Aufbau einer Testplattform für Energy Harvesting im Internet of Things

Michel Rottleuthner

HAW-Hamburg, Fakultät Technik und Informatik, Department Informatik,
Berliner Tor 7, 20099 Hamburg, Germany
michel.rottleuthner@haw-hamburg.de
http://informatik.haw-hamburg.de

Abstract. In dieser Arbeit wird die Entwicklung einer Testplattform zur Untersuchung von Energy Harvesting Systemen gezeigt. Gestützt durch Erkenntnisse aus verwandten Arbeiten wird ein abstraktes Konzept dargelegt, anhand dessen einsatzfähige Prototypen entworfen werden. Die Umsetzung umfasst dabei ein Energy Harvesting System und ein Modul zur Messung des Energieverbrauchs. Das Energy Harvesting System basiert auf einer Photovoltaik Zelle und einem Superkondensator. Zur Vereinigung dieser Bauteile wird ein Lademodul aufgebaut, das die flexible Einstellung der *Maximum Power Point*- und Ladeschlussspannung erlaubt. Das Messmodul wird für einen umschaltbaren Messbereich ausgelegt und für dessen Verwendung wird ein RIOT-Treiber implementiert. Zum Aufzeichnen der Messdaten wird ein Treiber für SD-Karten entwickelt. Die einfache Verwendung dieses Speichermediums und eine interoperable Möglichkeit zum Datenaustausch wird durch ein neu integriertes FAT-Dateisystem erreicht. Durch erste Tests wird die Funktionalität der Prototypen überprüft und es werden mögliche Optimierungen aufgezeigt. Abschließend werden Themen für die weitere praktische Verwendung der entwickelten Testplattform genannt.

1 Einleitung

Das Internet of Things (IoT) kann als Metapher für eine intelligente, vollständig vernetzte Welt verstanden werden. Im Themengebiet der IoT-Technologien wird ein erheblicher Einfluss auf Bereiche wie Transport, Energie, Gesundheit, Militär und allgemein dem Monitoring von Umwelt und urbanen Umgebungen erwartet [1]. Dieses breite Feld an Einsatzmöglichkeiten zeigt die Relevanz des Forschungsbereichs, welche sich auch in den Zahlen zum wirtschaftlichen Wachstum dieser Sparte widerspiegelt. Das Marktforschungsunternehmen Gartner prognostiziert für das Jahr 2017 eine Anzahl von 8.4 Milliarden vernetzten Geräten in Verwendung, welche bis zum Jahr 2020 auf 20.4 Milliarden steigen sollen [2]. Weiterhin platziert Gartner *Short-Range* und *Wide-Area* Low-Power Netzwerke innerhalb der wichtigsten zehn Technologien des IoT, für die Jahre 2017 und 2018 [3].

Jedes der allgegenwärtigen, vernetzten Geräte muss mit Energie versorgt werden und bietet somit das Potenzial, dabei auf neue, effektive und nachhaltige Wege zur Energieversorgung zurückzugreifen.

Der anschließende Teil der Arbeit ist in fünf Abschnitte gegliedert. In Abschnitt 2 wird zunächst ein Überblick zum Themengebiet Energy Harvesting gegeben. Dabei werden Forschungsthemen aufgezeigt und der Kontext der Arbeit dargestellt. Das Ziel der Arbeit wird in Abschnitt 3 definiert und mit einem Konzept für die geplante Testplattform konkretisiert. In Abschnitt 4 wird die Auswahl einzelner Bauteile für die Energieversorgung und die Entwicklung eines Lademoduls geschildert. Dem selben Aufbau folgend, werden in Abschnitt 5 diese Entwicklungsschritte ebenfalls für ein Modul zur Messung des Energieverbrauchs von Sensorknoten beschrieben. In Abschnitt 6 wird die Auswahl und Entwicklung von Teilkomponenten behandelt, die zur Aufzeichnung der Daten des Messmoduls benötigt werden. Beendet wird die Arbeit mit einer Zusammenfassung der behandelten Themen und einem Ausblick auf zukünftige Arbeiten, die von der Verwendung der Testplattform profitieren können.

2 Energy Harvesting: Ein Überblick

Energy Harvesting (EH) beschreibt einen Prozess, bei dem Energie aus der Umgebung „geerntet“und angesammelt wird. Es stellt eine praktikable Lösung zur unabhängigen Energieversorgung von kleinen eingebetteten Systemen wie kabellosen Sensorknoten dar. Dabei wird je nach Anwendung auf Energiequellen mit potentiell sehr unterschiedlicher Leistung zurückgegriffen, um einen wartungsarmen Dauereinsatz ohne Batteriewechsel zu erreichen [4].

Alternative Ansätze wie Thermoelektrizität, kinetische Energie aus menschlichen Bewegungen [5,6], Vibrationen oder Funkwellen [7] finden praktische Verwendung. Aber auch bewährte, erneuerbare Energiequellen wie Sonnen-, Wind- und Wasserkraft lassen sich im Kleinformat nutzen [8]. Eine häufige Problemstellung derartiger Systeme ist, dass die durch das Harvesting zur Verfügung stehende Energie zu gering für den Dauerbetrieb ist oder nicht permanent zur Verfügung steht. Dadurch entsteht ein Bedarf an Energieverwaltungsmechanismen, die unter Anwendung verschiedener Optimierungsstrategien versuchen, eine möglichst sinnvolle Nutzung der zur Verfügung stehenden Energiemenge zu erreichen. Ist es unter ordnungsgemäßem Betrieb des Systems möglich, höchstens die Menge an Energie zu verbrauchen, die vom System selbst umgewandelt und gesammelt werden kann, spricht man von einem Energy Neutral Operation (ENO) Zustand [9]. Kann ein System verlässlich im ENO Zustand betrieben werden, muss dieses gegenüber rein batterie- oder akkubetriebener Systeme seltener oder im Optimalfall nie gewartet werden, um den Energiespeicher zu wechseln.

Sudevalayam und Kulkarni [10] geben einen Überblick zu verschiedenen EH-Systemen und deren Konzepten. Als bevorzugte Energiequelle sehen die Autoren die Sonne bzw. Photovoltaik, da diese einfach verfügbar, kostengünstig und problemlos skalierbar ist. Die jeweils verwendeten Energiespeicher SLA, NiCd, NiMH, Li-ion, Li-polymer und Superkondensatoren werden einander gegenübergestellt und verglichen. Zusammenfassend lässt sich aus dem Vergleich ableiten, dass Lithium Akkus den anderen chemischen Akkus vorgezogen werden, was allerdings mit einer höheren Komplexität des Ladesystems einhergeht.

Superkondensatoren bieten die Vorteile einer weitaus höheren Lebensdauer, eines einfachen Ladeverfahrens und einer hohen physikalischen Robustheit. Die Autoren argumentieren für eine Überlegenheit der Superkondensatoren gegenüber herkömmlichen Akkus, wenn für den Anwendungsfall die niedrigere Energiedichte ausreicht und außerdem die zuvor genannten Vorteile relevant sind. Bezüglich des Ladereglers resümieren die Autoren, dass hierzu eine Lösung in Hardware einer Softwaresteuerung vorzuziehen ist, um garantieren zu können, dass nach einer vollständigen Entladung das System wieder anläuft. Als Kernthemen von EH-Systemen werden Performanz-Adaptionen und die Vorhersage von Energieverfügbarkeit genannt. Adaptionen für EH-Systeme gliedern die Autoren wiederum in die Felder *Node-level Adaptions* und *Network-level Design*. *Node-level Adaptions* beeinflussen dabei Duty-Cycling, Sendeleistung, Zuverlässigkeit der Messung und das Scheduling von Übertragungen, während zu *Network-level Design* Routing, Clustering, Data Collection, Knotenredundanz und MAC-Layer Optimierungen aufgezählt werden. Allgemein kann aus der Arbeit entnommen werden, dass es mit EH-Systemen unter Anwendung der genannten Optimierungen möglich ist, die beiden in Konflikt stehenden Designziele *Lebensdauer* und *Performanz* zu adressieren.

Shaikh und Zeadally vergleichen in ihrer Arbeit [8] ebenfalls verschiedene EH-Konzepte, gehen dabei aber detaillierter auf existierende Verfahren zur Vorhersage der Energieverfügbarkeit ein und kritisieren insbesondere deren bisher hohe Fehleranfälligkeit. Ein Vergleich verschiedener Energiequellen zeigt auch hier die Vorzüge von Photovoltaik (PV)-Zellen. Abgesehen von der Verwendung im Freien, verweisen die Autoren auch explizit auf die Möglichkeiten PV-Zellen im Innenbereich zu verwenden, wenn dieser gut ausgeleuchtet ist. Als Herausforderungen für zukünftige Forschungsarbeiten nennen sie generische Harvester, Miniaturisierung, Protokoll-Adaptionen, effiziente und weniger fehleranfällige Vorhersagen, die Entwicklung von Simulationsumgebungen und die weitere Untersuchung von Energiespeichern und der Zuverlässigkeit von EH-Systemen. Als langfristiges Ziel für EH-Sensornetze sehen die Autoren einen Paradigmenwechsel von Batteriebetrieb hin zu autonomen Lösungen welche ihre Energie ausschließlich aus ihrer Umwelt extrahieren.

Die Arbeitsgruppe iNET[1] der HAW-Hamburg gestaltet durch die Mitentwicklung des IoT-Betriebssystems RIOT [11] die Zukunft des IoT aktiv mit und bietet damit ein passendes Umfeld für aktuelle Forschungsarbeiten in diesem Bereich. Die von RIOT als Betriebssystem anvisierten Plattformen können potentiell erheblich von Energy Harvesting profitieren, wodurch weitere Einsatzgebiete und Anwendungsfälle erschlossen werden können. Wie die oben dargelegten Arbeiten veranschaulichen, sind die Probleme und Fragestellungen in EH-Systemen sehr vielseitig. Sensorknoten in Form eines zuverlässigen EH-Systems zu realisieren ist aufgrund der vielen Einflussfaktoren komplex und erfordert daher ein genaues Bild über das Verhalten einzelner Systemkomponenten. Um ein solches Bild zu erstellen, benötigt man eine flexible Versuchsinfrastruktur, die einem

[1] `https://inet.haw-hamburg.de/`

erlaubt in einem realistischen Szenario neue Verfahren zu testen und belastbare Messwerte zu erheben.

3 Testplattform für EH-Systeme

Das Ziel dieser Projektarbeit ist es, eine flexible Testplattform für EH-Systeme zu entwickeln. Diese soll dazu verwendet werden, realistische Versuche mit EH-Systemen durchzuführen um daraus weitere Erkenntnisse zu gewinnen. Für diesen Zweck soll die Testplattform ein eigenes EH-System und ein Modul zur Messung und Aufzeichnung von Energieverbrauchsmetriken beinhalten. In Zukunft soll dies dazu beitragen, vorhandene und künftig entwickelte Energieverwaltungsmechanismen in RIOT genauer untersuchen und vergleichen zu können, um anschließend RIOT-basierte EH-Systeme zu optimieren.

3.1 Konzept

Nachfolgend wird das Konzept für die geplante Testplattform beschrieben. Abbildung 1 zeigt schematisch deren Aufbau. Das EH-System wandelt mithilfe des Energiewandlers die Energie der Quelle in elektrische Form um. Eine Ladeelektronik auf Basis eines *Direct Current-Direct Current*(DC-DC)-Wandlers speichert diese Energie im Energiespeicher, aus dem sie wiederum über einen DC-DC-Wandler einer Mikrocontroller Unit (MCU) zur Verfügung gestellt wird. Die MCU stellt in diesem Fall einen Sensorknoten samt Funktransmitter dar. Zusätzlich existiert eine Monitoring-Komponente, die den Energieverbrauch an verschiedenen Stellen im System messen und aufzeichnen kann. Je nach Anwendungsszenario können Monitoring- und Sensorsystem sowohl als dedizierte Systeme, als auch kombiniert in Form eines einzelnen Systems aufgebaut werden. Für die konkrete Umsetzung werden aufgrund der in Abschnitt 2 genannten Vorteile, die Sonne als Energiequelle und eine PV-Zelle als Energiewandler verwendet. Als Energiespeicher wird ein Superkondensator eingesetzt. Bezüglich der MCU-Boards werden abgesehen von der RIOT-Kompatibilität keine allgemeinen Einschränkungen gemacht. Tests werden auf den Plattformen *remote-revb*[2] und *samr21-xpro*[3] durchgeführt.

4 Energieversorgung

In diesem Abschnitt werden wesentliche Aspekte der Teilkomponenten für die Energieversorgung des EH-Systems behandelt.

[2] `https://github.com/RIOT-OS/RIOT/wiki/Board%3A-Zolertia-remote`
[3] `https://github.com/RIOT-OS/RIOT/wiki/Board%3A-Samr21-xpro`

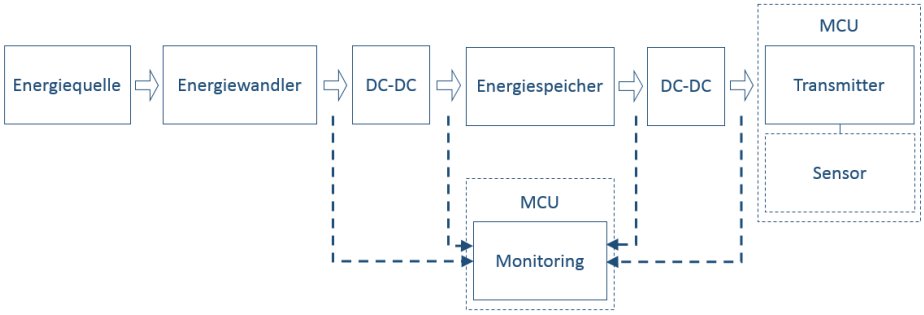

Abb. 1: Schematischer Aufbau eines EH-Systems mit Monitoring Komponente

4.1 Superkondensator und DC-DC-Konverter des Sensorknotens

Im Folgenden werden die grundlegenden Eigenschaften des verwendeten Superkondensators beschrieben und wie diese sich auf den MCU-seitigen DC-DC-Konverter auswirken. Abbildung 2 zeigt das verwendete Modell vom Hersteller Samwha. Es besitzt eine Kapazität von 100 F bei einer Nennspannung von 2,7 V. Nach Gleichung 1 kann damit eine Energiemenge von 364,5 W s gespeichert werden.

Abb. 2:
Superkondensator

$$E_1 = \frac{1}{2} C \cdot U^2 \qquad (1)$$

$$E_2 = \frac{1}{2} C \cdot (U^2 - {U_{min}}^2) \qquad (2)$$

Da die Spannung an einem Kondensator abfällt, sobald Energie daraus entnommen wird, kann nur ein Teil der Kapazität tatsächlich verwendet werden. Die tatsächlich verwertbare Energie eines voll geladenen Kondensators wird somit direkt durch die minimale Eingangsspannung des zweiten DC-DC Konverters (in Abbildung 1 rechts) bestimmt. Gleichung 2 liefert die verwertbare Energie in Abhängigkeit von Nennspannung U und Minimalspannung U_{min} des Superkondensators.

Je nach Betriebsspannung und Energieverbrauch der verwendeten MCU und der Effizienz des eingesetzten Konverters kann der Wert für U_{min} stark variieren. Mit einem einfachen, handelsüblichen Konverter (ME2108A33) mit fester Ausgangsspannung von 3,3 V und einem Verbraucher in Form eines SAM R21 Xplained Pro Boards lässt sich experimentell eine Untergrenze bestimmen. Bei einer Belastung von durchschnittlich 40 mA durch einen angeschlossenen Temperatursensor zur Messung und einem SPI-Display zur Anzeige, lag die Untergrenze für den erfolgreichen Betrieb des Systems bei knapp unter 1 V. Für weitere Überlegungen in Bezug auf die verfügbare Energiemenge wird diese Untergrenze angenommen, womit die verwertbare Energiemenge E_2 bei 314,5 W s liegt. Auf den Einsatz eines Konverters der geringere Eingangsspannungen erlaubt, wird vorerst verzichtet. Hierfür sind mehrere Gründe zu nennen. Bei weiter sinkender Spannung wird die Effizienz bei der Entnahme der Energie, als auch beim Wiederaufladen, zunehmend schlechter. Weiterhin verbleiben bei 1 V Ladespannung lediglich ca. 13,7 % der Energie im Kondensator, da sich die Spannung quadratisch auf die Energie im Kondensator auswirkt.

4.2 Ladeelektronik

Auf die Konzeption der Ladeelektronik wirkt sich neben dem Superkondensator auch maßgeblich die Auswahl von PV-Zellen als Energiewandler aus. So muss sie die Möglichkeit bieten, die Eingangspannung auf den Maximum Power Point (MPP) der PV-Zelle einzustellen um eine möglichst hohe Effizienz zu erreichen. Da in späteren Versuchen potentiell unterschiedliche PV-Zellen eingesetzt werden, soll diese Einstellung variabel sein. Weiterhin sollte die Ladeelektronik bis zu einer möglichst niedrigen Spannung betrieben werden können, um auch bei relativ schlechten Witterungsverhältnissen oder in der Dämmerung Energie aus der PV-Zelle extrahieren zu können. Die Ladeschlussspannung sollte flexibel einstellbar sein, um nötigenfalls andere Energiespeicher, wie Lithium Akkus verwenden zu können. Für den Aufbau der Ladeelektronik mit den zuvor genannten Komponenten und deren Eigenschaften eignet sich der LTC3105 von Lienar Technologies. Für den Entwurf werden anhand des Datenblattes [12] die benötigten Bauteilgrößen zur Beschaltung des LTC3105 berechnet. Die Ausgangsspannung des Bausteins wird über einen Spannungsteiler konfiguriert und ist nach Gleichung 3 zu berechnen.

$$V_{OUT} = 1{,}004\,\mathrm{V} \cdot \left(\frac{R1}{R2} + 1\right) \tag{3}$$

Um die Schaltung (siehe Abbildung 3) an dieser Stelle flexibel konfigurierbar zu halten, wird zusätzlich zwischen den beiden festen Widerständen R1 und R2 ein Potentiometer mit 500 kΩ (R3) platziert.

Mit R1=1,1 MΩ und R2=390 kΩ ergibt sich ein einstellbarer Spannungsbereich zwischen 2,24 V und 5,12 V. Die MPP-Spannung wird nach Gleichung 4 über einen einzelnen Widerstand (R_{MPPC}) festgelegt, welcher ebenfalls als Potentiometer ausgeführt ist und mit einem Wert von 500 kΩ die Einstellung im Bereich unterhalb von 5 V ermöglicht.

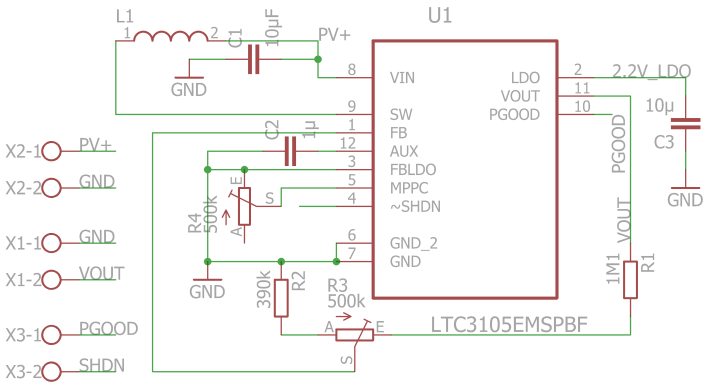

Abb. 3: Schaltplan des Ladereglers auf Basis des LTC3105

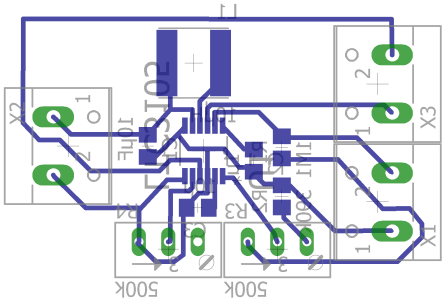

Abb. 4: Platinenlayout des Ladereglers auf Basis des LTC3105

$$V_{MPPC} = 10\,\mu A \cdot R_{MPPC} \tag{4}$$

Als Spule wird entsprechend dem Datenblatt eine niederohmige (44 mΩ) Speicherdrossel mit einer Induktivität von 10 µH verwendet. Die resultierende Beschaltung des Bausteins ist Abbildung 3 zu entnehmen, welche den entworfenen Schaltplan für die Ladeelektronik zeigt. Zum Entwerfen von Schaltplan und Platinenlayout (Abbildung 4) wird die Software EAGLE[4] verwendet. Der Prototyp der Platine wird eigens mittels Fotolithografie-Verfahren hergestellt, auf welches im Rahmen dieser Arbeit nicht weiter eingegangen wird. Abbildung 5 zeigt den Prototypen mit der Seite der Anschlussklemmen und den Potentiometern zum Einstellen der Ausgangs- sowie MPP-Spannung. Zusätzlich zum Ein- und Ausgang für die PV-Zelle (IN-, IN+) und den Superkondensator (OUT-, OUT+) verfügt das Modul über einen Eingang zum Aktivieren und Deaktivieren der Schaltung (SHDN) und einen Ausgang der signalisiert, ob der Ausgang im Rahmen der Toleranz auf die Zielspannung geregelt wird.

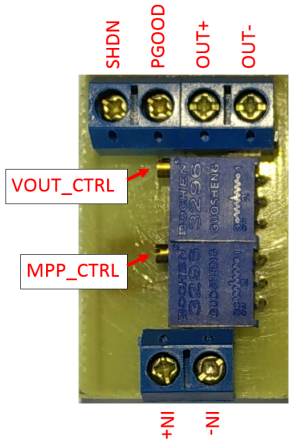

Abb. 5: Prototyp des Ladereglers

5 Messmodul

Das Messmodul soll dazu genutzt werden möglichst genaue Daten zum Energieverbrauch des Systems zu messen. Zu diesem Zweck wird eine Schaltung aufgebaut, die für ein permanentes Monitoring des EH-Systems verwendet werden kann. Diese Schaltung soll an einer MCU (Monitoring-Komponente in Abbildung 1) betrieben werden, welche die Messergebnisse kontinuierlich aufzeichnet. Die Anforderungen an das Messmodul werden nachfolgend angeführt. Der Messbereich soll konfigurierbar sein, um bei verschiedenen Mikrocontrollern mit unterschiedlicher Leistungsaufnahme eine möglichst hohe Auflösung zu erreichen. Für eine hohe Genauigkeit soll es außerdem möglich sein, die Schaltung über ein Referenzmessgerät zu kalibrieren. Weiterhin soll es möglich sein, das Messmodul durch eine eigene Spannungsquelle zu versorgen, um nötigenfalls den Energiebedarf des Moduls von dem überwachten System zu trennen.

Einen für diesen Einsatzzweck geeigneten Baustein bietet Texas Instruments in Form des INA226 an, der gleichzeitig die Spannung und den Strom (über den Spannungsabfall an einem Shunt-Messwiderstand) an einer angeschlossenen Last überwachen kann. Der Baustein erfüllt die zuvor genannten Anforderungen und kann über eine Inter-Integrated Circuit (I^2C) Schnittstelle an einen Mikrocontroller angebunden werden. Die entworfene Schaltung bietet die Möglichkeit die

[4] `http://www.cadsoft.de/eagle-pcb-design-software/ueber-eagle/`

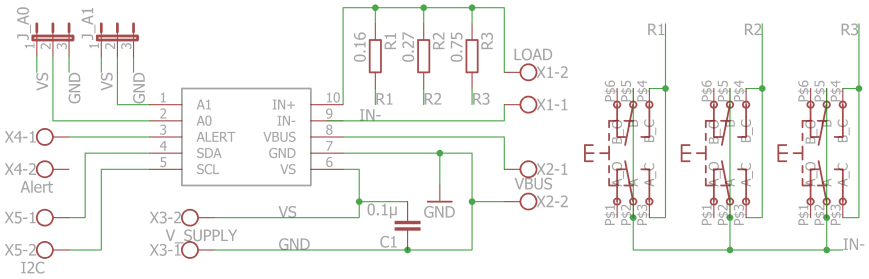

Abb. 6: Schaltplan des Messmoduls auf Basis des INA226

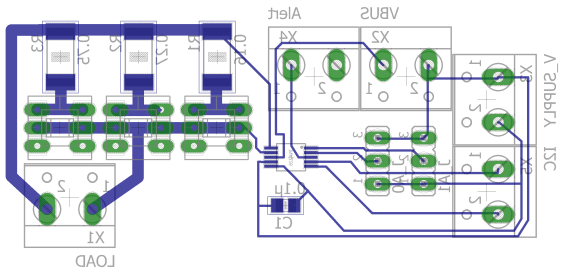

Abb. 7: Platinenlayout des Messmoduls auf Basis des INA226

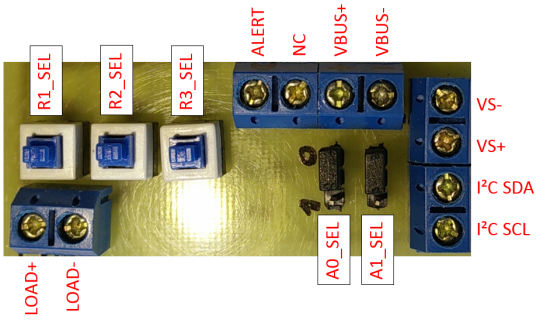

Abb. 8: Prototyp des Messmoduls

I²C-Adresse des Bausteins mit Jumper-Brücken einzustellen, um so mehrere baugleiche Module gleichzeitig an einer MCU betreiben zu können. Der Messbereich ist über Schalter einstellbar, die zwischen den verschiedenen Messwiderständen auswählen. In Abbildung 6 ist die Bestückung mit R1=0,16 Ω, R2=0,27 Ω und R3=0,75 Ω zu sehen. Durch den maximalen Spannungsabfall von 81,92 mV am Messwiderstand (siehe „Shunt voltage input range“ im Datenblatt [13]), ergeben sich einstellbare Messbereiche bis jeweils knapp über 500 mA, 300 mA und 100 mA. Um größere oder kleinere Messbereiche zu unterstützen müssen lediglich die Messwiderstände entsprechend ersetzt werden. Für die Verwendung des Messmoduls unter dem Betriebssystem RIOT wird ein Low-Level-Treiber und eine Messanwendung entwickelt.

Test und Optimierung Ergebnisse von ersten Tests mit dem Messmodul zeigen, dass dieses grundsätzlich ordnungsgemäß funktioniert und nach Kalibrierung mit einem Referenzmessgerät keine Unregelmäßigkeiten zeigt. Ein Problem besteht jedoch in der physikalischen Empfindlichkeit der Schalter für die einzelnen Messwiderstände. Werden diese leicht angedrückt oder Vibrationen ausgesetzt, meldet das Messmodul abweichende Werte und es wird eine erneute Kalibrierung notwendig. Der Grund dafür liegt bei dem Innenwiderstand des Schalters, der sich bereits bei geringer Bewegung der innen liegenden Schaltkontakte verändert. Da sich dieser veränderliche Widerstand mit dem Messwiderstand in einer Reihenschaltung befindet werden die Messwerte direkt beeinflusst. Im Rahmen einer zukünftigen Arbeit sollen zur Lösung des Problems robustere Schiebeschalter getestet werden, die laut Spezifikation zusätzlich einen sehr geringen Innenwiderstand garantierten.

6 Aufzeichnung der Messdaten

Ein Einsatzzweck der Testplattform ist es, unter möglichst authentischen Bedingungen Feldtests mit Sensorknoten durchführen zu können. Dazu ist eine Komponente für das Protokollieren der vom Messmodul geliefert Daten erforderlich. Aufgrund des eingeschränkten Volumens des internen Speichers einer klassischen MCU, wird hierzu ein externes Speichermedium benötigt. Softwareseitig soll für das Speichermedium ein RIOT-Treiber implementiert werden, der die Verwendung des Mediums auf allen Plattformen erlaubt, die entsprechende Unterstützung seitens der Hardware bieten.

6.1 Speichermedium

In diesem Abschnitt wird die Auswahl eines geeigneten Speichermediums erläutert. Hierzu werden die folgenden Anforderungen an das Medium betrachtet:

- hohe Kompatibilität zu diversen MCUs
- Speichervolumen für mehrtägige Messungen
- geringer Energieverbrauch
- interoperabler Datenzugriff

Ein potentiell geeignetes Medium unter diesen Gesichtspunkten stellen Speicherkarten im SD-Format dar. Um das weiter zu eruieren werden die diesbezüglichen Eigenschaften von SD-Karten genauer beschrieben.

SD-Karten unterstützen neben dem nativen SD-Bus einen SPI-Modus, der zwar langsamer ist, jedoch ohne dedizierte SD-Peripherie des Mikrocontrollers auskommt. Dadurch besteht eine sehr hohe Kompatibilität zu diversen Mikrocontrollern. Eine Überprüfung von 64 RIOT-kompatiblen Zielplattformen anhand der momentan aktuellen RIOT Version *2017.01* zeigt, dass bereits 44 davon eine SPI-Implementierung besitzen und damit potentiell kompatibel sind. Auf den anderen Plattformen besteht die Möglichkeit eine SPI-Implementierung in Form von Software durch manuelles Ansteuern der Pins bereitzustellen. Einzig fünf Plattformen verbleiben, welche aufgrund fehlender GPIO-Treiber weder mittels Hardware- noch Software-SPI verwendet werden können.

Das Speichervolumen beträgt derzeit im kleinsten Formfaktor *microSD* bis zu 256 GB[5] und bietet damit auch für Langzeitmessungen mit erhöhtem Datenaufkommen und kurzen Messintervallen eine ausreichende Kapazität.

Der Energiebedarf von SD Karten lässt sich nicht eindeutig aus der Spezifikation entnehmen. Es wird explizit genannt, dass der Energieverbrauch im Energiesparmodus nicht spezifiziert ist und zwischen verschiedenen Produkten variieren kann. Es werden lediglich Obergrenzen von einer Sekunde für das Zurückkehren aus dem Energiesparmodus und eine maximale Leistungsaufnahme von 0,36 W während des Schreibvorgangs im Default Speed Mode genannt. Diese Werte sagen zwar wenig über den tatsächlichen Energieverbrauch aus, erlauben aber vorab eine stark vereinfachte Worst-Case Abschätzung. Für die Abschätzung werden des weiteren die folgenden, pessimistisch ausgelegten Annahmen getroffen:

- Leistungsaufnahme stets: 0,36 W
- Dauer der Karteninitialisierung: 2 Sekunden
- Netto Schreibgeschwindigkeit: 1000 Bytes/Sekunde
- Logging-Intervall: 1 Minute
- Größe eines Log-Eintrags: 50 Bytes
- Größe des Puffers: 500 Bytes

Unter den genannten Annahmen liegt der Energieverbrauch pro Tag bei höchstens 130 W s, was mit Blick auf die verwertbare Kapazität des Superkondensators von 314,5 W s (siehe Unterabschnitt 4.1) in einer realisierbaren Größenordnung liegt. Als Arbeitspaket für das nachfolgende Hauptprojekt wird

[5] https://www.sandisk.de/about/media-center/press-releases/2016/western-digital-launches-worlds-fastest-256gb-microsd-card-Broadens-sandisk-memory-card-portfolio-with-new-high-capacity-solutions

eine Überprüfung der implementierten Lösung bezüglich der obigen Annahmen angesetzt.

Kartenleser für das SD-Format gehören heute bei vielen Computern oder gar Smartphones zur Standardausstattung und können nötigenfalls problemlos über externe Adapter bereitgestellt werden. Der interoperable Datenzugriff über einen PC wird dadurch mit einer einfachen Handhabung ermöglicht.

6.2 Implementierung Treibers: sdcard_spi

Nachfolgend wird die Implementierung des Treibers sdcard_spi behandelt. Dazu werden einige grundlegende Funktionen von SD-Karten und deren Umsetzung im Treiber beschrieben. Zusätzlich zum Treiber wird eine Testapplikation implementiert, die Entwicklern erlaubt den Treiber auf verschiedenen Zielplattformen zu testen.

Wie bereits erwähnt, kommunizieren SD-Karten nativ über den SD-Bus. Wird eine Karte eingeschaltet, befindet sie sich standardmäßig im SD-Modus. Um mit der Karte über SPI zu kommunizieren, muss diese zunächst in den SPI-Modus versetzt werden. Dazu ist in der SD Spezifikation eine Power-Up-Sequenz vorgesehen. Einige SD-Karten lassen sich nur dann erfolgreich in den SPI-Modus versetzen, wenn der MISO-Pin mit einem Pull-Up-Widerstand versehen ist. Dieser muss mindestens ab beginn der Power-Up-Sequenz bis zum erfolgreichen Software-reset aktiviert sein. Anschließend befindet sich die Karte im SPI-Modus und arbeitet streng mit Push-Pull Ansteuerung, wodurch der Widerstand nicht mehr benötigt wird. Auf MCU-Boards mit fest verbautem microSD-Slot, bei denen dieser Umstand nicht im Entwurf der Hardware berücksichtigt wurde, kann im Regelfall kein externer Pull-Up-Widerstand nachgerüstet werden. Die Kompatibilität des Treibers zu problematischen SD-Karten auszuschließen, soll, aufgrund der in Unterabschnitt 6.1 genannten Anforderungen bezüglich Kompatibilität und Interoperabilität vermieden werden. Eine einfache Lösung aus Sicht der Schnittstellen bietet die Erweiterung der Konfigurierbarkeit der Pull-Ups über die SPI-Schnittstelle. Jedoch unterstützt nicht jede Plattform die unabhängige Konfiguration von alternativen Funktionen der Pins (SPI) und deren Pull-Up Einstellungen. Zur Lösung wird daher entschieden die Initialisierung über eine Software-SPI Implementierung innerhalb des sdcard_spi Treibers abzuwickeln. Die Software-SPI Implementierung steuert die einzelnen Pins manuell über die GPIO-API an, welche die Konfiguration von internen Pull-Ups erlaubt. Die Software Variante wird nur während den ersten Schritten der Initialisierung verwendet, anschließend wird mittels Funktions-Pointer wieder auf die performantere Variante mit dedizierter SPI-Hardware umgeschaltet. Aus dieser Lösung ergibt sich der Nachteil, dass zur Initialisierung des Treibers statt eines einzelnen SPI-Enumerators alle Pins einzeln spezifiziert werden müssen, da eine Identifikation der jeweiligen Pins anhand des SPI-Enumerators bisher nicht plattformunabhängig möglich ist.

Auch die zum Testen verwendete remote-revb Plattform besitzt keinen fest verbauten Widerstand. Bei der vorhanden Revision B des Boards sind die Pins `PA6` und `PA7` zur Verwendung für die microSD-Karte vorgesehen. Diese wurden

bei der vorliegenden Produktionscharge fälschlicherweise zur Verwendung als ADC-Eingänge beschaltet. Um den microSD-Slot funktionstüchtig zu machen, sind zwei nachträgliche Brücken notwendig.[6]

Die Kommunikation mit SD-Karten basiert stets auf Kommandos, welche an die SD-Karte übertragen werden. Diese Kommandos können beispielsweise bestimmte Sequenzen einleiten, die Übertragung eines Registerwertes anweisen oder den Status einer Operation abfragen. Das Senden eines Kommandos wird in einer Methode gekapselt, welche blockiert, bis eine Rückmeldung der Karte empfangen wurde, oder es zu einem Timeout kommt.

Die Initialisierung der SD-Karte wird mit einer Finite-State-Machine (FSM) realisiert. Das Flussdiagramm in Abbildung 9 stellt die relevanten Fallunterscheidungen und den Ablauf der FSM dar. Bei der Initialisierung der Karte muss die Kommunikation mit einer Taktrate zwischen 100 kHz und 400 kHz durchgeführt werden. Alle Schritte, von der Konfiguration der verwendeten Peripherie (GPIO und Timer) im Zustand `SD_INIT_START`, bis zur Bereitstellung der Karte für anschließende Lese- und Schreiboperationen werden dabei von der FSM übernommen. Die zuvor erwähnte Power-Up-Sequenz wird im Zustand `SD_INIT_SPI_POWER_SEQ` ausgeführt. Am Ende des Zustands `SD_INIT_SEND_CMD_0` wird die Umschaltung von Soft- auf Hardware-SPI vorgenommen. Anschließende Kommandos werden dazu verwendet, die CRC-Funktion der Karte zu aktivieren, die ordnungsgemäße Kommunikation mit der Karte zu überprüfen und ihre Metadaten abzufragen, damit diese vom Treiber anschließend richtig angesprochen werden kann. Zum Abschluss wird die SPI-Bus Geschwindigkeit im Zustand `SD_INIT_SET_MAX_SPI_SPEED` auf die angegebene Maximalfrequenz von höchstens 50 MHz angehoben.

Lese- und Schreiboperationen des Treibers arbeiten block orientiert und verwenden gegebenenfalls automatisch die performantere Mehrfachübertragung für mehrere Blöcke hintereinander.[7]

Um die Schnittstelle übersichtlich zu gestalten, werden Low- und High-Level Operationen über zwei getrennte Header-Dateien bereitgestellt. Funktionen zur Initialisierung der Karte und zum Lesen und Schreiben von Blöcken werden in der Datei `drivers/include/sdcard_spi.h` deklariert. Funktionen zum Senden von Kommandos und zugehörige Literale befinden sich dagegen in der Datei `drivers/sdcard_spi/sdcard_spi_internal.h`.

Für einen detaillierteren Einblick sei an dieser Stelle auf den Quellcode im offiziellen git-Repository von RIOT verwiesen. Der im Rahmen dieser Arbeit entwickelte Treiber hat bereits das Community-Review durchlaufen und ist ab RIOT-Version 2017.01 Bestandteil des Betriebssystems.[8]

Test: sdcard_spi Um die Funktion des Treibers zu überprüfen, werden verschiedene microSD-Karten getestet. Die verwendeten Modelle sind in der folgenden Tabelle aufgelistet.

[6] `https://github.com/Zolertia/Resources/wiki/%5BRE-Mote%5D-Enabling-uSD`
[7] siehe *7.2.3 Data Read* und *7.2.4 Data Write* in [14]
[8] RIOT Release 2017.01 `https://github.com/RIOT-OS/RIOT/tree/2017.01-branch`

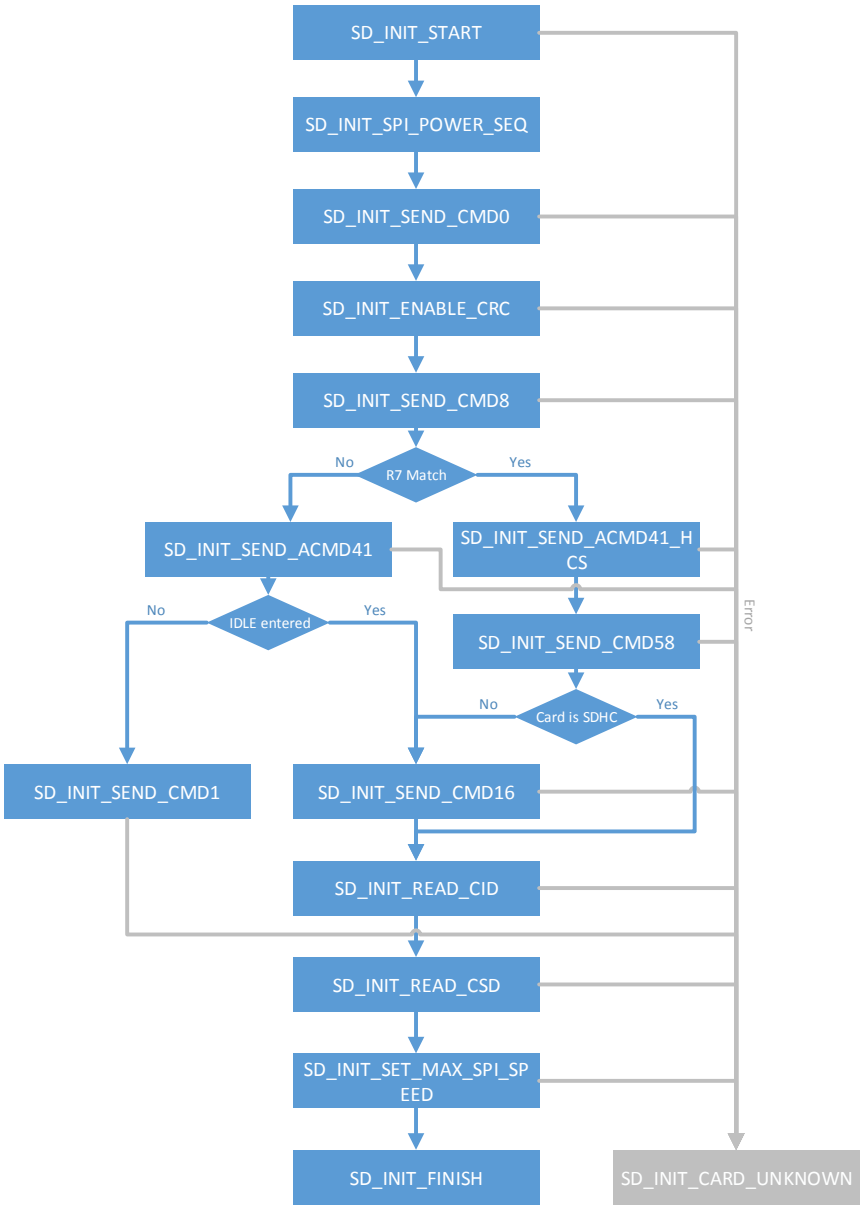

Abb. 9: Ablauf der Initialisierung der SD-Karte im sdcard_spi Treiber

Nr.	Hersteller	Modell	Kapazität	Typ
1	Kingston	SD-C01G	1GB	SDSC
2	SanDisk	SU02G	2GB	SDSC
3	OEM	SD	2GB	SDSC
4	Transcend	USD	16GB	SDHC
5	SanDisk	SL32G	32GB	SDHC
6	Samsung	EVO MB-MP32D	32GB	SDHC
7	SanDisk	SDSDQUAN-128G-G4	128GB	SDXC

Alle Karten werden unter Verwendung der implementierten Testapplikation auf den Plattformen *samr21-xpro* und *remote-revb* getestet. Zunächst wird mit den Kommandos init, csd und cid überprüft, ob die Initialisierung erfolgreich ausgeführt wird. Durch einen Abgleich der CID und CSD-Registerwerte mit Referenzwerten, welche über einen Kartenleser ausgelesen werden, kann überprüft werden, ob diese richtig übertragen werden. Anschließend wird mit den Kommandos write, read und copy getestet, ob Lese und Schreibvorgänge ordnungsgemäß ausgeführt werden. Während dem Test treten bei den Karten des OEM-Herstellers und Kingston teilweise Kommunikationsfehler aufgrund von Timeouts auf. Ein Erhöhen der Timeout-Parameter behebt dieses Problem. Abschließend kann in keinem dieser Tests ein fehlerhaftes Verhalten hervorgerufen werden.

Optimierungen Nachfolgend werden einige Optimierungen genannt, mit denen der Treiber in Zukunft verbessert werden kann. Die genannten Punkte sollten auch bei der Konzipierung von Anwendungen berücksichtigt werden, die den Treiber verwenden - insbesondere wenn Energieverbrauch und Performanz von erhöhter Wichtigkeit sind. Um den durchschnittlichen Energieverbrauch zu minimieren, können verschiedene Optimierungen angewandt werden. Eine Optimierung besteht darin, die Karte in den Energiesparmodus zu versetzen oder sie vollständig abzuschalten und nur für tatsächliche Schreibvorgänge zu aktivieren. Weiterhin sollte der SPI-Bus in dem schnellstmöglichen Modus verwendet werden, um die aktiven Phasen so kurz wie möglich zu halten. Die Daten sollten im RAM gepuffert werden und nur auf die Speicherkarte geschrieben werden, wenn eine bestimmte Datenmenge erreicht wurde. Um die Karte bezüglich ihrer Lebensdauer zu schonen und den Overhead für den Controller innerhalb der Karte zu minimieren, sollte diese Datenmenge einem vielfachen der Sektorgröße entsprechen, die bei einem Schreibzugriff durch interne Vorgänge der Karte gelöscht wird. Je nach Version der SD-Karte wird diese Größe durch das Feld `,SECTOR_SIZE` im CSD-Register oder durch `AU_SIZE` im SD Status Register definiert und unterscheidet sich zwischen verschiedenen Karten. In der Praxis bewegen sich `SECTOR_SIZE` und `AU_SIZE` zwischen 512 Byte und 64 MB, womit das Puffern eines vollständigen Sektors im RAM der MCU in vielen Fällen nicht praktikabel ist. Je nach Anwendung kann es daher sinnvoll sein, bestimmte Kartenmodelle für den Einsatz auszuwählen. Für den Schreibvorgang sollte bevorzugt die *Multiple Block Write* Operation verwendet werden, um auch

den Overhead für die Datenübertragung gering zu halten. Die SD-Spezifikation nennt zur weiteren Optimierung des Durchsatzes beim Schreiben die Verwendung des `SET_WR_BLK_ERASE_COUNT` Kommandos (ACMD23) vor dem Schreiben mehrerer neuer Blöcke. Weiteres Optimierungspotential besteht bei der Festlegung der Timeouts für Kommandos. Die derzeitige Lösung verwendet als Timeout stets eine feste Anzahl an Versuchen, die unternommen werden, bevor ein Kommando als nicht erfolgreich interpretiert wird. Je nach Art des Kommandos steht dazu ein eigenes Literal zur Verfügung, das nötigenfalls angepasst werden kann. Versuche mit den in 6.2 gelisteten Karten legen nahe, dass diese sich zwischen verschiedenen Karten erheblich unterscheiden, was zu einer langsameren Übertragungsgeschwindigkeit führen kann. Zur weiteren Optimierung sollten die Timeouts dynamisch anhand der Karteneigenschaften festgelegt werden.

6.3 Dateisystem

Um die aufgezeichneten Messdaten flexibel weiterverarbeiten zu können, soll mit einem Dateisystem eine einfache und interoperable Möglichkeit zum Datenaustausch gegeben werden. Unter RIOT ist derzeit kein Dateisystem verfügbar. FAT ist ein Dateisystem, das auf SD-Karten eine hohe Verbreitung findet und sich auf kleinen Ressourcenarmen Mikrocontrollern einsetzen lässt. Eine Quelloffene FAT Implementierung mit GNU GPL kompatibler Lizenz ist das FatFs Generic FAT File System Module[9], welches unter anderem FAT und exFAT Unterstützung bietet. Weiterhin lässt sich das Modul abhängig von den benötigten Funktionen konfigurieren, um Beispielsweise einen geringeren Speicherbedarf zu erreichen. Nach Außen bietet das FatFs Modul dem Entwickler eine API, die sich an bekannten Linux Systemaufrufen orientiert, während Hardwareoperationen über eine vorgegebene Schnittstelle abstrahiert werden. Die darunterliegende Low-Level Implementierung ist in FatFs nicht enthalten und muss separat entwickelt werden. Dazu muss für alle Methodendeklarationen in der Datei `diskio.h` eine Implementierung bereitgestellt werden. Über diese Abstraktion können transparent verschiedene Speichermedien bereitgestellt werden.

Um Komponenten auf Basis von fremdem Quellcode nahtlos in den Build-Prozess von RIOT zu integrieren, existiert ein Package-Konzept. Ein Package enthält dabei eine Makefile und ggf. Patches, welche den Quellcode kompatibel zum RIOT Build-System machen. Die in der Makefile enthaltenen Schritte dienen dann dazu, den Quellcode von seiner Originalquelle zu beziehen, ihn in einer geeigneten Struktur im Quellcodeverzeichnis von RIOT abzulegen und die Patches auf den Code anzuwenden.

Zur Verwendung des FatFs Moduls werden zwei Low-Level Implementierungen entwickelt. Die erste Variante basiert auf dem Treiber sdcard_spi und ermöglicht damit direkt den Dateizugriff auf FAT-formatierten SD-Karten. Die zweite Variante ist für den native-Modus[10] von RIOT gedacht. Im native Modus

[9] `http://elm-chan.org/fsw/ff/00index_e.html`
[10] `https://github.com/RIOT-OS/RIOT/wiki/Family:-native`

werden Lese- und Schreibzugriffe auf ein Dateisystemabbild im Hostsystem umgesetzt. Die Systemaufrufe im Hostsystem werden dabei wie ein Speichermedium verwendet, das Block-orientiert arbeitet, damit dieses sich möglichst ähnlich zur sdcard_spi Variante verhält. Da das Dateisystemabbild in einer einzelnen Datei auf dem Hostsystem liegt, kann dieses mit Linux-Boardmitteln als virtuelles Laufwerk bereitgestellt werden. So können während der Entwicklungsphase Dateien, die später auf dem RIOT-System Verwendung finden einfach auf dem virtuellen Laufwerk abgelegt werden. Anwendungen die später eine SD-Karte als Datenspeicher verwenden sollen, können dadurch vorab mit allen Funktionen in der native-Umgebung implementiert und getestet werden, ohne für jeden Entwicklungsschritt auf die konkrete Hardware zurückgreifen zu müssen. Die unter native verwendete Dateisystemabbilder können nach fertigstellen der Implementierung direkt auf die SD-Karte übertragen werden. Welche der beiden Low-Level Implementierungen verwendet wird, wird beim Kompilieren automatisch anhand der ausgewählten Zielplattform entschieden.

Test: Dateisystem Um die Funktion des FatFs Moduls zu testen wird, wie bereits für den Treiber sdcard_spi, eine interaktive Testapplikation implementiert. Mit dieser können beispielsweise Dateien gelistet, gelesen oder geschrieben werden. Um zu überprüfen, ob dies ordnungsgemäß funktioniert, werden die Dateiinhalte nach verschiedenen Dateioperationen mit den Erwartungswerten abgeglichen. Weiterhin wird überprüft, ob Änderungen an den Dateien (über einem PC oder RIOT) auf dem jeweils anderen System übereinstimmend angezeigt werden. In einem abschließenden Versuch wird über mehrere Stunden hinweg das in Abschnitt 5 vorgestellte Messmodul mit einem Messintervall von 0,6 s betrieben. Dabei werden alle aufgezeichneten Messwerte in eine Comma-separated values (CSV) Datei auf die SD-Karte geschrieben. Im Rahmen dieser Tests konnte kein fehlerhaftes Verhalten hervorgerufen werden.

7 Zusammenfassung und Ausblick

In dieser Arbeit wurde die Entwicklung einer Testplattform für EH-Systeme gezeigt. Anhand der Auseinandersetzung mit verwandten Arbeiten wurden Forschungsthemen identifiziert, für deren weitere Untersuchung die Testplattform eingesetzt werden kann. Es wurde aufgezeigt, dass sowohl im Bereich der Optimierungen einzelner Knoten, als auch auf (Sensor-)Netzwerkebene noch vielfältige Fragestellungen zu bearbeiten sind. Weitere Arbeiten könnten Beispielsweise Vorhersagen zur Energieverfügbarkeit oder das Routing auf EH-Systemen verbessern. Vor diesem Hintergrund wurden mit einem Konzept die abstrakten Einzelkomponenten der Testplattform eingeführt und im Anschluss konkrete Bauteile dafür ausgewählt und entwickelt.

Für die Energieversorgung des EH-Systems wurde eine flexible Ladeelektronik aufgebaut, welche mit verschiedenen PV-Zellen betrieben einen Superkondensator auflädt. Sowohl für das entwickelte Messmodul, als auch den SD-Karten

Treiber wurde eine Softwareunterstützung für RIOT bereitgestellt. In Kombination mit dem FatFs Modul wurde außerdem eine interoperable Möglichkeit zur Übertragung der aufgezeichneten Datensätze geschaffen. Damit können zukünftig auf einem RIOT-basierten System stationäre Messungen und Feldtests durchgeführt werden. Erste Tests haben die ordnungsgemäße Funktion der Prototypen gezeigt und Ansatzpunkte für weitere Optimierungen geliefert.

Nachfolgende Arbeiten sollten sich schwerpunktmäßig mit der detaillierten Validierung, Verifikation und Integration der entwickelten Systemkomponenten beschäftigen. Hierzu sind umfangreiche Messungen mit Referenzmessgeräten und die Auswertung und Quantifizierung von Abweichungen erforderlich. Nachdem sichergestellt ist, dass das vorliegende Gesamtsystem ordnungsgemäß funktioniert, können erste Versuche mit der Plattform durchgeführt werden. Denkbar sind zunächst Untersuchungen von bereits vorhandenen Energiesparmechanismen in RIOT. Mithilfe der daraus gewonnenen Erkenntnisse kann abgewogen werden, in welchen Bereichen weiterer Optimierungsbedarf besteht, um zuverlässige und performante EH-Systeme mit RIOT zu entwickeln.

Literatur

1. Maia, Pedro ; Cavalcante, Everton ; Gomes, Porfírio ; Batista, Thais ; Delicato, Flavia C. ; Pires, Paulo F.: On the Development of Systems-of-Systems Based on the Internet of Things: A Systematic Mapping. In: *ECSAW '14: Proc. of the 2014 European Conference on Software Architecture Workshops*. New York, NY, USA : ACM, August 2014, S. 1–8
2. Gartner: Gartner Says 8.4 Billion Connected "Things"Will Be in Use in 2017, Up 31 Percent From 2016. (2017), February. `http://www.gartner.com/newsroom/id/3598917`. – Version: 22.02.2017
3. Gartner: Gartner Identifies the Top 10 Internet of Things Technologies for 2017 and 2018. (2016), February. `http://www.gartner.com/newsroom/id/3221818`. – Version: 22.02.2017
4. Casas, Maria G.: *Transmission strategies for wireless energy harvesting nodes* , Universitat Politècnica de Catalunya, Departament de Teoria del Senyal i Comunicacions, Diss., June 2014. `http://upcommons.upc.edu/handle/2117/95379`
5. Xie, L. ; Cai, M.: An In-shoe Harvester with Motion Magnification for Scavenging Energy from Human Foot Strike. 20 (2015), December, S. 3264–3268
6. Gorlatova, Maria ; Sarik, John ; Cong, Mina ; Kymissis, Ioannis ; Zussman, Gil: Movers and Shakers: Kinetic Energy Harvesting for the Internet of Things. 33 (2015), January, S. 1624–1639
7. Chambe, P. ; Canova, B. ; Balabanian, A. ; Pele, M. ; Coeur, N.: Optimization of Energy Harvesting Systems for RFID Applications. 8 (2014), Nr. 7, S. 1147–1150
8. Shaikh, Faisal K. ; Zeadally, Sherali: Energy harvesting in wireless sensor networks: A comprehensive review. In: *Renewable and Sustainable Energy Reviews* 55 (2016), S. 1041–1054
9. Le, Trong N. ; Pegatoquet, Alain ; Berder, Olivier ; Sentieys, Olivier: A Power Manager with Balanced Quality of Service for Energy-Harvesting Wireless Sensor Nodes. In: *ENSsys'14: Proceedings of the 2Nd International Workshop on Energy Neutral Sensing Systems*. New York, NY, USA : ACM, November 2014, S. 19–24

10. Sudevalayam, Sujesha ; Kulkarni, Purushottam: Energy harvesting sensor nodes: Survey and implications. 13 (2011), S. 443–461
11. Baccelli, Emmanuel ; Hahm, Oliver ; Günes, Mesut ; Wählisch, Matthias ; Schmidt, Thomas C.: RIOT OS: Towards an OS for the Internet of Things. In: *Proc. of the 32nd IEEE INFOCOM. Poster*. Piscataway, NJ, USA : IEEE Press, 2013
12. Linear Technology: *LTC3105 Datasheet*, 2015. `http://cds.linear.com/docs/en/datasheet/3105fb.pdf`
13. Instruments, Texas: *INA226 High-Side or Low-Side Measurement, Bi-Directional Current and Power Monitor with I2C Compatible Interface, SBOS547A*, 2015. `http://www.ti.com/lit/ds/symlink/ina226.pdf`
14. Association, SD C.: *SD Specifications Part 1 Physical Layer Simplified Specification Version 5.00* , 2015. `https://www.sdcard.org/downloads/pls/pdf/part1_500.pdf`