



# School of Engineering

InES Institute of  
Embedded Systems

## **Bachelorarbeit Informatik**

### Messstation zur Registrierung von Geschiebe- Bewegungen im Fluss

<b>Autoren</b>	Tobias Keller Tobias Welti
<b>Betreuer</b>	Prof. Hans-Joachim Gelke, Dipl. El. Ing. FH ZHAW Institute for Embedded Systems
<b>Partner</b>	Carlos Rodrigo Wyss Eidg. Forschungsanstalt für Wald, Schnee und Landschaft WSL
<b>Datum</b>	18. Dezember 2014

## **Erklärung betreffend das selbständige Verfassen einer Bachelorarbeit an der School of Engineering**

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

**Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.**

Ort, Datum:

Unterschriften:

.....

.....

.....

.....

Das Original dieses Formulars ist bei der ZHAW-Version aller abgegebenen Bachelorarbeiten zu Beginn der Dokumentation nach dem Titelblatt mit Original-Unterschriften und -Datum (keine Kopie) einzufügen.

# Zusammenfassung

Zurzeit kommen zur Messung von Geschiebebewegungen in Gewässern diverse Lösungen zur Anwendung. Bei der Eidg. Forschungsanstalt für Wald, Schnee und Landschaft (WSL) werden hauptsächlich Geophone verwendet, welche kostspielig sind und aufwändige bauliche Massnahmen sowie eine verhältnismässig komplexe IT-Infrastruktur bedingen. Das Erstellen eines Prototypen, der die Vereinfachung dieses Systems durch die Konzeption eines geeigneten Bussystems und der Verwendung kompakter MEMS-Beschleunigungssensoren zur Ereigniserkennung demonstriert, war das Hauptziel dieser Arbeit.

Als Basis für die im Einsatz befindlichen Geräte dienen Boards mit einem ARM Cortex-M4 Mikrocontroller. Das konzipierte System besteht im wesentlichen aus zwei Komponenten: einem Datenlogger und den Sensoreinheiten. Der Datenlogger zeichnet die Messwerte auf einer SD-Karte auf und übernimmt die Konfiguration der angehängten Sensoreinheiten. Im Unterschied zum normalen CAN-Bus muss der Absender einer Meldung für das Bus-Protokoll identifizierbar sein. Deshalb fungiert der Datenlogger auch als Busmaster und weist den Teilnehmern am CAN-Bus eindeutige Identifikationen zu. Die Sensoreinheiten bestehen aus den MEMS-Beschleunigungssensoren und dem Mikrocontroller, der die Messdaten je nach gewähltem Modus verarbeitet. Möglich ist dabei das Aufzeichnen der wichtigsten Kenndaten eines Ereignisses, die detaillierte Aufzeichnung eines Einschlages in zwei Detailstufen oder das Senden der Rohdaten eines Sensors. Durch die direkte Verarbeitung der Messdaten wird einerseits der Bus entlastet. Es werden nur die verarbeiteten Daten übermittelt. Andererseits fällt für den Datenlogger weniger Arbeit an, da dieser keine Sensordaten auswerten muss und nur noch die fertigen Daten speichert.

Da das ganze System unter anderem in Gebirgsbächen eingesetzt werden soll, musste ein besonderes Augenmerk auf die Stabilität und die Autarkie der Lösung gelegt werden. Wurde die Konfiguration des Systems einmal manuell vorgenommen, so können die Werte abgespeichert und bei einem Neustart des Systems automatisch wieder geladen werden. Die Verwendung eines CAN-Busses garantiert zudem die fehlerfreie Übertragung der Daten.

Der erstellte Prototyp erfüllt die Erwartungen in Bezug auf die Vereinfachung des Aufbaus und den Verbrauch des ganzen Systems. Er könnte somit als Basis für ein finales Produkt dienen.

# Abstract

in English



# Vorwort

---

Vorwort: Hier ein  
chen blabla von  
mir

---

## Kontakte

**Verfasser**

Tobias Keller  
Schulstrasse 82  
CH-8952 Schlieren

E-Mail: kellet01@students.zhaw.ch

Tobias Welti, MSc ETH Zürich  
Wissenschaftlicher Assistent ZHAW Institute for Embedded Systems  
Technikumstrasse 9  
CH-8401 Winterthur

E-Mail: tobias.welti@zhaw.ch  
Homepage: <http://www.ines.zhaw.ch>

**Betreuer**

Prof. Hans-Joachim Gelke, Dipl. El. Ing. FH  
ZHAW Institute for Embedded Systems  
Technikumstrasse 9  
CH-8401 Winterthur

E-Mail: hans.gelke@zhaw.ch  
Homepage: <http://www.ines.zhaw.ch>

**Partner**

Carlos Rodrigo Wyss  
Eidg. Forschungsanstalt WSL  
Zürcherstrasse 111  
8903 Birmensdorf

E-Mail: carlos.wyss@wsl.ch  
Homepage: <http://www.wsl.ch>

# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>6</b>
1.1. Ausgangslage . . . . .	6
1.1.1. Hydrologische Messungen . . . . .	6
1.2. Projektidee . . . . .	6
1.3. Ziel . . . . .	7
<b>2. Aufgabenstellung</b>	<b>8</b>
2.1. Aufgabenstellung . . . . .	8
2.1.1. Musskriterien . . . . .	8
2.1.2. Wunschkriterien . . . . .	9
2.1.3. Abgrenzungskriterien . . . . .	9
<b>3. Vorgehen</b>	<b>10</b>
3.1. Überblick . . . . .	10
3.2. Datenlogger . . . . .	10
3.2.1. Messdaten sammeln . . . . .	11
3.2.2. Kontrolle über das Bussystem . . . . .	11
3.2.3. Steuerung des Betriebs . . . . .	11
3.2.4. Schnittstelle nach Aussen . . . . .	11
3.3. Sensoreinheit . . . . .	11
3.3.1. Messdatenerfassung . . . . .	11
3.3.2. Ereigniserkennung . . . . .	12
3.3.3. Datenübertragung . . . . .	12
3.4. Bussystem . . . . .	12
<b>4. Funktionale Anforderungen</b>	<b>13</b>
4.1. Datenlogger (F1...) . . . . .	13
4.1.1. F110 Busmaster . . . . .	13
4.1.2. F120 Sensorerkennung . . . . .	13
4.1.3. F130 Uhrzeit . . . . .	13
4.1.4. F140 Timestamp verteilen . . . . .	13
4.1.5. F160 Schnittstelle zum Steuerrechner . . . . .	13
4.1.6. F170 Steuerung Betriebsmodus . . . . .	13
4.1.7. F180 Daten sammeln . . . . .	14
4.1.8. F190 Daten speichern . . . . .	14
4.2. Sensoreinheit (F4...) . . . . .	14
4.2.1. F410 Ereignisdetektion . . . . .	14
4.2.2. F430 Datenübertragung . . . . .	14
4.2.3. F450 Rohdatenaufzeichnung . . . . .	14
<b>5. Nichteinfunktionale Anforderungen</b>	<b>15</b>
<b>6. Tests</b>	<b>16</b>
6.1. Testaufbau . . . . .	16
6.2. Datenlogger . . . . .	16
6.2.1. T110 Busmaster . . . . .	16
6.2.2. T120 Sensorerkennung . . . . .	16
6.2.3. T130 Uhrzeit . . . . .	16

6.2.4. T140 Timestamp zurücksetzen . . . . .	18
6.2.5. T160 Schnittstelle zum Steuerrechner . . . . .	18
6.2.6. T170 Steuerung Detail-Level . . . . .	18
6.2.7. T180 Daten sammeln und speichern . . . . .	18
6.3. Sensoreinheit . . . . .	19
6.3.1. T400 Konfiguration . . . . .	19
6.3.2. T410 Ereignisdetektion . . . . .	20
6.3.3. T430 Datenübertragung . . . . .	20
6.3.4. T450 Rohdatenaufzeichnung . . . . .	20
6.4. Nichtfunktionale Tests . . . . .	20
6.4.1. T710 Stromverbrauch . . . . .	20
<b>7. Hardware</b>	<b>22</b>
7.1. Hardware-Architektur . . . . .	22
7.1.1. Datenlogger . . . . .	22
7.1.2. Sensoreinheit . . . . .	22
7.1.3. Bussystem . . . . .	22
7.2. Komponentenauswahl . . . . .	22
7.2.1. Mikroprozessor . . . . .	22
7.2.2. Bus-System . . . . .	25
7.2.3. Speichermedium . . . . .	29
7.2.4. Sensor . . . . .	30
7.2.5. Schnittstelle . . . . .	30
7.2.6. Gehäuse . . . . .	30
7.3. Datenlogger . . . . .	30
7.4. Sensoreinheit . . . . .	31
<b>8. Software</b>	<b>33</b>
8.1. Software-Stack . . . . .	33
8.1.1. Überblick . . . . .	33
8.1.2. Messdatenerfassung . . . . .	35
8.1.3. Ereigniserkennung . . . . .	36
8.1.4. Detail-Level . . . . .	40
8.1.5. Timestamp . . . . .	40
8.2. Busverwaltung . . . . .	41
8.2.1. Verwaltung des Bussystems . . . . .	41
8.2.2. Busprotokoll . . . . .	44
8.2.3. Filesystem . . . . .	48
8.3. Konfiguration . . . . .	51
<b>9. Resultate</b>	<b>52</b>
9.1. Testfälle . . . . .	52
9.2. Ereigniserkennung . . . . .	52
9.3. Daten-Reduktion und -Speicherung . . . . .	52
9.3.1. Rohdaten (raw) . . . . .	52
9.3.2. Detaillierte Ereignisdaten (detailed) . . . . .	53
9.3.3. Peaks (peaks only) . . . . .	53
9.3.4. Minimale Daten (sparse) . . . . .	53
9.3.5. Messdauer . . . . .	53
9.4. Hardware . . . . .	54
<b>10. Diskussion</b>	<b>55</b>
10.1. Ergebnisse . . . . .	55
10.1.1. Funktionale Anforderungen . . . . .	55
10.1.2. Nichtfunktionale Anforderungen . . . . .	55
10.2. Rückblick auf die Aufgabenstellung . . . . .	55

10.3. Ausblick . . . . .	56
10.3.1. Hardware . . . . .	56
10.3.2. Software . . . . .	56
<b>11. Bedienungsanleitung</b>	<b>58</b>
11.1. Produktbeschrieb . . . . .	58
11.2. Aufbau der Messstation . . . . .	58
11.3. Datenlogger . . . . .	58
11.3.1. Anschlüsse . . . . .	59
11.4. Sensor . . . . .	60
11.4.1. Beschleunigungssensor . . . . .	60
11.4.2. A/D-Wandler . . . . .	60
11.4.3. Mikroprozessor . . . . .	61
11.4.4. Anschlüsse . . . . .	61
11.5. Bussystem . . . . .	61
11.5.1. Kabel . . . . .	61
11.5.2. Abschlusswiderstände . . . . .	61
11.5.3. Steckverbinder . . . . .	61
11.6. Ereignis . . . . .	63
11.6.1. Detail-Level . . . . .	63
11.7. Konfiguration . . . . .	68
11.7.1. Anschluss eines Computers . . . . .	68
11.7.2. Menü . . . . .	68
11.7.3. Befehle . . . . .	69
11.7.4. Konfigurationsdatei . . . . .	77
<b>12. Verzeichnisse</b>	<b>78</b>
<b>Literaturverzeichnis</b>	<b>79</b>
<b>Abbildungsverzeichnis</b>	<b>80</b>
<b>Tabellenverzeichnis</b>	<b>82</b>
<b>Glossar</b>	<b>84</b>
<b>Abkürzungen</b>	<b>89</b>
<b>A. Anhang</b>	<b>I</b>
A.1. Offizielle Aufgabenstellung . . . . .	I
A.2. Projektmanagement . . . . .	III
A.2.1. Zeitplanung und Meilensteine . . . . .	III
A.2.2. Besprechungsprotokolle . . . . .	III
A.3. Schaltpläne . . . . .	VI
A.4. Datenblätter . . . . .	X
A.4.1. NXP LPC4088 32-bit ARM Cortex-M4 microcontroller . . . . .	X
A.4.2. Embedded Artists NXP LPC4088 QuickStart Board . . . . .	XII
A.4.3. Analog Devices ADXL001 Beschleunigungssensor . . . . .	XV
A.5. Fotos . . . . .	XVII
A.6. Source Code, Daten und Multimedia . . . . .	XVII

# Liste der noch zu erledigenden Punkte

Abstract in English . . . . .	4
Vorwort: Hier ein bisschen blabla von dir und mir . . . . .	5
Hardware . . . . .	22
Sensorauswahl beschreiben . . . . .	30
software . . . . .	33
TK: Zusammenhänge A/D-Wandlung und Ereigniserkennung und Übertragung beschreiben, dazu ein Kommunikationsdiagramm, wo synchron, wo asynchron. . . . .	40
Bedienungsanleitung . . . . .	58
figure zu messstation einfgen . . . . .	58
Anhang . . . . .	I
Inhaltsverzeichnis der CD erstellen CD mit dem vollständigen Bericht als pdf-File inklusive Film- und Fotomaterial . . . . .	XVII

# 1. Einleitung

## 1.1. Ausgangslage

Die Eidg. Forschungsanstalt für Wald, Schnee und Landschaft (WSL) betreibt Messstationen zur Registrierung von Geschiebe-Bewegungen im Fluss mittels Geophonen, die unter Stahlplatten montiert sind. Diese Platten sind in einer Betonkonstruktion eingelassen, um sie im Flussbett zu fixieren. Die Betonkonstruktion dient gleichzeitig als Kabelkanal. Jedes Geophon ist über ein Kabel mit einem Auswertungs-Rechner (Embedded PC) verbunden. Der Rechner wertet die Signale aller angeschlossenen Geophone kontinuierlich aus, um die Ereignisse zu detektieren. Bei mehreren Geophonen ist hier ein recht leistungsfähiger Rechner nötig, der eine entsprechend hohe Leistungsaufnahme hat. Die baulichen Massnahmen für die Installation der Geophone, der Auswertungsstation sowie der Stromversorgung sind sehr teuer. Da viele dieser Messstationen in Gebirgsbächen installiert werden, fallen hohe Transportkosten für die schweren Materialien an.

### 1.1.1. Hydrologische Messungen

Diese Messstationen benutzt die WSL, um Daten über Geschiebetransport zu sammeln. Solche Datensammlungen spielen eine zentrale Rolle im Flussbau und in der Hydrologie. Die Vorhersage von Menge und Zeitpunkt des Auftretens von Geschiebetransport ist weiterhin schwierig, weshalb intensiv auf diesem Gebiet geforscht wird. Forschungsgruppen aus der ganzen Welt haben unterschiedlichste Sensoren und Messinstallationen entwickelt auf der Suche nach einem Instrument, um mehr und bessere Daten sammeln zu können. Dabei kommen Piezoelektrische Sensoren und Geophone zum Einsatz [1], aber auch Beschleunigungssensoren [2]. Hydrophone können ebenfalls eingesetzt werden, um die Geräusche der Geschiebekörper unter Wasser aufzuzeichnen. Dies erfordert aber eine viel höhere Abtastrate und daher mehr Rechenleistung [3]. Um die Daten in Korrelation zu Menge, Korngröße und Fliessgeschwindigkeit setzen zu können, gibt es Messanlagen mit weiteren Sensoren und Fangkörben (z.B. im Erlenbach im Alptal, Kanton Schwyz), die bei hohem Geschiebeaufkommen automatisch ins Flussbett gefahren werden, um das Geschiebe aufzufangen [4]. Auch Fangnetze werden zu diesem Zweck installiert [4].

Die hydrologische Forschung verfolgt unter anderem das Ziel, bessere Vorhersagen zu Naturereignissen machen zu können. Stauungen in einem Bach durch Geschiebeablagerungen können zu drastischen Verschlimmerungen führen, z.B. Ausuferungen bei einem Hochwasser [5].

## 1.2. Projektidee

Bruno Fritschi (WSL) hatte einige Ideen, wie die momentan sehr aufwändigen baulichen Massnahmen vereinfacht und der Sensoraufbau so modifiziert werden können, dass die Messstation mit weniger baulichem Aufwand eingerichtet werden kann.

Die wichtigste Idee sieht vor, die gemessenen Daten direkt am Sensor auszuwerten, und nur jene Daten zu übertragen und zu speichern, an denen die Forschung interessiert ist. Die Reduktion der Datenmenge an der Quelle ermöglicht die Verwendung eines Bussystems, also eines einzigen Kabels für alle Sensoren, statt eines eigenen Kabels für jeden Sensor. Da die Daten über das Bussystem aus dem Bach heraus auf einen Datensammler (Datenlogger) übertragen werden können, würde dies weitere Möglichkeiten eröffnen:

- Stromversorgung der Sensoren über das Kabel des Bussystems. Dies würde die Laufzeit einer solchen Anlage beträchtlich erhöhen, resp. Batterien in den Sensoren unnötig machen.
- Datensammlung an einem zugänglichen Gerät, statt Einsammeln der Sensoren aus dem Bach.
- Überwachung der Anlage zu Laufzeit, Kontrolle über den Zustand der Datensammlung und Änderung der Konfiguration der Sensoren im Betrieb.

Zukünftig sollen die Geophone durch ein- oder mehrachsige MEMS-Beschleunigungssensoren ersetzt werden, da diese wesentlich kleiner sind. Die Verkleinerung der Sensoren ermöglicht neue Bauformen der Messanlagen:

Die Integration der Sensoren in einer Elastomerplatte statt der bisherigen Montage unter Stahlplatten würde die baulichen Massnahmen und damit die Kosten drastisch senken. Die Verkabelung eines Bussystems ist weit weniger Komplex als für ein sternförmiges System wie bisher. Dies würde die Variante mit Sensoren in Elastomerplatten begünstigen.

### 1.3. Ziel

In diesem Projekt soll daher ein Messsystem entwickelt werden, das MEMS-Beschleunigungs-Sensoren in einem Bussystem einsetzt und die Signale direkt beim Sensor auswertet. Die Entwicklung zur Serienreife würde den Rahmen einer Bachelorarbeit für zwei Personen sprengen, weshalb nur ein Prototyp entwickelt werden soll. Anhand des Prototyps soll an der WSL untersucht werden können, ob sich die Weiterentwicklung lohnt.

Bei einem positiven Entscheid könnte in einer weiteren Bachelorarbeit die Weiterentwicklung zu einem Serienprodukt versucht werden.

## 2. Aufgabenstellung

Die offizielle Aufgabenstellung befindet sich im Anhang A.1.

### 2.1. Aufgabenstellung

Im Rahmen dieser Bachelorarbeit soll eine Lösung erarbeitet werden, um zukünftige Installationen günstiger zu machen. Da solche Messanlagen an sehr vielen Orten auf der ganzen Welt aufgebaut werden, kann durch eine Vereinfachung der Installation viel Aufwand gespart werden.

Die Projektidee stammt von Bruno Fritschi (WSL). Sein Vorschlag sieht vor, die aufgezeichneten Signale direkt am Sensor auszuwerten und nur die gewünschten Ereignis-Daten zu übertragen und zu speichern. Somit könnten die Daten über ein Bussystem übertragen werden und der Rechner für die Sammlung der Daten bräuchte weniger Rechenleistung.

Dank der Bustopologie kommt das Messsystem mit weniger Leitungen aus und kann einfacher installiert werden. Denkbar wäre die Integration in eine Elastomerplatte anstelle der Stahl- und Betonkonstruktion, da viel weniger Leitungen nötig sind. Die Elastomerplatte könnte einfacher im Bachbett verankert werden.

Ziel der Arbeit ist die Entwicklung der Auswertungshardware und des Bussystems. Die Qualität der gemessenen Signale soll mindestens erhalten werden. Die Auswertungsalgorithmen sind nicht Bestandteil der Arbeit und werden vom WSL zur Verfügung gestellt.

Die von der bisherigen Anlage gemachten Messdaten enthalten die Dauer und Intensität jedes Aufschlags (Ereignis) auf der Sensorplatte, sowie die Anzahl Ausschläge (Peaks) pro Aufschlag. Pro Minute wird ein Histogramm über die Intensitäten der Peaks gebildet und abgespeichert.

Denkbar wäre es, einen Prototyp für Vergleichsmessungen im Erlenbach (Alptal, SZ) an einer bestehenden Schwelle zu implementieren.

#### 2.1.1. Musskriterien

- Die Anlage zeichnet den Geschiebetransport im Bachbett auf. Die bisherige Aufzeichnungsrate von 10'000 Messpunkten pro Sekunde soll nicht unterschritten werden.
- Die Anlage liefert eine minütliche Zusammenfassung über die Ereignisse an jedem Sensor. Diese Zusammenfassung enthält die Anzahl Ereignisse, Dauer und Intensität der einzelnen Ereignisse sowie ein Histogramm über die Intensitätsverteilung.
- Die Messstation ist fähig, mindestens zehn Sensoren zu betreiben und ihre Messsignale aufzuzeichnen.
- Es ist möglich, die kompletten Rohdaten von einem Sensor über eine Dauer von 30 Minuten aufzuzeichnen. Während einer solchen Messung dürfen die anderen Sensoren ihre Messung einstellen.
- Die Sensoren können über bis zu fünfzehn Meter im Bachbett verteilt sein.
- Die Leistungsaufnahme der Anlage beim Betrieb von 10 Sensoren ist kleiner als zehn Watt.
- Die Datenaufzeichnung erfolgt in einem eigens entwickelten Datenlogger.

- Am Datenlogger kann ein Laptop angeschlossen werden, um Kontrollparameter der Messanlage zu setzen und um den Status der Anlage abzufragen.
- Die erfassten Messdaten werden im Datenlogger auf einer Speicherkarte gespeichert. Dies ermöglicht ein einfaches Abholen der Daten im Feld, indem die Speicherkarte ausgetauscht wird.

### 2.1.2. Wunschkriterien

- Die Anlage liefert für jedes Ereignis die Rohdaten in voller zeitlicher Auflösung.
- Der Sensoraufbau ermöglicht es, die Sensoren in einer Elastomerplatte zu verpacken. Die Elastomerplatte kann ohne Betonkonstruktion im Bachbett verankert werden.
- Am Datenlogger kann ein Laptop angeschlossen werden, um die erfassten Messdaten herunterzuladen.

### 2.1.3. Abgrenzungskriterien

- Es würde den Rahmen dieser Arbeit sprengen, die Messeinheiten zur Produktreife zu bringen. Es wird lediglich aufgezeigt, wie solche Messeinheiten realisiert werden könnten.
- Eine Testinstallation in einem Bach ist nicht möglich. Allenfalls kann ausserhalb der Bachelorarbeit in der Versuchsanstalt für Wasserbau, Hydrologie und Glaziologie der ETH Zürich ein kleiner Testlauf stattfinden.

# 3. Vorgehen

## 3.1. Überblick

Das zu entwickelnde Messsystem kann grob in drei Komponenten aufgeteilt werden.

1. Datenlogger
2. Sensoreinheit
3. Bussystem

Der Datenlogger hat die Aufgabe, von mehreren Sensoreinheiten registrierte Ereignisse zu empfangen und zu speichern. Die Sensoreinheiten messen kontinuierlich die Beschleunigung, werten die Signale aus und erkennen Ereignisse, die einer vordefinierten Signalform entsprechen. Alle Sensoreinheiten sind über ein Bussystem mit dem Datenlogger verbunden, um miteinander kommunizieren zu können. Der prinzipielle Aufbau ist in Abbildung 3.1 ersichtlich. Die Stromversorgung der Anlage wird am Datenlogger angeschlossen. Parallel zum Kabel des Datenbusses wird die Stromversorgung der Sensoreinheiten geführt.

Diese drei Einheiten werden im Folgenden genauer definiert.

## 3.2. Datenlogger

Der Datenlogger hat verschiedene Aufgaben zu erfüllen:

- Sammeln und speichern der Messdaten der Sensoreinheiten.
- Kontrolle über das Bussystem.
- Steuerung des Betriebs der Anlage.
- Schnittstelle für die Konfiguration der Anlage und für das Auslesen der Messdaten.

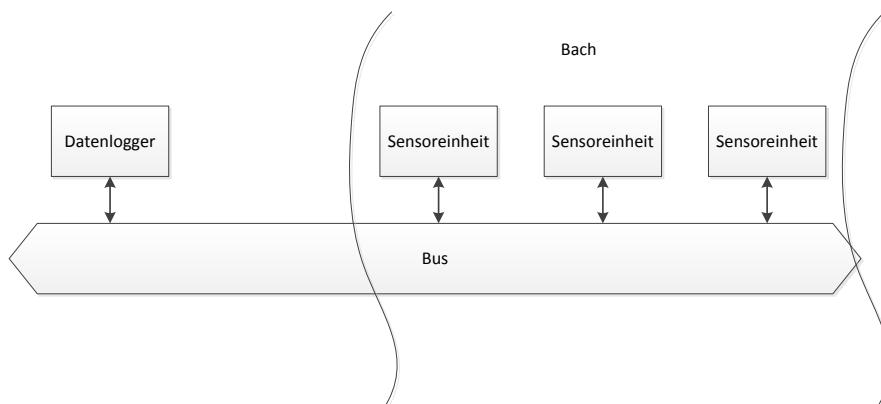


Abbildung 3.1.: Eine Messstation mit einem Datenlogger, der mehrere Sensoreinheiten im Bach steuert.

### 3.2.1. Messdaten sammeln

Für jede angeschlossene Sensoreinheit führt der Datenlogger eine Datensammlung, in der die registrierten Ereignisse zeitlich sortiert abgespeichert werden. Die Datensammlungen werden in Dateien abgelegt, die auf einem leicht auswechselbaren Medium abgespeichert werden. So können die Messdaten auf einfache Art für die weitere Auswertung abgeholt werden.

### 3.2.2. Kontrolle über das Bussystem

Als Busmaster hat der Datenlogger die Aufgabe, allen angeschlossenen Einheiten eine eindeutige Identifikationsnummer (ID) zuzuweisen. Über diese ID erkennt der Datenlogger, von welcher Sensoreinheit Daten übertragen werden. Anhand der ID kann der Datenlogger Konfigurationsnachrichten an bestimmte Sensoreinheiten addressieren. Für die Zuordnung der Messdaten zu einem bestimmten Sensor benötigen die Sensoreinheiten ein fixes Erkennungsmerkmal, z.B. eine Seriennummer, die mit den Messdaten abgespeichert werden soll. Wurde einer Sensoreinheit einmal eine ID zugewiesen, wird diese Zugehörigkeit in einer Konfigurationsdatei auf dem Datenlogger abgespeichert, um bei einem Neustart des Systems die gleichen IDs zu vergeben.

### 3.2.3. Steuerung des Betriebs

Die Messstation hat verschiedene Betriebsmodi, die über den Datenlogger angewählt werden können. Der Datenlogger steuert die einzelnen Sensoreinheiten entsprechend an. Je nach Betriebsmodus werden mehr oder weniger detaillierte Daten über die Ereignisse abgespeichert.

### 3.2.4. Schnittstelle nach Aussen

Über eine Schnittstelle am Datenlogger kann ein Computer angeschlossen werden. Per Kommandozeile wird die Messstation konfiguriert, der Zustand überprüft und der Betriebsmodus gewählt.

## 3.3. Sensoreinheit

Die Aufgaben der Sensoreinheit umfassen:

- Erfassung von Messdaten.
- Erkennung von Ereignissen.
- Übertragung der Ereignisdaten an den Datenlogger.

### 3.3.1. Messdatenerfassung

Der Sensor zur Erfassung der Daten wird mit einer vordefinierten Abtastrate ausgelesen. Die Abtastrate muss so gewählt werden, dass einzelne Ereignisse erkannt werden können, ohne unnötig viel Messdaten zu generieren.

### 3.3.2. Ereigniserkennung

Im Mikroprozessor werden die Messdaten fortlaufend analysiert. Überschreitet das gemessene Signal einen gewissen Schwellenwert (Threshold), markiert dies den Beginn eines Ereignisses. Das Ereignis ist beendet, wenn der Signalpegel für eine gewisse Zeit (Timeout) unterhalb des Threshold bleibt. Für jedes Ereignis wird abgespeichert, wann es aufgetreten ist (Timestamp), wie lange es gedauert hat, wie hoch der Signalpegel maximal ausschlug und wie viele Signalspitzen (Peaks) aufgetreten sind. Allenfalls können auch die Höhen und Timestamps aller Peaks übertragen werden.

### 3.3.3. Datenübertragung

Die Sensoreinheit sendet die Messdaten regelmässig über das Bussystem an den Datenlogger. Nach Bestätigung des Erhalts werden die Messdaten aus dem Speicher der Sensoreinheit gelöscht.

## 3.4. Bussystem

Das Bussystem verbindet die Einheiten der Messstation miteinander. Die gesamten Messdaten und Steuerkommandos werden über den Bus übertragen. Das Bussystem muss die Datenmenge der angeschlossenen Sensoren bewältigen können, über die geforderte Distanz funktionieren und möglichst robust gegenüber äusseren Einflüssen sein. Der Busmaster hat die Möglichkeit, laufende Übertragungen von Sensoreinheiten zu unterbrechen, um Steuerkommandos zu senden.

# 4. Funktionale Anforderungen

## 4.1. Datenlogger (F1...)

### 4.1.1. F110 Busmaster

Der Datenlogger übernimmt die Kontrolle des Bussystem. Bei Inbetriebnahme des Systems tastet der Datenlogger den Bus nach Sensoreinheiten ab und erteilt jeder Sensoreinheit eine eindeutige Identifikationsnummer (ID). Die ID des Datenloggers soll so gewählt werden, dass er jederzeit Priorität hat, auf den Bus zu schreiben.

### 4.1.2. F120 Sensorerkennung

Die angeschlossenen Sensoren werden vom Datenlogger erkannt und mit einer ID versehen. Anhand der ID wird die Priorität bei der Datenübertragung festgelegt und der Sensor identifiziert. Ein Sensor, der bereits am System angeschlossen war, erhält wieder die gleiche ID, sofern die Konfigurationsdatei nicht gelöscht wurde.

### 4.1.3. F130 Uhrzeit

Der Datenlogger verfügt über eine interne Uhr, um die Ereignisse in den Dateien mit einem lesbaren Zeitstempel zu versehen.

### 4.1.4. F140 Timestamp verteilen

Der Datenlogger sendet ein Signal an alle Sensoreinheiten, dass der Zeitstempel (Timestamp) neu gestellt werden soll. Ab dann beziehen sich die Timestamps auf die Dauer seit dem jetzigen Zeitpunkt.

### 4.1.5. F160 Schnittstelle zum Steuerrechner

Der Datenlogger bietet eine Schnittstelle, an der ein Steuerrechner (Laptop, Personal Computer (PC)) angeschlossen werden kann. Über diese Schnittstelle kann der Betrieb der ganzen Anlage gesteuert werden.

### 4.1.6. F170 Steuerung Betriebsmodus

Der Betriebsmodus der Sensoren wird vom Datenlogger aus gesteuert: Wie viele und welche Art von Daten gesammelt werden soll und ob alle Sensoren oder nur bestimmte aktiv sein sollen.

Folgende Betriebsmodi sind verfügbar:

- Normaler Modus: Alle Sensoren übermitteln die verarbeiteten Ereignisdaten. Zeitpunkt, Intensität, Dauer und Anzahl Ausschläge jeden Ereignisses werden gespeichert.
- Detaillierter Modus: Alle Sensoren übermitteln die verarbeiteten Ereignisdaten sowie die gesamten Messdaten für die Dauer des Ereignisses.

- Rohdatenmodus: Ein Sensor übermittelt kontinuierlich Rohdaten, die anderen Sensoren werden vorübergehend abgeschaltet.

#### 4.1.7. F180 Daten sammeln

Der Datenlogger fragt in regelmässigen Abständen bei den Sensoreinheiten an, ob Ereignisdaten zur Übertragung bereit sind. Diese übermitteln die vorliegenden Ereignisdaten.

#### 4.1.8. F190 Daten speichern

Die Daten werden vom Datenlogger auf einer Speicherplatte in Dateien abgelegt. Nach entsprechenden Befehlen vom Steuerrechner kann die Karte entfernt und ausgetauscht werden, um die Daten abzuholen.

### 4.2. Sensoreinheit (F4...)

#### 4.2.1. F410 Ereignisdetektion

Die Sensoreinheit liest den Sensor mit einer definierten Abtastrate aus und wertet die Messdaten aus. Der Prozessor erkennt Ereignisse anhand definierter Kriterien. Zu jedem Ereignis werden folgende Daten gespeichert: Zeitpunkt (Timestamp), Dauer, Anzahl Peaks und höchster Peak. In einem zweiten Betriebsmodus können alle Messpunkte während eines Ereignisses gespeichert werden.

#### 4.2.2. F430 Datenübertragung

Die Sensoreinheit übermittelt die Ereignisdaten über das Bussystem an den Datenlogger.

#### 4.2.3. F450 Rohdatenaufzeichnung

In einem Sondermodus werden alle Messpunkte gespeichert und über das Bussystem an den Datenlogger übertragen. In diesem Betriebsmodus kann, darf auch nur eine Sensoreinheit aktiv sein, die anderen werden auf Standby geschaltet.

## 5. Nichtfunktionale Anforderungen

- Die gesamte Messstation soll eine geringere Leistungsaufnahme haben als eine aktuelle Messstation mit Geophonen. Für zehn Geophone sind dies zur Zeit ungefähr zehn Watt.
- Die Installation soll weniger bauliche Massnahmen erfordern als eine aktuelle Messstation mit Geophonen.
- Die erfassten Ereignisdaten sollen mindestens so detailliert sein wie von den bisherigen Installationen.
- Sensoreinheiten müssen wasserdicht verpackt werden können.

# 6. Tests

Dieses Kapitel listet die Testfälle auf, die zu Beginn des Projekts definiert wurden. Für jeden Testfall sind, wo nötig, Vorbedingungen definiert. Ein Testfall beschreibt Aktionen, die vorgenommen werden müssen sowie das zu erreichende Resultat.

Die Testergebnisse sind am Ende jedes Testfalls angegeben.

## 6.1. Testaufbau

Die Verifikation der Sensoreinheiten macht einen Testaufbau nötig. Da das finale Produkt in einem Bach verbaut werden soll und die dort verwendeten Stahlplatten eine gewisse Grösse und aufgrund der Dicke von 15mm auch ein erhebliches Gewicht aufweisen, wurde zu Testzwecken ein kleinerer, einfacherer Aufbau realisiert. Statt einer Stahlplatte wurde eine Aluminiumplatte verwendet, was zur Folge hat dass bereits schwache Einschläge Signale generieren (Abbildung 6.1). Der Sensor ist unter der Aluminiumplatte verschraubt (Abbildung 6.2).

## 6.2. Datenlogger

### 6.2.1. T110 Busmaster

**Aktionen** Die Messstation wird an die Stromversorgung angeschlossen.

**Resultate** Der Datenlogger übernimmt die Kontrolle über den CAN-Bus und vergibt jedem angeschlossenen Sensor die eindeutige CAN-ID aus der Konfigurationsdatei.

**Testergebnis** Die Sensoreinheiten haben die vordefinierte CAN-ID erhalten. Der Test ist erfüllt.

### 6.2.2. T120 Sensorerkennung

**Aktionen** Eine neue Sensoreinheit wird an die Messstation angeschlossen, deren Seriennummer nicht in der Konfigurationsdatei erfasst ist. Die Messstation wird an die Stromversorgung angeschlossen.

**Resultate** Der Datenlogger erkennt die neue Sensoreinheit und teilt ihr eine eindeutige CAN-ID zu.

**Testergebnis** Die Sensoreinheit hat eine bisher unbenutzte, unreservierte CAN-ID erhalten. Der Test ist erfüllt.

### 6.2.3. T130 Uhrzeit

**Aktionen** Die Uhrzeit wird im laufenden Betrieb richtig eingestellt. Die Messstation wird während mind. 12 Stunden in Betrieb gehalten.

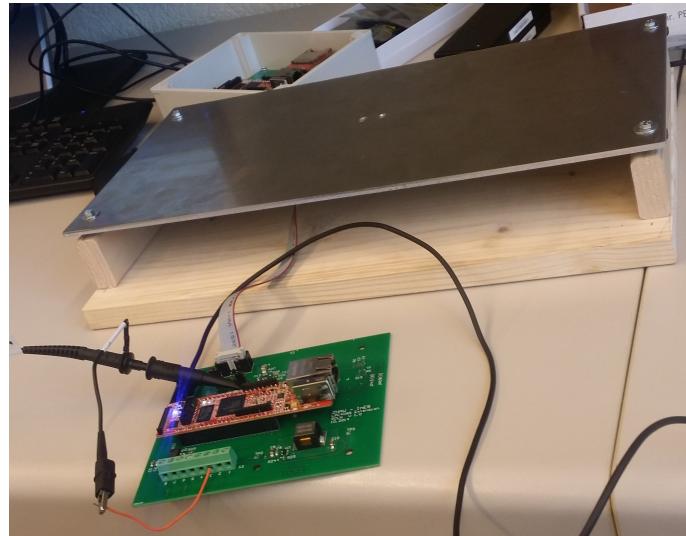


Abbildung 6.1.: Übersicht des Testaufbaus.

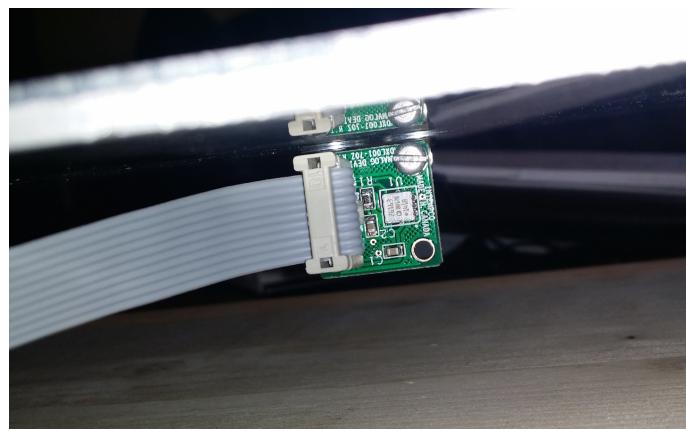


Abbildung 6.2.: Montage des Sensors im Testaufbau.

**Resultate** Die interne Uhrzeit weist nicht mehr als 2 Sekunden Fehlgang innert 12 Stunden auf.

**Testergebnis** Bis zur Berichterstellung konnte dieser Test noch nicht durchgeführt werden.

#### 6.2.4. T140 Timestamp zurücksetzen

**Aktionen** Über die Konfigurationsschnittstelle wird der Befehl zum Zurücksetzen der Timestamps in den Sensoreinheiten gegeben.

**Resultate** Alle Sensoreinheiten führen den Befehl innert weniger Sekunden aus und senden ab dann die Ereignisse mit dem neuen Timestamp.

**Testergebnis** Bis zur Berichterstellung konnte dieser Test noch nicht durchgeführt werden.

#### 6.2.5. T160 Schnittstelle zum Steuerrechner

**Aktionen** Ein Computer wird an die Konfigurationsschnittstelle angeschlossen und ein Terminal-Emulator gestartet.

**Resultate** Das Konfigurationsmenü erscheint in der Terminal-Emulation und kann über Tastatureingaben bedient werden.

Die Eingaben werden vom Datenlogger ausgeführt und bei Bedarf an die Sensoreinheiten kommuniziert.

**Testergebnis** Das Konfigurationsmenü funktioniert wie vorgesehen. Die Einstellungen werden an die Sensoreinheiten übertragen.

#### 6.2.6. T170 Steuerung Detail-Level

**Aktionen** Am Datenlogger wird für eine Sensoreinheit ein anderer Detail-Level eingestellt.

**Resultate** Die Sensoreinheit wechselt den Detail-Level bei der Übertragung neuer Ereignisse.

**Testergebnis** Bis zur Berichterstellung konnte dieser Test noch nicht durchgeführt werden.

#### 6.2.7. T180 Daten sammeln und speichern

**Aktionen** Mehrere Sensoreinheiten senden Ereignisdaten an den Datenlogger.

**Resultate** Der Datenlogger legt für jede Sensoreinheit eine Datei an und speichert die Ereignisdaten in der richtigen Datei.

**Testergebnis** Bis zur Berichterstellung konnte dieser Test noch nicht durchgeführt werden.

## 6.3. Sensoreinheit

### 6.3.1. T400 Konfiguration

**Aktionen** Via Konfigurationsmenü des Datenloggers werden die Parameter des Sensors auf neue Werte gesetzt.

**Resultate** Die Sensoreinheit übernimmt die neuen Einstellungen und passt die Ereigniserkennung entsprechend an.

**Testergebnis** Die Konfiguration wurde erfolgreich auf die Sensoreinheit übertragen. Die Konfiguration vor dem Test ist in Listing 6.1 gegeben, mit der Eingabe 99 werden alle Sensoren ausgewählt. Listings 6.2 und ?? zeigen, wie der zu konfigurierende Parameter ausgewählt und der neue Wert eingegeben wird. Am Datenlogger wird im Testmodus das Absenden einer CAN-Nachricht protokolliert, Listing 6.4. Die Sensoreinheiten geben im Testmodus ebenfalls eine Bestätigung aus (Listings 6.5 und 6.6), wenn sie eine neue Konfiguration erhalten. Die Änderung des Thresholds wurde also von beiden Sensoren übernommen.

```

1 Listing sensor config
2 Nr SID serial fs threshold baseline timeout detail started?
3 0) 2 061bfdf6 100 200 2047 30 sparse started
4 1) 3 15117738 100 150 2040 30 peaks only started
5 #) Select a sensor from the list.
6 99) Select all sensors.
7 0) cancel
8 >
9 your entry was: 99

```

Listing 6.1: T400 Vorbedingung und Auswahl aller Sensoren

```

1 1) set sampling rate
2 2) set threshold value
3 3) set baseline value
4 4) set timeout
5 5) set detail level
6 6) start or stop recording
7 0) exit
8 >
9 your entry was: 2

```

Listing 6.2: T400 Auswahl des Parameters

```

1 #) Enter threshold value.
2 baseline + threshold must not exceed 4096
3 and
4 baseline - threshold must not be below 0
5 0) cancel
6 >
7 your entry was: 250

```

Listing 6.3: T400 Eingabe des neuen Thresholds

```

1 prepare message id: 5020101
2 OK
3 prepare message id: 50301001
4 OK

```

Listing 6.4: T400 Sendeprotokoll am Datenlogger

```

1 threshold fa    -> geänderter Wert, von 150 (0x96) auf 250 (0xfa)
2 fsampling 64   -> wie zuvor, bei 100
3 timeout 1e     -> wie zuvor, bei 30
4 baseline 7f8   -> wie zuvor, bei 2040

```

Listing 6.5: T400 Konfiguration aus Sensor 3

```

1 threshold fa    -> geänderter Wert, von 200 (0xc8) auf 250 (0xfa)
2 fsampling 64   -> wie zuvor, bei 100
3 timeout 1e     -> wie zuvor, bei 30
4 baseline 7ff   -> wie zuvor, bei 2047

```

Listing 6.6: T400 Konfiguration aus Sensor 2

### 6.3.2. T410 Ereignisdetektion

**Aktionen** Die Sensoreinheit ist im Messbetrieb. Am Sensor wird ein Einschlag simuliert, indem ein Objekt auf die Metallplatte geschlagen wird.

**Resultate** Am Ausgang des Sensors misst ein Oszilloskop den Spannungsverlauf. Die so gemessene Referenzkurve wird mit den von der Sensoreinheit ausgegebenen Daten verglichen.

**Testergebnis** Bis zur Berichterstellung konnte dieser Test noch nicht durchgeführt werden.

### 6.3.3. T430 Datenübertragung

**Aktionen** Mehrere Einschläge werden simuliert und mit einem Oszilloskop aufgezeichnet, um Referenzdaten zu haben.

**Resultate** Die Sensoreinheit überträgt die Daten von Test T410 fehlerfrei an den Datenlogger.

**Testergebnis** Bis zur Berichterstellung konnte dieser Test noch nicht durchgeführt werden.

### 6.3.4. T450 Rohdatenaufzeichnung

**Aktionen** Eine Sensoreinheit wird in den Rohdatenmodus versetzt.

**Resultate** Die Sensoreinheit zeichnet Rohdaten während 60 Sekunden ohne einen Puffer-Überlauf auf und überträgt die Daten an den Datenlogger.

**Testergebnis** Bis zur Berichterstellung konnte dieser Test noch nicht durchgeführt werden.

## 6.4. Nichtfunktionale Tests

### 6.4.1. T710 Stromverbrauch

**Aktionen** Der Stromverbrauch der Messanlage mit dem Datenlogger und drei Sensoreinheiten wird gemessen.

**Resultate** Der Stromverbrauch sollte unter 3 Watt liegen.

**Testergebnis** Ein Datenlogger und zwei Sensoreinheit haben einen Stromverbrauch von

# 7. Hardware

## 7.1. Hardware-Architektur

Anhand der funktionalen Vorgaben für die Messstation werden der Datenlogger, die Sensoreinheit und das Bussystem im folgenden genauer spezifiziert und die Komponenten ausgewählt.

### 7.1.1. Datenlogger

Das Hardware-Konzept des Datenloggers ist in Abbildung 7.1 dargestellt. Der Datenlogger sammelt die Daten der Sensoreinheiten über das Bussystem ein und speichert sie ab. Dafür benötigt er das Bussystem, einen Mikrokontroller, einen internen Speicher und ein leicht auswechselbares Speichermedium. Außerdem soll über eine Schnittstelle ein Computer angeschlossen werden können, um den Betrieb der Messstation zu steuern. Der Datenlogger wird in einem wasserdichten Gehäuse untergebracht. Für den Austausch des Speichermediums wäre eine verschraubbare Öffnung denkbar.

### 7.1.2. Sensoreinheit

Die Sensoreinheit benötigt einen Beschleunigungssensor, um die Einschläge von Geschiebe zu messen. Über einen Analog-Digital-Wandler (A/D-Wandler, Englisch Analog Digital Converter (ADC)) werden die Messsignale digitalisiert. Die gemessenen Signale werden von einem Mikrokontroller verarbeitet, im internen Speicher zwischengespeichert und über das Bussystem an den Datenlogger übertragen. Abbildung 7.2 zeigt das Hardware-Konzept der Sensoreinheit.

### 7.1.3. Bussystem

Das Bussystem muss die Daten und Befehle zwischen Datenlogger und Sensoreinheiten übertragen. Die Reichweite des Bussystems muss genügen, um alle Komponenten der Messinstallation zu verbinden. Die Datenbandbreite muss die Übertragung der Messresultate aller Sensoren erlauben.

## 7.2. Komponentenauswahl

### 7.2.1. Mikroprozessor

Bei der Auswahl des Mikrocontrollers werden folgende Kriterien berücksichtigt:

- Genügend Rechenleistung für allfällige zusätzliche Anforderungen.
- A/D-Wandler mit genügender Abtastrate und Auflösung (Bit-Breite).
- DSP integriert für die schnelle Verarbeitung der Messdaten.
- Nested Vectored Interrupt Controller (NVIC).

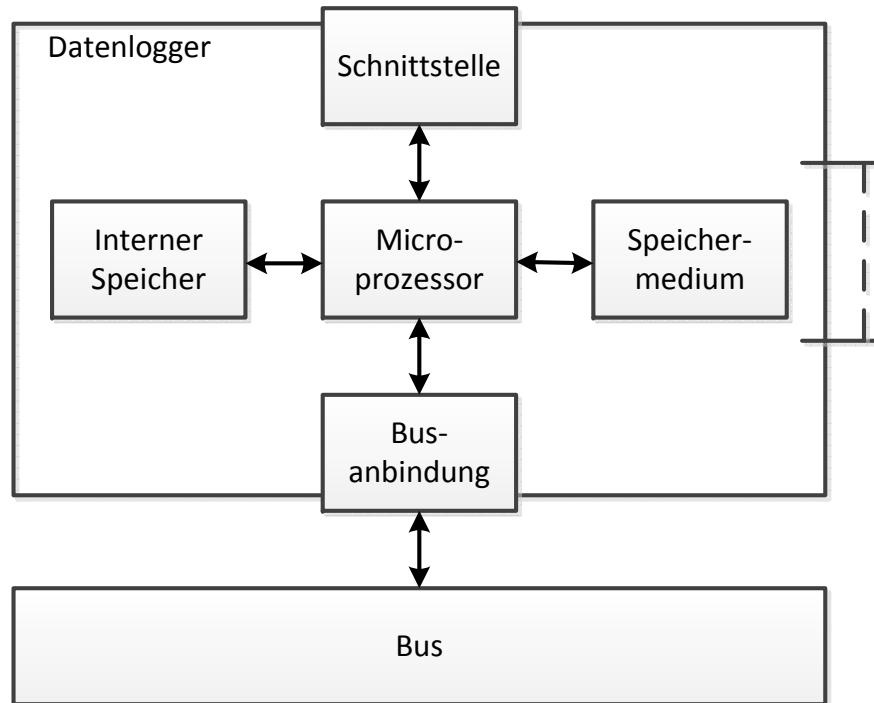


Abbildung 7.1.: Hardwarekonzept des Datenloggers.

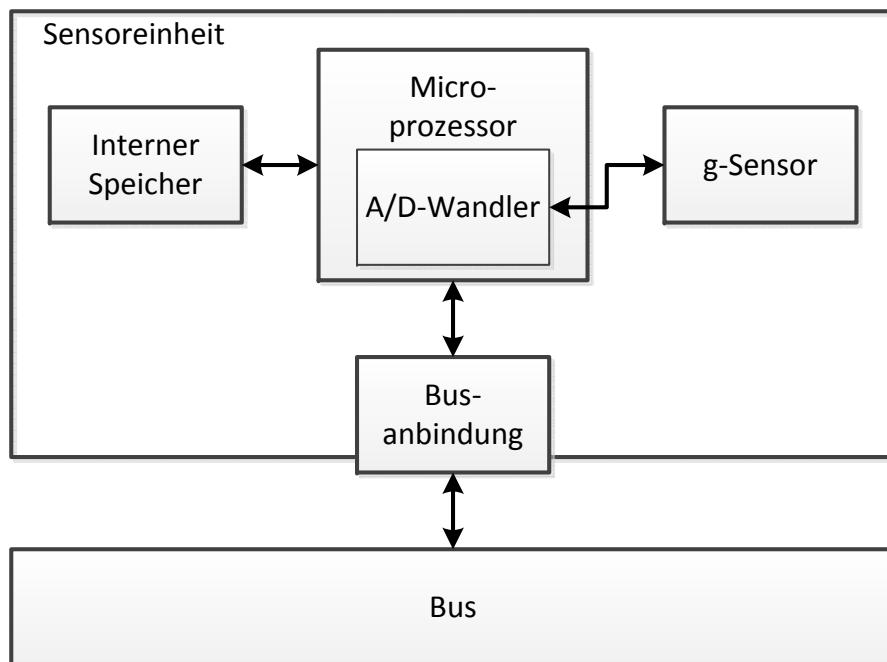


Abbildung 7.2.: Hardwarekonzept der Sensoreinheit.

- Ein-/Ausgänge für das Bussystem.
- Ein-/Ausgänge für den externen Speicher.
- möglichst geringer Stromverbrauch.

Für die vorliegende Anwendung eignen sich Mobile-Prozessoren sehr gut. Sie sind für den Einsatz in mobilen Geräten konzipiert, d.h. für den Batteriebetrieb, sind aber trotzdem sehr leistungsfähig. Die ARM Cortex-Reihe bietet ein breites Spektrum an Prozessoren an. Im Mobile-Segment der ARM Cortex-Reihe sind vier Prozessoren erhältlich. Da nur in einem Modell ein Digitaler Signal-Prozessor (DSP) integriert ist, fällt die Entscheidung leicht. Der ARM Cortex-M4 ist auch der neueste Prozessor aus dem Mobile-Segment. Mit dem Ausblick, das Messsystem in einem zukünftigen Projekt zur Serienreife zu bringen, macht es nur Sinn, den neuesten Prozessor zu verwenden. Ein Auszug aus dem Datenblatt des ARM Cortex-M4 befindet sich im Anhang A.4.1. Abbildung 7.3 zeigt die Fähigkeiten des ARM Cortex-M4.

**Nested Vectored Interrupt Controller** Ein Prozessor mit NVIC kann auf verschiedene Ereignisse reagieren, indem Interrupts ausgelöst werden. Jedem Interrupt kann eine Priorität zugewiesen werden, um festzulegen, ob ein Interrupt einen anderen unterbrechen darf, der gerade vom Prozessor abgearbeitet wird. Für das Messsystem wird ein NVIC benötigt, da mehrere zeitkritische Prozesse parallel ablaufen sollen. Einerseits muss die Abtastrate der Messwerterfassung genau eingehalten werden. Andererseits darf die Verarbeitung der Messwerte nicht zu lange unterbrochen werden, um einen Überlauf der Queue zu vermeiden. Die Daten der Ereignisse müssen parallel dazu an den Datenlogger übermittelt werden. Die Prioritäten dieser Prozesse müssen richtig gewählt werden. Im Abschnitt 8.1.1 ff. wird näher darauf eingegangen.

**DSP** Der ARM Cortex-M4 verfügt über DSP-Funktionen, eine sog. single-cycle multiply-accumulate unit (MAC). In einer single-cycle MAC können Multiplikationen in einem einzigen Prozessorzyklus ausgeführt werden. Normalerweise benötigt ein Prozessor für eine Multiplikation bis zu mehreren zehn Zyklen, bis das Resultat vorliegt. Mit einem DSP ist es daher möglich, z.B. eine Filterfunktion viel effizienter zu berechnen als mit einem normalen Prozessor, da für diese viele Multiplikationen und Additionen ausgeführt werden müssen.

**Floating Point Unit (FPU)** Eine FPU führt Berechnungen mit Dezimalbrüchen sehr rasch aus. Für unsere Anwendung ist eine FPU keine Voraussetzung. Falls in einem späteren Projekt komplexere Filter oder andere Anwendungen berechnet werden müssen, könnte die FPU aber ein Vorteil sein.

**Wake Up Interrupt Controller Interface** Ein Wake Up Interrupt Controller Interface ermöglicht es, einen Prozessor in einen Stromsparmodus zu versetzen und ihn durch ein definiertes Signal wieder zu wecken. Damit ist es möglich, eine Sensoreinheit praktisch ganz abzuschalten, wenn sie keine Messungen durchführt. Durch eine Nachricht über das Bussystem kann die Sensoreinheit wieder eingeschaltet werden. Der Cortex-M4 verfügt über 240 mögliche Wake Up Interrupts, kann also für 240 Ereignisse programmiert werden, die ihn aufwecken oder in einen Stromsparmodus senden können (vgl. [6]).

**Rechenleistung** Die Rechenleistung des ARM Cortex-M4 hängt von der Implementation ab. Die Firma ARM produziert den ARM Cortex-M4 nicht selbst, sondern lizenziert Chip-Hersteller für die Verwendung der Architektur in ihren Prozessoren. Vom ARM Cortex-M4 sind mehrere Ausführungen erhältlich. Verschiedene Chip-Hersteller implementieren eine Architektur des ARM Cortex-M4 in ihren Prozessoren.

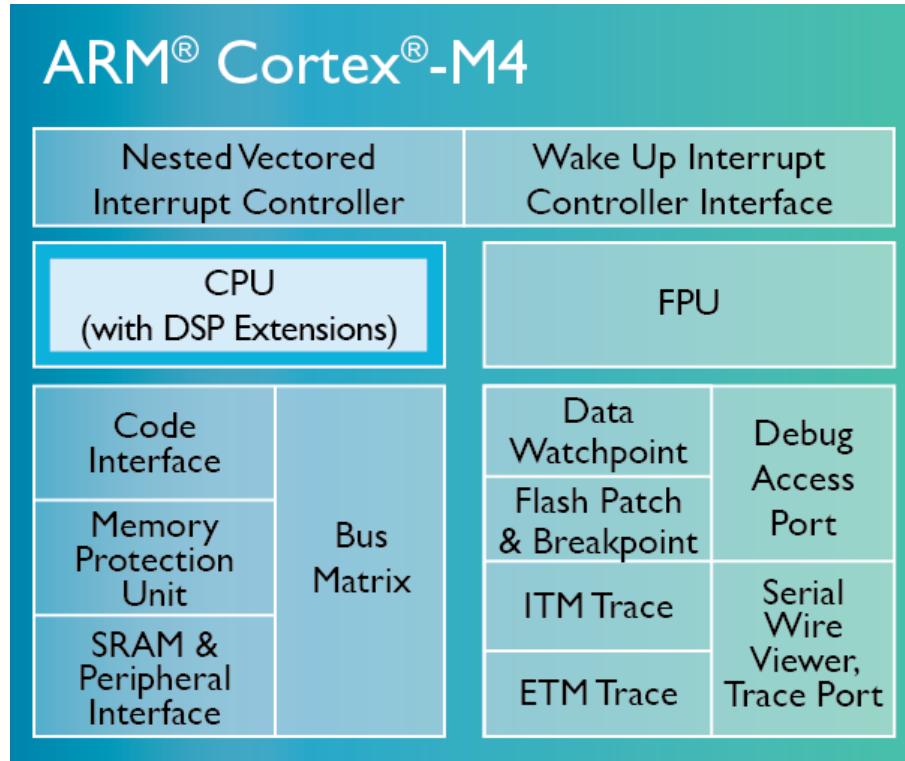


Abbildung 7.3.: Chipdiagramm der *ARM Cortex-M4* Architektur [6].

**A/D-Wandler** Bei den bestehenden Messstationen wird die Datenmenge stark reduziert, indem über eine Minute ein Histogramm mit den Peakintensitäten als Klassen berechnet wird. Die Intensitäten werden in 18 Klassen logarithmischer Abstufung eingeteilt. Das entspricht einer Bit-Breite von etwas mehr als vier Bit (4 Bit ermöglichen 16 Werte). Da für die Klasseneinteilung eine logarithmische Abstufung gewählt wurde, muss ein linearer A/D-Wandler trotzdem eine höhere Bit-Breite als vier aufweisen, um die gleiche Auflösung wie in den unteren logarithmischen Klassen zu erreichen. Mit 12 Bit Auflösung sind 4096 Stufen unterscheidbar, was für diese Anwendung genügt.

**Peripherie** Damit der Prozessor über das Bussystem kommunizieren und auf ein externes Speichermedium schreiben kann, sind genügend Ein- und Ausgabe-Pins nötig.

**Wahl eines Prozessors** Da der Entscheid für eine Hardware schon zu Beginn des Projekts gefällt werden musste, wurde auf eine grosszügige Sicherheitsmarge in Sachen Rechenleistung und Bit-Breite geachtet. Um Kosten und Baugröße der Sensoreinheit klein zu halten, suchten wir nach einem Evaluationsboard mit *ARM Cortex-M4* Prozessor. Das *LPC4088 QuickStart Board* von *NXP Semiconductors* hat genügend Arbeitsspeicher für den Prozessor, verfügt über die benötigten Pins für die Peripherie und hat bei weitem genügend Rechenleistung. Die Fähigkeiten des NXP *LPC4088FET208* Prozessors sind in Tabelle 7.1 dargestellt.

Auf dem *NXP LPC4088 QuickStart Board* sind zusätzliche Bauteile verbaut, z.B. Speicher (SDRAM und Flash-Speicher) und Pin-Stekkleisten, um CAN-Bus-Pins oder A/D-Eingänge anzuschliessen. Tabelle 7.2 listet die für dieses Projekt relevanten, zusätzlichen Eigenschaften auf.

### 7.2.2. Bus-System

Das Bussystem für die Messanlage muss die folgenden Kriterien erfüllen:

Taktfrequenz	bis 120 MHz
NVIC	vorhanden
FPU	vorhanden
Programmspeicher	512 kByte
Arbeitsspeicher (intern)	96 kByte
CAN-Bus	2
USB	2
SD-Card	Anschlüsse vorhanden
A/D-Wandler	8 Eingänge, 12 Bit

Tabelle 7.1.: Fähigkeiten des NXP LPC4088 Prozessors [7].

Prozessor	NXP LPC4088FET208
Taktfrequenz	bis 120 MHz
Flash-Speicher	8 MByte
SDRAM	32 MByte
A/D-Wandler	6 Eingänge nutzbar, 12 Bit

Tabelle 7.2.: Zusätzliche Fähigkeiten des NXP LPC4088 QuickStart Boards von *Embedded Artists* [8].

- Übertragungsbandbreite genügend für fortlaufende Übertragung von Rohdaten einer Sensoreinheit.
- Reichweite mindestens 20 Meter.
- Robust gegenüber äusseren Einflüssen.
- Mindestens zwanzig Busteilnehmer möglich.

In Tabelle 7.3 sind die Eigenschaften diverser Bussysteme aufgeführt.

**Kommentare** SPI und I2C sind nur für kurze Distanzen geeignet und sind deshalb keine Option. Die Verwendung von Ethernet zur Datenübertragung würde zwei Schnittstellen auf jeder Sensoreinheit voraussetzen, um die Sensoren hintereinander zusammenzuhängen (Daisy Chain). Jedes Paket müsste vom Mikrokontroller weitergeleitet werden, wenn es für einen anderen Empfänger bestimmt ist. Dies führt zu einer zusätzlichen Belastung der Microcontroller. Stromversorgung über Ethernet ist mit PowerOverEthernet (PoE) zwar möglich, erfordert aber spezielle Geräte zur Speisung über den Stecker des Datenkabels. Dies verunmöglicht eine Daisy Chain mit PoE, neben dem Datenkabel wäre noch ein Kabel für die Stromversorgung notwendig.

**Vergleich CAN-Bus und RS485** Die beiden Bussysteme CAN und RS485 sind nicht einfach zu vergleichen, da der RS485-Standard nur die elektrischen Eigenschaften des Systems beschreibt (OSI Layer 1). Der CAN-Standard beschreibt auch den Data Link Layer (OSI Layer 2). Der Data Link Layer beschreibt Methoden, die die Übertragung zuverlässig machen. Ein Beispiel dafür ist eine Prüfsumme, die es ermöglicht, eine fehlerhafte Übertragung im Empfänger festzustellen. Der Empfänger bestätigt bei korrekter Prüfsumme den Empfang. Stellt einer der Empfänger einen Fehler fest, sendet er eine Fehlermeldung über den Bus und stört damit die Übertragung. So ist es nicht möglich, dass einige Empfänger die Nachricht lesen konnten und andere nicht. Der Sender ist bei erfolgter Bestätigung sicher, dass seine Nachricht erfolgreich an alle Empfänger übertragen wurde (vgl. [9]).

CAN-Bus definiert auch eine Adressierung, Kollisions-Erkennung und -Auflösung, Prioritäten der Busteilnehmer und ein Nachrichtenformat. Bei RS485 besteht eine Nachricht aus einem einzelnen Zeichen. Der Data Link Layer muss gänzlich in Software gelöst werden. Dafür ist es möglich, das Protokoll komplett selbst zu definieren. Dies erlaubt beliebig lange Übertragungen von einem Busteilnehmer.

	<b>Bitrate</b>	<b>Distanz</b>	<b>Clients</b>	<b>Besonderheiten</b>
<b>CAN</b>	1 MBit/s 125 kBit/s	40 m 500 m	> 20	+ Collision Detection (CD) umgehen mit Polling durch Master. + Bei synchronem CAN wird CD durch ID gelöst. + CAN Controller sendet Interrupt Request bei erhaltener Nachricht.
<b>SPI</b>	..100 MBit/s	< 1 m	slave select	- Pro Client eine Slave Select Leitung - alternativ: Daisy Chain ⇒ alle Mikrokontroller (MC) beschäftigt. - Bei Ausfall eines MC ganzer Bus unterbrochen.
<b>RS485</b>	35 MBit/s 100 kBit/s	10 m 1200 m	>32	- Master am besten in der Mitte des Bus ⇒ ungünstig. - Braucht 2–4 Drähte (bei Full Duplex) - braucht pull-up und pull-down Widerstände ⇒ mehr Leistungsaufnahme.
<b>Ethernet</b>	100 MBit/s	100 m	> 20	+ Stromversorgung bei Power over Ethernet (PoE) integriert. - kein Bus sondern allenfalls Daisy Chain. - bei Daisy Chain kein PoE möglich.
<b>Feldbus</b>				ist eine Familie von Bussen, z.B. CAN-Bus
<b>I2C</b>	0.4..5 Mbit/s	wenige Meter	< 20	nur für kurze Distanzen, Bitrate nimmt mit zunehmender Distanz rasch ab.

Tabelle 7.3.: Entscheidungsmatrix für die Auswahl des Bussystems.

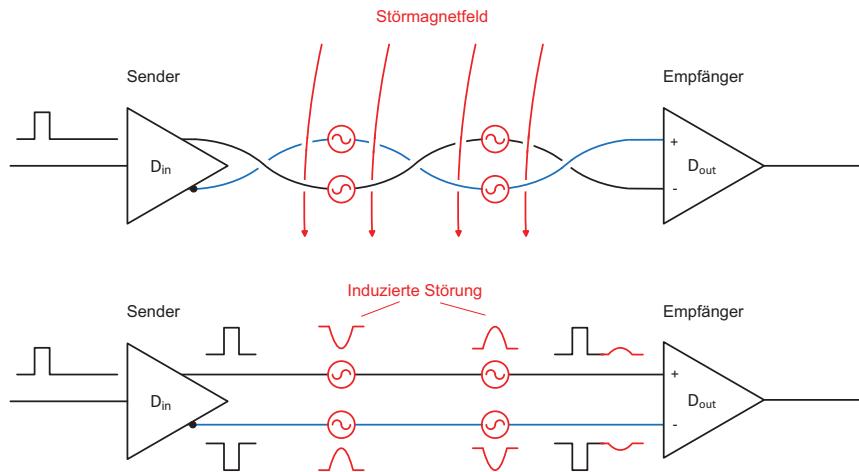


Abbildung 7.4.: Differentielle Leitung [11].

CAN-Bus limitiert die Nachrichtenlänge auf 8 Bytes. Die Übertragung längerer Nachrichten muss über Software geregelt werden (vgl. [9]).

Bei beiden Standards ist der Stecker nicht definiert. Das lässt die komplette Freiheit für die Wahl eines wasserdichten Steckverbinders.

Da der CAN-Bus bereits mit dem Standard viele benötigte Merkmale mitbringt, fällt die Entscheidung nicht schwer.

**Entscheidung** CAN-Bus erfüllt alle Kriterien und erlaubt es, den Busmaster am Ende des Bus zu platzieren. Dies ist ein weiterer Vorteil gegenüber RS485, wo der Master in der Mitte platziert werden sollte. CAN-Bus bietet bereits Collision Detection und Fehlererkennung, während dies bei RS485 in der Software gelöst werden muss. Für CAN-Bus sind Bus-Treiber (Transceiver) erhältlich, die mit hohen Spannungen umgehen können, was das Bussystem robuster gegenüber Umwelteinflüssen macht. Die Grösse der Datenpakete ist bei CAN-Bus auf 8 Byte begrenzt, bei RS485 werden die Datenpakete über die Software frei definiert, was einen Vorteil von RS485 darstellt. Insgesamt überwiegen die Vorteile von CAN-Bus klar.

**CAN-Bus** CAN-Bus ist ein Bussystem über eine differentielle Leitung. Auf zwei Drähten liegt eine bestimmte Spannung an. Der Empfänger misst die Spannungsdifferenz zwischen den beiden Drähten. Die Differenz wird vom Sender entweder klein oder gross gehalten, um ein 'dominantes' oder ein 'rezessives' Bit zu senden. 'Rezessiv' bedeutet dabei, dass ein anderer Busteilnehmer durch Anlegen eines 'dominannten' Bits das 'rezessive' Bit überschreiben kann. Welche Spannungsdifferenzen 'dominant' und 'rezessiv' darstellen, ist vom Standard nicht definiert und lässt dem Entwickler damit freie Hand, die elektrischen Eigenschaften seiner CAN-Implementation seinen Bedürfnissen entsprechend zu definieren. Um Signalreflexionen am Ende der Leitungen zu vermeiden, wird ein Terminierungswiderstand an beiden Enden benötigt, der die beiden Leitungen abschliesst. (vgl. [10]).

Die beiden Drähte der differentiellen Leitung sind verdrillt. Dadurch wird das Signal der Leitung besser vor äusseren Einflüssen geschützt. Elektrische und magnetische Felder induzieren in einem Draht einen Strom. Da das äussere Feld aber in beiden Drähten praktisch den gleichen Strom induziert, ändert sich an der Spannungsdifferenz in den Drähten kaum etwas (siehe Abbildung 7.4). Die beiden Drähte stellen auch eine Spule dar. Durch das Verdrillen der Drähte ändert sich die Ausrichtung der Spule im äusseren Feld auf kurzen Abständen, induzierte Störungen heben sich so gegenseitig auf (vgl. [11, Kap. 2, S. 14]).

Die Kollisions-Erkennung und -Auflösung wird über ID-Nummern der Teilnehmer gelöst. Vor dem Start einer Übertragung prüft der Teilnehmer, dass der Bus zur Zeit frei ist. Dann sendet der Teilnehmer seine ID-Nummer. Falls zwei Teilnehmer gleichzeitig zu senden beginnen, werden sie irgendwann unterschiedliche Bits senden. Der Teilnehmer, der dann das 'dominante' Bit sendet, liest vom Bus den gleichen Wert, den er gerade sendet. Für ihn ist die Übertragung nicht gestört. Der Teilnehmer mit dem 'reziproken' Bit liest aber das 'dominante' Bit des anderen Senders und muss seine Übertragung sofort abbrechen. Diese Art der Kollisionsauflösung hat den Vorteil, dass keine Übertragungszeit verloren geht, da einer der Teilnehmer seine Nachricht senden darf. Der Verlierer wartet, bis der Bus wieder frei ist und probiert dann erneut, die Nachricht zu senden (vgl. [10]).

Die korrekte Übertragung wird mittels eines Cyclic Redundancy Check (CRC)-Codes überprüft. Der Empfänger berechnet bereits während dem Empfang der Nachricht die CRC-Prüfsumme. Sobald die vom Sender mitgeschickte CRC-Prüfsumme übertragen ist, kann der Empfänger diese mit der berechneten CRC vergleichen. Stimmen die Prüfsummen überein, bestätigt der Empfänger die korrekte Übertragung (vgl. [10]).

### 7.2.3. Speichermedium

**Kriterien** Das externe Speichermedium soll möglichst klein sein, wenig Stromverbrauch haben und einfach auswechselbar sein. Bei Inaktivität sollte das Medium wenn möglich keinen Strom verbrauchen. Für einen mehrwöchigen unabhängigen Betrieb einer Messstation muss genügend Speicherkapazität bereitgestellt werden.

**Datenmenge** Pro Sensor werden bei hohem Geschiebeaufkommen maximal hundert Ereignisse pro Sekunde erwartet. Ein solches Geschiebeaufkommen stellt jedoch die Ausnahme dar. Ein Ereignis benötigt je nach verlangtem Detailgrad und Dauer des Ereignisses 10–200 Byte Speicherplatz. Für den normalen Betriebsmodus werden 50 Byte/Ereignis gerechnet, bei 5 Ereignissen pro Sekunde. Damit ergibt sich eine Datenrate von 250 Byte/s, die es pro Sensor abzuspeichern gilt. Mit zehn Sensoren im Einsatz müssen 2.5 kByte/s gespeichert werden.

**Unabhängige Betriebsdauer** Pro Gigabyte Speicherplatz können 111 Stunden Daten für zehn Sensoren gespeichert werden. Bei hohem Geschiebeaufkommen mit zwanzig Mal mehr Ereignissen bleiben immer noch 5 Stunden Aufzeichnungszeit pro Gigabyte. Begrüßt man sich mit weniger Details, fallen pro Sensor in zehn Sekunden rund 400 Byte Daten an. Bei dieser Datenrate reicht ein Gigabyte für rund 700 Stunden. Auch bei hohem Geschiebeaufkommen kann die Anlage über mehrere Tage Daten speichern.

**Kapazität** Heute sind Speichermedien mit Kapazitäten bis über 128 GB erhältlich, so dass die Detailrate kein entscheidendes Kriterium mehr darstellt.

**Datentransfer** Für den Transfer der Daten aus dem Datenlogger auf einen Computer gibt es grundsätzlich zwei Varianten. Entweder man liest die Daten über eine Schnittstelle auf den Computer aus, oder man tauscht das Speichermedium aus. Das Auslesen via Schnittstelle benötigt zusätzlich Strom, das Wechseln des Speichermediums setzt einen mehr oder weniger komfortablen und trotzdem wasserdichten Zugang zum Medium voraus. Da heute Speichermedien mit kleinem Platzbedarf erhältlich sind, kann ein solcher Zugang recht einfach mit einem Schraubverschluss realisiert werden.

**Vergleich** In Tabelle 7.4 werden verschiedene Speichermedien miteinander verglichen. In der Spalte 'Breite' ist aufgelistet, wie gross eine Öffnung mindestens sein muss, um das Speichermedium wechseln zu können. 'Pins' gibt an, wie viele Leitungen für den Anschluss des Mediums am Microcontroller nötig sind. Der Stromverbrauch in Klammern gilt für den Standby-Modus des Speichermediums.

	Breite	Pins	Stromverbrauch	Bemerkungen
<b>SD-Card</b>	24 mm	9	20..100 mA (0.2 mA)	4 bit breiter serieller Bus
<b>CompactFlash</b>	43 mm	50	max. 70 mA (k.A.)	paralleler Bus
<b>USB-Stick</b>	min. 12 mm	4	typ. 70 mA (k.A.)	

Tabelle 7.4.: Entscheidungsmatrix zur Auswahl des Speichermediums [12–14].

**Entscheid** Für einen verschraubbaren Verschluss ist die CompactFlash-Karte zu breit, das Gehäuse würde dadurch sehr gross. Die SD-Karte und der USB-Stick sind vergleichbar in der Grösse. Von der SD-Karte sind auch kleinere Varianten erhältlich. Eine Öffnung für den Austausch des Speichermediums kann eine gewisse Grösse ohnehin nicht unterschreiten, damit hineingegriffen werden kann. Da die SD-Karte im Standby den geringeren Stromverbrauch hat, wird der Datenlogger mit einem SD-Kartenleser ausgestattet.

#### 7.2.4. Sensor

auswahl beschrei-

#### 7.2.5. Schnittstelle

Das gewählte *NXP LPC4088 QuickStart Board* verfügt über einen Universal Serial Bus (USB)-Anschluss, über den eine serielle Schnittstelle angesprochen werden kann. Für eine einfache Kommandozeile oder ein Konfigurationsmenü ist diese Schnittstelle ausreichend. Die Schnittstelle kann mit einer Übertragsrate von 9600 bis 115200 Baud betrieben werden.

#### 7.2.6. Gehäuse

Um den Datenlogger und die Sensoreinheiten wasserdicht zu verpacken, wurden Gehäuse und Komponenten mit der Schutzklasse IP68 gesucht. Die wasserdichten Gehäuse der Firma *FIBOX* sind in verschiedenen Ausführungen erhältlich. Da diese Arbeit kein serienreifes Produkt zum Ziel hatte, wurden Gehäuse aus ABS-Kunststoff mit transparenten Deckeln gewählt. Die Gehäuse messen 180x130x75 mm.

Die Stecker wurden ebenfalls mit Schutzklasse IP68 gewählt. Als zusätzliches Kriterium galt der Durchmesser des Kabelmantels. Die Stecker *Lumberg 0332 05-1* verfügen über fünf Pole, die vierpolige Variante war zur Zeit nicht lieferbar.

Für den USB-Anschluss gibt es wasserdichte Ausführungen, an denen sowohl innen als auch aussen Standard-USB-Kabel angeschlossen werden können. Die *Buccaneer PCXP6043/B*-Gerätebuchse von *Bulgin* erfüllt die Anforderungen. Für diese Buchse ist auch eine wasserdichte Kappe erhältlich.

Die verschraubbare Öffnung für die SD-Karte besteht aus einer M36/M32-Gewindebuchse, in die ein M32-Schraubdeckel eingeschraubt wird.

### 7.3. Datenlogger

Die Hardware-Architektur des Datenloggers ist in Abbildung 7.5 schematisch dargestellt. Kernstück ist der Mikrokontroller *NXP LPC4088*. Der Mikrokontroller ist über eine serielle Schnittstelle mit dem CAN-Transceiver verbunden. Der Transceiver sendet Konfigurations- und Kontrolldaten und empfängt Messdaten über den CAN-Bus.

Dem Mikrokontroller verfügt über einen interner Programm- und Datenspeicher, dieser ist aber sehr beschränkt. Damit Konfigurations- und Messdaten zwischengespeichert werden können, steht dem Mikrokontroller auf dem *NXP LPC4088 QuickStartBoard* (gestrichelter Kasten) externer Speicher zur

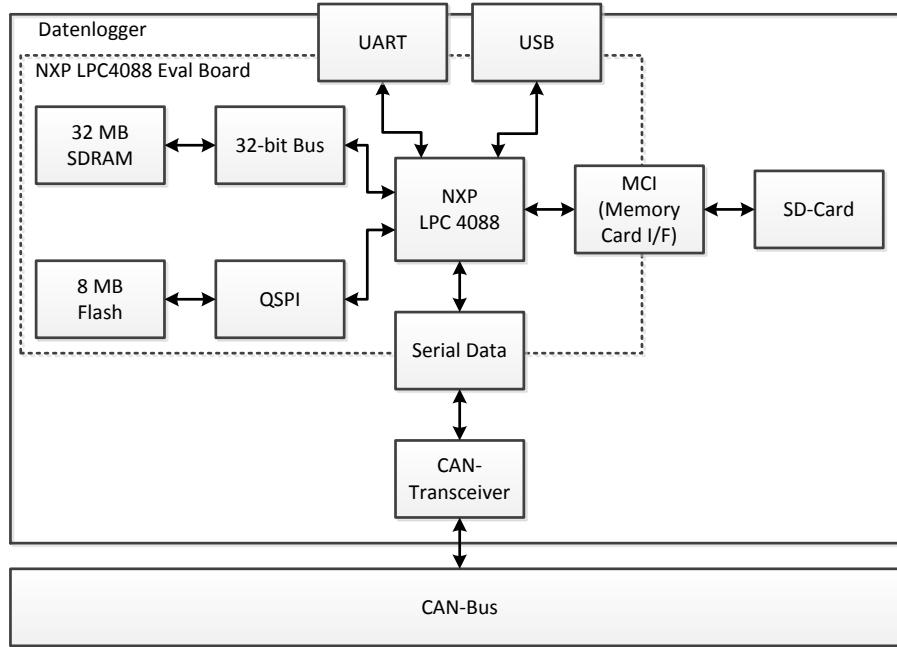


Abbildung 7.5.: Schematischer Hardware-Aufbau des Datenloggers.

Verfügung: 8 MByte Flash-Speicher und 32 MByte SDRAM. Dieser Speicher genügt, um Daten von mehreren Sensoreinheiten zwischenzuspeichern.

Über das Memory Card Interface (MCI) ist eine SD-Karte an den Mikrokontroller angebunden. Auf der SD-Karte werden sowohl die Konfigurationsdaten als auch die Messdaten gespeichert. Heute sind SD-Karten mit bis zu 256 GByte Speicherplatz erhältlich (SDXC-Karten). Zur Zeit unterstützt die verwendete Bibliothek zur Verwendung der SD-Karten nur SDHC-Karten mit bis zu 32 GByte Speicherplatz.

Für die Kommunikation mit einem Computer verfügt der Datenlogger über einen USB-Anschluss. Über einen Terminal-Emulator wie *PuTTY* kann via serielle Schnittstelle auf ein Konfigurationsmenü zugegriffen werden, um die Messanlage zu überwachen und zu steuern.

## 7.4. Sensoreinheit

Die Sensoreinheit ist ähnlich aufgebaut wie der Datenlogger. Das Schema in Abbildung 7.6 zeigt die Hardware. Wie im Datenlogger ist der *NXP LPC4088* Mikrokontroller mit zusätzlichem SDRAM und Flash-Speicher ausgerüstet. Der Anschluss an den CAN-Bus ist identisch. Die SD-Karte wird in der Sensoreinheit nicht benötigt und wurde deshalb weggelassen. Auch eine USB-Schnittstelle für die Konfiguration ist nicht nötig, da die Konfiguration über den CAN-Bus erfolgt.

Der Beschleunigungs-Sensor wird vom *QuickStart Board* mit Spannung versorgt und gibt die gemessene Beschleunigung als analoge Spannung aus. Diese Spannung wird vom A/D-Wandler des *NXP LPC4088* Mikrokontroller gemessen.

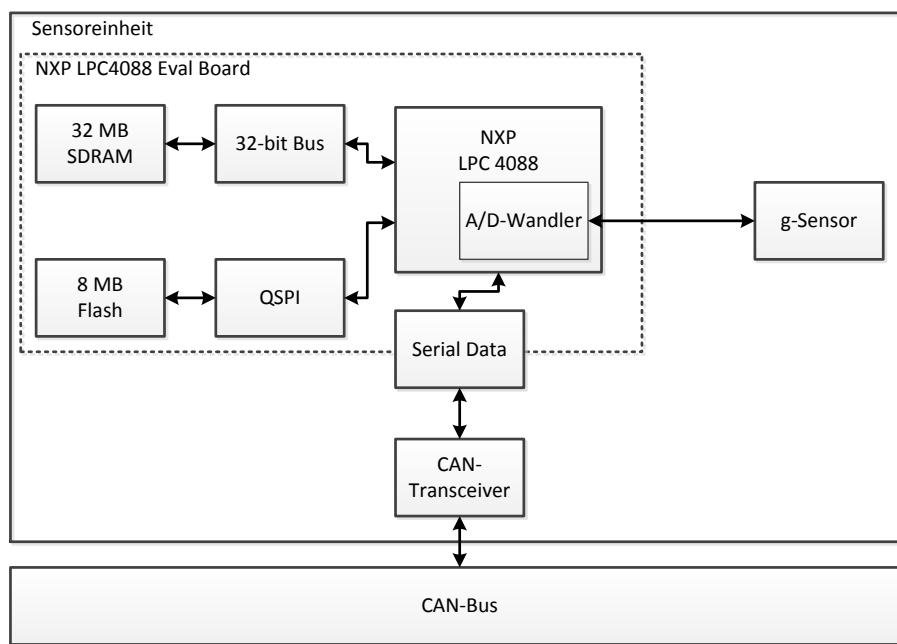


Abbildung 7.6.: Schematischer Hardware-Aufbau der Sensoreinheit.

# 8. Software

## 8.1. Software-Stack

### 8.1.1. Überblick

Die Hauptkomponenten (Abbildung 8.1) des Datenloggers sind die Datenablage und das Busprotokoll, welche jeweils in einem eigenen Threads implementiert wurden, damit sie unabhängig voneinander arbeiten können. Damit die Kommunikation zwischen den Teilnehmern reibungslos funktioniert wurde eine Busverwaltung implementiert, die mehrere Aufgaben wahrnimmt. Zum einen kümmert sie sich um die Teilnehmerverwaltung und verteilt eindeutige CAN-IDs, die aus einer Konfigurationsdatei gelesen werden. Weiter weist die Busverwaltung jeweils einem Teilnehmer ein Token zu als Erlaubnis, Daten zu schicken. Damit wird sichergestellt, dass kein Sensor verhungert, weil seine Nachrichten jeweils von anderen Sensoren überschrieben werden.

Das Busprotokoll kümmert sich um die technische Kommunikation mit den Sensoreinheiten am Bus und bietet Schnittstellen an, um Nachrichten abzuschicken oder eingetroffene Meldungen abzuholen. Trifft eine Meldung von einem Sensor ein wird ein Interrupt ausgelöst, der die Meldung ausliest und in einem Ringbuffer ablegt. Dieser Buffer wird periodisch vom Kommunikationsthread ausgelesen und die darauf enthaltenen Meldungen in ein Format gebracht, das für die Anwendung lesbar ist. Im Anschluss wird diese Meldung mittels einer Queue und unter Verwendung eines Memory-Pools dem Datenablage-Thread zur Verfügung gestellt. Dieser holt sich die vorbereitete Meldung und, sofern es sich dabei um Messdaten handelt, schreibt diese in die Datendatei der sendenden Sensoreinheit. Nach der Verarbeitung der Message gibt der Datenablage-Thread den Platz im Memory-Pool wieder frei. Bei ausgehenden Nachrichten wird gleich verfahren, auch hier schreibt der Absender die Meldung in eine Queue, die dann vom Busprotokoll-Thread abgearbeitet wird. Beim Datenlogger wurde die Queue für die eingehenden Meldungen gross genug gewählt, um ca. 800 Meldungen zwischen zu speichern, falls viele Ereignisse erfasst und übermittelt wurden. Dem gegenüber wurde die Queue für die ausgehenden Meldungen klein gehalten, da der Datenlogger nur Steuerkommandos oder Konfigurationen an die Sensoren übermittelt, was nicht so häufig vorkommt, bzw. schnell erledigt ist.

Die Software der Sensoreinheit (Abbildung 8.2) ist ähnlich der Datenlogger-Software strukturiert, auch in der Sensoreinheit operieren zwei Threads (Abbildung 8.3). Zum einen ist hier auch wieder der Busprotokoll-Thread anzutreffen, der wie beim Datenlogger die ganze Kommunikation kapselt. Im zweiten Thread läuft auf der Sensoreinheit die Ereigniserkennung, welche die Daten des angeschlossenen Beschleunigungssensors ausliest und die Ergebnisse basierend auf dem festgelegten Detail-Level aufbereitet. Ist ein Ereignis abgeschlossen (d.h. der Ausschlag des Beschleunigungssensors liegt unter einem festgelegten Schwellwert) wird dieses in die Queue zum Busprotokoll geschrieben. Hat der Sensor das Token bereits erhalten, wird er die Meldung unverzüglich übermitteln, andernfalls wird er die Message in der Queue belassen, bis ihm das Token zugewiesen wird. Damit die Queue nicht so schnell überläuft, wurde sie entsprechend gross gewählt und dient somit als eine Art Buffer. Die Queue für eingehende Meldungen bietet dafür nur wenigen Meldungen Platz, da die Sensoren einerseits nur Meldungen erhalten, die entweder an sie selbst oder als Broadcast adressiert wurden (andere Meldungen werden vom Acceptance-Filter ausgeschlossen) und andererseits nur Steuerkommandos vom Datenlogger zu erwarten sind.

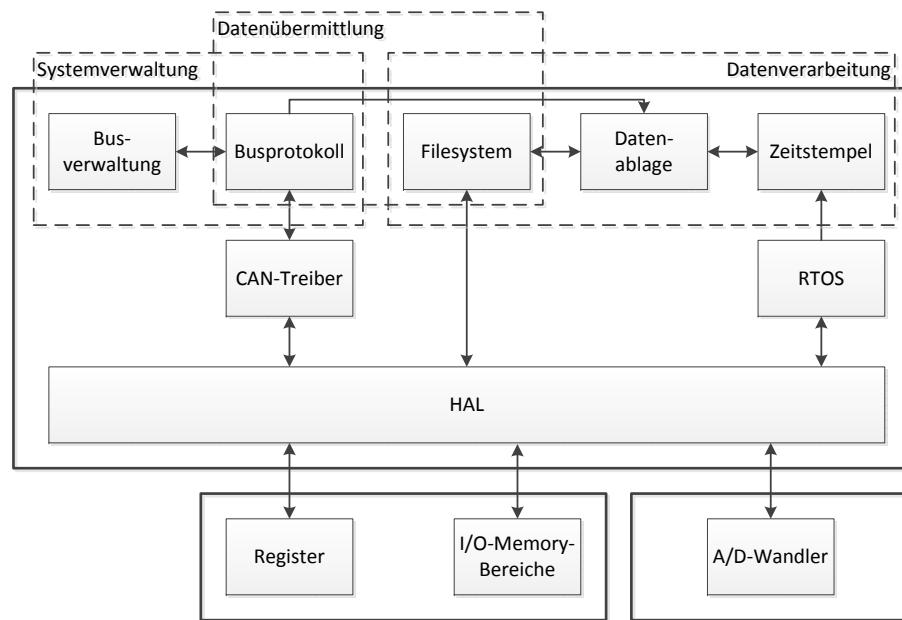


Abbildung 8.1.: Softwarestack des Datenloggers.

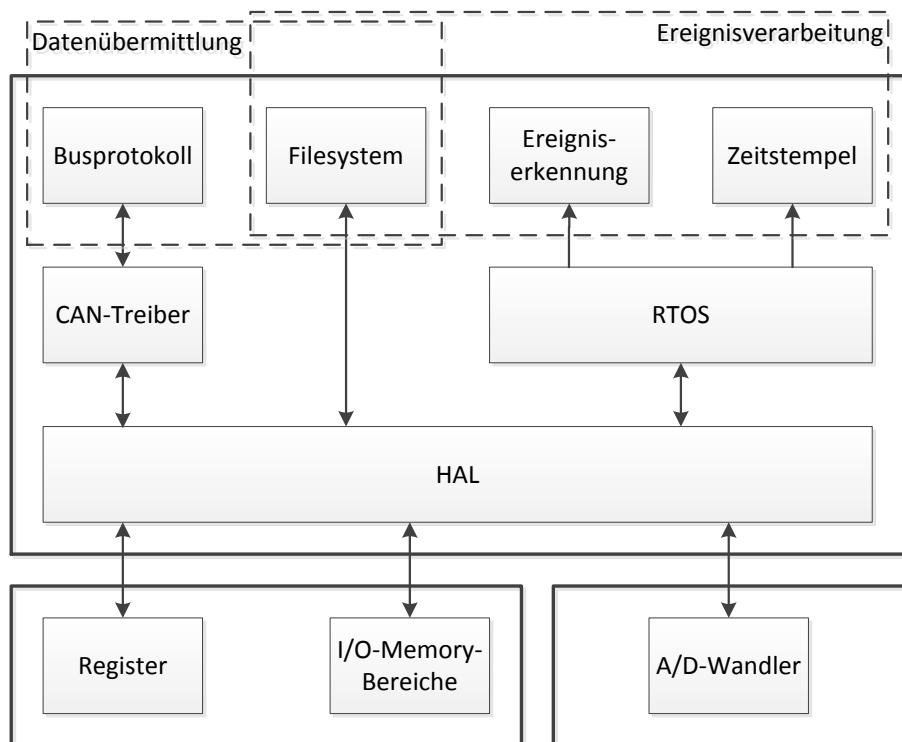


Abbildung 8.2.: Softwarestack der Sensoreinheit.

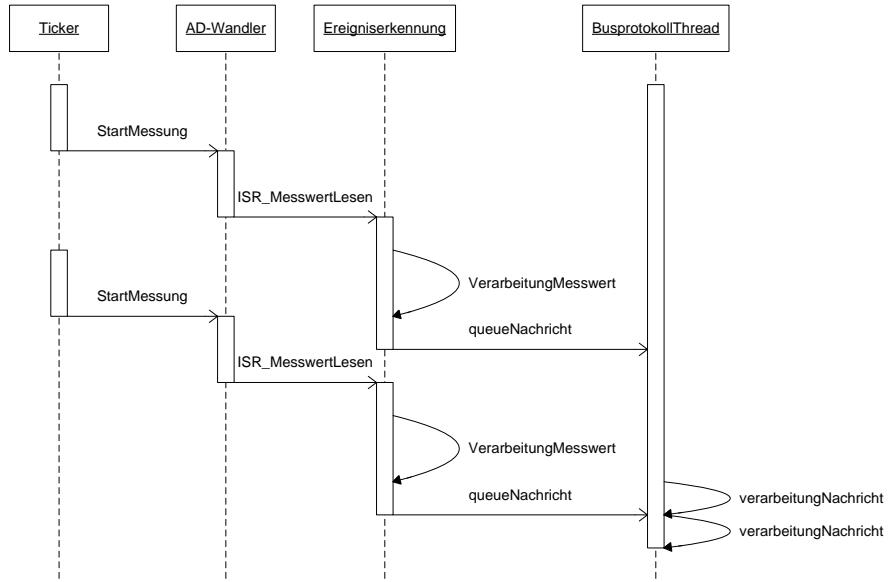


Abbildung 8.3.: Kommunikationsdiagramm der Threads in der Sensoreinheit.

### 8.1.2. Messdatenerfassung

Der NXP LPC4088 Mikroprozessor verfügt über einen 12-bit A/D-Wandler, der über einen Multiplexer auf acht Pins messen kann. Auf dem verwendeten Quickstart-Board stehen sechs Pins für A/D-Wandlung zur Verfügung. Für die geplante Anwendung reicht ein A/D-Pin, da der Beschleunigungs-Sensor die Beschleunigung nur auf einer Achse misst. Der A/D-Wandler des NXP LPC4088 wird mit einer Abtastrate von 10 kHz betrieben. Falls höhere Abtastraten nötig sind, kann der A/D-Wandler theoretisch mit bis zu 400 kHz betrieben werden.

**Abtastrate** Um die Abtastrate genau einzuhalten wird ein Ticker (Timer) des *NXP LPC4088* Mikrocontrollers eingesetzt, der die Messung des A/D-Wandlers anstösst (Listing 8.1). Das Intervall des Tickers kann dynamisch konfiguriert werden. Diese Methode ist genauer als das Heruntertakten des A/D-Wandlers, was nur in relativ groben Stufen erfolgen kann. Der Ticker löst einen Interrupt aus, der den A/D-Wandler anstösst. Die A/D-Wandlung nimmt immer genau 31 Taktzyklen in Anspruch. Nach abgeschlossener Messung schreibt der A/D-Wandler den Messwert in ein eigenes Register und setzt einen Interrupt Request (IRQ). Der IRQ signalisiert dem Mikrocontroller, dass ein Messwert zur Abholung bereit liegt. Damit der Messwert so rasch wie möglich, auf jeden Fall aber vor Ablauf der nächsten Messperiode, für die weitere Verarbeitung abgeholt werden kann, hat der IRQ die höchste Priorität. Der Interrupt des A/D-Wandlers unterbricht also jeden laufenden Prozess. Das Zusammenspiel von Ticker, A/D-Wandler, Ereigniserkennung und Busprotokoll ist in einem Kommunikationsdiagramm in Abbildung 8.3 dargestellt.

```

1 // ISR
2 void start_ADC_Conversion(){
3     // start conversion when the ticker fires
4     LPC_ADC->CR |=(1 << 24);
5 }
6
7 // ...
8 // register Interrupt Handler and attach ticker
9 int register_ADC_interrupt(analogin_s *obj, PinName pin, uint32_t
    ADC_IRQHandler, uint32_t interval){
10    // ...
11    ticker.attach_us(&start_ADC_Conversion, interval);
12    // ...

```

13

}

Listing 8.1: Timer mit Aufruf der A/D-Wandler-Funktion (ADC\_4088.cpp)

**ISR** Der Mikrokontroller ruft die Interrupt Service Routine (ISR) (Listing 8.2) auf, um den IRQ abzuhandeln. Die ISR holt den Messwert aus dem Register und speichert ihn zusammen mit dem Timestamp in einer FIFO-Queue ab. Die ISR erhöht den Timestamp um eins für den nächsten Messwert und gibt dann die Kontrolle an den unterbrochenen Prozess zurück.

```

1 void isr_nextMeasurement(){
2     // read ADC measurement from Register, automatically resets IRQ
3     value = LPC_ADC->GDR;
4     value = (value >> 4) & 0xFFFF;
5     timestamp++;
6
7     enqueue_impact_input(timestamp, value);
8 }
```

Listing 8.2: ISR zur Abhandlung des ADC-Interrupt Requests (impact\_event.cpp)

Sobald das Ereignis abgeschlossen ist (siehe 8.1.3), wird es mittels einer Queue asynchron dem Kommunikationsthread zur Verfügung gestellt, der die Meldungen dem Logger weiterleitet, sofern der Bus frei ist und der aktuelle Sensor den Token zur Datenübermittlung erhalten hat.

**Timestamp** Der Timestamp muss zwingend in der ISR (Listing 8.2) erhöht werden, damit er immer sofort nach der Erfassung eines neuen Messwerts aktualisiert wird. Würde der Timestamp mittels einem zweiten Timer und einer zweiten ISR erhöht, bestünde die Gefahr einer ungeregelten Reihenfolge der Abarbeitung der IRQs. Mal würde der Timestamp vor dem Kopieren der Messdaten erhöht, mal erst danach.

**Datenauswertung** Die Datenauswertung wird in regelmässigen Abständen aufgerufen und arbeitet die FIFO-Queue mit den neuen Messwerten ab, um Ereignisse zu erkennen. Der Aufruf der Ereigniserkennung ist nicht so stark an einen genauen Takt gebunden wie die A/D-Wandlung, da die FIFO-Queue genügend gross ist, um mehrere hundert Messwerte zwischenspeichern. Ein Thread ruft die Ereigniserkennung auf und wartet nach Beendigung der Subroutine 1 ms.

### 8.1.3. Ereigniserkennung

#### Hilbert-Transformation

Von der WSL wurde die Ereigniserkennung bisher mittels Hilbert-Transformation gelöst. Die Hilbert-Transformation liefert die umhüllende Kurve des gemessenen Signals. Überschreitet die Umhüllende den Threshold, markiert dies den Start eines neuen Ereignisses. Fällt die Umhüllende unter den Threshold, ist das Ereignis beendet. Abbildung 8.4 zeigt ein Beispiel von Messdaten aus einer bestehenden Anlage. Die Hüllkurve wurde von Hand eingezeichnet, um das Verfahren zu demonstrieren. Die Ereignisse werden als 'packet' bezeichnet, die Peaks als 'Impulses'.

Die Berechnung der Hilbert-Transformation erfordert einigen Aufwand. Durch diskrete Fourier-Transformation (DFT) wird das Spektrum des Signals berechnet. Negative Frequenzanteile werden auf null gesetzt und das resultierende Spektrum mittels inverse diskrete Fourier-Transformation (IDFT) wieder in ein Signal umgerechnet (vgl. [15]). Das resultierende Signal umhüllt das Eingangssignal.

Für die DFT und die IDFT ist der Rechenaufwand je  $N \cdot \log_2(N)$ . Je mehr Datenpunkte in einem Schritt verrechnet werden (Blockgrösse), desto höher ist der Aufwand, aber desto genauer ist das Resultat. Mit einer Blockgrösse von 128 Messwerten benötigt die DFT und die IDFT je 896 komplexe

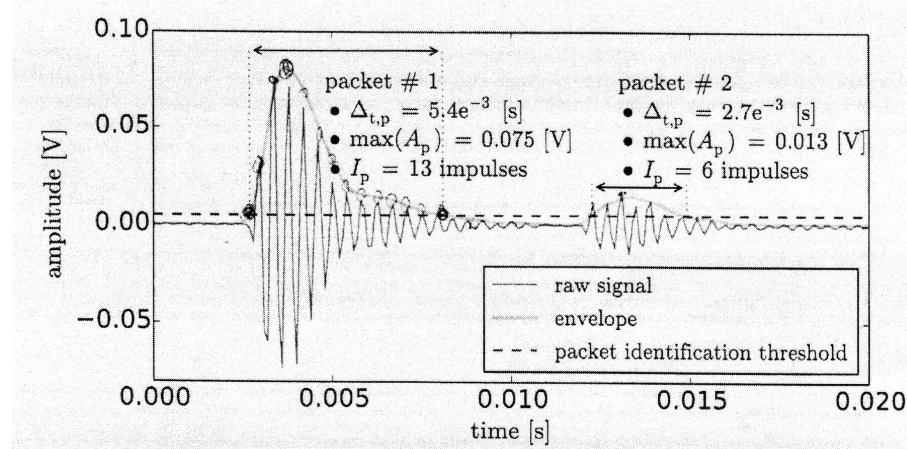


Abbildung 8.4.: Beispiel von Messdaten mit einer Hüllkurve (envelope).

Multiplikationen und Additionen (vgl. [16, Kap. 3, S. 48]). Pro Messwert sind das 7 komplexe Multiplikationen und Additionen. Dank der DSP-Fähigkeiten des gewählten Cortex™-M4 Prozessors liegt die zu erwartende Prozessorauslastung für die Hilbert-Transformation bei einer Abtastrate von 10 kHz bei wenigen Prozent.

### Hilbert-Transformation als FIR-Filter

Die Hilbert-Transformation kann mittels eines FIR-Filters angenähert werden. Ein Allpass mit gerader Filter-Ordnung und geeignet gewählten Koeffizienten (Abbildung 8.5) liefert eine gute Näherung. Je nach gewählter Ordnung des Filters ist der Rechenaufwand aber in ähnlicher Größenordnung wie mit DFT und IDFT.

### Zustandsmaschine

Um den Rechenaufwand der Hilbert-Transformation zu umgehen, lösen wir die Ereigniserkennung mittels einer Finite State Machine (FSM). Das Zustandsdiagramm der Finite State Machine in Abbildung 8.6 zeigt alle möglichen Zustände der FSM sowie welche Ereignisse einen Übergang in einen anderen Zustand auslösen.

**Konfiguration der Zustandsmaschine** Über Parameter wird definiert, welche Signalform als Ereignis erkannt werden soll. In Abbildung 8.7 sind die Parameter dargestellt.

**Nullpegel** Der Nullpegel kann angepasst werden, um die Erdanziehung, die als Beschleunigung auf den Sensor wirkt, zu kompensieren. Je nachdem wie der Sensor orientiert ist, ist die Erdanziehungskraft nicht parallel zur Mess-Achse des Sensors. Die vom Sensor gemessene Erdanziehungskraft in Richtung der Mess-Achse ist damit nicht immer gleich gross. Deshalb muss der Nullpegel angepasst werden können. Die mittlere gestrichelte schwarze Linie in Abbildung 8.7 stellt den Nullpegel dar.

**Threshold** Der Threshold definiert, ab welcher Abweichung des Signalpegels vom Nullpegel die FSM einen Messwert als 'hoch' betrachten soll. Zu beachten ist, dass der Threshold auf beide Seiten des Nullpegels gilt. Da der Sensor sowohl Beschleunigungen nach oben wie auch nach unten erfährt, unterscheidet die FSM dies mit den Ereignissen 'hoch\_positiv' resp. 'hoch\_negativ'. Signalpegel, die den Threshold nicht überschreiten, werden als 'niedrig' eingestuft. In Abbildung 8.7 ist das erste Erreichen des (negativen) Thresholds mit einer vertikalen roten Linie markiert.

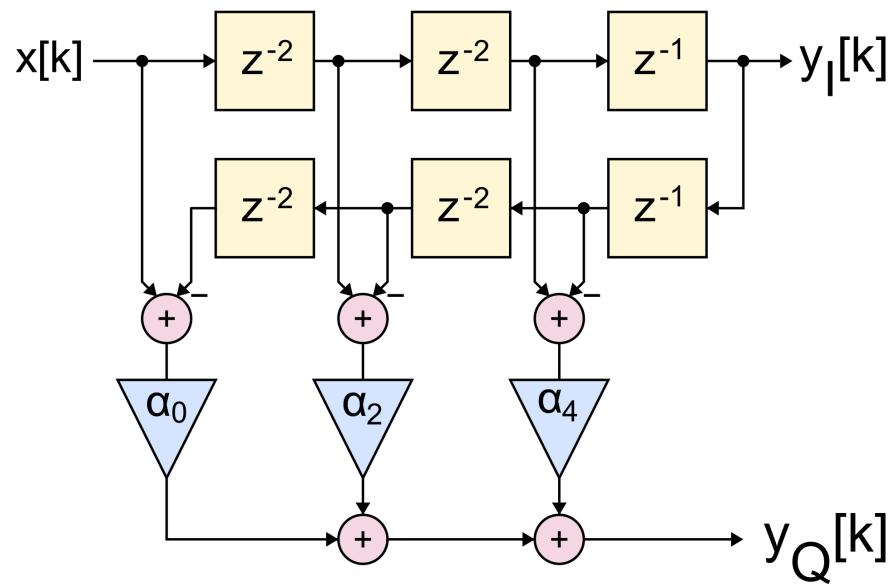


Abbildung 8.5.: Hilbert-Transformation als FIR-Filter [17].

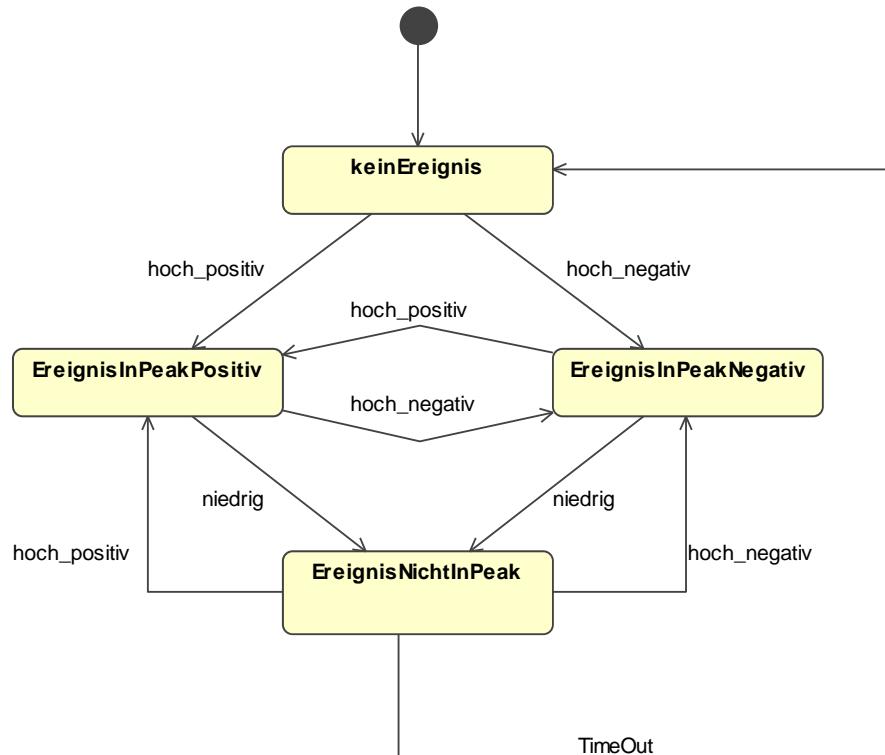


Abbildung 8.6.: Zustandsmaschine der Ereigniserkennung.

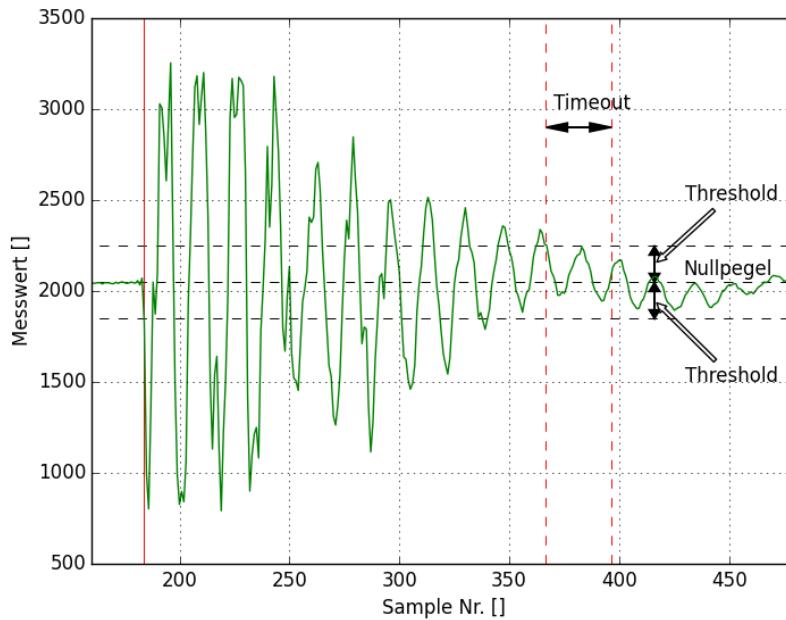


Abbildung 8.7.: Parameter der Ereigniserkennung.

**Timeout** Da ein Ereignis aus mehr als einem Peak besteht, muss eine Dauer (Timeout) definiert werden können, während der die FSM auf den Beginn eines neuen Peaks wartet. Tritt während des Timeouts kein neuer Peak auf, gilt das Ereignis als beendet. Die gestrichelten roten Linien in Abbildung 8.7 zeigen den Timeout.

### Ablauf der Zustandsmaschine

Im Folgenden wird der Ablauf in der Zustandsmaschine genauer erklärt. Im Zustandsdiagramm in Abbildung 8.6 sind die Namen der Zustände und Ereignisse ersichtlich. Der Übersichtlichkeit halber wurde auf die Auflistung der Aktionen im Diagramm verzichtet.

Die FSM wird im Zustand 'keinEreignis' initialisiert. Tritt ein Messwert auf, der als 'hoch\_positiv' klassiert wird, wechselt die FSM in den Zustand 'EreignisInPeakPositiv'. In diesem Zustand verbleibt die FSM, bis ein anders klassifizierter Messwert eintrifft.

Ein Messwert 'niedrig', also unterhalb des Thresholds, führt zu einem Übergang in den Zustand 'Ereignis\_NichtInPeak'. Dieser Übergang startet einen Timer, der während der im Parameter 'Timeout' definierten Anzahl Messwerte läuft. Falls die FSM bis zum Ablauf des Timers keinen Messwert 'hoch\_positiv' oder 'hoch\_negativ' erhält, wechselt sie wieder in den Zustand 'keinEreignis' und übergibt die Ereignisdaten dem Prozess, der für die Übertragung zum Datenlogger zuständig ist.

Der Timer läuft nicht in Echtzeit, sondern zählt die Anzahl Messwerte seit seinem Start, da die Verarbeitung asynchron zur Erfassung der Messwerte läuft. Das bedeutet, dass die Verarbeitung problemlos während mehreren Messwerten stillstehen kann, ohne dass Messwerte verloren gehen. Dies ist möglich, da die Messwerte in eine Warteschlange (FIFO-Queue) geschrieben werden, von wo sie von der FSM abgeholt werden. So lange die FIFO-Queue nicht überfüllt wird, gehen keine Messwerte verloren. Die Verarbeitung der Messwerte in der FSM erfolgt im *NXP LPC4088* Mikrokontroller schnell genug, um theoretisch mit einer Abtastrate bis 200 kHz messen zu können.

Falls die FIFO-Queue doch einmal überlaufen sollte, wird eine Nachricht an den Datenlogger übertragen, die eine entsprechende Meldung in die Datendatei der Sensoreinheit einträgt. Dies bedeutet, dass zu diesem Zeitpunkt einige Messwerte verloren gegangen sind. Da der Timestamp in der Sensoreinheit

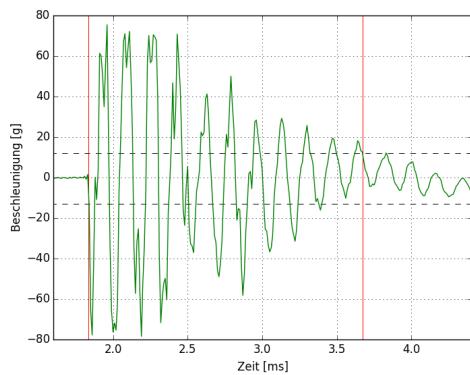


Abbildung 8.8.: Detail-Level 'raw'.

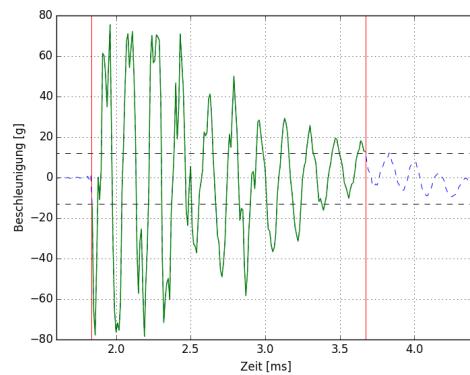


Abbildung 8.9.: Detail-Level 'detailed'.

trotzdem mit jedem Sample erhöht wird, stimmen die Zeitangaben in der Datendatei weiterhin. Es besteht lediglich eine Lücke im Datensatz.

zammenhänge  
ndlung und Er-  
ennung und  
gung beschrei-  
u ein Kommu-  
diagramm, wo  
o, wo asynchron.

#### 8.1.4. Detail-Level

Je nach Einsatz der Messstation variiert die benötigte unbeaufsichtigte Messdauer von einigen Tagen bis zu mehreren Monaten. Die zu speichernde Datenmenge muss an diese Dauer angepasst werden können. Für einen langen Einsatz werden von Vorteil weniger detaillierte Messdaten abgespeichert, während für einen eher kurzen Einsatz unter Umständen sogar unkomprimierte Daten abgespeichert werden können.

Für die Wahl einer geeigneten Datenrate stehen vier verschiedene Modi, sog. Detail-Level zur Verfügung.

Im 'raw'-Modus werden Rohdaten gespeichert (Abbildung 8.8). Dieser Modus soll hauptsächlich der Gewinnung von Kalibrierungsdaten dienen, da hier sehr viele Daten anfallen. Es wird empfohlen, nur wenige Sensoren in diesen Modus zu versetzen.

Im 'detailed'-Modus werden von jedem Ereignis alle Datenpunkte gespeichert (Abbildung 8.9). Die uninteressanten Datenpunkte zwischen den Ereignissen werden nicht übertragen und aufgezeichnet. Dieser Modus ist sowohl für kurze als auch für längere Messperioden interessant, da er nur interessante Daten aufzeichnet. Die Menge anfallender Daten hängt direkt von der Häufigkeit der Ereignisse ab.

Der 'peaks only'-Modus zeichnet die Dauer des Ereignisses sowie Zeitpunkt und Intensität aller Peaks eines Ereignisses auf. Damit ist ein grosser Teil der Information des 'detailed'-Modus vorhanden, aber in geringerer zeitlicher Auflösung.

Der 'sparse'-Modus liefert lediglich die Dauer des Ereignisses, die Anzahl Peaks und den maximalen Ausschlag. Dies ist eine minimale Informationsmenge, die dennoch eine Aussage über das Geschiebckorn zulässt.

Eine ausführliche Beschreibung der Detail-Level ist in der Bedienungsanleitung ab Abschnitt 11.6.1, Seite 63 zu finden.

#### 8.1.5. Timestamp

Der Timestamp ist ein Zähler, der in jeder Sensoreinheit die gemessenen Samples zählt. Die übertragenen Ereignisse werden mit dem Timestamp versehen. Der Datenlogger kann anhand des Timestamps, der

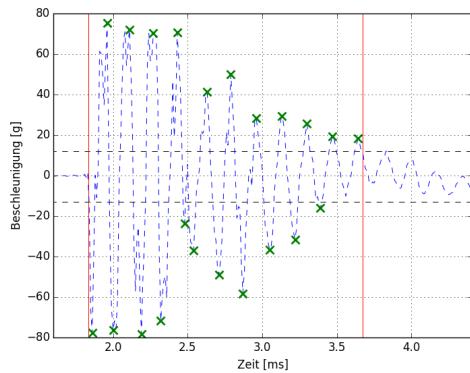


Abbildung 8.10.: Detail-Level 'peaks only'.

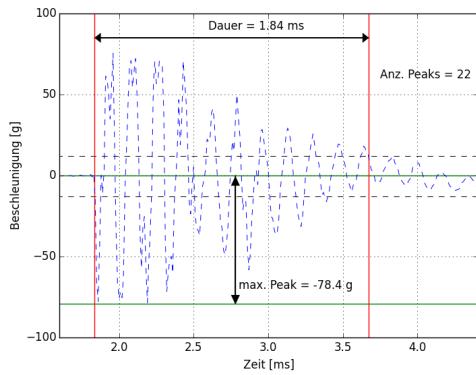


Abbildung 8.11.: Detail-Level 'sparse'.

Abtastrate der betreffenden Sensoreinheit und des Zeitpunkts des letzten Zurücksetzens der Timestamps den Zeitpunkt der Aufnahme des Ereignisses berechnen (Gleichung 8.1).

$$t_{Ereignis} = t_{reset} + (timestamp/f_s) \quad (8.1)$$

## 8.2. Busverwaltung

### 8.2.1. Verwaltung des Bussystems

Da der CAN-Bus für dieses Projekt in einem Internet Protocol (IP) ähnlichen Modus laufen soll, muss der Bus von einer zentralen Stelle verwaltet werden. Ansonsten wäre es schwierig, die einzelnen Teilnehmer am Bus eindeutig zu identifizieren oder zu verhindern, dass ein Sensor auf seinen Daten sitzenbleibt. Abbildungen 8.12 und 8.13 zeigen die Sequenzen der Verwaltung des Bussystems. Um das zu garantieren, wird der Datenlogger als Busmaster eingesetzt, der den Sensoren aufgrund einer Konfigurationsdatei und der Mikrocontroller-Seriennummer eine eindeutige CAN-ID zuweist. Hat der Datenlogger die Seriennummer aller Sensoreinheiten erhalten und diese registriert oder ist vorher ein Timeout aufgetreten, werden die CAN-Identifier verschickt. Als nächstes übermittelt der Datenlogger allen Sensoreinheiten ihre spezifischen Einstellungen (sofern in der Konfigurationsdatei vorhanden) oder die Standardwerte. Die Sensoreinheit erhalten daraufhin den Timesync-Broadcast des Datenlogger, der eine Zurücksetzung des Timestamps in den Sensoreinheiten veranlasst. Sobald alle Einstellungen gesetzt wurden, aktiviert der Logger standardmäßig den Aufnahmemodus der Sensoren und schickt dem ersten registrierten Sensor den Token zu. Dieser Token enthält auch die maximale Anzahl der Meldungen, die der Sensor dem Logger schicken darf. Der Sensor bestätigt den Token und sendet dem Logger die Anzahl der Meldungen, die er übermitteln wird. So weiß der Logger jederzeit, wieviele Meldungen des Sensors noch ausstehend sind und kann genug Speicher reservieren.

Der Start-Ablauf einer Sensoreinheit (Abbildung 8.13) ist das Gegenstück zu dem des Datenloggers. Zu Beginn erwartet die Sensoreinheit die Anforderung, ihre Seriennummer bekanntzugeben. Da die Sensoreinheiten keinen persistenten Speicher aufweisen, müssen die Einstellungen jeweils vom Datenlogger übermittelt werden. Erhält die Sensoreinheit diese nicht, läuft sie mit den Standardwerten weiter. Beim Eintreffen des Timesync-Broadcasts setzt die Sensoreinheit den Timestamp auf 0 und erwartet den Befehl zum Start der Aufnahme. Sobald dieser eingetroffen ist, wird die Ereigniserkennung gestartet und die Sensoreinheit ist betriebsbereit.

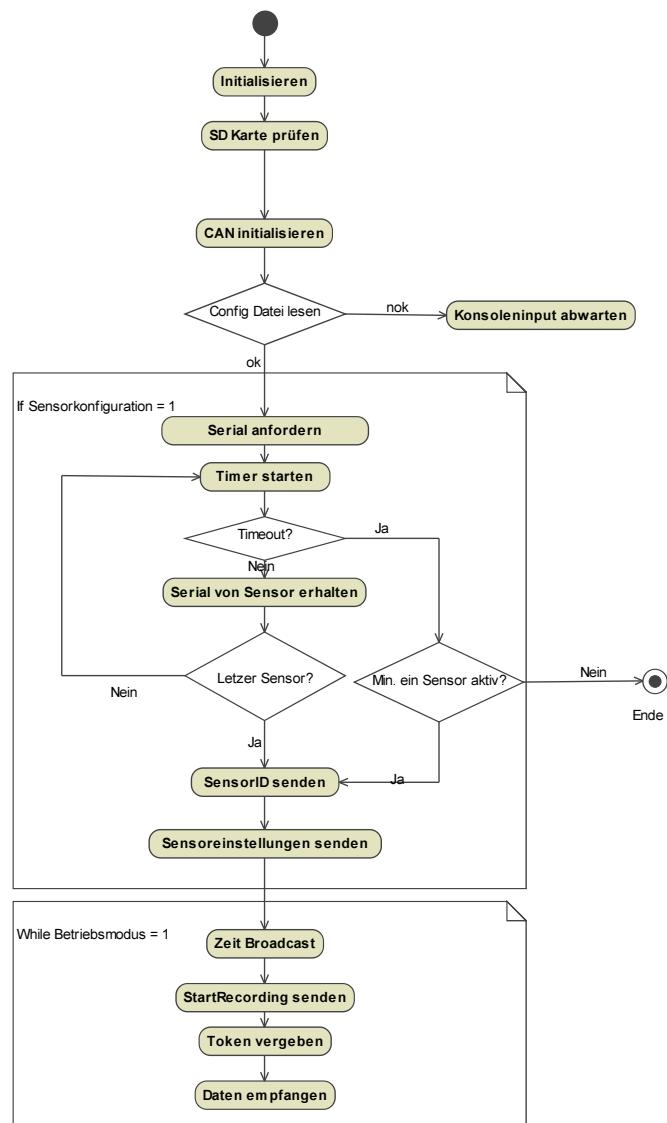


Abbildung 8.12.: Sequenzdiagramm des Startupvorgangs der Messstation.

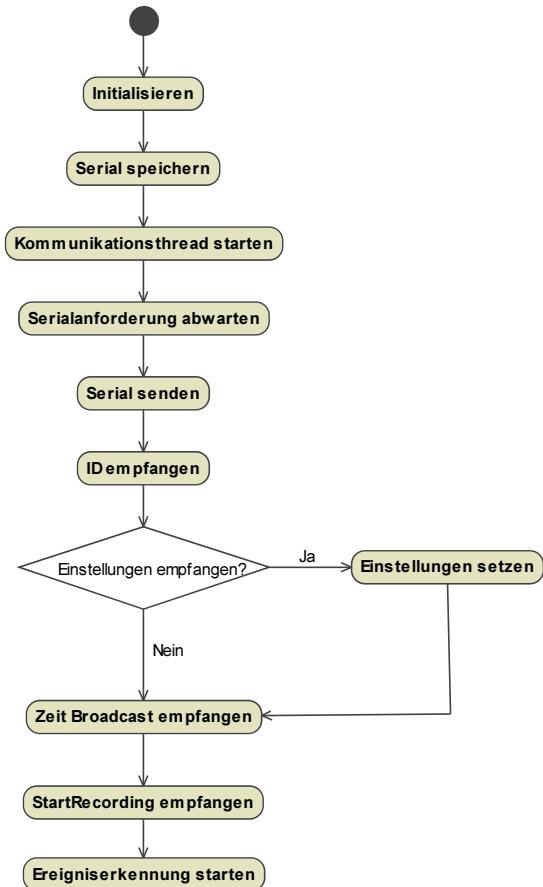


Abbildung 8.13.: Sequenzdiagramm des Startupvorgangs der Messstation.

### 8.2.2. Busprotokoll

Ursprünglich wurde das Controller Area Network (CAN) von Bosch entwickelt, um die Steuergeräte in Automobilen zu verbinden [10]. In einem solchen Umfeld ist es wichtig, dass die Kommunikation fehlerfrei und zeitnah erfolgt, da fehlerhafte oder verpasste Meldungen verheerende Folgen nach sich ziehen können (z.B. verpasstes Auslösen des ABS oder der Airbags). Um diese Anforderungen zu erfüllen, wurden einige Vorkehrungen getroffen. Zum einen werden die Daten auf dem CAN-Bus als differenzielles Signal übertragen. Äussere Einflüsse wirken auf beide Leitungen etwa gleich stark und verändern die Differenz deshalb praktisch nicht (siehe auch Abbildung 7.4, Seite 28).

Ein weiteres Problem, dass bei einem Bussystem auftreten kann, sind Daten-Kollisionen zwischen den einzelnen Teilnehmern. Um diese zu verhindern, wird beim CAN das 'Carrier Sense Multiple Access/Collision Resolution'-Verfahren angewandt, bei dem die Teilnehmer zum Sendezeitpunkt feststellen, dass ihre Nachricht mit der eines anderen Teilnehmers kollidiert und dann den Konflikt lösen. Derjenige Sender mit der niedrigeren CAN-ID hat Priorität vor dem anderen Sender, er darf seine Nachricht zu Ende senden [10]. Abbildung 8.15 zeigt den Aufbau eines CAN-Frames im Detail.

Um eine Kollision zu erkennen, prüft jeder Teilnehmer simultan zum Senden, was für ein Signal gerade auf dem Bus anliegt. Stimmt das Bit auf dem Bus mit dem gerade übermittelten überein, ist keine Kollision aufgetreten oder es wurde von allen Teilnehmern gerade ein dominantes Bit (0) übermittelt. Sendet der Teilnehmer im Arbitration-Feld (das aus der Message-ID und einem Control-Bit besteht) ein rezessives Bit und liegt auf dem Bus ein dominantes Bit an, stellt der Teilnehmer seine Übermittlung ein und schickt die Meldung später selbstständig erneut [10]. Abbildung 8.14 zeigt den Ablauf einer Kollision und ihrer Auflösung mit drei Sendern, die gleichzeitig eine Übertragung starten.

#### Implementation

In einem CAN-Bus wird normalerweise kein Bus-Master verwendet, da die Nachrichten nach Verwendungszweck mit IDs versehen sind und so jeder Teilnehmer herausfinden kann, welche Meldungen für ihn relevant sind (zum Beispiel interessiert sich der Auslöser des Airbags nicht für die Abgaswerte der Lambdasonde). Für dieses Bussystem ist aber neben der Art der Meldung auch noch der Absender und der Empfänger wichtig, um einerseits die direkte Kommunikation zwischen dem Datenlogger und einem Sensor und andererseits die Steuerung des Busses per Broadcast-Meldungen zu ermöglichen.

Da für den Datenlogger wichtig ist, wer welche Daten geschickt hat und die einzelnen Sensoren auch unabhängig voneinander konfiguriert werden müssen, muss jeder Sensor eine eindeutige ID erhalten. Zusätzlich soll unterschieden werden, welcher Art die Nachrichten sind und welche Nachrichten Vorrang haben, falls bei der Übertragung eine Kollision auftritt. Wenn die vorhandenen 11 Bit der Message-ID für den Message-Typ, die Sender- und die Receiveradresse verwendet würden, könnten die Adressen etwa vier Bit ausmachen, es wären also 16 Teilnehmer auf dem Bus erlaubt. Um diese Beschränkung zu umgehen, können Meldungen im „extended format“ geschickt werden. In diesem Format werden Identifier mit einer Länge von 29 Bit zugelassen, welche ausreichen, um den Bus für insgesamt 255 Teilnehmer mit 32 Nachrichtentypen und einem 8 Bit Paketzähler zu betreiben. Abbildung 8.16 zeigt einige Beispiele der Header von CAN-Frames. Das Busprotokoll mit dem Aufbau aller Nachrichtentypen befindet sich als Excel-Datei auf der beiliegenden CD.

Um sicherzustellen, dass der Busmaster (in diesem Fall der Datenlogger) immer Priorität vor den anderen Teilnehmern hat, wurde das erste Bit der Message-ID als Prioritätsbit verwendet und für alle Nachrichten vom Datenlogger auf 0 gesetzt. In der Konfiguration dieses Busses steht eine 0 für ein dominantes Bit, d.h. eine gleichzeitig gesendete 1 würde überschrieben und der andere Teilnehmer erkennt eine Kollision, woraufhin er die Übertragung abbricht und später erneut versucht. Die folgenden vier Bits dienen als Message-Typ (nicht zu verwechseln mit dem CAN-Message Typ, der nicht in der Message-ID vorkommt), welcher an erster Stelle ebenfalls das Prioritätsbit aufweist. Sollten mehr als die bereits verwendeten Message-Typen notwendig werden, kann der volle Bereich ausgeschöpft werden. Die nächsten acht Bits enthalten die Empfängeradresse, darauf folgen die Senderadresse und die eigentliche Message-ID (der Paketzähler). Für eine Broadcast-Message wird die Zieladresse auf 0xFF gesetzt,

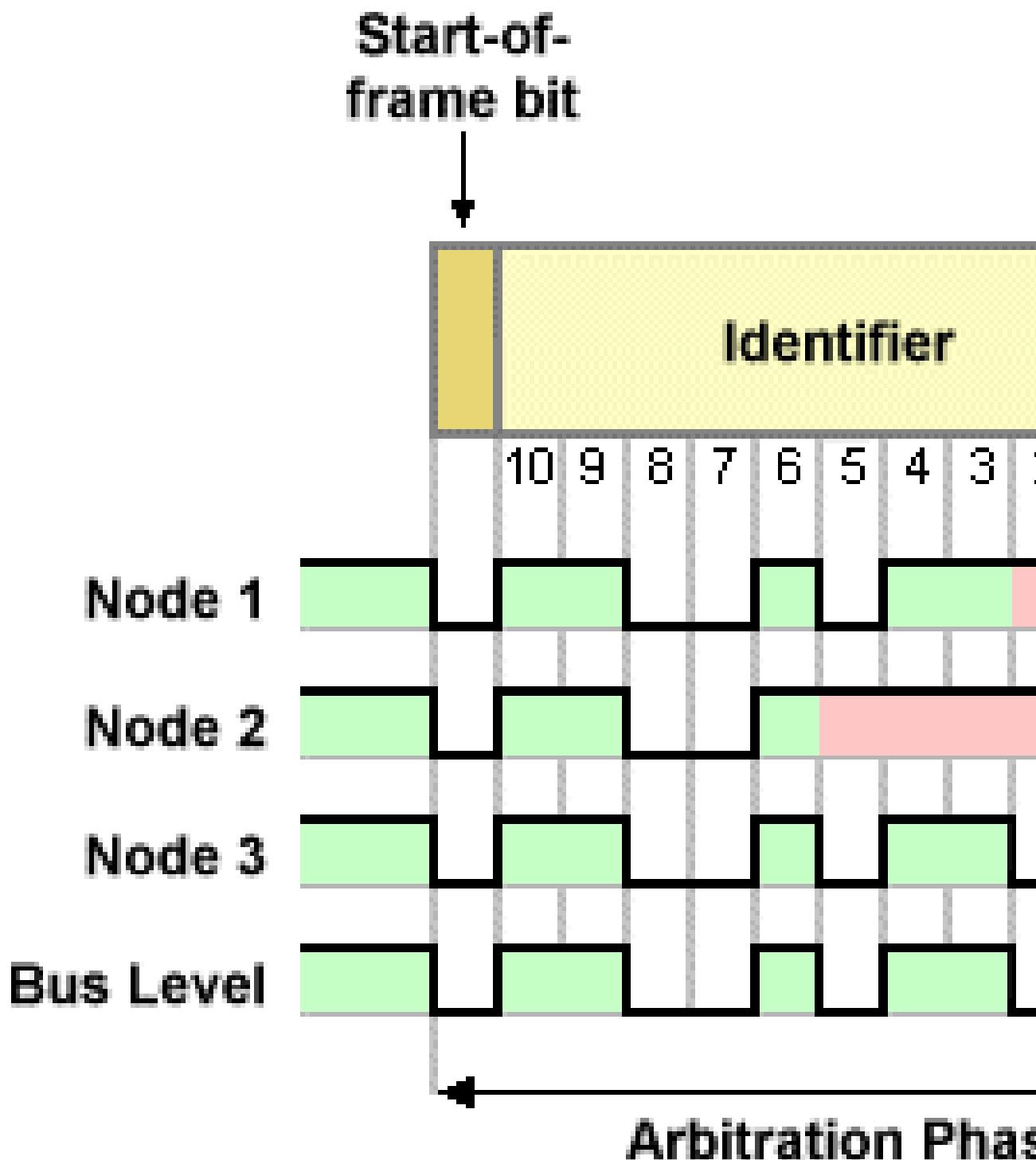


Abbildung 8.14.: Auflösung einer Kollision bei CAN-Bus.

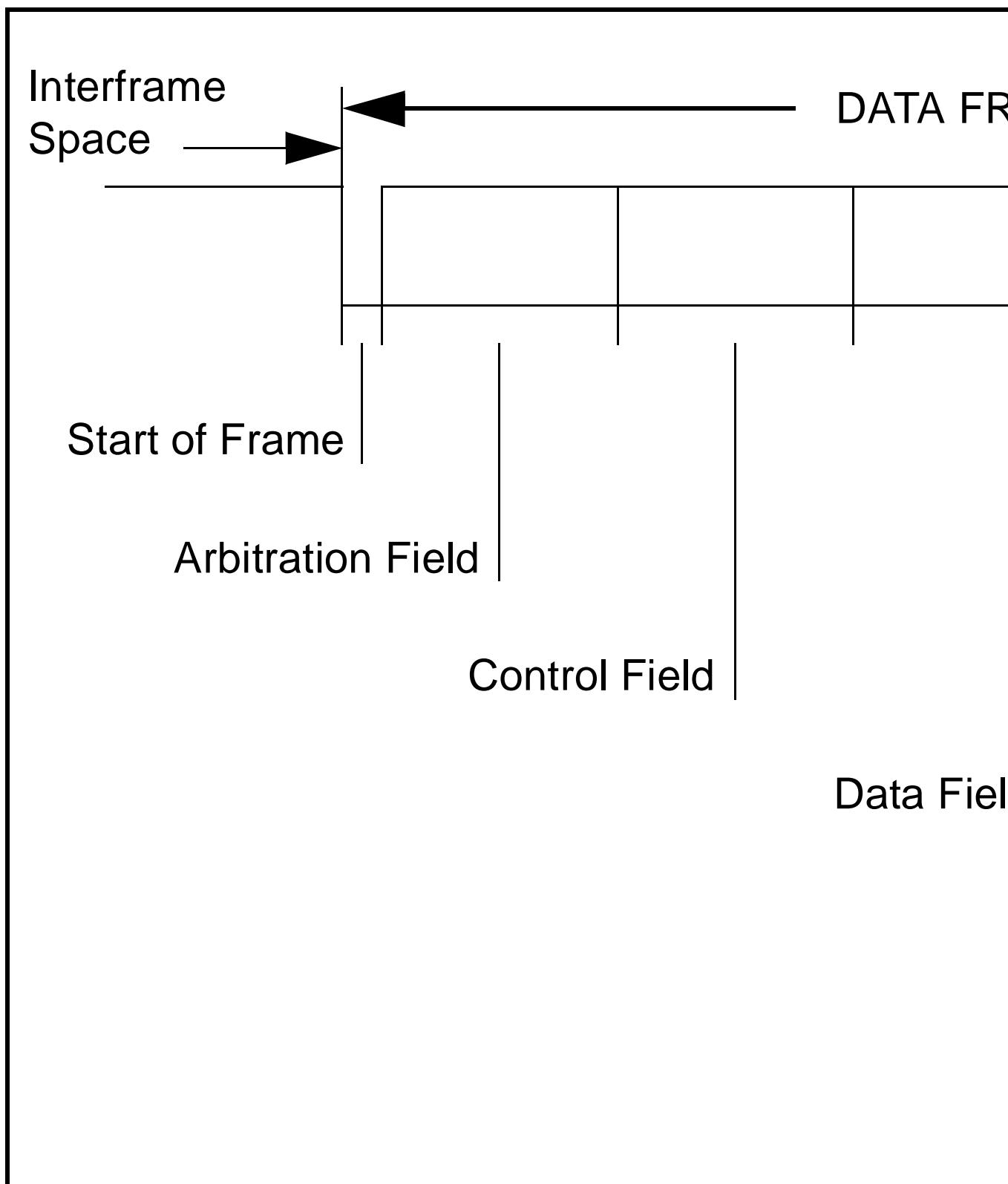


Abbildung 8.15.: Aufbau eines CAN-Frames [? ].

# CAN-Messages

Sender	Empfänger	Ziel
Logger	Alle	BC
Logger	Sensor	BC
Logger	Sensor	Single
Logger	Alle	BC
Sensor	Logger	Single

Abbildung 8.16.: Header verschiedener Message-Typen.

ansonsten muss der gewünschte Empfänger angegeben werden (0x01 für den Busmaster und eine aufsteigende Nummerierung für jeden Teilnehmer). Als Senderadresse verwenden die noch nicht registrierten Teilnehmer eine leere Sourceadresse, die sonst die zugewiesene CAN-ID des Teilnehmers enthalten sollte. Die Reihenfolge dieser Felder wurde so gewählt, um die Möglichkeit des CAN-Acceptancefilters auszunutzen.

### Acceptance-Filter

Da nicht jeder am Bus angehängte Teilnehmer alle Nachrichten empfangen muss (eine Sensoreinheit interessiert sich nicht für die Daten einer anderen Sensoreinheit oder die Konfigurationsmeldungen für diese), müssen die irrelevanten Nachrichten ignoriert werden. Dazu gibt es zwei mögliche Ansätze, entweder das Filtern per Software oder die Verwendung des CAN-Acceptance-Filters. Die Filterung per Software kostet zusätzliche Rechenleistung, sollte also nur angewendet werden, wenn die Performance nicht so wichtig ist oder kein Hardwarefilter verfügbar ist.

Im LPC4088 ist für beide CAN-Controller (Abbildung 8.17) ein Hardwarefilter eingebaut, der insgesamt 1024 Standard-Identifier oder 512 Extended Identifier aufnehmen kann. Neben dem spezifischen Filtern, das nur eindeutige Treffer zulässt, kann der Filter auch so konfiguriert werden, dass ganze Bereiche von Message-IDs akzeptiert werden. Zusätzlich besteht noch die Möglichkeit, den Filter in den Full-CAN-Modus zu stellen, bei dem die empfangenen Nachrichten gleich in einem definierten Buffer zwischengespeichert werden, das Auslesen des Empfangsbuffers entfällt hier also. Da dieser Modus aber nur mit spezifischen Standard-Filtern funktioniert, wurde er hier nicht implementiert.

Der Filter (Abbildung 8.18) besteht aus einer Lookup-Tabelle, die hierarchisch aufgebaut ist: zuerst sind die spezifischen Standard-Filter (16 Bit), dann die Gruppen Standard-Filter (2x16 Bit mit lower- und upper-Bound), dann die spezifischen Extended-Filter (32 Bit) und zuletzt die Gruppen Extended-Filter (2x32 Bit) eingetragen. Trifft nun eine CAN-Nachricht ein, wird geprüft, welcher Art die erhaltene Message-ID ist. Im Anschluss wird die ID zuerst gegen die spezifischen Filter geprüft, wird kein Treffer erzielt, werden die Gruppen-Filter geprüft. Bei einem Treffer wird die Nachricht in den Receive-Buffer geladen, der CAN-Controller löst einen Interrupt aus und die ganze Meldung kann ausgelesen werden.

Im vorliegenden Bussystem wurden die Filter des Loggers so konfiguriert, dass alle Sensor-Nachrichten akzeptiert werden. Bei den Sensoren wird die Konfiguration des Filters in drei Schritten vorgenommen: nach dem Reset eines Sensors wird der Filter so gesetzt, dass nur eine Seriennummer-Anfrage des Loggers durchgelassen wird. Auf diese Anfrage antworten die Sensoren mit dem versenden ihrer Seriennummer und dem Setzen der Broadcast-Filter. Nun können die Sensoren alle Broadcast-Meldungen des Loggers empfangen. Der Logger verschickt nun die zugeteilten CAN-Ids zusammen mit den entsprechenden Seriennummern als Broadcast. Empfängt ein Sensor den Broadcast, prüft er die Seriennummer und, falls diese mit seiner übereinstimmt, setzt die Filter für alle an ihn gerichteten Meldungen.

### 8.2.3. Filesystem

Der Datenlogger legt die Messdaten für jede Sensoreinheit in eine eigene Datei ab. Die Filepointer werden zusammen mit den Konfigurationsdaten im Arbeitsspeicher gehalten. Beim Stoppen einer Sensoreinheit wird die Datei geschlossen und beim erneuten Start der Sensoreinheit eine neue Datei eröffnet. Beim Stopp/Start des Datenloggers erfolgt dies für alle Dateien.

Vor dem Entfernen der SD-Karte muss deshalb der Datenlogger gestoppt werden. Zu diesem Zweck gibt es im Konfigurationsmenü einen eigenen Befehl 'umount SD card'.

Die Dateinamen haben die Form 'sxx\_MMDD\_hhmm.dat', wobei 'xx' der CAN-ID entspricht, 'MMDD' dem Monat/Tag und 'hhmm' der Stunde/Minute des Starts.

Bei Konfigurationsänderungen schreibt der Datenlogger einen entsprechenden Eintrag in die Sensordatei. Der Eintrag wird auf eine neue Zeile geschrieben und mit dem Symbol '#' als Log-Eintrag markiert. Der Eintrag enthält die Uhrzeit und den neuen Wert des Parameters. Am '#' -Symbol erkennt ein

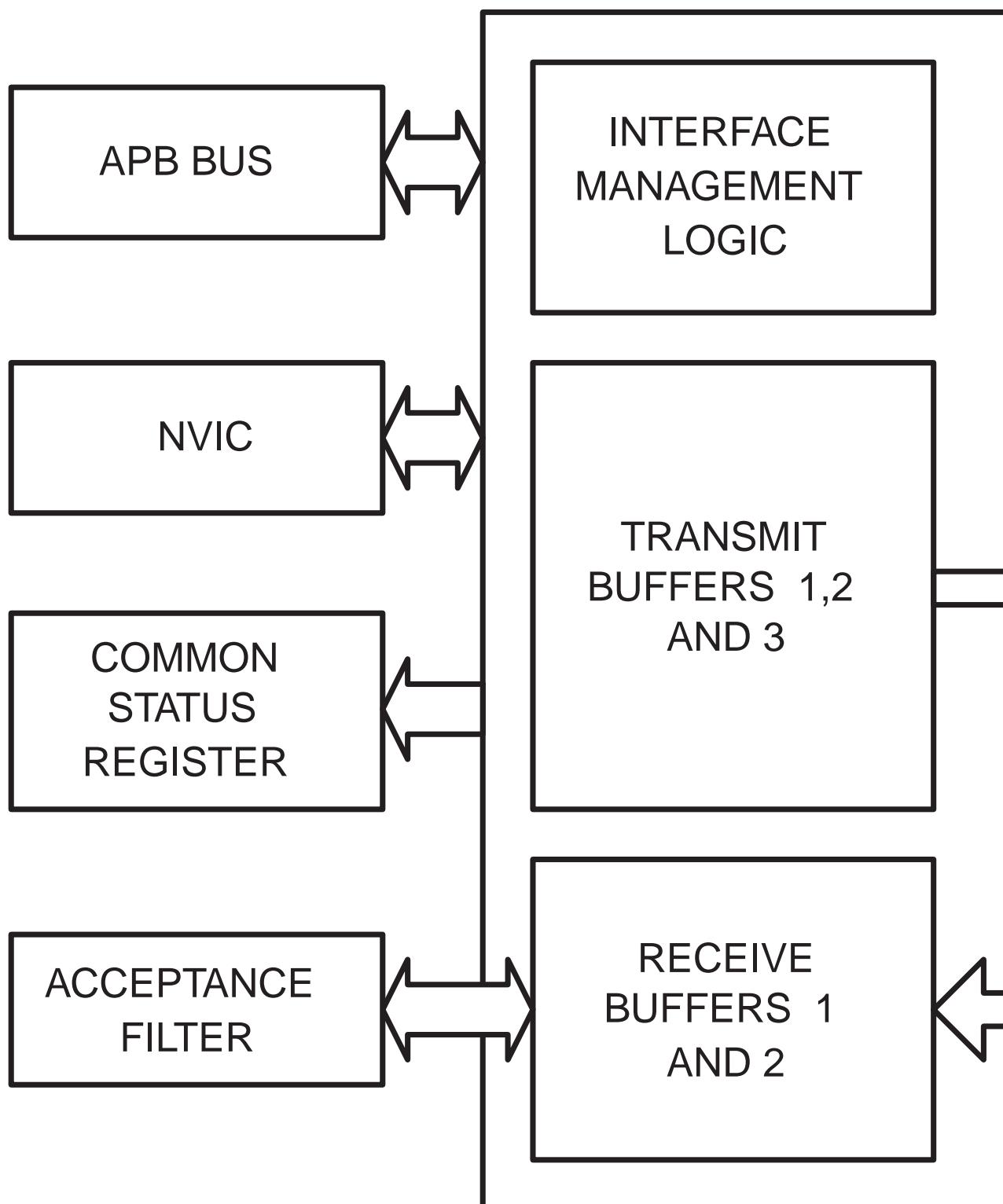


Abbildung 8.17.: Schema des CAN-Controllers.



images/canfilter.pdf

Abbildung 8.18.: CAN-Filter.

Auswertungsprogramm eine solche Informationszeile und kann diese ignorieren oder bei Bedarf auch entsprechend verarbeiten.

### 8.3. Konfiguration

Die Konfiguration der Messstation erfolgt einerseits über die Datei 'config.txt', die sich auf der SD-Karte befindet. Andererseits steht über einen USB-Anschluss eine serielle Schnittstelle zur Verfügung. Mit einem Terminal-Emulator kann auf ein Konfigurationsmenü zugegriffen werden, um die Funktionen der Messstation zu steuern.

Die Verwendung der seriellen Schnittstelle über USB ist in Abschnitt 11.7.1 der Bedienungsanleitung (ab Seite 68) beschrieben.

Die komplette Beschreibung des Konfigurationsmenüs befindet sich in der Bedienungsanleitung ab Abschnitt 11.7.2, Seite 68.

# 9. Resultate

## 9.1. Testfälle

Die wichtigsten Testfälle für die grundsätzliche Funktionalität der Messstation konnten erfolgreich abgeschlossen werden. Für einige Tests blieb jedoch nicht genügend Zeit vor der Fertigstellung des Berichts. Diese Tests werden so weit möglich noch nachgeholt.

Im Kapitel 6 ab Seite 16 sind die Testfälle und die Testergebnisse aufgeführt.

## 9.2. Ereigniserkennung

Der A/D-Wandler des *NXP LPC4088* Mikrokontroller wurde wie geplant in Betrieb genommen und liefert Messdaten, die sich sehr gut mit dem analogen Ausgangssignal des Sensors decken. Abbildung 9.1 zeigt den Vergleich zwischen der Messung von Oszilloskop (blau) und der Sensoreinheit (grün). Die Sensoreinheit arbeitete mit einer Abtastrate von 10000 Hz, das Oszilloskop zeichnete mit einer Abtastrate von 3.125 MHz auf. Um ein Ereignis zu simulieren, wurde ein Golfball auf den Testaufbau (siehe im Verzeichnis Fotos/Testaufbau/ auf der beiliegenden CD) fallen gelassen. Der Vergleich zeigt, dass die beiden Kurven gut übereinstimmen.

Der Betrieb mit 10000 Hz war für unseren Testaufbau genügend, die Peaks traten mit einer Frequenz von ungefähr 2000 Hz auf. Die Frequenz der Peakspitzen variiert sowohl mit der Plattenkonstruktion als auch mit der Korngrösse, Kornbeschaffenheit und der Art des Aufschlags. Daher muss für den tatsächlichen Messbetrieb eine Kalibrierung gemacht werden, um die geeigneten Parameter zu finden.

Für Versuche mit verschiedenen Abtastraten blieb keine Zeit mehr. So können wir zur Zeit nicht sagen, was die höchstmögliche Abtastrate mit der aktuellen Software ist.

## 9.3. Daten-Reduktion und -Speicherung

Die Datenreduktion durch Wahl der verschiedenen Detail-Level ist effizient gelöst und sehr effektiv. Pro Messwert müssen lediglich zwei Vergleiche für die Bestimmung des Signalpegels gemacht werden, sowie zwei Vergleiche mit je einem vorhergehenden Wert für die Bestimmung des Peak-Maximums und des Ereignis-Maximums.

Die Wahl des Detail-Levels beeinflusst lediglich, welche Daten übertragen werden. Auf die Zwischenspeicherung hat sie keinen Einfluss. Vor und nach der Übertragung finden keine komplexen Umrechnungen an den Messdaten statt. Die Bit-Breite wird von 12 Bit auf 8 Bit reduziert. Die grösste Datenreduktion erfolgt durch die Auswahl der relevanten Daten.

### 9.3.1. Rohdaten (raw)

Es werden alle Messpunkte übertragen und gespeichert. Es findet keine Datenreduktion statt. Pro Messpunkt fällt 1 Byte an. Bei einer Abtastrate von 10000 Hz resultiert ein Datenstrom von 10000 Byte/s.

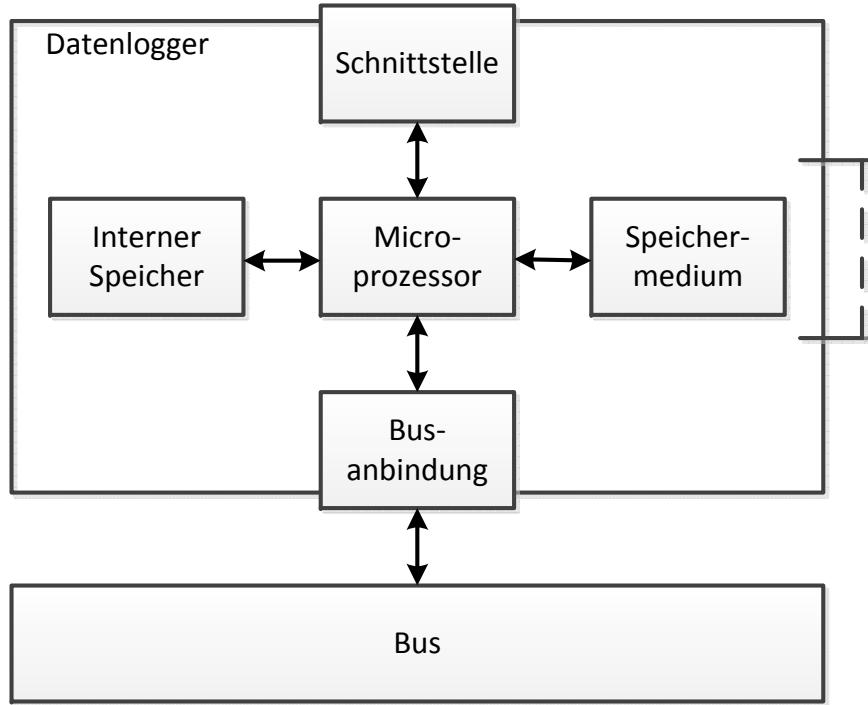


Abbildung 9.1.: Vergleichsmessung mit Oszilloskop und Sensoreinheit.

### 9.3.2. Detaillierte Ereignisdaten (detailed)

Es werden nur die Messpunkte der Ereignisse gespeichert. Messpunkte, die nicht zu einem Ereignis gehören, werden verworfen. Der Datenstrom variiert daher mit der Häufigkeit der Ereignisse. Liegt zu 10 % der Messzeit ein Ereignis vor, wird dies von der WSL als hohes Geschiebeaufkommen eingestuft. Eine solche Periode kann über mehrere Stunden anhalten, ist aber nicht jeden Tag zu erwarten.

Bei 10 % Ereigniszeit wird in diesem Modus eine Datenreduktion um 90 % erzielt. Für die durchschnittliche Ereigniszeit nehmen wir 5 % an. Dann resultiert pro Sensor ein Datenstrom von 500 Byte/s.

### 9.3.3. Peaks (peaks only)

Der Timestamp, die Dauer des Ereignisses, die Anzahl Peaks und alle Peakspitzen werden gespeichert. Pro Ereignis fallen 8 Byte an für die Eckdaten und 2 Byte pro Peak. Ein Ereignis hat im Normalfall weniger als 20 Peaks. Wir nehmen somit 48 Byte pro Ereignis an. Bei einem Ereignis pro Sekunde resultiert ein Datenstrom von rund 50 Byte/s.

### 9.3.4. Minimale Daten (sparse)

Es werden nur der Timestamp, die Dauer des Ereignisses, die Anzahl Peaks und der maximale Ausschlag des Ereignisses übertragen, das sind 8 Byte pro Ereignis. Der Datenstrom erreicht so lediglich 8 Byte/s.

### 9.3.5. Messdauer

Je nach Modus fallen sehr unterschiedliche Datenströme pro Sensor an. Ein Vergleich, wie lange mit einem GByte Speicherplatz gemessen werden kann, ist in Tabelle 9.1 aufgeführt. Anhand dieser Werte kann abgeschätzt werden, wie lange eine Messstation ohne Wechsel des Speichermediums betrieben werden kann.

Detail-Level	Byte/s	Messzeit/GByte
raw	10000	27 h
detailed	500	23 d
peaks only	50	231 d
sparse	8	3.9 yr

Tabelle 9.1.: Vergleich des Datenaufkommens verschiedener Detail-Levels.

## 9.4. Hardware

Für den Datenlogger und die Sensoreinheiten wurde mit der Hilfe von Erich Ruff (ZHAW InES) und Valentin Schlatter (ZHAW InES) eine Leiterplatte entworfen und zwei Gehäusetypen gebaut. Die Leiterplatte wurde so entworfen, dass über die Bestückung entschieden wird, ob ein Datenlogger oder eine Sensoreinheit gebaut wird. Für einen Datenlogger wird die Leiterplatte mit einem SD-Karten-Slot bestückt. Die Sensoreinheit wird mit ein Tiefpassfilter und dem Anschluss für den Sensor bestückt. Beide Varianten enthalten die Spannungsversorgung (12 V auf 5 V), einen CAN-Transceiver und die Anschlüsse für die Kabel. Der Schaltplan und das Leiterplattenlayout befinden sich im Anhang A.3

# 10. Diskussion

## 10.1. Ergebnisse

### 10.1.1. Funktionale Anforderungen

Wir haben eine Messstation entwickelt, die an einem Datenlogger mehrere Sensoreinheiten betreiben kann und die Messdaten nach definierbaren Kriterien verarbeitet, die Datenmenge um einen wählbaren Faktor reduziert und die interessanten Daten über ein Bussystem überträgt. Die Messstation speichert die Daten auf einem einfach austauschbaren Speichermedium. Die Konfiguration der Messstation kann im Feld überprüft und angepasst werden. Es ist möglich, die Sensoreinheiten im Feld zu überwachen und ihre Konfiguration anzupassen, ohne dass sie aus der Installation im Bach entfernt werden müssen. Die Sensoreinheiten sind in der Lage, Ereignisse nach konfigurierbaren Kriterien zu erkennen.

### 10.1.2. Nichtfunktionale Anforderungen

Zum Stromverbrauch der Anlage lagen beim Verfassen des Berichts noch keine Daten vor.

Mit den verschiedenen Detail-Levels und der Abtastrate kann die Datenqualität den Bedürfnissen in einem gewissen Rahmen angepasst werden.

Die vorliegenden Geräte sind in wasserdichten Gehäusen untergebracht. Die Gehäuse sind nicht stabil genug, um Einschlägen von grossen Geschiebekörnern zu widerstehen. Daher sind weiterhin Stahlplatten notwendig, um die Sensoreinheiten zu schützen. Mit dieser Messstation können in der Versuchsanlage für Wasserbau (VAW) an der ETH Hönggerberg in einer Versuchsrinne Tests durchgeführt werden. Die Installation in einem Bergbach ist beim gegenwärtigen Projektstand nicht möglich.

## 10.2. Rückblick auf die Aufgabenstellung

Die Aufgabenstellung sah vor, die Messstation fertig zu entwickeln. Schon zu Beginn der Arbeit zeichnete sich jedoch ab, dass die Unbekannten zu zahlreich sind:

- Die Neuentwicklung des Algorithmus für die Ereigniserkennung.
- Der Entwurf der Messdatenerfassung mit A/D-Wandler und Timestamp.
- Der Entwurf und die Entwicklung eines Protokolls für die Übertragung der Konfigurations- und Steuerbefehle und der Messresultate über CAN-Bus.
- Die Notwendigkeit sehr grosser Konfigurierbarkeit aufgrund des unbekannten Verhaltens der endgültigen Konstruktion des Sensorträgers.

Deshalb wurden grosse Sicherheitsmargen bei Rechenleistung und Fähigkeiten des Mikrokontroller gewählt. Mit dem Auftraggeber wurde vereinbart, dass das Ziel der Arbeit angepasst wird auf die Entwicklung eines Prototypen, an dem das Konzept getestet werden kann.

So konnten wir uns mehr um die Zusammenstellung der Hardware und die Entwicklung der Software kümmern.

Die Implementation des Busprotokolls gestaltete sich schwieriger als angenommen. Die ersten Testimplementierungen sahen zwar vielversprechend aus, es tauchten bei der Zusammenarbeit zwischen dem

Acceptance-Filter, dem Interrupt-Handling und dem RTOS aber Probleme auf, die massive Verzögerungen verursachten. Für den Acceptance-Filter wird von mbed.org eine Bibliothek (Library) zur Verfügung gestellt, sie ist aber nicht funktionsfähig und musste deshalb von uns implementiert werden.

Die Fehlersuche auf einem Bussystem gestaltet sich schwierig, da debuggen auf einem Bus, der Daten mit 1 MHz übermittelt, zu Timingproblemen führt. Auch die Verwendung von Ausgaben über den Serial-Port war ausgeschlossen, da diese in Interruptroutinen zu einem Hardfault führen. So blieb schlussendlich nichts anderes übrig, als das ganze Busprotokoll zu zerlegen und die einzelnen Teile nach dem oder den Fehlern zu untersuchen, was sehr zeitintensiv war.

Dies verzögerte die Fertigstellung des Projekts stark und verhinderte ausgiebige Tests. Die geplante Funktionalität wurde bis zur Fertigstellung des Berichts nur teilweise erreicht und getestet.

## 10.3. Ausblick

Mit dem vorliegenden Prototyp sind erste Tests in der Versuchsrinne in der VAW möglich. So können die Hydrologen Erfahrungen mit dem neuen Messsystem sammeln. Diese Erfahrungen sollten verwendet werden, um Mängel am System und fehlende Funktionalität aufzudecken.

Die Software ist nicht fertig und es sind einige Ideen für zusätzliche Ziele vorhanden.

### 10.3.1. Hardware

Die vorliegende Hardware demonstriert lediglich die Machbarkeit eines solchen Systems. Für einen Einsatz im Bergbach bedarf es einer Weiterentwicklung der Hardware: Die Quickstart Boards enthalten viele unbenutzte Bauteile. Mit der Entwicklung einer optimierten Hardware, die nur die wirklich nötigen Teile enthält, könnten wesentlich kleinere Gehäuse gebaut werden. Ein Gehäuse von 60x50x40 mm sollte machbar sein. Mit dieser Grösse sollte es auch möglich sein, mehrere Sensoreinheiten in einer Elastomerplatte zu vergiessen. Ein Thema für ein Folgeprojekt könnte die Entwicklung einer serienreifen Hardware für dieses Messsystem sein.

Die vorliegende Hardware ist nicht auf einen MEMS-Beschleunigungssensor beschränkt. Es könnten auch andere Sensorbauarten angeschlossen werden, wenn die Messwerte über eine analoge Spannung ausgegeben werden. Die Verwendung mehrerer Sensoren wäre mit der vorliegenden Hardware möglich. Die Software müsste dafür aber angepasst werden.

### 10.3.2. Software

Die vorliegende Hardware ist mit dem verwendeten Algorithmus nur schwach ausgelastet. Es besteht grosser Spielraum für andere, komplexere Algorithmen. Die Versuche an der VAW könnten hier eine neue Stossrichtung aufzeigen.

Falls eine höhere Datenqualität (bis 12 Bit statt nur 8 Bit) gewünscht wird, könnte das Busprotokoll angepasst werden.

Es wäre auch möglich, einen Algorithmus zu entwickeln, der das aktuelle Geschiebeaufkommen verfolgt und je nach Vorgaben in einen anderen Betriebsmodus wechselt. Bei hohem Geschiebeaufkommen könnten zum Beispiel einige Sensoren in höhere Detail-Level versetzt werden, um genauere Daten zu erhalten. Bei knapp werdendem Speicher würden die Detail-Levels automatisch gesenkt.

Der Bedienung der Messstation lässt noch einige Wünsche nach mehr Komfort offen, besonders im Bereich der Datei-Verwaltung. So ist es zum Beispiel noch nicht möglich, über den Datenlogger einzelne Dateien zu löschen. Dies wäre jedoch wünschenswert, wenn schnell freier Platz geschaffen werden soll. Die Auswahl aus mehreren Konfigurationsdateien wäre auch angenehm. Zudem kann noch nicht angezeigt werden, wie viel freier Platz noch auf der SD-Karte vorhanden ist.

In der vorliegenden Version werden die Ereignisdaten in lesbbarer Form in die Datei gespeichert. Dafür wird mehr Speicherplatz als in einem binären Format. Die Verwendung eines binären Formats würde die verfügbare Messdauer beträchtlich erhöhen. Zusätzlich könnte ein Kompressionsalgorithmus verwendet werden, um den benötigten Speicherplatz weiter zu verringern.

# 11. Bedienungsanleitung

ngsanleitung

## 11.1. Produktbeschrieb

Die Messstation wurde entwickelt, um Geschiebemessungen in einem Bach oder Fluss zu machen. Als Vorbild hat eine Messstation mit Geophonen als Sensoren gedient, die einen Embedded-PC als Auswertungsrechner einsetzt. Der bauliche Aufwand für eine solche Messstation ist ziemlich gross, da viele Kabel verlegt und vor dem Geschiebe geschützt werden müssen. Mit dem Ziel, den Aufwand für die Konstruktion und die Verkabelung zu reduzieren, wurde eine neue Messstation entwickelt, die über ein Bussystem kommuniziert und die Messdaten gleich am Sensor auswertet. Auf diese Weise müssen nur noch die gewünschten, vorher spezifizierten Messdaten übertragen und gespeichert werden.

Mit der neuen Messstation können bis zu 20 Sensoren an einer Kontrolleinheit angeschlossen werden, die alle Messdaten aufzeichnet. Diese Anzahl kann nach eingehenden Tests wahrscheinlich noch erhöht werden. Die Obergrenze wird auch von der gewünschten Messdatenqualität abhängen, da das Bussystem eine begrenzte Übertragungskapazität hat.

## 11.2. Aufbau der Messstation

Die Messstation besteht aus einem Datenlogger der über ein Bussystem mit den Sensoreinheiten verbunden ist. Die Stromversorgung (12 V Gleichspannung) erfolgt über zwei zusätzliche Leitungen im gleichen Kabel wie für das Bussystem verwendet wird. Das Kabel verläuft vom Datenlogger zur ersten Sensoreinheit und dann jeweils von einer zur nächsten Sensoreinheit (Abbildung 11.1). Der Datenlogger kann bis 20 Sensoren verwalten. Die tatsächliche maximale Anzahl der verwendbaren Sensoren ist auch von der Stromversorgung und dem gewünschten Detail-Level der Messdaten abhängig.

messstation

Die Sensoreinheiten sind in einem wasserdichten Gehäuse verbaut und über wasserdichte Steckverbinder (IP68) untereinander verbunden.

## 11.3. Datenlogger

Das Herzstück der Anlage ist der Datenlogger, der die Kontrolle über die Kommunikation hat, die Messdaten speichert und den Anschluss eines Computers für die Konfiguration der Anlage ermöglicht. Im



Abbildung 11.1.: Schematischer Aufbau einer Messanlage.

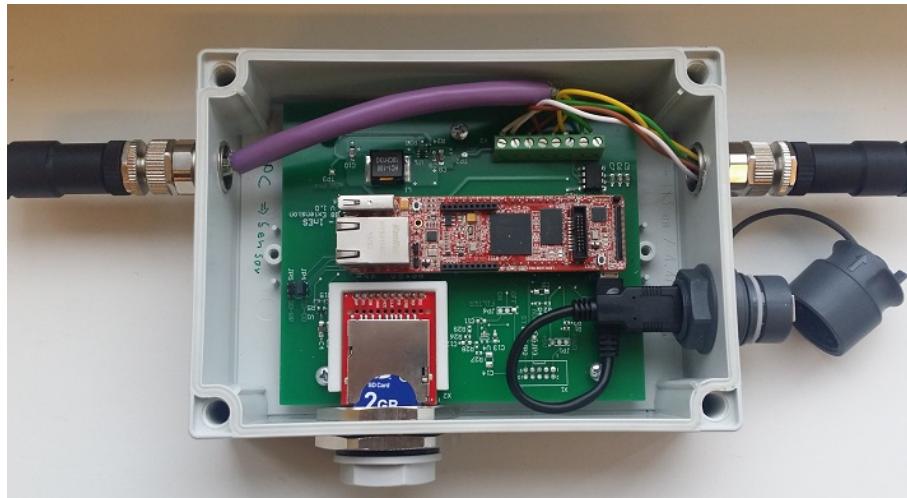


Abbildung 11.2.: Der Datenlogger in geöffnetem Zustand.

Datenlogger arbeitet ein *ARM Cortex-M4* Prozessor mit 120 MHz Taktgeschwindigkeit. Dieser Prozessor steuert die Kommunikation und wandelt die Messdaten für die Speicherung in lesbare Information um.

### 11.3.1. Anschlüsse

Der Datenlogger verfügt über vier Anschlüsse. Abbildung 11.2 zeigt einen geöffneten Datenlogger.

#### Stromversorgung

Ein Anschluss ist für die Stromversorgung, in Abbildung 11.2 auf der linken Seite. Hier werden 12V Gleichspannung angeschlossen. Der Stromverbrauch ist abhängig von der Anzahl angeschlossener Sensoren.

#### Busanschluss

Über den zweiten Anschluss wird das Kabel des Bussystems angeschlossen, in Abbildung 11.2 auf der rechten Seite oben. Der Steckverbinder ist wasserdicht (IP68, Binder), damit kein Regen- oder Kondenswasser in den Datenlogger eindringen kann. Die Messstation verwendet CAN-Bus für die Kommunikation, da dieser Standard ein sehr robustes Protokoll für die Sicherstellung der korrekten und fehlerfreien Übertragung hat.

#### Schnittstelle zum Computer

Ein wasserdichter Steckverbinder (in Abbildung 11.2 auf der rechten Seite unten) für ein USB-Kabel erlaubt den Anschluss eines Computers oder auch einen Smartphones. Über eine serielle Schnittstelle kann dann die Messstation konfiguriert werden. Der Anschluss eines Computers und die Konfiguration ist in den Abschnitten 11.7.1 bis 11.7.3 ab Seite 68 im Detail beschrieben.

#### Speicherkarte

Die Speicherkarte kann über einen wasserdichten Schraubdeckel (Abbildung 11.2: unten) ausgewechselt werden, ohne dass das ganze Gehäuse des Datenloggers geöffnet werden muss. Der Umgang mit der Speicherkarte ist im Abschnitt 11.7.3 ab Seite 70 erklärt.

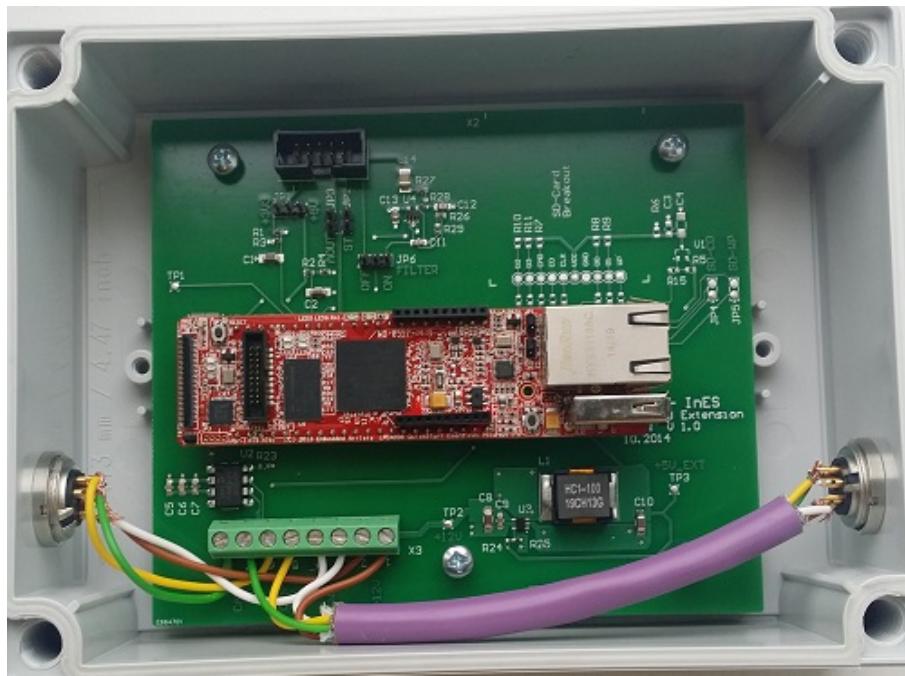


Abbildung 11.3.: Die Sensoreinheit in geöffnetem Zustand.

## 11.4. Sensor

Die Sensoreinheiten der Anlage verfügen über einen Beschleunigungssensor, einen Mikroprozessor und zwei Anschlüsse für das Bussystem (Abbildung 11.3). Der Sensor misst über die Beschleunigung die Vibrationen, die vom Einschlag des Geschiebes verursacht werden. Der Mikroprozessor wertet die Messdaten aus und erkennt nach vorher definierten Kriterien die Ereignisse, deren Daten gesammelt werden sollen. Diese bereinigten Daten werden dann über das Bussystem an den Datenlogger übertragen.

### 11.4.1. Beschleunigungssensor

Der Beschleunigungssensor (*Analog Devices ADXL001-70*) misst Beschleunigungen zwischen -70 g und +70 g. Ein g entspricht der Beschleunigung durch die Erdanziehungskraft, ungefähr  $10 \text{ m/s}^2$ . Sollten die Vibrationen stärkere Beschleunigungen erzeugen, kann entweder die Konstruktion der Messanlage angepasst werden um die Vibrationen abzuschwächen oder der Sensor kann ausgetauscht werden. Aus der Baureihe *ADXL001* von *Analog Devices* sind auch Sensoren mit Messbereichen von  $\pm 250 \text{ g}$  und  $\pm 500 \text{ g}$  erhältlich. Diese Modelle geben Messsignale im gleichen Spannungsbereich aus wie das verwendete Modell ADXL001-70 und können daher ohne Umkonfiguration oder Anpassung der Software verwendet werden. Der Sensor muss mit einem Objekt in festem Kontakt stehen, auf dem die Geschiebekörper aufschlagen. Dazu muss entweder das Gehäuse mit einer Platte verbunden werden, oder das Sensorkabel wasserfest aus dem Gehäuse zu einer Platte geführt werden.

### 11.4.2. A/D-Wandler

Die Messsignale vom Beschleunigungssensor werden von einem 12-Bit-A/D-Wandler digitalisiert. Der A/D-Wandler kann theoretisch mit bis zu 200'000 Hz betrieben werden.

### 11.4.3. Mikroprozessor

Der ARM Cortex-M4 mit 120 MHz Taktfrequenz hat weit mehr Rechenleistung, als für die vorliegende Anwendung benötigt wird. Sollten komplexere Algorithmen für die Erkennung von Ereignissen, für die Filterung der Messdaten oder für die Verarbeitung der Signale benötigt werden, wäre dies dank der bereits vorhandenen Rechenleistung möglich.

### 11.4.4. Anschlüsse

Die Sensoreinheit hat zwei Anschlüsse für das Bussystem. Mit dieser Bauweise ist es möglich, die Stichleitung zum Transceiver möglichst kurz zu halten, um die Signalqualität möglichst wenig negativ zu beeinflussen. Die Anschlüsse sind wassererdicht (IP68).

## 11.5. Bussystem

Als Bussystem wird CAN-Bus verwendet. Mit einer Datenübertragungsrate von 1 MBit/s bis zu einer Kabelgesamtlänge von 40 m, resp 125 kBit/s bis zu 500 m eignet sich CAN-Bus für diese Messstation sehr gut. Es sind keine Kabellängen zu erwarten, die weit über 40 m hinausgehen. CAN-Bus ist des Weiteren sehr robust gegenüber Umwelteinflüssen, die die Übertragung stören könnten.

### 11.5.1. Kabel

Die Kabel der Messstation verfügen über 4 Leitungen, die jeweils paarweise verdrillt sind. Zusätzlich zu den Leitungen ist eine Zugentlastung zum Schutz vor mechanischer Beschädigung im Kabel enthalten. Um die zwei Aderpaare und die Zugentlastung ist ein metallisches Geflecht als Schild angebracht, das sowohl gegen mechanische Beschädigung als auch gegen elektrische Störeinflüsse schützt. Die PVC-Hülle des Kabels ist ein zusätzlicher mechanischer Schutz.

Ein Aderpaar wird für die Spannungsversorgung verwendet, das zweite Aderpaar für die Kommunikation auf dem Bussystem.

### 11.5.2. Abschlusswiderstände

Damit die Bus-Signale an den Enden der Leitung nicht reflektiert werden und die Übertragung stören, müssen an beiden Enden des Bussystems Abschlusswiderstände eingesetzt werden. Die Abschlusswiderstände ( $120 \Omega$ ) verbinden die Anschlüsse CANH und CANL auf der Leiterplatte und werden wie in Abbildung 11.4 angebracht.

### 11.5.3. Steckverbinder

Die Steckverbinder sind fünfpolig und wassererdicht (IP68). Die Pole sind gemäss Tabelle 11.1 bestückt, die Polnummerierung in Stecker und Buchse sind in den Abbildungen 11.5 und 11.6 dargestellt.

Auf der Leiterplatte der Sensoreinheit ist eine Schraubklemmenleiste mit 8 Anschlüssen angebracht. Hier werden die von den Buchsen kommenden Leitungen angeschlossen. Jeweils zwei benachbarte Anschlüsse sind zusammengeschaltet, damit jede Leitung einzeln fest verschraubt werden kann (Abbildung 11.7). Es wird nicht empfohlen, zwei gleiche Leitungen in einer einzelnen Schraubklemme zusammen zu verschrauben.

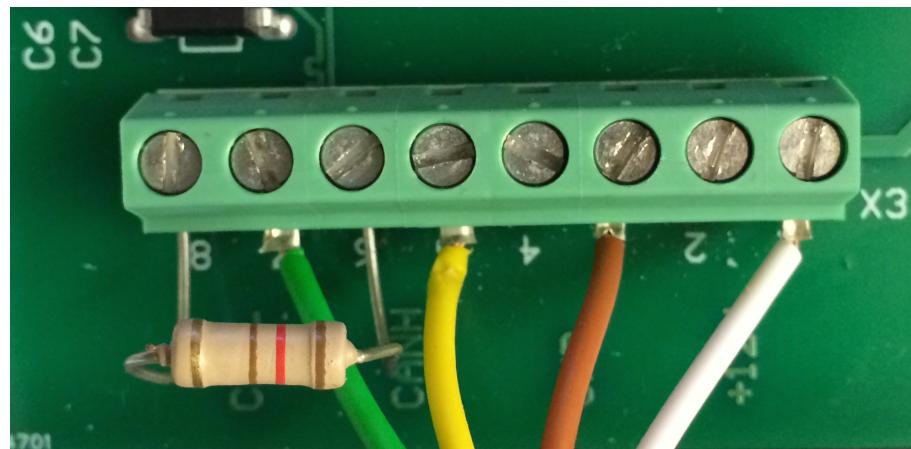


Abbildung 11.4.: Anbringung des Abschlusswiderstands.

Pol Nr.	Leitung	Anschluss Schraubklemme	Aderfarbe
1	Spannungsversorgung +12 V	+12V	weiss
2	Spannungsversorgung 0 V	GND	braun
3	unbenutzt		
4	CAN-Bus High	CANH	gelb
5	CAN-Bus Low	CANL	grün

Tabelle 11.1.: Steckerbestückung des Bussystems.

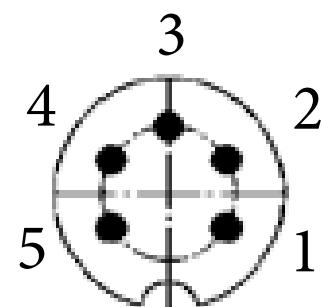
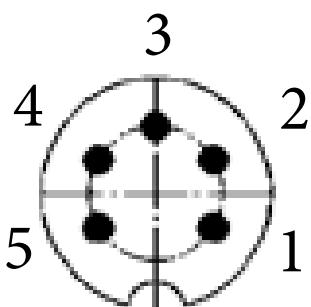


Abbildung 11.5.: Polnummerierung in der Buchse. Abbildung 11.6.: Polnummerierung im Stecker.

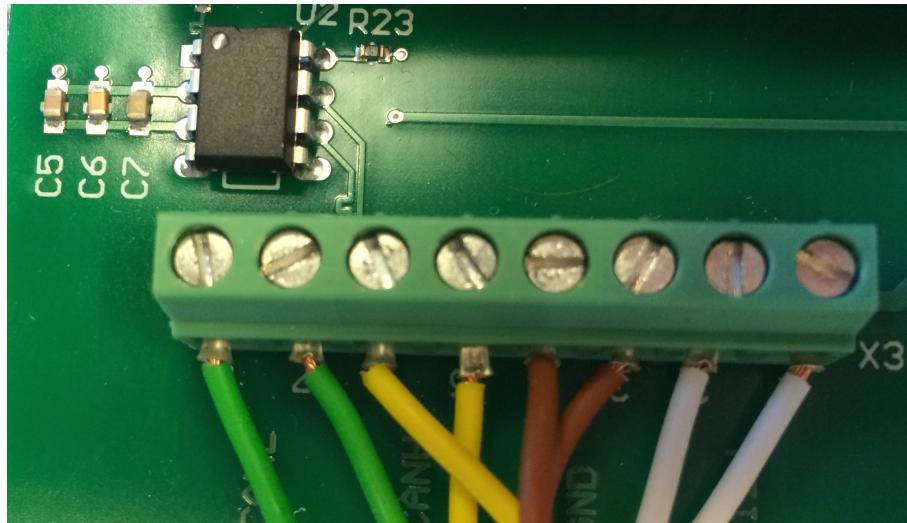


Abbildung 11.7.: Anschluss der Leitungen an den Schraubklemmen.

## 11.6. Ereignis

Als Ereignis wird eine Signalform bezeichnet, die einem Einschlag eines Steins auf dem Sensor entspricht. Um Ereignisse zu erkennen, wird ein Threshold für den Signalpegel definiert. Wird dieser Wert überschritten, beginnt ein Ereignis. Sobald der Signalpegel für eine gewisse Zeit unterhalb des Thresholds geblieben ist, ist das Ereignis beendet. Das Verfahren der Ereigniserkennung wird im Abschnitt 8.1.3 ab Seite 36 im Detail erklärt.

Ein Beispiel eines Ereignisses ist in Abbildung 11.8 gegeben. Es handelt sich um den Einschlag eines Golfballs auf einer Aluminiumplatte, unter der der Beschleunigungssensor montiert ist. Die schwarz gestrichelten Geraden zeigen den Threshold, die roten Geraden zeigen den erkannten Anfang und das Ende des Ereignisses. Das Ende des Ereignisses wird von der Ereigniserkennung auf jenen Messpunkt gelegt, an dem der Signalpegel den Threshold vor dem Timeout das letzte Mal unterschritten hat. Die Messwerte vor dem Timeout werden je nach Detail-Level abgeschnitten. Die folgende Beschreibung der Detail-Level bezieht sich auf dieses Beispiel-Ereignis.

### 11.6.1. Detail-Level

Die Messstation kann Daten mit unterschiedlichen Detailgraden speichern. Jede Sensoreinheit kann individuell auf einen Detail-Level konfiguriert werden.

- raw
- detailed
- peaks only
- sparse
- off

#### Rohdaten (raw)

Im Modus 'raw' werden Rohdaten ohne Ereigniserkennung übertragen und gespeichert. Es resultiert eine lückenlose Erfassung aller Messpunkte. Abbildung 11.9 zeigt in Grün, welche Daten des Beispieleignisses im Modus 'raw' übertragen werden.

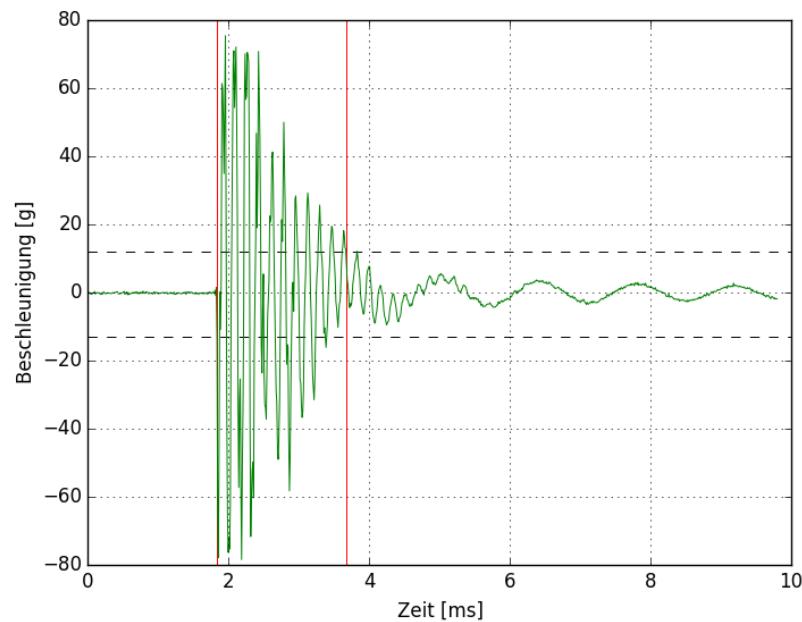


Abbildung 11.8.: Beispiel von Rohdaten.

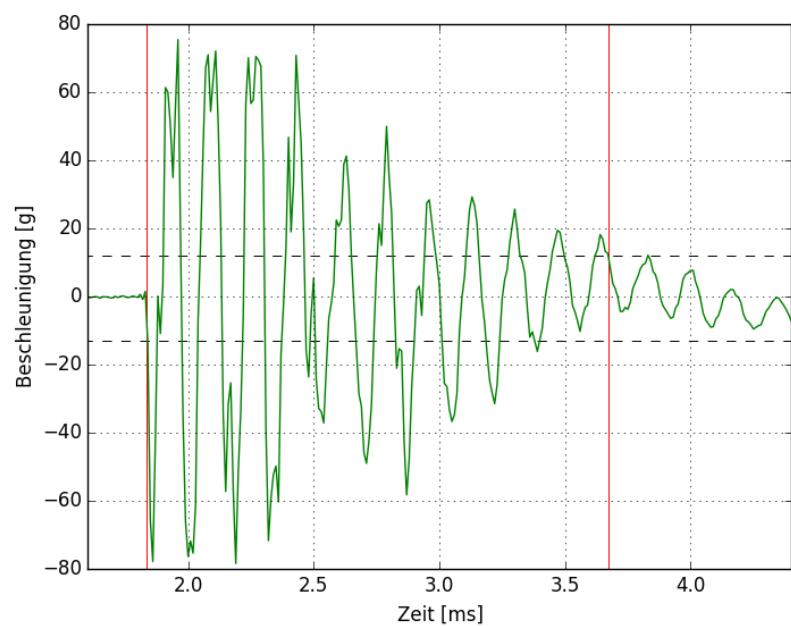


Abbildung 11.9.: Detail-Level 'raw'.

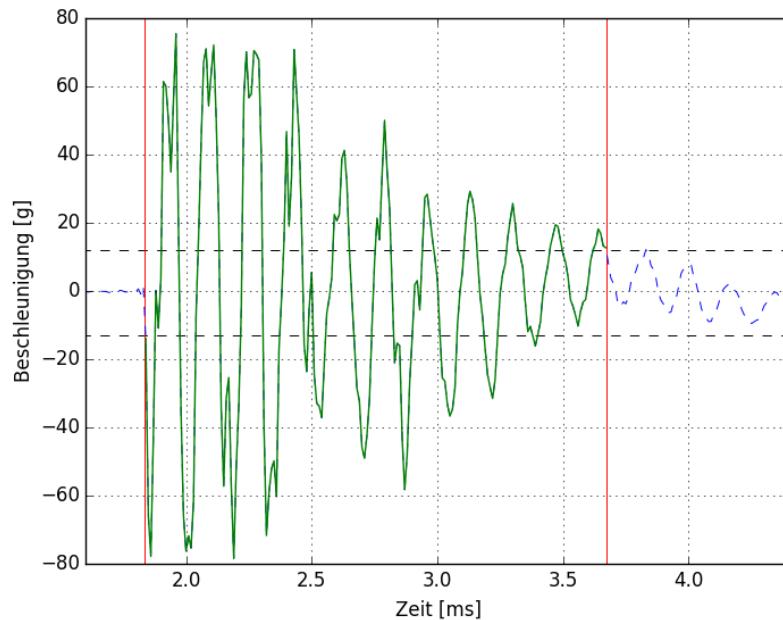


Abbildung 11.10.: Detail-Level 'detailed'.

**Datenaufkommen** Das Datenaufkommen ist in diesem Modus am grössten. Für jedes Sample wird ein 8-bit Wert abgespeichert. Bei einer Abtastrate von 10000 Hz fallen also 10000 Byte Daten pro Sensor und Sekunde an.

**Beispieldaten** In der Datei wird beim Start der Aufzeichnung eingetragen, wann die Aufzeichnung (in Sekunden seit 0:00 Uhr, 1.1.1970) beginnt und wie viele Samples gemessen werden sollen. Danach folgen nur noch Datenwerte zwischen -128 und 127. Listing 11.1 zeigt den Anfang und Ende eines Eintrags aus dem Rohdatenmodus. Die Messwerte sind durch Kommas getrennt, ein Strichpunkt markiert das Ende des Eintrags.

```
1 start=1417865674, samples=600000;0,3,15,-12,15, ... 97,12,-45,-100;
```

Listing 11.1: Beispieldaten auf Detail-Level 'raw'

### Detaillierte Ereignisdaten (detailed)

Die Ereigniserkennung sucht den Beginn und das Ende des Ereignisses. Es werden sämtliche Samples des Ereignisses übertragen. Im Konfigurationsmenü wird dieser Modus als 'detailed' bezeichnet. In Abbildung 11.10 entspricht die grüne Kurve den gespeicherten Daten.

**Datenaufkommen** Pro Sample wird ein Byte abgespeichert, der Startzeitpunkt und die Dauer belegen zusammen sechs Byte pro Ereignis. Die Datenmenge variiert mit der Dauer der Ereignisse. Es wird geschätzt, dass bei hohem Ereignisaufkommen während etwa 10 % der Messzeit ein Ereignis vorliegt. Somit wird die Datenrate ungefähr ein Zehntel der Abtastrate in Byte/Sekunde sein.

**Beispieldaten** Das Format der Datenablage ist im Detail-Level 'detailed' genau gleich wie bei 'raw'. Der Eintrag enthält jedoch nur Messwerte von einem einzigen Ereignis und ist daher viel kürzer (Listing 11.2).

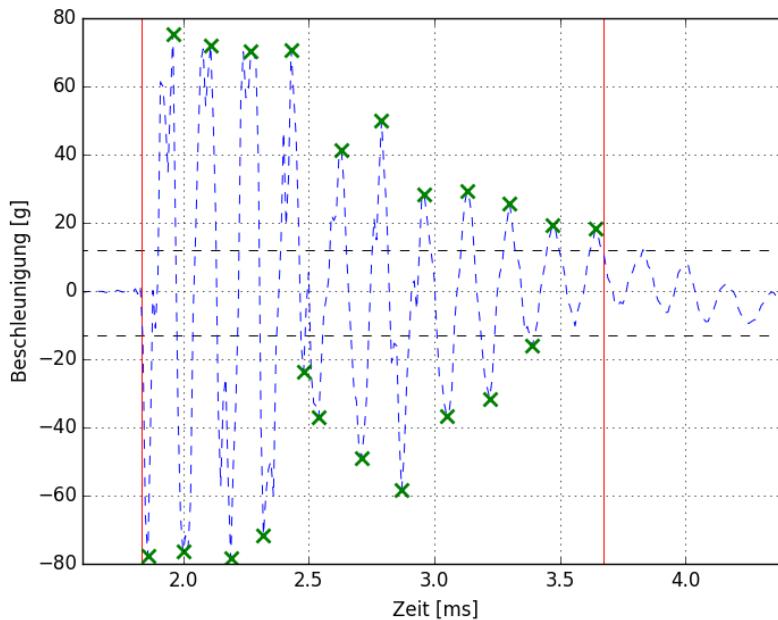


Abbildung 11.11.: Detail-Level 'peaks'.

```
1 start=1417865674, samples=187; -125, -107, -12, 15, ... 43, 7, -45;
```

Listing 11.2: Beispieldaten auf Detail-Level 'detailed'

### Peaks (peaks only)

Die Ereigniserkennung sucht im Modus 'peaks only' den Beginn und das Ende und somit die Dauer des Ereignisses. Außerdem werden alle Peakspitzen mit dem Timestamp und der Höhe des Ausschlags gespeichert.

**Datenaufkommen** Für die Eckdaten (Beginn, Dauer, Anzahl Peaks, maximaler Ausschlag) fallen 8 Byte Daten an. Jeder Peak benötigt zwei Byte: ein Byte für die Anzahl Samples seit dem letzten Peak und ein Byte für die Höhe des Ausschlags. In der Abbildung 11.12 zeigt die blau gestrichelte Linie den Verlauf der Messkurve, die roten Geraden markieren Beginn und Ende des Ereignisses. Die grünen 'X' markieren die Peakspitzen, die übertragen werden.

**Beispieldaten** Der Eintrag enthält die Startzeit in Sekunden, die Dauer in Samples und die Anzahl Peaks. Danach folgen Wertepaare mit dem Abstand zum vorherigen Peak (resp. zum Beginn des Ereignisses) in Samples und der Peakhöhe (Listing 11.3). Der Maximalwert des Ereignisses entspricht dem höchsten Peak und wird deshalb nicht extra gespeichert.

```
1 start=1417865674, samples=187, nrpeaks=22; 3 -120, 12 110, ..., 18 68, 7  
-55;
```

Listing 11.3: Beispieldaten auf Detail-Level 'peaks only'

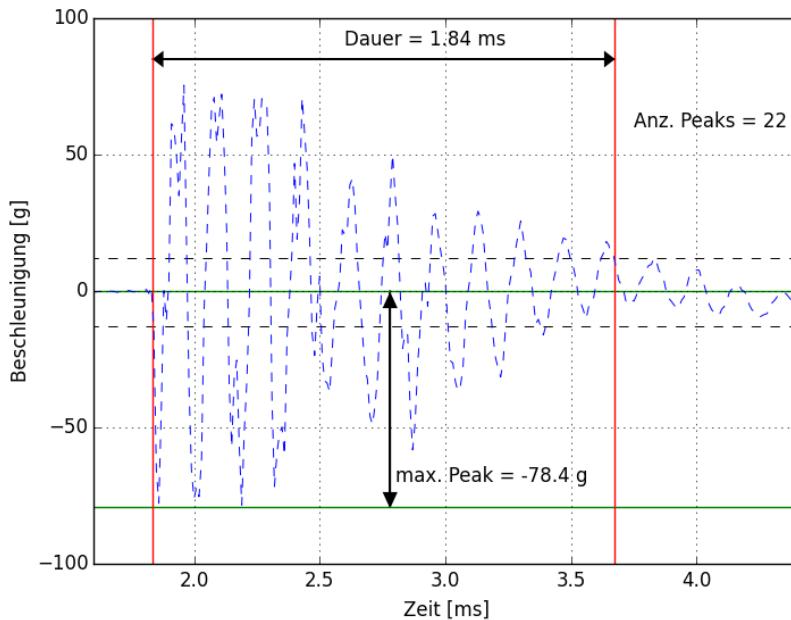


Abbildung 11.12.: Detail-Level 'sparse'.

### Minimale Daten (sparse)

Im Modus 'sparse' werden nur Eckdaten des Ereignisses gespeichert: Beginn und Dauer, Anzahl Peaks und maximaler Ausschlag. Dafür genügen acht Byte pro Ereignis. Die Eckdaten sind in Abbildung 11.12 dargestellt.

**Datenaufkommen** Der Timestamp für den Beginn des Ereignisses benötigt vier Byte, die Dauer (Anzahl Samples) zwei Byte. Die Anzahl Peaks und der maximale Ausschlag belegen je ein Byte.

**Beispieldaten** Der Eintrag enthält die Startzeit in Sekunden, die Dauer in Samples, die Anzahl Peaks und den maximalen Ausschlag (Listing 11.4).

```
1 start=1417865674, samples=187, nrpeaks=22, max=-125;
```

Listing 11.4: Beispieldaten auf Detail-Level 'sparse'

### Inaktiv (off)

Die Sensoreinheit kann in den Detail-Level 'off' gesetzt werden. In diesem Fall startet sie auch keine Messung, wenn der Datenlogger alle Sensoreinheiten startet.

In diesem Detail-Level fallen keine Daten an.

## 11.7. Konfiguration

### 11.7.1. Anschluss eines Computers

Am USB-Anschluss des Datenloggers kann ein Computer angeschlossen werden, um auf die serielle Schnittstelle des Datenloggers zuzugreifen. Um die serielle Schnittstelle zu verwenden, wird ein Terminal-Emulator wie *PuTTY* oder *minicom* benötigt. Um mit *PuTTY* eine Verbindung aufzubauen, muss die Schnittstelle und die Übertragungsrate (Baud) angegeben werden. Die Übertragungsrate ist 9600 baud, die Schnittstelle kann variieren.

**Windows** Unter *Windows* erfolgt die Verbindung auf eine der COMx-Schnittstellen. Die Nummer der COM-Schnittstelle kann im Gerät-Manager herausgesucht werden, die Bezeichnung lautet 'mbed Serial Port (COMx)', wobei 'x' eine Nummer ist. In *PuTTY* muss nur 'COMx' angegeben werden.

**Linux** Unter *Linux* findet man die Schnittstellenbezeichnung mit dem Befehl 'ls /dev/ttyACM\*' heraus, in *PuTTY* wird dann '/dev/ttyACMx' angegeben.

**Mac OS X** Unter *Mac OS X* lautet der Befehl 'ls /dev/tty.usbmodem\*', der in einem Terminal eingegeben werden muss. Als Terminal-Emulator kann 'screen' verwendet werden. Auf Apple Mac Computern mit USB 3.0 kann es zu Schwierigkeiten mit der Verbindung kommen. Den Herstellern des Prozessorboards ist dies bekannt, sie arbeiten an einer Lösung.

Die Einstellungen für die serielle Schnittstelle sind normalerweise bereits korrekt gesetzt. Es werden 8 Datenbits verwendet, 1 Stopbit und keine Parität (parity).

Weitere Hilfe für die Verwendung eines Terminal-Emulators findet man unter <http://developer.mbed.org/handbook/Terminals>.

### 11.7.2. Menü

Beim Herstellen der Verbindung über einen Terminal-Emulator wird das Basis-Menü angezeigt. Durch Eingabe der Zahl wird der entsprechende Menü-Eintrag gewählt. Im Folgenden wird das gesamte Menü im Detail beschrieben.

Das Basis-Menü (siehe Listing 11.7.2) listet alle Überwachungs- und Konfigurations-Möglichkeiten auf.

```

1) list files
2) format SD card
3) mount SD card
4) unmount SD card
5) logger status
6) start/stop logging
7) sensor parameters
8) sensor states
9) reset timestamp
10) internal clock
11) config file

```

**Dateien auflisten** Mit dem Befehl 'list files' wird eine Liste aller Dateien auf der SD-Karte angezeigt. Die Liste enthält die Dateigröße sowie den Dateinamen.

**SD-Karte formatieren** Um die SD-Karte für den ersten Gebrauch vorzubereiten, sollte sie formatiert werden. Dies erfolgt von Vorteil auf einem Computer, kann aber auch im Datenlogger mit dem Befehl 'format SD card' gemacht werden.

**SD-Karte anmelden** Nach dem Einsetzen einer SD-Karte erkennt der Datenlogger dies normalerweise automatisch. Es kann jedoch vorkommen, dass der Datenlogger auf die neue Karte aufmerksam gemacht werden muss. Dies erfolgt mit dem Befehl 'mount SD card'.

**SD-Karte abmelden** Vor dem Entfernen der SD-Karte müssen alle Dateien geschlossen werden. Dies erfolgt mit dem Befehl 'unmount SD card'.

**Status des Datenloggers** Mit dem Befehl 'logger status' werden einige Betriebszustandsdaten des Datenloggers angezeigt.

**Aufzeichnung starten/stoppen** Um die Aufzeichnung im ganzen System zu starten oder zu stoppen wird der Befehl 'start/stop logging' verwendet.

**Sensor-Einstellungen** Mit dem Befehl 'sensor parameters' kann eine einzelne Sensoreinheit oder alle Sensoreinheiten zusammen konfiguriert werden.

**Status der Sensoreinheiten** Der Betriebszustand aller angeschlossenen Sensoreinheiten kann mit dem Befehl 'sensor state' aufgelistet werden.

**Timestamp zurücksetzen** Um den Timestamp in allen Sensoreinheiten auf Null zurückzustellen, wird der Befehl 'reset timestamp' verwendet.

**Interne Uhr** Die interne Uhr wird mit dem Befehl 'internal clock' eingestellt.

**Konfigurations-Datei** Mit dem Befehl 'config file' wird die Konfiguration der Sensoren abgespeichert oder aus einer Datei eingelesen.

### 11.7.3. Befehle

#### Dateiliste

```
1 f:      90 /mci/config.txt
2 f:     122086 /mci/s02_1214_0530.dat
3
4 0) exit
```

### SD-Karte formatieren

```

1) confirm formatting of SD card.
2) All data will be erased.
3) cancel

```

Listing 11.5: Untermenü SD-Karte formatieren

Beim Formatieren werden alle Dateien auf der SD-Karte gelöscht, inklusive der Konfigurationsdatei mit allen Sensor-Einstellungen. Der Befehl 'format SD card' holt vor der Ausführung nochmals eine Bestätigung ein, ob sich der Benutzer wirklich sicher ist, dass er alle Dateien löschen will (Listing 11.5). Während dem Formatieren wird die Meldung 11.6 angezeigt.

```
1 formatting SD Card
```

Listing 11.6: Statusmeldung SD formatieren

Sind beim Formatieren Fehler aufgetreten, erhält man die Fehlermeldung 11.7. In diesem Fall sollte die Karte in einem Computer geprüft und formatiert werden.

```
1 Formatting SD card FAILED. Please use a Computer to format the card.
```

Listing 11.7: Fehlermeldung SD formatieren

Bei erfolgreicher Formatierung wird die Meldung 11.8 ausgegeben.

```

1 Formatting done
2 Returning to base menu.

```

Listing 11.8: Erfolgsmeldung SD formatieren

### SD-Karte anmelden

Damit Dateien auf die SD-Karte geschrieben werden können, muss sie vorher erkannt werden. Normalerweise geschieht dies, sobald die Karte eingesetzt wird. Wenn keine SD-Karte erkannt wird, wird dies im Basismenü angezeigt wie im Listing 11.9. Durch den Aufruf des Befehls 'mount SD card' im Basismenü (Listing 11.7.2) kann die eingesetzte SD-Karte angemeldet werden.

Nach erfolgreicher Anmeldung der SD-Karte wird das Basismenü angezeigt.

Wenn keine SD-Karte erkannt werden kann, wird die Fehlermeldung in Listing 11.9 ausgegeben.

```
1 No SD card detected! Please insert card and try again!
```

Listing 11.9: Fehlermeldung SD-Karte anmelden

### SD-Karte abmelden

Bevor die SD-Karte aus dem Datenlogger entfernt wird, sollte sie abgemeldet werden. Der Datenlogger schliesst bei diesem Vorgang alle geöffneten Dateien, um Datenverlust zu vermeiden. Da beim Abmelden der Karte die Aufzeichnung der Daten gestoppt wird, wird vorher eine Bestätigung verlangt (Listing 11.10).

```

1) unmount SD card
2) This will stop logging and close all data files.
3) cancel

```

Listing 11.10: Untermenü SD-Karte abmelden

Falls die Konfiguration der Sensoren verändert, aber noch nicht in die Konfigurationsdatei geschrieben wurde, wird eine Warnung angezeigt (Listing 11.11). Die Konfiguration bleibt im Speicher des Datenloggers erhalten, so lange die Spannungsversorgung angeschlossen ist und kann auch auf der neuen SD-Karte gespeichert werden.

```

1 ****
2 * WARNING: sensor configuration data has not been saved to file! *
3 * If you want to save config to file, cancel now. *
4 ****

```

Listing 11.11: Warnung vor SD-Karte abmelden bei ungespeicherter Konfiguration

### Logger-Status

Mit dem Befehl 'logger status' können einige Information über den Datenlogger angezeigt werden.

```

1 Time: Sat Dec 6 18:00:00 2014
2 Started?: 1

```

Listing 11.12: Untermenü Logger-Status

### Starten und stoppen der Aufzeichnung

Um die Datenspeicherung im Datenlogger zu unterbrechen oder wieder zu starten wird der Befehl 'start/stop logger' verwendet. Beim Aufruf des Befehls wird ein Untermenü gemäss Listing 11.13 oder Listing 11.14 angezeigt.

```

1 Logger is running.
2 1) stop the logging .
3 0) cancel

```

Listing 11.13: Untermenü Stoppen der Aufzeichnung

Da sich das Menü dem gegenwärtigen Zustand anpasst, sieht es bei gestoppter Aufzeichnung aus wie in Listing 11.14.

```

1 Logger is stopped.
2 1) start the logging .
3 0) cancel

```

Listing 11.14: Untermenü Starten der Aufzeichnung

Wenn die Aufzeichnung am Logger gestoppt wird, wird an alle Sensoreinheiten der Befehl zum Aufzeichnungsstopp gesendet. Die Einstellungen zum Detailmodus bleiben in der Sensoreinheit aber erhalten. Beim erneuten Starten der Aufzeichnung im Logger werden auch die Sensoreinheiten wieder gestartet. Es besteht auch die Möglichkeit, einzelne Sensoreinheiten zu stoppen (siehe Abschnitt 11.7.3). Eine gestoppte Sensoreinheit bleibt auch beim Starten der Aufzeichnung am Datenlogger gestoppt, da sie schon vor dem Stopp in diesem Zustand war.

### Sensor-Parameter

Die Parameter der Datenerfassung und Ereigniserkennung können für alle Sensoreinheiten gemeinsam oder für jede Sensoreinheit individuell eingestellt werden. Die Auswahl einer einzelnen oder aller Sensor-Einheiten erfolgt beim Einstieg in das Untermenü der Sensor-Parameter. Listing 11.15 zeigt die Auswahl der Sensoren. Die Auswahlliste enthält gleich die aktuellen Werte der Parameter, damit man eine Übersicht hat.

```

1  Nr      SID    serial          fs   threshold  baseline  timeout   detail
2  1)     2      461bfdf6    10000      200       2047      30        raw
3  2)     3      361509a5    10000      150       2000      20        peaks
4  ...
5  7)     8      1562a010    20000      400       2047      15        detailed
6
7  #) Select a sensor from the list.
8  99) Select all sensors.
9  0) cancel

```

Listing 11.15: Untermenü Sensor-Auswahl

Nach der Auswahl einer Sensoreinheit gelangt man zur Auswahl des anzupassenden Parameters, Listing 11.16. Ist ein einzelner Sensor ausgewählt, werden hier noch einmal die aktuellen Werte der Parameter angezeigt.

```

1  1) set sampling rate (current: 10000 Hz)
2  2) set threshold value (current: 200)
3  3) set baseline value (current: 2047)
4  4) set timeout (current: 30)
5  5) set detail level (current: peaks only)
6  6) start or stop recording (current: started)
7  0) exit

```

Listing 11.16: Untermenü Sensor-Parameter

**Abtastrate** Die Abtastrate legt fest, wie oft pro Sekunde ein Messwert vom Beschleunigungssensor eingelesen werden soll. Die Sensoreinheiten können in einem Bereich zwischen  $100\text{Hz}$  und  $200000\text{Hz}$  messen. Die Abtastrate kann in Schritten von  $100\text{Hz}$  eingestellt werden, Listing 11.17.

Die Abtastrate hat einen wesentlichen Einfluss auf die zu übertragende und zu speichernde Datenmenge, sowie die benötigte Rechenleistung. Davon wiederum hängt die Zeitspanne ab, wie lange die Messstation ohne Wartung betrieben werden kann. Es wird deshalb empfohlen, die Abtastrate nur so hoch wie nötig einzustellen. Wie hoch dieser Wert ist, hängt stark von den geplanten Auswertungen ab.

```

1  #) Enter sampling rate in Hz. (multiple of 100 Hz in range
   100..200'000 Hz)
2  0) cancel

```

Listing 11.17: Untermenü Abtastrate

Bei Eingabe einer ungültigen oder nicht unterstützten Abtastrate wird eine Fehlermeldung ähnlich Listing 11.18 angezeigt.

```

1  Sampling rate 220000 Hz not supported, too high.

```

Listing 11.18: Fehlermeldung bei ungültiger Abtastrate

Bei einer Änderung der Abtastrate wird automatisch der Timestamp zurückgesetzt, damit die Zuweisung der Messdaten zu einem Zeitpunkt weiterhin erfolgen kann.

**Ereigniserkennung** Die Ereigniserkennung hat drei Parameter, die die Form der gesuchten Ereignisse bestimmen. Abbildung 11.13 illustriert die Zusammenhänge zwischen Threshold und Nullpegel sowie die Funktionsweise des Timeouts.

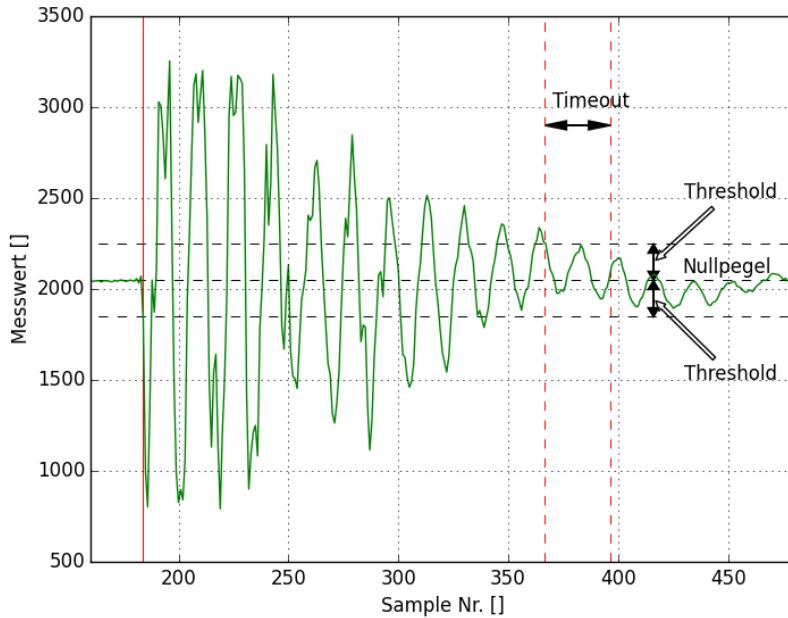


Abbildung 11.13.: Zusammenhänge der Parameter der Ereigniserkennung.

**Threshold** Der Threshold (Schwellenwert) ist ein Parameter der Ereigniserkennung. Er bestimmt, ab welcher Abweichung vom Nullwert ein Signal als Peak betrachtet werden soll. Bei der Wahl des Thresholds ist zu beachten, dass der Threshold auf beide Seiten des Nullwerts gilt. Daher darf die Summe des Nullpegels und des Thresholds nicht den maximalen Wert (4096) des A/D-Wandlers überschreiten. Ebenso muss der Wert des Thresholds kleiner sein als der Nullpegel, damit kein negativer Messwert anliegen müsste, um einen Peak zu erzeugen. Diese Einschränkungen werden bei der Eingabe noch einmal angezeigt, Listing 11.19.

```
1 #) Enter threshold value.  
2     baseline + threshold must not exceed 4096  
3     and  
4     baseline - threshold must not be below 0  
5 0) cancel
```

Listing 11.19: Untermenü Threshold

Bei Verletzung der Kriterien für den Threshold wird eine entsprechende Fehlermeldung angezeigt, Listing 11.20. Da die gleichen Kriterien auch bei der Einstellung des Nullpegels gelten, empfiehlt es sich, zuerst einen kleinen Wert für den Threshold zu wählen. Dann kann der Nullpegel ohne grosse Einschränkung eingestellt werden. Danach setzt man den passenden Threshold.

```
1 Invalid threshold value:  
2 threshold + baseline must not exceed 4096  
3 and  
4 threshold must be smaller than baseline value.
```

Listing 11.20: Fehlermeldung ungültiger Threshold

**Nullpegel** Der Nullpegel wird mit einer ähnlichen Maske (Listing 11.21) wie der Threshold eingestellt, auch die Einschränkungen für den Wertebereich sind die Gleichen.

```
1 #) Enter baseline value (default: 2047).
```

```
2 0) cancel
```

Listing 11.21: Untermenü Null-Level

Die Fehlermeldung bei ungültigen Werten für den Nullpegel ist in Listing 11.22 aufgeführt.

```
1 Invalid baseline value:  
2 threshold + baseline must not exceed 4096.  
3 and  
4 threshold - baseline must not be below 0 value
```

Listing 11.22: Fehlermeldung ungültiger Nullpegel

**Timeout** Der Timeout definiert, wie viele Samples der Signalwert unterhalb des Thresholds liegen kann, bevor das Ereignis als beendet betrachtet wird (Listing 11.23). Die Obergrenze für den Timeout ist 255.

```
1 #) Enter timeout in samples.  
2 0) cancel
```

Listing 11.23: Untermenü Timeout

Bei zu langem (Listing 11.24) oder sehr kurzem (Listing 11.25) Timeout wird eine Fehlermeldung resp. Warnung angezeigt.

```
1 Timeout too long, can not exceed 255.
```

Listing 11.24: Fehlermeldung zu langer Timeout

```
1 Timeout 0 will end impact after each peak.  
2 Timeout 0 in effect.
```

Listing 11.25: Warnung kurzer Timeout

**Detaillevel** Über die Wahl des Detaillevels wird bestimmt, wie viele und welche Daten von jedem Ereignis übertragen und gespeichert werden sollen (Listing 11.26). Die Detaillevel sind geordnet nach anfallender Datenmenge, beginnend mit dem grössten Aufwand. Die Detaillevel sind in Abschnitt 11.6, Seite 63 beschrieben.

```
1 1) raw (continuous data)  
2 2) detailed (start time, all samples of impact)  
3 3) peaks only (start time, nr of peaks, peaks)  
4 4) sparse (only start time, duration, nr of peaks, max peak)  
5 5) off  
6 0) cancel
```

Listing 11.26: Untermenü Detail-Level

**Start/Stop Sensor** Jede Sensoreinheit kann einzeln gestartet oder gestoppt werden, vorausgesetzt der Datenlogger ist gestartet. Im Untermenü ist ersichtlich, in welchem Zustand die ausgewählte Sensoreinheit gerade ist (listing 11.27). Falls die Konfigurationsänderung alle Sensoreinheiten betreffen soll, wird die Anzahl gestarteter und gestoppter Sensoren angezeigt (listing 11.28).

```
1 Selected sensor is currently stopped.  
2 1) start  
3 2) stop  
4 0) cancel
```

Listing 11.27: Untermenü Start/Stop einzeln

```

1 Started sensors: 3
2 Stopped sensors: 0
3 1) start
4 2) stop
5 0) cancel

```

Listing 11.28: Untermenü Start/Stop alle Sensoren

Wenn ein Sensor gestartet wird, muss für die anfallenden Daten eine Datei erzeugt werden. Schlägt dies fehl, wird dies mit der Fehlermeldung 11.29 angezeigt. Der Sensor wird dann nicht gestartet. Es wird empfohlen, in diesem Fall die SD-Karte zu überprüfen. Möglicherweise verfügt die SD-Karte nicht mehr über genügend Speicherplatz.

```

1 Could not create or open file. Please check SD card for free space.

```

Listing 11.29: Fehlermeldung beim Starten eines Sensors

### Sensor-Status

Um die Einstellungen aller Sensoren auf einen Blick zu überprüfen, kann mit dem Befehl 'sensor states' die Liste aller Einstellungen aufgerufen werden, Listing 11.30. In der Liste sind eine interne Nummer, die CAN-Bus-ID, die Seriennummer der Sensoreinheit sowie alle Einstellungen aufgeführt.

```

1 Listing sensor config:
2 Nr SID serial      fs threshold baseline timeout detail
3   started?
4 1) 2 461bfdf6 10000    200     2047    30 raw
5   started
6 2) 3 361509a5 10000    150     2000    20 peaks only
7   started
8 0) continue

```

Listing 11.30: Untermenü Sensor-Status

### Timestamp zurücksetzen

Der Timestamp kann manuell zurückgesetzt werden, dafür wird eine Bestätigung eingeholt: Listing 11.31. Das Zurücksetzen des Timestamps betrifft immer alle Sensoreinheiten.

```

1 1) re-synchronize timestamp
2 0) cancel

```

Listing 11.31: Untermenü Timestamp zurücksetzen

### Interne Uhr

Anhand der internen Uhr werden die Timestamps vom Datenlogger in einen realen Zeitpunkt umgerechnet. Deshalb wird empfohlen, die interne Uhr auf die korrekte Uhrzeit einzustellen. Dies erfolgt über den Befehl 'internal clock', Listing 11.32. Hier können das Datum (Untermenü 11.33) und die Tageszeit (Untermenü 11.34) eingestellt werden.

```

1) adjust date
2) adjust time
0) exit
4
5 current time: Sat Dec 6 18:00:00 2014

```

Listing 11.32: Untermenü interne Uhr

Das Einstellung des Datums erfolgt mit Inkrementen resp. Dekrementen von 365, 30, 10 Tagen oder 1 Tag.

```

adjust internal date
1) adjust date +365 days
2) adjust date -365 days
3) adjust date + 30 days
4) adjust date - 30 days
5) adjust date + 10 days
6) adjust date - 10 days
7) adjust date + 1 day
8) adjust date - 1 day
0) exit
11
12 current time: Sat Dec 6 18:00:00 2014

```

Listing 11.33: Untermenü Datum einstellen

Die Uhrzeit wird mittels Inkrementen resp. Dekrementen von 1 Stunde, 10 oder 1 Minute oder 1 Sekunde eingestellt. Die aktuelle Uhrzeit wird jeweils unterhalb des Menüs angezeigt.

```

adjust internal time
1) adjust time +1 hour
2) adjust time -1 hour
3) adjust time +10 minute
4) adjust time -10 minute
5) adjust time +1 minute
6) adjust time -1 minute
7) adjust time +1 second
8) adjust time -1 second
0) exit
11
12 current time: Sat Dec 6 18:00:00 2014

```

Listing 11.34: Untermenü Uhrzeit einstellen

## Konfigurations-Datei

Über den Befehl 'config file' (Listing 11.37) kann die aktuelle Konfiguration der Sensoreinheiten in die Konfigurationsdatei 'config.txt' abgespeichert werden, oder eine neue Konfiguration aus der Datei eingelesen und an die Sensoreinheiten gesendet werden. Es ist nicht möglich, aus mehreren Konfigurationsdateien auszuwählen.

```

1) read configuration from file and set up all sensors accordingly.
2) store current configuration in file. Old config file will be
   overwritten.
0) cancel

```

Listing 11.35: Untermenü Konfigurationsdatei

Falls die Konfigurationsdatei nicht gespeichert werden kann, wird die Fehlermeldung in Listing 11.36 angezeigt. Um die Konfigurationsdatei trotzdem speichern zu können, kann die SD-Karte abgemeldet und ausgetauscht werden. Die Konfiguration bleibt erhalten, so lange die Spannungsversorgung besteht.

Um einem möglichen Konfigurationsdatenverlust vorzubeugen, empfiehlt es sich, die Konfigurationsdatei zu speichern, sobald alle Einstellungen wie gewünscht gemacht wurden. Nach einem Unterbruch in der Spannungsversorgung wird automatisch die Konfiguration aus der Konfigurationsdatei gelesen und an die Sensoren gesendet.

```

1 The config file could not be written. Please check the SD card in a
   computer.
2 The configuration data will remain stored in the logger unless you
   turn off power.

```

Listing 11.36: Fehlermeldung beim Speichern der Konfigurationsdatei

#### 11.7.4. Konfigurationsdatei

In der Konfigurationsdatei werden die Einstellungen der Sensoreinheiten gespeichert. Die Datei kann auch auf einem Computer erstellt werden und über eine SD-Karte auf den Datenlogger übertragen werden. Ein Beispiel einer Konfigurationsdatei zeigt Listing 11.37. Die erste Zeile enthält die Anzahl Datensätze. Jeder Datensatz enthält die Konfiguration einer Sensoreinheit. Auf der zweiten und den folgenden Zeilen folgen die Datensätze.

Ein Datensatz enthält die CAN-Bus-ID, die Seriennummer der Sensoreinheit als Hexadezimalzahl, die Abtastrate (ohne die hintersten zwei Nullen), den Threshold, den Nullpegel und den Timeout sowie den Detail-Level und ob der Sensor Daten aufzeichnen soll (1) oder nicht (0).

Der Datenlogger versucht nach dem Einschalten der Spannungsversorgung die Konfigurationsdatei von der SD-Karte zu lesen. Falls keine SD-Karte vorhanden ist, startet der Datenlogger den Betrieb nicht, da keine Messdaten gespeichert werden können. Falls die SD-Karte vorhanden ist, aber keine Konfigurationsdatei 'config.txt' enthält, startet der Betrieb der Messstation im Standardmodus.

**Standardmodus** Im Standardmodus arbeiten alle Sensoreinheiten mit einer Abtastrate von  $10000\text{Hz}$ , Threshold 200, Nullpegel 2047, Timeout 30. Der Detail-Level wird auf 'peaks only' gesetzt, siehe Abschnitt 11.6, Seite 63. Alle Sensoreinheiten werden gestartet.

```

1 {3,
2 {2, 461bfdf6, 100, 200, 2047, 30, 3, 1},
3 {3, 15047b39, 100, 150, 2040, 30, 2, 1},
4 {4, b78d6dca, 100, 100, 2049, 30, 4, 1},
5 }

```

Listing 11.37: Beispiel einer Konfigurationsdatei

Falls Änderungen an der Konfiguration gemacht, aber nicht in die Konfigurationsdatei geschrieben wurden, erscheint im Basismenü die Information gemäss Listing 11.38.

```

1 Config has been modified but not saved to SD card.

```

Listing 11.38: Information bei ungespeicherter Konfiguration

## **12. Verzeichnisse**

# Literaturverzeichnis

- [1] D. Rickenmann, J. M. Turowski, B. Fritschi, A. Klaiber, A. Ludwig, Bedload transport measurements at the Erlenbach stream with geophones and automated basket samplers, *Earth Surf. Process. Landforms* **2012**, *37*, 1000–1011.
- [2] S. C. Reid, S. N. Lane, J. M. Berney, J. Holden, The timing and magnitude of coarse sediment transport events within an upland, temperate gravel-bed river, *Geomorphology* **2007**, *83*, 152–182.
- [3] C. R. Wyss, D. Rickenmann, B. Fritschi, J. M. Turowski, V. Weitbrecht, R. M. Boes, Measuring bedload transport rates by grain-size fraction using the swiss plate geophone signal at the erlenbach., *Submitted to Journal of Hydrologic Engineering* **2014**, *xx*, xxxx.
- [4] D. Rickenmann, J. M. Turowski, B. Fritschi, C. Wyss, J. Laronne, R. Barzilai, I. Reid, A. Kreisler, J. Aigner, H. Seitz, H. Habersack, Bedload transport measurements with impact plate geophones: comparison of sensor calibration in different gravel-bed streams, *Earth Surf. Process. Landforms* **2014**, *39*, 928–942.
- [5] J. M. Turowski, A. Badoux, D. Rickenmann, B. Fritschi, Erfassung des Sedimenttransports in Wildbächen und Gebirgsflüssen, Anwendungsmöglichkeiten von Geophonmessanlagen, *Wasser Energie Luft* **2008**, *100*(1), 69–74.
- [6] ARM Ltd., Cortex-M4 Processor, **o. J.**  
[Http://www.arm.com/products/processors/cortex-m/cortex-m4-processor.php](http://www.arm.com/products/processors/cortex-m/cortex-m4-processor.php), Stand: 7.12.2014.
- [7] NXP Semiconductors, LPC4088FBD208 Processor Datasheet, **o. J.** URL [http://www.nxp.com/products/microcontrollers/cortex\\_m/lpc4000/LPC4088FBD208.html](http://www.nxp.com/products/microcontrollers/cortex_m/lpc4000/LPC4088FBD208.html), Stand:7.12.2014.
- [8] NXP Semiconductors, NXP LPC4088 QuickStart Board, **o. J.** URL <http://www.nxp.com/demoboard/OM13063.html>,Stand:7.12.2014.
- [9] o. V., CAN vs. RS485, White paper, IXXAT Automation GmbH, **o. J.** URL [http://www.ixxat.com/download/artikel\\_20105\\_can-vs-rs485\\_e.pdf](http://www.ixxat.com/download/artikel_20105_can-vs-rs485_e.pdf),Stand:8.12.2014.
- [10] o. V., CAN Specification 2.0, **1991**.
- [11] T. Müller, Vorlesung Kommunikationstechnik 1, **2013**. Kap. 2, S. 14.
- [12] o. V., SD Specifications, Physical Layer Simplified Specification, Version 4.10, **o. J.**
- [13] o. V., Wikipedia: CompactFlash, **2014**. URL <http://en.wikipedia.org/wiki/CompactFlash>,Stand:8.12.2014.
- [14] o. V., Wikipedia: USB, **2014**. URL <http://en.wikipedia.org/wiki/USB>,Stand:8.12.2014.
- [15] o. V., Wikipedia: Hilbert-Transformation, **2014**. URL <https://de.wikipedia.org/wiki/Hilbert-Transformation>,Stand:6.12.2014.
- [16] S. Wyrsh, Vorlesung Digitale Signalverarbeitung 1, **2013**. Kap. 3, DFT und FFT, S. 48.
- [17] 'wdwd', FIR Hilbert Transform Filter, Licensed under the Creative Commons Attribution-Share Alike 3.0 Unported., **2013**. URL [https://commons.wikimedia.org/wiki/File:FIR\\_Hilbert\\_Transform\\_Filter.svg](https://commons.wikimedia.org/wiki/File:FIR_Hilbert_Transform_Filter.svg).
- [18] o. V., Wikipedia: Microelectromechanical systems, **2014**. URL [http://en.wikipedia.org/wiki/Microelectromechanical\\_systems](http://en.wikipedia.org/wiki/Microelectromechanical_systems),Stand:8.12.2014.

# Abbildungsverzeichnis

3.1.	Eine Messstation mit einem Datenlogger, der mehrere Sensoreinheiten im Bach steuert . . . . .	10
6.1.	Übersicht des Testaufbaus. . . . .	17
6.2.	Montage des Sensors im Testaufbau. . . . .	17
7.1.	Hardwarekonzept des Datenloggers. . . . .	23
7.2.	Hardwarekonzept der Sensoreinheit. . . . .	23
7.3.	Chipdiagramm der ARM Cortex-M4 Architektur [6]. . . . .	25
7.4.	Differentielle Leitung [11]. . . . .	28
7.5.	Schematischer Hardware-Aufbau des Datenloggers. . . . .	31
7.6.	Schematischer Hardware-Aufbau der Sensoreinheit. . . . .	32
8.1.	Softwarestack des Datenloggers. . . . .	34
8.2.	Softwarestack der Sensoreinheit. . . . .	34
8.3.	Kommunikationsdiagramm der Threads in der Sensoreinheit. . . . .	35
8.4.	Beispiel von Messdaten mit einer Hüllkurve (envelope). . . . .	37
8.5.	Hilbert-Transformation als FIR-Filter [17]. . . . .	38
8.6.	Zustandsmaschine der Ereigniserkennung. . . . .	38
8.7.	Parameter der Ereigniserkennung. . . . .	39
8.8.	Detail-Level 'raw'. . . . .	40
8.9.	Detail-Level 'detailed'. . . . .	40
8.10.	Detail-Level 'peaks only'. . . . .	41
8.11.	Detail-Level 'sparse'. . . . .	41
8.12.	Sequenzdiagramm des Startupvorgangs der Messstation. . . . .	42
8.13.	Sequenzdiagramm des Startupvorgangs der Messstation. . . . .	43
8.14.	Auflösung einer Kollision bei CAN-Bus. . . . .	45
8.15.	Aufbau eines CAN-Frames [? ]. . . . .	46
8.16.	Header verschiedener Message-Typen. . . . .	47
8.17.	Schema des CAN-Controllers. . . . .	49
8.18.	CAN-Filter. . . . .	50
9.1.	Vergleichsmessung mit Oszilloskop und Sensoreinheit. . . . .	53
11.1.	Schematischer Aufbau einer Messanlage. . . . .	58

11.2.	Der Datenlogger in geöffnetem Zustand. . . . .	59
11.3.	Die Sensoreinheit in geöffnetem Zustand. . . . .	60
11.4.	Anbringung des Abschlusswiderstands. . . . .	62
11.5.	Polnummerierung in der Buchse. . . . .	62
11.6.	Polnummerierung im Stecker. . . . .	62
11.7.	Anschluss der Leitungen an den Schraubklemmen. . . . .	63
11.8.	Beispiel von Rohdaten. . . . .	64
11.9.	Detail-Level 'raw'. . . . .	64
11.10.	Detail-Level 'detailed'. . . . .	65
11.11.	Detail-Level 'peaks'. . . . .	66
11.12.	Detail-Level 'sparse'. . . . .	67
11.13.	Zusammenhänge der Parameter der Ereigniserkennung. . . . .	73
A.1.	Gantt-Chart des Projekts. . . . .	IV
A.2.	Hauptkomponenten des NXP LPC4088 QuickStart Boards von Embedded Artists. . . . .	XIII
A.3.	Pins des NXP LPC4088 QuickStart Boards von Embedded Artists. . . . .	XIII

# Tabellenverzeichnis

7.1.	Fähigkeiten des NXP LPC4088 Prozessors [7]. . . . .	26
7.2.	Zusätzliche Fähigkeiten des NXP LPC4088 QuickStart Boards von <i>Embedded Artists</i> [8]. . . . .	26
7.3.	Entscheidungsmatrix für die Auswahl des Bussystems. . . . .	27
7.4.	Entscheidungsmatrix zur Auswahl des Speichermediums [12–14]. . . . .	30
9.1.	Vergleich des Datenaufkommens verschiedener Detail-Levels. . . . .	54
11.1.	Steckerbestückung des Bussystems. . . . .	62

# Listings

6.1.	T400 Vorbedingung und Auswahl aller Sensoren . . . . .	19
6.2.	T400 Auswahl des Parameters . . . . .	19
6.3.	T400 Eingabe des neuen Thresholds . . . . .	19
6.4.	T400 Sendeprotokoll am Datenlogger . . . . .	19
6.5.	T400 Konfiguration aus Sensor 3 . . . . .	20
6.6.	T400 Konfiguration aus Sensor 2 . . . . .	20
8.1.	Timer mit Aufruf der A/D-Wandler-Funktion (ADC_4088.cpp) . . . . .	35
8.2.	ISR zur Abhandlung des ADC-Interrupt Requests (impact_event.cpp) . . . . .	36
11.1.	Beispieldaten auf Detail-Level 'raw' . . . . .	65
11.2.	Beispieldaten auf Detail-Level 'detailed' . . . . .	66
11.3.	Beispieldaten auf Detail-Level 'peaks only' . . . . .	66
11.4.	Beispieldaten auf Detail-Level 'sparse' . . . . .	67
11.5.	Untermenü SD-Karte formatieren . . . . .	70
11.6.	Statusmeldung SD formatieren . . . . .	70
11.7.	Fehlermeldung SD formatieren . . . . .	70
11.8.	Erfolgsmeldung SD formatieren . . . . .	70
11.9.	Fehlermeldung SD-Karte anmelden . . . . .	70
11.10.	Untermenü SD-Karte abmelden . . . . .	70
11.11.	Warnung vor SD-Karte abmelden bei ungespeicherter Konfiguration . . . . .	71
11.12.	Untermenü Logger-Status . . . . .	71
11.13.	Untermenü Stoppen der Aufzeichnung . . . . .	71
11.14.	Untermenü Starten der Aufzeichnung . . . . .	71
11.15.	Untermenü Sensor-Auswahl . . . . .	72
11.16.	Untermenü Sensor-Parameter . . . . .	72
11.17.	Untermenü Abtastrate . . . . .	72
11.18.	Fehlermeldung bei ungültiger Abtastrate . . . . .	72
11.19.	Untermenü Threshold . . . . .	73
11.20.	Fehlermeldung ungültiger Threshold . . . . .	73
11.21.	Untermenü Null-Level . . . . .	73
11.22.	Fehlermeldung ungültiger Nullpegel . . . . .	74
11.23.	Untermenü Timeout . . . . .	74
11.24.	Fehlermeldung zu langer Timeout . . . . .	74
11.25.	Warnung kurzer Timeout . . . . .	74
11.26.	Untermenü Detail-Level . . . . .	74
11.27.	Untermenü Start/Stop einzeln . . . . .	74
11.28.	Untermenü Start/Stop alle Sensoren . . . . .	75
11.29.	Fehlermeldung beim Starten eines Sensors . . . . .	75
11.30.	Untermenü Sensor-Status . . . . .	75
11.31.	Untermenü Timestamp zurücksetzen . . . . .	75
11.32.	Untermenü interne Uhr . . . . .	76
11.33.	Untermenü Datum einstellen . . . . .	76
11.34.	Untermenü Uhrzeit einstellen . . . . .	76
11.35.	Untermenü Konfigurationsdatei . . . . .	76
11.36.	Fehlermeldung beim Speichern der Konfigurationsdatei . . . . .	77
11.37.	Beispiel einer Konfigurationsdatei . . . . .	77
11.38.	Information bei ungespeicherter Konfiguration . . . . .	77

# Glossar

**A/D-Wandler** Analog/Digital-Wandler, (Englisch: ADC). Ein A/D-Wandler misst die Spannung, die an einem Pin anliegt und gibt einen digitalen Wert aus, der die Höhe der Spannung angibt. Bei der Umwandlung in einen digitalen Wert erfolgt eine Quantisierung. Je grösser die Bit-Breite des A/D-Wandlers ist, umso kleiner wird die Schrittgrösse von einem digitalen Wert zum nächst höheren Wert. 21, 24, 25, 30, 32, 41, 43, 46, 58, *siehe* Quantisierung & Bit-Breite

**Abschlusswiderstand** Bei elektrischen Signalübertragungen treten am Ende der Leitung Signalreflexionen auf. Die Reflexionen pflanzen sich in der Leitung in entgegengesetzter Richtung fort und stören so die Übertragung. Um die Reflexionen zu verhindern, können Widerstände eingesetzt werden, die die elektrische Energie in Wärme umwandeln. Dadurch treten keine Reflexionen mehr auf. 48, 66

**Abtastrate** Definiert, in welchen zeitlichen Abständen ein Messwert erfasst werden soll. Üblicherweise wird dieser Wert in  $Hz$  oder  $s^{-1}$  angegeben. 15, 32, 34, 37, 38, 41, 43, 51, 62, 63

**Betriebsmodus** Ein Modus bestimmt die Verhaltensweise eines Systems. Je nach gewähltem Modus können mit den gleichen Eingaben und Befehlen andere Aktionen ausgeführt und andere Resultate ausgegeben werden. 12

**Bit-Breite** Die Bit-Breite gibt an, wie viele Bit für die Darstellung eines Wertes verwendet werden. Je grösser die Bit-Breite, desto mehr unterschiedliche Werte können dargestellt werden. Mit einer Bit-Breite von  $n$  können  $2^n$  Werte dargestellt werden. 21, 24, 41

**Blockgrösse** Anzahl Werte, die in einer DFT oder IDFT verrechnet wird. 34

**Busmaster** Ein Gerät, das die Kontrolle über ein Bussystem hat. Der Busmaster kann den anderen Busteilnehmern (Slaves) eine Genehmigung erteilen, Daten über das Bussystem zu übertragen. Der Busmaster hat aber jederzeit die Möglichkeit, einen Slave in der Übertragung zu unterbrechen. *siehe* Slave

**Bussystem** Ein elektrisches System für die Kommunikation zwischen mehreren Geräten. Ein Bussystem besteht aus Datenleitungen, über welche Signale gesendet werden, und aus Schnittstellen, an denen die Busteilnehmer angeschlossen werden. Die Besonderheit liegt darin, dass über ein Leitungssystem mehr als zwei Geräte miteinander kommunizieren können. Mittels eines Adressierungsschemas können die Empfänger ausgewählt werden. 8, 9, 11, 13–15, 21, 23–27, 43, 45, 46, 48, 67, *siehe* Signal

**Collision Detection** Kollisionserkennung. Als Kollision bezeichnet man den gleichzeitigen Versuch mehrerer Busteilnehmer, eine Nachricht zu übermitteln. Dies führt zu einer Überlagerung der Nachrichten und macht diese unlesbar. Gleichzeitig mit dem Schreiben liest der Transceiver die Signale vom Bus. Stimmen die gelesenen Signale nicht mit den Geschriebenen überein, bedeutet dies, dass ein anderer Teilnehmer ebenfalls auf den Bus schreibt und eine Kollision vorliegt. Das Verhalten bei einer Kollision ist vom Bussystem abhängig. 27, *siehe* Transceiver

**Computer** Englisch für Elektronenrechner. Unter einem Computer versteht man umgangssprachlich einen Personal Computer (PC) oder einen tragbaren Computer (Laptop). Heute sind Computer so leistungsfähig und so stark miniaturisiert, dass sie mühelos in einer Aktentasche Platz finden. Weniger leistungsfähig, dafür noch kleiner sind Embedded Systems, eingebettete Systeme, die von Laien nicht als Computer erkannt werden. 12, 18, 30, 45, 46, 53–55, 62, *siehe* PC & Embedded System

**CRC** Cyclic Redundancy Check. Ein Codeverfahren, das eine Prüfsumme über eine Nachricht berechnet. Durch Nachrechnen der Prüfsumme im Empfänger kann die Nachricht auf Fehlerfreiheit überprüft werden. 28

**Daisy Chain** Englisch für Gänseblümchenkette. Gemeint ist das Aneinanderreihen mehrerer Geräte. Im Unterschied zu einem Bussystem müssen die Geräte alle Nachrichten, die für andere Empfänger bestimmt sind, aktiv weiterleiten. Dies braucht Rechenleistung in den Geräten. Wenn ein Gerät ausfällt, gehen alle Nachrichten in diesem Gerät verloren. 25, 26

**Datenlogger** Ein Gerät zur Sammlung und Speicherung von Messdaten von mehreren Sensoreinheiten. 7, 9–15, 17–19, 21–23, 28–30, 33, 37–39, 43, 45, 46, 52–54, 56, 57, 60–62, 66, *siehe* Sensoreinheit

**DSP** Digitaler Signal-Prozessor. Ein Mikroprozessor oder ein Bestandteil eines solchen, der dank spezieller Hardware fähig ist, Multiplikationen und Additionen extrem schnell auszuführen. Außerdem verfügt ein DSP über Schieberegister, um Rechenoperationen über eine Serie der aktuellsten Messwerte auszuführen. Da bei der digitalen Signalverarbeitung oft für jeden Messwert viele Multiplikations- und Additions-Schritte ausgeführt werden müssen, ist es wichtig, dass Multiplikations-Operationen in einem Taktzyklus ausgeführt werden können. In einem normalen Prozessor werden für eine Multiplikation mehrere Taktzyklen benötigt. 21, 23, *siehe* Mikrocontroller

**Embedded System** Deutsch: eingebettetes System. Ein Computer, der nicht über die üblichen Ein- und Ausgabemöglichkeiten eines Computers wie Bildschirm und Tastatur verfügt. Oft werden deshalb Embedded Systems nicht als Computer wahrgenommen. Sie sind im Gegensatz zu PCs, die als Alleskönnner konzipiert sind, auf eine bestimmte Aufgabe zugeschnitten und deshalb nur mit den nötigen Bedien-Elementen versehen. Oft genügen für die Aufgaben weniger leistungsfähige Prozessoren als in einem PC, sogenannte Mikroprozessoren. *siehe* Computer & Mikrocontroller

**Ereignis** Eine Abfolge von Messwerten, die einer vordefinierten Form entspricht. Es kann zum Beispiel ein Schwellenwert (engl. threshold) definiert sein. Das Überschreiten dieses Wertes kann dann den Beginn, das Unterschreiten des Schwellenwertes das Ende eines Ereignisses markieren. 9–15, 23, 28, 34, 35, 38, 49, 52, 58–60, 69

**Ereigniserkennung** Auswertung von Messdaten um definierte Signalformen (Ereignisse) zu erkennen. 34, 35, 49, 51, 58, 59, 66, *siehe* Ereignis & Signal

**FIFO-Queue** First In First Out Queue. Bezeichnung für eine Warteschlange, bei der der älteste Eintrag als nächstes herauskommt. Im Gegensatz zu LIFO (Last In First Out), wo jeweils der jüngste Eintrag zuerst ausgelesen wird. 34, 37

**Finite State Machine** Englisch für Zustandsmaschine. Eine Finite State Machine definiert eine endliche Anzahl Zustände, die die Maschine einnehmen kann. Die FSM reagiert auf Ereignisse, indem sie Aktionen auslöst und allenfalls in einen anderen Zustand wechselt. Für jeden Zustand ist definiert, welche möglichen Ereignisse welche Aktionen auslösen, und in welchen Folgezustand gewechselt werden soll. 35

**FIR-Filter** Finite Impulse Response, Englisch für Filter mit endlicher Impuls-Antwort. Ein FIR-Filter hat immer eine endliche Impulsantwort, d.h. auf einen kurzen Impuls am Eingang des Filters folgt am Ausgang eine Antwort des Filters, die garantiert endet. Die Ordnung des FIR-Filters gibt an, wie lange die Antwort dauern wird. 35, 66

**Flash-Speicher** Nicht-flüchtiger Speicher. Der Inhalt des Speichers bleibt auch bestehen, wenn keine Versorgungsspannung anliegt. 24, 30

**FPU** Floating Point Unit, ein Rechenwerk in der CPU, die Berechnungen mit Dezimalbrüchen sehr schnell ausführen kann. 23

**Geophon** Ein Messgerät für Vibrationen des Bodens. Ein Geophon misst Bewegungen mittels einer magnetischen Masse, die beweglich in einer Spule aufgehängt ist. Wird das Geophon in Bewegung versetzt, schwingt die magnetische Masse aufgrund ihrer Trägheit und induziert dadurch einen Strom in der Spule. Durch Messung dieses Stroms kann die Bewegung registriert werden. 7, 16, 45

**Geschiebkorn** Ein Stein oder Kiesel, der vom Fluss transportiert wird. Die Grösse der Steine wird als Korngrösse bezeichnet. 38

**Hardware** Die Hardware ist das eigentliche Rechenwerk eines Computers, worauf die Software ausgeführt wird. Dazu gehören alle elektrischen, elektronischen und mechanischen Bauteile eines Computers. Die Central Processing Unit (CPU) eines Computers könnte mit dem Hirn verglichen werden, hier laufen fast sämtliche Informationen und Instruktionen zusammen. 21

**Hilbert-Transformation** Mathematische Umrechnung einer Wertfolge, um die umhüllende Kurve zu erhalten. 34, 35, 66

**Hydrologie** Wissenschaft über das Wasser. 7

**Interrupt** Englisch für Unterbrechen. Durch ein spezielles Signal wird dem Prozessor mitgeteilt, dass ein Ereignis eingetreten ist. Der Prozessor unterbricht die laufende Funktion und ruft eine spezielle Routine, die ISR auf. Die ISR verarbeitet das Ereignis und gibt danach die Kontrolle an die vorher unterbrochene Funktion zurück. Mit einem Interrupt wird zwar die Abarbeitung einer Funktion kurzzeitig unterbrochen, dafür verliert der Prozessor keine Rechenzeit mit Polling. *siehe* Polling

**Kommandozeile** Eine Eingabeaufforderung auf dem Bildschirm, wo der Benutzer über eine Tastatur Befehle eingeben kann, die vom Computer interpretiert und ausgeführt werden. 12, *siehe* Computer

**MAC** Multiply-ACcumulate unit: Ein Rechenwerk in der CPU, wo Multiplikationen und Additionen ausgeführt werden. 23

**MEMS** Microelectromechanical System, Englisch für Mikroelektromechanisches System. Bezeichnung für Elektromechanische Systeme, die sehr stark miniaturisiert worden sind. MEMS-Geräte verwenden Bauteile, die nur wenige Mikrometer bis einen Millimeter gross [18]. 8, 44

**Mikrokontroller** Englisch Microcontroller (MC). Ein Prozessor mit weniger universellen Fähigkeiten als eine CPU, mit entsprechend geringerem Stromverbrauch. Ein Mikrokontroller verfügt meistens über spezielle Ein- und Ausgänge (Pins), über die zum Beispiel die anliegende Spannung gemessen oder eine bestimmte Spannung ausgegeben werden kann. Durch die kleinere Bauform und die geringere Leistungsaufnahme eignen sich diese Prozessoren besonders für den Einsatz in Embedded Systems, die oft längere Zeit unabhängig vom Stromnetz funktionieren müssen. 21, 25, 29, 30, 32, 34, 37, 41, 44, *siehe* Pin

**Nested Vectored Interrupt Controller** Wird benötigt, um möglichst rasch auf mehrere asynchron eintreffende Ereignisse reagieren zu können. Ein Prozessor mit NVIC kann auf verschiedene Ereignisse reagieren, indem Interrupts ausgelöst werden. Jedem Interrupt kann eine Priorität zugewiesen werden, um festzulegen, ob ein Interrupt einen anderen unterbrechen darf, der gerade vom Prozessor abgearbeitet wird. 23, *siehe* Interrupt

**Nullpegel** Signalpegel, wenn keine Beschleunigung gemessen wird. 35, 58, 59, 62, 63

**PC** PC oder umgangssprachlich Computer. Eine elektronische Rechenmaschine mit der sehr viele Rechenschritte in sehr kurzer Zeit ausgeführt werden können. Mit der richtigen Programmierung (Software) können PCs sehr unterschiedliche, umfangreiche Aufgaben lösen. *siehe* Software

**Peak** Englisch für Spitzenwert oder Signalspitze. 13, 37, 38, 42

**Pin** Ein Anschluss an einem Chip oder einer Leiterplatte. Über Pins werden elektronische Bauteile miteinander verbunden und Peripheriegeräte wie z.B. Sensoren angeschlossen. 24, 32, *siehe Sensor*

**Polling** Englisch für Abfragen. Bezeichnet das wiederholte Abfragen einer Funktion oder eines Wertes, bis ein bestimmtes Ereignis auftritt. Der Prozessor ist mit der Abfrage beschäftigt und verliert dadurch Rechenzeit. 26

**Quantisierung** Einteilung eines Werts aus einer kontinuierlichen Skala in eine abgestufte Skala. Bei der Quantisierung wird der nächstgelegene Wert auf der abgestuften Skala ausgewählt. Je nach Anwendung wird die abgestufte Skala mit konstanter Stufengröße (linear) gewählt, oder mit unterschiedlichen Stufengrößen je nach absolutem Wert (z.B. exponentiell). Der Quantisierungsfehler entspricht der Differenz zwischen dem analogen und dem quantisierten, digitalen Wert. Je grösser die Bit-Breite der Quantisierung ist, desto kleiner wird der Quantisierungsfehler. *siehe Bit-Breite*

**Sample** Englisch für Messwert, Messpunkt. 37, 38, 51

**SDRAM** Synchronous Dynamic RAM. Flüchtiger Arbeitsspeicher. Der Inhalt des Arbeitsspeichers muss regelmässig aufgefrischt werden, sonst geht die Information verloren. 24, 30, *siehe*

**Sensor** Ein Messgerät für physikalische Größen wie Temperatur, Feuchtigkeit, Luftdruck oder Beschleunigung. 7–10, 12–15, 17, 19, 21, 25, 28, 30, 32, 41, 42, 45

**Sensoreinheit** Ein kombiniertes elektronisches Gerät zur Messung von physikalischen Daten und der Verarbeitung dieser Daten. Das Gerät verfügt über einen Mikroprozessor und einen Sensor. Optional kann auch eine Schnittstelle für die Kommunikation mit einem Datenlogger vorhanden sein. 11–19, 21–25, 29–31, 33, 37–39, 41, 43–47, 49, 52, 54, 57, 58, 60–63, 66, *siehe Mikrokontroller, Sensor & Datenlogger*

**serielle Schnittstelle** Eine Schnittstelle, über die Daten bitweise übertragen werden, im Gegensatz zu paralleler Übertragung, wo auf mehreren Leitungen mehrere Bits gleichzeitig übertragen werden. 72

**Signal** Ein Signal ist ein Informationsträger. Die Information wird einem Signalwert zugeordnet. Ein einfaches Beispiel ist die Spannung am Ausgang eines Beschleunigungs-Sensors. Der Sensor gibt über die Höhe der Spannung an, wie stark die Beschleunigung von einem definierten Referenzwert abweicht. Die Spannung trägt also eine Information über den Messwert und ist daher ein Signal. Aus dem Datenblatt kann herausgelesen werden, wie hoch die ausgegebene Spannung bei einer bestimmten Beschleunigung ist. So kann das Signal in Information umgewandelt werden. 34, *siehe Sensor*

**Slave** Ein Busteilnehmer, der nur Daten über den Bus übertragen darf, wenn ihm der Busmaster die Genehmigung dafür erteilt. Dies kann z.B. in Form eines sog. Tokens geschehen. *siehe Busmaster*

**Software** Programm zur Steuerung der Rechenabläufe in einem Computer. Durch die Software wird einem Computer die Fähigkeit gegeben, bestimmte Aufgaben zu lösen und Resultate in einer für Menschen lesbaren Form darzustellen. Die Software enthält Instruktionen, die in der CPU des Computers ausgeführt werden. 46

**Speichermedium** Ein Stück Hardware, auf dem Daten gespeichert werden können. 21

**Stichleitung** Abzweigungsleitung an einer Busleitung. Um das Signal vom Bus zum Transceiver zu führen, wird eine Stichleitung benötigt, die wie eine T-Kreuzung von der Busleitung wegführt. Je länger die Stichleitung ist, desto stärker leider die Signalqualität auf der Busleitung. 47, *siehe Transceiver*

**Terminal-Emulator** Ein Programm, das ein Terminal emuliert. Der Benutzer sieht ein Fenster mit einer Kommandozeile zur Befehlseingabe, das aussieht wie auf einem Terminal. 18, 30, 39, 53, 54, *siehe Terminal*

**Threshold** Englisch für Schwellenwert. 13, 34, 35, 37, 49, 58, 59, 62, 63

**Timeout** Englisch für Zeitüberschreitung. 13, 37, 49, 58, 59, 62, 63

**Timestamp** Englisch für Zeitmarke. Mittels eines Timestamps kann ein Ereignis oder ein Messwert einem genauen Zeitpunkt zugeordnet werden. Der Timestamp wird dafür zu einem bestimmten Zeitpunkt auf null gesetzt (Reset) und in allen Messgeräten in vordefiniertem Takt erhöht. Der Timestamp gibt an, wie viel Zeit seit dem Reset vergangen ist. Durch die Wahl des Takts wird die zeitliche Auflösung definiert. 13–15, 18, 34, 37, 38, 43, 51, 52

**Transceiver** Ein Gerät das gleichzeitig Transmitter und Receiver, also Sender und Empfänger ist. Der Prozessor sendet die zu übertragenden Daten über eine serielle Schnittstelle an den Transceiver. Der Transceiver erzeugt dann die notwendigen elektrischen Signale auf dem Bus, um die Daten zu übertragen. Beim Empfangen einer Meldung liest der Transceiver die Signale vom Bus und übersetzt sie für die serielle Übertragung zum Prozessor. Ein Transceiver ist notwendig, wenn der Prozessor die vom Bus erwarteten elektrischen Signale nicht selbst erzeugen kann. 27, 47, *siehe* serielle Schnittstelle

# Abkürzungen

**ADC** Analog Digital Converter 21, 68

**CD** Collision Detection 26

**CD** Compact Disc VII, XIII

**CRC** Cyclic Redundancy Check 28

**DFT** diskrete Fourier-Transformation 34, 35, 68

**DSP** Digitaler Signal-Prozessor 23, 69

**FPU** Floating Point Unit 23

**FSM** Finite State Machine 35, 37, 69

**ID** Identifikationsnummer 12, 14

**IDFT** inverse diskrete Fourier-Transformation 34, 35, 68

**IRQ** Interrupt Request 32, 34

**ISR** Interrupt Service Routine 34, 70

**MAC** multiply-accumulate unit 23

**MC** Microkontroller 26

**MCI** Memory Card Interface 30

**NVIC** Nested Vectored Interrupt Controller 21, 23

**PC** Personal Computer 14, 70

**USB** Universal Serial Bus 29, 30, 39, 46, 53

**VAW** Versuchsanlage für Wasserbau 43, 44

**WSL** Eidg. Forschungsanstalt für Wald, Schnee und Landschaft 7, 8, 34, 42

## **A. Anhang**

---

Anhang

### **A.1. Offizielle Aufgabenstellung**

## **Titel: Messstation zur Registrierung von Geschiebe-Bewegungen im Fluss**

**Betreuer:** Hans-Joachim Gelke, gelk

**Fachgebiet:** Mikroelektronik (ME)

**Studiengang:** IT

**Zuordnung:** Institute of Embedded Systems (InES)

**Industriepartner:**

Eidg. Forschungsanstalt für Wald, Schnee und Landschaft WSL,

8903 Birmensdorf (<http://www.wsl.ch/>)

Bruno Fritschi, [bruno.fritschi@wsl.ch](mailto:bruno.fritschi@wsl.ch) (Projektidee)

Dieter Rickenmann, [dieter.rickenmann@wsl.ch](mailto:dieter.rickenmann@wsl.ch) (Betreuung)

**Gruppengrösse:** 2,

**Reserviert für:** Tobias Welti (weltitob), Tobias Keller (kellet01)

### **Ausgangslage:**

Das WSL betreibt eine Messstation zur Registrierung von Geschiebe-Bewegungen im Fluss mittels Geophonen, die unter Stahlplatten montiert sind. Diese Platten sind in einer Betonkonstruktion eingelassen, um sie im Flussbett zu fixieren. Die Geophone sind über Kabel mit einem Auswertungs-Rechner (Embedded PC) verbunden, der die Signale auswertet. Die baulichen Massnahmen für die Installation der Sensoren, der Auswertungsstation sowie der Stromversorgung sind sehr teuer. Zukünftig sollen die Geophone durch eindimensionale MEMS Beschleunigungssensoren ersetzt werden, da diese kleiner sind.

### **Aufgabenstellung:**

Um die Kosten zu senken und die zu übertragende Datenmenge zu reduzieren, soll die Auswertung der Daten direkt am Sensor erfolgen. Somit könnten die Daten über ein Bussystem übertragen werden und der Auswertungsrechner bräuchte weniger Leistung.

Dank der Bustopologie ist das Messsystem weniger komplex und kann einfacher installiert werden. Denkbar wäre die Integration in einer Gummimatte anstelle der Stahl- und Betonkonstruktion, da viel weniger Leitungen nötig sind.

Ziel der Arbeit ist die Entwicklung der Auswertungshardware und des Bussystems. Die Auswertungsalgorithmen sind nicht Bestandteil der Arbeit und werden vom WSL zur Verfügung gestellt.

Denkbar wäre es, einen Prototyp für Vergleichsmessungen im Erlenbach (Alptal, SZ) an einer bestehenden Schwelle zu implementieren.

## A.2. Projektmanagement

### A.2.1. Zeitplanung und Meilensteine

Abbildung A.1 zeigt den Gantt-Chart des Projekts. Die Konzeptphase wurde gemeinsam bearbeitet. Danach wurden die Arbeiten aufgeteilt, T. Keller entwickelte das CAN-Protokoll und die zugehörigen Inter Process Communication (IPC)-Abläufe. Die Konfiguration und Bedienung der Messstation wurde von T. Welti entwickelt. Die Aufteilung aller Arbeitspakete ist im Gantt-Chart auf Seite IV aufgeführt.

#### Meilenstein 1: Hardware in Betrieb

Ein erstes Konzept wurde pünktlich zum Meilenstein 1 erstellt. Bezuglich der Art, wie die Dokumentation (vor allem welche Diagramme am besten geeignet wären) im Detail auszusehen hat, bestanden noch Unsicherheiten. Soweit sinnvoll soll UML verwendet werden, damit die Diagramme verständlich sind.

#### Meilenstein 2: Gehäuse und Leiterplatte

Die Leiterplatte wurde in unserem Auftrag und unter Mithilfe von T. Welti durch Valentin Schlatter (ZHAW InES) entwickelt. Beim Entwurf und dem Bau der Gehäuse stand Erich Ruff (ZHAW InES) als Berater zur Seite und übernahm die Bohrarbeiten. Dank ihrer Mithilfe konnte der Meilenstein 2 termingerecht abgeschlossen werden.

#### Meilenstein 3: Software

Der dritte Meilenstein konnte nicht eingehalten werden. Die ersten proof-of-concept-Implementationen sahen zwar vielversprechend aus, allerdings tauchten bei der Detailimplementation des CAN-Protokolls Schwierigkeiten auf, die einige Umstellungen notwendig machten. So konnte der Meilenstein erst mit drei Wochen Verspätung erreicht werden.

#### Meilenstein 4: Testing

Aufgrund der Verzögerung des zweiten Meilensteins konnte auch dieser Meilenstein nicht fristgerecht erreicht werden. Zudem ist das Testing rudimentär ausgefallen, da nach der Verzögerung kaum mehr Zeit für intensive Tests blieb. Wir waren aus terminlichen Gründen gezwungen, die Arbeit auf den MS 5 zu fokussieren.

#### Meilenstein 5: Dokumentation

Der letzte Meilenstein konnte fristgerecht erreicht werden.

### A.2.2. Besprechungsprotokolle

#### Projektskizzierung an der WSL, Birmensdorf

Die erste Sitzung zum Projekt fand am 16. Juli 2014 an der WSL in Birmensdorf statt. Anwesende waren Bruno Fritschi (Projektidee), Carlos Wyss (Doktorand), Alexandre Badoux (Gruppenleiter) sowie Tobias Keller und Tobias Welti.

Die Projektidee wurde von Bruno Fritschi noch einmal erklärt. Carlos Wyss zeigte einige Beispiele von bestehenden Messresultaten und erläuterte den Algorithmus der Ereigniserkennung, die mittels Hilbert-Transformation gelöst wird. Die Datenreduktion erfolgt durch Berechnung eines Histogramms

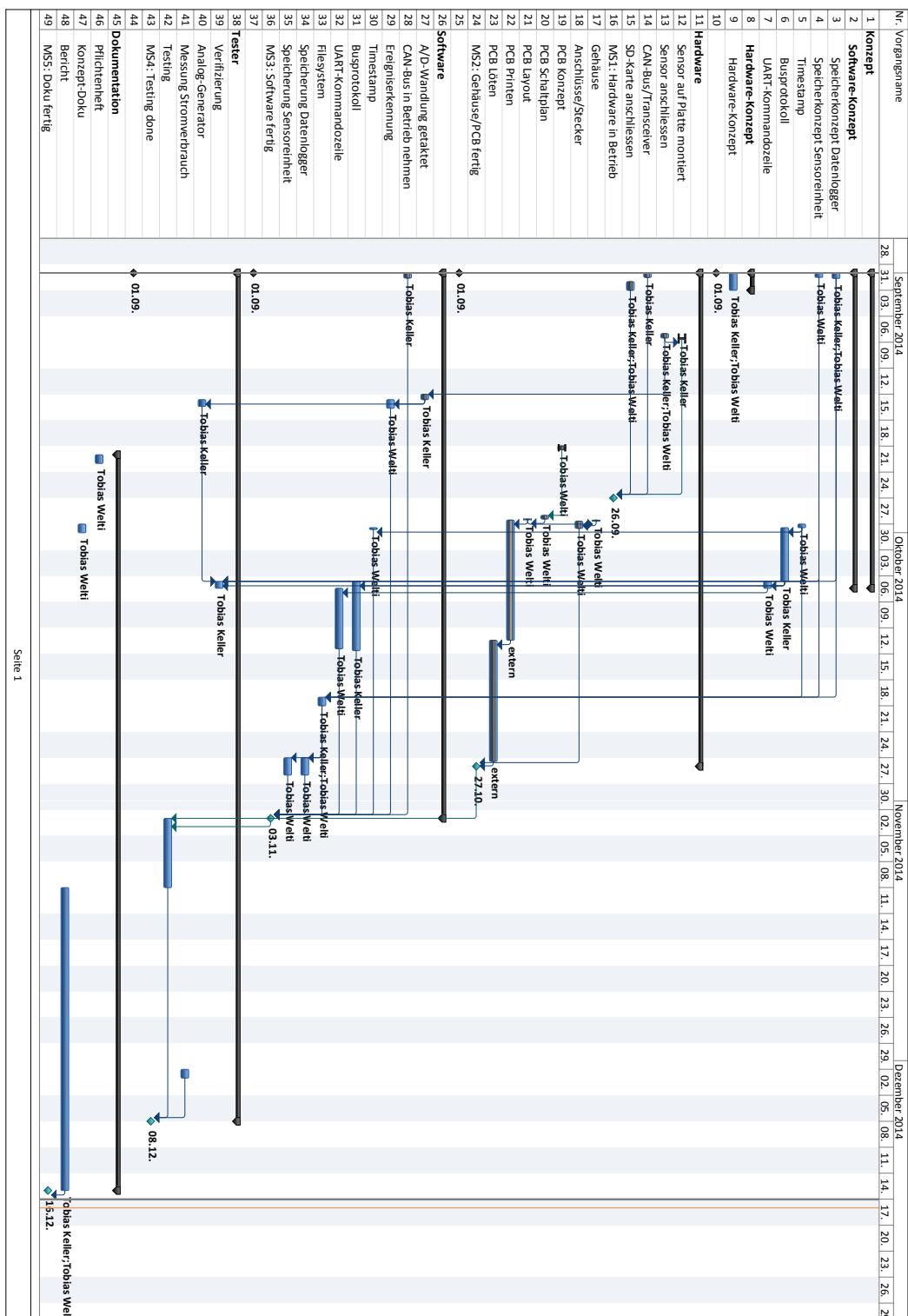


Abbildung A.1.: Gantt-Chart des Projekts.

der Peakintensitäten über eine Minute und die Speicherung der Dauer, Intensität und Anzahl Peaks jedes Ereignisses. Als minimale Anforderung an die Messdaten wurde vereinbart, dass die Abtastrate mindestens 10 kHz beträgt und mindestens die gleichen Werte gespeichert werden wie bisher.

Als Wunsch wurde von Carlos Wyss geäussert, die Originaldaten von Anfang bis Ende der Ereignis speichern zu können.

### Zwischenbesprechung an der ZHAW, Winterthur

Eine Zwischenbesprechung am 28. Oktober 2014 an der ZHAW Winterthur verlief sehr zufriedenstellend. Anwesend waren Prof. Hans Gelke (Betreuer), Bruno Fritschi sowie Tobias Keller und Tobias Welti.

**Projektstand** An der Zwischenbesprechung konnte bereits der Testaufbau mit einer Sensoreinheit vorgeführt werden. Die Signale wurden auf einem Oszilloskop visualisiert und entsprachen den Erwartungen von Bruno Fritschi. Die Dimensionen der Geräte waren jedoch wesentlich grösser als erwartet.

**Beschlüsse** In der Diskussion wurde entschieden, die bereits im Aufbau befindlichen Gehäuse trotzdem fertig zu bauen. Das Projekt war zu diesem Zeitpunkt auf einem guten Weg, weshalb keine Massnahmen oder Änderungen an den Zielen beschlossen wurden.

### Zwischenbesprechung an der VAW, ETH Hönggerberg

Am 12. November 2014 fand eine zweite Projektsitzung statt, diesmal an der VAW an der ETH Hönggerberg. Anwesend waren Prof. Hans Gelke, Prof. Dieter Rickenmann (WSL), Bruno Fritschi, Carlos Wyss sowie Tobias Keller und Tobias Welti.

**Projektstand** Die Methode der Ereigniserkennung mittels einer State Machine mit konfigurierbaren Parametern wurde vorgestellt und von den Anwesenden gelobt. Die verschiedenen Detail-Level wurden intensiv diskutiert und Vor- und Nachteile aufgeführt.

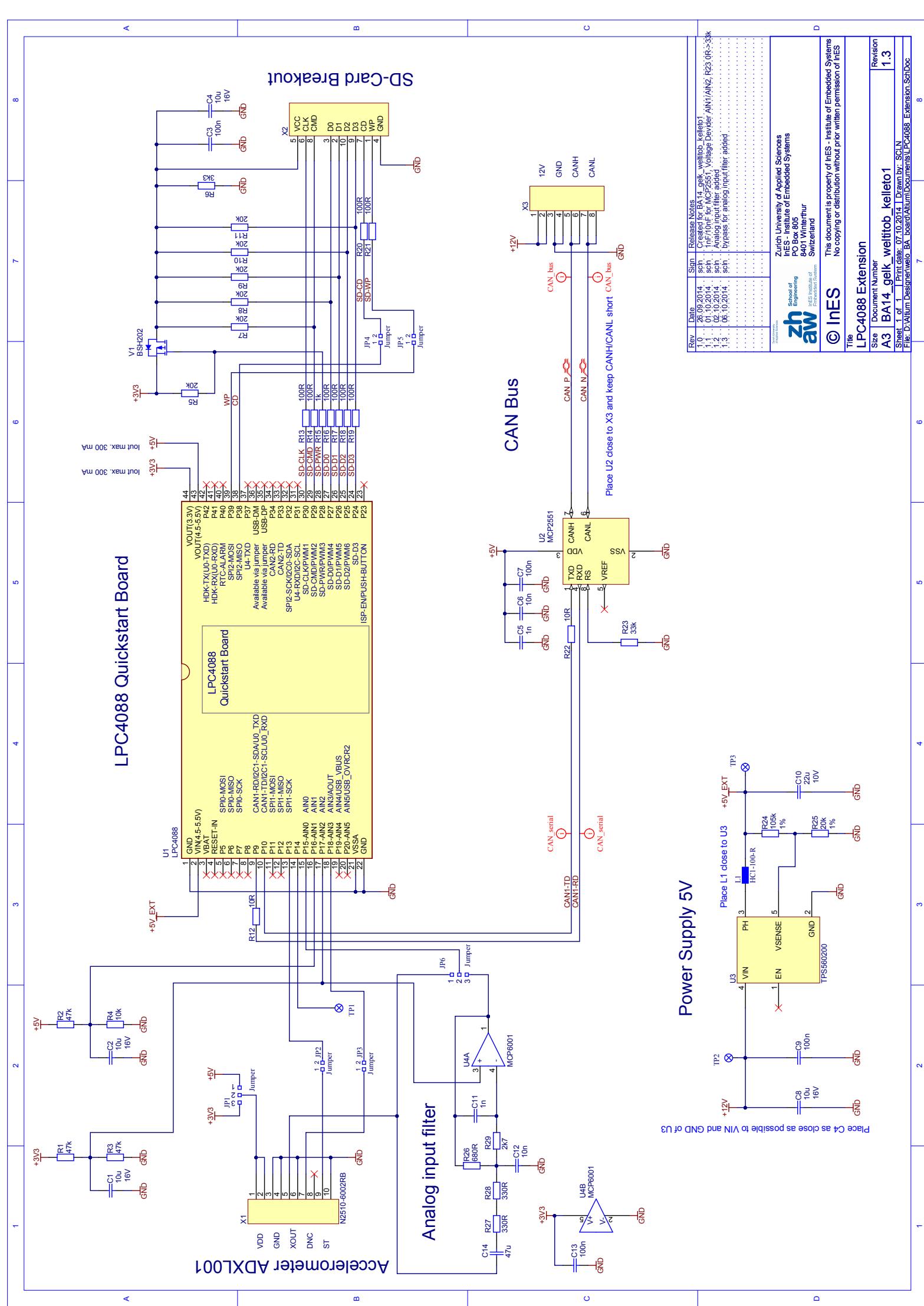
**Beschlüsse** Es wurde beschlossen, das Projekt mit allen vorgeschlagenen Detail-Levels weiterzuführen, da für jeden Modus ein plausibles Szenario gefunden wurde. Da das Projekt mit der Ereigniserkennung und Teilen des CAN-Protokolls gut im Zeitplan lag, wurden keine Änderungen der Ziele definiert. Die geplanten Tests an der VAW wurden aber auf einen Zeitpunkt nach dem Abschluss der Bachelorarbeit verschoben.

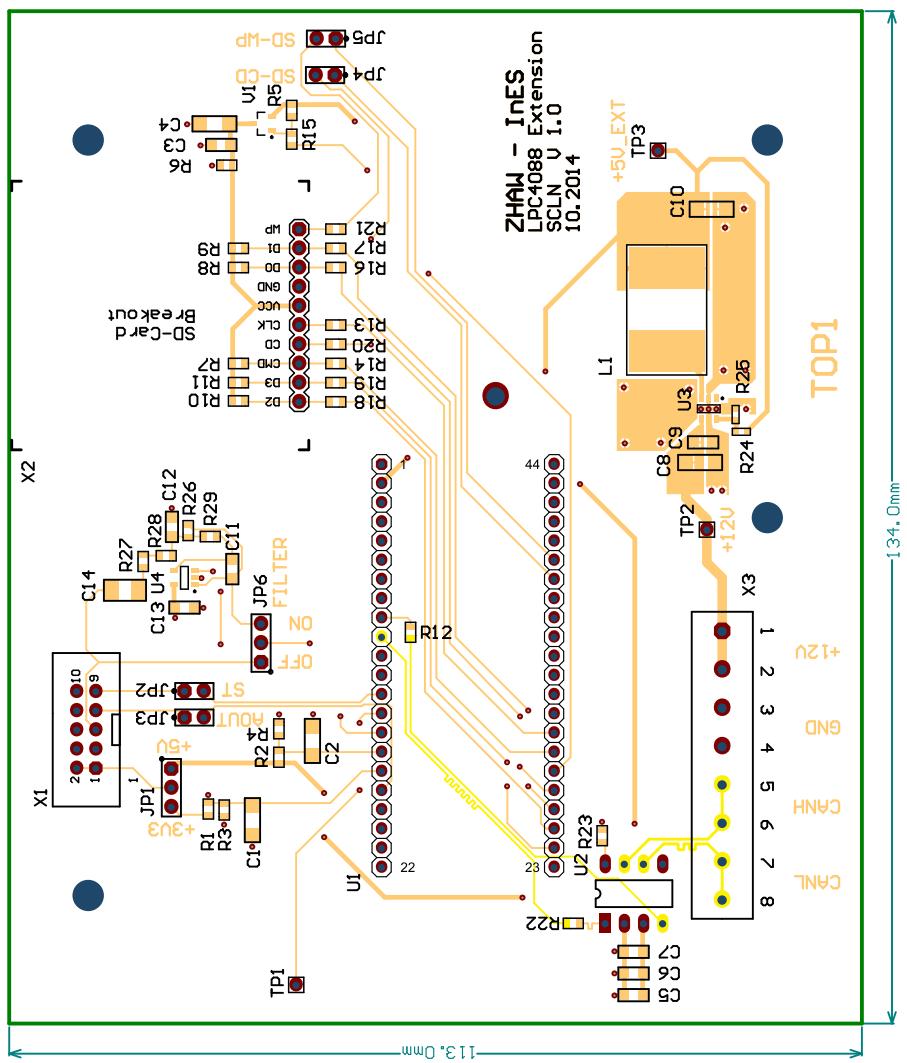
Für die Tests in der Versuchsrinne ist vorgesehen, die Sensoren ausserhalb der Gehäuse der Sensoreinheiten anzuschliessen und unter die bestehende Stahlplatte zu montieren. Dann können Vergleichsmessungen mit Geophonen und MEMS-Beschleunigungssensoren durchgeführt werden.

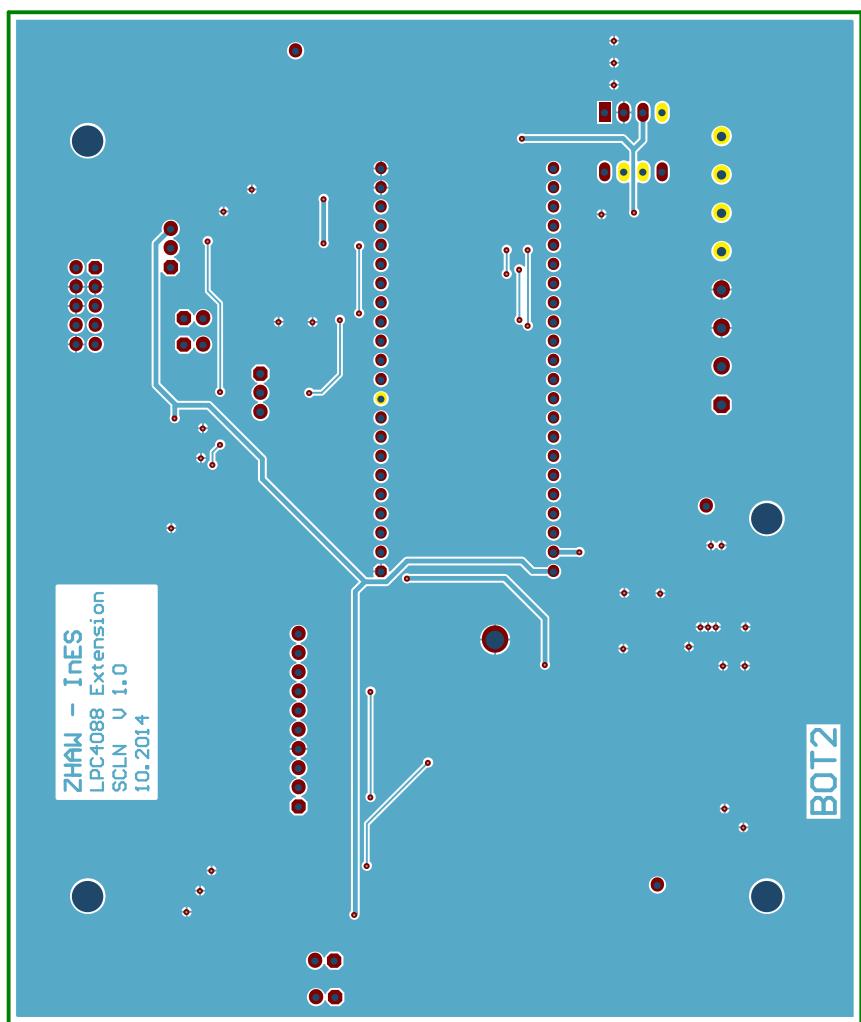
Als Folgeprojekt oder möglicher Entwicklungsauftrag an die ZHAW wurde die Miniaturisierung der Sensoreinheit und deren Entwicklung zur Produktreife ins Auge gefasst.

Im Anschluss an die Sitzung wurde die VAW besichtigt. Hier konnten sich die Entwickler zum ersten Mal ein (eindrückliches!) Bild der bestehenden Messinstallationen mit Geophonen machen.

### **A.3. Schaltpläne**





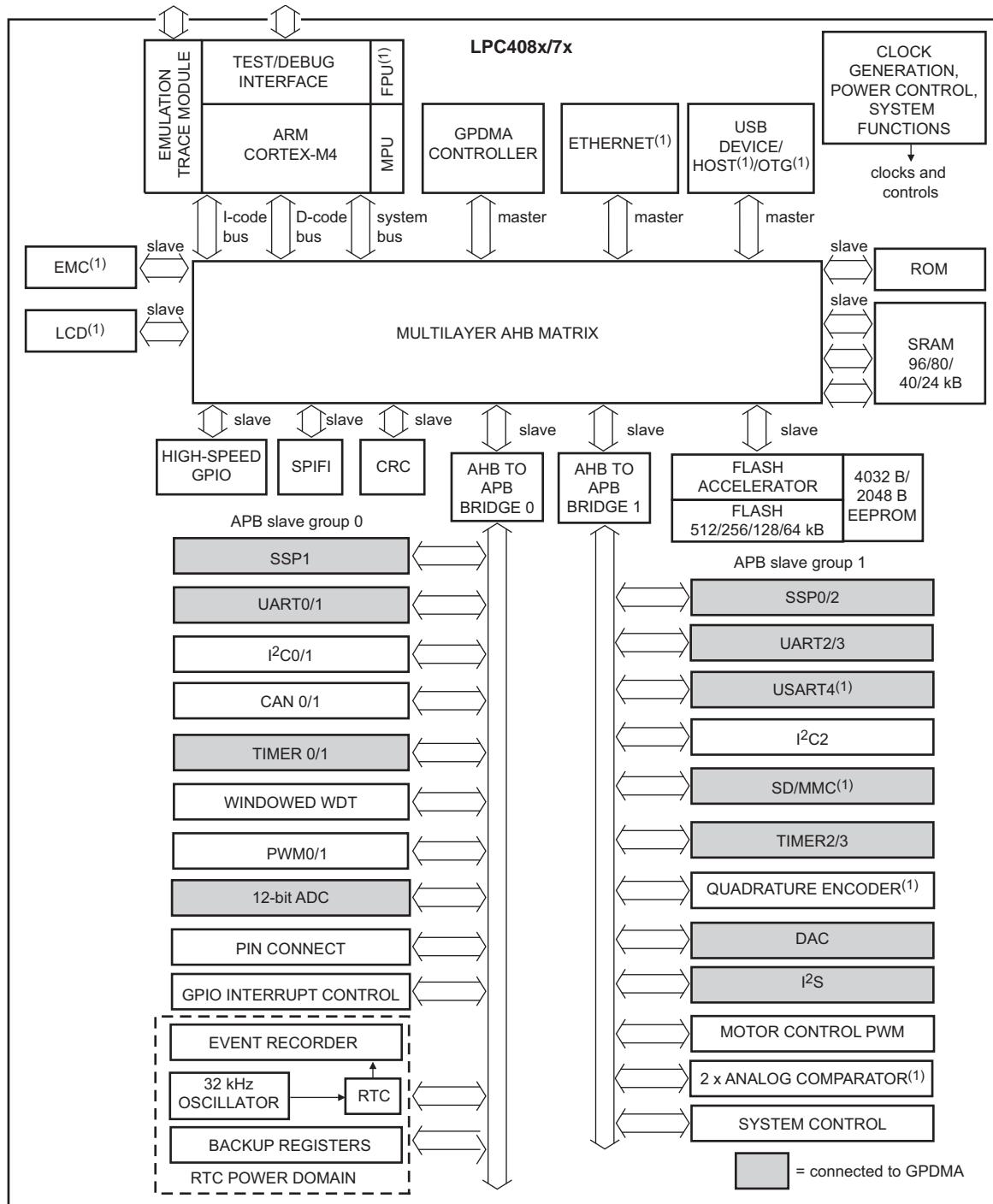


## A.4. Datenblätter

Um die Dokumentation übersichtlich zu halten, wird der Grossteil der Datenblätter nicht mit der Dokumentation ausgedruckt, sondern auf der beiliegenden Compact Disc (CD) mitgeliefert.

### A.4.1. NXP LPC4088 32-bit ARM Cortex-M4 microcontroller

## 5. Block diagram



002aag491

(1) Not available on all parts.

**Fig 1. Block diagram**

#### A.4.2. Embedded Artists NXP LPC4088 QuickStart Board

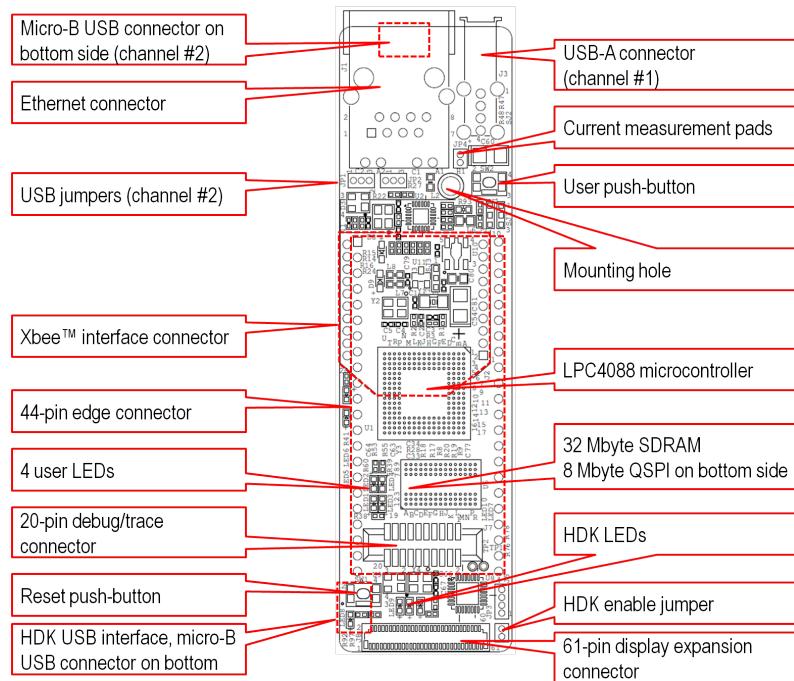


Abbildung A.2.: Hauptkomponenten des NXP LPC4088 QuickStart Boards von Embedded Artists.

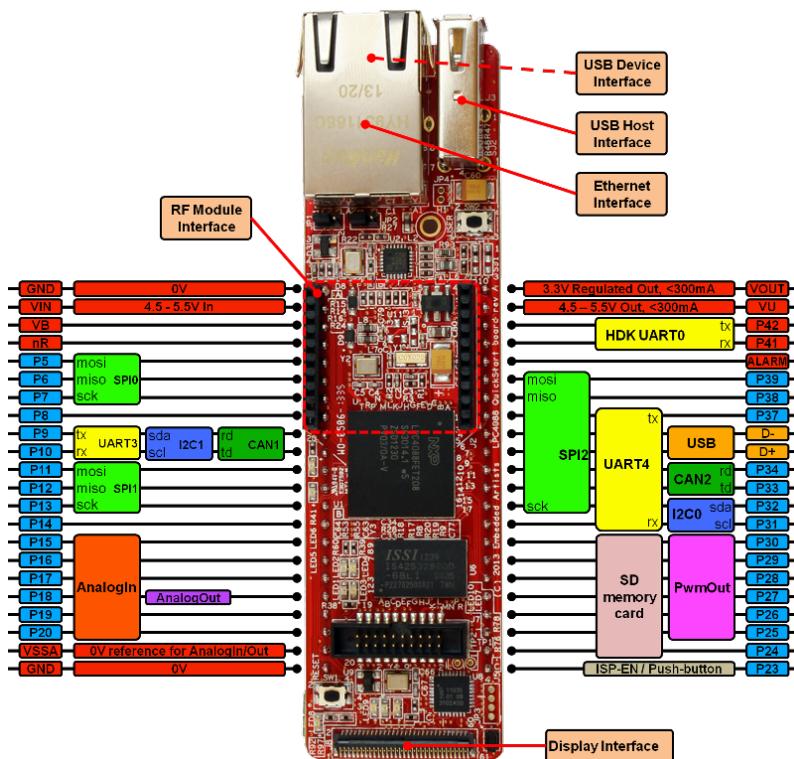


Abbildung A.3.: Pins des NXP LPC4088 QuickStart Boards von Embedded Artists.



Search mbed.org...

Go

Login or signup (/account/login/?)

# (<https://mbed.org/>)

[Users\(/activity/\)](#) » [embeddedartists\(/users/embeddedartists/\)](#) » [Notebook\(/users/embeddedartists/notebook\)](#) » LPC4088 QuickStart Board - Hardware Information

## LPC4088 QuickStart Board - Hardware Information

Page last updated 14 Nov 2013(14 Nov 2013), by  [EmbeddedArtists AB](#) (<https://developer.mbed.org/users/embeddedartists/>). 17 replies ([/users/embeddedartists/notebook/lpc4088-quickstart-board---hardware-information/#commentform](#))

## LPC4088 QuickStart Board Hardware Features

### MCU

- LPC4088, Cortex-M4F core running at up to 120MHz

### Memory

- 512 KByte on-chip FLASH
- 96 KByte on-chip SRAM
- 4 KByte on-chip E2PROM
- 8 MByte QSPI FLASH (can execute program code and/or contain a file system)
- 32 MByte SDRAM with 32-bit databus access
- On-board globally unique MAC address.

### Connectors

- 2x22 pin edge pins that are very compatible with the original 2x20 pin LPC1768 mbed pinning
- 10/100Mbps Ethernet (RJ45)
- USB-A (USB Host interface)
- USB-micro B (USB Device interface)
- USB-micro B (mbed HDK debug interface)
- 20 pos SWD/Trace connector (ARM standard debug connector)
- 61 pos 0.3mm pitch FPC connector for **display** expansion
- 20 pos **XBee** compatible connector for RF module add-on

### Display expansion

- Up to 24-bit pixel data (16-bit most common) via 61 pos, 0.3mm pitch FPC connector.

### Other

- Proper **ESD protection** on communication interfaces.
- On-board HDK (debug interface functions)
- Supported by the on-line mbed SDK.
- Supported by a lot of professional quality software and examples by EA.
- Industrial temperature specified (-40 to +85 degrees Celsius).
- **ISO 9001** produced.
- Production compensated for carbon dioxide emission.
- Current consumption down to about **5 mA**.

## Pin Usage

LPC4088 QuickStart Board - Hardware Information

Page owner:  [EmbeddedArtists AB](#) (<https://developer.mbed.org/users/embeddedartists/>)

Created 05 Sep 2013(05 Sep 2013).

Last updated 14 Nov 2013(14 Nov 2013)

#### **A.4.3. Analog Devices ADXL001 Beschleunigungssensor**

## SPECIFICATIONS

### SPECIFICATIONS FOR 3.3 V OPERATION

$T_A = -40^\circ\text{C}$  to  $+125^\circ\text{C}$ ,  $V_S = 3.3 \text{ V} \pm 5\%$  dc, acceleration = 0 g, unless otherwise noted.

Table 1.

Parameter	Conditions	ADXL001-70			ADXL001-250			ADXL001-500			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
SENSOR											
Nonlinearity		0.2	2		0.2	2		0.2	2		%
Cross-Axis Sensitivity	Includes package alignment	2			2			2			%
Resonant Frequency		22			22			22			kHz
Quality Factor		2.5			2.5			2.5			
SENSITIVITY											
Full-Scale Range Sensitivity	$I_{\text{OUT}} \leq \pm 100 \mu\text{A}$ 100 Hz	-70		+70	-250		+250	-500		+500	$g$ $\text{mV/g}$
16.0					4.4			2.2			
OFFSET	Ratiometric	1.35	1.65	1.95	1.35	1.65	1.95	1.35	1.65	1.95	V
Zero-g Output											
NOISE											
Noise	10 Hz to 400 Hz	85			95			105			$\text{mg rms}$
Noise Density	10 Hz to 400 Hz	3.3			3.65			4.25			$\text{mg}/\sqrt{\text{Hz}}$
FREQUENCY RESPONSE											
-3 dB Frequency		32			32			32			kHz
-3 dB Frequency Drift Over Temperature		2			2			2			%
SELF-TEST											
Output Voltage Change		400			125			62			mV
Logic Input High		2.1			2.1			2.1			V
Logic Input Low			0.66			0.66			0.66		V
Input Resistance	To ground	30	50		30	50		30	50		$\text{k}\Omega$
OUTPUT AMPLIFIER											
Output Swing	$I_{\text{OUT}} = \pm 100 \mu\text{A}$	0.2		$V_S - 0.2$	0.2		$V_S - 0.2$	0.2		$V_S - 0.2$	V
Capacitive Load		1000			1000			1000			pF
PSRR (CFSR)	DC to 1 MHz	0.9			0.9			0.9			V/V
POWER SUPPLY ( $V_S$ )											
Functional Range		3.135	6		3.135	6		3.135	6		V
$I_{\text{SUPPLY}}$		2.5	5		2.5	5		2.5	5		mA
Turn-On Time		10			10			10			ms

## A.5. Fotos

Die Fotos befinden sich auf der beiliegenden CD und können auch aus dem Git-Repository unter <https://github.com/tokeller/WeKeBA> heruntergeladen werden.

## A.6. Source Code, Daten und Multimedia

Da der Source Code sehr umfangreich ist, wird darauf verzichtet, ihn ausgedruckt zur Verfügung zu stellen. Er befindet sich auf der beiliegenden CD.

Inhaltsverzeichni  
CD erstellen  
CD mit dem voll  
gen Bericht als p  
inklusive Film- u  
tomaterial