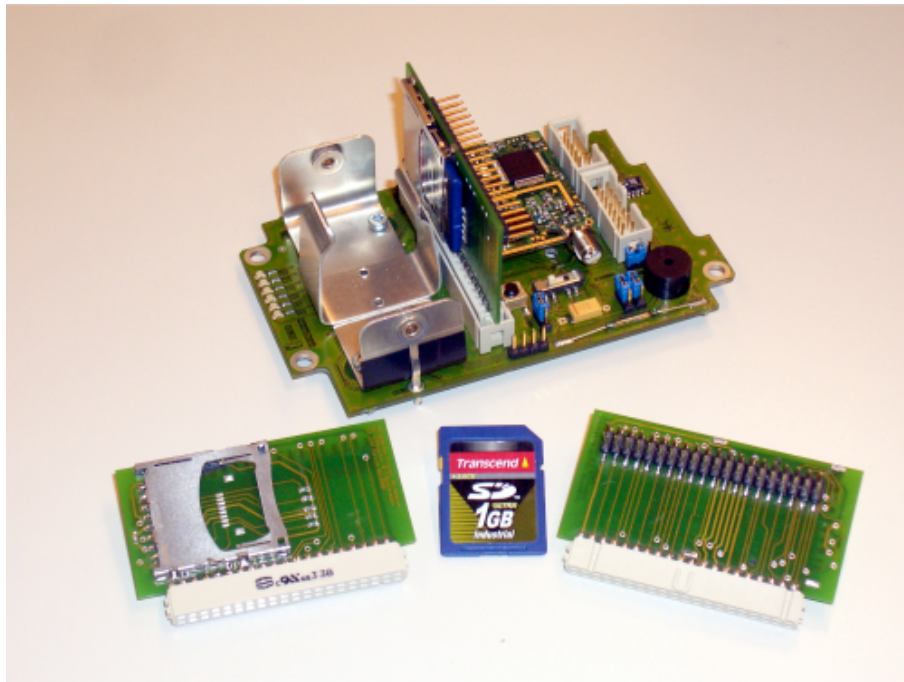


# PermaSense Persistent Storage Solution



## Gruppenarbeit

Christoph Guntermann & Josua Hunziker

4. Juli 2008

Betreut durch Dr. Jan Beutel und Roman Lim  
Prof. Dr. Lothar Thiele

Institut für Technische Informatik und Kommunikationsnetze  
Departement für Informationstechnologie und Elektrotechnik  
ETH Zürich



# Inhalt

<b>1: Systemüberblick</b>	<b>1</b>
1.1 Systemarchitektur ohne PermaStore . . . . .	1
1.2 Der Ansatz von PermaStore . . . . .	2
<b>2: SPI - Serial Peripheral Interface</b>	<b>3</b>
2.1 Allgemein . . . . .	3
2.2 Daten- und Steuerleitungen . . . . .	3
<b>3: Speichermedium</b>	<b>5</b>
3.1 Allgemein . . . . .	5
3.2 SD-Karte: Variationen . . . . .	5
3.3 Pinbelegung . . . . .	6
<b>4: Bau der Prototypen</b>	<b>7</b>
4.1 Pull-up Widerstände . . . . .	7
<b>5: Messungen der SD-Karten</b>	<b>11</b>
5.1 Aufbau der Messung . . . . .	11
5.2 Quellcode . . . . .	11
5.3 Resultate . . . . .	12
<b>6: Hardware Design</b>	<b>17</b>
6.1 Footprints . . . . .	17
6.2 Schema . . . . .	17
6.3 Printplattendesign . . . . .	17
<b>7: Software</b>	<b>23</b>
7.1 Die PermaStore-Komponente . . . . .	23
7.1.1 Grundsätzliche Softwarearchitektur . . . . .	23
7.1.2 Die Schichten im Detail . . . . .	24
7.2 Applikationen . . . . .	27
7.2.1 Die TestSD-Applikation . . . . .	27
7.2.2 Die FormatSDCard-Applikation . . . . .	29

## *Inhalt*

---

<i>A: Messungen mit dem SIB</i>	<i>31</i>
<i>B: Patchen eines Tinynodes</i>	<i>33</i>
<i>C: Liste der verwendeten Teile</i>	<i>35</i>
<i>D: Liste der verwendeten SD-Cards</i>	<i>37</i>
<i>E: Konfiguration von TinyOS und Tinynode unter Ubuntu 8.04</i>	<i>39</i>
E.1 TinyOS Basisinstallation . . . . .	39
E.2 Permasense-Installation . . . . .	40
E.3 Abschluss der Installation . . . . .	40
<i>F: CD-Inhalt</i>	<i>43</i>

# Abbildungen

1-1	Systemarchitektur . . . . .	1
1-2	Erweiterte Systemarchitektur . . . . .	2
2-1	Grundzüge des SPI . . . . .	3
2-2	Parallelschaltung der Slaves . . . . .	4
2-3	Serieschaltung der Slaves . . . . .	4
3-1	Pinbelegung der verschiedenen Karten . . . . .	6
3-2	Micro-SD-Karte . . . . .	6
3-3	SD-Karte . . . . .	6
4-1	Schaltungsaufbau: Computer - Tinynode - Prototyp . . . . .	8
4-2	Prototyp für SD-Karten . . . . .	9
4-3	Prototyp für MicroSD-Karten . . . . .	9
5-1	PowerMessungen mit Agilent Technologies DC Power Analyzer . . . . .	13
5-2	Ergebnisse der Powermessungen . . . . .	14
5-3	Messungen auf dem Extensionboard . . . . .	14
5-4	Messungen auf dem Sensor Interface Board . . . . .	15
5-5	Zeitmessungen . . . . .	15
6-1	Doppelter SD-Halter . . . . .	17
6-2	Buchsenleiste . . . . .	17
6-3	Schaltungsaufbau: Print für SD-Karte . . . . .	18
6-4	Horizontale Montage, SD . . . . .	19
6-5	Horizontale Montage, MicroSD . . . . .	19
6-6	Vertikale Montage, SD . . . . .	19
6-7	Vertikale Montage, MicroSD . . . . .	19
6-8	Der endgültige PCB-Print . . . . .	20
6-9	Print-Debugger . . . . .	21
6-10	Das fertige Erweiterungsboard . . . . .	21
7-1	PermaStore-Softwareschichtung . . . . .	24

## Abbildungen

---

7-2	Block-Test Prinzip . . . . .	28
7-3	Jump-Test Prinzip . . . . .	28
A-1	Gesetzte Pins im SDTestM.nc . . . . .	31
A-2	Veränderung im hardware.h-Headerfile . . . . .	31
B-1	Patch Schema . . . . .	33
B-2	Patch Foto . . . . .	33
C-1	Liste mit Hersteller und Produkt-ID . . . . .	35
D-1	Verwendete SD-Karten . . . . .	37

# *Einleitung*

Das PermaSense Projekt hat zum Ziel in hochalpinen Regionen mittels eines drahtlosen Sensornetzes Umweltdaten zu messen. Die hierfür notwendige Infrastruktur besteht aus einem Sensornetzwerk auf Basis von Shockfish TinyNodes [7] (MSP 430 Microcontroller [5]) und TinyOS, einer Basisstation sowie einem Datenbackend. Als Erweiterung dieses Systems sollte nun ein Backup Speichersystem entwickelt werden um grössere Mengen von Daten lokal auf jedem Datenlogger Node speichern zu können.

Dieser Idee liegen zwei Motive zu Grunde:

- Bei Verlust der Datenübertragung (z.B. durch Schneefall oder Ausfall eines Relay-Knotens) können die Daten vorübergehend lokal gespeichert und bei Wiedererlangung der Konnektivität zur Basestation übertragen werden.
- Die gespeicherten Daten können nach Demontage der Sensoren am Berg zur Validierung der empfangenen Daten verwendet werden, um mögliche Übertragungsfehler zu auszuschliessen.

TinyNode verfügt über einen SPI-Bus (Serial Peripheral Interface Bus [3]), welcher zur Kommunikation mit externen Speichern verwendet werden kann. Die Idee war, SD-Karten (Secure Digital Memory Cards [2]) als Speichermedium zu verwenden, da diese Speicherkapazitäten von bis zu 1GB zu mittlerweile sehr moderaten Preisen bieten. Zudem bringen sie durch ihre Auswechselbarkeit hohe Flexibilität und können ohne zusätzliche Controller im SPI-Modus betrieben werden.

Dieser Bericht dokumentiert unsere Schritte auf dem Weg zu einem lauffähigen Speichersystem bestehend aus Erweiterungsboard für das in PermaSense verwendete SIB (Sensor Interface Board [8]) sowie passenden Softwarekomponenten, um den Flashspeicher nahtlos in die Systemsoftware einbinden zu können. Dabei mussten folgende Teilaufgaben gelöst werden:

- Einarbeitung in TinyOS [6] und die bestehende Hardware
- Entwurf eines ersten funktionalen Hardware-Prototypen
- Evaluierung einer geeigneten Speicherkarte
- Grundsätzliche Softwareansteuerung der SD-Karte
- Entwurf und Herstellung einer Erweiterungskarte zum SIB
- Integration der erstellten Softwarekomponenten in die bestehende Applikation

- Test von Hard- und Software

Unsere Überlegungen zu all diesen Punkten finden sich in den entsprechenden Kapiteln dieser Dokumentation. Zudem sind in den Anhängen detailliertere technische Informationen enthalten. Weiter enthält dieser Bericht eine CD mit allen Quellcodes, Datenblättern und einigen nützlichen Skripts (Anh. F).

Die Arbeit hat uns grossen Spass gemacht und wir konnten viele neue Erfahrungen und Erkenntnisse sammeln. Wir hoffen, damit einen entscheidenden Beitrag zum Gelingen des PermaSense Projekts beigetragen zu haben.



# 1

## Systemüberblick

### 1.1 Systemarchitektur ohne PermaStore

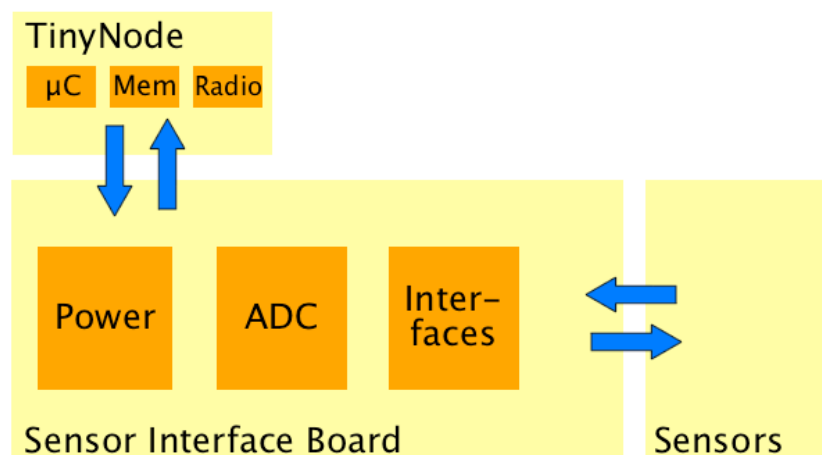


Abbildung 1-1  
Systemarchitektur

Jeder Datenlogger Node besteht aus drei Hauptkomponenten:

- TinyNode: Eine Microcontroller-Plattform entwickelt an der EPFL Lausanne. Er ist bestückt mit einem MSP430-Mikrocontroller von Texas Instruments [5], einem Flash-Memory von 520kByte Grösse, sowie einem Low-Power-Radio (Xemics XE1205) zur Kommunikation im Netzwerk [7].
- Sensor: An jeden Datenlogger Noder können verschiedene Sensoren angeschlossen werden, darunter Temperatur-, Feuchtigkeits-, Druck- und Distanzsensoren.

- Sensor Interface Board (SIB, [8]): Das SIB stellt Basisservices wie Stromversorgung, Analog-Digitalwandlung und Kommunikationsschnittstellen zur Verfügung. Es dient somit als Verbindung zwischen dem TinyNode und dem Sensor.

## 1.2 Der Ansatz von PermaStore

Die entwickelte Speicherlösung wird physisch zwischen dem SIB und den Sensor platziert. Dabei wird die bestehende Verbindung nicht unterbrochen; vielmehr greift das entwickelte Board die für den Betrieb der SD-Karte benötigten Signale einfach ab. So können zumindest hardwareseitig Interferenzen mit anderen Systemkomponenten oder Blockierung derselben weitestgehend ausgeschlossen werden.

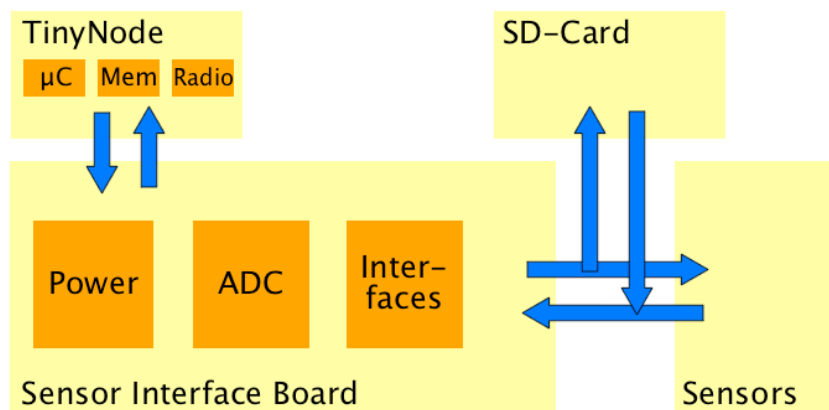


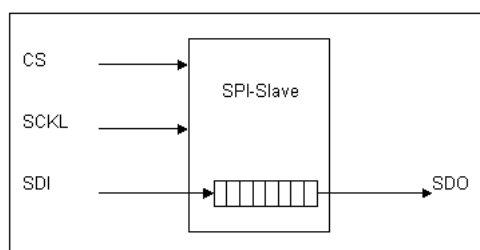
Abbildung 1-2  
Erweiterte Systemarchitektur

# 2

## *SPI - Serial Peripheral Interface*

### *2.1 Allgemein*

Das Serial Peripheral Interface [12] wurde von Motorola entwickelt und ist ein synchrones serielles Bussystem, das auf dem Master-Slave-Prinzip basiert. Der SPI-Bus ist auf zwei Steuerleitungen (CS, SCLK) und zwei Datenleitungen (SDI, SDO) aufgebaut (Abb. 2-1). Der Microcontroller, welcher sich im Mastermodus befindet, gibt die ChipSelect-Leitung vor und bestimmt die Taktfrequenz. Der Slave auf der anderen Seite erhält diese Steuersignale vom Master als Eingänge. Für einen SPI-Bus gibt es immer einen einzigen Master.



*Abbildung 2-1  
Grundzüge des SPI*

Der Microcontroller kann über den SPI-Bus auf mehrere Geräte zugreifen. Entweder werden diese Slaves durch verschiedene ChipSelect-Leitungen ausgewählt (Abb. 2-2) oder die Slaves sind kaskadiert und stellen für den Master einen einzigen Slave dar (Abb. 2-3).

### *2.2 Daten- und Steuerleitungen*

Die ChipSelect Leitung ist im inaktiven Zustand High. Wählt der Master ein Gerät aus, so wird der zugehörige ChipSelect auf Low gesetzt (active-low). Die Datenleitung befinden sich, während der Slave nicht selektiert ist, in einem hochohmigen Zustand. Man verwendet für sie auch die Namen MOSI (Master Out Slave

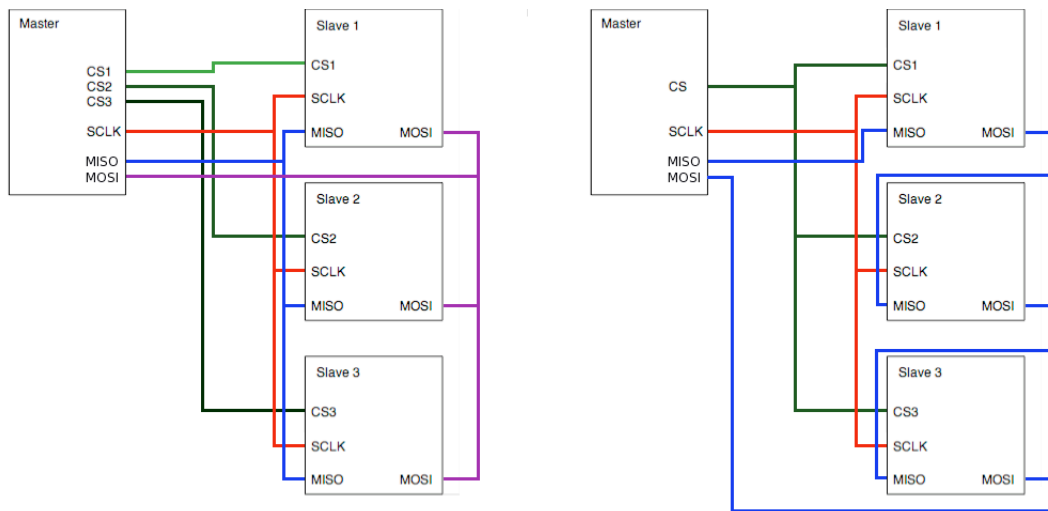


Abbildung 2-2 Parallelschaltung der Slaves    Abbildung 2-3 Serieschaltung der Slaves

In) und MISO (Master In Slave Out). Das SPI-Protokoll wurde von Motorola nicht genau festgelegt. Grundsätzlich haben sich für die SCLK-Leitung vier Modi verbreitet. Diese Modi werden durch die beiden Parameter Clock Polarität (CPOL) und Clock Phase (CPHA) festgelegt, womit man den Taktzeitpunkt und den Ein-Ausgabemodus mit der High oder Low-Flanke angeben kann [4].

- CPOL (Clock Polarity)
  - 0: Takt ist in Ruhe LOW, ein Wechsel auf HIGH zählt als steigende Taktflanke
  - 1: Takt ist invertiert: in Ruhe HIGH, ein Wechsel auf LOW zählt als steigende Taktflanke
- CPHA (Clock Phase)
  - 0: Daten werden bei steigender Taktflanke (abh. von CPOL) eingelesen, bei fallender ausgegeben
  - 1: Daten werden bei fallender Taktflanke eingelesen, bei steigender ausgegeben

# 3

## *Speichermedium*

### *3.1 Allgemein*

Eine SD-Karte (Secure Digital-Card) ist ein Flash-Speicher, welcher auf dem Format der älteren MMC-Karte basiert. Dabei sind verschiedene Übertragungsgeschwindigkeiten möglich. Die SD-Karte wurde als Speichermedium gewählt, weil sie einerseits über einen grossen Speicherplatz verfügt und andererseits sowohl SD als auch SPI als Übertragungsprotokoll unterstützt. Viele Microcontroller-Plattformen unterstützen einen Datentransfer über SPI, so auch der Tynode.

Die SD-Karte verfügt über drei Transfermodi:

- SPI Modus (einzelner serieller Eingang und Ausgang)
- 1-Bit SD Modus (je ein einzelner Steuer- und Datenkanal)
- 4-Bit SD Modus (benutzt drei zusätzliche Datenkanäle)

Die SD-Karten sind für eine Speisespannung zwischen 2.7V und 3.6V ausgelegt. Sie reagieren jedoch sehr empfindlich auf Spannungsrippel von über 60mV. Um diese Rippen abzdämpfen ist es sinnvoll, in einer Schaltung mit SD-Karte zwischen VDD und GND eine Kapazität von etwa 100nF einzusetzen.

### *3.2 SD-Karte: Variationen*

Es gibt 3 Varianten der SD-Karte: Die SD-, die MiniSD- und die MicroSD-Karte. Die MicroSD-Karte unterscheidet sich von der SD-Karte dadurch, dass sie nur acht anstatt neun Pins besitzt (auf den zweiten Ground-Pin wurde verzichtet). Die elektrische Spezifikation bleibt jedoch dieselbe. Auf die MiniSD-Karte wird im vorliegenden Bericht nicht weiter eingegangen da sie für dieses Projekt ohne Bedeutung war.

### 3.3 Pinbelegung

Neben Ground und VDD spielen vier weitere Pins an der SD-Karte eine wichtige Rolle für den SPI-Datentransfer:

- CS / SS :            Chip Select
- MOSI oder SDO:    Master Out Slave In oder Serial Data Out
- MISO oder SDI:     Master In Slave Out oder Serial Data In
- SCLK:                Serial Clock

Die anderen Pins sind nicht mit dem Microcontroller verbunden.

Micro SD-Karte		SD-Karte	
1	Dat3 / SS / CS	1	Dat3 / SS / CS
2	CMD / MOSI	2	CMD / MOSI
3	Vcc	3	Vss1 (GND)
4	CLK / SCLK	4	Vcc
5	Vss (GND)	5	CLK / SCLK
6	DAT0 / MISO	6	Vss2 (GND)
7	DAT1 / NC	7	DAT0 / MISO
8	DAT2 / NC	8	DAT1 / NC
		9	DAT2 / NC

Abbildung 3-1  
Pinbelegung der verschiedenen Karten

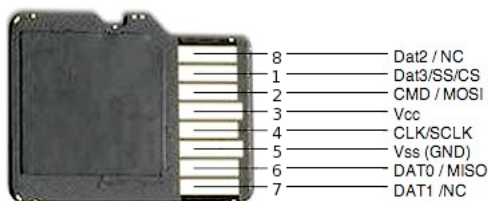


Abbildung 3-2 Micro-SD-Karte

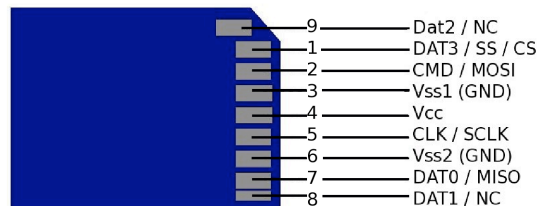


Abbildung 3-3 SD-Karte

Die Messungen (Kap. 5) haben später gezeigt, dass der Stromverbrauch einer MicroSD-Karte bei Schreib- und Leseoperationen beträchtlich geringer ist als bei SD-Karten. Da jedoch noch keine MicroSD-Karten für den Temperaturbereich von bis zu -40°C erhältlich sind, muss vorerst auf SD-Karten ausgewichen werden.

# 4

## *Bau der Prototypen*

Als Grundlage zur Softwareentwicklung und zur Überprüfung der gesammelten Erkenntnisse wurden zwei funktionale Prototypen entworfen. Auf dieser Basis konnten danach Soft- und Hardware parallel miteinander entwickelt werden. Da das Shockfish Extensionboard [7] schon über SPI-Schnittstellen vom Microcontroller MSP430 verfügt, mussten nur die Pins zwischen SD-Karte und Board miteinander verbunden werden.

Im Schaltplan (Abb. 4-1) sieht man neben den Verbindungsstellen des MSP430 und einer SD-Karte auch Pull-up Widerstände und eine Kapazität. Um Floating-Zustände zu vermeiden werden alle Pins bis auf GND, VDD und SCLK mit einem Pull-up Widerstand versehen. Bei den Prototypen wurde auf die Kapazität noch verzichtet, weil die Messungen am VDD Pin des Extensionboards eine sehr glatte Gleichspannung zeigten. Da die Pinbelegung einer SD-Karte und einer Micro-SD-Karte verschieden sind, erstellten wir zwei Prototypen für die beiden Varianten. Beim Belöten des Micro-SD Prototypen war das Raster der Lötbahnen so klein, dass eine ungebrauchte BTNode-Printplatte zweckentfremdet werden musste, um den Sockel anlöten zu können.

### *4.1 Pull-up Widerstände*

Die Pull-up Widerstände dienen dazu, besser zwischen High- und Low-Zustand besser zu unterscheiden. Somit ist die Datenübertragung gegen von aussen kommende Störungen - wie z.B. hohe Ströme - besser geschützt. Hinzu kommt, dass während eines Controller-Resets (ISP) die ChipSelect-Leitungen seitens Controller offen liegen. Ohne Pull-up Widerstände können sich SPI-Slaves aktivieren da hier active-low gilt, was zu unvorhersehbarem Verhalten führen kann.

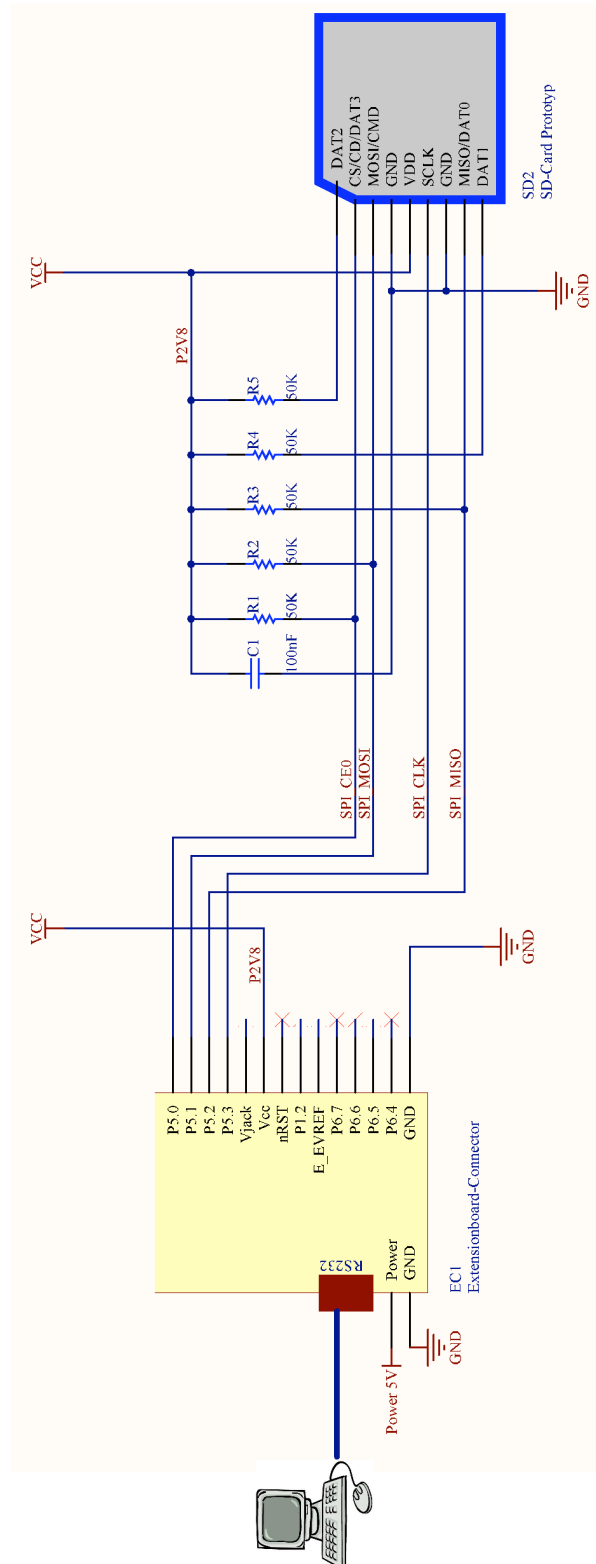


Abbildung 4-1  
Schaltungsaufbau: Computer - Tinynode - Prototyp



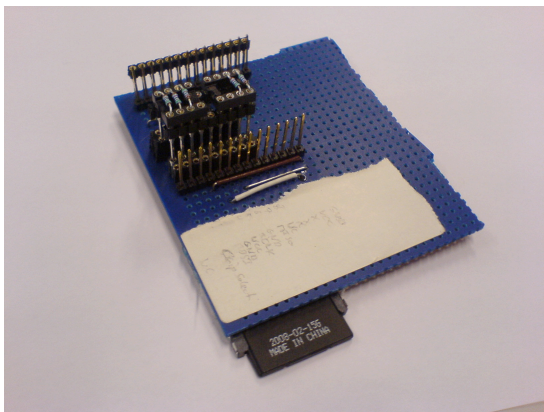


Abbildung 4-2 Prototyp für SD-Karten

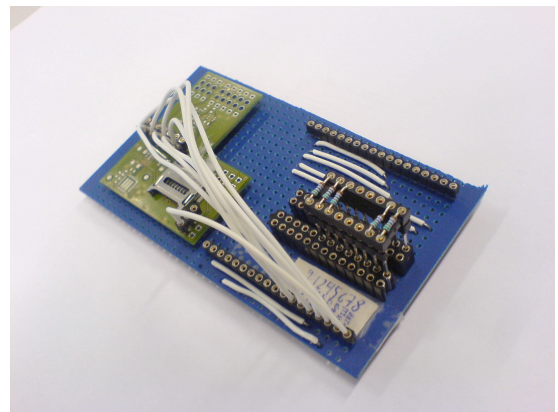


Abbildung 4-3 Prototyp für MicroSD-Karten



# 5

## Messungen der SD-Karten

### 5.1 Aufbau der Messung

Für die Messung des Stromverbrauchs wurde der 'Agilent Technologies DC Power Analyzer' [1] zur Versorgung am Extensionboard angehängt. Der Power Analyzer kann jedoch den Strom nur messen, wenn er als Spannungsquelle verwendet wird. Da das Extensionboard über eine eigene Voltage-Regulation verfügt, wurde das Extensionboard über die VCC und GND Pins gespiesen. Um keinen unnötigen Strom über den Voltage-Regulator zu verlieren und auch um Komplikationen mit ihm zu vermeiden ist es ratsam, den SMD Widerstand R1 vom Extensionboard zu entfernen. Somit wurde der Tinynode über die an den Pins angelegte Spannung versorgt, das Extensionboard weiterhin über den Voltage-Regulator. Diese Versorgung war auch nötig um den Datentransfer zwischen PC und TinyNode sicherzustellen. In Abb. 5-1 ist der Messaufbau ersichtlich.

### 5.2 Quellcode

Mit dem PowerAnalyzer war es möglich den Stromverbrauch sowie Lese- / Schreibzeiten auszumessen. Folgendes Testprogramm wurde dazu verwendet (Code auf CD, Anh. F):

```
...
TOSH_CLR_ADC4_PIN();
TOSH_MAKE_ADC4_OUTPUT();           // Pin 6.4 initialisiert
...
TOSH_SET_ADC4_PIN();               // Pin 6.4 auf High gesetzt
    for(i=0;i<100;i++)
    {
        call SD.writeSector(i,&i);
    }
TOSH_CLR_ADC4_PIN();               // Pin 6.4 auf Low gesetzt
```

. . .

Für die Tests wurden also 100 Sektoren (50kB Daten) auf die SD-Karte geschrieben und später auch wieder gelesen. Der Pin 6.4 dient zur genauen zeitlichen Lokalisation von Beginn und Ende des Schreibvorganges.

### *5.3 Resultate*

Aus den Messresultaten (Tabelle 5-2) lassen sich folgende Schlüsse ziehen:

- Mit den Pull-up Widerständen wird weniger Strom verbraucht als ohne.
- Der Stromverbrauch zwischen der SD-Karte und der MicroSD-Karte unterscheiden sich stark, und zwar sowohl beim Schreib- als auch beim Lesevorgang.
- Die Zeitdauer der Lese- und Schreiboperationen sind bei der SD- und MicroSD-Karte weitgehend identisch.
- Der Unterschied beim Stromverbrauch zwischen Idle und der Write-Operation ist auf dem Extensionboard grösser als auf dem SIB. Hingegen ist der Idle-Strom auf dem SIB grösser.
- Auf dem SIB dauern die Schreibvorgänge länger als auf dem Extensionboard.
- Die eine MicroSD-Karte hatte ein extrem schnelle Lesegeschwindigkeit, die in keiner Weise zu den anderen Messungen passt.

Aus diesen Messungen lässt sich schliessen, dass auf jeden Fall Pull-up Widerstände und wenn möglich MicroSD-Karten verwendet werden sollten. Wie schon früher erwähnt sind jedoch noch keine MicroSD-Karten für einen Temperaturbereich von bis zu  $-40^{\circ}\text{C}$  erhältlich, so dass vorerst SD-Karten zum Einsatz kommen.

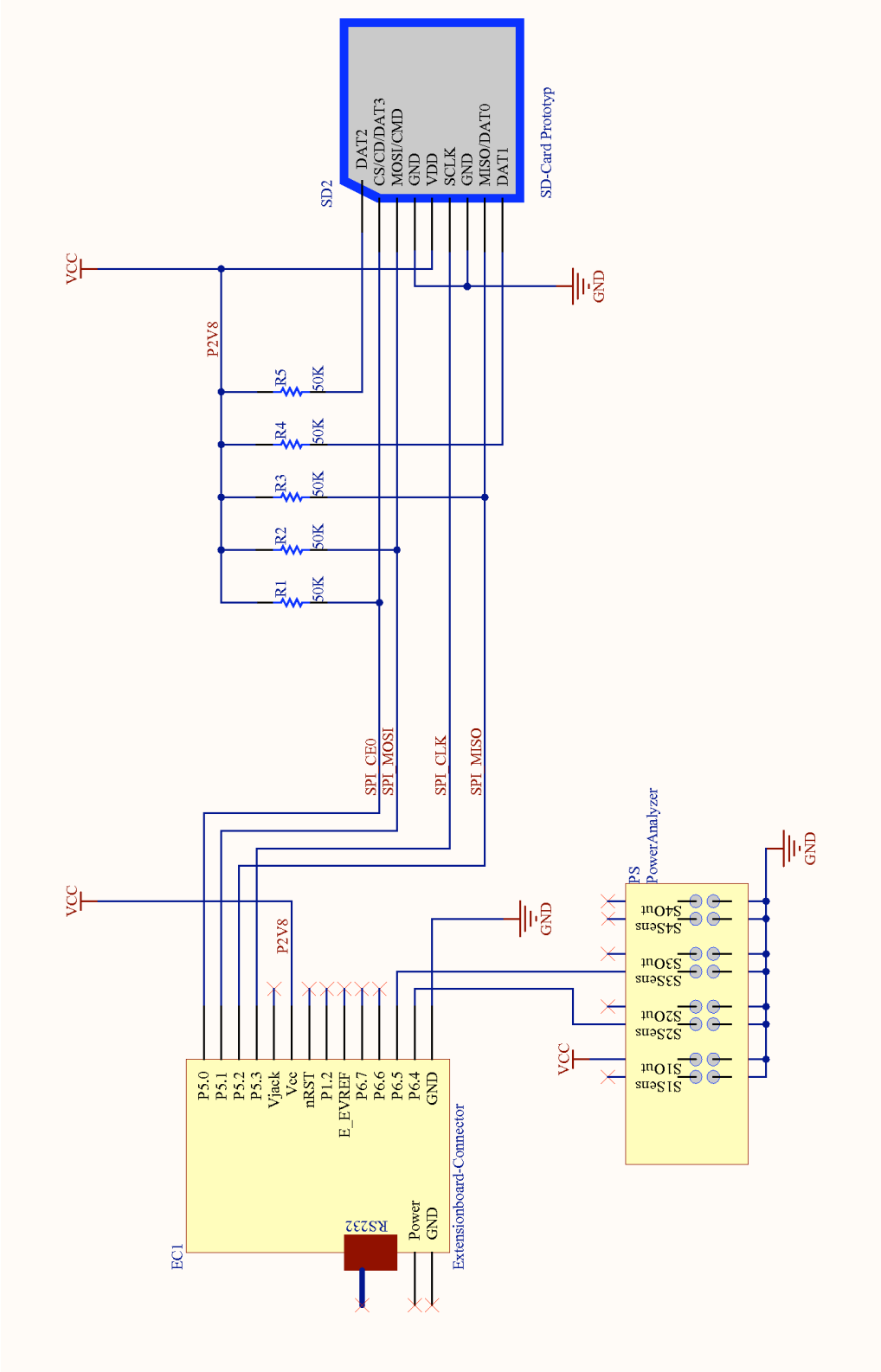


Abbildung 5-1  
PowerMessungen mit Agilent Technologies DC Power Analyzer

## Kapitel 5: Messungen der SD-Karten

Karten-Typ	Stromwert im Ruhezustand	Stromwert Schreibvorgang	Stromwert Lesevorgang	Zeit zum Schreiben	Zeit zum Lesen
mit Extensionboard:					
ohne SD-Karte	2.85mA	-	-	-	-
SanDisk Micro SD 1GB OC	2.9mA	49.3mA	12.0mA	5.64s	5.64s
SanDisk Micro SD 1GB OC mit Pull-Up	3.1mA	48.4mA	12.0mA	5.64s	5.7s
SanDisk Micro SD 1GB VW	2.9mA	48.7mA	12.3mA	5.64s	5.52s
SanDisk Micro SD 1GB VW mit Pull-Up	2.9mA	48.3mA	12.1mA	5.64s	5.52s
SanDisk SD 1GB	2.9mA	72.3mA	48.7mA	5.62s	5.7s
SanDisk SD 1GB mit Pull-Up	2.9mA	72mA	48.4mA	5.64s	5.7s
mit SIB:					
SanDisk SD 1GB VW	11.0mA	35.9mA	26.8mA	14.06s	6.86s
SanDisk SD 1GB VW mit Pull-Up	10mA	33.7mA	27.1mA	13.20s	6.75s
SanDisk Micro SD 1GB VW	11.5mA	30.8mA	19.1mA	13.22s	200ms
SanDisk Micro SD 1GB VW mit Pull-Up	11.4mA	30.9mA	18.4mA	13.31s	200ms

Abbildung 5-2  
Ergebnisse der Powermessungen

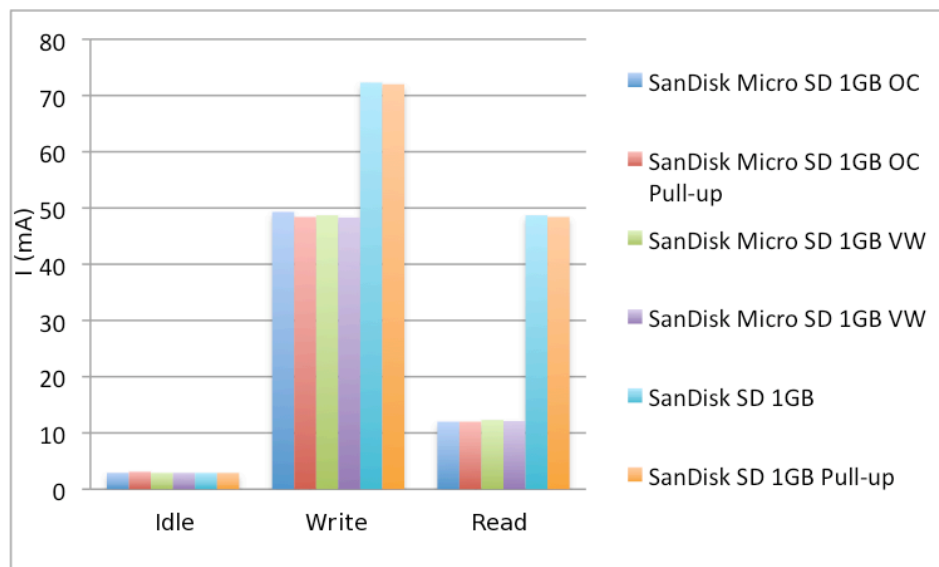


Abbildung 5-3  
Messungen auf dem Extensionboard

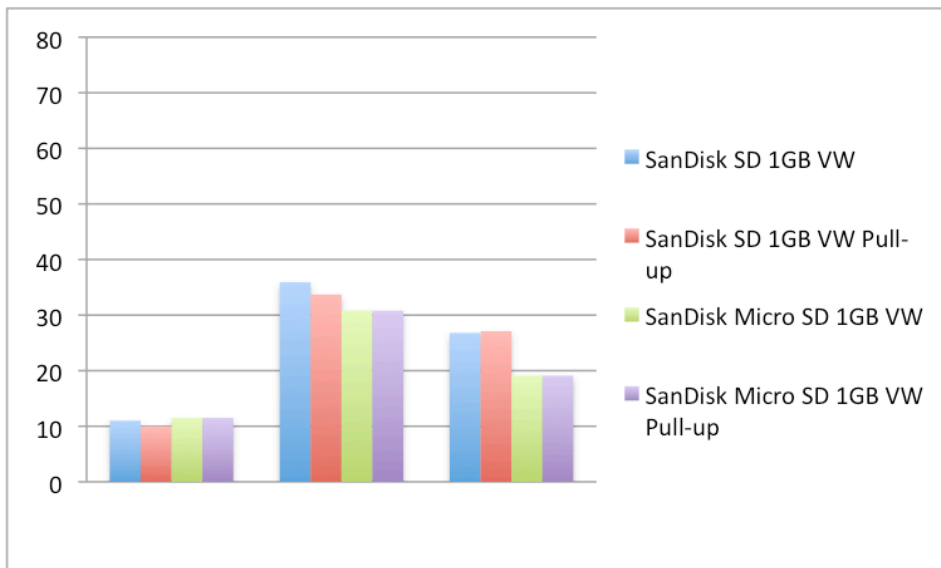


Abbildung 5-4  
Messungen auf dem Sensor Interface Board

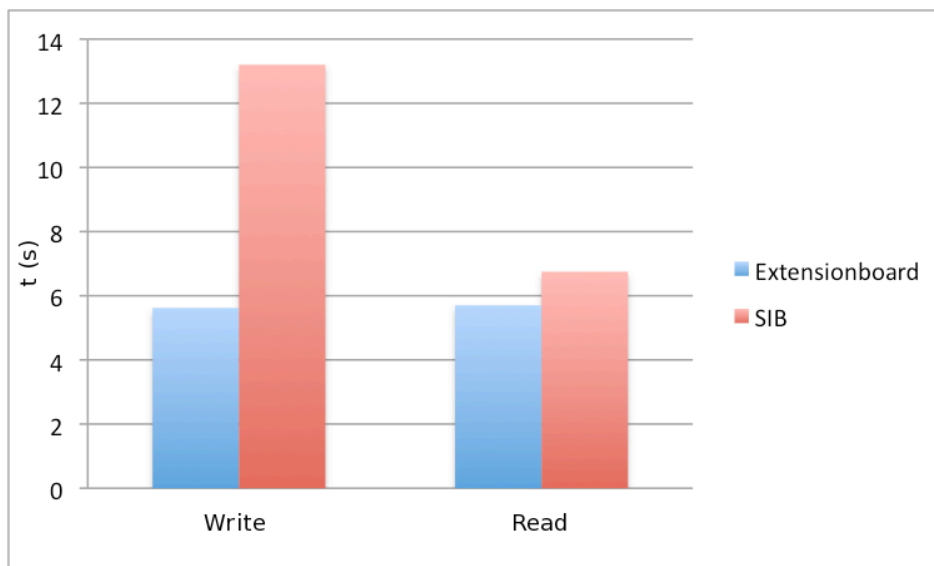


Abbildung 5-5  
Zeitmessungen





# 6

## Hardware Design

### 6.1 Footprints

Für das Erweiterungsboard werden sechs verschiedene Komponenten benötigt (Anh. C). Da diese nicht in der Standard-Library von 'Altium Designer' enthalten waren, mussten bis auf die Widerstände alle Footprints manuell gezeichnet werden. Die dazu notwendigen Masse sind in den entsprechenden Datenblättern (Anh. F) enthalten. Folgende Abbildungen zeigen zwei der erstellten Footprints:

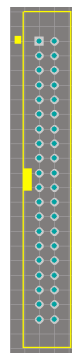
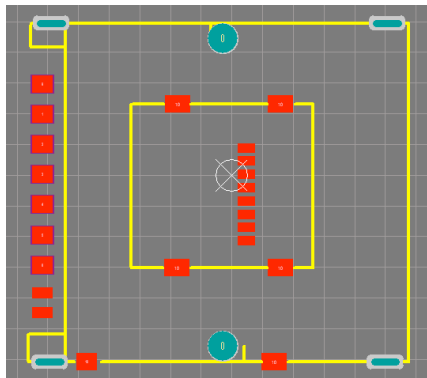


Abbildung 6-1 Doppelter SD-Halter    Abbildung 6-2 Buchsenleiste

### 6.2 Schema

Die fertigen Footprints wurden im Schema platziert (Abb. 6-3). Zur besseren Übersichtlichkeit wurden Busse verwendet. Die verwendeten Signalnamen entsprechen denjenigen aus den SIB-Schemas [8].

### 6.3 Printplattendesign

Beim Printplattendesign gab es eine Vielzahl von Möglichkeiten und Freiheiten: Sollte die Printplatte vertikal oder horizontal auf das SIB gesteckt werden? Sollte

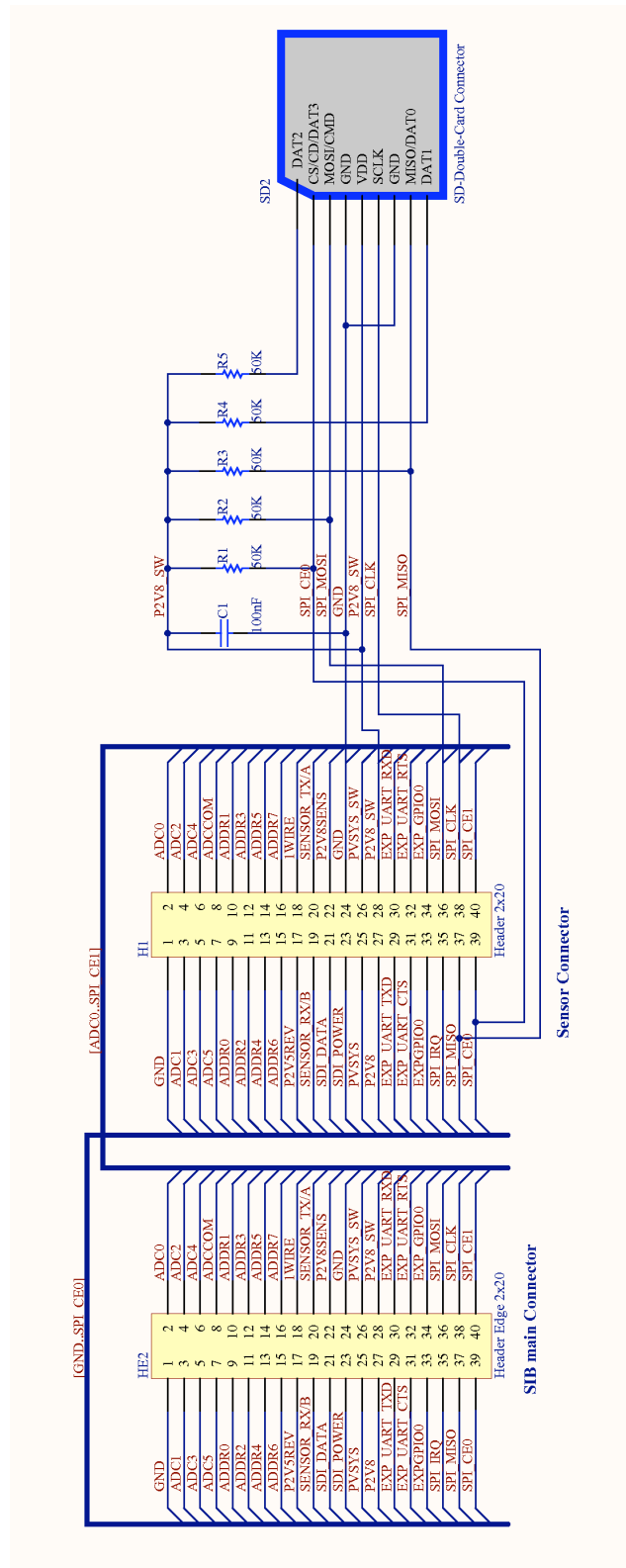


Abbildung 6-3  
Schaltungsaufbau: Print für SD-Karte

auf die kleineren MicroSD-Halter oder auf die grösseren SD-Halter gesetzt werden? Hat die ganze Anordnung mit dem zur Verfügung stehenden Raum überhaupt im SIB-Gehäuse Platz?

Der erste Lösungsansatz ging von einer horizontalen Montage aus, wobei das Board direkt auf dem TinyNode aufliegt:

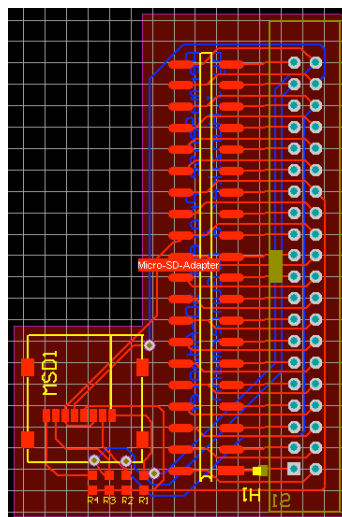
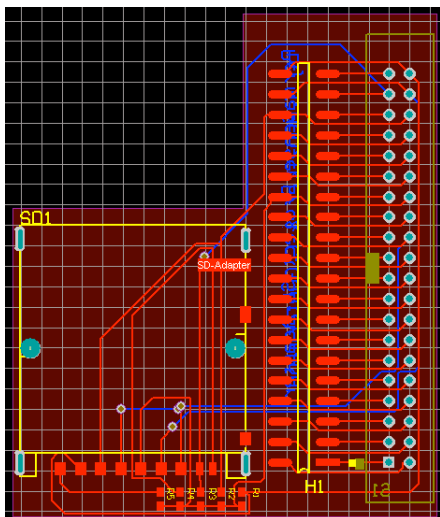


Abbildung 6-4 Horizontale Montage, SD    Abbildung 6-5 Horizontale Montage, MicroSD

Diese Lösung beansprucht aber mit dem SD-Halter (Abb. 6-4) zu viel Platz und kommt deshalb nicht in Frage. Mit dem MicroSD-Halter (Abb. 6-5) war es vom Platz her prinzipiell machbar, jedoch würde der Reset-Button verdeckt und man hätte keine freie Sicht auf die TinyNode-LED mehr. Vielversprechender war da die Idee der vertikalen Montage:

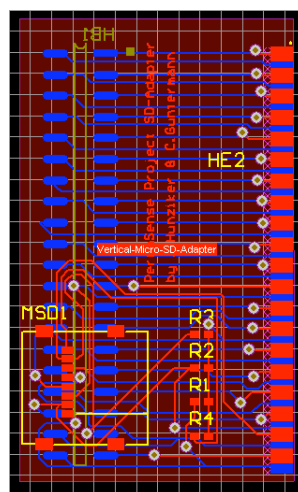
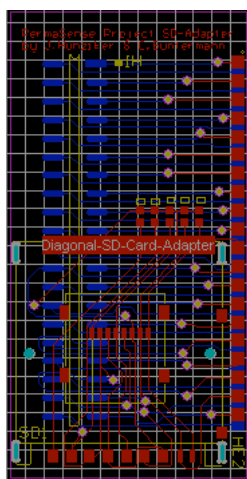


Abbildung 6-6 Vertikale Montage, SD    Abbildung 6-7 Vertikale Montage, MicroSD

Wir entschieden uns für die vertikale Lösung, wobei der SD-Karten-Adapter gerade an der Batterie anliegt und auf der Rückseite die 40Pin-Stiftleiste mit Flachbandkabel verbunden werden sollte. Diese Stiftleiste hat den Zweck die Datenleitungen

weiter zu verbinden. Die Karte greift einzelne Daten- und Powerleitungen ab, aber alle Leitungen bleiben für das übrige System erhalten. Diese Lösung nutzt die vorhandenen Platzverhältnisse optimal aus; vom Board bis zum SIB-Gehäusedeckel besteht weniger als ein Millimeter Spielraum. Das definitive Layout enthält sowohl einen MicroSD- als auch einen SD-Footprint (Abb. 6-8). So kann zum Zeitpunkt der Bestückung über die Wahl des Halters entschieden werden.

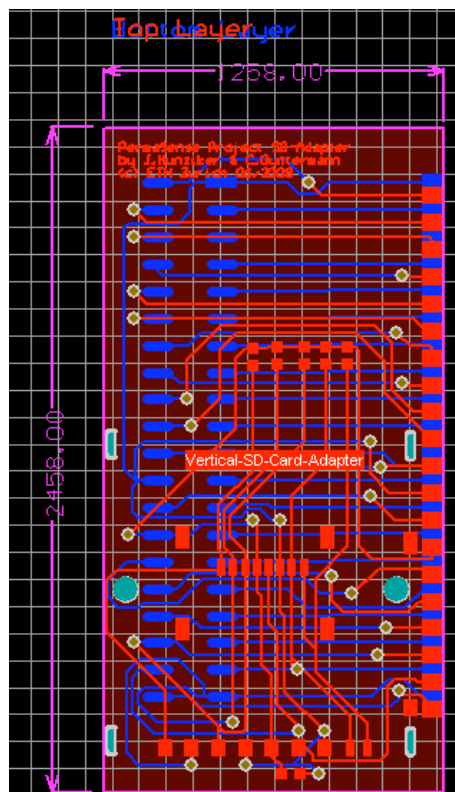
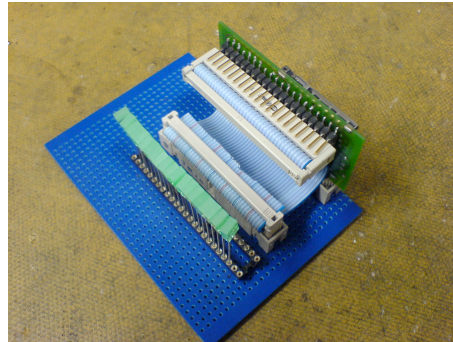
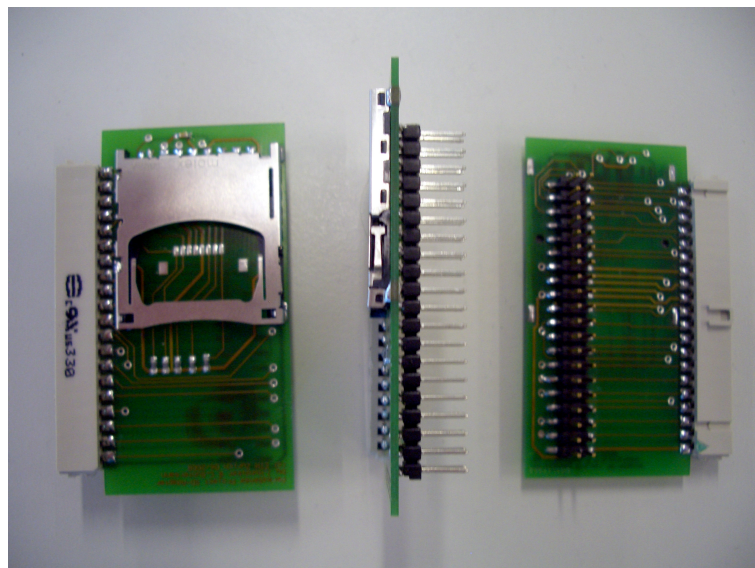


Abbildung 6-8 Der endgültige PCB-Print

Die Produktion der Prints dauerte acht Tage. Um die Lötarbeit zu überprüfen wurde ein Anzeigergerät entworfen mit welchem unsaubere Lötstellen schnell gefunden werden können. Zusätzlich wurden alle Prints einem funktionalen Systemtest unterzogen.



*Abbildung 6-9*  
*Print-Debugger*



*Abbildung 6-10*  
*Das fertige Erweiterungsboard*



# 7

## *Software*

Um einen einfachen Zugriff auf die Speicherfunktionen der SD-Karte zu ermöglichen, wurde neben der Hardwareerweiterung des SIB auch eine Softwarekomponente namens PermaStore entwickelt. Dieses Kapitel erläutert unsere konzeptuellen Überlegungen zur Softwarearchitektur und dokumentiert die einzelnen entwickelten Softwarelayers im Detail.

### *7.1 Die PermaStore-Komponente*

#### *7.1.1 Grundsätzliche Softwarearchitektur*

Folgende Anforderungen für die PermaStore-Komponente wurden in Absprache mit dem Betreuer definiert:

- Einfache Einbindung in die PermaDozer-Applikation
- Speicherung der Daten in einem Format, welches ein einfaches Auslesen am PC ermöglicht

PermaStore ist aus mehreren Layern aufgebaut, welche aufeinander aufbauen und jeweils auf bereitgestellte Services des nächst unteren Layers zugreifen.

Diese Schichtung bedeutet einen grossen Gewinn an Flexibilität im Vergleich zu einer einzelnen 'All-in-One'-Komponente: Gegen unten wird eine gewisse Hardwareunabhängigkeit erreicht (das PermaFAT-Filesystem würde auch für andere Speicher funktionieren, sofern deren Treiber das SD-Interface implementiert). Gegen oben ergibt sich andererseits eine grössere Anwendungsflexibilität, da die Anwendungsspezifischen Operationen erst im obersten Layer ins Spiel gebracht werden. PermaStore implementiert die TinyOS-Standardinterfaces 'BlockWrite', 'BlockRead', 'Mount' und 'StorageRemap' und könnte somit ohne grossen Portierungsaufwand auch in anderen Anwendungen eingesetzt werden. Auch die Frage der einfachen Integration ist mit diesem Ansatz elegant gelöst, denn PermaDozer griff auch schon vor PermaStore auf diese Interfaces zu um eine begrenzte Anzahl Pakete im TinyNode-eigenen Flashspeicher zu loggen.

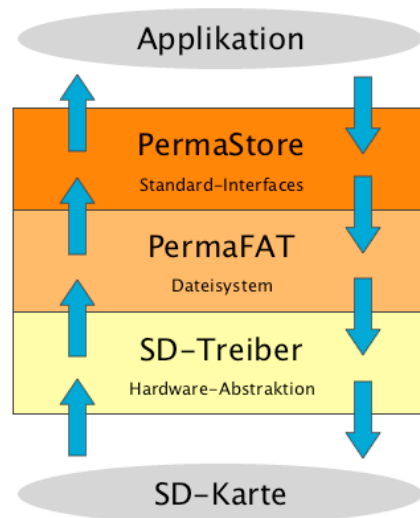


Abbildung 7-1  
PermaStore-Softwareschichtung

## 7.1.2 Die Schichten im Detail

### 7.1.2.1 SD-Treiber

Der unterste Layer des Speichersystems ist der SD-Treiber, welcher direkt für die Kommunikation mit der Karte verantwortlich ist. Diese geschieht über den SPI-Bus des TinyNode. Der SPI-Modus bietet bei SD-Karten zwar ziemlich magere Performance (im konkreten Fall bewegen sich die Werte etwa um die 10kB/sec, im Vergleich dazu bieten moderne SD-Cards mit den entsprechenden Controllern eine Schreibgeschwindigkeit von bis zu 10MB/sec), jedoch ist er einfach zu implementieren und stellt recht geringe Anforderungen an den Controller.

Der verwendete Treiber stammt ursprünglich von Steven Ayer (Intel Digital Health Group) und wurde eigentlich für die SHIMMER-Plattform [10] geschrieben. Er basiert grösstenteils auf dem von Texas Instruments bereitgestellten Sample Code für die Kommunikation zwischen dem MSP430-Microcontroller und einer MMC-Karte [11]. Folgende Funktionen werden im SD-Interface definiert und somit durch den Treiber zur Verfügung gestellt:

- `init`: Initialisiert die SD-Karte.
- `setIdle`: Versetzt die Karte in den Idle-Modus.
- `setBlockLength`: Ändert die Lese- oder Schreibblocklänge<sup>1</sup>.
- `readBlock`: Liest einen Block von beliebiger Länge.
- `writeBlock`: Schreibt einen Block von beliebiger Länge<sup>2</sup>.
- `readSector`: Liest einen Sektor à 512 Byte.

---

<sup>1</sup>Achtung: die meisten SD-Karten unterstützen nur Schreibvorgänge mit einer Blocklänge von 512 Bytes

<sup>2</sup>Falls dies von der SD-Karte unterstützt wird



- `writeSector`: Schreibt einen Sektor à 512 Byte.
- `readRegister`: Liest ein Register der SD-Karte.
- `readCardSize`: Liest die Kartengrösse einer SD-Karte aus.

Um den SD-Treiber auch auf dem SIB verwenden zu können wurden folgende Anpassungen vorgenommen:

- Verwendung des SPIByte-Interfaces an Stelle der direkten Übertragung via UART
- Anpassung des Chip-Select Pins

### 7.1.2.2 PermaFAT

Grundsätzlich wäre es ohne weiteres möglich gewesen, die von PermaDozer hauptsächlich verwendeten `BlockRead` und `BlockWrite`-Interfaces direkt 'auf' der SD-Treiberschicht zu implementieren. Ziel der PermaFAT-Zwischenschicht ist es, ein Dateisystem zur Verfügung zu stellen welches ein komfortables Auslesen der Daten an einem PC ermöglicht. PermaFAT stellt eine reduzierte Implementierung eines FAT32-Dateisystems dar, welches eines der am weitesten verbreiteten Dateisysteme überhaupt ist [9]. Stark reduziert bedeutet hier, dass nur die nötigsten Funktionen implementiert wurden, damit eine mit PermaFAT beschriebene SD-Karte durch jedes Betriebssystem als FAT32-Volume erkannt und gemountet wird.

Das für PermaFAT definierte Interface bietet folgende Operationen:

- `init`: Initialisiert das Filesystem und signalisiert `PermaFAT.initDone` wenn fertig. Dieser Befehl beinhaltet auch die Initialisierung der SD-Karte, folglich muss dies nicht separat durchgeführt werden.
- `checkFileID`: Überprüft, ob die übergebene FileID gültig ist, das File also existiert.
- `getSize`: Gibt die Grösse des Datenträgerbereichs zurück, welcher für Nutzdaten verfügbar ist. Dafür werden von der physikalischen Grösse der SD-Karte die für das Dateisystem benötigten Anteile abgezogen.
- `writeToFile`: Schreibt eine Anzahl Bytes in eine Datei (Random Access) und signalisiert `PermaFAT.writeDone` wenn fertig.
- `appendToFile`: Hängt eine Anzahl Bytes ans Ende der Datei an und signalisiert `PermaFAT.writeDone` wenn fertig.
- `commit`: Führt eventuell gepufferte Schreibvorgänge aus und stellt somit sicher, dass alle geschriebenen Daten auch wirklich auf der SD-Karte gespeichert sind. Signalisiert `PermaFAT.commitDone` wenn fertig.
- `readFromFile`: Liest eine Anzahl Bytes aus der Datei (Random Access) und signalisiert `PermaFAT.writeDone` wenn fertig.
- `clearFile`: Löscht die Inhalte einer Datei vollständig - nicht aber die Datei an sich. Signalisiert `PermaFAT.clearDone` wenn fertig.

Wie in obiger Aufzählung ersichtlich sind alle Operationen welche Lese- und Schreibvorgänge auf der SD-Karte benötigen 'Split-Phase'-Operationen. Das bedeutet, dass zu jeder dieser Operation ein zugehöriges Event existiert ('initDone', 'writeDone' etc.), welches signalisiert dass die Ausführung der Operation abgeschlossen ist. Dies ist notwendig, weil die SD-Karte nicht der einzige Benutzer des SPI-Busses ist und darum evtl. warten muss bis ihr der Bus von der Arbitrierungskomponente zugeteilt wird. So lange eine Operation im Gang ist oder auf den Bus wartet wird der Aufruf einer anderen Operation in jedem Fall FAIL zurückgeben und das zugehörige Done-Event wird nie gepostet.

Eine Besonderheit von PermaFAT ist das Fehlen jeglicher Funktionen zur Dateiverwaltung. Dateien können weder dynamisch erstellt noch gelöscht werden und es gibt auch keine Directories. Der Grund dafür liegt einerseits in der fehlenden Notwendigkeit für PermaSense, andererseits im beträchtlich erhöhten Implementierungsaufwand, welcher sich wiederum in einem viel grösseren Bedarf an Programmspeicher für PermaStore bemerkbar machen würde. Bei der initialen Formatierung wird in einer speziell dafür definierten Struktur angegeben, welche Dateien das Volume enthalten soll; dabei erhält jede Datei eine eindeutige ID beginnend mit 0. Dies kann nur durch eine weitere Formatierung - welche natürlich auch den Verlust aller gespeicherten Daten mit sich bringt - geändert werden. Weitere Informationen zur Formatierung finden sich unter Abschnitt 7.2.2.

Es ist zu beachten dass mit PermaFAT nur Karten verwendet werden können, welche auch mit der FormatSDCard-Applikation (Abschn. 7.2.2) formatiert wurden. Bei einer an einem anderen Gerät formatierten SD-Karte wird die Initialisierung von PermaFAT in jedem Fall fehlschlagen.

### 7.1.2.3 PermaStore

Sinn und Zweck von der PermaStore-Komponente ist es, applikationsseitig ein Standardinterface zu bieten und damit die Integration in bestehende Applikationen zu vereinfachen. Letztlich ist PermaStore nichts anderes als ein Wrapper der PermaFAT-Komponente, welcher die Spezialitäten eines Dateisystems in das BlockStorage-Konzept übersetzt: Anstatt mehrerer Volumes gibt es nun einfach mehrere Dateien in PermaFAT, das Mounten eines Volumes reduziert sich damit zu einem Wechsel der gerade aktiven Datei<sup>3</sup>. PermaStore implementiert folgende Operationen der Interfaces 'Mount', 'BlockRead', 'BlockWrite' und 'StorageRemap':

- *Mount.mount*: Setzt die aktive Datei, auf welche in den folgenden Lese- und Schreibzugriffen zugegriffen wird. Signalisiert *Mount.mountDone* wenn fertig.
- *BlockRead.read*: Liest eine Anzahl Bytes aus der aktiven Datei in den übergebenen Puffer. Signalisiert *Read.readDone* wenn fertig.
- *BlockRead.verify*: In PermaStore nicht verwendet. Signalisiert sofort *BlockRead.verifyDone*.
- *BlockRead.computeCrc*: In PermaStore nicht verwendet. Signalisiert sofort *BlockRead.computeCrcDone*.

---

<sup>3</sup>Eine Ausnahme bildet hier die erste Mount-Operation, welche PermaFAT und die SD-Karte initialisiert

- *BlockRead.getSize*: Gibt die Grösse des für Nutzdaten verfügbaren Speicherplatzes auf der SD-Karte zurück.
- *BlockWrite.write*: Schreibt eine Anzahl Bytes aus dem übergebene Puffer in die aktive Datei. Signalisiert *BlockWrite.writeDone* wenn fertig.
- *BlockWrite.erase*: Löscht die Inhalte der aktiven Datei vollständig. Signalisiert *BlockWrite.eraseDone* wenn fertig.
- *BlockWrite.commit*: Führt eventuell gepufferte Schreibvorgänge aus und stellt somit sicher, dass alle geschriebenen Daten auch wirklich auf der SD-Karte gespeichert sind. Signalisiert *BlockWrite.commitDone* wenn fertig.
- *StorageRemap.physicalAddr*: In PermaStore nicht verwendet. Gibt die übergebene Adresse unverändert zurück.

Damit bietet PermaStore den gleichen Funktionsumfang wie die Standard 'BlockStorageC'-Komponente.

## 7.2 Applikationen

Neben den oben beschriebenen Komponenten wurden im Laufe des Designprozesses auch zwei wichtige Applikationen entwickelt. Die eine Applikation ist ein Memory-Test, welcher es erlaubt die Gesamtkonfiguration (Soft- und Hardware) zu testen und gewisse Performance-Daten zu gewinnen. Bei der zweiten Anwendung handelt es sich um ein Formatierungsprogramm, welches eine SD-Karte mit dem PermaFAT-Filesystem formatiert und die zuvor spezifizierten Dateien anlegt. Daneben entstanden im Verlaufe des Design-Prozesses natürlich eine Unmenge an kleinen Testapplikationen um eine bestimmte Funktionalität auszutesten, doch sind diese sehr spezifisch auf ein spezielles Problem ausgerichtet und deshalb keiner weiteren Erwähnung wert.

Beide vorgestellten Applikationen sind auf der beigelegten CD zu finden (Anh. F).

### 7.2.1 Die TestSD-Applikation

TestSD (zu finden im Permasense-Trunk unter *sensor\_interface\_board/apps/SibTestSD*) führt einen Memory-Test auf der SD-Karte durch. Die Ergebnisse dieser Tests werden direkt auf der SD-Karte in einem File geloggt (aus diesem Grund muss die zu testende Karte vorher auch mit der unter 7.2.2 beschriebenen Applikation mit mindestens einer Datei formatiert werden). TestSD führt 3 Arten von Tests durch:

- Block-Test
- Jump-Test
- Performance-Test

Die grundsätzlichen Ideen für den Block- und den Jump-Test sind der *bttest*-Applikation für den *BTNode* entnommen [13].

Mit der zu Beginn des Files definierten Variable *FRAC\_TO\_TEST* wird der Bruchteil der Karte definiert, welcher bei jedem Test getestet werden soll. Ist die Karte also 1GB gross und *FRAC\_TO\_TEST*=1000, so wird pro Test ca. 1MB Speicher getestet.

Beim *Block-Test* wird jeweils ein zusammenhängender Block von Sektoren getestet (vgl. Abb. 7-2). Die Grösse dieses Blocks entspricht dem durch `FRAC_TO_TEST` definierten Bruchteil der Gesamtspeichergrösse. Der Block wird zweimal mit einem bestimmten Muster beschrieben und danach wieder ausgelesen. Stimmt das Geschriebene nicht mit dem Gelesenen überein wird eine Fehlermeldung geloggt, welcher die genaue Adresse des Fehlers, das geschriebene Byte und das gelesene Byte ausgibt. Beim ersten Durchgang wird der Block mit `0x00FF00FF00FF...` beschrieben, beim zweiten mit `0xFF00FF00FF00...`. Damit wird sichergestellt, dass jedes Bit einmal auf 0 und einmal auf 1 gesetzt war. Der Block-Test wird dreimal ausgeführt: Zu Beginn, in der Mitte und am Ende des zu testenden Speicherbereichs (welcher sich wegen den Platzanforderungen für Filesystem und Logfile nicht ganz über die gesamte SD-Karte erstreckt).

Der *Jump-Test* testet einzelne Sektoren im Testbereich (vgl. Abb. 7-3). Er beginnt an dessen Anfang, testet einen Sektor und 'springt' dann zum nächsten Testsektor, welcher `FRAC_TO_TEST` Sektoren weiter liegt. Die dabei in die Sektoren geschriebenen Werte werden durch eine XOR-Verknüpfung der aktuellen Sektornummer mit dem Byte-Offset innerhalb des Sektors generiert um eine gleichmässige Verteilung der 0-Bits und 1-Bits zu erreichen. Wiederum werden die Sektoren nach dem Schreibdurchgang wieder ausgelesen, mit den geschriebenen Werten verglichen und allfällige Fehler geloggt.

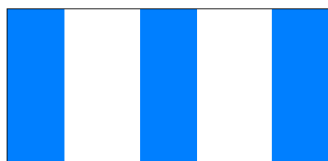


Abbildung 7-2 Block-Test Prinzip

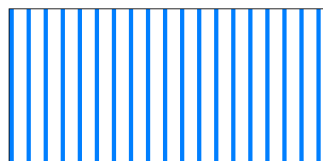


Abbildung 7-3 Jump-Test Prinzip

Zu guter Letzt wird mit dem *Performance-Test* die Schreib- und Leseperformance des Speichers gemessen. Dafür wird ein Block von gleicher Grösse wie beim *Block-Test* geschrieben und wieder ausgelesen. Die dafür benötigte Zeit wird dann über die Blockgrösse gemittelt und als durchschnittliche Schreib- und Lesezugriffszeit im Logfile vermerkt.

Da Permasense im Hochgebirge und somit in einer Umgebung mit extremen Temperaturen eingesetzt wird, kann *TestSD* auch verwendet werden um in einer Klimakammer einen 'Temperature Cycle Test' durchzuführen. Dabei wird *TestSD* bei schwankender Temperatur immer und immer wieder ausgeführt um z.B. temperaturbedingte Performanceveränderungen feststellen zu können. In *TestSD* wird deshalb auch die lokale Temperatur vor jedem Test gemessen und ins Logfile geschrieben. Mit dem define-Flag `LOOP_TESTING` wird festgelegt, ob die Testreihe nur einmal oder in einer Endlosschleife ausgeführt wird.

Leider konnten wir im Laufe unserer Arbeit diesen Temperature Cycle Test nicht selber durchführen, da die dafür vorgesehene Klimakammer zum Zeitpunkt wo dies möglich gewesen wäre defekt war.

Die SIB-LEDs werden als in *TestSD* Statusanzeige gebraucht; sobald alle 6 LEDs leuchten ist der Test beendet. Falls die SD-Karte nicht initialisiert werden kann,

leuchten direkt nach Programmstart die LEDs 0 und 5.

### 7.2.2 Die FormatSDCard-Applikation

FormatSDCard (zu finden im Permasense-Trunk unter *sensor\_interface\_board/apps/SibFormatSDCard*) wird dazu benötigt, ein SD-Karte vor ihrer Verwendung mit PermaFAT zu formatieren. Diese Funktionalität wurde aus dem eigentlichen PermaFAT-Modul ausgelagert um dessen Platzbedarf im Programmspeicher zu verringern, da sie für den eigentlichen Betrieb des Speichermoduls nicht benötigt wird (es reicht eine initiale Formatierung vor Einbau ins Zielsystem).

FormatSDCard nimmt die benötigten Konstanten für die Formatierung aus der Datei *PermaFAT.h*, darunter z.B. die FAT32 Sektor- und Clustergrösse. Für eine genaue Beschreibung der Konstanten sei auf [9] verwiesen.

Für den Benutzer ist vor allem die Datei *InitialFiles.h* von Bedeutung, welche sich im Applikationsverzeichnis selbst befindet. Dort werden nämlich alle Dateien spezifiziert, welche während der Formatierung auf der SD-Karte erstellt werden. Dabei können der gewünschte Dateiname und falls erwünscht der initiale Dateinhalt angegeben werden (genauere Instruktionen sind im Quellcode als Kommentar enthalten). *In PermaFAT können nur an dieser Stelle Dateien erstellt werden. Es gibt keine Möglichkeit, im laufenden Betrieb weitere Dateien zu erstellen oder schon bestehende Dateien zu löschen.*

Die SIB-LEDs werden als Statusanzeige gebraucht; sobald die ersten zwei LEDs leuchten ist die Formatierung abgeschlossen. Falls die SD-Karte nicht initialisiert werden kann, leuchten direkt nach Programmstart die LEDs 0 und 5.



# A

## Messungen mit dem SIB

Im Prinzip waren die Messungen mit dem SIB identisch mit denjenigen auf dem Extensionboard. Nur verwendeten wir hier gepatchte Tinynodes (App. B), damit man den Voltage Regulator umgehen konnte. Ebenfalls mussten einige Pin-Bezeichnungen im hardware.h-Headerfile abgeändert werden. Beim Extensionboard wurde der Pin 5.0 als ChipSelect verwendet. Dieser steuert nun beim SIB den 2V8\_SW an. Die Pins 4.0 und 4.1 dienen hier jetzt für zwei ChipSelect-Leitungen. Um die Schreib- und Lesezeit zu messen, speisten wir die Karte vorübergehend über den fixen 2V8 Pin und verwendeten den P2V8\_SW zur Anzeige. Daneben gab es noch die zweite ChipSelect-Leitung, welche wir ebenfalls als Datenleitung gebraucht haben.

Das hardware.h-Headerfile findet man unter `/opt/tinynos-1.x/contrib/shockfish/tos/platform/tinynode`

```
...
TOSH_CLR_P40_PIN();
TOSH_MAKE_P40_OUTPUT();
TOSH_CLR_P41_PIN();
TOSH_MAKE_P41_OUTPUT();
...
TOSH_SET_P40_PIN();
...
TOSH_CLR_P40_PIN();
....
```

*Abbildung A-1*  
*Gesetzte Pins im SDTestM.nc*

```
...
TOSH_ASSIGN_PIN(STE1,5,0);  -->  TOSH_ASSIGN_PIN(STE1,4,0);
...
TOSH_ASSIGN_PIN(P40,4,0);  -->  TOSH_ASSIGN_PIN(P40,5,0);
...
```

*Abbildung A-2*  
*Veränderung im hardware.h-Headerfile*





# B

## Patchen eines Tinynodes

Um einen Tinynode zu patchen muss - wie in den Abbildungen (B-1, B-2) ersichtlich - der Bypass-Widerstand entfernt und der Pull-Up Widerstand als Pull-Down Widerstand angelötet werden. Somit ist nun der nREGE-Pin standardmässig auf Low gesetzt und aktiviert den Voltage-Regulator. Der Bypass-Widerstand wird dadurch überflüssig.

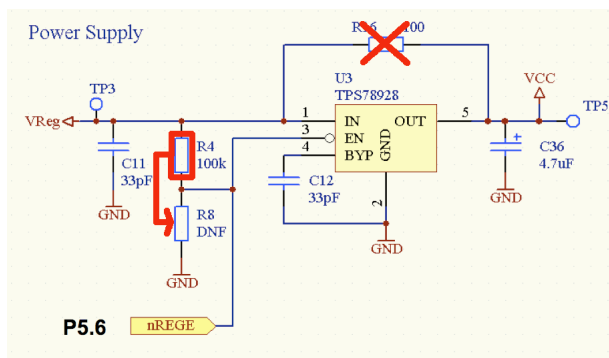


Abbildung B-1 Patch Schema

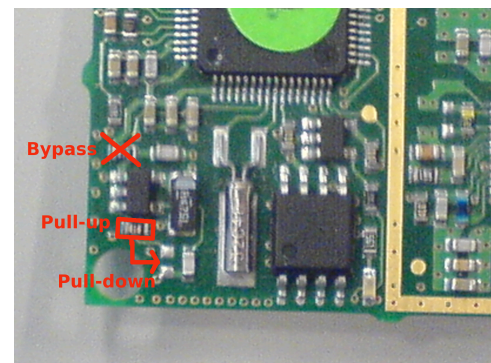


Abbildung B-2 Patch Foto

Durch diese Massnahme sind nun immer 2.8 V auf dem Tinynode und dem SIB-Netz.



# C

## Liste der verwendeten Teile

Die Teile haben wir meist von Farnell bestellt. Auf der CD findet man die dazugehörigen Datasheets (App. F). Was die Wahl der Teile anbelangt, gibt es eine fast unbeschränkte Anzahl an Möglichkeiten. Hier könnte man bestimmt noch einige Dinge optimieren. Der grössere SD-Halter hat zum Beispiel runde Noppen auf der Unterseite für die man extra Löcher in der Printplatte bohren muss. Diese Löcher geben jedoch dem SD-Halter nicht viel mehr Halt. Auch was den Preis anbelangt könnte man bestimmt noch andere und billigere Lösungen finden.







Name	Hersteller	Produkt-ID	
Micro SD Connector	MULTICOMP	460DA40013	
SD Connector	MOLEX	679938001	
SMD Stiftleisten	FISCHER ELEKTRONIK	9729046	
Steckverbinder	HARTING	09185407813	
R51K 0603	MULTICOMP	9331310	
C100nF 0603	AVX	06035C104KAT2A	

Abbildung C-1




Liste mit Hersteller und Produkt-ID



# D

## *Liste der verwendeten SD-Cards*

Auf der CD findet man die dazugehörigen Datenblätter (Anh. F). Für die Messungen (Kap. 5) wurden teilweise mehrere Karten vom gleichen Typ verwendet, dadurch wurden die Toleranzen der einzelnen Messungen sichtbar.

Name	Hersteller	Specification	
Micro SD 1GB	SanDisk	SD Association	
SD 1GB	SanDisk	SD Association	
SD 1GB industrial	Transcend	Transcend	

*Abbildung D-1*  
*Verwendete SD-Karten*



# E

## *Konfiguration von TinyOS und Tinynode unter Ubuntu 8.04*

Die erste Hürde in der Arbeit mit TinyOS ist die Installation, welche sich unter Umständen als ziemlich kompliziert herausstellen kann da die benötigten Dateien aus verschiedenen Quellen stammen und einige Umgebungsvariablen korrekt gesetzt werden müssen. Aus diesem Grund haben wir ein einfaches Installationsskript geschrieben, welches einen Grossteil der Konfiguration automatisiert.

*Dabei sei hervorgehoben dass das Skript zwar zum Erstellungszeitpunkt auf unseren Systemen funktioniert hat, aber bei Änderungen des Systems oder Updates von TinyOS Änderungen im Skript nötig sein können welche nicht vorhersehbar sind. Wir geben keine Garantie auf die eine korrekte Ausführung, hoffen jedoch dass es dem Einen oder Anderen Neueinsteiger nützlich sein kann.* Es handelt sich hier um die Umgebung, welche wir verwendet haben um unsere Anwendungen zu kompilieren. Andere Anwendungen benötigen evtl. weitere Komponenten, deren Pfade zusätzlich im Makefile definiert werden müssen.

Alle benötigten Dateien sind auf der CD unter im Ordner 'Scripts' enthalten (Anh. F). Alle TinyOS-Komponenten werden ins '/opt'-Verzeichnis installiert.

### *E.1 TinyOS Basisinstallation*

Um grundsätzliche TinyOS 1.x und 2.x-Applikationen auf Tinynode zu installieren, kann das Skript 'tinyos-install.sh' benutzt werden. Dazu sind folgende Schritte nötig:

- Kopieren Sie den 'Scripts'-Ordner in Ihr Homeverzeichnis
- Öffnen Sie ein Terminal und wechseln Sie in den Skriptordner:

```
cd ~/Scripts
```

- Machen Sie die Scripts ausführbar:

```
chmod 777 *.sh
```

- Führen Sie das Basis-Installationsskript aus:

```
sudo ./tinyos-install.sh
```

Geben Sie Ihr Passwort ein und beantworten Sie die alle erscheinenden Fragen mit 'y'. (Beachten Sie: Die Ausführung dieses Skripts kann einige Zeit in Anspruch nehmen da beträchtliche Datenmengen aus heruntergeladen werden müssen.)

Sobald die Meldung erscheint, dass die Installation abgeschlossen sei, können Sie mit Öffnen einer neuen Shell die Änderungen in der Arbeitsumgebung wirksam machen.

Falls Sie Zugang zum Permasense SVN-Repository haben, können Sie mit der Permasense-Installation (E.2) weiterfahren. Ansonsten springen Sie direkt zum Abschluss der Installation unter E.3.

## *E.2 Permasense-Installation*

Damit die PermaStore-Komponenten verwendet werden können, sind einige zusätzliche Files aus dem Permasense SVN-Repository sowie weitere Änderungen in der Systemumgebung nötig. Führen Sie dazu folgende Schritte aus:

- Wechseln Sie wieder in den 'Scripts'-Ordner

```
cd ~/Scripts
```

- Führen Sie das Permasense-Installationsskript aus:

```
sudo ./dozer-install.sh username password
```

und ersetzen Sie dabei 'username' und 'password' mit Ihrem persönlichen Benutzernamen und Passwort für das Permasense SVN-Repository ([svn://svn.ee.ethz.ch/permasense](http://svn://svn.ee.ethz.ch/permasense)).

Warten Sie auch hier wieder bis die Abschluss-Meldung erscheint.

## *E.3 Abschluss der Installation*

Um die Installation abzuschliessen müssen nun noch die Rechte der erstellten Ordner geändert werden. Geben Sie dazu folgende Zeile in einer Shell ein:

```
sudo chown -R $USER /opt/tinyos-*
```

Falls Sie auch Permasense installiert haben, ist zusätzlich ein

```
sudo chown -R $USER /opt/permasense
```

nötig.

*Fertig!* Beim Öffnen einer neuen Shell werden die Umgebungsvariablen automatisch für TinyOS 1.x gesetzt. Um TinyOS 2.x zu benutzen geben Sie

```
tos2
```



ein. Ein Wechsel zurück zu TinyOS 1.x ist mit

tos1

möglich.



# F

## *CD-Inhalt*

Die beigelegte CD enthält alle relevanten Dateien, die entweder im Laufe dieser Arbeit erstellt wurden oder als Referenz von Nutzen sein können. Im Folgenden ist die Ordnerstruktur kurz beschrieben:

- Datenblätter: Enthält alle Datenblätter der verwendeten Komponenten (Anh. C).
- Dokumente: Enthält diesen Bericht sowie die Abschlusspräsentation als PDF-Datei.
- Hardware Design: Enthält alle erstellten 'Altium Designer'-Projektdateien (Kap. 6).
- Messungen: Enthält GIF-Dumps der Performance-Messungen (Kap. 5).
- Quellcode
  - Library: Enthält alle nötigen Quelldateien für PermaStore. Dies beinhaltet sowohl Komponenten als auch Interfaces.
  - Measuring Apps: Enthält die Applikationen, welche zu Messzwecken verwendet wurden (Kap. 5).
  - Production Apps: Enthält die Applikationen 'TestSD' (7.2.1) und 'FormatSDCard' (7.2.2).
- Scripts: Enthält die TinyOS Installations- und Konfigurationsskripts (Anh. E).



# Literaturverzeichnis

- [1] DC Power Analyzer. [www.agilent.com/find/N6705](http://www.agilent.com/find/N6705).
- [2] SD-Cards. Technical report, SD Card Association, SanDisk. <http://www.sdcard.com>.
- [3] Serial Peripheral Interface Bus. Technical report, Motorola. [http://en.wikipedia.org/wiki/Serial\\_Peripheral\\_Interface\\_Bus](http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus).
- [4] SPI. [http://www.mikrocontroller.net/articles/Serial\\_Peripheral\\_Interface](http://www.mikrocontroller.net/articles/Serial_Peripheral_Interface).
- [5] TI MSP430. <http://www.ti.com/msp430>.
- [6] TinyOS. <http://www.tinyos.net>.
- [7] TinyNode 584, Standard Extension Board. Technical report, Shockfish SA, 2006. <http://www.tinynode.com/>.
- [8] Jan Beutel, Roman Lim, and Mustafa Yücel. SIB - Sensor Interface Board. Technical report, ETH Zurich, 2008.
- [9] Microsoft Corporation. *Microsoft Extensible Firmware Initiative FAT32 File System Specification, rev. 1.03*, 2000. <http://www.microsoft.com/whdc/system/platform/firmware/fatgen.mspx>.
- [10] The Sensor Network Museum. The Sensor Network Museum - SHIMMER, 2007. <http://www.btnode.ethz.ch/Projects/SHIMMER>.
- [11] Stefan Schauer and Christian Speck. *Interfacing the MSP430 With MMC/SD Flash Memory Cards (Rev. B)*, 2005. <http://focus.ti.com/general/docs/techdocsabstract.tsp?abstractName=slaa281b>.
- [12] Martin Schwertfeger. SPI - Serial Peripheral Interface. Technical report, MCT Paul & Scherer Mikrocomputertechnik GmbH, 2006. <http://www.mct.de/faq/spi.html>.
- [13] The BTNode Team. *BTNode System Software*, 2008. <http://sourceforge.net/projects/btnode>.