

HSR Hochschule für Technik
Abt. Elektrotechnik

Multi-Sensor-Erschütterungs- Messgerät Diplomarbeit 2008

Modul: Digitale Signalverarbeitung
Name: Remo Fehr
Betreuung: Prof. Dr. Guido M. Schuster
Beginn: 03.03.2008
Abgabe: 25.04.2008

Inhaltsverzeichnis

1. Abstract	1
2. Einleitung	2
2.1. Zielsetzung	2
2.2. Inhalt der Dokumentation	2
3. Aufgabenstellung	3
4. FAT	5
4.1. Bootsektor	5
4.2. Reservierte Sektoren	5
4.3. FAT	6
4.4. Stammverzeichnis	6
4.5. Versionen	7
5. Hardware	8
5.1. Konzept	8
5.2. Power-Supply	8
5.3. Microcontroller	9
5.4. Beschleunigungssensor	9
5.5. MicroSD-Karte	10
5.6. PCB	11
5.7. Gehäuse	12
6. Software Messgerät	13
6.1. Implementation	13
6.2. Messung	14
6.3. Probleme mit dem Sensor	16
6.4. Loggen	17
7. Software Matlab	19
7.1. GUI	19
7.2. Variablen	19
7.3. Berechnungen	20
7.4. Ausgabe	22
8. Verwendete Software Werkzeuge	23
8.1. IAR Embedded Workbench	23
8.2. Matlab	23
8.3. OrCAD	24
8.4. LateX	24
9. Zusammenfassung	25

10. Ausblick	26
11. Schlusswort	27
A. Schemata	28
B. Stückliste	31
C. Abkürzungsverzeichnis	32
D. Abbildungsverzeichnis	33
E. Literatur	34

1. Abstract

Ziel dieser Diplomarbeit war es, ein kostengünstiges, portables Gerät zu entwickeln, welches Erderschütterungen messen und loggen kann.

Eines der Hauptprobleme bei der seismischen Messung ist, dass die zu messenden Erdschütterungen sehr klein sind, genau genommen zu klein für moderne MEMS Beschleunigungssensoren, denn diese haben ein Rauschen in der gleichen Grössenordnung. Rauschen wird normalerweise durch zeitliches Mitteln reduziert, was aber die Frequenzauflösung der Sensordaten signifikant verschlechtert und somit für diese Anwendung nicht geeignet ist.

Um das relativ grosse Rauschen zu reduzieren, ohne dass die Frequenzauflösung darunter leidet, wurden mehrere MEMS Sensoren, welche gleichzeitig messen, in einem Messgerät eingebaut. Die MEMS Sensoren wurden an einen Microcontroller(CC2430) angeschlossen, welcher die erhaltenen Beschleunigungsmesswerte aller Sensoren auf einer MicroSD-Karte im Dateiformat FAT16 loggt. Somit können die Daten von einem PC aus wieder eingelesen werden. Mit einem Matlab-Skript wird dann von den eingelesenen Beschleunigungsmesswerten aller Sensoren, zur Reduktion des Rauschens, das Scharmittel gebildet. Aus dem gemittelten Beschleunigungssignal wird mittels Integration das verlangte Geschwindigkeitssignal berechnet.

2. Einleitung

Eines der Hauptprobleme bei der Messung von Erderschütterungen ist, dass die zu messenden Erschütterungen sehr klein sind, genau genommen zu klein für moderne MEMS Beschleunigungssensoren, da diese ein Rauschen in der gleichen Größenordnung haben. Rauschen wird meistens durch zeitliches Mitteln reduziert, was aber die Frequenzauflösung der Sensordaten signifikant verschlechtert.

Eine andere Möglichkeit ist mehrere Sensoren zu verwenden welche gleichzeitig messen. So kann das Rauschen durch mitteln über die Sensoren, also durch das Bilden des Scharmittels, reduziert werden, wobei sich aber die Frequenzauflösung der Sensoren nicht verändert. Das Rauschen sollte mit dieser Methode um den Faktor $\sqrt{\text{AnzahlSensoren}}$ kleiner werden.

2.1. Zielsetzung

Diese Diplomarbeit dient dazu, ein kostengünstiges und portables Messsystem zu entwickeln, mit dem Erderschütterungen detektiert und ausgewertet werden können. Dazu sollen mehrere MEMS Beschleunigungssensoren, welche gleichzeitig messen, in ein Messgerät eingebaut werden. Die erhaltenen Beschleunigungsdaten sollen auf einer MicroSD-Karte so geloggt werden, dass sie von einem PC wieder eingelesen und dargestellt werden können.

2.2. Inhalt der Dokumentation

Zu Beginn wird eine kurze Einführung in das FAT-Dateisystem gegeben. Anschliessend wird die selbst entwickelte Hardware erläutert. In den folgenden Kapiteln wird die implementierte Applikation des Messgerätes erklärt. Auf die von der FAT-Library ausgeführten Funktionen wird, abgesehen von einer Ausnahme, nicht weiter eingegangen. In einem weiteren Kapitel ist die Matlab Software beschrieben.

3. Aufgabenstellung

Fachhochschule Ostschweiz
**HOCHSCHULE
RAPPERSWIL
HSR**

Aufgabenstellung

Thema: Multisensor Erschütterungs-Messgerät

Student: Remo Fehr
Betreuer: Guido Schuster
Partner: IBU, Fachstelle für Geotechnik

Kurzbeschreibung

Ziel dieser Arbeit ist es, ein kostengünstiges, portables Gerät zu entwickeln, welches seismische Erderschütterungen messen und loggen kann. Eines der Hauptprobleme bei der seismischen Messung ist, dass die zu messenden Erschütterungen sehr klein sind, genaugenommen zu klein für moderne MEMS Beschleunigungssensoren, da diese ein Rauschen in der gleichen Grössenordnung haben. Rauschen wird normalerweise durch Mittelung reduziert, was aber die Frequenzauflösung der Sensordaten signifikant reduziert und somit nicht zulässig ist. In dieser Arbeit soll versucht werden, durch viele (3, 5, 7 oder noch mehr) MEMS Sensoren welche gleichzeitig messen das Rauschen zu reduzieren, ohne dass die Frequenzauflösung darunter leidet. Diese Sensoren müssen an einen Mikrocontroller (CC2430) angeschlossen werden und gleichzeitig sollen die Daten aller Sensoren auf einer SD Karte so geloggt werden, dass sie von einem PC gelesen werden können, damit man die Signale später genauer betrachten kann. Das Gerät soll mit zwei handelsüblichen AAA Akkus betrieben werden können, was gewisse Anforderungen an die SD Karte stellt, welche heutzutage von zum Beispiel vom SanDisk Memory Stick Micro (M2) erfüllt wird. Einen Grossteil der Hardware ist schon vorhanden nur die SD Kartenanbindung und die Sensorenanbindung müssen noch entwickelt werden.

Aufgabenstellung

- Einarbeitung in die bestehende Hardware/Software und Theorie
- Selektieren und Anschliessen geeigneter MEMS Beschleunigungssensoren
- Selektieren und Anschliessen einer geeigneten SD Karte
- Datenlogging der Daten implementieren, so dass SD Karte von einem PC gelesen werden kann.
- Entwicklung der Rauschunterdrückung in Matlab mit den geloggt Daten
- Austesten der Lösung

Abbildung 3.1: Aufgabenstellung Seite 1

Fachhochschule Ostschweiz

**HOCHSCHULE
RAPPERSWIL
HSR**

Aufgabenstellung

Erwartete Ergebnisse

- Dokumentation der Theorie, der Lösungsansätze, der Lösung, der Hardware und der Software
- Ein funktionsfähiges Multisensor Erschütterungs-Messgerät mit Logging auf der SD Karte
- Ein Laborbuch

Arbeitsweise

- Sie führen ein persönliches Laborbuch, wo Sie aufschreiben wann Sie was für wie lange machen und was die Ergebnisse sind
- Sie schicken mir vor jeder Sitzung eine Zusammenfassung welche dokumentiert, was Sie in der letzten Woche gemacht haben.

Abbildung 3.2: Aufgabenstellung Seite 2

4. FAT

Die File Allocation Table (FAT) ist eine Art Verwaltungssystem und Inhaltsverzeichnis für Speichermedien.

Ein FAT Dateisystem gliedert sich in fünf Bereiche:

Bootsektor	reservierte Sektoren	FAT	Stammverzeichnis	Datenbereich
------------	----------------------	-----	------------------	--------------

Tabelle 4.1: FAT Aufbau[1]

4.1. Bootsektor

Der Bootsektor enthält ausführbaren x86-Maschinencode, der das Betriebssystem laden soll. Jedoch enthält er an bestimmten Stellen Informationen über das FAT-Dateisystem. Einige Beispiele sind hier kurz aufgelistet:[1]

Offset (Hex)	Länge (in Byte)	Inhalt
0B	2	Bytes pro Sektor
0D	1	Sektoren pro Cluster
0E	2	Anzahl reservierter Sektoren(inkl. Bootsektor)
10	1	Anzahl der FAT-Kopien
11	2	Max. Anzahl der Verzeichniseinträge im Stammverzeichnis
16	2	Anzahl der Sektoren pro FAT
36	8	FAT-Variante, mit Leerzeichen aufgefüllt, z.B. "FAT16 "

Tabelle 4.2: Bootsektor[1]

4.2. Reservierte Sektoren

Es können zwischen Bootsektor und der ersten FAT Sektoren reserviert werden, die vom Dateisystem nicht benutzt werden. Dieser Bereich kann von einem Bootmanager oder für betriebssystemspezifische Erweiterungen genutzt werden. Auf den meisten FAT12 oder FAT16 Dateisystemen existieren, ausser dem Bootsektor, keine weiteren reservierten Sektoren. Die FAT folgt somit direkt anschliessend an den Bootsektor. FAT32 Dateisysteme enthalten in der Regel noch einige Erweiterungen zum Bootsektor sowie eine komplette Sicherungskopie des Bootsektors und der Erweiterungen.[1]

4.3. FAT

Die FAT ist eine Art Tabelle fester Grösse. In dieser Tabelle wird eingetragen, welche Cluster eines FAT-Datei Systems frei und welche belegt sind. Ein Cluster besteht aus einem oder mehreren Sektoren, und kann von einer Datei belegt werden. Der Datenbereich ist in eine feste Anzahl von Clustern eingeteilt. Zu jedem dieser Cluster existiert ein Eintrag in der FAT, der folgendes über den Cluster angeben kann:[1]

- Der Cluster ist nicht belegt,
 - der Cluster ist frei.
 - das Medium ist an der Position dieses Clusters beschädigt.
- Der Cluster ist von einer Datei belegt,
 - der nächste Cluster der Datei ist der Cluster mit der Nummer $(X - 2)$.
 - dies ist der letzte Cluster der Datei.

Die belegten Cluster einer Datei bilden eine verkettete Liste, die Clusterkette genannt wird. Wegen ihrer grundlegenden Bedeutung für das Dateisystem existieren in der Regel zwei Kopien der FAT, um bei Datenverlust noch immer eine funktionsfähige andere FAT zu haben. Mit diversen Programmen ist eine Datenwiederherstellung in vielen Fällen möglich.[1]

4.4. Stammverzeichnis

Im Stammverzeichnis, auch Wurzelverzeichnis oder Hauptverzeichnis genannt, wird in der Regel von jeder Datei oder jedem Unterverzeichnis ein Eintrag gemacht. Bei FAT12 und FAT16 hat das Stammverzeichnis eine feste Grösse und folgt direkt nach der FAT. Die Grösse wird beim Formatieren festgelegt und kann später nicht mehr geändert werden. Bei FAT32 hat das Stammverzeichnis eine variable Grösse und kann an irgendeiner Position im Datenbereich beginnen.[1]

Ein Eintrag im Stammverzeichnis besteht aus 32 Byte.

Länge (in Byte)	Inhalt
8	Dateiname ohne Erweiterung (mit Leerzeichen aufgefüllt)
3	Erweiterung (mit Leerzeichen aufgefüllt)
1	Dateiattribute
10	Reserviert
2	Zeit
2	Datum
2	(Offset des Start-Clusters)+2
4	Dateigrösse in Bytes

Tabelle 4.3: Stammverzeichniseintrag[1]

4.5. Versionen

Es gibt verschiedene Versionen von FAT, die gängigen sind FAT12, FAT16 und FAT32. Hier eine kurze Übersicht über die verschiedenen Versionen:[1]

FAT12: 12-Bit-Clusternummern, 4096 Einträge möglich, 224 Dateien, 8.3 Dateinamensformat

FAT16: 16-Bit-Clusternummern, 65536 Einträge möglich, 512 Dateien, 8.3 Dateinamensformat

FAT32: 32-Bit-Clusternummern, 2^{28} Dateien, lange Dateinamen

VFAT: Erweiterung des FAT-Formats, lange Dateinamen

exFAT: max. Dateigrösse 2^{64} Byte, neue Tabelle welche die freien Cluster indiziert, Vergabe von Zugriffsberechtigungen möglich

5. Hardware

5.1. Konzept

Die Hardware des ganzen Erschütterungsmessgerätes befindet sich auf einem einzigen PCB in einem kompakten Gehäuse und enthält folgende Komponenten:

- Power-Supply
- Microcontroller
- Beschleunigungssensoren
- MicroSD-Kartensteckplatz



5.2. Power-Supply

Abbildung 5.3: Messhardware

Das Erschütterungsmessgerät soll mit zwei handelsüblichen AAA Batterien betrieben werden können. Das heisst die Betriebsspannung liegt bei $2 * 1.2V = 2.4V$, wenn Akkus verwendet werden, oder bei $2 * 1.5V = 3.0V$, wenn Batterien verwendet werden. Die MicroSD-Karte benötigt eine Spannung von $2.7 - 3.6V$, die Beschleunigungssensoren benötigen eine Spannung von $3.0 - 3.6V$ und der Microcontroller benötigt eine Spannung von $2.0 - 3.6V$. Eine geeignete Versorgungsspannung wäre $3.3V$. Dies kann mit einem Step-up Wandler erreicht werden. Da die Beschleunigungssensoren eine sehr hohe Auflösung aufweisen, ist es von Vorteil, wenn die Versorgungsspannung möglichst rauschfrei ist. Von MAXIM wird ein solcher Step-up Wandler angeboten, der diese Anforderungen erfüllt.

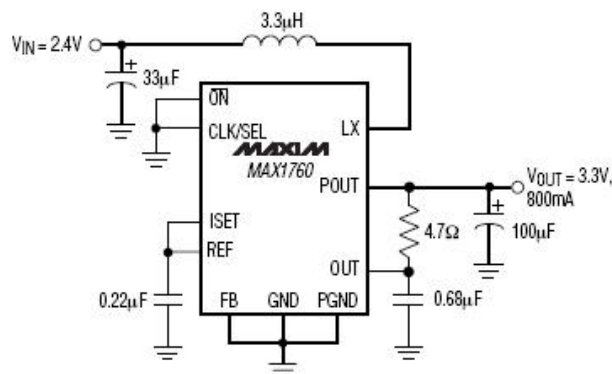


Abbildung 5.4: Step-up Wandler[2]

Um den Aufwand der Spannungsversorgung gering zu halten, wurde das im Datenblatt des Step-up Wandlers vorgeschlagene Schaltschema für den Standardgebrauch übernommen.

5.3. Microcontroller

Kernstück der Hardware ist der Microcontroller. Dieser nimmt die Aufgabe war, Messdaten, welche von den Beschleunigungssensoren erfasst werden, auf einer Speicherkarte so zu speichern, dass sie von einem PC wieder eingelesen werden können. Um den Hardwareaufwand nicht unnötig zu erhöhen, wurde für den Microcontroller das Evaluationboard CC2430EM von Chipcon eingesetzt. Über dieses können schliesslich alle verfügbaren IO's ab zwei Stiftleisten abgegriffen werden. Somit können die Sensorsignale und die Datensignale für die Speicherkarte einfach über diese Schnittstelle dem Microcontroller zugeführt werden. Die Datenleitungen zu der Speicherkarte und den Sensoren wurde mit den zwei im CC2430 verfügbaren SPI-Schnittstellen realisiert. Der CC2430 verfügt gerade über genügend I/O-Ports, damit alle Sensoren mit den von der SPI-Schnittstelle benötigten Chipselectleitungen angesteuert werden können. Somit kann auf einen Multiplexer verzichtet werden.



Abbildung 5.5:
CC2430EM

5.4. Beschleunigungssensor

Beim Beschleunigungssensor handelt es sich um einen 2-achsigen Beschleunigungssensor der Firma Analog Devices. Dieser ist für Beschleunigungen, respektive Verzögerungen bis 1.7 facher Gravitationsbeschleunigung ausgelegt. In Abbildung 5.6 ist der Aufbau des Beschleunigungssensors grob dargestellt. Der Sensor zeichnet sich durch seine hohe Auflösung von 0,244mg/LSB aus. Eine weitere Stärke des Sensors liegt in der relativ simplen Ansteuerung über SPI.

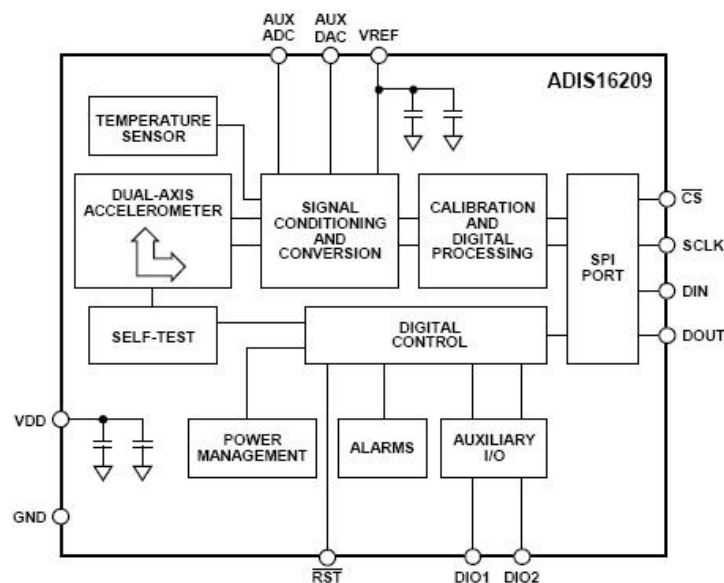


Abbildung 5.6: ADIS16209[3]

Der Beschleunigungssensor besitzt für jede Funktion ein einzelnes Register mit einer 6-Bit-Adresse, in das geschrieben oder aus dem gelesen werden kann. Um Messwerte zu erhalten, kann einfach mit SPI aus dem Register mit der entsprechenden Adresse ausgelesen werden. Es ist auch möglich, einige Parameter im Sensor, wie zum Beispiel die Samplerate, zu setzen. In der Tabelle 5.4 sind die wichtigsten verwendeten Register und deren Adresse aufgeführt. An der angegebenen Adresse befindet sich jeweils das niederwertige Byte, die Adresse des höherwertigen Byte ist einfach die Adresse des niederwertigen Byte um eins inkrementiert.

Name	R/W	Adresse	Grösse	Inhalt
XACCL_OUT	R	0x04	2 Byte	Output X-Achse
XACCL_NULL	R/W	0x12	2 Byte	Kalibration Null-Offset X-Achse
SMPL_PRD	R/W	0x36	2 Byte	Samplerate Konfiguration
STATUS	R	0x3C	2 Byte	System Status Register
COMMAND	W	0x3E	2 Byte	System Command Register

Tabelle 5.4: Register ADIS16209[4]

Um die Versorgungsspannung der Beschleunigungssensoren möglichst störungsfrei zu halten, wurde vor jedem Sensor ein Entstörkondensator mit der Kapazität 100nF eingebaut. Zur Strombegrenzung auf 1mA wurde jeweils vor dem Slave-Data-Out der Sensoren ein Widerstand $3.3V/1mA = 3.3k\Omega$ eingebaut. Weshalb die Strombegrenzung notwendig ist, wird in Kapitel 6.2 genauer erklärt.

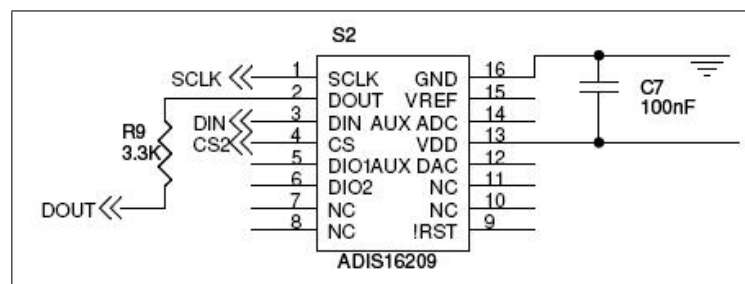


Abbildung 5.7: Anschlüsse Beschleunigungssensor

Wie bereits im oberen Abschnitt erwähnt wurde, wird der Beschleunigungssensor mit SPI angesteuert. Deshalb sind, wie in Abbildung 5.7 ersichtlich, nicht mehr Anschlüsse notwendig als die vier SPI-Leitungen, Speisung und Ground.

5.5. MicroSD-Karte

Als Speichermedium wird eine MicroSD-Karte verwendet, weil diese trotz sehr grossem Speicherplatz sehr klein ist. Die MicroSD-Karte kann via SPI-Schnittstelle angesteuert

werden. Die Pinbelegung der MicroSD-Karte weicht von der Pinbelegung der normalen SD-Karte ab.

Pin	MicroSD	SD
1	NC	CS
2	CS	SDI
3	SDI	GND
4	VDD	VDD
5	SCLK	SCLK
6	GND	GND
7	SDO	SDO
8	NC	NC

Tabelle 5.5: Pinbelegung MicroSD

Der MicroSD-Kartensteckplatz wurde, wie in Abbildung 5.8 dargestellt, an den Microcontroller angeschlossen. Die Widerstände in den vier SPI-Leitungen sind notwendig, um bei einem mechanischen Kurzschluss im Steckplatz den Strom zu begrenzen.

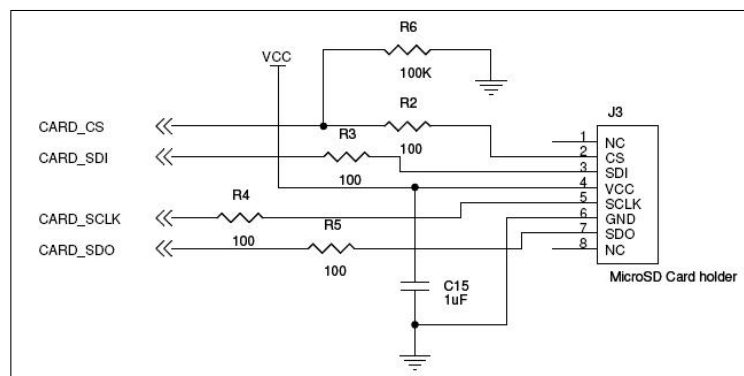


Abbildung 5.8: Schema MicroSD-Kartensteckplatz

5.6. PCB

Um das PCB möglichst klein zu halten, wurde ein zweiseitiges Layout gewählt. Auf der Unterseite wurden neun Beschleunigungssensoren und zwei AAA-Batterien platziert. Auf der Oberseite befinden sich die Stromversorgung, Power-on-off-Switch, der MicroSD-Kartensteckplatz und das Microcontrollerboard CC2430EM. Um das Debuggen des Microcontrollers zu ermöglichen wurde eine SoC debug Schnittstelle integriert. Damit es möglich ist, die Datensignale zu kontrollieren, wurden diese auf eine Stiftleiste geführt, wo sie abgegriffen werden können. Weiter befindet sich auf der Oberseite noch ein Taster, mit dessen Betätigung die MicroSD-Karte gelöscht werden kann.

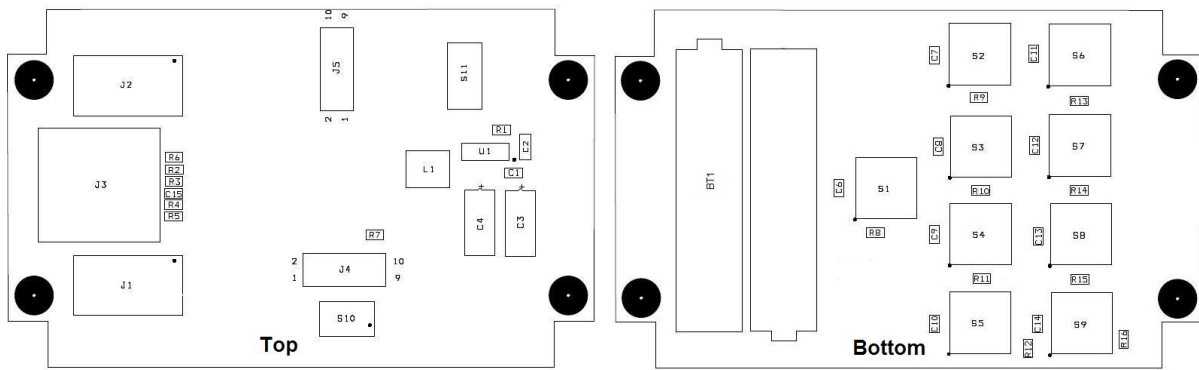


Abbildung 5.9: PCB Layout

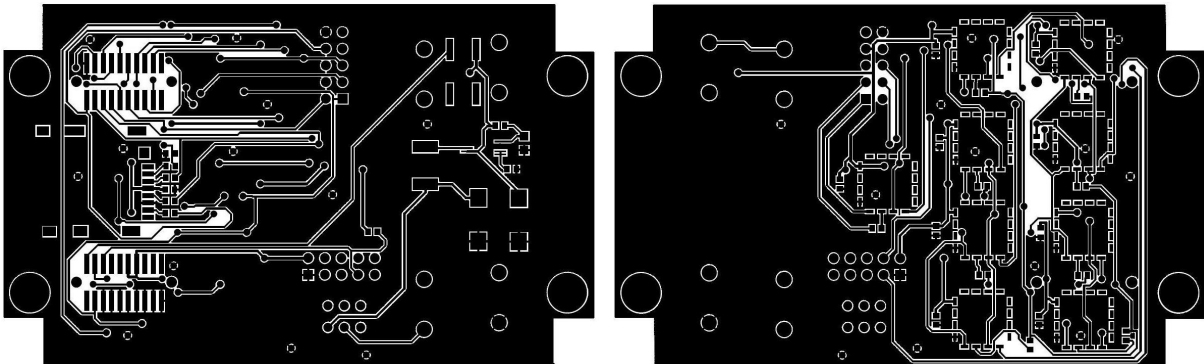


Abbildung 5.10: Bestückungsplan

5.7. Gehäuse

Die Messhardware wurde in ein Gehäuse von HAMMOND, mit den Massen 105x65x45mm, wie in Abbildung 5.3 dargestellt, eingebaut. Das Gehäuse wurde relativ klein gewählt da das Messgerät handlich und portabel sein soll. Darum wurde auch darauf verzichtet dass ev. bei einer weiterführenden Arbeit zusätzlich eine Funkantenne darin Platz hätte. In diesem Fall müsste ein neues Gehäuse gewählt werden.

6. Software Messgerät

6.1. Implementation

Die Software des Erschütterungsmessgerätes kann in zwei Teile geteilt werden. Zum einen Teil werden Messwerte von den Beschleunigungssensoren eingelesen. Zum anderen Teil werden die erhaltenen Messwerte von allen Sensoren im ASCII-Format auf eine MicroSD-Karte geschrieben. Für das Loggen dieser Messdaten auf der mit FAT16 formatierten MicroSD-Karte wurde eine vorhandene FAT Library eingebunden, welche die benötigten Funktionen bereits implementiert hat. Da die FAT-Library für einen MSP430 Microcontroller vorgesehen war, musste diese noch für den verwendeten CC2430 angepasst werden.

In Abbildung 6.11 ist ein Flussdiagramm dargestellt, das den Ablauf der Software grob beschreibt.

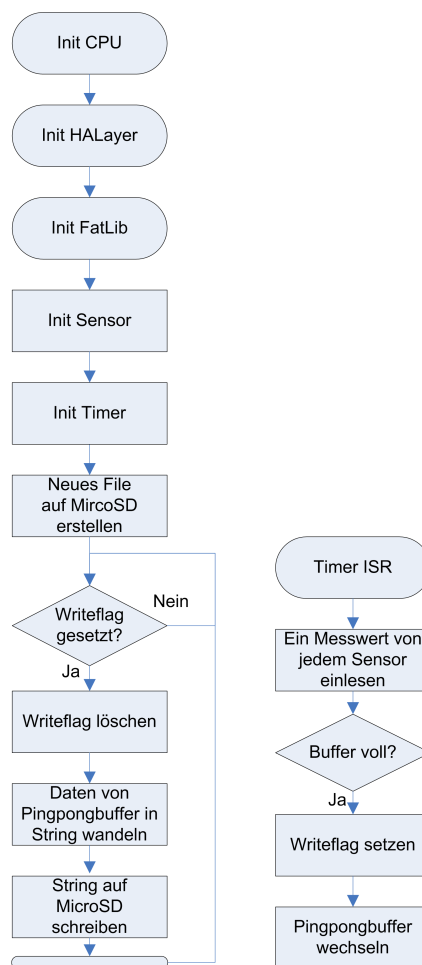


Abbildung 6.11: Flussdiagramm

6.2. Messung

Die Beschleunigungssensoren werden, wie im vorherigen Kapitel bereits erwähnt, via SPI angesteuert. Der Datenaustausch erfolgt in 16-Bit Blöcken. Für das Schreiben in ein Register des Sensors, zum Beispiel das Setzen der Samplerate, ist lediglich ein solcher Block, wie in Abbildung 6.12 dargestellt, als Write-Befehl notwendig. Darin enthalten sind ein Write/Read-Bit, die 6-Bit-Adresse des Registers und die 8-Bit Daten, welche in das Register mit der angegebenen Adresse geschrieben werden.

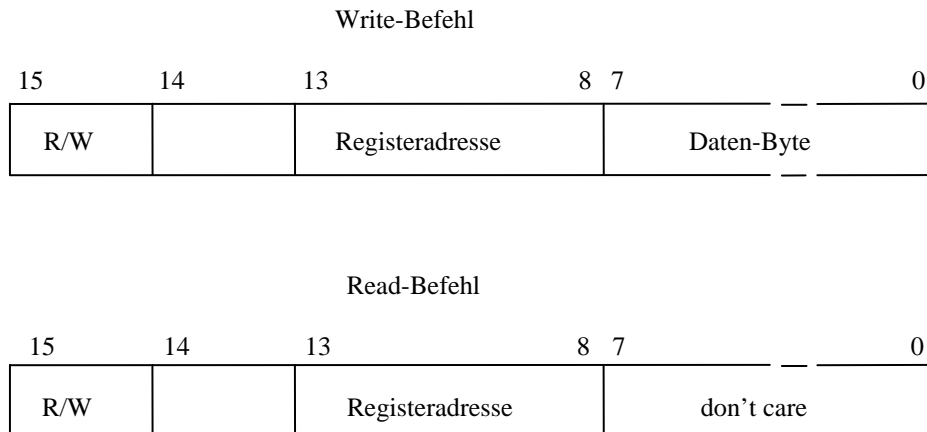


Abbildung 6.12: 16-Bit Sensor-Befehle

Um Messwerte zu erhalten, wird aus dem entsprechenden Register gelesen. Dazu wird zuerst ein 16-Bit Block als Read-Befehl übertragen, worauf der Sensor die verlangten Daten bereit stellt. Bei einem Read-Befehl sind die zweiten 8 Bit nicht relevant. Es müssen aber immer 16-Bit übertragen werden. Anschliessend können die Daten mit dem Schreiben des nächsten Befehls ausgelesen werden. Es wird immer das ganze 2 Byte grosse Register zurückgesendet, beginnend beim höherwertigen Byte. Es muss darauf geachtet werden, dass zum Auslesen der Daten ein gültiger Befehl gesendet wird. Die Messwerte können so sehr effizient ausgelesen werden, da bei jedem Schreiben eines neuen Befehls gleichzeitig die Daten des vorangehenden Befehls erhalten werden.

Für diese Anwendung ist es wichtig, dass die Sensoren gleichzeitig messen. Daher ist es nicht möglich die Messwerte, wie im oberen Abschnitt beschrieben, auszulesen.

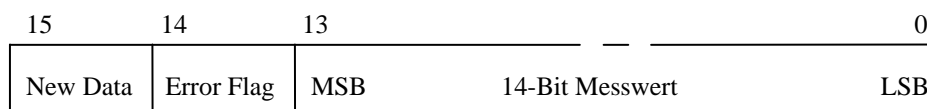


Abbildung 6.13: Messwert

Beim Auslesen eines Beschleunigungsmesswertes wird ein Messwert in der Form wie in Abbildung 6.13 dargestellt erhalten. Dieser beinhaltet ein *New Data* Flag, welches

aussagt, ob der erhaltene Messwert bereits ausgelesen wurde, ein *Error* Flag und den 14-Bit Messwert. Der Messwert wird in Zweierkomplement erhalten. Die 14-Bit werden jeweils auf 16-Bit erweitert, in dem das Bit an der Stelle 13, welches das Vorzeichen darstellt, an die Positionen 14 und 15 kopiert wird. Das *New Data* und *Error* Flag werden nicht beachtet.

Um von allen Sensoren über SPI einen Messwert vom genau gleichen Zeitpunkt zu erhalten, werden alle Chip Select der Sensoren gesetzt, um einen Read-Befehl an alle gleichzeitig zu senden. Um sicherzustellen, dass von den Sensoren während dem Senden des Read-Befehls an alle keine Antwort von einem vorherigen Befehl erhalten wird, was zu Kurzschlüssen zwischen den Sensoren führen kann, wird zum Auslesen ein Schreibbefehl für ein nicht benötigtes Register gesendet. Zusätzlich wurde in den Datenleitungen SDO kurz vor den Sensoren noch ein Widerstand zur Strombegrenzung eingebaut. Danach können die Messwerte von einem Sensor nach dem anderen ausgelesen werden.

Die höchste auftretende Frequenz bei Erderschütterungen liegt bei maximal 200Hz. Damit eine Erderschütterung dieser Frequenz detektiert werden kann, muss, unter Verwendung des Abtasttheorems, mit mindestens der doppelten Frequenz abgetastet werden. Das bedeutet, die Samplerate muss auf mindestens 400 Samples/s eingestellt werden.

Der Sensor kann in zwei verschiedenen Modi betrieben werden, im normalen Modus oder im schnellen Modus. Dies hängt von der internen Samplerate des Sensors ab. Bei einer Samplerate grösser als 481 Samples/s befindet sich der Sensor im schnellen Modus. Da der Stromverbrauch im schnellen Modus um ein Vielfaches höher ist als im normalen Modus, wird der Sensor im normalen Modus betrieben. Deshalb wurde die Samplerate kleiner als 481 Sample/s gewählt.

Um die Samplerate zu setzen, muss, wie in der Tabelle 6.6 vorgegeben, in das Register *SMPL_PRD* an der Adresse *0x36* geschrieben werden.

Bit	Description
15:8	Not used
7	Time base (t_b) 0 = 244.14 μ s, 1 = 7.568ms
6:0	Increment setting (N_S)

Tabelle 6.6: Register *SMPL_PRD* Bit Description [5]

Der Wert der in das Register geschrieben werden muss, wird mit den Formeln

$$t_S = t_B \cdot N_S + 122.07\mu s$$

und

$$f_S = 1/t_S$$

berechnet. Mit $t_B = 244.14\mu s$ und $N_S = 0x09$ erhält man für die Samplerate $f_S = 431\text{Samples/s}$.

Mit dieser Samplerate wird der Stromverbrauch niedrig gehalten und es werden alle auftretenden Frequenzen genügend schnell abgetastet.

Damit in einem definierten Zeitabstand gemessen wird, wurde ein Timer implementiert, welcher bei Erreichen eines Vergleichswertes einen Interrupt auslöst. Jedes Mal wenn der Interrupt ausgelöst wird, wird von jedem Sensor ein Messwert eingelesen. Der Timer wurde so implementiert dass alle $2.32ms$ ein Interrupt ausgelöst wird. Somit werden $1/0.00232 = 431$ Messwerte pro Sekunde eingelesen. Dies entspricht gerade der Samplerate der Sensoren.

Im normalen Modus kann für den SPI-Bus die Baudrate bis auf maximal $1MHz$ [6] gesetzt werden. Von den Sensoren wird aber im normalen Modus eine Zeit $t_{Datarate}$ von $100\mu s$ [6] für den Abstand zwischen zwei 16-Bit Blöcken vorgegeben, welche benötigt wird, um die Daten bereit zu stellen und somit nicht unterschritten werden darf. Es muss also eine Baudrate von kleiner $160kBit/s$ gewählt werden oder ein Delay eingefügt werden. Die Baudrate wurde auf $460,8kBit/s$ gesetzt und zwischen dem Read-Befehl an alle und dem anschliessenden Auslesen wurde ein Delay von $100\mu s$ eingefügt. Da nur alle $2,3ms$ ein Messwert eingelesen wird, muss kein weiterer Delay eingefügt werden.

6.3. Probleme mit dem Sensor

Das Einlesen von Messwerten von einem Sensor mit Evaluationboard funktioniert. Es konnten aber keine Messwerte von den Sensoren ausgelesen werden, welche wie vorgesehen auf das selbst entwickelte PCB gelötet wurden. Im Sensor ist ein Statusregister vorhanden in dem jedes Bit für ein bestimmter Fehler steht. Beim Auslesen des Statusregisters kommt die Meldung, dass die Speisespannung zu hoch ist. Über das Register *SUPPLY_OUT* mit der Adresse $0x02$ kann die über einen ADC digitalisierte Speisespannung ausgelesen werden. Die erhaltenen Werte sind nicht brauchbar und ändern auch nicht mit einer Spannungsänderung am Eingang des Sensors. Folglich sind die Sensoren defekt.

Zwei weitere Sensore haben einen Kurzschluss von Speisung auf Ground und wurden auf der Stelle wieder vom PCB entfernt. Ein fünfter auf das PCB gelötete Sensor sendet überhaupt keine Daten zurück und wenn andere Sensoren Daten senden erreichen diese nur noch die halbe Pegelspannung. Das heisst, der Sensor zieht das Datensignal auf Ground und ist damit ebenfalls defekt. Es wurden keine weiteren Sensoren auf das PCB gelötet, da es wahrscheinlich der Fall ist, dass diese durch das Löten kaputt gehen.

Da die Sensoren defekt sind, konnte keine Messung mit mehreren Sensoren durchgeführt werden. Da der eine Sensor mit Evaluationboard funktioniert, kann ausgeschlossen werden, dass es im Programmcode oder auf dem PCB einen Fehler hat.

6.4. Loggen

Die Beschleunigungsmesswerte werden im ASCII-Format auf die MicroSD-Karte geschrieben. Das heisst jeder 16-Bit Messwert wird in einen Character String umgewandelt. Die Messwerte werden, wie im oberen Abschnitt erwähnt, im Zweierkomplement erhalten. Da es eigentlich nur vorzeichenbehaftete 14-Bit Werte sind, liegen die Messwerte im Bereich von $\pm 2^{13} = \pm 8192$.

Für das umwandeln in einen String wird normalerweise die Funktion *sprintf()* verwendet. Da diese aber sehr langsam ist, wurde für die Umwandlung eine eigene Funktion implementiert. Unabhängig von der Anzahl Ziffern der Messwerte wird immer in einen 5 Byte langen ASCII-String umgewandelt.

Mit der Funktion *fat_openWrite()* wird auf der MicroSD-Karte ein neues File angelegt. Mit dem als Rückgabewert dieser Funktion erhaltenen *handle* und der Funktion *fat_write()* wird dann in das File geschrieben.

Um die benötigte Zeit für das Schreiben auf die MicroSD Karte zu optimieren, musste die Funktion *fat_write()* angepasst werden. Darum wird hier explizit auf einige von der FAT-Library ausgeführten Funktionen eingegangen.

Die Funktion *fat_write()* verwendet die drei verschiedenen Funktionen *writeSector()*, *writePartialSector()* und *writePartialMultiSector()* für das Schreiben der Daten auf die MicroSD-Karte. In der folgenden Tabelle werden die Unterschiede kurz erläutert.

<i>writeSector()</i> :	beschreibt einen Sektor auf der MicroSD-Karte
<i>writePartialSector()</i> :	beschreibt einen Teil eines Sektors auf der MicroSD-Karte
<i>writePartialMultiSector()</i> :	beschreibt Teile in zwei Sektoren auf der MicroSD-Karte

Da aber immer ein ganzer Sektor auf die MicroSD-Karte geschrieben werden muss, müssen bei den Funktionen *writePartialSector()* und *writePartialMultiSector()* zuerst die angefangenen Sektoren gelesen werden, um dann zusammen mit den neuen Daten wieder beschrieben werden zu können. Folglich dauert der Schreibvorgang bei den drei Funktionen nicht gleich lang. Bei der Funktion *writeSector()* wird am wenigsten Zeit benötigt, da nur ein Sektor geschrieben werden muss. Nachteil ist, dass Speicherplatz verloren geht, wenn nicht Blöcke der Länge eines Sektors verwendet werden. Die Funktion *writePartialSector()* benötigt doppelt so viel Zeit, da der Sektor jeweils zuerst gelesen und dann geschrieben wird. Die Funktion *writePartialMultiSector()* benötigt nochmals doppelt so lange, also viermal so lange, weil zuerst der erste Sektor gelesen und dann geschrieben wird und dann der zweite.

Die Sektorgrösse kann je nach Speicherkarte variieren. Bei der verwendeten MicroSD-Karte beträgt die Sektorgrösse 512 Byte.

Der Programmcode wurde so angepasst, dass bei jedem Schreibbefehl die Funktion *writeSector()* verwendet wird. Damit nicht jeder eingeleseene Messwert einzeln auf die MicroSD-Karte geschrieben wird, wird ein Buffer mit mehreren Messwerten gefüllt. Ist

dieser Buffer voll, wird er auf die MicroSD-Karte geschrieben. Die Grösse des Buffers ist auf die Sektorgrösse der MicroSD-Karte abgestimmt.

Da immer ein StringBuffer mit der Grösse eines Sektors der MicroSD-Karte geschrieben wird, kann es vorkommen, dass der Schreibvorgang durch den vom Timer ausgelösten Interrupt für das Einlesen neuer Daten unterbrochen wird. Um zu verhindern, dass der Buffer, der gerade auf die MicroSD-Karte geschrieben wird, nicht von neu eingelesenen Messwerten überschrieben wird, ist ein Pingpongbuffer implementiert worden.

7. Software Matlab

7.1. GUI

Zur Visualisierung der gemessenen Beschleunigungswerte und für die Umrechnung in Geschwindigkeitswerte wird Matlab verwendet. Zur benutzerfreundlichen Bedienung wurde ein GUI implementiert. Nachfolgende Abbildung zeigt das GUI der Matlab Software.

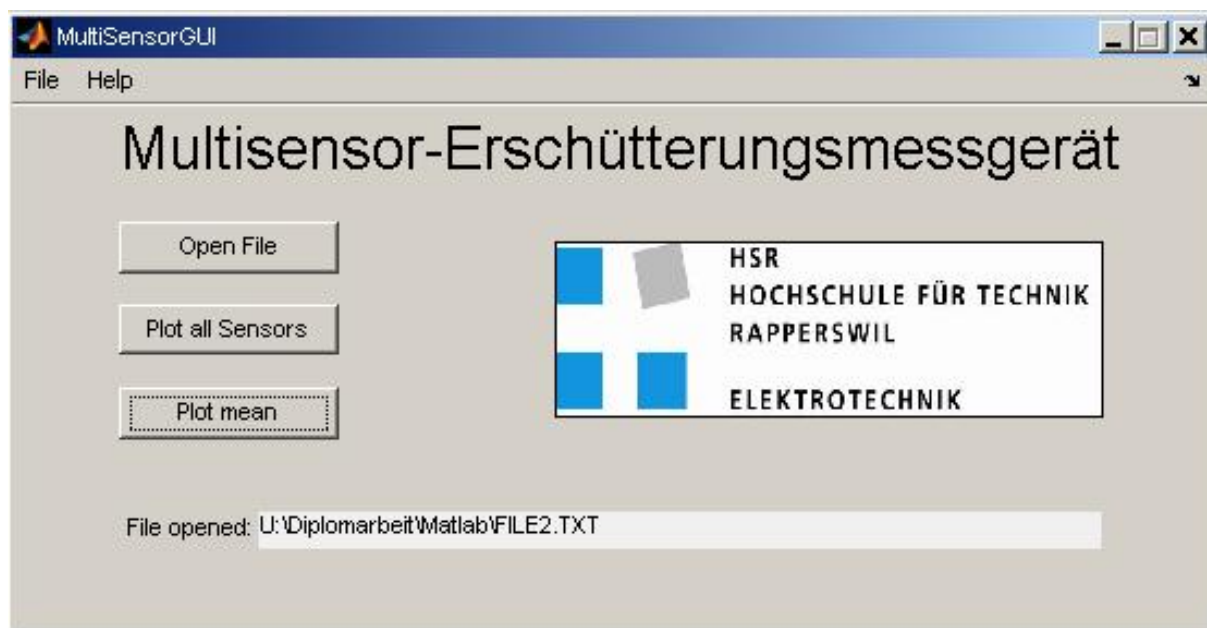


Abbildung 7.14: Matlab GUI

Im Textfeld *File opened:* wird der Pfad des zuletzt geöffneten Files ausgegeben. Die Bedienung der Software erfolgt über drei Buttons. Deren Bedeutung wird kurz erläutert.

Open File	Über ein Pop-up Fenster wird das gewünschte File eingelesen, es werden auch bereits alle für die Darstellung benötigten Variablen berechnet.
Plot all Sensors	Die Daten der einzelnen Sensoren werden unverändert dargestellt.
Plot mean	Es wird das Scharmittel der Sensorendaten und das daraus berechnete Geschwindigkeitssignal dargestellt.

7.2. Variablen

Nach dem Einlesen des Files werden gleich alle Variablen berechnet, welche für das Darstellen der Messung benötigt werden. Die wichtigsten Variablen werden nachfolgend kurz erläutert.

Variable	Typ	Beschreibung
input	Array (<i>Samples · Sensoren</i>)	Enthält die rohen Messwerte aller Sensoren.
Acclg	Array (<i>Samples · Sensoren</i>)	Enthält die Beschleunigungswerte in [g] aller Sensoren
Acclgmean	Array (Anz Samples)	Enthält das Scharmittel der Beschleunigungswerte
Speedmean	Array (Anz Samples)	Enthält die durch integration berechneten Geschwindigkeitswerte.
Time	Array (Anz Samples)	Zeitvektor für den genauen Abstand der einzelnen Samples.

Tabelle 7.7: Matlab Variablen

Variablen, welche in unterschiedlichen M-Files oder in mehreren Funktionen verwendet werden, müssen explizit als global deklariert werden.

7.3. Berechnungen

Da die Daten auf der MicroSD-Karte im ASCII-Format gespeichert wurden, können sie mit der von Matlab generierten Funktion *importfile()* eingelesen werden. Als Rückgabewert der Funktion erhält man einen 2-dimensionalen Array. Die Zeilen des Array's entsprechen der Anzahl Samples, die Spalten entsprechen der Anzahl Sensoren. Um von eingelesenen Beschleunigungssignalen einen allfälligen Offset zu entfernen, wird als erstes der Mittelwert der einzelnen Sensorsignale abgezogen.

```
...
input = input - mean(input); %zieht den offset von den Samples ab.
...
```

Der Sensor liefert 14-Bit Messwerte im Zweierkomplement. Das heisst, der Wertebereich liegt bei $\pm 2^{13} = \pm 8192$. Um von den rohen Messwerten Beschleunigungswerte in [g] zu erhalten, wird durch $8192/2 = 4096$ dividiert. Maximal erreichbare Werte ± 2 [g].

```
...
Acclg = input./4096; %berechne Werte in [g]
...
```

Aus den Messwerten der verschiedenen Sensoren wird das Scharmittel gebildet. Das Scharmittel kann mit der Funktion *mean()* gebildet werden. Da die Matlab-Funktion *mean()* die Mittelwerte der Kolonnen berechnet, aber die Mittelwerte der Zeilen benötigt werden muss das transponierte Array übergeben werden.

```
...  
Acclgmean = mean(Acclg'); %berechnet das Scharmittel der Sensoren  
...
```

Um aus den gemittelten Beschleunigungswerten Geschwindigkeitswerte zu erhalten, wird integriert. Das Integrieren wird mit der Trapezmethode durchgeführt (lineare Interpolation). Dies entspricht dem einfachen IIR-Filter

$$H(z) = \frac{T}{2} \cdot \frac{1 + z^{-1}}{1 - z^{-1}}$$

mit den A-Koeffizienten [1 -1] und den B-Koeffizienten [1 1]. Es könnte auch mit der Rechteckmethode integriert werden, aber dann wäre der Fehler grösser.

```
...  
% integrieren mit der Trapezmethode (IIR-Filter)  
T = 1/Samplerate; %Sampleabstand < 1Sek.  
for n = 2 : length(input)  
    Speedmean(n) = Speedmean(n-1)+(T/2)*(Acclmms2mean(n) + Acclmms2mean(n-1));  
end %for  
...
```

7.4. Ausgabe

Bei Betätigung des Buttons *Plot all Sensors* öffnet sich ein neues Fenster, wie in Abbildung 7.15 dargestellt, in dem die Beschleunigungswerte der einzelnen Sensoren visualisiert werden.

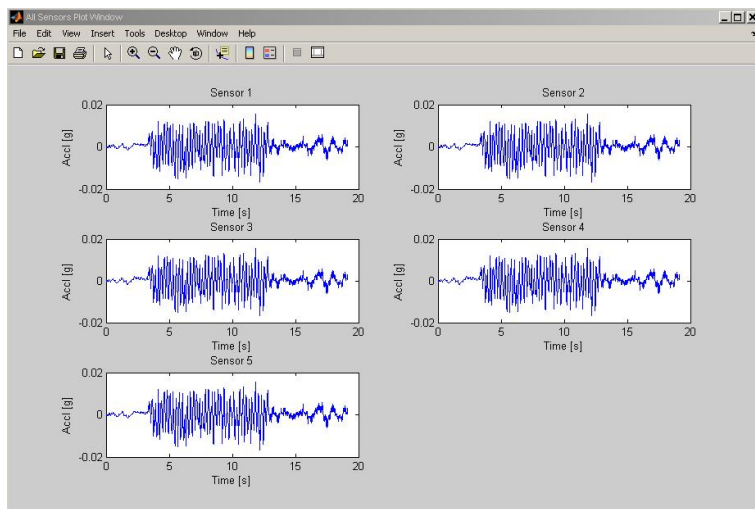


Abbildung 7.15: Ausgabe Beschleunigungswerte aller Sensoren

Wird der Button *Plot mean* betätigt, öffnet sich ebenfalls ein neues Fenster, wie in Abbildung 7.16 dargestellt, in dem die aus allen Sensoren gemittelten Beschleunigungswerte und die daraus berechneten Geschwindigkeitswerte visualisiert werden.

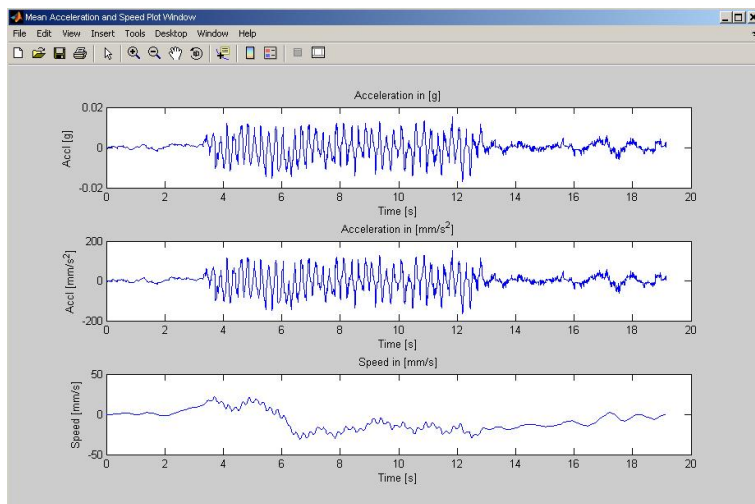


Abbildung 7.16: Ausgabe gemittelte Beschleunigung und Geschwindigkeit

8. Verwendete Software Werkzeuge

8.1. IAR Embedded Workbench

IAR Embedded Workbench ist eine integrierte Entwicklungsumgebung für verschiedenste Microcontrollerarchitekturen. Nebst Compiler ist auch ein Debugginginterface integriert, das es erlaubt, den verwendeten Microcontroller während des Betriebes zu kontrollieren und dessen Speicher zu beeinflussen.

Auf der Internetseite der *IAR Systems AG* [7] kann eine 30 Tage Evaluationssoftware kostenlos heruntergeladen werden.

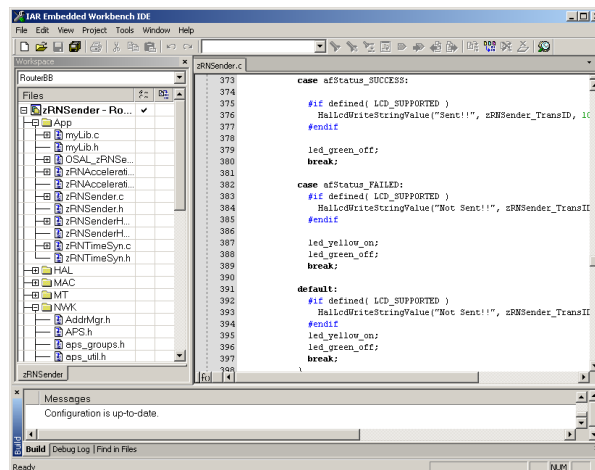


Abbildung 8.17: IAR Embedded Workbench

8.2. Matlab

Matlab ist ein Produkt des Unternehmens *The MathWorks, Inc.* Es eignet sich hervorragend für mathematische Berechnungen sowie die Visualisierung von Messdaten. Mit der Toolbox *guide* lassen sich damit auch grafische Benutzerschnittstellen (GUI) sehr einfach realisieren.

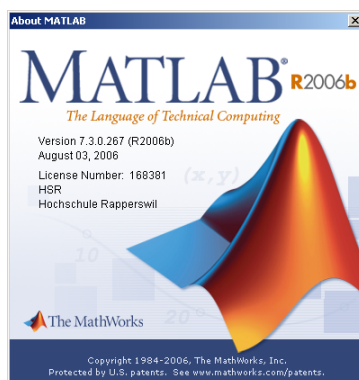


Abbildung 8.18: Matlab

8.3. OrCAD

OrCAD ist ein Konstruktionssoftware um Leiterplatten zu entwickeln. Sie beinhaltet einen Schemaeditor, sowie ein Layout Programm. OrCAD wurde dazu verwendet, um das Beschleunigungssensorboard zu konstruieren.



Abbildung 8.19: OrCAD

8.4. LateX

Diese Dokumentation wurde mit \LaTeX geschrieben. Als Texteditor wurde dazu TeXnic-Center verwendet.

9. Zusammenfassung

In dieser Diplomarbeit wurde ein Messsystem entwickelt, welches es ermöglicht, Erschütterungen zu visualisieren.

In der durchgeführten Arbeit ist ein Multisensor Erschütterungsmessgerät entwickelt worden, in das bis zu 9 MEMS-Beschleunigungssensoren eingebaut werden können. Durch eine raffinierte Implementierung der Messsoftware werden über eine SPI-Schnittstelle von allen Sensoren Messwerte vom selben Zeitpunkt eingelesen.

Damit kontinuierlich Messwerte eingelesen werden und dies nicht durch die benötigte Zeit für das Schreiben auf die Speicherkarte unterbrochen wird, wurden mit einem Timerinterrupt die Messzeitpunkte festgelegt. Es werden immer eine bestimmte Anzahl Messwerte eingelesen, bevor diese auf die MicroSD-Karte gespeichert werden. So wird die Anzahl Schreibvorgänge niedrig gehalten.

Um eine geeignete Grösse der Hardware zu erreichen, wurde eine eigene Leiterplatte entwickelt, welche auf beiden Seiten bestückt wurde.

Damit ein genügend grosser Speicher für die vielen Messdaten vorhanden ist, wurde auf der Hardware ein MicroSD-Kartesteckplatz integriert. Die Messdaten werden während der Messung laufend im ASCII-Format auf der MicroSD-Karte gespeichert. Damit die Messdaten von einem PC aus wieder eingelesen werden können, wurde für die MicroSD-Karte das Dateisystem FAT16 verwendet. Für das Schreiben auf die mit FAT16 formatierte MicroSD-Karte wurde eine bereits vorhandene FAT-Library eingebunden, welche bereits alle benötigten Funktionen beinhaltet.

Die Visualisierung der gemessenen Daten erfolgt auf einem PC mittels einer Matlab Software. Um die Bedienung dieser Software angenehm zu gestalten, wurde eine grafische Benutzerschnittstelle erstellt. Damit können die Daten eingelesen und dargestellt werden. Es werden jeweils die Beschleunigungssignale der einzelnen Sensoren sowie das gemittelte Signal aller Sensoren und das daraus berechnete Geschwindigkeitssignal dargestellt. Die aufgezeichneten Messdaten können zu jeder Zeit erneut eingelesen und dargestellt werden.

Da die Sensoren, welche auf das selbst entwickelte PCB gelötet wurden defekt sind, konnte keine Messung mit mehreren Sensoren durchgeführt werden. Es konnte aber ausgeschlossen werden, dass sich das Problem in der Software oder auf der entwickelten Hardware befindet. Mit einem Sample-Sensor mit Evaluationboard funktioniert das Messgerät.

10. Ausblick

In einer 8-wöchigen Diplomarbeit ist es nicht möglich, ein Produkt ohne Verbesserungsmöglichkeiten zu entwickeln. Dafür ist die zur Verfügung stehende Zeit viel zu kurz. Deshalb sind nachfolgend die interessantesten Optimierungspotentiale kurz dargestellt.

- Vernetzung von mehreren Geräten via Funk: Die entwickelte Hardware wird momentan als Einzelstück gebraucht. Das verwendete Evaluation Modul CC2430EM besitzt aber ein integriertes Funkmodul. So könnten mehrere Messgeräte vernetzt werden und die Daten via Funk an eine Basisstation gesendet werden.
- Miniaturisierung: Ein weiterer interessanter Schritt besteht in der Miniaturisierung der Messgerätes. Anstatt das Evaluationsboard CC2430EM von Chipcon zu verwenden und durch Optimierung des PCB Layouts, könnte man die Hardware noch weiter verkleinern.
- Winkelmessung: Die MEMS Sensoren von Analog Devices sind in der Lage auch Winkel zu messen. Das Messen der Winkel wurde in der entwickelten Software nicht implementiert. Es könnte durch Messen der Winkel und somit der Lage des Messgerätes erreicht werden, dass das Messgerät nicht in der Waagerechten ausgerichtet werden muss.
- Stromverbrauch: Der Stromverbrauch der MicroSD-Karte und der Sensoren zusammen ist ziemlich hoch. Zwei AAA-Batterien sind für eine angemessene Messdauer etwas zu klein. In einer weiterführenden Arbeit könnte der Stromverbrauch noch optimiert werden, z.B. könnte nur ein Sensor betrieben und auch keine Daten auf die MicroSD Karte geschrieben werden, so lange keine Erschütterungen gemessen werden. Alle anderen Sensoren können während dieser Zeit in einen Sleep-Mode gesetzt werden. Erst wenn der eine Sensor eine Erschütterung detektiert werden die anderen Sensoren Eingeschaltet und die Daten auf die MicroSD-Karte geschrieben.

11. Schlusswort

Zum Schluss möchte ich noch meinen Eindruck zur Arbeit einbringen. Diese hat mir viele nützliche Erfahrungen eingebracht und hat mir viel Freude bereitet, obwohl nicht immer alles von Anfang an funktioniert hat.

Allen Beteiligten, die mich während der Arbeit unterstützt haben, möchten ich meinen Dank aussprechen. Den Laborassistenten, Herr Matthias Engelhardt, Herr Ursin Tuor, Herr Reto Flütsch, Herr Philipp Derksen und Herr Armin Gujan danke ich für ihre Bereitschaft, mir bei Problemen zu helfen. Für das maschinelle Löten der difisilen Beschleunigungssensoren auf meine Leiterplatte möchte ich mich an dieser Stelle bei Herrn Casper Naef bedanken. Für die Beschaffung von Material und Software danke ich Herrn Peter Roffler. Nicht zuletzt möchte ich mich noch bei meinem Professor Herrn Guido Schuster bedanken, der mir diese Arbeit ermöglicht hat. Von seinem Wissen konnte ich stets profitieren.

Rapperswil, den 25.04.2008
Remo Fehr

A. Schemata

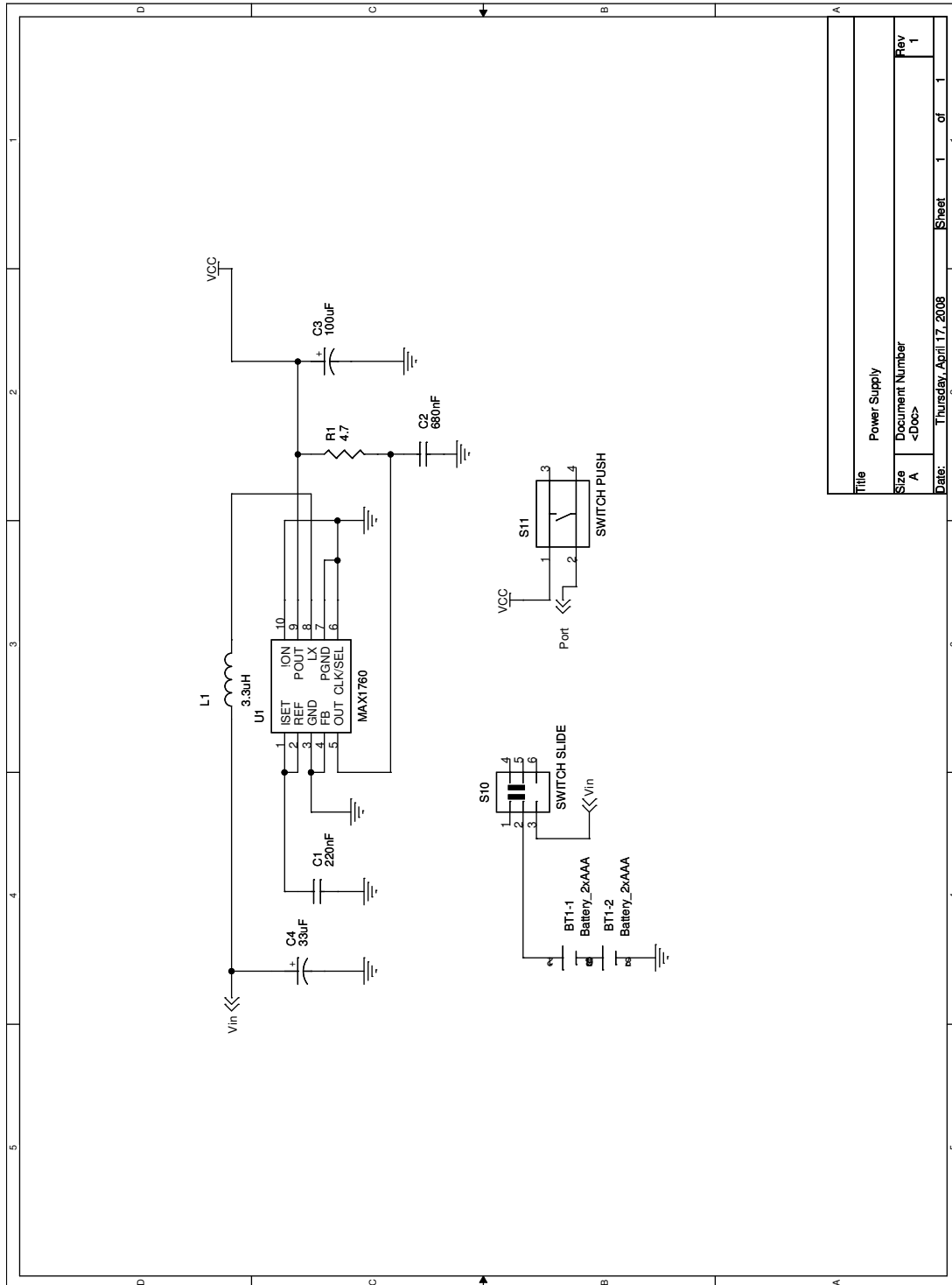


Abbildung 1.20: Schema Power Supply

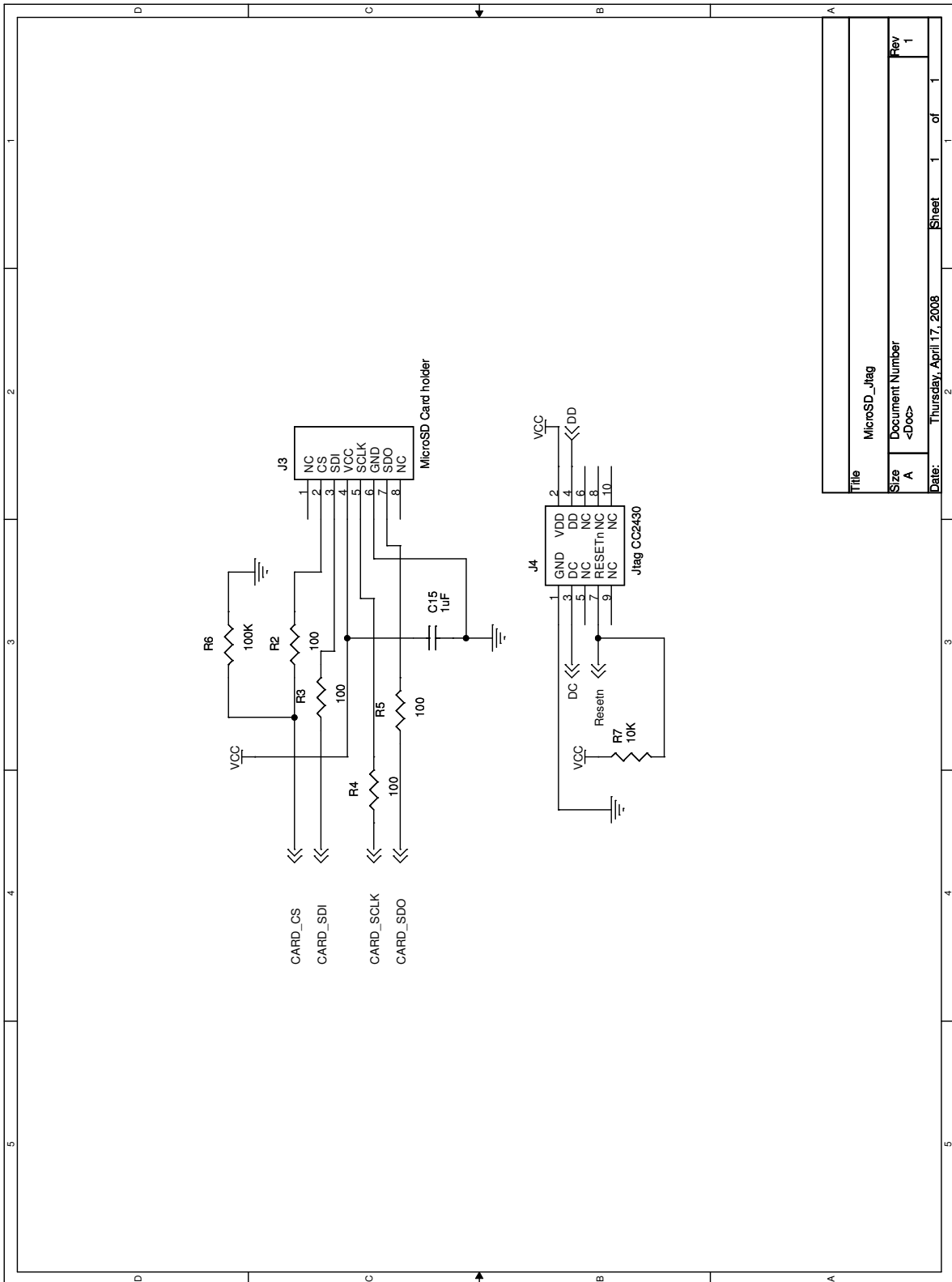


Abbildung 1.21: Schema MicroSD und CC2430 Debug Control

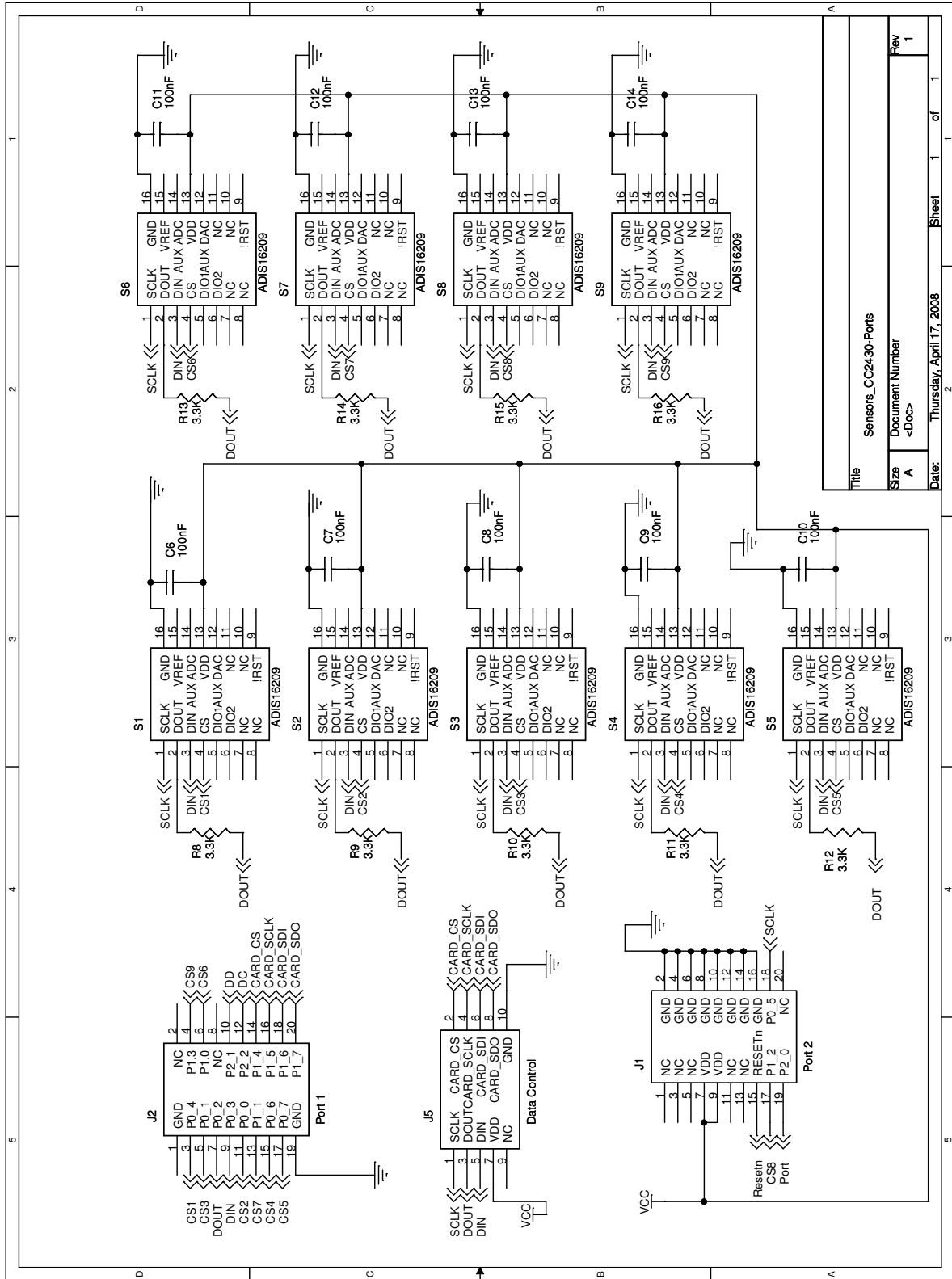


Abbildung 1.22: Schema Sensors und CC2430-Ports

B. Stückliste

Position	Wert	Beschreibung	Vertreiber	Bestellnr.
BT1		SMD Batteriehalter	www.farnell.ch	
C1	220nF	Kondensator	www.farnell.ch	431199
C2	680nF	Kondensator	www.farnell.ch	1414702
C3	100 μ F	Kondensator	www.distrelec.ch	810814
C4	33 μ F	Kondensator	www.distrelec.ch	811028
C6-C14	100nF	Kondensator	www.farnell.ch	8820023
C15	1 μ F	Kondensator	www.farnell.ch	9527699
J1,J2	2x10	CC2430 Port1/2	www.farnell.ch	1106323
J3		MicroSD Card holder	www.farnell.ch	1366700
J4,J5	2x5	Stiftleiste	www.farnell.ch	1022227
L1	3.3 μ H	geschirmte Spule	www.farnell.ch	1198591
R1	4.7 Ω	Widerstand	www.farnell.ch	9331280
R2-R5	100 Ω	Widerstand	www.farnell.ch	9330364
R6	100k Ω	Widerstand	www.farnell.ch	9330402
R7	10k Ω	Widerstand	www.farnell.ch	9330399
R8-R16	3.3k Ω	Widerstand	www.farnell.ch	9331026
S1-S9		Sensor ADIS16209	www.analog.com	ADIS16209CCCZ
S10		ON-ON Switch 0.2A	www.farnell.ch	1218896
S11		Taster, SMD	www.farnell.ch	9962905
U1	3.3V 0.8A	Step-up MAX1760	www.farnell.ch	1422291

Tabelle 2.8: Stückliste PCB

C. Abkürzungsverzeichnis

<i>GUI</i>	Graphical User Interface
<i>HW</i>	Hardware
<i>ISR</i>	Interrupt Service Routine
<i>LSB</i>	Least Significant Bit
<i>MEMS</i>	Micro-Electro-Mechanical Systems
<i>PC</i>	Personal Computer
<i>PCB</i>	Printed Circuit Board
<i>SDI</i>	Slave-Data-In
<i>SDO</i>	Slave-Data-Out
<i>SoC</i>	System on Chip (CC2430)
<i>SPI</i>	Serial Peripheral Interface
<i>SW</i>	Software
<i>TI</i>	Texas Instruments
<i>UART</i>	Universal Asynchronous Receiver Transmitter

D. Abbildungsverzeichnis

3.1. Aufgabenstellung Seite 1	3
3.2. Aufgabenstellung Seite 2	4
5.3. Messhardware	8
5.4. Step-up Wandler[2]	8
5.5. CC2430EM	9
5.6. ADIS16209[3]	9
5.7. Anschlüsse Beschleunigungssensor	10
5.8. Schema MicroSD-Kartensteckplatz	11
5.9. PCB Layout	12
5.10. Bestückungsplan	12
6.11. Flussdiagramm	13
6.12. 16-Bit Sensor-Befehle	14
6.13. Messwert	14
7.14. Matlab GUI	19
7.15. Ausgabe Beschleunigungswerte aller Sensoren	22
7.16. Ausgabe gemittelte Beschleunigung und Geschwindigkeit	22
8.17. IAR Embedded Workbench	23
8.18. Matlab	24
8.19. OrCAD	24
1.20. Schema Power Supply	28
1.21. Schema MicroSD und CC2430 Debug Control	29
1.22. Schema Sensors und CC2430-Ports	30

E. Literatur

- [1] File Allocation Table Wikipedia
http://de.wikipedia.org/wiki/File_Allocation_Table
- [2] Datenblatt des Step-up-Wandlers MAX1760 von *MAXIM*,
Seite 7, Bild 2, Standard Application Circuit
- [3] Datenblatt des Beschleunigungssensors ADIS16209 von
Analog Devices, Seite 1, Bild 1, Functional Blockdiagramm
- [4] Datenblatt des Beschleunigungssensors ADIS16209 von
Analog Devices, Seite 11, Tabelle 6, User Register Map
- [5] Datenblatt des Beschleunigungssensors ADIS16209 von
Analog Devices, Seite 12, Tabelle 8, SMPL_PRD Bit Description
- [6] Datenblatt des Beschleunigungssensors ADIS16209 von
Analog Devices, Seite 5, Tabelle 2, Timing Specifications
- [7] IAR Systems AG <http://www.iar.se>