{2.1}$$

Wird eine elektrische Leitung mit ihrem Wellenwiderstand abgeschlossen, so wird diese Leitung als *angepasste Leitung* bezeichnet. Der Vorteil angepasster Leitungen gegenüber nicht angepasster Leitungen ist, dass bei ihnen das übertragene Signal am Leitungsende nicht reflektiert wird. Es entsteht auf dem Übertragungsmedium also keine rücklaufende Welle. Bei fehlangepassten Leitungen, also Leitungen, die nicht mit ihrem Wellenwiderstand abgeschlossen sind, überlagern sich die hinlaufende und die rücklaufende Welle. Dies führt unter Umständen dazu, dass der übertragene Signalpegel so stark verfälscht wird, dass eine Datenkommunikation nicht mehr möglich ist.

Nach [Wit02] wirken sich Reflexionen fehlangepasster Leitungen störend auf die die Datenkommunikation aus, wenn folgende Bedingung gilt:

$$\ell_{Leitung} > \frac{\lambda_{min}}{10}. \tag{2.2}$$

Ist die Länge der Leitung $\ell_{Leitung}$ größer als ein Zehntel der kleinsten im Signal vorkommenden Wellenlänge λ_{min}, so sollte die Leitung mit ihrem Wellenwiderstand abgeschlossen werden.

Die kleinste im Signal vorkommende Wellenlänge λ_{min} lässt sich mit der Beziehung

$$\lambda_{min} = \frac{c_{Kupfer}}{f_{max}} \tag{2.3}$$

ermitteln. Dabei ist c_{Kupfer} die Ausbreitungsgeschwindigkeit der elektromagnetischen Welle in Kupfer, f_{max} ist die im Signal maximal vorkommende Frequenz.

Nach [Bos96] hängt die Ausbreitungsgeschwindigkeit einer elektromagnetischen Welle in einem elektrischen Leiter vom Induktivitätsbelag L' sowie vom Kapazitätsbelag C' der Leitung folgendermaßen ab:

$$c_{Kupfer} = \frac{1}{\sqrt{L'C'}} \tag{2.4}$$

Nun soll noch am Beispiel eines Twisted-Pair-Kabels die Verwendung der Gleichungen (2.1) bis (2.4) erläutert werden.

Gegeben ist ein Twisted-Pair-Kabel mit den Leitungsbelägen $L' = 865\ nH/m$ und $C' = 60\ pF/m$. Die Leitung wird mit einem rechteckigen Datensignal der Frequenz $f = 80\ kHz$ betrieben.

Nach Gleichung (2.1) ergibt sich der Wellenwiderstand dieser Leitung zu

$$Z_w = \sqrt{\frac{L'}{C'}} = \sqrt{\frac{865\ nH/m}{60\ pF/m}} \approx 120\ \Omega.$$

Die Ausbreitungsgeschwindigkeit des Signals beträgt nach Gleichung (2.4)

$$c_{Kupfer} = \frac{1}{\sqrt{L'C'}} = \frac{1}{\sqrt{865\ nH/m \cdot 60\ pF/m}} \approx 1,388 \cdot 10^8\ \frac{m}{s}.$$

Mit Gleichung (2.2) und der Beziehung aus Gleichung (2.3) lässt sich die maximale Leitungslänge einer nicht angepassenten Leitung ermitteln, bei der noch eine störungsfreie Datenkommunikation möglich ist:

$$\ell_{Leitung,max} = \frac{\lambda_{min}}{10} = \frac{c_{Kupfer}}{10 f_{max}} = \frac{1,388 \cdot 10^8\ m/s}{10 \cdot 80\ kHz} \approx 173,5\ m.$$

3. Konzeption des medizinischen Netzwerks

3.1. Überblick

Im vorliegenden Kapitel werden alle Schritte beschrieben, die zur Konzipierung eines intelligenten, mobilen medizinischen Ad-hoc Netzwerks nötig sind. Die Konzipierung des Systems hat großen Einfluss auf die Einhaltung der in Abschnitt 1.3 beschriebenen Systemanforderungen.

Zunächst werden in Abschnitt 3.2 einige Netztopologieen vorgestellt, im Anschluss daran wird eine aus Anforderungsgesichtspunkten geeignete Netztopologie für die Realisierung des vernetzten Systems ausgewählt.

In Abschnitt 3.3 werden geeignete Bussysteme, ebenfalls unter Anforderungsgesichtspunkten, vorgestellt und eines davon für die Entwicklung des medizinischen Netzwerks ausgewählt.

Abschnitt 3.4 gibt einen schematischen Überblick über das Gesamtsystem des medizinischen Netzwerks. Es werden alle Komponenten, die zur Funktionalität des medizinischen Netzwerks beitragen übersichtlich in einem Strukturdiagramm dargestellt.

In Abschnitt 3.5 wird auf den Entwurf eines Kommunikationsprotokolls für den Einsatz in einem medizinischen Netzwerk eingegangen. Das Kommunikationsprotokoll dient zur Datenübertragung zwischen einzelnen Netzteilnehmern.

3.2. Vorstellung und Auswahl geeigneter Netztopologieen

3.2.1. Überblick

In den Abschnitten 3.2.2 bis 3.2.6 werden die fünf grundlegenden Netztopologieen erläutert. Zu diesen zählen die Stern-Topologie, die Ring-Topologie, die Bus-Topologie, die Baum-Topologie sowie die vermaschten Netzwerke. In komplex strukturierten Netzwerken können die genannten grundlegenden Netztopologien in beliebiger Weise miteinander verschaltet sein. Ein solches Netzwerk wird *hybrides* Netzwerk genannt.

Die Wahl der richtigen Netztopologie für das medizinische Netzwerk ist ein entscheidendes Kriterium für die Zuverlässigkeit (Punkt 3 in Abschnitt 1.3) des gesamten Systems. Des weiteren spielt die Netztopologie eine zentrale Rolle für die Erweiterbarkeit des medizinischen Netzwerks (Punkt 5 in Abschnitt 1.3) durch Sensoren bzw. Aktoren. In Abschnitt 3.2.7 werden die vorgestellten Netztopologieen miteinander verglichen und eine geeignete Netztopologie für das medizinische Netzwerk ausgewählt.

Mit Hilfe der Graphentheorie, die die mathematische Grundlage aller Netztopologieen darstellt, lassen sich sämtliche Arten an Netztopologieen analysieren und auch bezüglich Performance (Datendurchsatz) und wirtschaftlichen Faktoren (z.B. Kosten für die Verkabelung) bewerten. Bei hybriden Netzwerken kommt es beispielsweise häufig vor, dass Teilstrecken des Netzwerks mit Glasfaserkabeln realisiert werden. Andere Netzsegmente werden mit Kupferkabeln verbunden. Durch die Einführung von Kantengewichten im resultierenden Netzgraphen und anschließender mathematischer Analyse des Netzwerks können die Kosten optimiert werden. Auf die Graphentheorie soll in dieser Arbeit nicht näher eingegangen werden. Ein ausführlicher Überblick zu diesem Thema befindet sich in [Die06].

3.2.2. Die Stern-Topologie

Bei Netzen in Stern-Topologie sind alle Netzteilnehmer über eine Point-to-point-Verbindung[1] an einem zentralen Verteilerknoten angeschlossen.

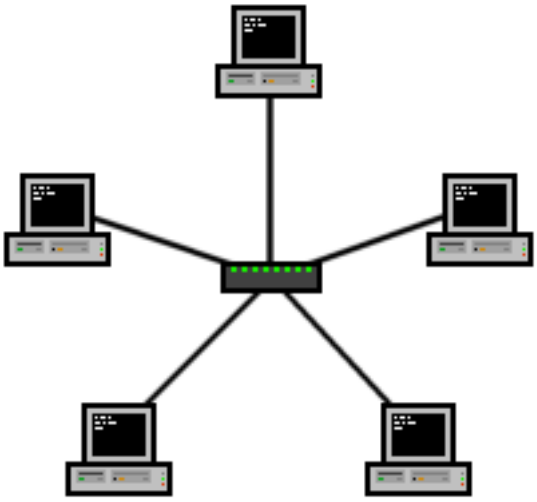

Abbildung 3.1.: Netzwerk in Stern-Topologie [Wik09]

Abbildung 3.1 zeigt, dass die Netzteilnehmer untereinander keine direkte Verbindung haben. Möchte ein Netzteilnehmer A mit einem Netzteilnehmer B kommunizieren, so muss die Verbindung über den zentralen Verteilerknoten hergestellt werden.

[1] Eine Point-to-point-Verbindung ist eine Direktverbindung. Im Falle der Stern-Topologie kann also ein Netzteilnehmer direkt mit dem Verteilerknoten kommunizieren.

Fällt ein Netzteilnehmer aus, so hat dies keine Auswirkungen auf die Funktionalität des verbleibenden Netzwerks. Fällt jedoch der zentrale Verteilerknoten aus, so ist das gesamte Netzwerk funktionsunfähig. Durch Hinzufügen von Redundanz im zentralen Verteilerknoten kann die Ausfallwahrscheinlichkeit des Verteilerknotens reduziert werden. Im Gegenzug steigt die Zuverlässigkeit des Netzwerks. Das Hinzufügen von Redundanz ist allerdings mit Kosten verbunden.

Soll ein Netzwerk in Stern-Topologie um einen weiteren Netzteilnehmer erweitert werden, so kann dies einfach durch Hinzufügen eines weiteren Netzteilnehmers an zentralen Verteilerknoten geschehen.

3.2.3. Die Ring-Topologie

Bei Netzen in Ring-Topologie sind jeweils zwei benachbarte Netzteilnehmer über eine Point-to-point-Verbindung so miteinander verbunden, dass ein geschlossener Ring entsteht. In Abbildung 3.2 ist ein Netzwerk in Ring-Topologie dargestellt.

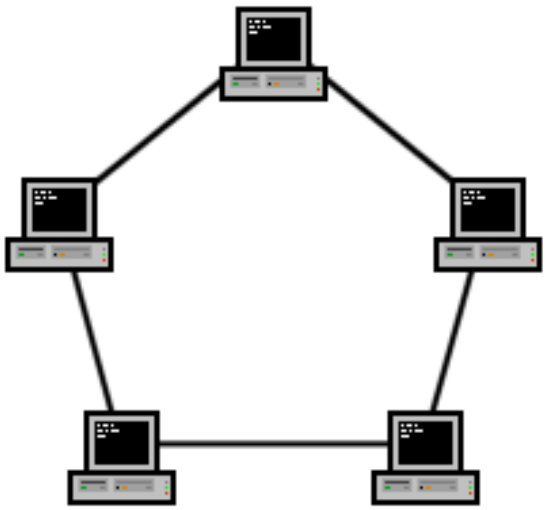

Abbildung 3.2.: Netzwerk in Ring-Topologie [Wik09]

Die zu übertragende Information wird von Netzteilnehmer zu Netzteilnehmer weitergeleitet, bis sie ihren Bestimmungsort erreicht. Dabei führt jeder Netzteilnehmer, der an der Weiterleitung der Information beteiligt ist eine Signalaufbereitung durch (die Information muss zur Weiterleitung zuerst empfangen werden, bevor sie wieder gesendet werden kann). Daher sind mit der Ring-Topologie sehr große Übertragungsdistanzen (mehrere Kilometer) realisierbar.

Fällt ein Netzteilnehmer aus, so wird aus der Ring-Topologie eine Spezialform, die sog. Linien-Topologie. Die Funktionsfähigkeit eines Netzwerkes in Linien-Topologie ist im Allgemeinen noch gewährleistet, fällt jedoch noch ein weiterer Teilnehmer aus, so zerbricht

die Linienstruktur in zwei voneinander getrennte Segmente. Eine Kommunikation zwischen den beiden Segmenten ist dann nicht mehr möglich.

Netzwerke in Ring-Topologie sind nicht sehr einfach erweiterbar, da immer darauf geachtet werden muss, dass die Ringstruktur nicht verloren geht. D.h. jeder neue Netzteilnehmer benötigt einen Vorgänger und einen Nachfolger. Der Vorteil bei Netzwerken in Ring-Topologie liegt darin, dass zwei benachbarte Netzteilnehmer über eine garantierte Übertragungsbandbreite verfügen, da benachbarte Netzteilnehmer über eine Direktverbindung miteinander verbunden sind.

3.2.4. Die Bus-Topologie

Bei Netzen in Bus-Topologie sind alle Netzteilnehmer über ein T-Stück[2], an einer gemeinsamen Datenleitung, dem sog. Bus angeschlossen.

Abbildung 3.3.: Netzwerk in Bus-Topologie [Wik09]

Die in Abbildung 3.3 dargestellte Bus-Topologie darf nicht mit der in Abschnitt 3.2.3 erwähnten Spezialform der Ring-Topologie (Linien-Topologie) verwechselt werden. Bei der Bus-Topologie ist die einzige aktive Komponente derjenige Netzteilnehmer, der gerade Daten sendet. Alle anderen Netzteilnehmer verhalten sich völlig passiv und führen im Gegensatz zu Netzteilnehmern in der Linien-Topologie *keine* Signalaufbereitung durch.

Die Verwendung einer einzigen gemeinsamen Datenleitung hat den Vorteil, dass beliebig viele Netzteilnehmer ausfallen dürfen ohne das verbleibende Bussystem funktional zu beeinträchtigen. Eine gemeinsame Datenleitung hat aber auch den Nachteil, dass sich alle Netzteilnehmer eine bandbreitenbeschränkte Ressource, nämlich die Busleitung, teilen müssen. Dies kann vor allem bei sehr hoher Busauslastung zu langen Wartezeiten führen bis ein sendewilliger Netzteilnehmer das Übertragungsmedium nutzen kann (siehe [Kie06]).

Netzwerke in Bus-Topologie lassen sich sehr einfach und kostengünstig erweitern. Hierzu

[2]Ein T-Stück ist eine mechanische Steckverbindung mit zwei horizontal angeordneten und einem vertikalen angeordnetem Anschluss. Aufgrund der optischen Ähnlichkeit zum Buchstaben „T“ wird dieser Steckverbinder T-Stück genannt.

müssen die neu hinzugekommenen Netzteilnehmer lediglich über T-Stücke an die gemeinsame Busleitung angeschlossen werden. Des weiteren besitzt die Bus-Topologie ein sehr hohes Maß an Zuverlässigkeit, denn der Ausfall eines oder mehrerer Busteilnehmer hat, wie bereits weiter oben beschrieben, keine funktionalen Auswirkungen auf das verbleibende Bussystem. Lediglich der unwahrscheinliche Fall eines Kabelbruchs in der Busleitung kann dazu führen, dass das gesamte Netzwerk ausfällt.

3.2.5. Die Baum-Topologie

Bei Netzen in Baum-Topologie sind alle Netzteilnehmer hierarchisch über eine Baumstruktur verbunden. Genau wie bei der Stern-Topologie sind niemals zwei Netzteilnehmer direkt miteinander verbunden.

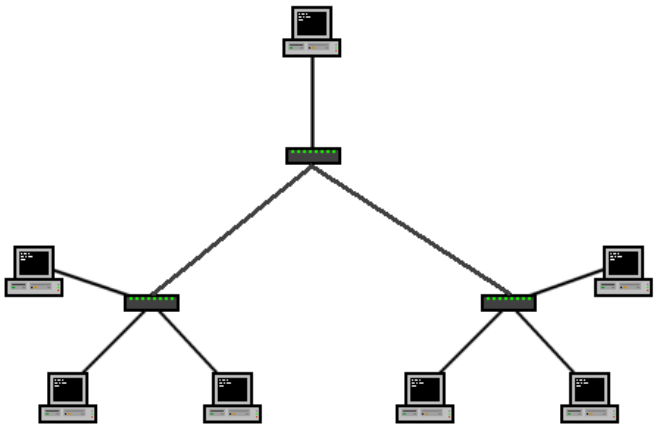

Abbildung 3.4.: Netzwerk in Baum-Topologie [Wik09]

Abbildung 3.4 zeigt schematisch die Struktur eines Netzwerks in Baum-Topologie. Die oberste Ebene dieses Netzwerks besteht genau aus einem Verteilerknoten (mit evtl. daran angeschlossenen Netzteilnehmern), dieser wird Wurzel des Baumes genannt. Die Netzteilnehmer werden Blätter des Baumes genannt, da von ihnen keine direkte Verbindung zur nächst tieferen Hierarchiestufe des Baumes ausgeht.

Bei Ausfall eines oder mehrerer Netzteilnehmer wird die Funktionalität des verbleibenden Netzwerks nicht beeinträchtigt, da wie schon gesagt, kein Netzteilnehmer eine direkte Verbindung zu einem anderen Netzteilnehmer hat. Fällt jedoch ein Verteilerknoten aus, so sind alle Hierarchiestufen unterhalb des ausgefallenen Verteilerknotens funktionsunfähig. Im schlimmsten Fall, nämlich bei Ausfall der Wurzel des Baumes, fällt das gesamte Netzwerk aus. Auch hier kann, wie bei Netzwerken in Stern-Topologie, durch Schaffung zusätzlicher Redundanz in den Verteilerknoten eine höhere Zuverlässigkeit des Netzwerks erreicht werden. Bei großen Bäumen wird dies allerdings sehr kostspielig, da mehrere Verteilerknoten zur Funktionsfähigkeit des Netzwerks beitragen.

Netzwerke in Baum-Topologie können einfach erweitert werden. Ein neu hinzugekommener Netzteilnehmer wird dabei entweder direkt an die Wurzel des Baumes angeschlossen oder an einen darunter liegenden Verteilerknoten.

3.2.6. Vermaschte Netzwerke

3.2.6.1. Teilweise vermaschtes Netz

In einem teilweise vermaschten Netz ist jeder Netzteilnehmer mit mindestens einem oder mehreren anderen Netzteilnehmern verbunden.

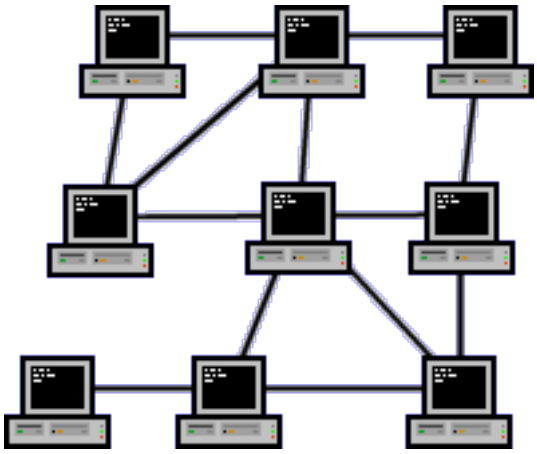

Abbildung 3.5.: Teilweise vermaschtes Netz [Wik09]

In Abbildung 3.5 ist diese Netzwerkstruktur schematisch dargestellt. Möchte nun ein Netzteilnehmer A mit einem Netzteilnehmer B kommunizieren, so gibt es *meistens* mehrere alternative Wege, um eine Nachricht von A nach B zu übertragen. Diese Redundanz trägt zwar zu einer hohen Zuverlässigkeit des Netzwerks bei, erfordert jedoch komplexe Routingverfahren[3].

Der Ausfall eines oder mehrerer Netzteilnehmer hat in der Regel keine funktionalen Konsequenzen auf das verbleibende Netzwerk, solange das verbleibende Netzwerk durch die Ausfälle *nicht* segmentiert wird. D.h. das verbleibende Netzwerk bleibt funktionsfähig solange es nicht in zwei oder mehr Teilnetze zerfällt.

Ein teilweise vermaschtes Netz kann nur sehr schwer erweitert werden, da jeder neu hinzugekommene Netzteilnehmer üblicherweise mit mehreren bereits vorhandenen Netzteilnehmern verbunden wird. Dies bedeutet einen hohen Verkabelungsaufwand und damit auch hohe Kosten. Prinzipiell wäre es denkbar einen neu hinzugekommenen Netzteilnehmer mit nur einem bereits vorhandenen Netzteilnehmer zu verbinden, dies hat aber auf Dauer den

[3] Routingkonzepte dienen dazu einen möglichst effizienten Weg (z.B. einen kurzen Weg) von einem Sender A zu einem Empfänger B zu bestimmen.

Nachteil, dass die alternativen Wege, um eine Nachricht von A nach B zu transportieren, immer weniger werden. Dadurch sinkt die Zuverlässigkeit des teilweise vermaschten Netzwerks enorm.

3.2.6.2. Vollständig vermaschtes Netz

Im Gegensatz zum teilweise vermaschten Netz ist beim vollständig vermaschten Netz *jeder* Netzteilnehmer mit *jedem* anderen Netzteilnehmer verbunden (siehe [Sch06]).

Das bedeutet, dass ein Netzteilnehmer A mit einem Netzteilnehmer B über eine Direktverbindung kommunizieren kann.

Der Ausfall eines oder mehrerer Netzteilnehmer beeinträchtigt zu keiner Zeit die Funktionsfähigkeit des verbleibenden Netzwerks, da die verbleiben Netzteilnehmer immernoch über Direktverbindungen kommunizieren können. Selbst der Ausfall einer Direktverbindung ist kein Problem, da es einen alternativen Weg gibt, der einen Netzteilnehmer A mit einem Netzteilnehmer B verbindet.

Vollständig vermaschte Netze lassen sich nur sehr schwer erweitern, da jeder neu hinzugekommene Netzteilnehmer mit jedem bereits vorhandenen Netzteilnehmer verbunden werden *muss*. Schon bei kleinen Netzwerken bedeutet dies einen enormen Verkabelungsaufwand, der mit hohen Kosten verbunden ist.

3.2.7. Auswahl einer geeigneten Netztopologie für den Einsatz in einem medizinischen Netzwerk

Die Auswahl einer geeigneten Netztopologie für den Einsatz in einem medizinischen Netzwerk hängt maßgeblich von den vier Faktoren Kosten, Zuverlässigkeit, Skalierbarkeit und Performance ab.

In Abbildung 3.6 werden die in Abschnitt 3.2.2 bis 3.2.6 vorgestellten elementaren Netztopologieen nun nach obigen Kriterien bewertet.

	Kosten	Zuverlässigkeit	Skalierbarkeit	Performance
Stern-Topologie	-	O	+	++
Ring-Topologie	O	-	O	O
Bus-Topologie	++	+	++	O
Baum-Topologie	-	O	+	+
Teilweise vermaschtes Netz	-	+	-	+
Vollständig vermaschtes Netz	--	++	--	++

Abbildung 3.6.: Vergleich der beschriebenen Netztopologieen

Aus Kostengesichtspunkten hat das vollständig vermaschte Netzwerk im Vergleich zu anderen Netztopologieen die deutlichsten Nachteile. Dies ist auch nicht weiter verwunderlich, da bei einem vollständig vermaschten Netzwerk jeder Netzteilnehmer mit jedem anderen Netzteilnehmer verbunden ist. Mit diesem hohen Verkabelungsaufwand gehen auch hohe Kosten einher. Die Stern-Topologie, die Baum-Topologie sowie die teilweise vermaschten Netze sind aus Kostensicht als etwa gleichwertig zu beurteilen. Bei Netzen in Stern-Topologie als auch bei Netzen in Baum-Topologie werden Verteilerknoten benötigt, um die Funktionsfähigkeit des Netzwerks zu gewährleisten. Teilweise vermaschte Netze kommen zwar ohne Verteilerknoten aus, dafür ist der Verkabelungsaufwand im Vergleich zu anderen Netztopologieen erhöht (ein Netzteilnehemr ist normalerweise mit mehreren anderen Netzteilnehmern verbunden). Netzteilnehmer in Netzwerken mit Ring-Topologie benötigen nur eine direkte Verbindung zum Vorgänger und zum Nachfolger. Damit ist der Verkabelungsaufwand gegenüber vermaschten Netzwerken deutlich reduziert und die Kosten sind geringer. Bei Netzwerken in Bus-Topologie werden alle Netzteilnehmer an eine gemeinsame Datenleitung angeschlossen. Dies verringert die Kosten im Vergleich zu anderen Netztopologieen enorm, da sich alle Netzteilnehmer das Übertragungsmedium teilen und dieses somit nicht mehrfach zur Verfügung gestellt werden muss. Aus Kostensicht ist also die Bus-Topologie für den Einsatz in einem medizinischen Netzwerk allen anderen Netztopologieen vorzuziehen.

Vollständig vermaschte Netze sind die zuverlässigsten Netzwerke. Auch wenn in einem vollständig vermaschten Netz eine oder mehrere Verbindungen unterbrochen sind, so gibt es immer einen Alternativweg über den zwei Netzteilnehmer miteinander kommunizieren können. Im Gegensatz dazu sind Netzwerke in Ring-Topologie relativ unzuverlässig im Vergleich zu anderen Topologieen. Wenn bei einem Netzwerk in Ring-Topologie zwei Netzteilnehmer ausfallen zerbricht der Ring in zwei Segmente und es ist keine Kommunikation mehr möglich. Unter Zuverlässigkeitsgesichtspunkten liegen die Stern- und die Baum-Topologie ebenfalls gleichauf. Fällt ein Verteilerknoten aus, so bedeutet das bei einem Netzwerk in Baum-Topologie, dass zumindest ein Teil des Netzwerkes funktionsunfähig ist. Bei einem Netzwerk in Stern-Topologie ist bei einem Ausfall des Verteilerknotens das ganze Netzwerk funktionsunfähig. Bei Netzwerken in Bus-Topologie ist ein Ausfall des Netzwerks nur durch einen Kabelbruch der gemeinsamen Datenleitung möglich. Da es sich bei Kabel um eine rein passive Komponente handelt, ist ein Kabelbruch deutlich unwahrscheinlicher als z.B. der Ausfall eines Verteilerknotens.

Die Skalierbarkeit, d.h. die möglichst einfache Erweiterbarkeit des Netzwerks durch neue Netzteilnehmer ist ein sehr wichtiger Gesichtspunkt bei der Auswahl einer geeigneten Netztopologie für ein medizinisches Netzwerk. Vermaschte Netzwerke lassen sich aufgrund der bereits mehrfach angesprochenen aufwändigen Verkabelung sehr schlecht skalieren. Netzwerke in Stern- bzw. Baum-Topologie lassen sich dagegen recht einfach erweitern. Ein neuer Netzteilnehmer muss dazu lediglich an einen Verteilerknoten angeschlossen werden. Netzwerke in Ring-Topologie lassen sich im Vergleich zu Netzwerken in Stern- oder Baum-Topologie schlechter skalieren, da bei Netzwerken in Ring-Topologie sichergestellt werden muss, dass bei hinzukommen eines neuen Netzteilnehmers die Ringstruktur nicht verletzt wird. D.h. der neue Netzteilnehmer benötigt einen Vorgänger und einen Nachfolger. Netzwerke in Bus-Topologie lassen sich am flexibelsten erweitern, hierzu genügt es einen neuen

Netzteilnehmer über ein T-Stück an einer beliebigen Stelle der Busleitung anzuschließen.

Die Performance, d.h. der Datendurchsatz, spielt bei der Auswahl einer geeigneten Netztopologie für ein medizinisches Netzwerk eine untergeordnete Rolle, da das medizinische Netzwerk im größten Teil der Zeit nur sporadisch Nachrichten verschickt. Netzwerke in Bus- und Ring-Topologie haben den geringsten Datendurchsatz der fünf grundlegenden Netztopologieen. Bei einem Bus kann während ein Netzteilnehmer Daten sendet kein anderer Netzteilnehmer schreibend auf den Bus zugreifen. D.h. ein zweiter sendewilliger Netzteilnehmer muss solange warten bis der Bus wieder frei ist bevor er Daten senden kann. Vermaschte Netzwerke und Netzwerke in Stern-Topologie sind sehr performant. Bei vermaschten Netzwerken liegt die hohe Performance hauptsächlich in redundanten Wegen.

Abschließend lässt sich sagen, dass Netzwerke in Bus-Topologie hauptsächlich wegen der sehr geringen Kosten für Verkabelung und Hardware und der zugleich hohen Skalierbarkeit und Zuverlässigkeit für den Einsatz in einem kabelgebundenen medizinischen Netzwerk am besten geeignet sind.

3.3. Vorstellung und Auswahl geeigneter Bussysteme

3.3.1. Überblick

In Abschnitt 3.2.7 wurde bereits herausgearbeitet, dass sich ein Bussystem für den Einsatz in einem medizinischen Netzwerk aufgrund der geringen Kosten und der hohen Skalierbarkeit am besten eignet.

In den Abschnitten 3.3.2 bis 3.3.5 werden einige Bussysteme vorgestellt, die in der Medizintechnik häufig verwendet werden bzw. bei vielen Herstellern bereits in Microcontrollern integriert sind. Zu diesen Bussystemen zählen der SPI-Bus, der I^2C-Bus, der CAN-Bus sowie ARCNET. Natürlich erhebt diese Auflistung keinen Anspruch auf Vollständigkeit, dies ist bei der Vielzahl an auf dem Markt existierender Bussysteme auch nicht möglich. Einen umfangreicheren Überblick über gängige Bussysteme liefert z.B. [Rei02] oder [Zim07].

In Abschnitt 3.3.6 werden die vorgestellten Bussysteme anhand wichter Kriterien (z.B. Kosten) miteinander verglichen und ein geeignetes Bussystem für den Einsatz in einem medizinischen Netzwerk ausgewählt.

3.3.2. Der SPI-Bus

Der SPI[4]-Bus ist ein von Motorola entwickeltes serielles synchrones[5] vollduplex[6] fähiges Bussystem, das hauptsächlich zur Verbindung digitaler Schaltungen auf Platinenebene nach dem Master-Slave-Prinzip verwendet wird. Bei vielen Herstellern von Microcontrollern ist der SPI-Bus bereits on-chip auf dem Microcontroller vorhanden.

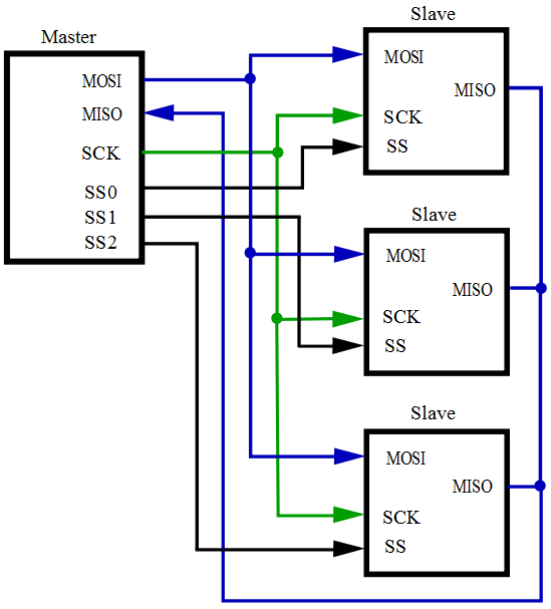

Abbildung 3.7.: Verbindung zwischen SPI-Master und SPI-Slaves [Wik09]

Die Abbildung 3.7 zeigt eine mögliche Verbindung zwischen einem Master und drei Slaves. Der SPI-Bus ist typischerweise ein 4-wire-Bus, das bedeutet, dass zur Busarbitrierung[7] und zur Kommunikation 4 Leitungen verwendet werden.

Die Kommunikation zwischen Master und Slave wird eröffnet, indem der Master die SS[8]-Leitung des betreffenden Slaves, mit dem er kommunizieren möchte auf einen logischen „0“-Pegel (SS-Pins sind normalerweise „active low“) zieht. Anschließend gibt der Master den *gemeinsamen* Takt auf die SCK[9]-Leitung. Der Master sendet Daten über die MOSI[10]-Leitung zum Slave, zeitgleich sendet der Slave Daten über die MISO[11]-Leitung zum Master.

[4] Serial Peripheral Interface
[5] Bei der synchronen Datenübertragung wird die Übertragung einzelner Bits zwischen Sender und Empfänger mit einem auf einer Taktleitung übertragenem Taktsignal zeitlich synchronisiert.
[6] Bei einer vollduplex fähigen Datenübertragung kann ein Netzteilnehmer gleichzeitig senden und empfangen.
[7] Unter Arbitrierung wird die möglichst gerechte Zuteilung des Übertragunsmediums an einen oder mehrere sendewillige Netzteilnehmer verstanden.
[8] Slave Select
[9] Serial Clock
[10] Master Out Slave In
[11] Master In Slave Out

Auf diese Weise können beliebig viele Datenbytes hintereinander zwischen Master und Slave übertragen werden. Die Kommunikation wird beendet, indem der Master den Takt stoppt und anschließend die SS-Leitung des betreffenden Slaves auf einen logischen „1“-Pegel zieht.

Für die Datenübertragung über die beiden Datenleitungen MOSI und MISO gibt es vier verschiedene Modi, die sich in Timing und im Ruhezustand der Taktleitung unterscheiden.

Abbildung 3.8 zeigt den Zusammenhang zwischen den Parametern CPOL[12], CPHA[13] und dem Timing auf den Datenleitungen MOSI und MISO.

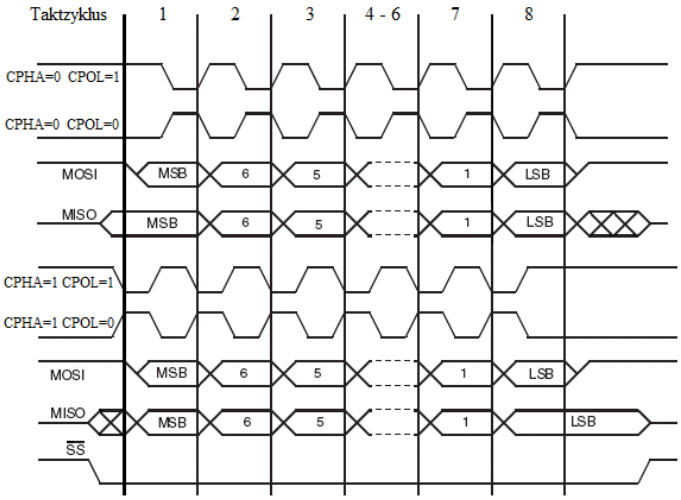

Abbildung 3.8.: Timingdiagramm für verschiedene CPOL/CPHA-Werte [Wik09]

Dabei beschreibt CPOL den Ruhezustand der Taktleitung. Bei CPOL=0 ist die Taktleitung im Ruhezutand logisch „0“, bei CPOL=1 ist sie logisch „1“.

CPHA legt fest, ob die angelegten Daten bei der steigenden oder bei der fallenden Taktflanke übernommen werden sollen. Bei CPHA=0 werden die Daten bei der steigenden Taktflanke übernommen, bei CPHA=1 nach der fallenden Taktflanke. Die Abbildung 3.9 ordnet den 4 SPI Modes jeweils eine mögliche Kombinationen von CPOL und CPHA zu.

3.3.3. Der I²C-Bus

Der I²C-Bus[14] ist ein in den frühen 1980er Jahren von Philips Semiconductors (heute NXP Semiconductors) entwickeltes serielles synchrones Multi-Master-Bussystem zur Verbindung

[12] Clock Polarity
[13] Clock Phase
[14] Inter-Integrated-Circuit-Bus. Bei einigen Halbleiterherstellern wird der I²C-Bus aus lizenzrechtlichen Gründen TWI-Bus genannt (Two-Wire-Interface-Bus).

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Abbildung 3.9.: Mögliche Kombinationen für CPOL und CPHA [Wik09]

von digitalen Schaltkreisen auf Platinenebene. Ebenso wie der SPI-Bus ist der I²C-Bus bei vielen Herstellern von Microcontrollern bereits on-chip auf dem Microcontroller vorhanden.

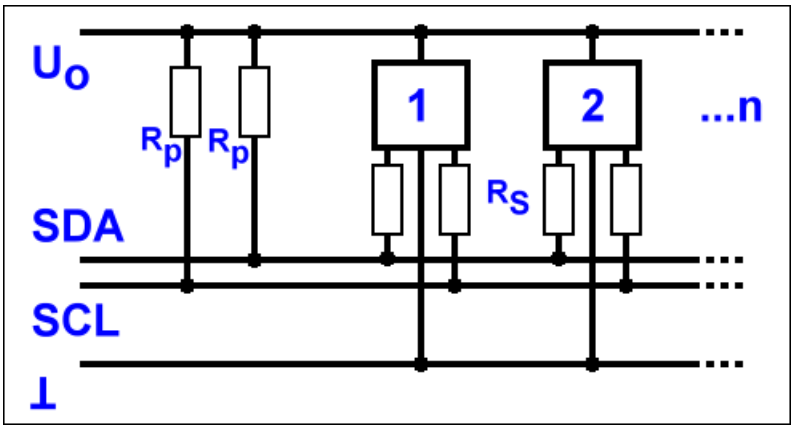

Abbildung 3.10.: Physikalische Struktur eines I²C-Busses [Wik09]

Abbildung 3.10 zeigt die physikalische Struktur des I²C-Busses mit zwei Busteilnehmern. Der I²C-Bus ist ein klassischer 2-wire-Bus, d.h. zur Kommunikation zwischen zwei Busteilnehmern werden nur zwei Leitungen benötigt. Da es sich um ein synchrones Bussystem handelt wird auf der SCL[15]-Leitung der Sendetakt übertragen, auf der SDA[16]-Leitung werden die zu sendenden Daten übertragen.

Das Arbitrierungsverfahren beim I²C-Bus ist CSMA/CR[17], d.h. ein sendewilliger Busteilnehmer überwacht die beiden Busleitungen und beginnt, wenn sich der Bus im Ruhezustand befindet (die SCL- und SDA-Leitung führen durch die beiden Pull-Up-Widerstände R_p einen HIGH-Pegel), mit der Datenübertragung (CSMA). Es kann passieren, dass gleichzeitig zwei oder mehr Busteilnehmer schreibend auf den Bus zugreifen möchten. In diesem Fall sorgt die Open-Drain-Struktur (siehe Abschnitt 2.3.2) der Ausgangstreiber jedes Busteilnehmers, dass es zu keinen Kurzschlüssen auf den Busleitungen kommt. Im Konfliktfall

[15] Serial Clock
[16] Serial Data
[17] Carrier Sense Multiple Access/Collision Resolution

„gewinnt“ derjenige Busteilnehmer den Arbitrierungsprozess, der zuerst ein dominates Bit (LOW-Pegel) auf die Datenleitung gibt. Dieser Busteilnehmer darf die Datenübertragung bis zum Ende durchführen, während sich alle anderen Busteilnehmer zurückziehen und warten müssen, bis der Bus wieder frei ist, ehe sie einen erneuten Sendeversuch starten (CR).

Die Serienwiderstände R_s in Abbildung 3.11 sind optional und dienen nur dem Überspannungsschutz der Eingangsstufen der Busteilnehmer.

Abbildung 3.11 zeigt den grundlegenden Verlauf eines Datentransfers auf dem I²C-Bus.

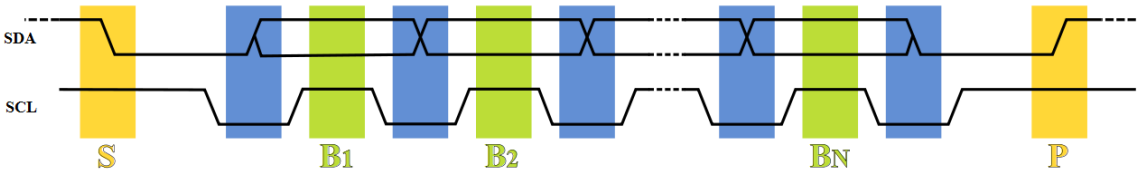

Abbildung 3.11.: Timingdiagramm eines I²C-Datentransfers [Wik09]

Die Kommunikation wird eröffnet, indem ein Master die sog. „START Condition“ sendet, d.h. der Master zieht die SDA-Leitung auf einen LOW-Pegel während die SCL-Leitung auf ihrem HIGH-Pegel verbleibt (mit „S“ bezeichnter orangefarbener Bereich in Abbildung 3.11). Darauf hin gibt der Master den Takt auf die SCL-Leitung. Nun legt der Sender während jeder LOW-Periode der SCL-Leitung (hellblaue Bereiche in Abbildung 3.11) ein Datenbit auf die SDA-Leitung. Der Empfänger tastet dieses gesendete Datenbit in der darauf folgenden HIGH-Periode der SCL-Leitung ab (hellgrüne Bereiche in Abbildung 3.11). Wenn alle zu sendenden Datenbytes gesendet wurden beendet der Master die Kommunikation durch das Senden einer „STOP Condition“. Dabei wird zunächst die SCL-Leitung auf den HIGH-Pegel gebracht anschließend wechselt die SDA-Leitung vom LOW-Pegel zum HIGH-Pegel. Der I²C-Bus befindet sich nun wieder im Ruhezustand und es kann eine erneute Kommunikation beginnen. Mit Ausnahme der „START Condition“ und der „STOP Condition“ darf die SDA-Leitung ihren Pegel nur während einer LOW-Periode der SCL-Leitung ändern.

Beim I²C-Bus wird jeder Busteilnehmer durch eine mindestens sieben Bit breite eindeutige Adresse identifiziert (einige neuere I²C-Bausteine haben eine zehn Bit breite Adresse).

Abbildung 3.12 zeigt das Kommunikationsprotokoll eines I²C-Datentransfers.

Möchte ein Master mit einem Slave Daten austauschen so eröffnet er die Kommunikation indem er eine „START Condition“ sendet und das Bustiming aus Abildung 3.11 einhält. Darauf hin sendet der Master die sieben Bit Adresse des Slaves mit dem er kommunizieren möchte, beginnend mit dem höchstwertigsten Bit (MSB[18]). Das achte Bit ist das „Read-Bit“. Wenn der Master Daten *an* den Slave senden möchte, setzt er das Read-Bit auf null. Möchte der Master dagegen Daten *vom* Slave empfangen, so setzt er dieses Bit auf eins.

[18] Most Significant Bit

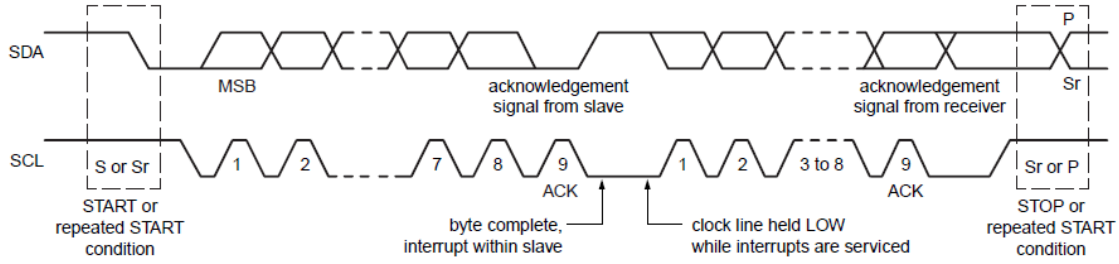

Abbildung 3.12.: Kommunikationsprotokoll eines I²C-Datentransfers [NXP07]

Das neunte Bit ist das sog. „Acknowledge-Bit“. Der adressierte Slave sendet im neunten Bit einen LOW-Pegel, um dem Master anzuzeigen dass der Slave empfangs- bzw. sendebereit ist (je nach dem, ob der Master als Read-Bit null oder eins gesendet hat). Nun legt der Sender acht Datenbits mit dem höchstwertigsten beginnend auf die SDA-Leitung. Im neunten Bit bestätigt der Empfänger den Erhalt der Daten mit einem LOW-Pegel. Dieses Verfahren wird so lange durchgeführt, bis der Sender keine Daten mehr zum Versenden hat und beendet die Kommunikation mit der „STOP Condition“. Darauf hin ist der I²C-Bus wieder im Ruhezustand und ein Master kann eine weitere Kommunikation mit einem Slave eröffnen.

Für weitere Informationen zum I²C-Bus siehe [NXP07], die offizielle Dokumentation von NXP Semiconductors.

3.3.4. Der CAN-Bus

Der CAN[19]-Bus ist ein 1983 von der Robert Bosch GmbH entwickeltes serielles asynchrones[20] Multi-Master-Bussystem. Er wurde ursprünglich für die Steuergerätekommunikation in Kraftfahrzeugen entwickelt wird aber mittlerweile aufgrund seiner hohen Robustheit gegenüber elektromagnetischen Störungen auch in weiten Bereichen in der Automatisierungstechnik sowie in verstärktem Maße auch in der Medizintechnik eingesetzt. Aufgrund der hohen Verfügbarkeit an Komponenten (die im Vergleich zu anderen Feldbussystemen[21] günstig sind) wird sich der CAN-Bus auch in den nächsten Jahren, vor allem auch in der Medizintechnik, immer größerer Beliebtheit erfreuen. Mittlerweile wurde der CAN-Bus unter ISO[22] 11898 bzw. SAE[23] J1939 standardisiert.

Abbildung 3.13 zeigt die physikalische Struktur des CAN-Busses mit drei Busteilnehmern.

[19] Controller Area Network
[20] Bei der asynchronen Datenübertragung werden Datenbits auf einer Datenleitung ohne zusätzliche Taktleitung übertragen. Dies setzt voraus, dass dem Empfänger entweder der Sendetakt bekannt ist oder dass der Sendetakt aus den gesendeten Bits zurückgewonnen werden kann.
[21] Ein Feldbus ist ein industrielles Kommunikationssystem, das eine Vielzahl von Feldgeräten wie Sensoren und Aktoren miteinander verbindet.
[22] International Organization for Standardization
[23] Society of Automotive Engineers

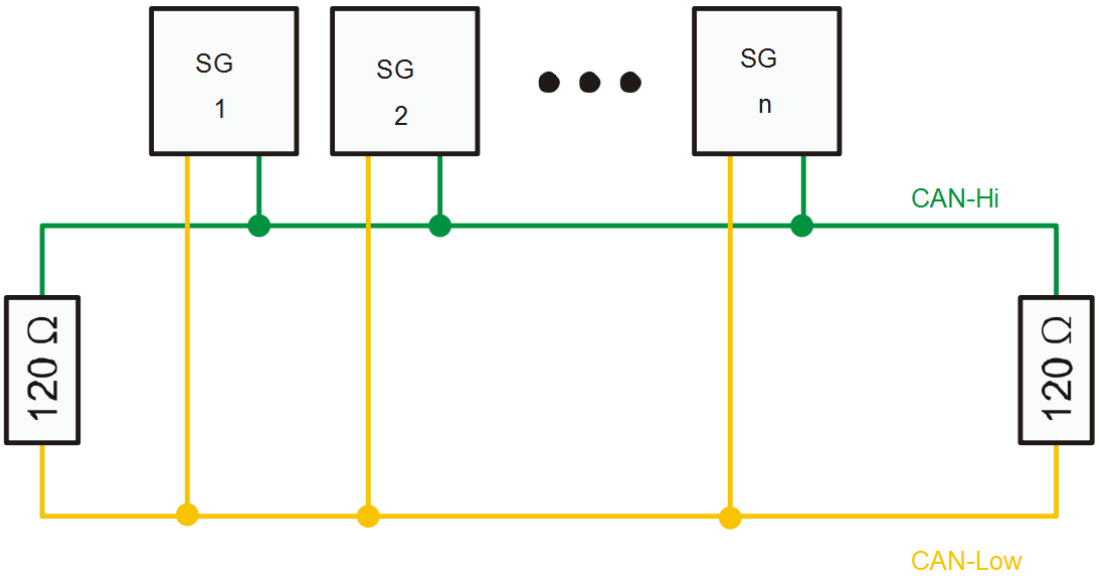

Abbildung 3.13.: Physikalische Struktur eines CAN-Busses [Wik09]

Der CAN-Bus ist ein 2-wire-Bus mit vollständig differenzieller Signalübertragung (siehe Abschnitt 2.2.3.2). Alle Busteilnehmer werden an die gemeinsamen Datenleitungen CANH[24] und CANL[25] angeschlossen. Der CAN-Bus arbeitet asynchron, d.h. es wird zur Datenübertragung keine gemeinsame Taktleitung benötigt. Als Übertragungsmedium wird ein verdrilltes doppeladriges Kupferkabel (Twisted-Pair-Kabel) verwendet, das an beiden Busenden mit dem Wellenwiderstand des Übertragungsmediums (ca. 120 Ohm bei Twisted-Pair-Kabeln) abgeschlossen wird, um Signalreflexionen zu minimieren (siehe Abschnitt 2.4).

Als Arbitrierungsverfahren wird beim CAN-Bus, ebenso wie beim I²C-Bus, CSMA/CR[26] verwendet. Versuchen zwei oder mehr Busteilnehmer gleichzeitig schreibend auf das Übertragungsmedium zuzugreifen, so gewinnt derjenige Busteilnehmer den Arbitrierungsprozess, der zuerst ein dominantes Bit sendet. Alle anderen Busteilnehmer, die ein rezessives Bit gesendet haben beenden ihren Schreibzugriff auf den Bus und warten bis das Übertragungsmedium wieder frei geworden ist, bevor sie einen erneuten Sendeversuch unternehmen. Das durch CSMA/CR geforderte dominant-rezessive Busverhalten wird durch eine Open-Drain-Endstufe im CAN-Transceiver[27] erreicht.

Abbildung 3.14 zeigt die prinzipielle Struktur eines CAN-Transceivers.

Soll auf dem CAN-Bus eine logische Eins gesendet werden, so sind beide Transistoren T1 und T2 geöffnet. Die beiden Datenleitungen CANH und CANL führen denselben Span-

[24] CAN High
[25] CAN Low
[26] manchmnal auch als CSMA/BA (Carrier Sense Multiple Access/Bitwise Arbitration) bezeichnet
[27] Ein CAN-Transceiver stellt die physikalische Verbindung zum Übertragungsmedium her.

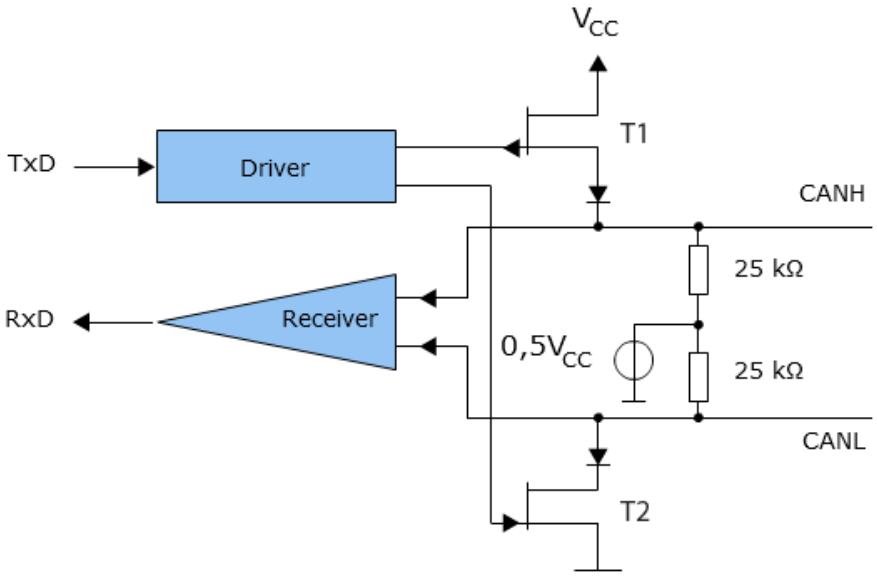

Abbildung 3.14.: Prinzipielle Struktur eines CAN-Transceivers [Vec09]

nungspegel, nämlich $0,5 \cdot V_{cc}$. Die Differenz zwischen CANH und CANL beträgt in diesem Zustand 0 Volt. Dieser Zustand wird *rezessiver* Zustand genannt. Ihm wird der logische Wert Eins zugeordnet.

Soll auf dem CAN-Bus eine logische Null gesendet werden, so sind beide Transistoren T1 und T2 geschlossen. Der Spannungspegel der CANH-Leitung ist dann V_{cc}, der Spanungspegel der CANL-Leitung ist GND (Masse). Die Differenz zwischen CANH und CANL beträgt in diesem Zustand V_{cc}. Dieser Zustand wird *dominanter* Zustand genannt. Ihm wird der logische Wert Null zugeordnet.

Abbildung 3.15 fasst die beiden obigen Absätze in grafischer Form zusammen.

Die Transistoren T1 und T2 sind zur selben Zeit immer entweder beide geöffnet oder beide geschlossen. Es kommt niemals vor, dass ein Transistor geöffnet ist, während der andere Transistor geschlossen ist. Wie in Abbildung 3.15 dargestellt ist, werden die logischen Pegel Null und Eins durch unterschiedliche Spannungsdifferenzen zwischen der CANH- und der CANL-Leitung dargestellt. Gemäß [Int03] wird eine Spannungsdifferenz zwischen CANH und CANL kleiner als 1,0 Volt als rezessiven Pegel (logisch Eins) interpretiert. Eine Spannungsdifferenz zwischen CANH und CANL von mehr als 1,5 Volt wird dagegen als dominanten Pegel (logisch Null) interpretiert.

Im Folgenden soll nun noch auf das beim CAN-Bus verwendete Übertragungsprotokoll eingegangen werden. Abbildung 3.16 zeigt den Aufbau eines sog. „Base Frames“.

An dieser Stelle kann aufgrund der Komplexität des Übertragungsprotokolls nur auf die wichtigsten Elemente des „Base Frames“ eingegangen werden, eine vollständige Beschrei-

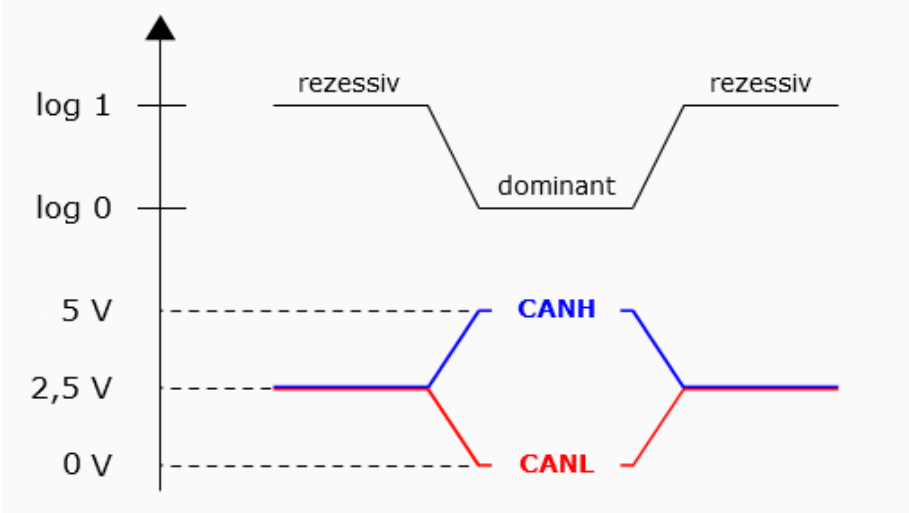

Abbildung 3.15.: Zuordnung von logischen zu physikalischen Buspegeln [Vec09]

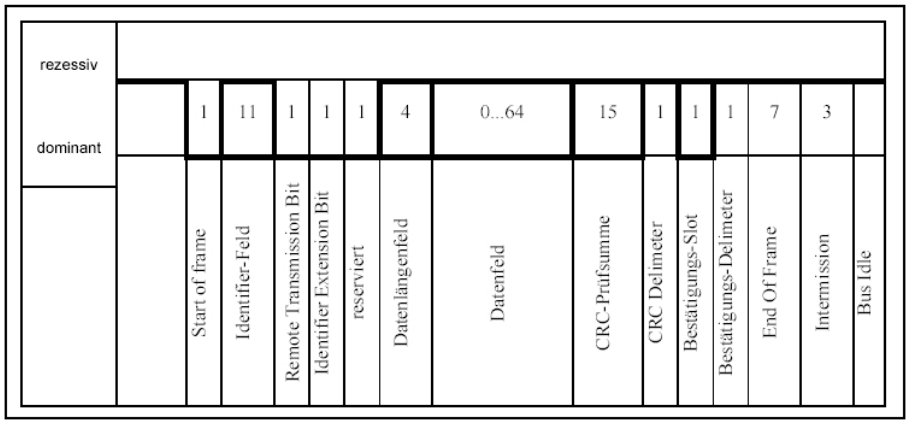

Abbildung 3.16.: Base Frame einer Übertragung auf dem CAN-Bus [Wik09]

bung des „Base Frames“ befindet sich jedoch in [Ets02] oder [Law00].

Die Kommunikation wird eröffnet, indem der Sender ein dominantes Bit auf die Datenleitungen gibt, den sog. „Start Of Frame (SOF)“.

Im darauf folgenden Identifier-Feld wird die 11 Bit breite „Message ID“ übertragen. Der CAN-Bus arbeitet Nachrichtenorientiert, d.h. jede gesendete Nachricht wird prinzipiell von jedem am Bus angeschlossenen Busteilnehmer empfangen (Broadcast). Jeder Empfänger entscheidet dann anhand der „Message ID“, ob er die empfangene Nachricht verarbeitet oder nicht.

Das Datenlängenfeld gibt die Anzahl der im Datenfeld übertragenen Nutzdatenbytes an. Es können pro „Base Frame“ maximal acht Nutzdatenbytes übertragen werden.

Im CRC[28]-Prüfsummenfeld wird eine 15 Bit breite Prüfsumme zur Wahrung der Datenintegrität übertragen. Diese Prüfsummenberechnung geschieht beim CAN-Bus vollständig hardwarebasiert, d.h. es ist kein Softwarealgorithmus nötig, um die CRC-Prüfsumme zu berechnen. Dies spart Rechenzeit.

Nach sieben rezessiven „End Of Frame (EOF)“ Bits und mindestens 3 weiteren rezessiven „Intermission“ Bits ist der CAN-Bus wieder im Ruhezustand und es kann eine weitere Datenübertragung beginnen.

3.3.5. ARCNET

ARCNET[29] wurde 1976 vom US-amerikanischen Unternehmen Datapoint, ursprünglich als Vernetzungstechnologie für LANs,[30] entwickelt. Es konnte sich in diesem Bereich aber nie durchsetzen, da es bereits in den frühen 1980er Jahren durch das deutlich leistungsstärkere Ethernet vollständig verdrängt wurde. Heute ist ARCNET ein überaus beliebtes serielles asynchrones Multi-Master-Feldbussystem bei der Vernetzung zeitkritischer Systeme in der Automatisierungstechnik sowie bei der sicherheitsrelevanten Datenkommunikation in der Medizintechnik.

Abbildung 3.17 zeigt die physikalische Struktur eines ARCNET-Busses.

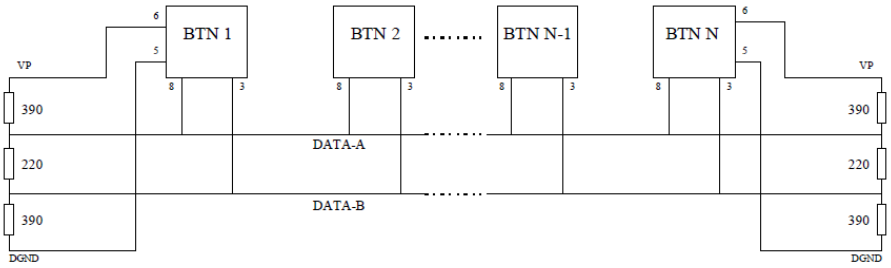

Abbildung 3.17.: Physikalische Struktur eines ARCNET-Busses [ARC09]

Der ARCNET-Bus ist, ebenso wie der CAN-Bus, ein 2-wire-Bus mit vollständig differenzieller Signalübertragung (siehe Abschnitt 2.2.3.2). Alle Busteilnehmer werden an die gemeinsamen Datenleitungen DATA-A und DATA-B angeschlossen. Aufgrund der asynchronen Datenübertragung wird bei ARCNET keine Taktleitung benötigt. Als Übertragungsmedium kann ein verdrilltes doppeladriges Kupferkabel (Twisted-Pair-Kabel) verwendet werden, das an beiden Busenden mit dem Wellenwiderstand des Übertragungsmediums abgeschlossen wird. Es sind aber auch andere Übertragungsmedien wie etwa Koaxialkabel oder Lichtwellenleiter zugelassen.

ARCNET verwendet ein deterministisches Arbitrierungsverfahren names „Token Passing“.

[28] Cyclic Redundancy Check
[29] Attached Resources Computer Network
[30] Local Area Network

Hierbei wird im Netzwerk eine Sendeberechtigung (das Token) von Busteilnehmer zu Busteilnehmer weitergereicht. Derjenige Busteilnehmer, der im Besitz des Tokens ist, darf mit einem beliebigen anderen Busteilnehmer kommunizieren. Bei ARCNET ist die maximale Datenlänge pro Kommunikationszyklus auf 508 Datenbytes beschränkt. Möchte der Sender mehr als 508 Datenbytes versenden, so *muss* er zunächst die Kommunikation nach dem 508ten Datenbyte abbrechen und das Token an den nächsten logischen Busteilnehmer weiterreichen. Der Sender kann, nachdem er das Token wieder von seinem logischen Vorgänger erhalten hat, mit der Datenübertragung fortfahren. „Token Passing“ ist ein gerechtes Arbitrierungsverfahren, denn jeder Busteilnehmer hat die gleiche Priorität und bekommt einmal pro Buszyklus[31] die Möglichkeit Daten zu versenden.

3.3.6. Auswahl eines geeigneten Bussystems für den Einsatz in einem medizinischen Netzwerk

Die Auswahl eines geeigneten Bussystems für den Einsatz in einem medizinischen Netzwerk hängt maßgeblich von den vier Faktoren Kosten, Multi-Master Fähigkeit, Störsicherheit und Reichweite ab.

In Abbildung 3.18 werden die in Abschnitt 3.3.2 bis 3.3.5 vorgestellten Bussysteme nun nach obigen Kriterien bewertet. Dem Kostenfaktor kommt bei der Bewertung eine besondere Bedeutung zu, da es erstrebenswert ist, ein möglichst günstiges medizinisches Netzwerk zu konzipieren.

	Kosten	Multi-Master fähig?	Störsicherheit	Reichweite
SPI-Bus	+	nein	O	-
I²C-Bus	++	ja	O	O
CAN-Bus	O	ja	++	++
ARCNET	--	ja	++	++

Abbildung 3.18.: Vergleich der beschriebenen Bussysteme

Unter Kostengesichtspunkten hat der SPI-Bus und der I²C-Bus deutliche Vorteile gegenüber ARCNET und leichte Vorteile gegenüber dem CAN-Bus. Der I²C-Bus, als auch der SPI-Bus ist bei fast allen Herstellern von Microcontrollern bereits on-chip integriert. Zumindest bei kurzen Bussen wird außer dem Microcontroller keine gesonderte Hardware benötigt, um das Bussystem betreiben zu können. Aufgrund der einfachen Verkabelung und der Tatsache, dass es sich beim I²C-Bus um einen 2-wire-Bus handelt, muss dem I²C-Bus im Vergleich zum SPI-Bus aus Kostensicht der Vorzug gegeben werden. Der CAN-Bus ist im Vergleich zu ARCNET aufgrund der hohen Komponentenverfügbarkeit deutlich günstiger aber im Vergleich zum SPI-Bus oder I²C-Bus teurer, da sowohl beim CAN-Bus als auch bei ARCNET externe Buscontroller[32] sowie externe Bustransceiver[33] für die Leitungsan-

[31] Der Buszyklus ist die Zeit, die ein Token braucht um jeden Busteilnehmer einmal zu durchlaufen.

[32] Ein Buscontroller bettet die zu sendenden Nutzdaten in ein geeignetes Übertragungsprotokoll ein und stellt sicher, dass Konflikte während der Arbitrierungsphase beim Buszugriff aufgelöst werden.

[33] Ein Bustransceiver nimmt die physikalische Impulsformung für das angekoppelte Übertragungsmedium vor. Zusätzlich liefert der Bustransceiver genügend Strom, um das Übertragungsmedium zu treiben.

kopplung benötigt werden.

Der SPI-Bus ist das einzige vorgestellte Bussystem, das nicht Multi-Master fähig ist. Dies lässt sich bereits an der physikalischen Struktur des SPI-Busses aus Abbildung 3.7 leicht erkennen. Beim SPI-Bus gibt es genau *einen* Master, der über eine Slave Select-Leitung genau *einem* Slave den Zugang zum Bus gewährt. Bei den restlichen drei vorgestellten Bussystemen stellt das Arbitrierungsverfahren sicher, dass es mehrere Busmaster geben kann, die zu unterschiedlichen Zeiten das Übertragungsmedium nutzen können. Beim I²C-Bus als auch beim CAN-Bus werden eventuelle Medienzugriffskonflikte durch das Arbitrierungsverfahren CSMA/CR aufgelöst. Bei ARCNET wird im Vorfeld durch Token Passing verhindert, dass es überhaupt zu Medienzugriffskonflikten kommt.

Die beiden technischen Faktoren Störsicherheit und Reichweite müssen beim CAN-Bus als auch bei ARCNET als etwa gleichwertig angesehen werden. Die Störsicherheit hängt maßgeblich vom verwendeten Signalübertragungsverfahren ab, hier wird beim CAN-Bus als auch bei ARCNET eine vollständig differenzielle Signalübertragung verwendet. Des weiteren verwendet sowohl der CAN-Bus als auch ARCNET ein hardwareseitiges Prüfsummenverfahren, das die Datenintegrität steigert und somit die Störsicherheit weiter erhöht. ARCNET und auch der CAN-Bus benötigen zur Busankopplung Treiberbausteine, die Ströme im dreistelligen Milliamperebereich liefern können, somit können auch lange Leitungen (mehrere hundert Meter) getrieben werden. Der I²C-Bus reagiert durch die allenfalls pseudo-symmetrische Signalübertragung anfälliger auf elektromagnetische Störungen als der CAN-Bus oder ARCNET. Des weiteren besitzt der I²C-Bus kein hardwareseitiges Prüfsummenverfahren um die Datenintegrität sicher zu stellen. Dies ist kein großer Nachteil gegenüber dem CAN-Bus oder ARCNET, da ein Prüfsummenverfahren leicht in Software nachgebildet werden kann (sofern auf dem Microcontroller genügend Rechenkapazität zur Verfügung steht). I²C-Hardware besitzt nur sehr schwache Treiberstufen (ca. 3 mA) mit denen bei 80 kBit/s nur ca. 15 Meter Leitung getrieben werden können. Auf dem Markt sind jedoch leistungsstärkere Treiberstufen erhältlich, mit denen dieses Problem umgangen werden kann.

Abschließend lässt sich sagen, dass unter rein technischen Gesichtspunkten dem CAN-Bus für den Einsatz in einem mobilen medizinischen Netzwerk aufgrund der hohen Störsicherheit sowie der großen Reichweite gegenüber dem I²C-Bus der Vorzug gegeben weden müsste. Jedoch wird bei der Konzeption des medizinischen Netzwerks sehr großen Wert auf eine wirtschaftlich tragbare Lösung gelegt, daher wird als zu verwendendes Bussystem der I²C-Bus ausgewählt. Zur Realisierung großer Reichweiten (mehr als 15 Meter) sind auch beim I²C-Bus externe Treiberbausteine nötig. Dies bedeutet zwar einen zusätzlichen Kostenaufwand, jedoch ist ein I²C basiertes Netzwerk mit Treiberbausteinen immernoch günstiger als ein CAN basiertes Netzwerk. Ein I²C-Bustreiber kostet auf dem Markt etwa 1,50 $ während die Kombination aus CAN-Controller und CAN-Transceiver etwa 3,00 $ bis 3,50 $ kostet.

3.4. Systemüberblick des medizinischen Netzwerks

Abbildung 3.19 zeigt einen schematischen Überblick eines intelligenten, mobilen medizinischen Ad-hoc Netzwerks, das aus zwei Teilsystemen besteht. Das erste Teilsystem ist das in dieser Arbeit konzipierte und entwickelte Sensor-/Aktor-Netzwerk (hellblauer Bereich in Abbildung 3.19). Das zweite Teilsystem ist der Anwendungsserver (hellgrauer Bereich in Abbildung 3.19). Die Aufgabe des Anwendungsservers ist, die gesammelten Daten des Sensor-/Aktor-Netzwerks in einer grafischen Benutzeroberfläche (GUI[34]) zu visualisieren und evtl. in einer Datenbank abzuspeichern. Des weiteren dient die grafische Oberfläche zur Konfiguration des Sensor-/Aktor-Netzwerks. Die Konzeption und Entwicklung des Anwendungsserver ist nicht Gegenstand dieser Arbeit.

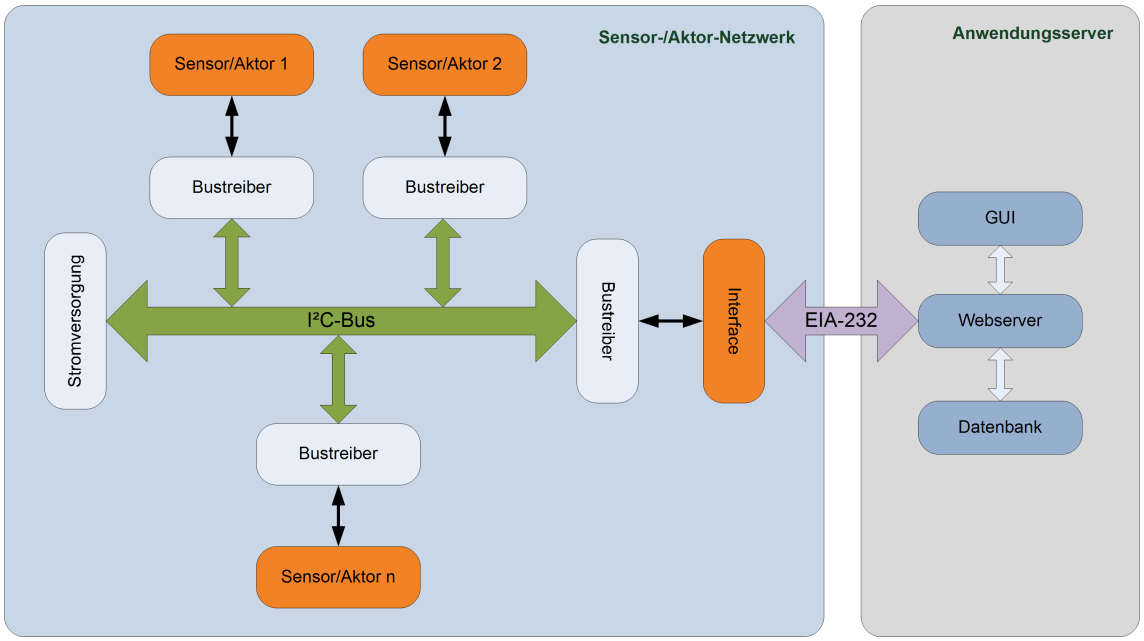

Abbildung 3.19.: Struktur eines medizinischen Ad-hoc-Netzwerks

Aufgrund der Ergebnisse aus den Abschniten 3.2.7 und 3.3.6 wird das Sensor-Aktor-Netzwerk in Bus-Topologie realisiert. Als Bussystem kommt der I²C-Bus zum Einsatz.

Die Busteilnehmer sind Sensoren/Aktoren und ein Interface (orange Blöcke in Abbildung 3.19). Alle Busteilnehmer können über das in Abschnitt 3.5 beschriebene Kommunikationsprotokoll in beliebiger Kombination miteinander kommunizieren. Der Interface-Knoten dient als Bridge zum Anwendungsserver, d.h. der Interface-Knoten transportiert Daten vom I²C-Bus zum Anwendungsserver und umgekehrt. Die Kommunikation zwischen Anwendungsserver und Interface-Knoten findet dabei über die EIA-232[35]-Schnittstelle statt. Die Kommunikation zwischen dem Interface-Knoten und einem beliebigen Sensor-/Aktor-Knoten findet über den I²C-Bus statt.

[34] Graphical User Interface
[35] auch als RS-232 bekannt

Die Hardware aller Sensor-/Aktor-Knoten ist identisch. Allein die Konfiguration eines Sensor-/Aktor-Knotens entscheidet darüber, ob sich der betreffende Knoten als Sensor oder als Aktor verhält. Die Konfiguration jedes Sensor-/Aktor-Knotens kann über die grafische Benutzeroberfläche des Anwendungsservers vorgenommen werden. Der Interface-Knoten leitet diese Konfigurationsnachricht an den entsprechenden Sensor-/Aktor-Knoten im I^2C-Netzwerk weiter. Eine ausführliche Hardwarebeschreibung der Sensor-/Aktor-Knoten befindet sich in Abschnitt 4.2.1. Die Hardware des Interface-Knotens wird in Abschnitt 4.2.2 erläutert. Die Beschreibung der Firmware der Sensor-/Aktor-Knoten sowie des Interface-Knotens befindet sich in Abschnitt 4.3.1 bzw. in Abschnitt 4.3.2.

Um eine Kommunikation über mehrere Meter auf dem I^2C-Bus zu ermöglichen müssen Bustreiber zur Leitungsankopplung an den I^2C-Bus verwendet werden. Diese Bustreiber können hohe Ströme liefern und somit auch sehr lange Busleitungen (über 100 Meter) treiben. Eine Beschreibung der Hardware befindet sich in Abschnitt 4.2.3.

Das gesamte Sensor-/Aktor-Netzwerk wird von einer zentralen Stelle aus mit Energie versorgt. Alle Sensor-/Aktor-Knoten sowie der Interface-Knoten beziehen ihre Energie von der zentralen Stromversorgung. Zusätzlich enthält die Stromversorgung zwei Pull-up-Widerstände für die beiden Busleitungen SDA und SCL, um im Ruhezustand einen logischen HIGH-Pegel auf den Busleitungen zu erzeugen. Die Hardwarebeschreibung der Stromversorgung befindet sich in Abschnitt 4.2.4.

3.5. Entwurf eines Kommunikationsprotokolls für den Einsatz in einem medizinischen Netzwerk

3.5.1. Überblick

In Abschnitt 3.5.2 wird zuerst auf die Anforderungen an ein Kommunikationsprotokoll für den Einsatz in einem medizinischen Netzwerk eingegangen. Des weiteren wird der Aufbau und die Struktur eines den Anforderungen genügenden Kommunikationsprotokolls erläutert.

Abschnitt 3.5.3 geht auf die Berechnung der Prüfsumme mit dem Fletcher-Algorithmus ein. Es gibt dutzende Algorithmen zur Berechnung von Prüfsummen, für den Einsatz in einem medizinischen Netzwerk wurde der Fletcher-Algorithmus aufgrund seiner besonders Resourcen schonenden Strukur ausgewählt. Die Prüfsummenberechnung ist ein zentraler Bestandteil des Kommunikationsprotokolls, um die Datenintegrität zu sichern.

In Abschnitt 3.5.4 werden die definierten Nachrichtenklassen und Nachrichtentypen erläutert.

3.5.2. Anforderungen und Aufbau eines Kommunikationsprotokolls für ein medizinisches Netzwerk

An das Kommunikationsprotokoll werden für den Einsatz in einem medizinischen Netzwerk folgende Anforderungen gestellt:

1. Das Kommunikationsprotokoll soll modular aufgebaut sein. D.h. das Kommunikationsprotokoll soll einfach erweitert werden können ohne dabei die Kompatibilität zu einer früheren Version des Kommunikationsprotokolls zu gefährden.

2. Das Kommunikationsprotokoll soll effizient sein. Das bedeutet, dass das Kommunikationsprotokoll in der Lage sein muss Nutzdaten variabler Länge aufzunehmen. Könnte es nur Nutzdaten fester Länge aufnehmen, so wäre die Effizienz des Kommunikationsprotokolls beim Verschicken von nur wenig Nutzdaten sehr schlecht.

3. Das Kommunikationsprotokoll soll Mechanismen zur Sicherung der Datenintegrität bereit stellen. Insbesondere ist hier die Berechnung von Prüfsummen gemeint, um eine hohe Datenintegrität zu gewährleisten.

Abbildung 3.20 zeigt den Aufbau eines Kommunikationsprotokolls, das obige Anforderungen erfüllt.

Abbildung 3.20.: Kommunikationsprotokoll für den Einsatz in einem medizinischen Netzwerk

Im Folgenden werden die einzelnen Felder des Datenframes aus Abbildung 3.20 erläutert.

Das Feld SYNC Das Feld *Synchronization* ist ein 2-Byte Feld. Es dient zur Kennzeichnung des Anfangs eines Datenframes. Die beiden Bytes sind in hexadezimaler Schreibweise immer **0x4D** und **0x42**. Als ASCII-Zeichen interpretiert lauten diese Zeichen **M** und **B**.

Das Feld SRC Das Feld *Source* ist ein 1-Byte Feld. SRC kennzeichnet die Adresse des Absenders des Datenframes. Möchte z.B. ein Sensor mit der Adresse **0x14** Daten an einen Aktor senden, so versendet der Sensor einen Datenframe in dem das Feld **SRC** den Wert **0x14** hat.

Das Feld DST Das Feld *Destination* ist ein 1-Byte Feld. DST kennzeichnet die Adresse des Empfängers des Datenframes. Möchte z.B. ein Sensor Daten an einen Aktor mit der Adresse **0x28** senden, so versendet der Sensor einen Datenframe in dem das Feld **DST** den Wert **0x28** hat.

Das Feld CID Das Feld *Class ID* ist ein 1-Byte Feld. CID kennzeichnet zu welcher Nachrichtenklasse der Datenframe gehört. Es werden z.B. alle Nachrichten, die der Konfiguration von Sensor-/Aktor-Knoten dienen, der Nachrichtenklasse „Konfigurationsnachrichten“ zugeordnet.

Das Feld MID Das Feld *Message ID* ist ein 1-Byte Feld. MID kennzeichnet den Nachrichtentyp. Ein Nachrichtentyp könnte beispielsweise „Pulsmesswert“ sein. Das Konzept aus CID und MID stellt den modularen Aufbau des Kommunikationsprotokolls sicher. Es erfüllt damit die Anforderungen aus Punkt 1 in Abschnitt 3.5.

Das Feld LOP Das Feld *Length of Payload* ist ein 1-Byte Feld. LOP wird als vorzeichenlose 8-Bit Zahl interpretiert und kennzeichnet die Länge des Nutzdatenfeldes PLO in Bytes. Im Falle von $LOP = 0$ fällt das Feld PLO weg, da in diesem Fall null Byte Nutzdaten übertragen werden.

Das Feld PLO Das Feld *Payload* ist ein Feld variabler Länge. PLO enthält die zu übertragenden Nutzdaten in Little Endian[36] Codierung. Die maximale Länge von PLO ist 255 Bytes. Die Länge von PLO wird durch das oben beschriebene Feld LOP festgelegt. Ist LOP gleich 0xFF oder dezimal geschrieben 255, so enthält das Nutzdatenfeld PLO 255 Bytes an Daten. Durch die variable Länge des Feldes PLO wird die Effizienz des Kommunikationsprotokolls sicher gestellt. Es erfüllt die Anforderungen aus Punkt 2 in Abschnitt 3.5.

Das Feld CHK Das Feld *Checksum* ist ein 2-Byte Feld. CHK enthält die Prüfsumme aus allen in Abbildung 3.20 grün hervorgehobenen Bytes. Die Prüfsumme wird mit dem 8-Bit Fletcher-Algorithmus (siehe Abschnitt 3.5.3, [Zwe90] und [IEE82]) ermittelt. Durch die Prüfsumme wird die Datenintegrität sicher gestellt. Das Kommunikationsprotokoll erfüllt somit auch die Anforderungen aus Punkt 3 in Abschnitt 3.5.

3.5.3. Prüfsummenberechnung mit dem Fletcher-Algorithmus

Der folgende in C geschriebene Algorithmus zeigt exemplarisch die Berechnung der 16-Bit Fletcher-Prüfsumme, die in jedem Datenframe im Feld CHK übertragen wird. Eine ausführlichere Beschreibung mit weiterführenden Beispielen befindet sich in [IEE82] oder [Zwe90].

```
1 void CalculateChecksum(const unsigned char *Buffer, unsigned char
      DataLength){
2
3   unsigned char CK_A=0;
4   unsigned char CK_B=0;
5
```

[36] Bei Multibyte Daten in Little Endian Codierung steht das niederwertigste Byte an der kleinsten Speicheradresse. Im Falle des hier beschriebenen Kommunikationsprotokolls wird das niederwertigste Byte zuerst übertragen.

```
6   int i=0;
7
8   //8-Bit Fletcher Algorithm
9   for (i=2; i<DataLength + 7;i++){
10
11    CK_A=CK_A + *(Buffer+i);
12    CK_B=CK_B + CK_A;
13
14  }
15
16  Checksum[0]=CK_A;
17  Checksum[1]=CK_B;
18
19 }
```

Listing 3.1: Prüfsummenberechnung mit dem Fletcher-Algorithmus

Der Fletcher-Algorithmus aus Listing 3.1 berechnet die Prüfsumme über den in Abbildung 3.20 hervorgehobenen grünen Bereich, d.h. die Prüfsummenberechnung beginnt mit dem Feld SRC (Source) und endet mit dem letzten Byte des Feldes PLO (Payload).

In den Zeilen 3 und 4 werden zunächst zwei vorzeichenlose 8-Bit Zahlen CK_A und CK_B definiert, die konkateniert[37] die 16-Bit Prüfsumme ergeben.

Das Kernstück des Fletcher-Algorithmus besteht aus den zwei Anweisungen in den Zeilen 11 und 12, die insgesamt über die for-Schleife $5 + LOP$ mal ausgeführt werden. Der neue Wert von CK_A ergibt sich aus dem alten Wert von CK_A plus dem Wert des Bytes, auf das der Zeiger Buffer zeigt. Der neue Wert von CK_B ergibt sich aus dem alten Wert von CK_B plus dem aktuellen Wert von CK_A. Die Berechnung der Prüfsumme ist beendet, wenn der Zeiger Buffer auf dem letzten Byte des Payloads PLO steht.

3.5.4. Die Nachrichtenklassen und Nachrichtentypen

3.5.4.1. Überblick

In den Abschnitten 3.5.4.2 bis 3.5.4.5 werden die vier Nachrichtenklassen EVN, RAW, CON und CFG mit ihren zugehörigen Nachrichtentypen beschrieben. Mit den Nachrichtenklassen EVN und RAW werden Ereignisse (z.B. Herzstillstand) bzw. Messwerte (z.B. Pulsmesswerte) übertragen. Die Nachrichtenklasse CON dient um Bestätigungsnachrichten zu versenden. Mit der Nachrichtenklasse CFG können Sensoren/Aktoren konfiguriert werden. Es ist z.B. möglich den Betriebsmodus eines Sensor-/Aktor-Knotens zu verändern oder den Analog-Digital-Wandler eines Sensor-Knotens zu aktivieren.

Ein ausführlicher Überblick über alle Nachrichtenklassen und Nachrichtentypen befindet sich in Anhang A.

[37] aneinander gereiht

An einigen Stellen in den Abschnitten 3.5.4.2 bis 3.5.4.5 wird die Kenntnis der Hardware des Sensor-/Aktor-Knotens vorausgesetzt. Eine ausführliche Beschreibung der Hardware des Sensor-/Aktor-Knotens befindet sich in Abschnitt 4.2.1.

Im Folgenden werden noch einige Vereinbarungen getroffen, die alle Nachrichtentenklassen und -typen betreffen.

Bezeichnung von Nachrichten

Alle im medizinischen Netzwerk verschickten Nachrichten setzen sich aus der Nachrichtenklasse und dem Nachrichtentyp zusammen. Die Bezeichnung einer Nachricht erfolgt durch Nennung der Nachrichtenklasse gefolgt von einem Bindestrich und anschließender Nennung des Nachrichtentyps. Eine Nachricht der Nachrichtenklasse EVN und des Nachrichtentyps GEV wird beispielsweise als EVN-GEV bezeichnet.

Acknowledgement/Not-Acknowledgement

Alle Nachrichten, ausgenommen Nachrichten der Nachrichtenklassen RAW und CON, werden vom Nachrichtenempfänger mit einem Acknowledge (CON-ACK) oder mit einem Not-Acknowledge (CON-NCK) bestätigt. Hat der Empfänger die Nachricht korrekt verarbeitet, so bestätigt er diese mit CON-ACK. Konnte der Empfänger die Nachricht nicht korrekt verarbeiten, so bestätigt er die empfangene Nachricht mit CON-NCK.

Datenformate

Im Feld PLO können mehrere Bytes an Nutzdaten übertragen werden. Die Nutzdaten haben ein bestimmtes Datenformat und werden wie bereits angesprochen Little Endian codiert übertragen.

Abbildung 3.21 gibt einen Überblick über die erlaubten Datenformate.

Kürzel	Typ	Größe (in Bytes)	Min...Max
U8	Unsigned Char	1	0...255
U16	Unsigned Short	2	0...65535
CH	ASCII	1	

Abbildung 3.21.: Erlaubte Datenformate im Payload-Feld

3.5.4.2. Die Nachrichtenklasse EVN (0x0A)

Mit der Nachrichtenklasse EVN *(Events)* werden Ereignisse übertragen. Diese Ereignisse können z.B. Herz- oder Atemstillstand sein oder noch genereller ein nicht näher spezifiziertes allgemeines Ereignis, im Folgenden kritischer Kreislaufzustand genannt.

EVN-GEV (0x0A 0x01)

Nachricht	**EVN-GEV (Events-General Event)**									
Beschreibung	**Diese Nachricht zeigt einen kritischen Kreislaufzustand an**									
Hinweise	Ein Sensor-/Aktor-Knoten, der sich im Sensor-Mode befindet kann durch drücken des Knopfes B1 einen kritischen Kreislaufzustand simulieren. An den über die Nachricht CFG-DES definierten Empfänger wird die Nachricht EVN-GEV übertragen.									
Nachrichtenstruktur	**SYNC**		**SRC**	**DST**	**CID**	**MID**	**LOP**	**PLO**	**CHK**	
	0x4D	0x42	Quelladr.	Zieladr.	0x0A	0x01	0	siehe Payloadstruktur	CK_A	CK_B
Payloadstruktur	Byte Offset	Datenformat	Beschreibung							
			Diese Nachricht hat keinen Payload							

Abbildung 3.22.: Nachrichtenstruktur der Nachricht EVN-GEV

Beispiel Ein Sensor S1 mit der Adresse **0x14** sendet eine Nachricht vom Typ EVN-GEV an einen Aktor A1 mit der Adresse **0x28**. In Abbildung 3.23 ist die Nachrichtenstruktur der gesendeten Nachricht dargestellt.

Nachrichtenstruktur	**SYNC**		**SRC**	**DST**	**CID**	**MID**	**LOP**	**CHK**	
	0x4D	0x42	0x14	0x28	0x0A	0x01	0x00	0x47	0x24

Abbildung 3.23.: Beispiel einer Nachricht vom Typ EVN-GEV

3.5.4.3. Die Nachrichtenklasse RAW (0x0B)

Mit der Nachrichtenklasse RAW *(Raw Measurements)* werden Messwerte eines aktivierten Analog-Digital-Wandlers übertragen. Der Analog-Digital-Wandler eines Sensor-Knotens kann über die Konfigurationsnachricht CFG-ADT aktiviert oder deaktiviert werden.

RAW-GDA (0x0B 0x01)

Nachricht	**RAW-GDA (Raw Measurements-General Data)**									
Beschreibung	**Diese Nachricht überträgt Messwerte eines Analog-Digital-Wandlers**									
Hinweise	Ein Sensor-/Aktor-Knoten, der sich im Sensor-Mode befindet und dessen Analog-Digital-Wandler aktiviert ist, sendet die Nachricht RAW-GDA periodisch mit der konfigurierten Abtastfrequenz des Analog-Digital-Wandlers.									
Nachrichtenstruktur	**SYNC**		**SRC**	**DST**	**CID**	**MID**	**LOP**	**PLO**	**CHK**	
	0x4D	0x42	Quelladr.	Zieladr.	0x0B	0x01	2	siehe Payloadstruktur	CK_A	CK_B
Payloadstruktur	Byte Offset	Datenformat	Beschreibung							
	0	U16	Vorzeichenloser AD-Messwert in Binärcodierung von 0x0000 bis 0x0FFF							

Abbildung 3.24.: Nachrichtenstruktur der Nachricht RAW-GDA

Beispiel Ein Sensor S1 mit der Adresse **0x14** sendet eine Nachricht vom Typ RAW-GDA an einen Aktor A1 mit der Adresse **0x28**. Der Messwert am Eingang J2 des Analog-Digital-Wandlers des Sensors S1 sei 1,25V. In hexadezimaler Schreibweise entspricht der Messwert **0x08 0x00**. Durch die Little Endian codierung des Messwertes wird daraus **0x00 0x08**. In Abbildung 3.25 ist die Nachrichtenstruktur der gesendeten Nachricht dargestellt.

Nachrichtenstruktur	**SYNC**		**SRC**	**DST**	**CID**	**MID**	**LOP**	**PLO**	**CHK**	
	0x4D	0x42	0x14	0x28	0x0B	0x01	0x02	0x00 0x08	0x52	0xC5

Abbildung 3.25.: Beispiel einer Nachricht vom Typ RAW-GDA

3.5.4.4. Die Nachrichtenklasse CON (0x0C)

Mit der Nachrichtenklasse CON *(Confirmations)* werden Bestätigungsnachrichten übertragen. Korrekt verarbeitete Nachrichten werden mit CON-ACK bestätigt. Nicht korrekt verarbeitete Nachrichten werden mit CON-NCK bestätigt.

CON-ACK (0x0C 0x01)

Nachricht	**CON-ACK (Confirmation-Acknowledgement)**									
Beschreibung	**Bestätigung einer korrekt verarbeiteten Nachricht**									
Hinweise										
Nachrichtenstruktur	**SYNC**		**SRC**	**DST**	**CID**	**MID**	**LOP**	**PLO**	**CHK**	
	0x4D	0x42	Quelladr.	Zieladr.	0x0C	0x01	2	siehe Payloadstruktur	CK_A	CK_B
Payloadstruktur	Byte Offset	Datenformat	Beschreibung							
	0	U8	Class ID der empfangenen Nachricht							
	1	U8	Message ID der empfangenen Nachricht							

Abbildung 3.26.: Nachrichtenstruktur der Nachricht CON-ACK

Beispiel Ein Sensor S1 mit der Adresse **0x14** sendet eine Nachricht vom Typ EVN-GEV (Class ID **0x0A** und Message ID **0x01**) an einen Aktor A1 mit der Adresse **0x28**. Der Aktor A1 hat die Nachricht korrekt empfangen und verarbeitet. Er bestätigt dem Sensor S1 den korrekten Empfang der Nachricht mit einer Bestätigungsnachricht vom Typ CON-ACK. In Abbildung 3.27 ist die Nachrichtenstruktur der Bestätigungsnachricht dargestellt.

Nachrichtenstruktur	**SYNC**		**SRC**	**DST**	**CID**	**MID**	**LOP**	**PLO**	**CHK**	
	0x4D	0x42	0x28	0x14	0x0C	0x01	0x02	0x0A 0x01	0x56	0xEB

Abbildung 3.27.: Beispiel einer Nachricht vom Typ CON-ACK

CON-NCK (0x0C 0x02)

Nachricht	**CON-NCK (Confirmation-Not-Acknowledgement)**									
Beschreibung	**Bestätigung einer nicht korrekt verarbeiteten Nachricht**									
Hinweise										
Nachrichtenstruktur	**SYNC**		**SRC**	**DST**	**CID**	**MID**	**LOP**	**PLO**	**CHK**	
	0x4D	0x42	Quelladr.	Zieladr.	0x0C	0x02	0	siehe Payloadstruktur	CK_A	CK_B
Payloadstruktur	Byte Offset	Datenformat	Beschreibung							
			Diese Nachricht hat keinen Payload							

Abbildung 3.28.: Nachrichtenstruktur der Nachricht CON-NCK

Beispiel Ein Sensor S1 mit der Adresse **0x14** sendet eine Nachricht vom Typ EVN-GEV (Class ID **0x0A** und Message ID **0x01**) an einen Aktor A1 mit der Adresse **0x28**. Der Aktor A1 hat die Nachricht zwar empfangen, konnte sie aber aufgrund eines internen Fehlers nicht korrekt verarbeiten. Er bestätigt dem Sensor S1 die Nachricht mit einer Bestätigungsnachricht vom Typ CON-NCK. Diese Nachricht teilt dem Sensor S1 mit, dass der Aktor A1 die Nachricht nicht korrekt verarbeitet hat. In Abbildung 3.29 ist die Nachrichtenstruktur der Bestätigungsnachricht dargestellt.

Nachrichtenstruktur	**SYNC**		**SRC**	**DST**	**CID**	**MID**	**LOP**	**CHK**	
	0x4D	0x42	0x28	0x14	0x0C	0x02	0x00	0x4A	0x40

Abbildung 3.29.: Beispiel einer Nachricht vom Typ CON-NCK

CON-ADB (0x0C 0x03)

Nachricht	**CON-ADB (Confirmation-Address Broadcast)**									
Beschreibung	**Diese Nachricht wird zum versenden des Address Broadcasts verwendet**									
Hinweise	Ein Sensor-/Aktor-Knoten, der sich im Undefined-Mode befindet sendet diese Nachricht periodisch alle 2000 ms an das Interface.									
Nachrichtenstruktur	**SYNC**		**SRC**	**DST**	**CID**	**MID**	**LOP**	**PLO**	**CHK**	
	0x4D	0x42	Quelladr.	Interface	0x0C	0x03	1	siehe Payloadstruktur	CK_A	CK_B
Payloadstruktur	Byte Offset	Datenformat	Beschreibung							
	0	U8	Eindeutige Adresse des Sensor-/Aktor-Knotens							

Abbildung 3.30.: Nachrichtenstruktur der Nachricht CON-ADB

Beispiel Ein Sensor-/Aktor-Knoten K1 mit der Adresse **0x14** sendet direkt nach dem Verbinden mit dem I^2C-Bus eine Konfigurationsanforderungsnachricht vom Typ CON-ADB an den Interface-Knoten I1 mit der Adresse **0xFE**. Über die Konfigurationsanforderungsnachricht teilt der neu hinzugekommene Sensor-/Aktor-Knoten dem An-

wendungsserver seine Bereitschaft mit. In Abbildung 3.31 ist die Nachrichtenstruktur der Konfigurationsanforderungsnachricht dargestellt.

Nachrichtenstruktur	SYNC		SRC	DST	CID	MID	LOP	PLO	CHK	
	0x4D	0x42	0x14	0xFE	0x0C	0x03	0x01	0x14	0x36	0xBD

Abbildung 3.31.: Beispiel einer Nachricht vom Typ CON-ADB

3.5.4.5. Die Nachrichtenklasse CFG (0x0D)

Mit der Nachrichtenklasse CFG *(Configuration)* werden Konfigurationsnachrichten übertragen. Konfigurationsnachrichten dienen der Konfiguration von Sensor-/Aktor-Knoten. Mit den Nachrichten der Nachrichtenklasse CFG lässt sich das prinzipielle Betriebsverhalten eines Sensor-/Aktor-Knotens verändern, d.h. es kann festgelegt werden ob sich ein Sensor-/Aktor-Knoten als Sensor oder als Aktor verhalten soll. Des weiteren ist es möglich die Peripherie der Sensor-/Aktor-Knoten zu konfigurieren (Analog-Digital-Wandler und Digital-Analog-Wandler).

CFG-MOD (0x0D 0x01)

Nachricht	**CFG-MOD (Configuration-Mode)**									
Beschreibung	**Konfiguriert den Betriebsmodus eines Sensor-/Aktor-Knotens**									
Hinweise	Wird diese Nachricht ohne Payload verschickt, d.h. LOP = 0, so antwortet der Empfänger mit einer CFG-MOD Nachricht, der die aktuelle Konfiguration des Busteilnehmers entnommen werden kann.									
Nachrichtenstruktur	SYNC		SRC	DST	CID	MID	LOP	PLO	CHK	
	0x4D	0x42	Interface	Zieladr.	0x0D	0x01	1	siehe Payloadstruktur	CK_A	CK_B
Payloadstruktur	Byte Offset	Datenformat	Beschreibung							
	0	CH	U = Undefined-Mode; S = Sensor-Mode; A = Actor-Mode							

Abbildung 3.32.: Nachrichtenstruktur der Nachricht CFG-MOD

Beispiel Der Interface-Knoten I1 mit der Adresse **0xFE** möchte einen Sensor-/Aktor-Knotens mit der Adresse **0x14** als Sensor konfigurieren. Dazu sendet der Interface-Knoten I1 eine Nachricht vom Typ CFG-MOD an den Sensor-/Aktor-Knoten. Um den Sensor-/Aktor-Knoten als Sensor zu konfigurieren sendet der Interface-Knoten im Feld PLO das ASCII-Zeichen **S**, in hexadezimaler Schreibweise **0x53**. In Abbildung 3.33 ist die Nachrichtenstruktur der Konfigurationsnachricht dargestellt.

Nachrichtenstruktur	SYNC		SRC	DST	CID	MID	LOP	PLO	CHK	
	0x4D	0x42	0xFE	0x14	0x0D	0x01	0x01	0x53	0x74	0xE4

Abbildung 3.33.: Beispiel einer Nachricht vom Typ CFG-MOD

CFG-DES (0x0D 0x02)

Nachricht	CFG-DES (Configuration-Destination)									
Beschreibung	Konfiguriert die Zieladresse von Nachrichten der Nachrichtenklassen EVN und RAW									
Hinweise	Wird diese Nachricht ohne Payload verschickt, d.h. LOP = 0, so antwortet der Empfänger mit einer CFG-DES Nachricht, der die aktuelle Konfiguration des Busteilnehmers entnommen werden kann.									
Nachrichtenstruktur	SYNC		SRC	DST	CID	MID	LOP	PLO	CHK	
	0x4D	0x42	Interface	Zieladr.	0x0D	0x02	1	siehe Payloadstruktur	CK_A	CK_B
Payloadstruktur	Byte Offset	Datenformat	Beschreibung							
	0	U8	Legt die Zieladresse von Nachrichten der Nachrichtenklassen EVN und RAW fest							

Abbildung 3.34.: Nachrichtenstruktur der Nachricht CFG-DES

Beispiel Der Interface-Knoten I1 mit der Adresse **0xFE** möchte die Nachrichtenzieladresse eines Sensor-Knotens mit der Adresse **0x14** konfigurieren. Die Nachrichtenzieladresse soll **0x28** sein. D.h. die Nachrichten EVN-GEV (kritischer Kreislaufzustand) und RAW-GDA (Analoge Messwerte) werden in Zukunft an den Busteilnehmer, der die Adresse **0x28** hat versendet. In Abbildung 3.35 ist die Nachrichtenstruktur der Konfigurationsnachricht dargestellt.

Nachrichtenstruktur	SYNC		SRC	DST	CID	MID	LOP	PLO	CHK	
	0x4D	0x42	0xFE	0x14	0x0D	0x02	0x01	0x28	0x4A	0xBC

Abbildung 3.35.: Beispiel einer Nachricht vom Typ CFG-DES

CFG-ADT (0x0D 0x03)

Nachricht	CFG-ADT (Configuration-Analog-/Digital-Transformer)									
Beschreibung	Konfiguriert den Analog-Digital-Wandler eines Sensor-/Aktor-Knotens									
Hinweise	Wird diese Nachricht ohne Payload verschickt, d.h. LOP = 0, so antwortet der Empfänger mit einer CFG-ADT Nachricht, der die aktuelle Konfiguration des Busteilnehmers entnommen werden kann.									
Nachrichtenstruktur	SYNC		SRC	DST	CID	MID	LOP	PLO	CHK	
	0x4D	0x42	Interface	Zieladr.	0x0D	0x03	2	siehe Payloadstruktur	CK_A	CK_B
Payloadstruktur	Byte Offset	Datenformat	Beschreibung							
	0	U16	Legt die Samplingrate des ADC in Hz fest. Wird die Samplingrate zu 0 Hz gesetzt, ist der ADC deaktiviert.							

Abbildung 3.36.: Nachrichtenstruktur der Nachricht CFG-ADT

Beispiel Der Interface-Knoten I1 mit der Adresse **0xFE** möchte den Analog-Digital-Wandler eines Sensor-Knotens mit der Adresse **0x14** konfigurieren. Die Abtastrate soll **75 Hz** sein, in hexadezimaler Schreibweise **0x00 0x4B**. Durch die Little Endian codierung

wird daraus **0x4B 0x00**. In Abbildung 3.37 ist die Nachrichtenstruktur der Konfigurationsnachricht dargestellt.

Nachrichtenstruktur	SYNC		SRC	DST	CID	MID	LOP	PLO	CHK	
	0x4D	0x42	0xFE	0x14	0x0D	0x03	0x02	0x4B 0x00	0x6F	0x53

Abbildung 3.37.: Beispiel einer Nachricht vom Typ CFG-ADT

CFG-DAT (0x0D 0x04)

Nachricht	CFG-DAT (Configuration-Digital-/Analog-Transformer)									
Beschreibung	Konfiguriert den Digital-Analog-Wandler eines Sensor-/Aktor-Knotens									
Hinweise	Wird diese Nachricht ohne Payload verschickt, d.h. LOP = 0, so antwortet der Empfänger mit einer CFG-DAT Nachricht, der die aktuelle Konfiguration des Busteilnehmers entnommen werden kann.									
Nachrichtenstruktur	SYNC		SRC	DST	CID	MID	LOP	PLO	CHK	
	0x4D	0x42	Interface	Zieladr.	0x0D	0x04	1	siehe Payloadstruktur	CK_A	CK_B
Payloadstruktur	Byte Offset	Datenformat	Beschreibung							
	0	CH	O = DAC aktiviert; X = DAC deaktiviert							

Abbildung 3.38.: Nachrichtenstruktur der Nachricht CFG-DAT

Beispiel Der Interface-Knoten I1 mit der Adresse **0xFE** möchte den Digital-Analog-Wandler eines Aktor-Knotens mit der Adresse **0x14** konfigurieren. Der Digital-Analog-Wandler soll aktiviert werden. Dazu sendet der Interface-Knoten im Feld PLO das ASCII-Zeichen **O**, in hexadezimaler Schreibweise **0x4F**. In Abbildung 3.39 ist die Nachrichtenstruktur der Konfigurationsnachricht dargestellt.

Nachrichtenstruktur	SYNC		SRC	DST	CID	MID	LOP	PLO	CHK	
	0x4D	0x42	0xFE	0x14	0x0D	0x04	0x01	0x4F	0x73	0xE9

Abbildung 3.39.: Beispiel einer Nachricht vom Typ CFG-DAT

4. Entwicklung des medizinischen Netzwerks

4.1. Überblick

In Abschnitt 4.2 werden alle Hardwarekomponenten erläutert, die für die Funktionsfähigkeit eines medizinischen Netzwerks nötig sind. Zu den erläuterten Hardwarekomponenten gehört der Sensor-/Aktor-Knoten, der Interface-Knoten, der Bustreiber-Adapter sowie die Stromversorgung des medizinischen Netzwerks.

Sensor-/Aktor-Knoten (siehe Abschnit 4.2.1) sind das Kernstück des medizinischen Netzwerks. Mit Sensor-/Aktor-Knoten ist es möglich Vitalparameter verletzter Personen zu erfassen. Außerdem sind Sensor-/Aktor-Knoten in der Lage bei Vorliegen eines kritischen Kreislaufzustandes Ereignisse auszulösen, die gegebenenfalls von einem weiteren Sensor-/Aktor-Knoten visualisiert werden können.

Der Interface-Knoten (siehe Abschnitt 4.2.2) dient als Bridge zwischen dem I^2C-Bus und dem Anwendungsserver. Die Aufgabe des Interface-Knotens ist es Daten vom I^2C-Bus zum Anwendungsserver zu übertragen und umgkehrt Daten vom Anwendungsserver zum I^2C-Bus zu übertragen.

Der Bustreiber-Adapter (siehe Abschnitt 4.2.3) ist nötig, um große Distanzen (mehrere hundert Meter) bei der Kommunikation auf dem I^2C-Bus zu überbrücken.

Das gesamte medizinische Netzwerk wird von einer zentralen Stelle aus mit Energie versorgt (siehe Abschnitt 4.2.4). Dies hat den Vorteil, dass nicht jeder Busteilnehmer separat mit Strom versorgt werden muss und spart somit Kosten.

In Abschnitt 4.3 wird anhand von Flussdiagrammen die Firmware erläutert, die auf dem Sensor-/Aktor-Knoten sowie auf dem Interface-Knoten Anwendung findet.

Aus Gründen der Übersichtlichkeit wird die Firmware des Sensor-/Aktor-Knotens (siehe Abschnitt 4.3.1) für jeden möglichen Betriebszustand des Sensor-/Aktor-Knotens separat betrachtet. Zu den möglichen Betriebszuständen zählen, wie in Abschnitt 4.2.1 beschrieben, der Undefined-Mode, der Sensor-Mode und der Actor-Mode.

Die Funktionsweise der Firmware des Interface-Knotens wird in Abschnitt 4.3.2 erläutert.

4.2. Entwicklung der Hardware

4.2.1. Der Sensor-/Aktor-Knoten

In Abbildung 4.1 ist die Hardware des Sensor-/Aktor-Knotens dargestellt.

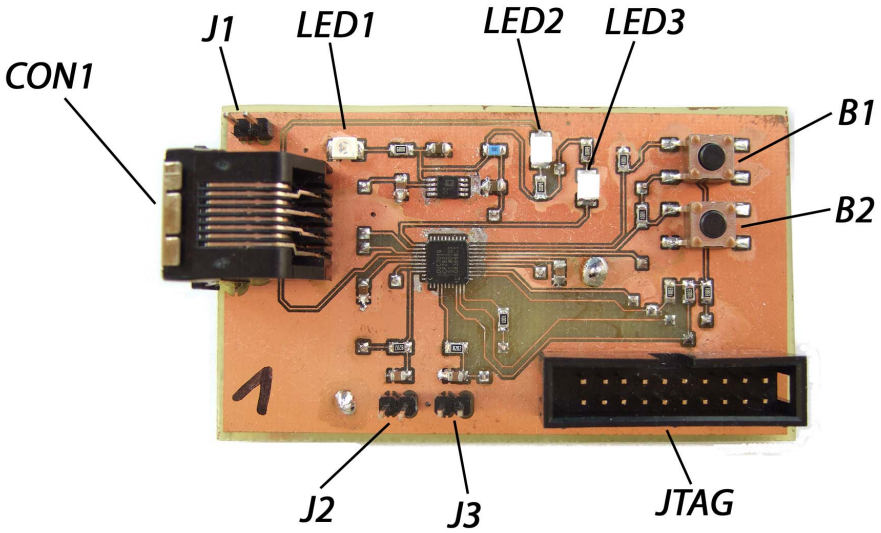

Abbildung 4.1.: Hardware des Sensor-/Aktor-Knotens

Als Prozessor dient ein ARM7TDMI®-Microcontroller von Analog Devices vom Typ ADuC7019 (siehe [Ana07]). Der Microcontroller kann über die JTAG[1]-Schnittstelle programmiert und auf Funktionsfähigkeit überprüft werden. Die Pinbelegung der 20-poligen JTAG-Schnittstelle ist nach IEEE[2] 1149.1 standardisiert und kann z.B. in [SEG09] nachgeschlagen werden. Zum Programmieren der Hardware kann ein belieber ARM7 kompatibler JTAG-Adapter verwendet werden (z.B. Analog Devices mIDAS-Link oder SEGGER J-Link).

Die RJ-45-Buchse CON1 dient zum Anschluss des Sensor-/Aktor-Knotens an den I²C-Bus. Als Verbindungskabel zwischen I²C-Bus und Sensor-/Aktor-Knoten wird ein handelsübliches Cat. 6 Patchkabel verwendet. In Abbildung 4.2 ist die Pinbelegung von CON1 dargestellt.

Über Pin 2 wird der Sensor-/Aktor-Knoten mit der nötigen Betriebsspannung VDD versorgt. Die Betriebsspannung wird von der in Abbildung 3.19 dargestellten Stromversorgung geliefert und hat einen Wert zwischen 5V und 15V DC (siehe auch Abschnitt 4.2.4). Eine korrekt angelegte Betriebsspannung wird durch das Leuchten der Leuchtdiode LED1 angezeigt. Alternativ zur Spannungsversorgung über die RJ-45-Buchse CON1 kann die

[1] Joint Test Action Group
[2] Institute of Electrical and Electronics Engineers

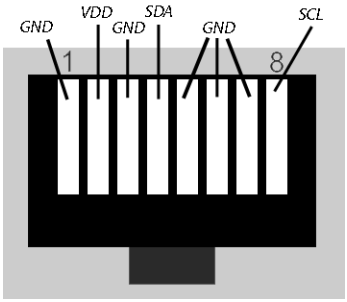

Abbildung 4.2.: Pinbelegung der RJ-45-Buchse

Spannungsversorgung des Sensor-/Aktor-Knotens auch über die 2-polige Stiftleiste J1 erfolgen. Der linke Stift der Stiftleiste wird in diesem Fall mit GND verbunden, der rechte Stift wird mit VDD verbunden.

Über Pin 4 und Pin 8 wird der Sensor-/Aktor-Knoten mit den beiden Busleitungen des I²C-Busses verbunden. Pin 4 ist dabei die Datenleitung SDA, Pin 8 ist die Taktleitung SCL.

Alle restlichen Pins, d.h. die Pins 1, 3, 5, 6 und 7 werden mit der Systemmasse GND verbunden.

Über die 2-polige Stiftleiste J2 können analoge Messwerte aufgenommen werden. Der linke Stift ist mit der Systemmasse GND verbunden, der rechte Stift wird mit dem zu messenden analogen Signal verbunden. Der Messbereich liegt zwischen 0V und 2,5V. J2 ist mit einem analogen Anti-Aliasing-Filter (RC-Glied) verbunden, um bei der Analog-Digital-Wandlung das Abtasttheorem einhalten zu können. Abbildung 4.3 zeigt den Aufbau des Anti-Aliasing-Filters.

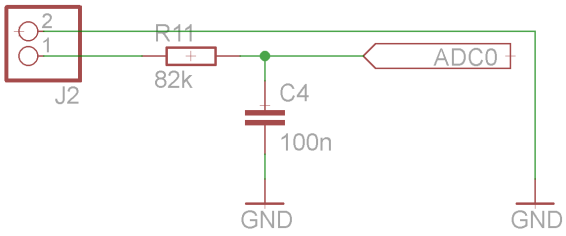

Abbildung 4.3.: Anti-Aliasing-Filter für die Einhaltung des Abtasttheorems

Nach [rnOD08] ergibt sich die Grenzfrequenz des in Abbildung 4.3 dargestellten Anti-Aliasing-Filters mit den Werten des Widerstandes $R11 = 82\ k\Omega$ und des Kondensators $C4 = 100\ nF$ zu

$$f_g = \frac{1}{2 \cdot \pi \cdot R \cdot C} = \frac{1}{2 \cdot \pi \cdot 82\ k\Omega \cdot 100\ nF} \approx 19,4\ Hz.$$

Über die Konfigurationsnachricht CFG-ADT (siehe Abschnitt 3.5.4.5 auf Seite 58) kann eine geeignete Abtastfrequenz eingestellt werden, um das Abtasttheorem einzuhalten.

Über die 2-polige Stiftleiste J3 können analoge Messwerte zur Visualisierung an einem Oszilloskop ausgegeben werden. Der linke Stift ist mit der Systemmasse GND verbunden, der rechte Stift ist mit dem analogen Ausgangssignal verbunden. Die Spannung des analogen Ausgangssignals liegt zwischen 0V und 2,5V. Die Stiftleiste J3 ist mit einem analogen Rekonstruktionsfilter verbunden, dessen Aufbau mit dem in Abbildung 4.3 dargestellten Anti-Aliasing-Filers identisch ist. Über die Konfigurationsnachricht CFG-DAT (siehe Abschnitt 3.5.4.5 auf Seite 59) kann der Digital-Analog-Wandler aktiviert bzw. deaktiviert werden.

Durch drücken des Knopfes B2 kann der Sensor-/Aktor-Knoten zurückgesetzt werden (Reset). Die Funktion des Knopfes B1 hängt vom aktuellen Betriebszustand des Sensor-/Aktor-Knotens ab. Die möglichen Betriebszustände werden im Folgenden erläutert.

Die beiden LEDs LED2 und LED3 zeigen den aktuellen Betriebszustand des Sensor-/Aktor-Knotens an. Abbildung 4.4 gibt einen Überblick über die möglichen Betriebszustände des Sensor-/Aktor-Knotens.

LED2	LED3	Betriebszustand
blinkt	blinkt	Undefined-Mode
leuchet	aus	Sensor-Mode; Analog-Digital-Wandler deaktiviert
leuchtet	blinkt	Sensor-Mode; Analog-Digital-Wandler aktiviert
blinkt	aus	Actor-Mode

Abbildung 4.4.: Mögliche Betriebszustände des Sensor-/Aktor-Knotens

Undefined-Mode

Direkt nach dem Anlegen der Betriebsspannung über die RJ-45-Buchse CON1 oder alternativ über die Stiftleiste J1 befindet sich der Sensor-/Aktor-Knoten im sog. *Undefined-Mode*.

In diesem Betriebszustand blinken die beiden LEDs LED2 und LED3 abwechselnd. Des weiteren Sendet der Sensor-/Aktor-Knoten im Abstand von 2000 Millisekunden eine Konfigurationsanforderungsnachricht CON-ADB (siehe Abschnitt 3.5.4.4 auf Seite 56) an den Interface-Knoten. In der Nachricht CON-ADB teilt der Sensor-/Aktor-Knoten seine eindeutige Adresse dem Interface-Knoten mit. Der Anwendungsserver erkennt den neu hinzugekommenen Busteilnehmer und kann dem Sensor-/Aktor-Knoten über die Konfigurationsnachricht CFG-MOD eine Funktion zuweisen.

Im Undefined-Mode besitzt der Knopf B1 keine Funktionalität. Der Analog-Digital-Wandler als auch der Digital-Analog-Wandler ist im Undefined-Mode deaktiviert. Über die Konfigu-

rationsnachricht CFG-MOD kann der Sensor-/Aktor-Knoten in einen beliebigen anderen Betriebszustand versetzt werden.

Sensor-Mode

Im *Sensor-Mode* ist der Sensor-/Aktor-Knoten in der Lage Ereignisse, die z.B. von einem kritischen Kreislaufzustand herrühren können, an einen definierten Empfänger zu übertragen. Im Sensor-Mode kann ein Ereignis (kritischer Kreislaufzustand) durch drücken des Knopfes B1 ausgelöst werden. Der Empfänger des Ereignisses wird über die Konfigurationsnachricht CFG-DES (siehe Abschnitt 3.5.4.5 auf Seite 58) bestimmt.

Zusätzlich zu Ereignissen können im Sensor-Mode auch Messwerte von einer analogen Quelle an einen definierten Empfänger übertragen werden. Das zu übertragende Analogsignal wird an die Stiftleiste J2 angelegt, zusätzlich muss der Analog-Digital-Wandler des Sensor-/Aktor-Knotens über die Konfigurationsnachricht CFG-ADT (siehe Abschnitt 3.5.4.5 auf Seite 58) konfiguriert werden. Ein aktivierter Analog-Digital-Wandler wird durch blinken der Leuchtdiode LED3 angezeigt.

Actor-Mode

Im *Actor-Mode* ist der Sensor-/Aktor-Knoten in der Lage auf Ereignisse, die von einem anderen Busteilnehmer übermittelt wurden, zu reagieren. Der Sensor-/Aktor-Knoten reagiert auf eine ankommende Nachricht vom Typ EVN-GEV (siehe Abschnitt 3.5.4.2 auf Seite 53) durch aktivieren der Leuchtdiode LED3. Das empfangene Ereignis kann durch drücken des Knopfes B1 gelöscht werden. Darauf hin wird auch die Leuchtdiode LED3 deaktiviert.

Zusätzlich zu Ereignissen können im Actor-Mode auch Messwerte, die von einem anderen Busteilnehmer übertragen wurden, empfangen werden. Um die Messwerte zu visualisieren kann ein Oszilloskop an die Stiftleiste J3 angeschlossen werden. Zuvor muss über die Konfigurationsnachricht CFG-DAT (siehe Abschnitt 3.5.4.5 auf Seite 59) der Digital-Analog-Wandler des Sensor-/Aktor-Knotens aktiviert werden.

4.2.2. Der Interface-Knoten

Der Interface-Knoten dient als Bridge zwischen dem I^2C-Bus und der EIA-232-Schnittstelle. D.h. Daten vom I^2C-Bus werden über die EIA-232-Schnittstelle an den Anwendungsserver weitergeleitet, Daten vom Anwendungsserver werden an den I^2C-Bus weitergeleitet (siehe Abbildung 3.19 auf Seite 47).

In Abbildung 4.5 ist die Hardware des Interface-Knotens dargestellt.

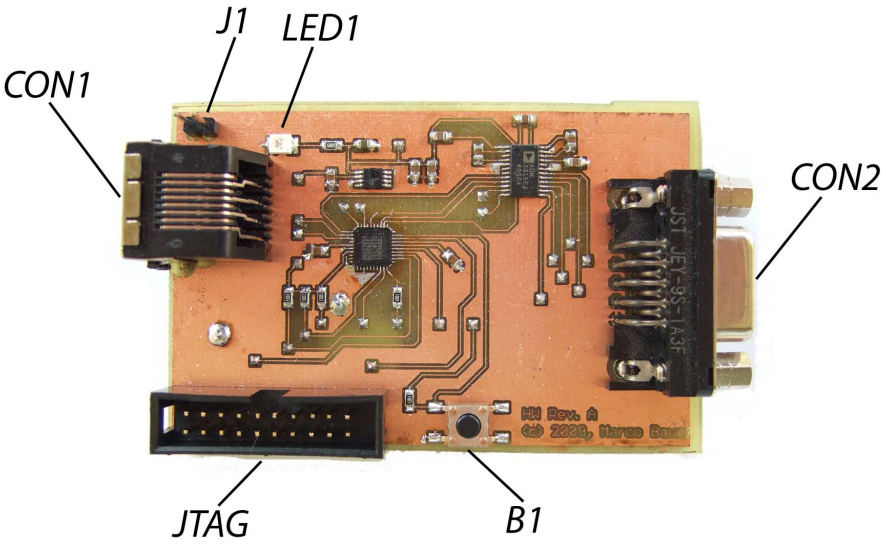

Abbildung 4.5.: Hardware des Interface-Knotens

Als Prozessor dient, wie auch beim Sensor-/Aktor-Knoten ein ARM7TDMI®-Microcontroller von Analog Devices vom Typ ADuC7019. Der Microcontroller kann über die JTAG-Schnittstelle programmiert und auf Funktionsfähigkeit überprüft werden.

Über die RJ-45-Buchse CON1 wird der Interface-Knoten mit der nötigen Betriebsspannung versorgt und mit den beiden I²C-Busleitungen SDA und SCL verbunden. Alternativ zur Stromversorgung über die RJ-45-Buchse CON1 kann auch die 2-polige Stiftleiste J1 verwendet werden. Die Pinbelegungen von CON1 bzw. J1 des Interface-Knotens entsprechen den Pinbelegungen von CON1 und J1 des Sensor-/Aktor-Knotens aus Abschnitt 4.2.1. Eine korrekt angelegte Betriebsspannung wird durch das Leuchten der Leuchtdiode LED1 angezeigt.

Über die 9-polige D-Sub-Buchse CON2 wird der Interface-Knoten mit einem PC verbunden, auf dem der Anwendungsserver läuft. Die Verbindung erfolgt über ein herkömmliches 1:1 belegtes Schnittstellenkabel. Das Frameformat der Datenübertragung ist 8N1, d.h. es werden zur Datenübertragung 8 Datenbits, kein Paritätsbit und 1 Stopbit verwendet. Die Übertragungsrate beträgt 115200 Bit pro Sekunde. Abbildung 4.6 zeigt die Pinbelegung der 9-poligen D-Sub-Buchse.

Für die Datenkommunikation zwischen Interface-Knoten und PC werden die Leitungen Tx[3] und Rx[4] verwendet. Auf der Tx-Leitung werden Daten vom Interface-Knoten zum PC gesendet, auf der Rx-Leitung werden Daten vom PC empfangen. Die Signalpegel werden bezüglich der Masse GND interpretiert.

[3] Transmit
[4] Receive

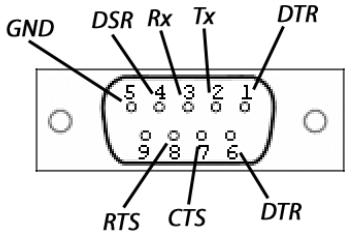

Abbildung 4.6.: Pinbelegung der 9-poligen D-Sub-Buchse

Über die DTR[5]-Leitung signalisiert der Interface-Knoten dem PC seine Betriebsbereitschaft. Über die DSR[6]-Leitung signalisiert der PC dem Interface-Knoten seine Betriebsbereitschaft.

Die beiden Leitungen RTS[7] und CTS[8] werden für die hardwareseitige Datenflusskontrolle[9] verwendet. Der Interface-Knoten signalisiert dem PC über die RTS-Leitung seine Empfangsbereitschaft. Der PC signalisiert dem Interface-Knoten über die CTS-Leitung seine Empfangsbereitschaft.

Durch drücken des Knopfes B1 kann der Interface-Knoten zurückgesetzt werden (Reset).

4.2.3. Der Bustreiber-Adapter

Der Bustreiber-Adapter wird zur physikalischen Leitungsankopplung an den I²C-Bus benötigt. Er ist in der Lage Ströme im dreistelligen Milliamperebereich zu treiben, somit kann ein Bus mit einer Ausdehnung von mehreren hundert Metern realisiert werden.

In Abbildung 4.7 ist die Hardware des Bustreiber-Adapters dargestellt.

Über den RJ-45-Stecker PLG1 wird der Bustreiber-Adapter mit einem Busteilnehmer (entweder ein Sensor-/Aktor-Knoten oder ein Interface-Knoten) verbunden. Die RJ-45-Buchse CON1 wird mit dem I²C-Bus verbunden. Als Verbindungskabel kommen handelsübliche Cat. 6 Patchkabel zum Einsatz.

Die Pinbelegung der RJ-45-Buchse CON1 ist mit der Pinbelegung der RJ-45-Buchse des Sensor-/Aktor-Knotens identisch (siehe Abbildung 4.2 auf Seite 63). Abbildung 4.8 zeigt die Pinbelegung des RJ-45-Steckers PLG1.

[5] Data Terminal Ready
[6] Data Set Ready
[7] Request To Send
[8] Clear To Send
[9] Unter Datenflusskontrolle werden Verfahren verstanden, mit denen die Datenübertragung von Endgeräten in einem Datennetz, die nicht synchron arbeiten, so gesteuert wird, dass eine möglichst kontinuierliche Datenübertragung erfolgen kann.

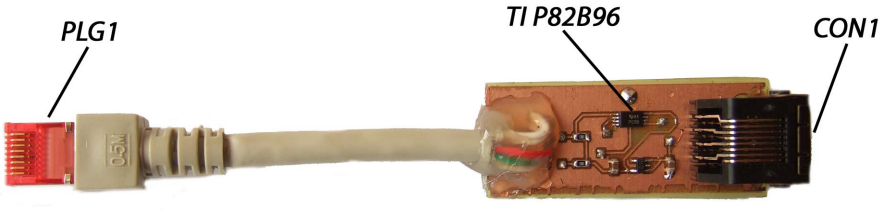

Abbildung 4.7.: Hardware des Bustreiber-Adapters

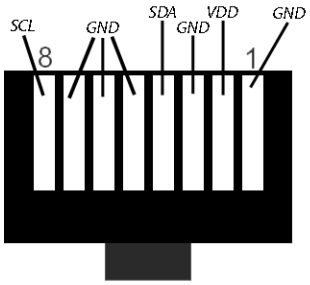

Abbildung 4.8.: Pinbelegung des RJ-45-Steckers

Als Bustreiber wird ein Baustein von Texas Instruments vom Typ P82B96 verwendet (siehe [Tex07]).

4.2.4. Die Stromversorgung des medizinischen Netzwerks

Über die in Abbildung 4.9 dargestellte Hardware wird das gesamte medizinische Netzwerk mit Energie versorgt.

Über die Hohlsteckerbuchse PWR1 wird ein Netzteil angeschlossen, das die nötige Versorgungsspannung zwischen 5V und 15V DC liefert. Der Stift der Hohlsteckerbuchse ist mit der Masse GND verbunden, der Mantel der Hohlsteckerbuchse ist mit der Versorgungsspannung VDD verbunden. Eine korrekt angelegte Versorgungsspannung wird durch das Leuchten der Leuchtdiode LED1 angezeigt.

Die Hardware aus Abbildung 4.9 verfügt außerdem noch über zwei Pull-up-Widerstände, über die die beiden I^2C-Busleitungen SDA und SCL mit einem logischen HIGH-Pegel (3 Volt) verbunden werden.

Die Pinbelegung der RJ-45-Buchse CON1 ist mit der Pinbelegung der RJ-45-Buchse des Sensor-/Aktor-Knotens identisch (siehe Abbildung 4.2 auf Seite 63).

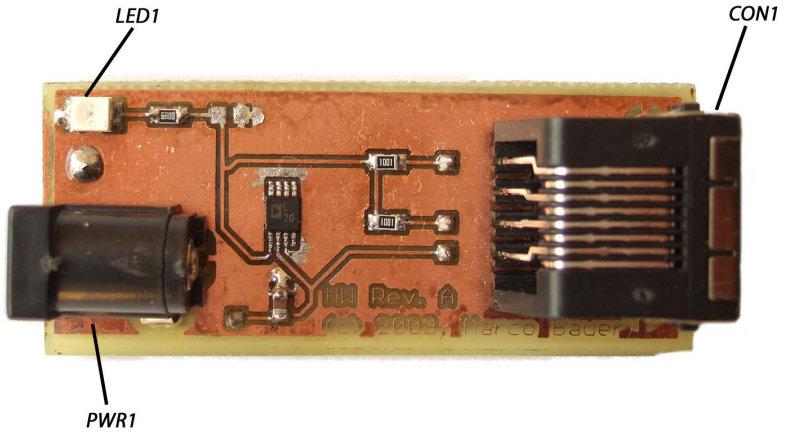

Abbildung 4.9.: Hardware der Stromversorgung des medizinischen Netzwerks

4.3. Entwicklung der Firmware

4.3.1. Die Firmware des Sensor-/Aktor-Knotens

Im Folgenden wird die Funktionsweise der Firmware des Sensor-/Aktor-Knotens beschrieben. Aus Gründen der Übersichtlichkeit erfolgt die Erläuterung der Firmware separat nach den möglichen Betriebszuständen des Sensor-/Aktor-Knotens.

Die Firmware des Sensor-/Aktor-Knotens besteht aus einem Hauptprogramm, das für die Initialisierung des Sensor-/Aktor-Knoten verantwortlich ist und mehreren Interrupt Service Routinen. Die Interrupt Service Routinen dienen zum Empfangen und Versenden von Nachrichten über den I^2C-Bus. Des weiteren werden empfangene Nachrichten von den Interrupt Service Routinen interpretiert (geparst) und evtl. geeignete Aktionen ausgelöst.

Die Funktionsweise der Firmware im Undefined-Mode

In Abbilding 4.10 ist der Programmablauf eines Sensor-/Aktor-Knotens, der sich im Undefined-Mode befindet, dargestellt.

Nach Anlegen der Betriebsspannung an den Sensor-/Aktor-Knoten beginnt die Programmausführung im Hauptprogramm. Als erstes wird eine Initialisierungsfunktion aufgerufen, die den I^2C-Bus sowie die vorhandenen Timer des ADuC7019 konfiguriert. Nach beendeter Initialisierung befindet sich der Sensor-/Aktor-Knoten in einer Endlosschleife, die jederzeit durch einen Interrupt unterbrochen werden kann.

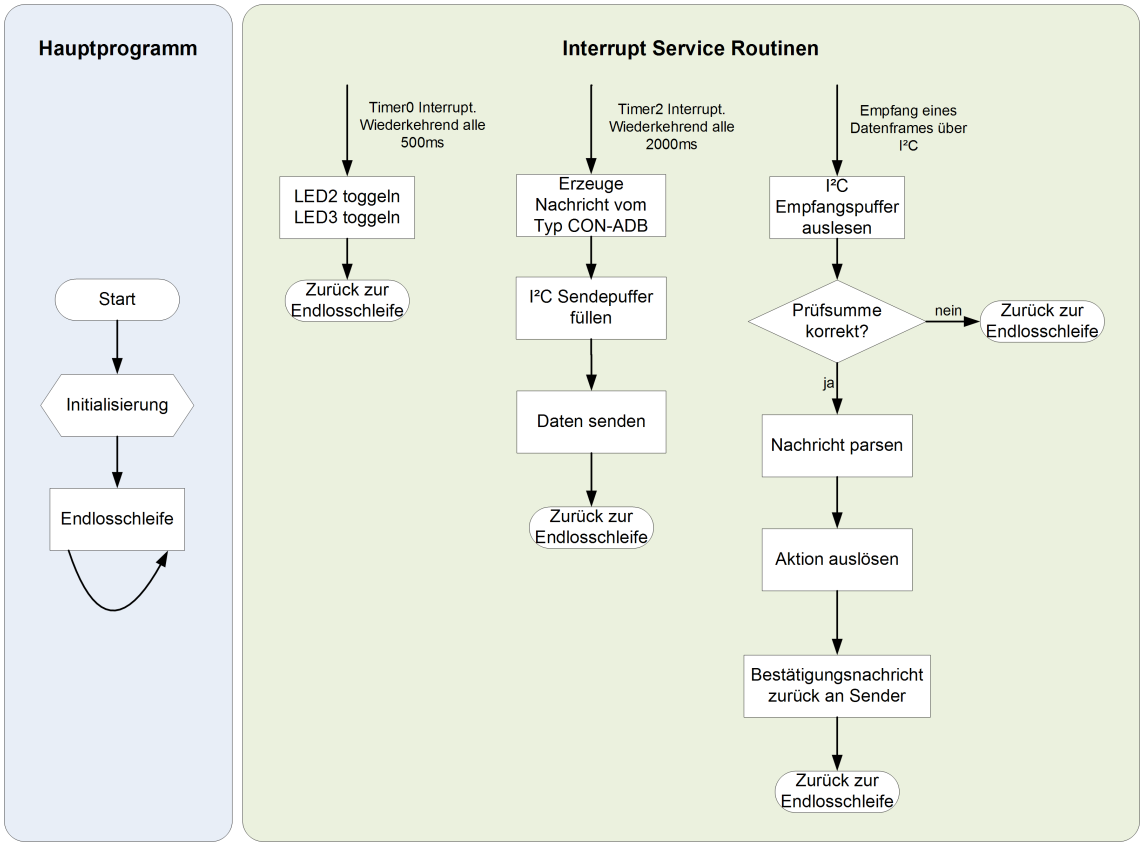

Abbildung 4.10.: Programmablauf im Undefined-Mode

Timer0 löst periodisch alle 500 Millisekunden einen Interrupt aus. In der darauf hin ausgeführten Interrupt Service Routine wird der Zustand der beiden Leuchtdioden LED2 und LED3 gewechselt. Dies dient zur visuellen Anzeige des Betriebszustandes des Sensor-/Aktor-Knotens (siehe Abbildung 4.4 auf Seite 64).

Timer2 löst periodisch alls 2000 Millisekunden einen Interrupt aus. In der darauf hin ausgeführten Interrupt Routine wird eine Konfigurationsanforderungsnachricht CON-ADB (siehe Abschnitt 3.5.4.4 auf Seite 56) an den Interface-Knoten gesendet.

Im Undefined-Mode kann jederzeit ein Datenframe über den I²C-Bus empfangen werden. Der Empfang eines Datenframes löst einen Interrupt aus. In der Interrupt Service Routine wird zunächst der I²C Empfangspuffer ausgelesen und anhand der übertragenen Prüfsumme ermittelt, ob der empfangene Datenframe korrekt empfangen wurde. Wurde der Datenframe nicht korrekt empfangen, d.h. stimmt die übertragene Prüfsumme nicht mit der berechneten Prüfsumme überein, so wird der empfangene Datenframe verworfen und die Programmausführung kehrt zur Endlosschleife zurück. Wurde der Datenframe korrekt empfangen, d.h. die übertragene Prüfsumme stimmt mit der errechneten Prüfsumme überein, so wird der empfangene Datenframe geparst. D.h. von der empfangenen Nachricht wird die Class ID und die Message ID ausgelesen, um den Typ der empfangenen Nachricht

festzustellen. In Abschnitt 3.5.4 sind alle möglichen Nachrichten definiert. Abhängig vom Nachrichtentyp der empfangenen Nachricht wird eine Aktion ausgelöst (z.B. Wechsel des Betriebsmodus vom Undefined-Mode in den Sensor-Mode). Zuletzt wird der Empfang der Nachricht dem Sender mit einer Bestätigungsnachricht aus der Nachrichtenklasse CON quittiert. Danach kehrt die Programmausführung zur Endlosschleife zurück.

Die Funktionsweise der Firmware im Sensor-Mode

In Abbilding 4.11 ist der Programmablauf eines Sensor-/Aktor-Knotens, der sich im Sensor-Mode befindet, dargestellt.

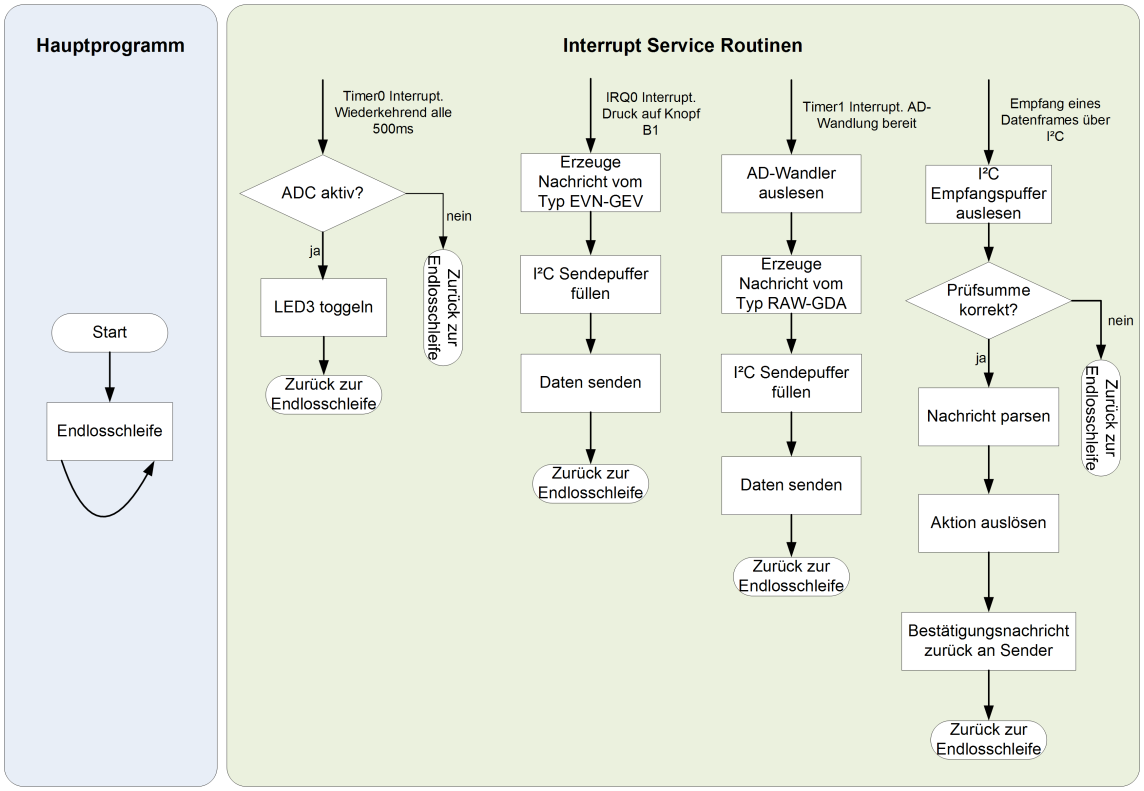

Abbildung 4.11.: Programmablauf im Sensor-Mode

Die Programmausführung beginnt in der Endlosschleife im Hauptprogramm. Die Endlosschleife kann jederzeit durch eine Interrupt Service Routine unterbrochen werden.

Timer0 löst periodisch alle 500 Millisekunden einen Interrupt aus. In der darauf hin ausgeführten Interrupt Service Routine wird überprüft, ob der Analog-Digital-Wandler des Senor-/Aktor-Knotens aktiv ist. Ist der Analog-Digital-Wandler nicht aktiviert, so kehrt die Programmausführung zur Endlosschleife zurück. Ist der Analog-Digital-Wandler aktiviert, so wird der Zustand der Leuchtdiode LED3 gewechselt. Dies dient zur visuellen

Anzeige des Betriebszustandes des Sensor-/Aktor-Knotens (siehe Abbildung 4.4 auf Seite 64).

Durch drücken des Knopfes B1 wird ein Interrupt ausgelöst, der einen kritischen Kreislaufzustand einer verletzten Person simuliert. In der ausgeführten Interrupt Service Routine wird zunächst eine Nachricht vom EVN-GEV erzeugt (siehe Abschnitt 3.5.4.2 auf Seite 53). Diese Nachricht wird anschließend dem I²C Sendepuffer übergeben und an den definierten Empfänger übertragen. Nach erfolgreichem Versenden der Daten über den I²C-Bus kehrt die Programmausführung zur Endlosschleife zurück.

Bei aktiviertem Analog-Digital-Wandler löst Timer2 periodisch, mit der über die Konfigurationsnachricht CFG-ADT (siehe Abschnitt 3.5.4.5 aus Seite 58) eingestellten Abtastfrequenz, einen Interrupt aus. In der darauf hin ausgeführten Interrupt Service Routine wird zunächst der aktuelle Wert des Analog-Digital-Wandlers ausgelesen und anschließend eine Nachricht vom Typ RAW-GDA (siehe Abschnitt 3.5.4.3 auf Seite 54) über den I²C-Bus an den definierten Empfänger versendet. Nach erfolgreichem Versenden der Daten über den I²C-Bus kehrt die Programmausführung zur Endlosschleife zurück.

Im Sensor-Mode kann jederzeit ein Datenframe über den I²C-Bus empfangen werden. Der Programmablauf ist mit dem bereits weiter oben im Absatz „Die Funktionsweise der Firmware im Undefined-Mode“ beschriebenen Programmablauf identisch.

Die Funktionsweise der Firmware im Actor-Mode

In Abbildung 4.12 ist der Programmablauf eines Sensor-/Aktor-Knotens, der sich im Actor-Mode befindet, dargestellt.

Die Programmausführung beginnt in der Endlosschleife im Hauptprogramm. Die Endlosschleife kann jederzeit durch eine Interrupt Service Routine unterbrochen werden.

Timer0 löst periodisch alle 500 Millisekunden einen Interrupt aus. In der darauf hin ausgeführten Interrupt Service Routine wird der Zustand der Leuchtdioden LED2 gewechselt. Dies dient zur visuellen Anzeige des Betriebszustandes des Sensor-/Aktor-Knotens (siehe Abbildung 4.4 auf Seite 64).

Empfängt der Sensor-/Aktor-Knoten eine Nachricht vom Typ EVN-GEV, die von einem anderen Sensor-/Aktor-Knoten versendet wurde, so wird die Leuchtdiode LED3 aktiviert. Die aktivierte Leuchtdiode visualisiert einen kritischen Kreislaufzustand eines Patienten. Durch Drücken des Knopfes B1, kann das empfangene Ereignis (der kritische Kreislaufzustand) gelöscht werden. Die Leuchtdiode LED3 wird deaktiviert.

Im Actor-Mode kann jederzeit ein Datenframe über den I²C-Bus empfangen werden. Der Programmablauf ist mit dem bereits weiter oben im Absatz „Die Funktionsweise der Firmware im Undefined-Mode“ beschriebenen Programmablauf identisch.

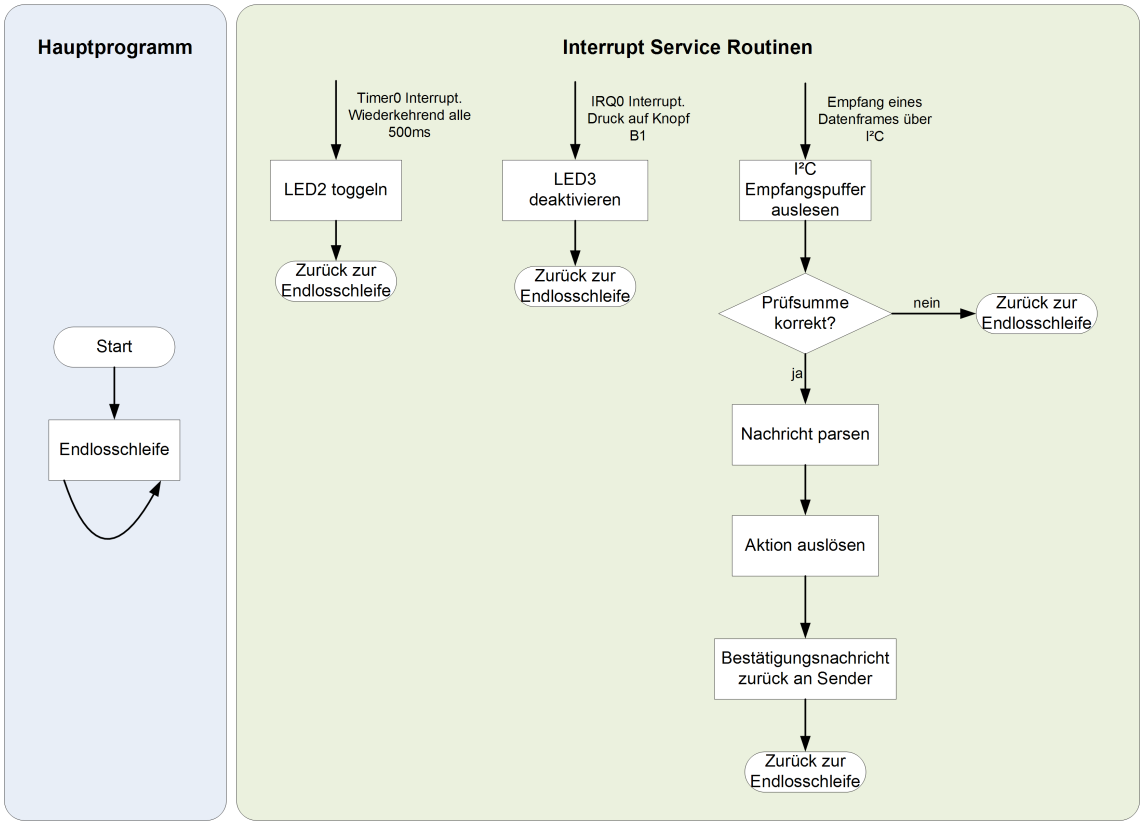

Abbildung 4.12.: Programmablauf im Actor-Mode

4.3.2. Die Firmware des Interface-Knotens

In Abbildung 4.13 ist der Programablauf des Interface-Knotens dargestellt.

Nach Anlegen der Betriebsspannung an den Interface-Knoten beginnt die Programmausführung im Hauptprogramm. Als erstes wird eine Initialisierungsfunktion aufgerufen, die den I²C-Bus als auch die UART-Schnittstelle konfiguriert. Nach beendeter Initialisierung befindet sich der Interface-Knoten in einer Endlosschleife, die jederzeit durch einen Interrupt unterbrochen werden kann.

Der Interface-Knoten dient als Bridge zwischen I²C-Bus und EIA-232-Schnittstelle, d.h. auf dem I²C-Bus ankommende Datenframes werden an die EIA-232-Schnittstelle weitergeleietet, ankommende Datenframes an der EIA-232-Schnittstelle werden an den I²C-Bus weitergeleitet. Der Empfang eines Datenframes löst einen Interrupt aus. Die Interrupt Service Routine ist für die Weiterleitung der Daten an die EIA-232-Schnittstelle bzw. an den I²C-Bus zuständig.

Bei Empfang eines Datenframes über die EIA-232-Schnittstelle wird zunächst der Empfangspuffer des UARTs ausgelesen und anhand der übertragenen Prüfsumme ermittelt, ob der empfangene Datenframe korrekt empfangen wurde. Wurde der Datenframe nicht

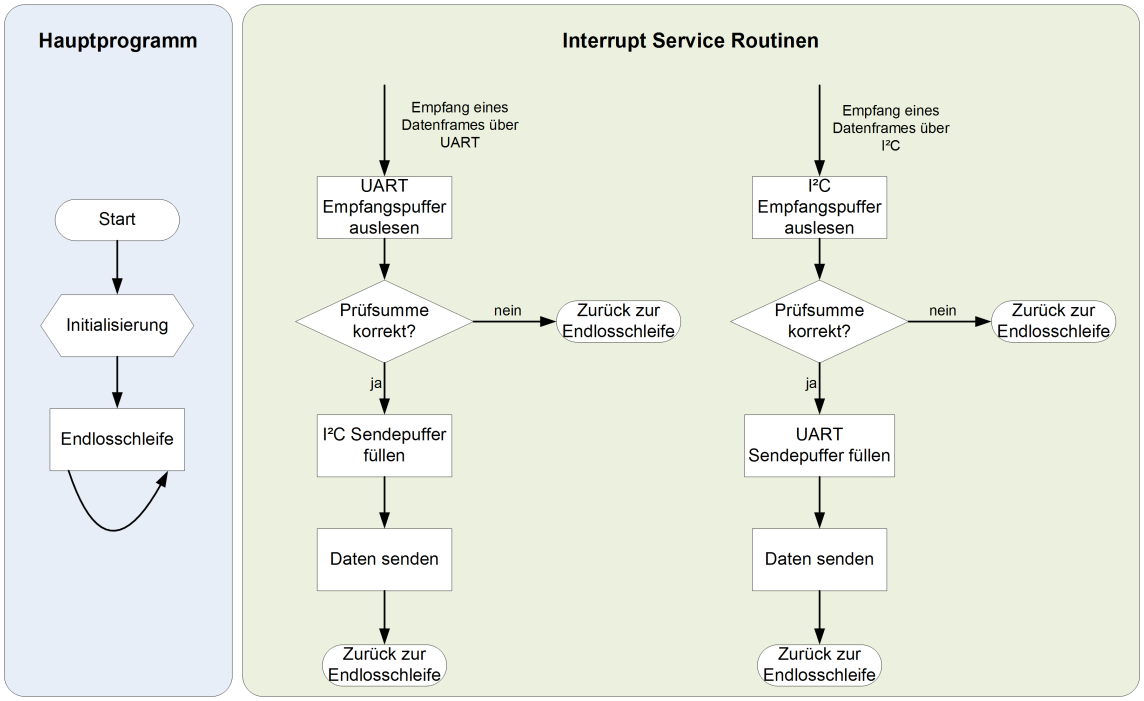

Abbildung 4.13.: Strukturdiagramm der Firmware des Interface-Knotens

korrekt empfangen, d.h. stimmt die übertragene Prüfsumme nicht mit der berechneten Prüfsumme überein, so wird der empfangene Datenframe verworfen und die Programmausführung kehrt zur Endlosschleife zurück. Wurde der Datenframe korrekt empfangen, d.h. die übertragene Prüfsumme stimmt mit der errechneten Prüfsumme überein, so wird der empfangene Datenframe an den I²C-Bus weitergeleitet. Nach erfolgreichem Versenden der Daten über den I²C-Bus kehrt die Programmausführung zur Endlosschleife zurück.

Der Empfang eines Datenframes über den I²C-Bus läuft in gleicher Weise wie der oben beschrieben Empfang eines Datenframes über die EIA-232-Schnittstelle ab.

5. Evaluation des medizinischen Netzwerks

5.1. Überblick

In Abschnitt 5.2 werden die Signalverläufe verschiedener I²C-Buslängen miteinander verglichen. Außerdem werden die Unterschiede zwischen einem I²C-Bus mit Bustreiber-Adapter (siehe Abschnitt 4.2.3) und einem I²C-Bus ohne Bustreiber-Adapter dargelegt.

In Abschnitt 5.3 wird das gesamte medizinische Netzwerk auf elektromagnetische Verträglichkeit überprüft. Dazu werden Messungen mit zwei in der Praxis häufig vorkommenden Strahlungsquellen durchgeführt.

In Abschnitt 5.4 wird schließlich der Stromverbrauch des medizinischen Netzwerks ermittelt. Der Stromverbrauch des gesamten medizinischen Netzwerks setzt sich aus dem Stromverbrauch der angeschlossenen Sensor-/Aktor-Knoten sowie aus dem Stromverbrauch des Interface-Knotens zusammen.

5.2. Vergleich verschiedener Buslängen

Vergleich verschiedener Buslängen ohne Bustreiber-Adapter

Zunächst wurden drei Messungen an einem ohne Bustreiber-Adapter betriebenen I²C-Bus durchgeführt. D.h. die Datenleitung SDA und die Taktleitung SCL des I²C-Busses sind direkt mit den entsprechenden Pins des Microcontrollers ADuC7019 verbunden.

In Abbildung 5.1 ist der Signalverlauf auf einem 1,5 Meter langen I²C-Bus dargestellt. Mit der ersten fallenden Flanke des Taktsignals SCL (gelbe Kurve in Abbildung 5.1) wird das erste Datenbit auf die Datenleitung SDA (magenta Kurve in Abbildung 5.1) gegeben. Mit der zweiten fallenden Flanke des Taktsignals SCL wird das zweite Datenbit auf die Datenleitung SDA gelegt usw. In Abbildung 5.1 ist die Übertragung von sieben Datenbits der Folge `1 0 1 0 1 0 0` dargestellt. Die Bitdauer ist circa $12,5\ \mu s$, d.h. die Datenrate ist $80\ kBit/s$.

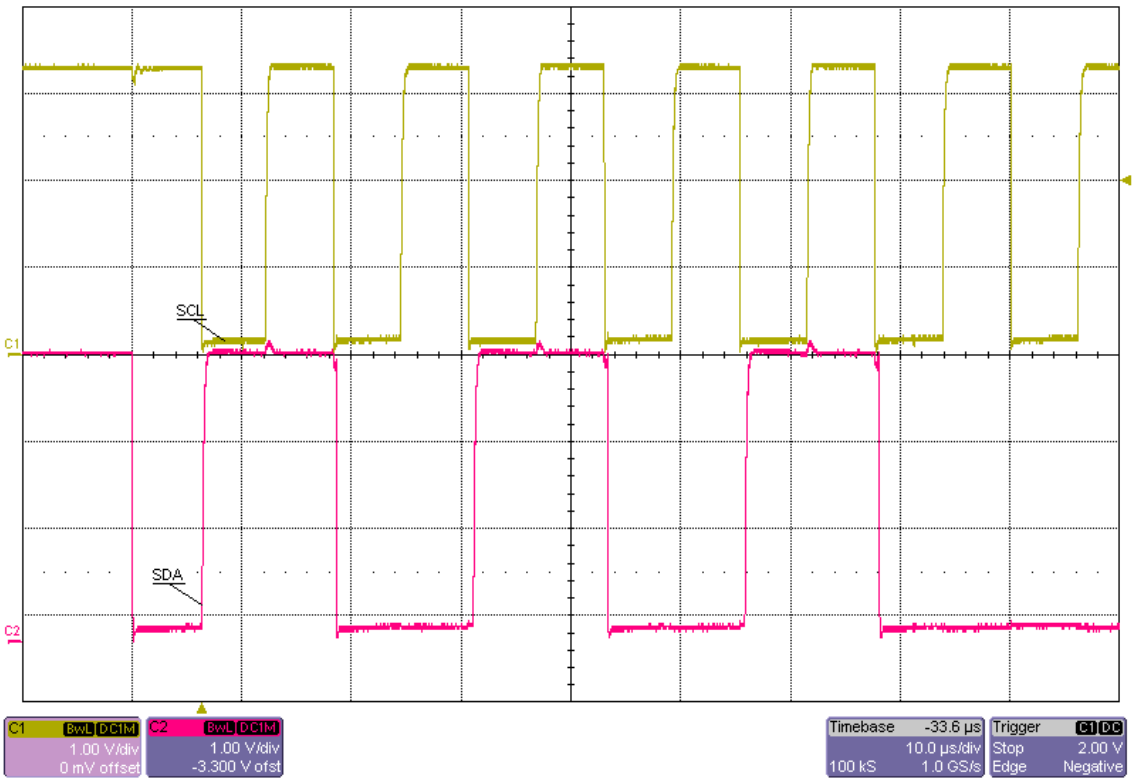

Abbildung 5.1.: Typischer Signalverlauf auf einem 1,5m langen Bus ohne Treiberbausteine

In Abbildung 5.2 ist der Signalverlauf auf einem 2 Meter langen I²C-Bus, der ebenfalls ohne Bustreiber-Adapter betrieben wurde, dargestellt. Auf der Datenleitung SDA wurde dieselbe Bitfolge `1 0 1 0 1 0 0` mit der selben Datenrate von 80 $kBit/s$ übertragen.

Der Sender gibt mit der ersten fallenden Flanke des Taktsignals SCL das erste Datenbit (eine `1`) auf die Datenleitung SDA. Die gesendete `1` wird jedoch nach kurzer Zeit vom Sender wieder mit einer `0` überschrieben. Dieses Verhalten des Senders führt zu einer fehlerhaften Datenübertragung. Zwei Meter Kabellänge entsprechen ca. 120 pF kapazitiver Buslast. Gemäß der I²C-Spezifikation von NXP (siehe [NXP07]) sind jedoch mindestens 400 pF kapazitive Buslast erlaubt. Somit wird der geforderte Wert um mehr als Faktor drei unterschritten.

In Abbildung 5.3 ist die Datenübertragung der Bitfolge `1 0 1 0 1 0 0` über ein 6,5 Meter langes Kabel dargestellt. Diese Kabellänge entspricht einer kapazitiven Buslast von etwa 390 pF.

Die Datenübertragung startet, wie auch in den beiden obigen Fällen, mit der ersten fallenden Flanke auf der Taktleitung SCL. Kurz nach Auftreten der ersten fallenden Taktflanke bricht der Sender die Datenübertragung ab. Die beiden Busleitungen kehren in den Ruhezustand zurück. Auch wenn nach abgebrochener Datenübertragung eine kürzere Busleitung angesteckt wird, ist keine weitere Datenübertragung mehr möglich. Erst durch einen Reset

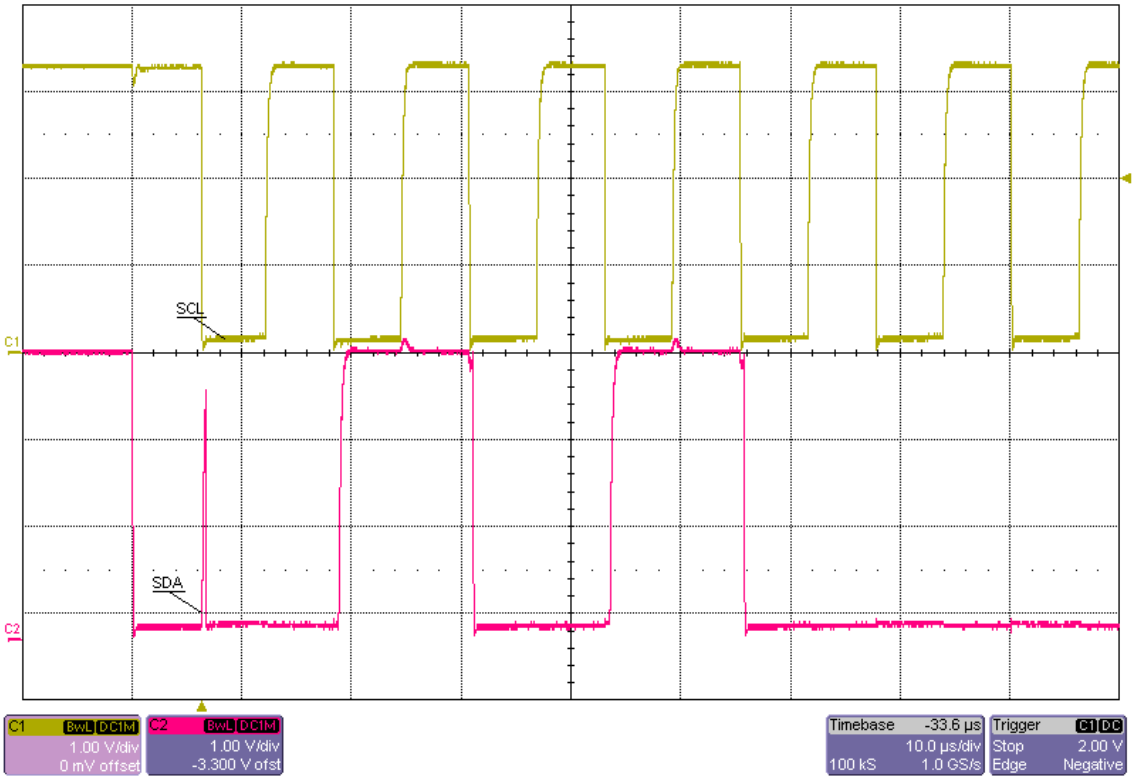

Abbildung 5.2.: Typischer Signalverlauf auf einem 2m langen Bus ohne Treiberbausteine

des Sensor-/Aktor-Knotens ist die Funktionsfähigkeit der I²C-Peripherie des ADuC7019 wieder gewährleistet.

Die Messungen aus den Abbildungen 5.2 und 5.3 bestätigen, dass sich die I²C-Peripherie des ADuC7019 nicht an die von NXP vorgeschriebene Spezifikation hält. Um dennoch eine Datenübertragung über mehrere hundert Meter zu ermöglichen, wird auf dem in Abschnitt 4.2.3 beschriebenen Bustreiber-Adapter ein Treiberbaustein von Texas Instruments vom Typ P82B96 eingesetzt.

Vergleich verschiedener Buslängen mit Bustreiber-Adapter

Es wurden drei weitere Messungen an einem mit Bustreiber-Adapter betriebenen I²C-Bus durchgeführt. Als Bustreiber kommt ein Baustein von Texas Instruments vom Typ P82B96 zum Einsatz. Die Datenleitung SDA und die Taktleitung SCL des I²C-Busses sind direkt mit den entsprechenden Pins des Treiberbausteins P82B96 verbunden. Ebenso wird der Microcontroller ADuC7019 mit dem Treiberbaustein P82B96 verbunden. Der P82B96 isoliert die kapazitive Last eines langen Busses vollständig vom Microcontroller ADuC7019. D.h. aus Sicht des Microcontrollers ist der Bus nur wenige Zentimeter lang, nämlich die Distanz vom Microcontroller selbst bis zum Bustreiber P82B96. Die eigentliche Busleitung

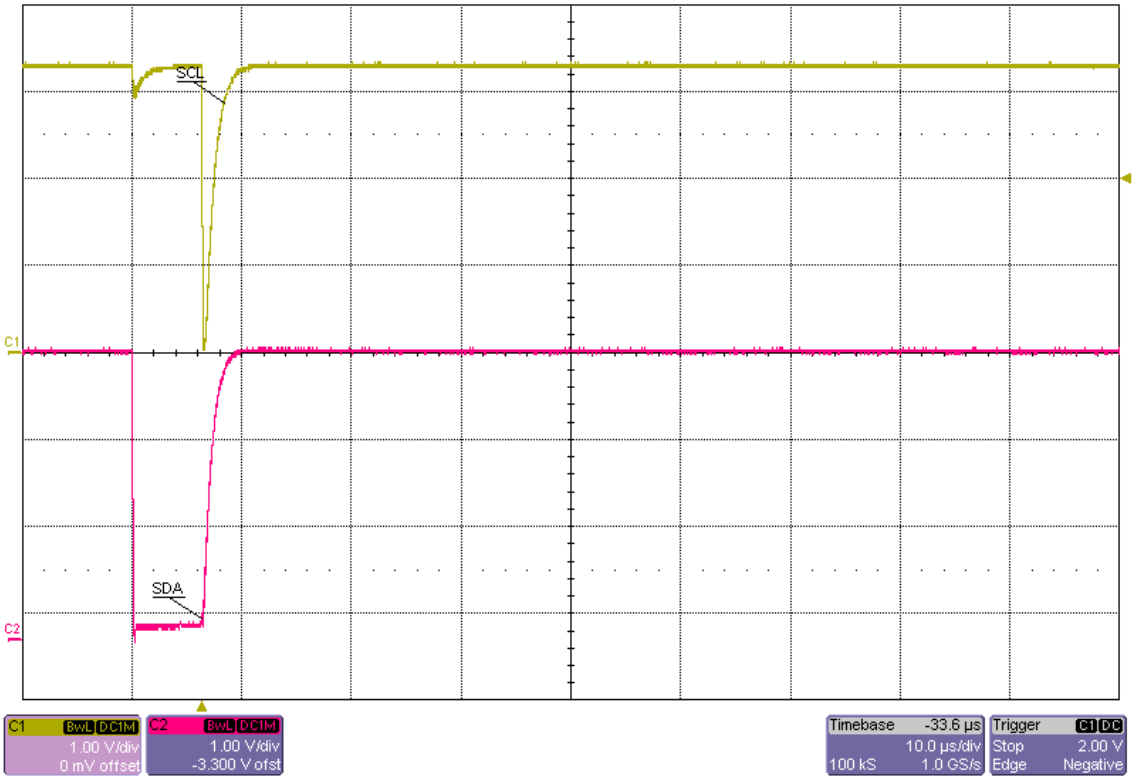

Abbildung 5.3.: Typischer Signalverlauf auf einem 12m langen Bus ohne Treiberbausteine

ist für den Microcontroller „unsichtbar“.

In den Abbildungen 5.4 bis 5.6 sind die Signalverläufe verschiedener Buslängen von I²C-Bussen mit Treiberbausteinen dargestellt.

Die übertragene Bitfolge ist, wie bei allen voran gegangenen Beispielen `1 0 1 0 1 0 0`. Je länger die Busleitung wird, desto größer wird auch die kapazitive Last der Leitung, die vom Bustreiber getrieben werden muss. Die Abbildungen 5.4 bis 5.6 unterscheiden sich lediglich in der Flankensteilheit der ansteigenden Flanke. Je länger die Leitung, also je größer die kapazitive Last ist, desto flacher wird die ansteigende Flanke.

Nach [NXP07] berechnet sich die Flankenanstiegszeit[1] zu

$$t_{r,10\%-90\%} = (\ln 0,9 - \ln 0,1) \cdot R_{PU} \cdot C_{Leitung} \approx 2,1972 \cdot R_{PU} \cdot C_{Leitung}. \tag{5.1}$$

R_{PU} ist der Wert der Pull-up-Widerstände, die in der Stromversorgung des medizinischen Netzwerks integriert sind (siehe Abschnitt 4.2.4). Der Wert dieser Widerstände beträgt 100 Ω. $C_{Leitung}$ ist die Kapazität der Busleitung.

[1] In diesem Fall ist die 10%-90% Anstiegszeit gemeint.

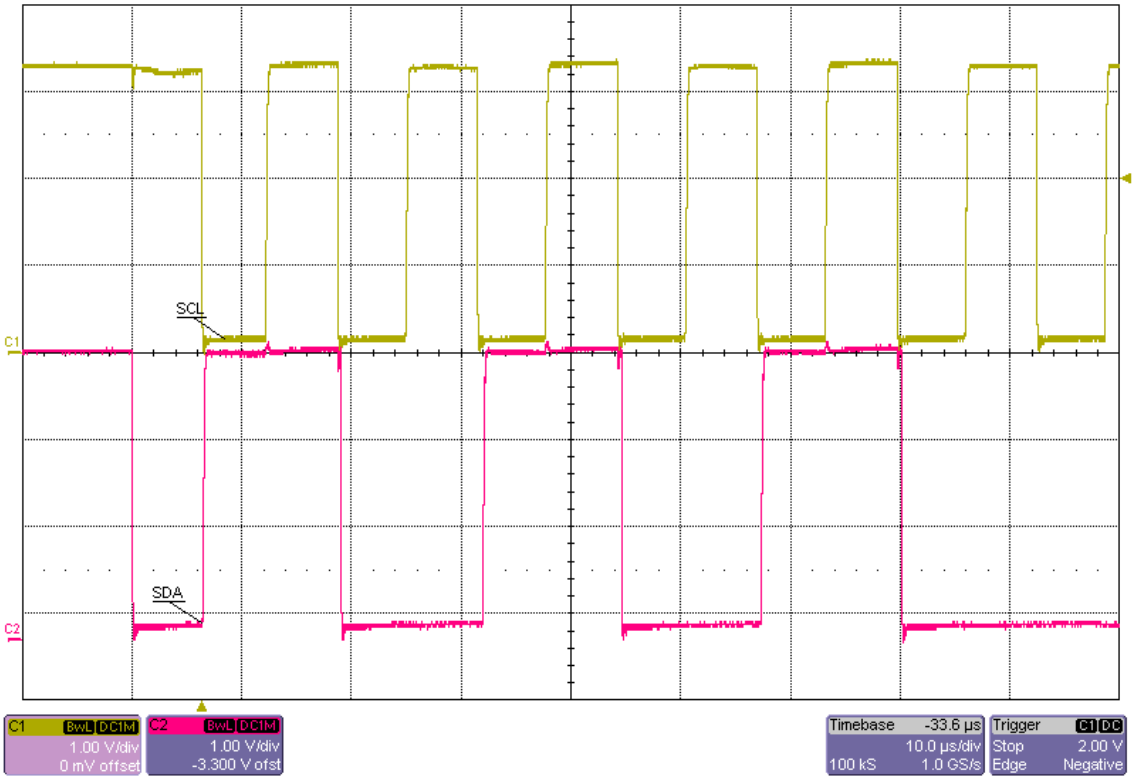

Abbildung 5.4.: Typischer Signalverlauf auf einem 12m langen Bus mit Treiberbausteinen

Um eine fehlerfreie Datenübertragung zu ermöglichen, muss die Flankenanstiegszeit kleiner als die halbe Periodendauer t_P des Taktsignals sein. Bei $f_{SCL} = 80\ kHz$ Taktfrequenz gilt für die Flankenanstiegszeit

$$t_{r,10\%-90\%} < \frac{t_P}{2} = \frac{1}{f_{SCL} \cdot 2} = 6,25\ \mu s.$$

Mit Gleichung (5.1) und einer Flankenanstiegszeit von $6,25\ \mu s$ ergibt sich die maximale Leitungskapazität zu

$$C_{Leitung} = \frac{t_{r,10\%-90\%}}{2,1972 \cdot R_{PU}} = \frac{6,25\ \mu s}{2,1972 \cdot 100\Omega} \approx 28,44\ nF$$

Bei einem Kapazitätsbelag der Leitung von 60 pF/m lässt sich mit Hilfe des Bustreiber-Adapters unter Berücksichtigung der Wellenausbreitungseffekte aus Abschnitt 2.4 (Verwendung von Abschlusswiderständen) eine Buslänge von etwa 475 Meter realisieren.

Abschließend lässt sich sagen, dass der Betrieb von Sensor-/Aktor-Knoten ohne Bustreiber-Adapter keine zufrieden stellenden Ergebnisse liefert. Die maximal erreichbare Buslänge

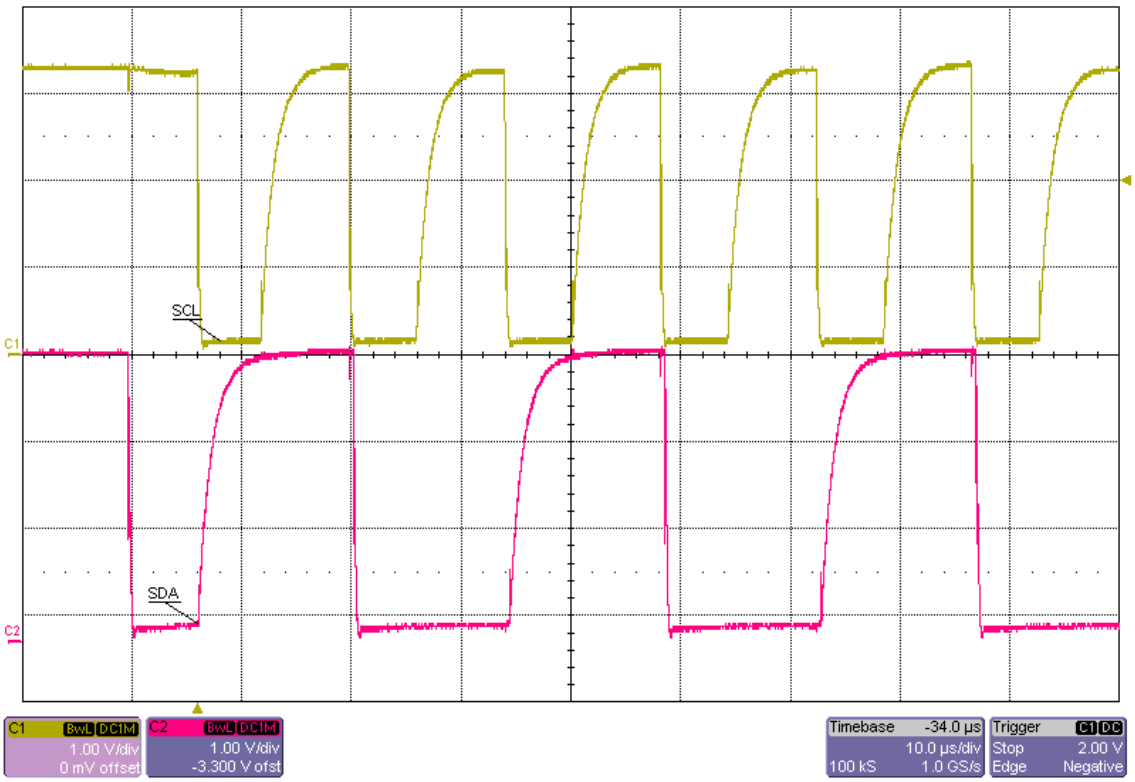

Abbildung 5.5.: Typischer Signalverlauf auf einem 178m langen Bus mit Treiberbausteinen

ist, aufgrund von Unvollkommenheiten der I²C Hardware, beim Betrieb ohne Bustreiber-Adapter auf etwa 1,5 Meter beschränkt. Dieser Wert ist für den Einsatz in einem mobilen medizinischen Netzwerk nicht praxistauglich. Durch den Einsatz von Bustreiber-Adaptern lässt sich die maximale Buslänge auf mehr als 400 Meter erweitern. Dieser Wert kann als praxistauglich angesehen werden.

5.3. Elektromagnetische Verträglichkeit

In Abbildung 5.7 ist der Signalverlauf auf einem 12 Meter langen Bus dargestellt, der Störeinflüssen eines Mobiltelefons unterlag. Das Mobiltelefon sendet im GSM[2]1800-Band. Es wurde ein Abstand von 0,5 Metern zwischen Mobiltelefon und Busleitung gewählt.

Abbildung 5.8 zeigt das gleiche Szenario wie Abbildung 5.7 jedoch wurde als Störquelle eine WLAN[3]-Karte eines Notebooks gewählt, die im 2,4 GHz-Band sendet. Der Abstand zwischen der WLAN-Antenne und der Busleitung betrug ebenfalls 0,5 Meter.

Die beiden Abbildungen 5.7 und 5.8 zeigen deutlich, dass weder Mobiltelefone noch WLAN-

[2]Global System for Mobile Communications
[3]Wireless Local Area Network

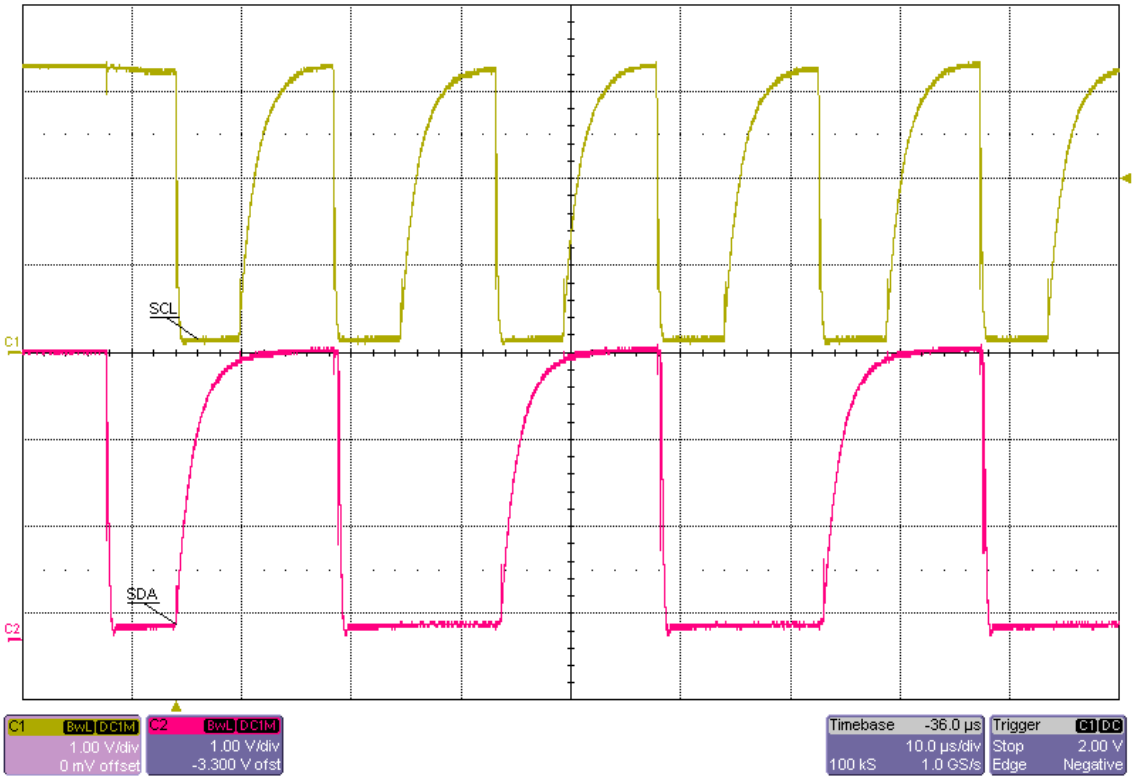

Abbildung 5.6.: Typischer Signalverlauf auf einem 260m langen Bus mit Treiberbausteinen

Netzwerke die Funktionsfähigkeit des medizinischen Netzwerks beeinträchtigen. Diese hohe Störresistenz ist zum einen der pseudo-symmetrischen Signalübertragung auf den beiden Busleitungen zuzuschreiben, zum anderen aber auch dem verwendeten abgeschirmten Patchkabel.

5.4. Stromverbrauch des medizinischen Netzwerks

In Abbildung 5.9 ist die Stromaufnahme der einzelnen Hardwarekomponenten inkl. Bustreiber des medizinischen Netzwerks dargestellt. Beim Sensor-/Aktor-Knoten wurde der Stromverbrauch in den verschiedenen Betriebsmodi Undefined-Mode, Sensor-Mode und Aktor-Mode ermittelt.

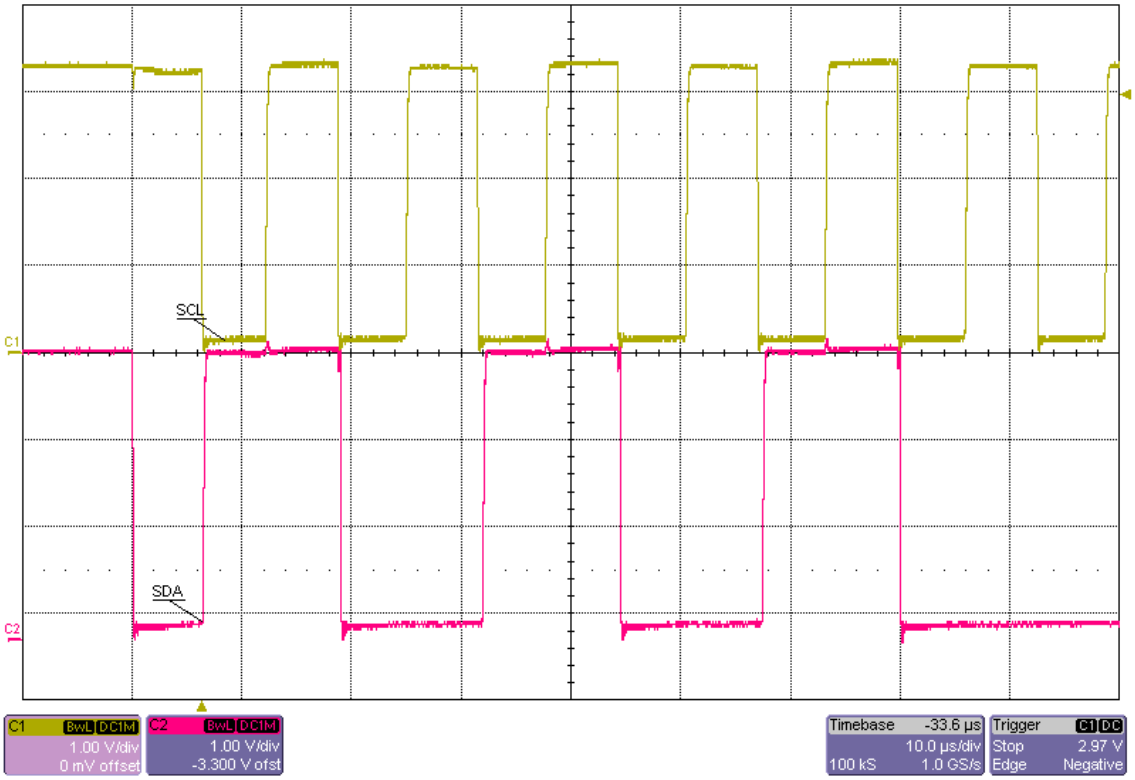

Abbildung 5.7.: Typischer Signalverlauf auf einem 12m langen Bus unter Störeinfluss eines Mobiltelefons

	Betriebsspannung	Stromverbrauch	Leistungsaufnahme
Sensor-/Aktor-Knoten			
Undefined-Mode	3,30 V	15,6 mA	51,48 mW
Sensor-Mode; Analog-Digital-Wandler deaktiviert	3,30 V	15,6 mA	51,48 mW
Sensor-Mode; Analog-Digital-Wandler aktiviert	3,30 V	16,8 mA	55,44 mW
Actor-Mode	3,30 V	13,8 mA	45,54 mW
Interface-Knoten	3,30 V	35,7 mA	117,81 mW

Abbildung 5.9.: Stromverbrauch der einzelnen Hardwarekomponenten

Die Unterschiede im Stromverbrauch zwischen den einzelnen Betriebsmodi des Sensor-/Aktor-Knotens liegen hauptsächlich in der unterschiedlichen Anzahl aktiver Leuchtdioden (siehe Abbildung 4.4 auf Seite 64). Die Differenz zwischen Interface-Knoten und Sensor-/Aktor-Knoten ergibt sich aus dem verwendeten UART-Transmitter (siehe [Ana06]), dieser hat eine Stromaufnahme von ca. 20 mA.

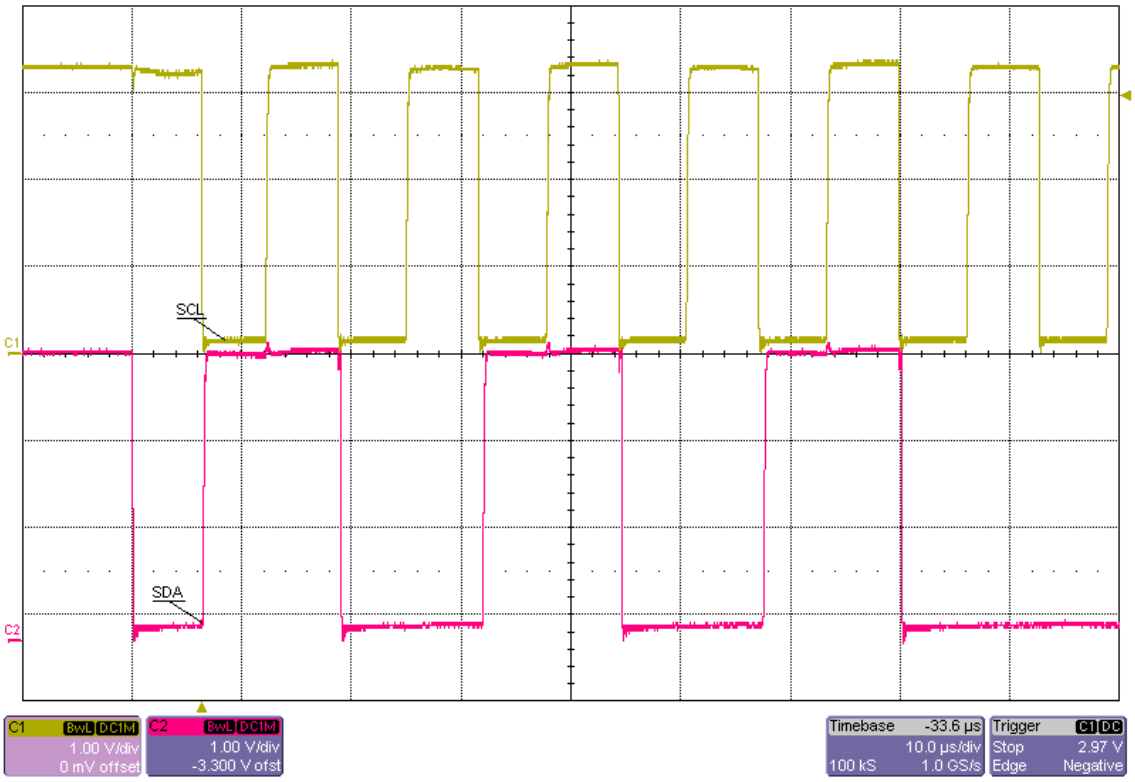

Abbildung 5.8.: Typischer Signalverlauf auf einem 12m langen Bus unter Störeinfluss eines WLANs

6. Diskussion

Nachfolgend sollen die Ergebnisse dieser Diplomarbeit in Bezug auf die Anforderungen aus Abschnitt 1.3 diskutiert werden.

Das mobile medizinische Ad-hoc Netzwerk ist in der Lage durch Verwendung von vernetzten Sensor-/Aktor-Knoten Vitalparameter verletzter Personen zu erfassen und bei Eintreten eines kritischen Kreislaufzustandes (z.B Herz- oder Atemstillstand) eine Nachricht an das behandelnde medizinische Personal zu verschicken. Des weiteren ist es möglich Messwerte von Biosignalen verletzter Personen über das Netzwerk zu übertragen. Das mobile medizinische Ad-hoc Netzwerk wird somit den beiden funktionalen Anforderungen aus Abschnitt 1.3 gerecht.

Aus technischer Sicht wurde darüber hinaus die Multi-Master-Fähigkeit des Netzwerks gefordert. Als Bussystem wurde in dieser Arbeit der I^2C-Bus ausgewählt. Dadurch wird gewährleistet, dass jeder Busteilnehmer unabhängig von allen anderen Busteilnehmern ist. Dies wird durch das von I^2C verwendete Arbitrierungsverfahren CSMA/CR sichergestellt, d.h. es gibt kein Busteilnehmer, der für die Zuteilung des Busses an andere Busteilnehmer verantwortlich ist. Der Ausfall eines Busteilnehmers hat demnach keine funktionalen Konsequenzen auf das verbleibende Netzwerk. Die Forderung nach Multi-Master-Fähigkeit des medizinischen Netzwerks, sowie die Forderung nach einer hohen Ausfallsicherheit kann somit erfüllt werden.

Des weiteren kann durch Verwendung der Bustopologie das medizinische Netzwerk ohne großen Aufwand um weitere Sensor-/Aktor-Knoten erweitert werden (es sind bis zu 126 Sensor-/Aktor-Knoten möglich). Es genügt einen neuen Busteilnehmer mittels eines T-Stücks an den vorhandenen Bus anzukoppeln. Die Forderung nach einer hohen Skalierbarkeit des mobilen medizinischen Ad-hoc Netzwerks ist somit erfüllt.

Mit dem verwendeten Kommunikationsprotokoll können maximal 264 Bytes pro Datenframe übertragen werden, d.h. bei einer Übertragungsrate von 80 $kBit/s$ ist das Busmedium im schlimmsten Fall (Worst Case) wieder nach $\frac{264\ Bytes \cdot 8\ Bit/Byte}{80\ kBit/s} \approx 30\ ms$ frei. Dadurch ist die Echtzeitfähigkeit des mobilen medizinischen Ad-hoc Netzwerks gewährleistet, denn spätestens 30 ms nach Eintreten eines Sendewunsches hat ein Busteilnehmer die Möglichkeit Daten zu senden. Durch die Übertragung einer 16-Bit Prüfsumme am Ende eines jeden Datenframes kann eine hohe Datenintegrität gewährleistet werden. Jede - gewollt oder ungewollt - manipulierte Datenübertragung kann somit identifiziert werden.

Der Interface-Knoten stellt die Verbindung zwischen dem mobilen medizinischen Ad-hoc

Netzwerk und einem PC her. Auf dem PC ist es möglich, durch den Einsatz eines Datenbanksystems, Messwerte eines oder mehrerer Sensor-/Aktor-Knoten abzuspeichern. Des weiteren können die Sensor-/Aktor-Knoten durch das Versenden von Konfigurationsnachrichten von einem PC aus konfiguriert werden. Da alle Sensor-/Aktor-Knoten über die selbe Hardware verfügen, ist es z.B. möglich einen Teil der Sensor-/Aktor-Knoten als Sensoren zu konfigurieren während die restlichen Sensor-/Aktor-Knoten als Aktoren arbeiten. Das in dieser Arbeit konzipierte und entwickelte mobile medizinische Ad-hoc Netzwerk wird somit auch den beiden letzten Anforderungen aus Abschnitt 1.3 gerecht.

In Abbildung 6.1 wird das in dieser Arbeit entwickelte medizinische Netzwerk mit dem Stand der Technik aus Abschnitt 1.2 verglichen.

	Multi-Master fähig?	Maßnahmen zur Sicherung der Datenintegrität?	Große Buslängen erzielbar?	Kann "jeder mit jedem" kommunizieren"?
Stand der Technik	ja	nein	nein	nein
Diplomarbeit	ja	ja	ja	ja

Abbildung 6.1.: Vergleich dieser Diplomarbeit mit dem Stand der Technik

Beide Systeme, sowohl das System aus dieser Diplomarbeit als auch das in [Fer09] vorgestellte medizinische Netzwerk, sind Multi-Master fähig. D.h. in beiden Systemen können Busteilnehmer schreibend auf den Bus zugreifen ohne sich vorher die Sendeerlaubnis bei einem anderen Busteilnehmer einholen zu müssen.

Das medizinische Netzwerk aus Abschnitt 1.2 bietet nur rudimentäre Möglichkeiten zur Sicherung der Datenintegrität. Zur Datensicherung wird lediglich das von UART bekannte Paritätsbit eingesetzt. Dies entspricht nicht den Anforderungen an ein medizinischen Netzwerk, da Übertragungsfehler eventuell unerkannt bleiben. Das in dieser Arbeit entwickelte medizinische Netzwerk verwendet zur Datensicherung eine 16-Bit Prüfsumme, durch die gewährleistet wird, dass jede Manipulation der gesendeten Daten empfängerseitig erkannt wird.

Durch den Einsatz von Bustreibern können mit dem in dieser Arbeit entwickelten medizinischen Netzwerk Distanzen von mehreren hundert Metern überbrückt werden (siehe Abschnitt 5.2). Das in Abschnitt 1.2 beschriebene medizinische Netzwerk verzichtet auf den Einsatz von Bustreibern. Bei Datenratenraten von etwa 100 $kBit/s$ wären damit nur Buslängen im unteren zweistelligen Meterbereich möglich.

Bei dem in [Fer09] vorgesetllten System handelt es sich streng genommen nicht um ein „echtes“ Bussystem. Zur Kommunikation zwischen zwei Netzteilnehmern werden zwei unidirektionale Leitungen, anstatt der bei Bussystemen üblichen bidirektionalan Leitung verwendet. Dies hat zur Folge, dass nicht jeder Netzteilnehmer mit jedem anderen kommunizieren kann. Sensoren können z.B. nicht mit Sensoren kommunizieren und Aktoren können auch nicht mit Aktoren kommunizieren. Dies ist zunächst zwar kein großer Nachteil, da normalerweise Daten von Sensoren zu Aktoren übertragen werden oder umgekehrt von Aktoren zu Sensoren. Der Interface-Knoten jedoch muss in der Lage sein, sowohl mit Sensoren als auch mit Aktoren zu kommunizieren. Bei dem in Abschnitt 1.2 beschriebenen System muss

der Interface-Knoten manuell umgeschaltet werden, je nach dem, ob der Interface-Knoten mit einem Sensor oder mit einem Aktor kommunizieren soll. Das in dieser Arbeit entwickelte medizinische Netzwerk verwendet zur Kommunikation zwischen Busteilnehmern eine bidirektionale Datenleitung und eine bidirektional Taktleitung. Dadurch kann jeder Busteilnehmer mit jedem anderen Busteilnehmer kommunizieren. Auch der Interface-Knoten, der selbst ein Busteilnehmer ist, kann mit jedem beliebigen Busteilnehmer kommunizieren.

Abschließend lässt sich sagen, dass das in dieser Arbeit konzipierte und entwickelte medizinische Netzwerk allen in Abschnitt 1.3 genannten Anforderungen gerecht wird. Zusätzlich wurden mit dieser Diplomarbeit die wesentlichen Schwachstellen des in Abschnitt 1.2 beschriebenen medizinischen Netzwerks beseitigt.

7. Zusammenfassung und Ausblick

In dieser Diplomarbeit wurde ein intelligentes, mobiles medizinisches Ad-hoc Netzwerk zur Überwachung von Personen bei Massenanfällen von Verletzten (MANV) konzipiert und entwickelt.

In Kapitel 1 wurde zunächst auf den aktuellen Stand der Technik, der aus der in [Fer09] beschriebenen Arbeit besteht, eingegangen. Darauf hin wurden die Ziele, die Vorgehensweise und die Gliederung der Arbeit beschrieben.

Wichtige theoretische Grundlagen für das Verständnis dieser Arbeit wurden in Kapitel 2 erläutert. Zu den erläuterten Grundlagen zählen die Signalübertragungsverfahren für elektrische Leiter, verwendete Endstufentypen in integrierten digitalen Schaltkreisen sowie die Leitungstheorie homogener Leitungen.

In Kapitel 3 wurde ein grundlegendes Konzept für das medizinischen Netzwerk entworfen. Zunächst wurden einige Netztopologieen und Bussysteme vorgestellt und ein den Anforderungen genügendes Bussystem für den Einsatz in einem medizinischen Netzwerk ausgewählt. Zusätzlich wurde in Kapitel 3 ein geeignetes Kommunikationsprotokoll für den Einsatz in einem medizinischen Netzwerk konzipiert.

In Kapitel 4 wurden alle Hardwarekomponenten, die für die Funktionsfähigkeit des medizinischen Netzwerks erforderlich sind, erläutert. Zusätzlich zur Hardware wurde auch die Firmware beschrieben, die für den Betrieb der Sensor-/Aktor-Knoten sowie des Interface-Knotens erforderlich ist.

Um die Praxistauglichkeit des medizinischen Netzwerks zu überprüfen, wurde dieses einigen Tests unterzogen. In Kapitel 5 wurde die maximal mögliche Buslänge des medizinischen Netzwerks ermittelt. Ebenso wurden Tests bezüglich elektromagnetischer Verträglichkeit und des Stromverbrauchs des medizinischen Netzwerks durchgeführt.

In Kapitel 6 wurden die Ergebnisse dieser Arbeit diskutiert, d.h. die Arbeit wurde gegenüber den Anforderungen aus Abschnitt 1.3 bewertet. Zusätzlich wurde in Kapitel 6 das in dieser Diplomarbeit entwickelte medizinische Netzwerk mit dem Stand der Technik verglichen.

Die vorliegende Arbeit hat gezeigt, dass das in Abschnitt 1.2 beschriebene medizinische Netzwerk in wesentlichen Punkten verbessert werden konnte. In einem weiteren Schritt kann der verwendete Bustreiber direkt auf dem Sensor-/Aktor-Knoten bzw. auf dem Inter-

face-Knoten untergebracht werden. Dies verringert die Anzahl nötiger Komponenten, die an den I²C-Bus angesteckt werden müssen und spart somit Kosten.

Des weiteren ist es wünschenswert einen Anwendungsserver zu entwickeln, mit dem das mobile medizinische Ad-hoc Netzwerk über eine webbasierte Benutzeroberfläche einfach konfiguriert werden kann. Im Moment wird zur Konfiguration des medizinischen Netzwerks noch ein Terminalprogramm verwendet, über das die Konfigurationsnachrichten an die jeweiligen Busteilnehmer gesendet werden können. Eine grafische Benutzeroberfläche stellt im Vergleich zu einem Terminalprogramm einen deutlichen Mehrwert dar. Zusätzlich zur grafischen Benutzeroberfläche könnte auch noch ein Datenbanksystem entwickelt werden, das in der Lage ist Messwerte von Sensoren zu speichern, um auf diese jederzeit zugreifen zu können.

Es wäre außerdem lohnenswert zu untersuchen, inwiefern es möglich ist das kabelgebundene medizinische Netzwerk durch ein funkbasiertes medizinisches Netzwerk zu ersetzen. Als mögliche Technologieen seien hier ZigBee, Bluetooth oder auch WLAN erwähnt.

A. Die Nachrichtenklassen und Nachrichtentypen im Überblick

Nachricht	CID/MID	LOP (in Bytes)	Beschreibung	Seite
Nachrichtenklasse EVN (Events)				
EVN-GEV	0x0A 0x01	0	Zeigt einen kritischen Kreislaufzustand an	53
Nachrichtenklasse RAW (Raw Measurements)				
RAW-GDA	0x0B 0x01	2	Überträgt Messwerte eines Sensor-Knotens an einen Aktor-Knoten oder an den Interface-Knoten	54
Nachrichtenklasse CON (Confirmation)				
CON-ACK	0x0C 0x01	2	Bestätigung einer korrekt verarbeiteten Nachricht	55
CON-NCK	0x0C 0x02	0	Bestätigung einer nicht korrekt verarbeiteten Nachricht	56
CON-ADB	0x0C 0x03	1	Versenden des Address Broadcasts im Undefined-Mode eines Sensor-/Aktor-Knotens	56
Nachrichtenklasse CFG (Configuration)				
CFG-MOD	0x0D 0x01	1	Konfiguriert den Betriebsmodus eines Sensor-/Aktor-Knotens	57
CFG-DES	0x0D 0x02	1	Konfiguriert die Zieladresse von Nachrichten der Nachrichtenklassen EVN und RAW	58
CFG-ADT	0x0D 0x03	2	Konfiguriert den Analog-Digital-Wandler eines Sensor-/Aktor-Knotens	58
CFG-DAT	0x0D 0x04	1	Konfiguriert den Digital-Analog-Wandler eines Sensor-/Aktor-Knotens	59

Abbildung A.1.: Die Nachrichtenklassen und -typen im Überblick

B. Abbildungsverzeichnis

C. Literaturverzeichnis

[Ana06] Analog Devices: *ADM3315E - 15 kV ESD Protected, 2.7 V to 3.6 V Serial Port Transceivers with Green Idle™*, 2006. http://www.analog.com.

[Ana07] Analog Devices: *Precision Analog Microcontroller, 12-Bit Analog I/O, ARM7TDMI MCU*, 2007. http://www.analog.com.

[ARC09] ARCNET User Group e.V.: *ARCNET - Die universelle echtzeitfähige Feldbuslösung*, 2009. http://www.arcnet.de.

[Bos96] Bosse, Georg: *Grundlagen der Elektrotechnik III - Wechselstromlehre, Vierpol- und Leitungstheorie*. VDI, 3. Auflage, 1996.

[Die06] Diestel, Reinhard: *Graph Theory*. Springer, 3. Auflage, 2006.

[Ets02] Etschberger, Konrad: *Controller-Area-Network*. Hanser, 3., aktualisierte Auflage, 2002.

[Fer09] Fernsner, Stefan: *Entwurf und Konzeption eines autonomen medizinischen Netzwerkes*. Studienarbeit, Universität Karlsruhe (TH), 2009.

[IEE82] IEEE Transactions on Communication: *An Arithmetic Checksum for Serial Transmissions*, Band COM-30. J. Fletcher, Januar 1982.

[Int03] International Organization for Standardization: *ISO 11898-1: Controller area network - Part 1: Data link layer and physical signalling*, 2003. http://www.iso.org.

[Kie06] Kiencke, Uwe: *Ereignisdiskrete Systeme: Modellierung und Steuerung verteilter Systeme*. Oldenbourg, 2. Auflage, 2006.

[Law00] Lawrenz, Wolfhard: *CAN - Controller Area Network*. Hüthig, 4., überarbeitete Auflage, 2000.

[NXP07] NXP Semiconductors: *UM10294 I2C-bus specification and user manual Rev. 03*, 2007. http://www.nxp.com.

[Rei02] Reissenweber, Bernd: *Feldbussysteme zur industriellen Kommunikation.* Oldenbourg, 2. Auflage, 2002.

[rnOD08] Olaf Dössel, Prof. Dr. rer. nat.: *Lineare Elektrische Netze.* Vorlesungsskript, Universität Karlsruhe (TH), 2008.

[Sch06] Schnell, Gerhard: *Bussysteme in der Automatisierungs- und Prozesstechnik.* Vieweg, 6. Auflage, 2006.

[SEG09] SEGGER Microcontroller GmbH & Co. KG: *J-Link / J-Trace ARM User guide of the JTAG emulators for ARM Cores Rev. 61*, 2009. http://www.segger.com.

[Tex07] Texas Instruments: *P82B96 - Dual Bidirectional Bus Buffer*, 2007. http://www.ti.com.

[Tie02] Tietze, Ulrich: *Halbleiter-Schaltungstechnik*, Kapitel 7, Seiten 633–635. Springer, 12. Auflage, 2002.

[Vec09] Vector Informatik GmbH, 2009. http://www.vector.com.

[Wik09] Wikipedia, 2009. http://www.wikipedia.org.

[Wit02] Wittgruber, Friedrich: *Digitale Schnittstellen und Bussysteme.* Vieweg, 2., überarbeitete und erweiterte Auflage, 2002.

[Zim07] Zimmermann, Werner: *Bussysteme in der Fahrzeugtechnik.* Vieweg, 2. Auflage, 2007.

[Zwe90] Zweig, J.: *RFC 1146 - TCP Alternate Checksum Options.* http://tools.ietf.org/html/rfc1146, 1990.