

Energy Harvesting Bicycle Computer



**ZÜRCHER HOCHSCHULE
FÜR ANGEWANDTE WISSENSCHAFTEN**

INSTITUTE OF EMBEDDED SYSTEMS

Autoren Katrin Bächli,
 Manuel König

Hauptbetreuer Prof. Dr. Marcel Meli

Nebenbetreuer Herr Dario Dündar

Datum 9. Juni 2016

Abstract

The exchange of data between devices - commonly referred to as „Internet of things“ - should be made accessible to cyclists. The mobile device should start to work at a cycling speed of 10km/h, transmitting data through the Bluetooth Low Energy (BLE) protocol. The task is based on a feasibility study which handles the energy management through the chip EM8500. A „SensorTag“ - available on Texas Instruments Simple Link devices - is responsible for sending the data. This board contains a wireless MCU and a low power Cortex M3.

The goal of the task was the development of a miniaturised board, which should not be bigger than the utilized TI-SensorTag while adjusting the energy storage, energy management and the power consumption of the code. The finished product contains the hardware and a user friendly Android application. The „app“ allows adjusting the sensor readouts and displays a tachometer and a speedometer showing the current speed, which is based on the radius of the wheel).

At the start, a feasibility study was conducted. The feasibility study version showed promise for enough energy being produced at speeds over 45km/h.

After improving the energy harvest we were able to generate $20 \mu\text{W}$ at 10km/h. The energy management of the EM8500-Chip and the firmware of the TI-SensorTag were then completely rewritten. The threshold of the energy management is based on the MPP of the harvester and the goal is to send constant BLE-data at 10km/h. The firmware of the TI-SensorTag uses sleep functions to reduce the energy requirements.

The prototype is a configurable BLE-application, which sends speed, pressure, temperature and air moisture data in 1.5 minute intervals at 10km/h. From 20km/h onward, the data is sent every 20 seconds and over 45km/h the data is sent continuously.

Zusammenfassung

Internet of Things, die Unterstützung des Menschen im Alltag durch intelligente Geräte, soll für einen Fahrradfahrer auf innovative Art nutzbar gemacht werden. Das mobile Gerät soll durch Energy Harvesting gespiesen werden und bei einer Geschwindigkeit von 10 km/h Sensordaten mit Bluetooth Low Energy (BLE) senden. Die Arbeit baut auf einer Machbarkeitsstudie auf, die die gewonnene Energie mit dem Chip EM8500 verwaltet. Als Verarbeitungs- und Sendemodul wird das SensorTag von Texas Instruments der Serie Simple Link benutzt. Dieses Board beinhaltet einen Wireless MCU und den Low Power Cortex M3.

Aufgabe der Arbeit sind das Entwickeln einer miniaturisierten Leiterplatte, die nicht grösser als das verwendete TI-SensorTag ist. Das Speichern der Energie und das Einstellen von Schwellwerten an den Speicherelementen, damit der Energiezustand im System bekannt ist. Die Freigabe der gesammelten Energie zur Nutzung aufgrund vordefinierten Schwellwerten. Der Code wurde aufgrund zweier Aspekte power-optimiert: Die Funktionen laufen ohne RTOS und beinhalten das Setzen von Registern. Zudem schlafen grundsätzlich alle Power Domains. Sie werden nur für kurze Zeit für eine spezifische Aktion geweckt. Als Produkt steht neben der Hardware eine benutzerfreundliche Android Applikation zur Verfügung. Diese beinhaltet Einstellungen der Sensoren und einen ansprechenden Tachometer, der die Fahrgeschwindigkeit anzeigt.

Anfangs wird der Aufbau der Machbarkeitsstudie in Betrieb genommen. Die bestehende Version sendet Geschwindigkeit ab 45 km/h und basiert auf einem fliegenden Aufbau.

Nach der Verbesserung der Harvesterschaltung, sodass bei 10 km/h rund 20 μ W zur Verfügung stehen, wird der Print designt. Das Energy Management im EM8500-Chip und die Firmware des TI-SensorTags werden komplett neu geschrieben. Die Schwellwerte beim Energy Management basieren auf dem ausgewerteten Leistungsmaximum des Harvesters und dem Ziel, konstant BLE-Daten bei 10 km/h zu senden. Bei der Firmware des TI-SensorTags werden über Sleep-Funktionen dem System genügend Zeit zum wieder Aufladen gegeben.

Der Prototyp ist eine konfigurierbare BLE-Applikation, die bei 10 km/h jede 1.5 Minute Geschwindigkeits-, Druck-, Temperatur- und Luftfeuchtigkeitsdaten erhält. Bei 20 km/h werden die Daten nach 20 s und bei über 45 km/h konstant aktualisiert.

Vorwort

Die Idee, Firm- und Software für einen Fahrrad-Computer zu schreiben, stammt vom Research-Assistenten vom InES, Herrn Dario Dündar. Die Themen vom Entwickeln einer Android-Applikation über das Schreiben einer Firm- und Software für einen Cortex M3 bis hin zum Layouten eines Prints sprachen uns, Manuel König und Katrin Bächli, sofort an. Ein Grund ist, dass das vorgeschlagene Projekt unsere unterschiedlichen Schwerpunkte bestens vereint. Manuel König wollte sich ins Schreiben einer Android-Applikation vertiefen und Katrin Bächli hat grosses Interesse an hardwarenahen Programmierung.

Manuel König erarbeitete bereits in seiner Projektarbeit eine Android-Applikation und wollte das Wissen vertiefen. Auch interessierte ihn, von Grund auf eine einfache Leiterplatte zu designen. Katrin Bächli hatte generelles Interesse an Energy Harvesting und fand die Idee, ihre C-Kenntnisse durch dieses Projekt zu vertiefen verlockend. Wir wussten, dass wir auf eine funktionstüchtige Erstversion eines Fahrrad-Computers zurückgreifen konnten. Dieser wurde in einer Projektarbeit entwickelt und dient als Starthilfe.

Während der Umsetzung des Prototypen verlagerte sich der Entwicklungsschwerpunkt zunehmend von der Software weg Richtung Hardware. Die Harvesterschaltung wird mehrfach optimiert, damit der Fahrrad-Computer schlussendlich bereits ab 10 km/h Daten an die Android-Applikation senden kann.

Bei der Entwicklung des Fahrrad-Computers wurden wir sehr gut betreut. In gemeinsamen, wöchentlichen Sitzungen nahm sich Prof. Dr. Marcel Meli und Research-Assistent Dario Dündar Zeit. Sie setzten sich intensiv mit unseren Fragen auseinander und gaben sehr gute Vorschläge. Durch das Fachwissen konnten viele Fragen und Probleme gelöst werden. Wir lernten somit sehr viel durch dieses Bachelorarbeit.

Wir möchten Dario Dündar und Marcel Meli an dieser Stelle nochmals explizit danken. Prof. Dr. Marcel Meli kennt das Konzept des verwendeten EM8500 sehr gut und half uns, die richtigen Korrekturen vorzunehmen. Besonders in der Hardware-Entwicklung stellte er wichtige Fragen, die uns halfen, noch bessere Lösungen zu finden. Die Unterstützung durch Dario war unglaublich. Das TI-SensorTag konnte falsch reagieren wie es wollte, er gibt nie auf und verhalf so, zu immer neuen Ansätzen und Lösungs wegen. Ohne seine grosse Mithilfe würde der Fahrrad-Computer nicht ab 10 km/h funktio-

nieren. Eine weitere wichtige Unterstützung erhielten wir durch Erich Ruff vom InES. Wir kamen beim Messen der Energie bei einer gewissen Geschwindigkeit an die Genaugkeitsgrenze. Er baute für uns einen Rad-Imitator auf. So konnten wir genaue und reproduzierbare Messergebnisse erzielen.



Weiter danken wir Herrn Blum von Delectric GmbH, welcher uns die Leiterplatten für die Arbeit kostenlos und zeitnah zur Verfügung gestellt hat.

In dieser Projektarbeit konnte das ganze Wissen des Elektrotechnikstudiums angewendet werden. Speziell vertieft worden sind die Themengebiete Elektrotechnik für die Bewegungsinduktion, Elektronik für die Schaltungsoptimierung und das Leiterplatten-Design, die hardwarenahe Programmierung durch das Aufsetzen der Firmware und das Interrupt-Konzept.

Winterthur, 10. Juni 2016

Katrin Bächli
Manuel König

Inhaltsverzeichnis

1 Einleitung	13
1.1 Ausgangslage	13
1.2 Definition der Aufgabenstellung	14
1.3 Übersicht der Aufgabenblöcke	15
2 Theoretische Grundlagen	17
2.1 Energy Harvesting	17
2.1.1 Energy-Harvesting-Methoden	17
a. Energy Harvesting mit einer Solarzelle	18
b. Energy Harvesting mit einem TEG	18
2.1.2 Energy Harvesting über Bewegungsinduktion	18
2.1.3 Unterschiede Harvestermethoden	19
a. Gleichmässige Energie versus gepulster Energie	20
b. Konstanter MPP zu dynamischem MPP	21
2.2 Energy Management	22
2.2.1 Kontrollierte Energiespeicherung	23
2.2.2 Regelung des optimalen Leistungsbezugs	24
2.2.3 Spannung auf- und abwärtsregeln	25
2.2.4 Energiezustand kennen und Ein- und Ausgänge schalten	25
2.3 Power Management	27
2.3.1 Einbauen von Schlafmodi	28
a. Schlafen zwischen Ausführungen	28
b. Schlafen innerhalb einer Aktion	29
2.3.2 Interrupt Driven Application	29
a. Aufwachen durch interne Interrupts	29
b. Aufwachen durch externe Events	30
2.4 Bluetooth Low Energy	31
2.4.1 BLE im Vergleich zu Bluetooth	31
2.4.2 Advertising und Connected Mode	31
2.4.3 BLE Pakete	32
3 Vorgehen	35
3.1 Inbetriebnahme Machbarkeitsstudie	36
3.1.1 Funktionsblöcke	36
3.1.2 Verhalten des Vorgängermodells	37
3.1.3 Optimierungsliste	38
3.1.4 Vertiefung Harvestereingangs	39
3.2 Hardware entwickeln	41

3.2.1	Das Schema	41
a.	Harvesterschaltung	42
b.	EM8500-Chip	42
c.	Energiespeicher	42
d.	Umlauferfassung	43
3.2.2	Bauteildefinition und Optimierung	43
a.	Die Spule	44
b.	Der Gleichrichter	45
c.	Der Limiter	46
3.2.3	Layout	47
a.	Positionierung	47
b.	Das erste Layout	48
c.	Das Redesign	49
3.3	Inbetriebnahme des Prototypen	50
3.3.1	Testen der Harvesterschaltung	51
a.	Leistungsverifizierung	51
b.	Testen der Spule	52
c.	Testen Magnete in Serie	52
3.3.2	Ausmessen der Leistung endgültige Hardware	53
3.3.3	Ausmessen der Energieabgabe EM8500-Chip	54
3.4	Energy Management	54
3.4.1	Energiemessungen	55
a.	Überblick Energiemessungen	55
b.	Resultate Energieverbrauch TI-SensorTag Applikation V4	57
3.4.2	Energiekalkulation	64
3.4.3	Einstellen der Schwellwerte	66
3.4.4	Energiezustand des Systems	68
3.5	Power Management	69
3.5.1	Programmieren für Low Power Applikation im Advertising Mode	69
3.5.2	Power Domains beim TI-SensorTag	70
3.5.3	Schlafenszeiten implementieren	71
a.	Standby-Prozedur	72
b.	Abschalt-Prozedur	73
c.	Sleep Time-Übersicht bei Version V4	73
3.6	Applikationsentwicklung	74
3.6.1	Aufbau der Applikation	74
a.	Home-Screen	75
b.	Sensorwahl	75
c.	Einheiten und Einstellungen	75
d.	BLE-Kommunikation	75
e.	Inbetriebnahme des Codes der PA	75
f.	Filter einbauen	76
g.	Berechnung der Daten	76
3.6.2	Einstellungsmöglichkeiten	78
a.	Adressauswahl	78
b.	Einheiten	79

c. Kalibrierung	81
3.6.3 Animierter Tachometer	81
a. Funktionsweise	81
b. Mathematik der Tachonadel	82
3.6.4 Modularität der Applikation	84
4 Resultate	85
4.1 Harvesterschaltung	85
4.1.1 Der Print	85
4.1.2 Leistung am Harvesterausgang	86
4.1.3 Verhalten des Harvesterausgangs	87
4.1.4 Energie am EM8500-Chipausgang	87
4.1.5 Wirkungsgrad des Bicycle Computers	88
4.2 Energy Management	89
4.2.1 Verhalten bei tiefen und mittleren Geschwindigkeiten	89
4.2.2 Laden und Entladen des LTS	90
4.2.3 Verwendete Werte	92
4.3 Energiebilanz	93
4.3.1 Energiebilanz Bicycle Computer	93
a. Energiebilanz beim Beginn der Fahrradnutzung	93
4.3.2 Energiebilanz nach 1 min Fahrzeit	95
4.4 Ergebnisse BLE-Applikation	96
4.4.1 Applikationsstruktur	96
4.4.2 Paketverlust	98
4.4.3 Korrektheit der Daten	99
5 Diskussion	101
6 Verzeichnisse	103
6.1 Glossar und Abkürzungen	103
6.2 Abbildungsverzeichnis	104
6.3 Tabellenverzeichnis	106
Anhang	
A Ausschreibung Bachelorarbeit	I
B Blockdiagramm EM8500	III
C Funktionsblöcke des TI-SensorTags	V
D Digital Power Partitioning beim TI-SensorTag	VII
E Messaufbau	IX
F Messprotokolle per Datum und Messobjekt	XI

1

Einleitung

In einer vernetzten Welt senden Geräte Daten über ihren Zustand oder den ihrer Umgebung. Diese Technologie wird für eine Fahrradfahrerin bzw. einen Fahrradfahrer nutzbar gemacht. Das Handy soll die aktuelle Geschwindigkeit, die Höhe über Meer, die Temperatur und die Luftfeuchtigkeit während der Fahrt empfangen.

Diese Idee ist nicht neu. Erhältlich sind batteriebetriebene Modelle (siehe 1.1), die Daten auf einem Display anzeigen. Das Neue an dieser Arbeit ist, dass die Energie aus der Fahrradumdrehung geerntet (engl. to harvest) wird und dass die Benutzerin bzw. der Benutzer sein eigenes Handy für das Anzeigen der Daten nutzen kann.

1.1 Ausgangslage

Als Inspiration für den Prototypen dienten zwei batteriebetriebene Modelle der Hersteller Sigma Sport (?) und Polar (?). Sigma Sport bietet Geräte mit eigenem Display und Sensoren an. Auf dem Display erscheinen neben der Geschwindigkeit die Daten der Sensoren, die GPS-Ortung und der aktuelle Ladestand der Batterie. Der Hersteller Polar stellt ein Gerät her, welches die Fahrt über GPS aufzeichnet und wichtige Informationen zur Trainingsverbesserung liefert. Jedoch wird ein (verdrahtetes) Display gebraucht.

Ein weiterer, interessanter Hersteller ist Reelight (?). Reelight gewinnt über Wirbelströme Energie und schafft es, bei seinem Produkt City Supreme genügend Energie für eine LED-Lampe zu erzeugen. Da auf der Webseite keine Dokumentation des Funktionsprinzips erhältlich ist, ergaben eigene Untersuchungen, dass sich im Innern der Lampe "etwas" bewegt. Es steht zu Vermutung, dass dies ein Magnet ist, der so gelagert ist, dass er sich drehen kann. Der an der Felge vorbeiziehende Magnet erzeugt einen Wirbelstrom in der Felge. Der Wirbelstrom wirkt auf den Magneten im Fahrradlicht. Der Magnet im Fahrradlicht beginnt sich zu drehen. Befindet sich neben dem sich

drehenden Magneten eine Spule, so wird genügend Spannung für das Betreiben einer LED induziert. Diese Harvesting-Methode könnte interessant für eine zukünftige Arbeit sein. Der Nachteil dieser Methode ist, dass das Erzeugen eines Wirbelstroms sich auf den Felgen bremsend auswirkt. Bei dieser Bachelorarbeit ist die Harvesting-Methode bereits vorgegeben, da sie auf der nachfolgend genannten Projektarbeit basiert (?).

Als Grundlage dient der Aufbau aus der Machbarkeitsstudie “Bicycle computer and sensoric powered with harvested energy” (?). In dieser Projektarbeit wird der Beweis erbracht, dass durch Bewegungsinduktion genug Energie erzeugt werden kann, um die Geschwindigkeit des Fahrrads per Bluetooth Low Energy zu übermitteln. Der Aufbau funktioniert nach vorangehendem Laden der Kondensatoren zuverlässig ab 20 km/h. Das Ziel dieser Arbeit besteht aus einer verbesserten Energiegewinnung aus der Radumdrehung, einem besseren Verbrauchsmanagement bei der Datenverarbeitung durch einen Mikrokontroller und einer ansprechenden Applikation. Konkret soll ein attraktives Produkt ohne Aufladen der Kondensatoren für eine Geschwindigkeit von 10 km/h entstehen. Dieser Fahrrad-Computer-Prototyp wird in dieser Arbeit kurz Bicycle Computer genannt.

1.2 Definition der Aufgabenstellung

Durch die offizielle Ausschreibung der Bachelorarbeit an der ZHAW ist der Inhalt der Bachelorarbeit vorgegeben (siehe Anhang A). Das Ziel der Arbeit ist es, aus dem Aufbau einer Machbarkeitsstudie einen Prototypen eines batterielosen Fahrrad-Computers zu entwickeln. Zusammen mit den Auftraggebern Prof. Dr. Marcel Meli und Herr Dario Dündar wurde die Aufgabenstellung auf folgende Anforderungen konkretisiert:

1. Die Inbetriebnahme des Vorgängermodells, das Einlesen in die vorangegangene Projektarbeit und Beschäftigung mit der Materie sind die Hauptpunkte des ersten Schrittes.
2. Die bestehende Hardware muss verkleinert und überarbeitet werden. Dafür wird ein neues Printed Circuit Board (PCB) entworfen, welches verschiedene vorhandene Platinen vereint.
3. Die Inbetriebnahme der Bluetooth-Schnittstelle muss auf dem Android-Endgerät und der Hardware vorgenommen werden. Eine erste Bluetooth-Kommunikation zwischen der Hardware und der Applikation ist implementiert.
4. Das bestehende Energiemanagement soll auf die Anwendung eines Fahrradcomputers optimiert werden.
5. Die Benutzeroberfläche der Android-Applikation soll benutzerfreundlich und optisch ansprechend gestaltet werden.
6. Die erfassten Messwerte der Geschwindigkeit und der aktuellen Höhe sollen über Bluetooth übermittelt werden.

7. Die erfassten Daten sollen gespeichert und nur dann übertragen werden, wenn die nötige Energie vorhanden ist.
8. Per GPS soll die aktuelle Position ermittelt sowie die bereits abgefahrenen Route erfasst werden. Alles soll auf einer Karte veranschaulicht werden.
9. Die Beschleunigung, Luftfeuchtigkeit und Temperatur sollen ebenfalls erfasst und über Bluetooth übermittelt werden.
10. Das Energiemanagement soll für verschiedene Geschwindigkeiten optimiert werden.

Bei der Festlegung der Arbeitsschritte half die Vision eines innovativen Fahrradcomputers. Die Miniaturisierung, Punkt 2 in der Aufzählung oben, ist für die Attraktivität des Produkts entscheidend. Ebenso wird Vorteil dahingehend gesehen, dass das eigene Handy als Display für die Daten zu verwenden. Aus diesem Grund wurde Punkt 5 als eine der Hauptaufgaben definiert. Die App soll ansprechend und einfach für die Benutzerin oder den Benutzer sein. Das verbesserte Energy Management, Punkt 4, hat mit dem Ziel zu tun, dass der Fahrradcomputer bereits bei 10 km/h die Geschwindigkeit ausgeben soll. So gelten für die Bachelorarbeit die Punkte 1 bis 6 als Minimalanforderungen, während sich die Punkte 7 bis 10 dynamisch und in Abhängigkeit des Projektfortschritts einbauen lassen. Aus den definierten Anforderungen entstand der auf der CD abgelegte Projektplan.

1.3 Übersicht der Aufgabenblöcke

Um den Überblick der zu erledigenden Punkte zu behalten, werden die Aufgaben in Arbeitsblöcke (siehe Abbildung 1.3) aufgeteilt. Die Abbildung unterscheidet graphisch die oben genannten Minimalanforderungen von den optionalen Anforderungen. Die Arbeitsblöcke mit grauem Hintergrund sind die Minimalanforderungen, die auf weissem entsprechen den optionalen Anforderungen. Die Projektplanung wurde so aufgebaut, dass bei Meilenstein 1 das Layout gezeichnet ist, bei Meilenstein 2 die Kommunikation zur App besteht und bei Meilenstein 3 die überarbeitete Version des Prototyps gezeigt wird und bis dahin die Minimalanforderungen erreicht sind. Die Arbeitsblöcke tragen eine Meilenstein-Farbe, bis wann sie erledigt sein sollen. Auf der CD im Anhang ist die Projektplanung abgelegt.

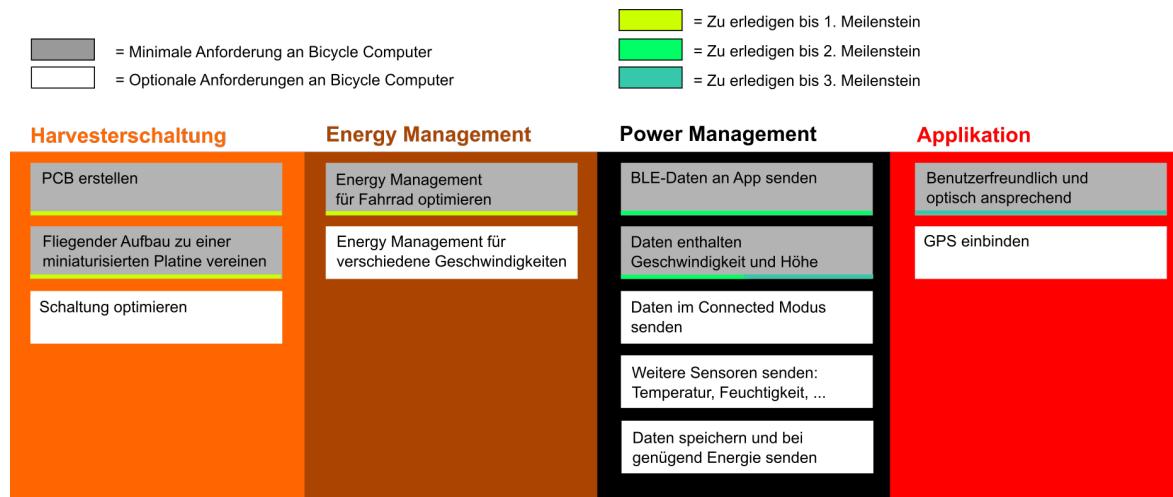


Abbildung 1.1: Arbeitsblöcke

2

Theoretische Grundlagen

Der Bicycle Computer basiert auf Energy Harvesting, was Energie ernten bedeutet. Welche Art von Energy Harvesting in dieser Arbeit angewandt wird, wird im ersten Unterkapitel 2.1 beschrieben. Im folgenden Unterkapitel 2.2 geht es um Ansätze zum Sammeln (engl. to accumulate) und Weiterleiten (engl. to manage) von Energie. Da die im Unterkapitel 2.1 beschriebene Energie im μW -Bereich liegt, ist zuerst ein Sammeln der Energie notwendig, sodass Leistungen im mW-Bereich zur Verfügung stehen. Das nächste Unterkapitel 3.1.1 befasst sich mit dem notwendigen Power Management. Denn die Energie soll nicht sofort verbraucht werden. Power Management regelt, wie schnell und wie viel Energie aufs Mal verbraucht werden soll. Als letzte Stufe in der Umsetzung ist eine energiearme Kommunikation notwendig. Bluetooth-Low-Energie-Technologie ist der Ansatz, der in dieser Arbeit verwendet wird. Das Protokoll und die Technologie werden im letzten Grundlagenteil 2.4 vorgestellt.

2.1 Energy Harvesting

“Mit Energy Harvesting [...] wird die Gewinnung von elektrischer Energie in kleinen Mengen aus dem Umfeld [...] bezeichnet.” ?. Als erstes werden Methoden zur Energiegewinnung vorgestellt (2.1.1) und danach die im Bicycle Computer verwendete Harvesting-Art genauer beschrieben (2.1.2). Als letztes wird der Unterschied zwischen den unterschiedlichen Harvestingmethoden festgehalten. Denn diese Unterschiede werden in der Implementation des Bicycle Computers wichtig.

2.1.1 Energy-Harvesting-Methoden

Bekannte Methoden des Energy Harvesting sind die Solarzelle, die aus der Energie der Sonnenstrahlen Strom erzeugt, die Thermogeneratoren (TEG), die aus Umge-

bungswärme Energie gewinnen, passive RFID-Tags, die aus der elektromagnetischen Strahlung Energie gewinnen und der piezoelektrische Effekt, der mechanischen Druck in elektrische Spannung umwandelt. Da der im Prototyp verwendete Energy-Mangement-Chip ? auf die Energieoptimierung von Solarzellen oder von Thermogeneratoren spezialisiert ist, werden diese zwei Methoden vorgestellt.

a. Energy Harvesting mit einer Solarzelle

Bei der Umwandlung von elektromagnetischen Wellen (Licht) in Strom wird eine spezielle Eigenschaft des Halbleiters genutzt: Führt man einem Halbleiter Energie in Form von Sonnenlicht zu, entstehen freie Ladungsträger, bzw. Elektronen und Löcher. Um aus diesen Ladungen einen elektrischen Strom zu erzeugen, ist es nötig, die erzeugten freien Ladungsträger in unterschiedliche Richtungen zu lenken; dies geschieht durch ein internes elektrisches Feld, welches durch einen p-n-Übergang erzeugt werden kann. Auf der einen Seite sammelt sich positive, auf der anderen Seite negative Ladung an. Werden diese verbunden, entsteht ein Strom (?). Diese Harvestermethode produziert einen Gleichstrom. Größen- und materialabhängig kann Energie im kW-Bereich gesammelt werden.

urstrom
cht schule:
tfehler.
n Jahr

b. Energy Harvesting mit einem TEG

TEG steht für Thermoelectric Generator und bezeichnet eine Konstruktion, die aus einem Temperaturunterschied elektrische Spannung erzeugt. Erzeugt wird die Spannung am Ende zweier metallischer Leiter aus unterschiedlichem Material, die an einem Ende verbunden sind (?). Diese Harvestermethode produziert eine Gleichspannung. Die produzierte Spannung ist vergleichsweise klein und bewegt sich im Bereich einiger $10 \mu\text{V}$ pro 1°C Temperaturdifferenz.

2.1.2 Energy Harvesting über Bewegungsinduktion

Beim Bicycle Computer wird Energie über Bewegungsinduktion gewonnen. Die Funktionsweise ist in der Machbarkeitsstudie beschrieben ? S. 8.:

Befindet sich eine Spule in einem *dynamischen* „Magnetfeld“, wird in der Spule eine Spannung induziert. Dies sieht man in der Formel (2.1).

$$U_{ind} = -\frac{d}{dt} \int A dB \quad (2.1)$$

Der magnetische Fluss B durch die Fläche einer Spule A ist gleich dem magnetischen Fluss ϕ . Hat die Spule mehrere Wicklungen N , so steigt die durchflossene Fläche und mehr Spannung wird induziert.

$$\frac{d}{dt} \int A dB = \phi \cdot N \quad (2.2)$$

Verläuft der magnetische Fluss ϕ senkrecht zur Fläche der Spule A kann das Integral durch eine Multiplikation ersetzt werden (siehe Formel 2.3).

$$\frac{d}{dt} \int A \perp dB = \frac{d}{dt} \int \phi \cdot N = B \cdot A \cdot N \quad (2.3)$$

In diesem Fall berechnet sich die induzierte Spannung in einer Spule vereinfacht mit

$$U_{ind} = -N \cdot A \cdot B \quad (2.4)$$

Bild einfügen

Das dynamische Magnetfeld wird durch das Bewegen, oder im Fall eines Fahrrads einem Vorbeiziehen eines Magneten an einer fix verankerten Spule erzeugt. Die produzierte Spannung hängt von folgenden vier Faktoren ab:

1. die eingeschlossene Fläche A der Spule
2. die magnetische Flussdichte des Magneten B
3. die Anzahl Windungen N der Spule und
4. die Zeit Δt , in der das Magnetfeld an der Spule anliegt

Diese Harvestermethode produziert einen Wechselstrom, deshalb sind ein Gleichrichter und ein Kondensator zur Glättung der Rippelspannung nach der Energiegewinnung notwendig. Die Leistung der produzierten Spannung geht vom μW -Bereich bis zu für die Industrie optimierten Anlagen mit Leistungen im MW-Bereich wie z. B. durch Drehstrom-Generatoren.

2.1.3 Unterschiede Harvestermethoden

Satz unverståndlich
neu schreibe
Siehe Notize
Alexey

gliedern in

Unterschiede zwischen den Harvesting-Methoden bestehen in der Art, in der die Energie zur Verfügung steht. Die nachfolgenden zwei Unterpunkte zeigen zwei Differenzen auf:

Punkt a verweist auf die unterschiedliche Art, in der die Energie erhältlich ist:

Entspricht dies einer gleichmässigen Energie, analog zum DC, oder liefert die Quelle Energie in Form von Wechselstrom? Folgt der Strom konstant oder in Pulsen?

Die aufwändige Art der Energieerntung wird für den Prototypen relevant. Der zweite Unterschied unter den Harvestermethoden ist die Leistungskurve. An welcher Stelle zwischen Kurzschluss und Leerlauf liegt das Leistungsmaximum? Die unterschiedlichen Leistungsmaxima der einzelnen Harvestermethoden werden unter Punkt b zusammengefasst. Beide Unterschiede werden für die Entwicklung des Prototypen über Bewegungsinduktion relevant.

a. Gleichmässige Energie versus gepulster Energie

Die Solarzelle und ein TEG liefern Gleichstrom bzw. -spannung (siehe Abbildung 2.1), weshalb keine Gleichrichterschaltung und Glättung notwendig sind. Die durch Bewegungsinduktion gewonnene Energie ist eine Wechselspannung (siehe Abbildung 2.2, aus Messprotokoll ?). Im Fall des Bicycle Computers ist diese gleichzeitig gepulst. Die Energie ist somit nicht konstant da, sondern nur in Zeitintervallen. Dadurch ergeben sich zwei Probleme bei der Energiegewinnung:

Durch den Gleichrichter geht Energie verloren und durch die Pulsform entsteht, trotz Signalglättung über einen Kondensator, eine Ripplespannung (siehe Abbildung 2.3, aus Messprotokoll ?).

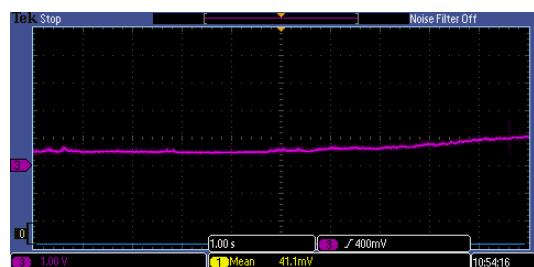


Abbildung 2.1: Theoretische Abbildung einer Gleichspannung am Ausgang eines TEG

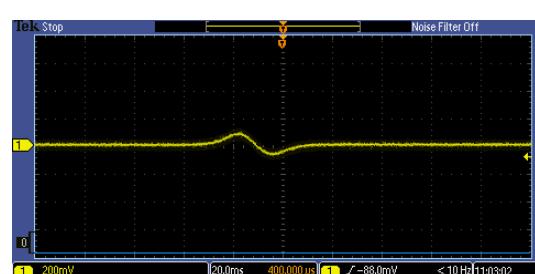


Abbildung 2.2: Wechselspannung bei Bewegungsinduktion

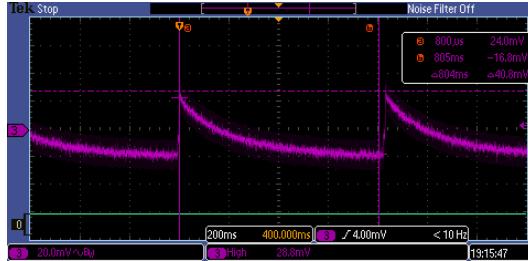


Abbildung 2.3: Rippelspannung aufgrund der gepulsten Eingangsenergie

b. Konstanter MPP zu dynamischem MPP

Die drei Harvester unterscheiden sich in ihrer Leistungskurve.

Zur Verdeutlichung der Unterschiede sind für jedes Leistungsverhalten eine Abbildung angefügt. Das TEG hat unabhängig von der gewonnenen Energie und der Temperatur das Leistungsmaximum immer bei 50 %. Die Abbildung 2.4 zeigt dieses unabhängige Verhalten.

Das Leistungsmaximum, der Maximum Power Point (MPP), liegt auf der Skala von Kurzschluss bis Leerlauf proportional an unterschiedlichen Stellen. Bei einem TEG liegt das MPP in der Mitte dieser Skala (siehe Abbildung 2.4). Die Maximum Power Point Tracking Ratio, kurz MPPT-Ratio, beträgt 50 %. Bei der Solarzelle liegt das Leistungsmaximum auf der Skala bei ca. 80 % der maximalen Spannung. Die MPP-Ratio ist 80 % (siehe Abbildung 2.5). Bei der Bewegungsinduktion existiert keine fixe MPP-Ratio. Wie bei der Spule, wandert das Leistungsmaximum aufgrund mehrerer Indikatoren (wie Geschwindigkeit des Magneten durch die Spule, Abstand von Magnet und Spule) auf der Skala.

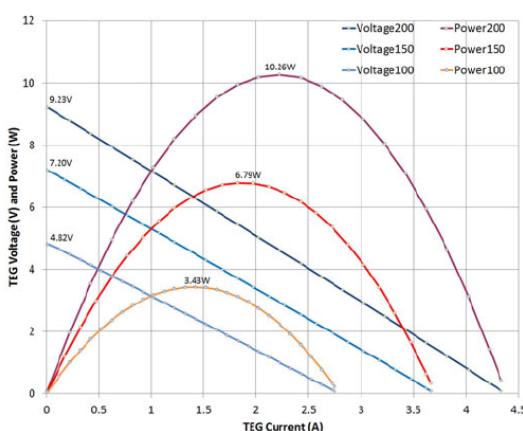


Abbildung 2.4: MPP TEG (?)

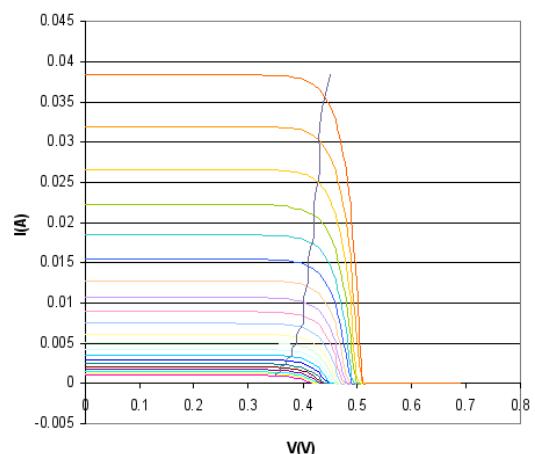


Abbildung 2.5: MPP Solarzelle (?)

Erklärung d
Leistungsku

Erklären, w
Leistungsm
mum ist

Übergang t
neu

Abbildung 2.5 zeigt, dass das Leistungsmaximum bei der Solarzelle unabhängig von der zur Verfügung stehenden Energie immer bei 80 % liegt.

Die Stelle des Leistungsmaximums wandert bei einer Spule und somit bei der Bewegungsinduktion auf der Skala. Exemplarisch sind drei MPPT-Ratios einer Spule in der Abbildung 2.6 abgebildet. In dieser Abbildung zeigt sich der Einfluss des Abstands der Spule vom Magnetfeld auf die Stelle der maximalen Leistung. Diese Abbildung wurde ausgewählt, weil beim Ausmessen des Harvesters der Abstand des Magneten als einer der Einflüsse festgestellt wurde. Im Kapitel 4 Resultat sind die ausgemessenen Leistungskurven des Prototypen in der Graphik 4.3 abgebildet. Die Leistungskurven wurden für verschiedene Geschwindigkeiten aufgenommen und es zeigt sich, dass bei der Harvesterschaltung analog zur Spule, die MPPT-Ratio wandert. Für den theoretischen Teil wurde die MPPT-Kurven des Harvester stark geplättet und vereinfacht (siehe untenstehende Abbildung 2.7). So ist der Effekt der Verschiebung des Leistungsmaximums besser ersichtlich. Es lässt sich grob über die MPPT-Ratio des Bicycle Computers sagen, dass sie sich zwischen 35 - 75 % bewegt.

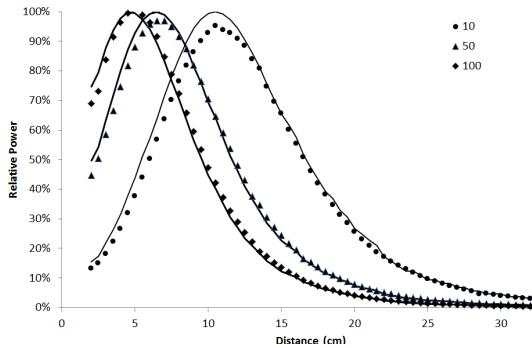


Abbildung 2.6: MPP Spule (?)

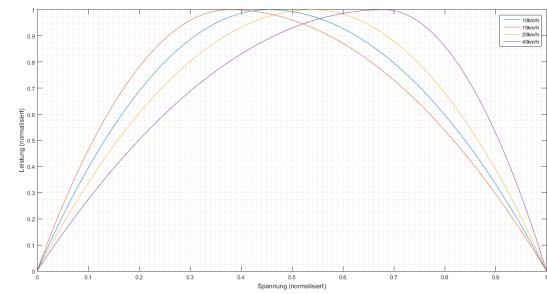


Abbildung 2.7: MPP Harvester (?)

2.2 Energy Management

Der Harvester des Bicycle Computer erntet eine gepulste Energie im μJ -Bereich. Um diese für eine Applikation zu verwenden, müssen die geringen Energieportionen summiert werden. Sind Energiemengen im höheren μJ -Bereich verfügbar, kann die Energie kontrolliert freigegeben werden.

Energy Management bezeichnet das Sammeln von Energie in Speichern, das Regeln des Eingangssignals auf das Leistungsmaximum und das Aufwärtswandeln von Spannung oder Strom auf einen geforderten Wert sowie die kontrollierte Freigabe der gesammelten Energie.

In der Bachelorarbeit ist das Verwenden des Chip EM8500 vorgegeben. Der EM8500 ist ein Power Management Controller für den Low Power-Bereich. Das Datenblatt

des EM8500 ist der CD beigelegt. Als erstes wird das kontrollierte Energiespeichern anhand dieses Chips erklärt. Danach folgt die Umsetzung des Maximum Power Point Trackings (MPPT) und eine kurze Erklärung der Wirkung des Boosters auf das Energy Managment. Zuletzt wird auf das Freischalten von Ausgängen eingegangen, da dies für das Verwenden der Energie die wesentliche Schnittstelle ist.

2.2.1 Kontrollierte Energiespeicherung

Bei einer Low-Power-Harvesting-Applikation ist wesentlich, dass vor der Verwendung der Energie durch einen Mikrokontroller genug Energie in Speichern gesammelt wurde. Ansonsten kann der Mikrokontroller nicht sauber aufgestartet, da für den Startvorgang eines Mikrocontrollers mehr Energie benötigt wird als für die Ausführung eines Programms. Dies ist in der Abbildung 2.8 dargestellt. Sie zeigt, dass die Freigabe der Energie an eine Applikation erst nach dem Erreichen eines gewissen Speicherzustands erfolgt. Die Höhe des Speicherwertes kann im EM8500 programmiert werden. Die Ladespannung des sogenannten Primärspeichers, dem Short Time Storage (STS), ist mit VSTS in der Abbildung 2.8 abgebildet. Das Signal der Speisung der Applikation wird als VSUP bezeichnet.

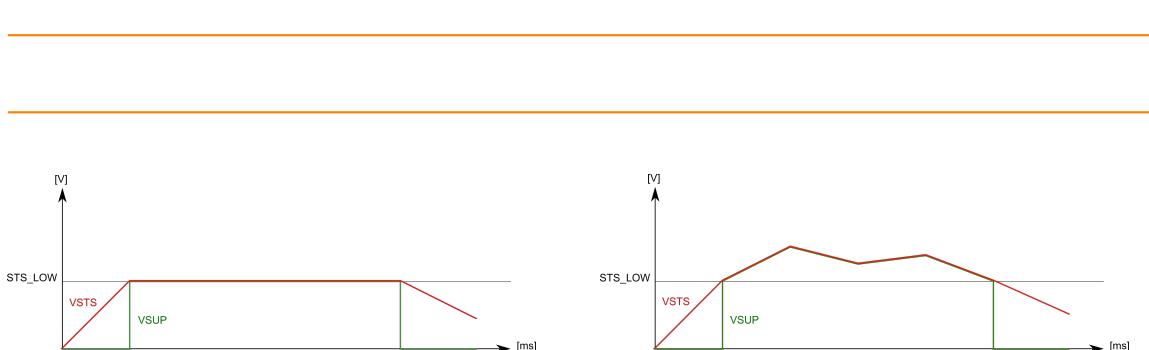


Abbildung 2.8: Grundprinzip Applikationsspeisung

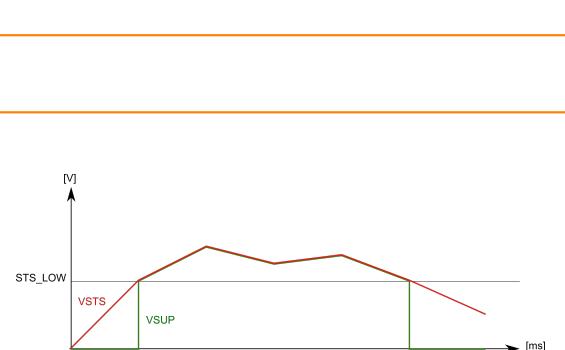


Abbildung 2.9: Applikationsspeisung EM8500

ev. Blockdiagramm mit allen Bezeichnungen. oder verweis auf Graphik

Graphik falsch. VSUP auf b. minlow. vsts geht auf v b. min hi dis

sofort erklären warum. Denn...

Im EM8500 wird dies folgendermassen umgesetzt (Abbildung 2.9): Erreicht der Primärspeicher STS den Schwellwert $v_{bat_min_low}$, wird VSUP mit der eingestellten Spannung gespiesen. Der Registerwert $v_{bat_min_low}$ wird zukünftig STS_LOW bezeichnet. Denn es ist der tiefere Schwellwert des STS, der die Spannungshöhe der Applikation definiert. Die Applikation sollte nicht die gesamte Energie verbrauchen, sodass sich der Speicher weiter lädt. VSUP folgt der Speicherspannung. Verbraucht die Applikation viel Energie, fallen VSUP und VSTS parallel. Speiste der Harvester viel Energie, steigt bei beiden die Spannung an. Unterschreitet VSTS/VSUP den Schwellwert von STS_LOW, so wird die Speisung der Applikation gestoppt.

Der Primärspeicher STS ist für die kurzfristige Speisung der Applikation verantwort-

lich. So bedeutet STS Short Time Storage. Für das langfristige, sichere Ausführen braucht das System ein Long Term Storage (LTS). Seine Aufgabe ist, Reserveenergie aufzubauen. Diese überbrückt die Energieengpässe, wenn der Harvester zu wenig Energie liefert. VLTS wird geladen, wenn der Schwellwert $v_appl_max_lo$ überschritten wird (siehe Abbildung 2.10).

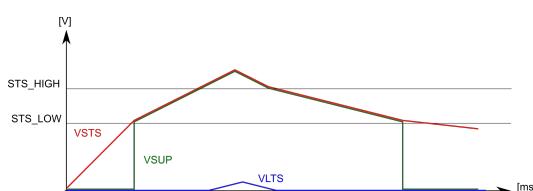


Abbildung 2.10: Sicheres Betreiben durch Long Term Storage

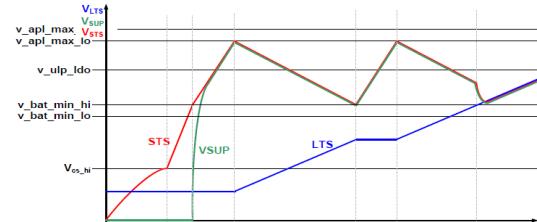


Abbildung 2.11: Konzept Hersteller (?)

Im Datenblatt des EM8500 (siehe ?) sind weitere Feineinstellungen beschrieben und drei Application Notes helfen bei der Berechnung der Schwellwerte für ein sicheres Betreiben. Die Dateien sind auf der CD abgelegt.

Grundsätzlich ist zur Berechnung der Speicher und den Schwellwerten zu sagen, dass der erste Schwellwert (STS_LOW), bei dem die Speisung der Applikation beginnt, genug Energie für die Initialisierung der Applikation gesammelt haben muss. Zudem muss das Abschalten von VSUP vermieden werden. Denn ein Neustart braucht aufgrund der Initialisierung viel Energie und ist ein unnötiger Kraftakt in einem Low Power System. In den Beispielkonfigurationen des Herstellers (? S. 5 - 8) sieht man, dass in deren Überlegungen VSUP nicht abgeschaltet wird. Der Hersteller geht davon aus, dass sogar bei dem Freischalten von VSUP die Spannung am STS nicht aufgrund der Last der Applikation fällt, sondern sich weiterhin auflädt.

2.2.2 Regelung des optimalen Leistungsbezugs

Wichtigster Punkt in der Energieoptimierung ist, das Maximum aus der produzierten Energie weiterzuverwenden. Aus diesem Grund wird vor Inbetriebnahme eine Leistungskurve des Harvesters erstellt. Wie in Unterkapitel 2.1.3 beschrieben, unterscheidet sich der Maximum Power Point (MPP) unter den Harvestern stark.

EM8500 versucht die Quelle stets in der Nähe dieses Optimums zu betreiben. Dies geschieht über eine Innenwiderstand-Regelung, sodass die Eingangsleistung möglichst dem MPP entspricht. Wie häufig die aktuelle Leistung überprüft wird, ist einstellbar. Der EM8500 hat eine Auflösung von 37 mV bei der Spannungsmessung bei der Eingangsspannungsmessung. Die Abbildung 2.12 zeigt das periodische Messen des Spannungswertes des Harvesters. Da die Kurzschlussmessung für das Messen des Stromwerts

eine Spannungsspitze verursacht, (gut zu sehen in der Abbildung bei höherer Spannung), sollte die Leistungsüberprüfung nicht zu oft geschehen. Spannungsspitzen bedeuten Energieverlust, weshalb die Energiemessung nur so oft wie absolut notwendig erfolgen soll. In der Abbildung 2.12 beträgt die Periode 8 Sekunden.

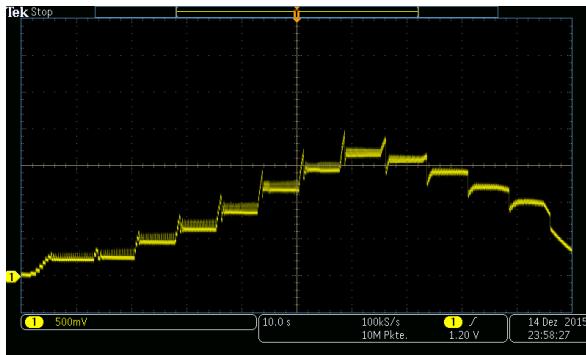


Abbildung 2.12: Leistungsmessung des Harvesters

2.2.3 Spannung auf- und abwärtsregeln

Beim EM8500 bilden der MPPT-Kontroller und der Booster eine Einheit (siehe Blockdiagramm im Anhang B). Die Aufgabe des Boosters ist es, das interne Spannungsniveau (VREG) zu heben. Der Booster arbeitet ab einer Eingangsspannung von 0.3 V. Danach regelt er in Schritten von 0.3 V. Der Booster geht von einer Gleichspannung eines TEG oder einer Solarzelle aus. Für diese zwei Anwendungen ist der EM8500 konzipiert (siehe ?, Abschnitt Description, S. 1).

Titelname
nicht gut

2.2.4 Energiezustand kennen und Ein- und Ausgänge schalten

$E_{Applikation}$ bezeichnet die minimale Energie, die die Applikation braucht, also mindestens die Initialisierung der Applikation. Die Applikation wird über das Signal V_SUP, das die Energie vom Primärspeicher STS erhält, gespiesen.

$$E_{Applikation} = C_{STS} \times \frac{1}{2} (V_{STS_LOW})^2 \quad (2.5)$$

Da sich der Kondensator C_{STS} bei der Freigabe der Energie über V_SUP nicht auf 0 entlädt, sondern die minimale Applikationsspannung erhalten bleibt, gilt für die Berechnung des Kondensators C_{STS} nur die Energiedifferenz als verwendbar. Es ist die

Differenz zwischen den zwei eingestellten Primärspeicher-Schwellwerten STS_LOW (ab diesem Schwellwert wird V_SUP gespiesen) und v_bat_min_low (ab diesem Schwellwert hört die Speisung von V_SUP auf). Der zweite Schwellwert wird in der Gleichung vereinfacht als STS_HIGH bezeichnet.

on
her die
schwellwerte
ein-
stellt als STS
GH,STS_LOW
berechnen

$$\begin{aligned} E_{Applikation} &= E_{STS_HIGH} - E_{STS_LOW} \\ &= C_{STS} \times \frac{1}{2} (V_{STS_HIGH})^2 - C_{STS} \times \frac{1}{2} (V_{STS_LOW})^2 \end{aligned} \quad (2.6)$$

Kennt man die notwendige Energie, so kann die Grösse des Kondensator C_{STS} aus der notwendigen Energie berechnet werden:

$$C_{STS} = \frac{E_{Applikation}}{(V_{STS_HIGH})^2 - (V_{STS_LOW})^2} \quad (2.7)$$

Umgekehrt argumentiert, kann man aus der Höhe der Applikationsspannung (V_SUP bzw. V_{STS_LOW}), des bekannten Energiekonsums $E_{Applikation}$ den notwendigen Schwellwert für den Primärspeicher berechnen:

$$(V_{STS_HIGH})^2 - (V_{STS_LOW})^2 = \frac{2 \times E_{Applikation}}{C_{STS}} \quad (2.8)$$

$$(v_{bat_min_low})^2 = \frac{2 \times E_{Applikation}}{C_{STS}} + (V_{STS_LOW})^2 \quad (2.9)$$

Neben VSUP kann per I2C oder SPI der aktuelle Spannungspegel der Regelung (VREG), die Speicher (VDD_STS und VDD_LTS) und des Harvestereingangs (VDD_HRV) abgefragt werden. So kennt die Applikation jederzeit den aktuellen Energiezustand der gesammelten Energie.

EM8500 stellt zwei digitale Überwachungssignale zur Verfügung:
Der Ausgang HRV_LOW ist auf logisch '0', wenn die Eingangsspannung vom Harvester grösser als 0.3 V ist. Fällt diese darunter, geht HRV_LOW auf logisch '1'. Der Ausgang BAT_LOW zeigt die Zeittdauer an, in der nur STS die Applikation speist:

- BAT_LOW = '0'
Nicht genügend Energie zur Speisung der Applikation. VSUP ist ausgeschaltet.

- BAT_LOW = '1'
Genügend Energie zur Speisung der Applikation.
VSUP ist eingeschaltet.
- HRV_LOW = '0'
Genügend Energie zur Speisung von LTS.
VSUP ist eingeschaltet.
Der Zustand entspricht nicht mehr BAT_LOW.

Mit den zwei digitalen Signalen kann der Energiezustand grob abgebildet werden.

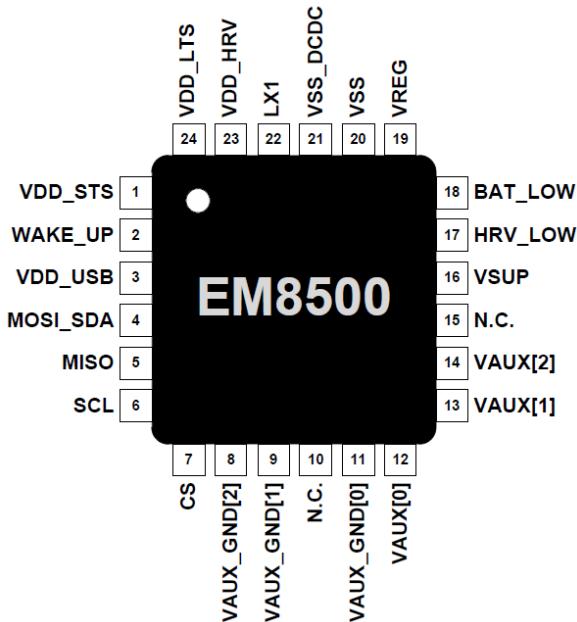


Abbildung 2.13: In- und Outputs EM8500 (? , p.11)

2.3 Power Management

Die Aufgabe des EM8500-Chip ist es, Energie zu sammeln und kontrolliert frei zu geben. Die Aufgabe des nachfolgenden Mikrocontrollers ist es, die freigegebenen Energieportionen optimal zu verwenden. Das bedeutet, möglichst wenig Energie bei der Datenverarbeitung zu benötigen. Dies wird durch Abstellen aller unnötigen Mikrocontroller-Bereiche erreicht und einem zusätzlichen Schlafen während allen Warteprozessen.

In diesem Kapitel werden zwei Ansätze zum Umsetzen eines Low Power Systems vorgestellt. Das Hauptthema ist das Schlafen zwischen allen Prozessen. Dies wird im ersten Unterkapitel beschrieben. Das Schlafen bedingt ein Aufwecken aufgrund von Ereignissen.

nissen. Dadurch ergibt sich eine Interrupt Driven Applikation. Diese wird im zweiten Unterkapitel erklärt.

Vor der technischen Beschreibung der Konzepte in den drei Unterkapiteln wird kurz auf die verwendete Hardware eingegangen. In der Bachelorarbeit war als Mikrokontroller das Simple Link TI-SensorTag von Texas Instrument vorgegeben. Der Grund für dieses Board ist, dass das TI-SensorTag drei Anforderungen auf einem Board vereint:

- Ein Cortex M3 dient als Haupt-Mikrokontroller und ist aufgrund seiner hohen Rechenleistung und seiner Low Power-Fähigkeiten für eine Harvester-Anwendung wie der Bicycle Computer geeignet.
- Auf dem Board ist zusätzlich ein Cortex M0 für die Wireless-Anbindungangeschlossen. Dieser Prozessor kann von der Benutzerin oder dem Benutzer nicht programmiert werden, da die Schnittstelle zur Low Power Datenkommunikation fix aufgesetzt ist. Neben Bluetooth Low Energy kann auch Zigbee verwendet werden.
- Auf dem Board sind 10 Sensoren eingebunden.

Die Funktionsblöcke des TI-SensorTags befinden sich im Anhang C.

2.3.1 Einbauen von Schlafmodi

Low Power Microcontroller können Gebiete des Prozessors oder von Peripherieelementen temporär ausschalten. Das System befindet sich im Standby Modus. Nur die für die Applikation unabdingbaren Aktivitäten laufen mit niedrigstem Takt weiter. Über Interrupts können einzelne Bereiche aufgeweckt werden, die ihre Aktionen ausführen und danach geht das System wieder in den Standby Modus.

Das Verhalten, dass Prozessoren nach längerer Zeit ohne externen Input in den Schlafmodus gehen, ist typisch für Low-Power-Prozessoren. Bei einer optimierten Ultra-Low-Power-Applikation geht der Prozessor nach kleinsten Ausführungsblöcken schlafen. So gehört zum Aufwecken einer Peripherie, das Schließen aller anderen Power Domains. Und nach dem Aufwecken der Peripherie, geht diese selbst auch wieder in den Schlafmodus. In diesem Unterkapitel werden zwei Umsetzungen des Sleep-Modus konzeptionell erklärt.

a. Schließen zwischen Ausführungen

Die Abbildung 2.14 zeigt das Grundprinzip. Das Programm besteht aus verschiedenen Aktionsblöcken. Diese dauern unterschiedlich lang und verbrauchen unterschiedlich viel Energie. Zwischen den Aktionen ist eine frei wählbare Wartezeit einbaubar ($\Delta t_0 - t_3$). Die graue Markierung in jedem Aktionsblock sind die Initialisierungen vor jeder Aktion.

Während der Schlafenszeit sind alle Peripherien abgeschaltet und im Prozessor (hier als Bsp. Cortex M3) werden nur die Konfigurationen im Flash regelmässig “refreshed” (deutsch: neu laden). Texas Instruments entschied sich für diese Methode. Die Alternative ist, die MCU mit einem minimalen Grundpegel konstant zu speisen. Dies ist jedoch beim TI-SensorTag nicht der Fall. Texas Instruments nennt das Refreshen “retention”, siehe Anhang D.1. Die Refreshing-Peaks des TI-SensorTags sieht man im Standby-Modus. Dies ist in der Abbildung 2.14 an den grünen Stromspitzen zu sehen. Ohne Refreshen des Speichers gehen die Konfigurationen verloren und vor jeder Aktion muss das System komplett initialisiert werden.

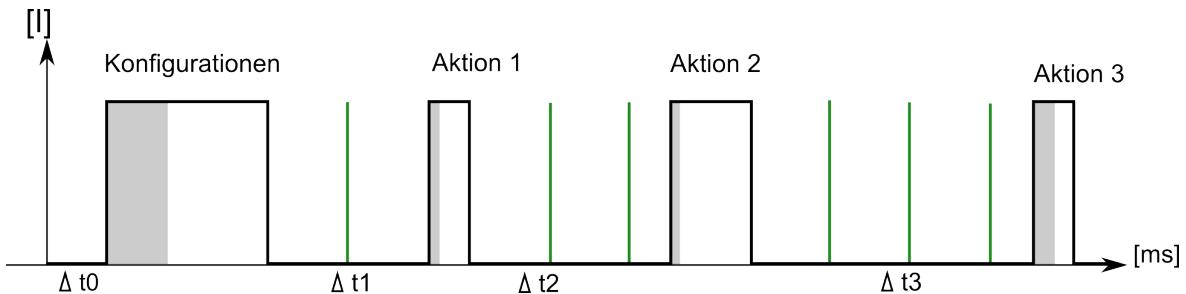


Abbildung 2.14: Schläfen zwischen Ausführungen

b. Schlafen innerhalb einer Aktion

Bei einer Applikation im μ - oder mW-Bereich geht das Gerät bei jedem Warteprozess, wie z.B. die Zeit, die der Sensor zum Aufwachen braucht, in den Sleep-Mode. Innerhalb des Codes dominieren die Aufwach- und Abstelleinstellungen. Für jede Aktion, wird nur die PowerDomain dieser Funktionalität eingeschaltet und nach Ausführen der Aktion wieder abgeschaltet. Die Abbildung 2.15 zeigt dieses Prinzip als Flussdiagramm dargestellt.

2.3.2 Interrupt Driven Application

Zum Schläfen gehört auch ein Aufwachen. Dies ist ein nicht-triviales Problem, welches sich durch Interrupts lösen lässt. In diesem Unterkapitel werden zwei Konzepte einer Interrupt Driven Application erklärt: fixes Aufwachen aufgrund interner Interrupts und asynchrones Aufwachen aufgrund externer Events.

a. Aufwachen durch interne Interrupts

Ein System kann intern seine Signale auswerten und aufgrund kombinatorischer Logik, dem Erreichen eines Schwellwertes oder dem Ablauen eines Timers, aufwachen.

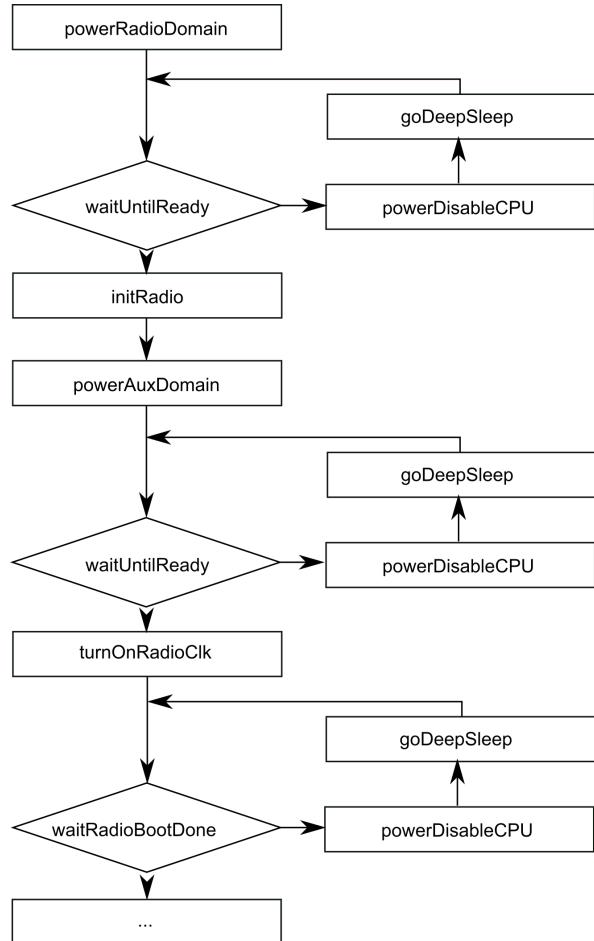


Abbildung 2.15: Schlafen innerhalb des Codes

Solche Wakeups sind fix und unabhängig von äusseren Einflüssen. Ein System mit internen Interrupts ist determinierbar. Das heisst, der Empfang interner Interrupts ist vorhersehbar und kann über Prioritäten geregelt werden.

b. Aufwachen durch externe Events

Der Prozessor, oder Teile davon, können auch aufgrund äusserer Impulse aufwachen. Die Verarbeitung des Interrupts ist dieselbe, nur weiss man nicht, wann das Ereignis auftritt. Die Gefahr, dass zwei Interrupts zur selben Zeit eintreffen oder eine Quelle mehrere Interrupts sendet, ist gegeben. Letzteres wird durch eine Entprellschaltung oder einer Entprelllösung in Software gelöst. Ansonsten können zu viele Interrupts das System absorbieren und es “verhungert”.

2.4 Bluetooth Low Energy

Bluetooth Low Energy (BLE) bezeichnet eine Funktechnik, welche es ermöglicht, Daten zwischen Geräten auszutauschen. Der Vorteil von Bluetooth Low Energy ist der niedrigere Energieverbrauch im Gegensatz zum traditionellen Bluetooth. Das Bluetooth Low Energy Protokoll gehört zum Bluetooth Core Specification Version 4.0, wo auch das Classic Bluetooth Protokoll und das Bluetooth High Speed Protokoll enthalten sind. Die Bluetooth Core Specification Version 4.0 ist besser unter dem Namen Bluetooth Smart bekannt und wurde im Juli 2010 veröffentlicht (?).

2.4.1 BLE im Vergleich zu Bluetooth

Bluetooth Low Energy verwendet das gleiche Frequenzband wie das traditionelle Bluetooth, jedoch sind nur 40 Kanäle à 2 MHz verfügbar, anstatt 79 Kanäle à 1 MHz beim traditionellen Bluetooth. Ausserdem verbraucht BLE, wie der Name bereits indiziert, weniger Energie als andere Übertragungsmedien. So sendet BLE mit maximal 10 mW, was einer Reichweite von ca. 10 Metern entspricht im Gegensatz zu Klasse 1 Bluetooth-Geräten, welche mit 100 mW eine Reichweite von rund 100 Metern erreichen. Ein Vorteil der BLE-Technik ist, dass die Bauteile für eine BLE-Kommunikation relativ günstig sind und damit die Geräte ebenfalls günstiger hergestellt werden können (? , Abschnitt Bluetooth Range).

2.4.2 Advertising und Connected Mode

BLE wird vor allem für batterielose Sensoren verwendet, welche die Energie aus der Umwelt beziehen. Diese Sensoren arbeiten meist als Beacon, was bedeutet, dass sie Daten senden, ohne eine aktive Verbindung mit einem Gerät aufzubauen oder nur eine Verbindung auf Anfrage eingehen, diese jedoch nach kurzer Zeit wieder beenden. Dieser Modus nennt sich Advertising Mode, was vom Englischen advertisement stammt, es soll aussagen, dass der Beacon eine Werbung aussendet und diese nicht auf eine spezielle Person zugeschnitten ist, sondern an die breite Masse gesendet wird.

Eine aktive Verbindung ist bei den meisten Sensoranwendungen auch nicht notwendig, da die Daten einfach gesendet werden können und das empfangende Gerät entscheidet was mit den vorliegenden Daten gemacht wird. Wenn das Gerät mehr Informationen benötigt kann eine Verbindung aufgebaut werden. Trotzdem kann mit BLE eine aktive Verbindung eingerichtet werden, jedoch verbraucht eine aktive Verbindung mehr Energie, da Daten gesendet und empfangen werden müssen. Das bedeutet der Sensor kann nicht in einen Standby- Modus gehen, in welchem weniger Energie verbraucht wird, da auf ankommende Daten gewartet wird (? ,?).

2.4.3 BLE Pakete

Der Aufbau eines BLE Pakets ist in Abbildung 2.16. Als erstes wird ein Preamble gesendet, bestehend aus abwechselnden 1 und 0, womit der Empfänger sich auf die richtige Frequenz synchronisieren kann. Diese Preamble wird auch dafür verwendet, die Verstärkung des Empfängers einzustellen. Dies kann sehr wichtig sein bei Signalen, welche von einer grösseren Distanz versendet werden, da eine falsche Verstärkung des Signals in Fehlern resultieren kann.

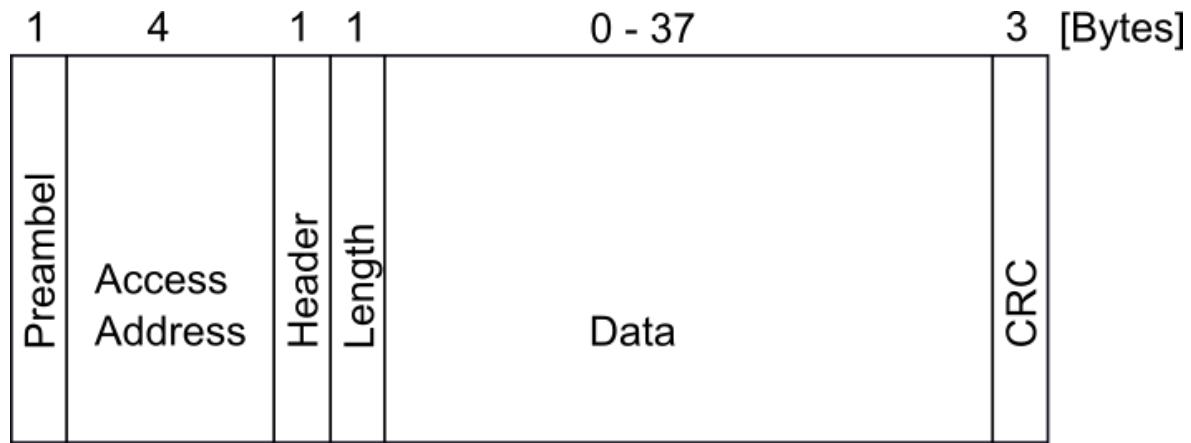


Abbildung 2.16: BLE Paketstruktur

Anschliessend wird die Access Address verschickt, anhand dieser Adresse kann der Empfänger die Nachricht einem ganz bestimmten Sender zuordnen und somit entscheiden, ob die Daten vom richtigen Sender kommen.

Der Header enthält Informationen zum Aufbau der Daten, welche folgen. Es gibt sieben verschiedene Arten von Aufbauten der Daten.

- ADV_IND – general advertising indication
- ADV_DIRECT_IND – direct connection indication
- ADV_NONCONN_IND – nonconnectable indication
- ADV_SCAN_IND – scannable indication
- SCAN_REQ – active scanning request
- SCAN_RSP – active scanning response
- CONNECT_REQ – connection request

Nachfolgend wird die Length eingereiht, welche Informationen über die Anzahl Bytes der Daten enthält. Es wird unterschieden zwischen der Länge eines Advertising Pakets und eines Data Pakets. Die Länge eines Advertising Pakets wird mit sechs Bits

dargestellt, welche die Werte von 6 – 37 einnehmen können. Ein Data Paket hingegen arbeitet nur mit fünf Bits, welche die Werte 0 – 31 einnehmen können.

Anschliessend werden die Nutzdaten übertragen, welche je nach gewählter Art, einen anderen Aufbau aufweisen. Es können zwischen 0 bis 296 Bits, also 0 bis 37 Bytes übertragen werden.

Abgeschlossen wird ein Paket mit dem CRC, welcher die Checksumme der Nachricht enthält. Die Checksumme wird über den Header, Length und die Nutzdaten gebildet (? , Kapitel 7.2).

3

Vorgehen

Das Ziel ist die Entwicklung eines Prototypen aus dem bestehenden Modell der Projektarbeit von ?. Die Vorgehensschritte sind in der untenstehenden Auflistung abgebildet. Sie bilden die Struktur dieses Kapitels. Die Definition der konkreten Schritte geschah in Absprache mit Prof. Dr. M. Meli und dem Research Assistenten Herr Dario Dünar vom Institut of Embedded Systems (InES). Auf der CD finden sich die Sitzungsprotokolle über den aktuellen Entwicklungsstand, die offenen Fragen und die Entscheidungen.

Liste der Arbeitsschritte

1. Inbetriebnahme Machbarkeitsstudie
2. Hardware entwickeln
3. Inbetriebnahme Prototyp
4. Energy und Power Management
5. Entwickeln einer BLE-Applikation

Die sprachliche Unterscheidung von Energy - und Power-Management entspricht der Unterscheidung zweier “Management-Teile” im Prototypen: Das Endprodukt regelt an zwei Stellen auf unterschiedliche Art die zur Verfügung stehende Energie. Der Begriff “Energy Management” wird in dieser Arbeit für das Sammeln und Weiterleiten von Energie über eine Hardwareimplementation gebraucht. Der Begriff “Power Management” wird für die Softwareimplementation gebraucht. Diese regelt, dass die zur Verfügung gestellte Energie nicht sofort verbraucht wird. Die sprachliche Trennung ist künstlich, denn in der Umsetzung spielen Hard- und Softwareregelung Hand in Hand. Die sprachliche Unterscheidung dient der Lesbarkeit und bezeichnet keinen physikalischen Unterschied.

3.1 Inbetriebnahme Machbarkeitsstudie

Ziel der Inbetriebnahme des Aufbaus der vorangehenden Arbeit von ? ist es, zu definieren, welche Funktionalitäten verbessert werden sollen. Zur Orientierung werden im ersten Unterkapitel 3.1.1 die Funktionsblöcke und deren Aufgaben festgehalten. Danach wird das Verhalten des Vorgängermodells in Unterkapitel 3.1.2 ausgemessen. Aus der Analyse entsteht die in Unterkapitel 3.1.3 aufgelistete erste Optimierungsliste. Als letztes folgt eine Vertiefung in das auffällige Verhalten des Eingangssignals in Unterkapitel 3.1.4.

3.1.1 Funktionsblöcke

Der Bicycle Computer besteht aus vier Funktionsblöcken, die in der Abbildung 3.1 dargestellt sind. Der erste Funktionsblock ist der Harvester (siehe 1) in der Abbildung 3.1). Die Aufgabe des Harvesters besteht darin, Energie zu Ernten und dem nächsten Funktionsblock (Nummer 2) in der Abbildung 3.1 zur Verfügung zu stellen. Der zweite Funktionsblock wird als Energy Management-Teil in der Arbeit bezeichnet. Die Aufgabe des zweiten Blocks ist es, Energie zu Sammeln und kontrolliert an die Verbrauchsstelle freizuschalten. Detaillierte Informationen finden sich in den Theoretischen Grundlagen im Unterkapitel 2.2. Der dritte Funktionsblock ist der Ort, an dem die Energie verbraucht wird. In dieser Arbeit dient die Energie dem Betreiben von Sensoren und dem Versenden derer Daten. Dies wird auf dem Sensortag (siehe Anhang C) umgesetzt. Der Grund dafür wird in der Einleitung des Kapitels dargelegt. Der letzte Funktionsblock bezeichnet das Ziel, das Erhalten von Sensordaten in einer Applikation.

rekter Sen-
Tag Name

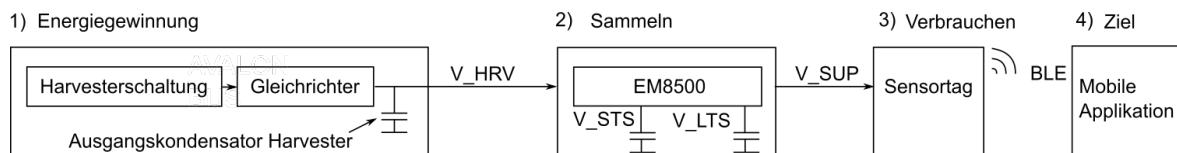


Abbildung 3.1: Funktionsblöcke Bicycle Computer

Bezeichnung	Beschreibung
V_HRV	Ausgangsspannung Harvesterquelle, Eingangsspannung Energy Management
V_STS	Spannung am STS-Kondensator (Primärspeicher)
V_LTS	Spannung am LTS-Kondensator (Sekundärspeicher)
V_SUP	Ausgangsspannung Energy Management, Eingangsspannung Sensortag
BLE	Senden der Daten per Bluetooth Low Energy (siehe 2.4)

Den Funktionsblöcken sind Spannungsbezeichnungen sowie Kondensatoren beigefügt. Dies, weil bei der Beschreibung des Verhaltens des Vorgängermodells, diese Spannungslevel für die Funktionsbeurteilung wichtig werden. Die nachfolgende Legende beschreibt die Beschriftung näher.

3.1.2 Verhalten des Vorgängermodells

Die Inbetriebnahme bestätigte das in der Dokumentation von ? beschriebene Verhalten. Die Abbildung 3.2 zeigt den zeitlichen Verlauf der Energiestände zwischen den Funktionsblöcken (siehe Abbildung 3.1) und an den Speicherelementen. Die Legende zur Abbildung 3.2 erklärt die Signale.



Abbildung 3.2: Spannungswerte Modell der Machbarkeitsstudie

Channel	Farbe	Beschreibung
CH1	gelb	Spannungsverlauf V_HRV
CH2	blau	Spannungsverlauf am STS-Kondensator
CH3	violet	Spannungsverlauf am LTS-Kondensator
CH4	grün	Ausgangsspannung nach Energy Management Eingangsspannung Sensorstag

Im Folgenden werden die einzelnen Spannungsverläufe chronologisch der Kanalnummer nach analysiert. Kanal 1 spiegelt die Spannung am Harvesterausgang wieder (V_HRV).

Gemäss Theorie 2.2.3 bzw. gemäss Datenblatt des EM8500 ist der EM8500-Chip für ein DC-Signal ausgelegt. Er geht von einem regelmässigen Eingangssignal aus und regelt die Spannung auf den MPPT. Diese Regelung sollte wie in Abbildung 2.12 aussehen. Das reale Signal entspricht nicht diesem Verhalten. Die Regelung ist abrupt und überspringt mehrere Spannungslevel. Der Ursache für die schlechte Regelung soll nachgegangen werden.

Kanal 2, blau, gibt den Spannungsverlauf am Hauptspeicher, dem Primärspeicher, der im Datenblatt von EM8500 STS heisst, wieder. Das Energy Management des Vorgängermodells setzt den Schwellwert für den Primärspeicher (STS) mit 3.6 V hoch (Details zu den Schwellwerten im Kapitel Theoretische Grundlagen 2.2). Der hohe Wert erklärt sich durch das Ziel, genug Energie für ein konstantes Paketversenden zu haben. Dies gelingt für fünf BLE-Pakete. Danach reicht die Energie nicht mehr aus. V_SUP, grünes Signal, wird nicht mehr gespiesen. Nach 30 s ohne Pakete senden ist wieder genug Energie für weitere 5 Datenpakete gespeichert.

In der Auswertung fiel uns auf, dass dieser Signalverlauf nur bei einer Geschwindigkeit von 45 km/h möglich ist. Die Speicherkapazität von $470 \mu\text{F}$ und ein Schwellwert der Spannung von 3.6 V ist mit normaler Geschwindigkeit (10 km/h) nach 30 min nicht zu erreichen. Fährt man rund 45 km/h so erhält man die in der Abbildung 3.2 gezeigte Ladezeit von rund 25 s. Eine exakte Geschwindigkeitsmessung ist zu Beginn der Arbeit nicht möglich. Das Rad wird von Hand gedreht und mit einem Metronom wird die Umdrehungsgeschwindigkeit vorgegeben. Bei einem Radumfang von 2.04 m und einer Zeitdifferenz von 160 ms zwischen den Reed-Inpulsen, ergibt sich die Geschwindigkeit von 45 km/h. Da diese Messmethode über längere Zeit nicht sehr genau ist, bestand eine der Aufgaben nach der Inbetriebnahme im Organisieren eines Messaufbaus. Dieser professionellere Messaufbau wird im Anhang E beschrieben.

Kanal 3, das rosa Signal, zeigt die Spannung am Long Time Speicher. Die Inbetriebnahme zeigt, dass sich der LTS lädt. Es erstaunt jedoch, dass seine geerntete Energie nicht verwendet wird. Der Spannungswert von LTS geht nie herunter. Die Vermutung ist, dass der eingestellte Schwellwert für den Bezug von Energie von LTS 2.10 nicht stimmt.

Kanal 4, das grüne Signal zeigt, die Speisung des Sensortags. Im Modell der Projektarbeit steuert der Microkontroller des Sensortags den Verbrauch. Alle 10 s wacht das System auf, bezieht Energie vom EM8500-Ausgang für das Senden eines Paketes und geht dann wieder schlafen. Das Aufwachintervall ist fix.

3.1.3 Optimierungsliste

Aus den Messungen der Inbetriebnahme konkretisierten sich die generellen Aufgaben, die in der ersten Liste der Arbeitsschritte zu Beginn des Vorgehens beschrieben wurden. Folgende vier Punkte sollen durch den Prototypen verbessert werden:

- Der Verlauf des Harvester-Eingangs wechselt abrupt. Der Eingang soll besser geregelt werden.
- Für das Laden des Primärspeichers von $470 \mu\text{F}$ in 25 s benötigt es eine Geschwindigkeit von mehr als 60 km/h. Die Harvesterschaltung soll so weiterentwickelt werden, dass bei 10 km/h genug Energie zum Senden von BLE-Paketen besteht.
- Das zweite Speicherelement, der LTS, entlädt sich nicht. Dadurch kann seine Energie nicht verwendet werden. Die Schwellwerte am EM8500 und ev. die Kondensatorenwerte sollen so angepasst werden, dass sich der zweite Kondensator entlädt
- Die Energie wird statisch nach einem fixen Zeitintervall von 10 s genutzt. Das Zeitintervall soll der Geschwindigkeit angepasst werden. Bei höherer Geschwindigkeit soll das Intervall kürzer werden.

Im Resultatteil Kapitel 4 werden die Fortschritte in diesen vier Punkten ausgewiesen.

3.1.4 Vertiefung Harvestereingangs

Herr Ives Théoduloz hatte uns im Rahmen eines Besuchs auf die Problematik, einer zu grossen Kapazität am Eingang des EM-Chips, hingewiesen. Laut seiner Aussage sollte der Kondensator am Eingang des EM-Chips, bzw. dem Ausgang des Harvesters am besten kleiner sein als $4.7 \mu\text{F}$. Bisher wurde im Aufbau der Machbarkeitsstudie ein Kondensator mit $470 \mu\text{F}$ verwendet. Herr Ivex Théoduloz meinte, dass dieser Kondensator viel zu gross sei und dass die Regelung am Eingang des EM-Chips möglicherweise nicht richtig funktionieren würde. Daher wurde dieses Problem näher untersucht und in mehreren Schritten optimiert.

Im ersten Schritt wurde der Kondensator verkleinert, bis die Rippelspannung noch annehmbar war. Die Rippelspannung bei dem bestehenden Kondensator von $470 \mu\text{F}$ beträgt ca. 10 mVpp. Die Rippelspannung bei einer Kapazität von $47 \mu\text{F}$ beträgt bereits ca. 40 mVpp, jedoch ist die Rippelspannung noch in einem Bereich der annehmbar scheint. Bei einer Kapazität von $10 \mu\text{F}$ liegt die Rippelspannung bei ca. 500 mVpp. Die Abbildung 3.3 veranschaulicht die Rippelspannung von einem Kondensator mit $47 \mu\text{F}$, die Abbildung 3.4 zeigt die Rippelspannung über einem Kondensator von $10 \mu\text{F}$.

Jedoch muss auch ein Blick auf die Spannung am Harvesterausgang geworfen werden, wenn dieser Ausgang mit dem EM-Chip belastet wird. Es kann beobachtet werden, dass der Eingang des EM-Chips ein fluktuierendes Spannungslevel (siehe Abbildung 3.5) erhält.

Problematisch ist, dass die Spannung am Eingang des EM8500-Chip wiederholt unter die 0.3 V Grenze fällt, bei einer Spannung von 0.3 V kann keine Energie gespeichert werden. Die Regelung am Eingang des EM-Chips kann mit der anliegenden Rippel-

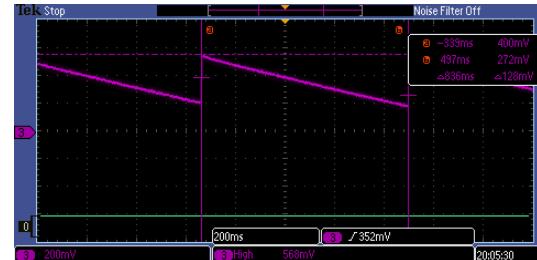
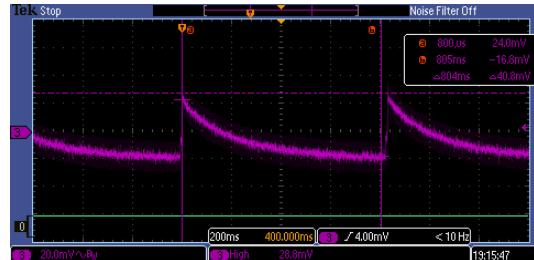
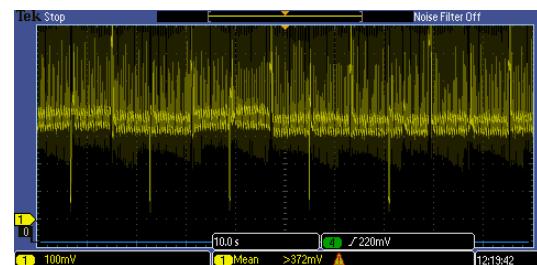
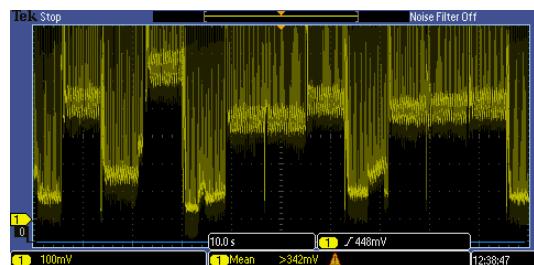


Abbildung 3.5: Spannung am EM8500-Eingang über einem $47 \mu\text{F}$ Kondensator



Spannung nicht richtig arbeiten. Die Rippelspannung muss verringert werden, bzw. die Kapazität am Ausgang des Harvesters muss erhöht werden.

Das Problem ist, dass durch den Rippel die periodische Open Loop Spannungsmessung keine korrekten Messwerte erhält. Sobald die Last vom Harvester ausgang abgehängt wird, steigt die Spannung über dem Kondensator, jedoch wird die Open Loop Spannung bei einer grossen Kapazität nicht erreicht, da die Zeit, bis der Kondensator komplett geladen ist, sehr gross ist. Jedoch bedeutet eine niedrige Kapazität, dass die Rippelspannung relativ hoch ist und je nach Zeitpunkt der Open Loop Messung eine andere Spannung am Kondensator ergibt und die Spannung am Kondensator steigt bei jeder Messung auf ein anderes Spannungslevel.

Es musste ein Kompromiss gefunden werden, so dass die Spannung am Eingang des EM-Chips auf ein konstantes Level geregelt werden kann. Der Kondensator sollte nicht zu klein sein, damit der Rippel nicht zu gross wird. Andererseits darf der Kondensator nicht zu gross sein, da ansonsten die Open Loop Messung keine richtigen Werte liefert. Der Kompromiss stellt der $100 \mu\text{F}$ Kondensator dar, bei diesem ist die Spannung am Eingang relativ konstant und es kann damit gearbeitet werden (siehe Abbildung 3.6).

Aus den Beobachtungen ergaben sich folgende Aufgaben für die Entwicklung des Prototypen:

1. Die Harvesterschaltung funktioniert nicht optimal. Die Auswirkung des zu hohen Kondensator vor Harvestereingang soll getestet werden

2. Die Schaltung soll für eine Geschwindigkeit von 10 km/h ausgelegt werden
3. Die Konfigurationen beim EM8500 sollen überarbeitet werden, sodass LTS genutzt wird
4. Das Senden der Pakte soll der Geschwindigkeit angepasst werden

Punkte 1 und 2 haben Auswirkung auf das Layout, Punkte 3 und 4 auf die Entwicklung des Energymanagements und der Firmware für das TI-SensorTag.

3.2 Hardware entwickeln

Aus der Inbetriebnahme der Machbarkeitsstudie wurden einige Punkte ersichtlich, welche mit einer neuen Leiterplatte, bzw. einer neuen Hardware verbessert werden können. Im speziellen muss die Harvesterschaltung genauer angeschaut werden und es soll versucht werden, die nachfolgenden Punkte aus der Inbetriebnahme der Machbarkeitsstudie umzusetzen.

Die Schaltung soll für eine Geschwindigkeit von 10 km/h ausgelegt werden.

Ausserdem soll die neue Leiterplatte den fliegenden Aufbau der Harvesterschaltung und den EM-Chip beherbergen. Es wird in diesem Schritt darauf verzichtet, das TI-SensorTag ebenfalls in die neue Leiterplatte zu integrieren, die Hardware kann gut in mehreren Schritten überarbeitet werden.

3.2.1 Das Schema

Bevor ein Leiterplattenlayout entwickelt werden kann, muss das Schema gezeichnet werden. Es wurden verschiedenste Vorgaben von den Dozenten vorgegeben, welche im besten Fall alle eingehalten werden.

1. Die Grösse der Leiterplatte soll die Grösse des TI-SensorTags nicht überschreiten.
2. Alle Netze sollen mit Testpunkten ausgestattet werden.
3. Alle Anschlüsse des TI-SensorTags auf der Leiterplatte zugänglich sein.
4. Alle Testpunkte des TI-SensorTags sollen im Rastermass 2.5 mm angeordnet werden.
5. Strommesspunkte sollen an der Speisung des TI-SensorTags, des Longterm Storage und des Shortterm Storage angebracht werden. (optional)

Diese Vorgaben sollen die Leiterplatte für eventuelle Laborübungen der Schule verwendbar machen. Das Schema besteht in den Grundzügen aus vier Teilen:

1. Harvesterschaltung
2. EM-Chip inklusive zugehöriger Peripherie
3. Energiespeicher
4. Umlauferfassung

a. Harvesterschaltung

Die Harvesterschaltung wurde in der Machbarkeitsstudie als fliegender Aufbau realisiert, was zu einigen Problemen führen kann, da viele lange Kabel verwendet wurden und somit die Signallaufwege lang sind. Bisher hat dies keine Probleme verursacht, doch es geht wichtige Energie in den Kabeln verloren. Ebenfalls wurde in der Inbetriebnahme bemerkt, dass die gewonnene Leistung sehr gering ist, weswegen die Schaltung in einem zweiten Schritt optimiert werden muss.

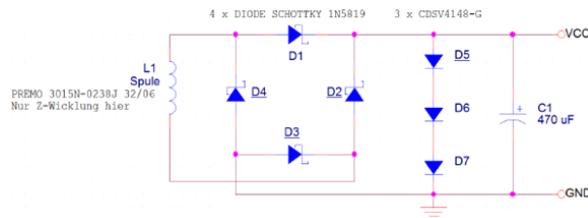


Abbildung 3.7: Harvesterschaltung der PA15

Die Harvesterschaltung der Abbildung 3.7 wurde im Rahmen der Machbarkeitsstudie erarbeitet. Die Schaltung besteht aus wenigen Teilen, die Spannungsbegrenzung wurde mit drei Dioden in Durchlassrichtung realisiert, hier gibt es wahrscheinlich bessere Möglichkeiten die Spannung zu begrenzen, ohne unnötig Leistung zu verschwenden.

b. EM8500-Chip

Das EM-Evaluationsboard soll ebenfalls auf der neuen Leiterplatte Platz finden, das Schema inklusive aller Kondensatoren ist im Datenblatt zu finden. Jedoch wurden die Jumper nicht übernommen und nur einige Stecker wurden übernommen, bzw. mit eigenen Signalen ergänzt.

c. Energiespeicher

Die Energiespeicher waren zum Zeitpunkt der Entwicklung der Hardware noch nicht definiert. Als Platzhalter wurden zwei Elektrolytkondensatoren mit den Werten aus der Machbarkeitsstudie eingefügt.

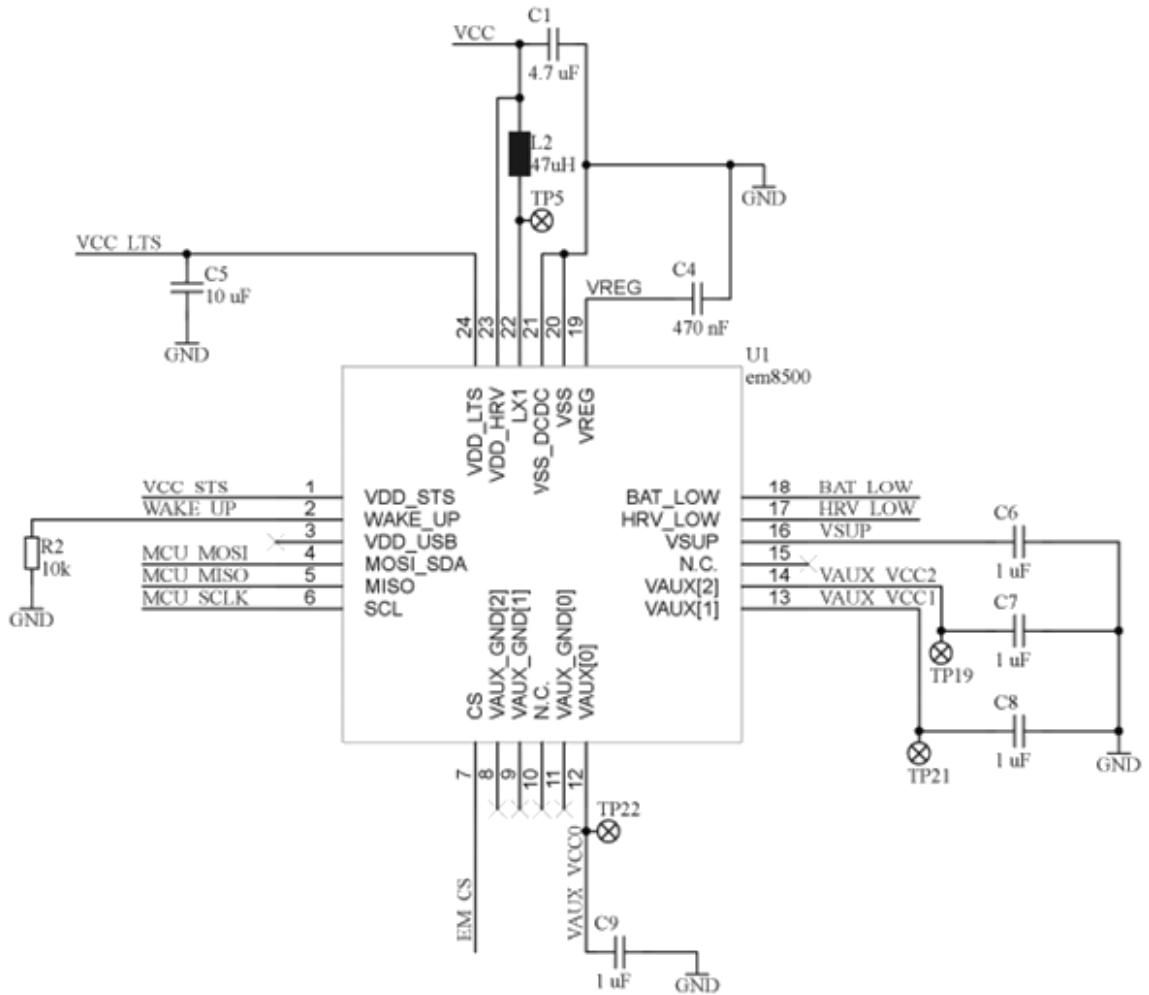


Abbildung 3.8: Schema des EM-Chips mit der benötigten Peripherie

d. Umlauferfassung

Die Umlauferfassung kann nicht mit der verwendeten Spule realisiert werden, da bei der Energiegewinnung mit der Spule, das Signal gleichgerichtet wird. Die nicht benutzten Wicklungen der X- und Y-Wicklung liefern kein verwertbares Signal, was bereits in der Machbarkeitsstudie bewiesen wurde. Aus diesem Grund wird der Magnetdurchlauf mit einem Reedswitch detektiert und an das TI-SensorTag weitergegeben. Ein Magnetdurchlauf erzeugt einen positiven Puls, solange sich der Magnet in der Reichweite des Reedswitch befindet.

3.2.2 Bauteildefinition und Optimierung

Nachdem klar war, welche Teile auf der Leiterplatte Platz finden müssen, konnte die Optimierung der Schaltung in Angriff genommen werden. Ziel der Optimierung war es,

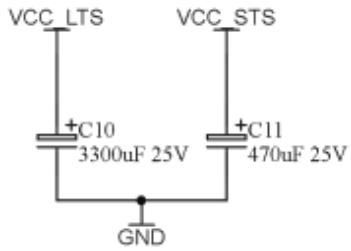


Abbildung 3.9: Schema der Energiespeicher (Elkos sind nur Platzhalter)

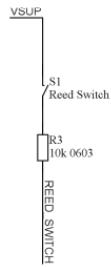


Abbildung 3.10: Schema Umlauferfassung

bei 10 km/h genügend Energie zu gewinnen, damit das TI-SensorTag damit versorgt werden könnte. Dafür musste vor allem die Harvesterschaltung optimiert werden, damit keine Energie verloren geht. Es wurden folgende Punkte angeschaut und versucht zu optimieren:

1. die Spule
2. der Gleichrichter
3. der Limiter

a. Die Spule

Die Spule gewinnt die Energie aus dem an einer Speiche des Fahrrads befestigten Magneten. Eine stärkere Spule könnte mehr Energie aus dem vorbei schnellenden Magneten gewinnen. Entscheidend ist das die Fläche der Spule nicht vergrössert werden darf, bestenfalls sollte eine kleinere Spule gefunden werden, welche mehr Energie gewinnt. Gemäss der Formel 2.4 ist für die gewonnene Energie vor allem die Wicklungszahl entscheidend, jedoch wird bei den Spulen meistens nur die Induktivität angegeben. Die Induktivität hängt quadratisch von der Wicklungszahl und linear von der Fläche (siehe Formel ?) ab.

$$L = \mu_0 \times \frac{N^2 \times A}{l} \quad (3.1)$$

Es wurde nach einer Spule mit einer höheren Induktivität und der gleichen Fläche gesucht, somit können nur noch zwei Variablen sich verändern. Zum einen kann sich die Wicklungszahl verändern und zum andern kann sich die Länge der Spule verändern. Die Spule von Würth Elektronik mit der Bezeichnung 74458308 hat eine ähnliche Fläche und eine grössere Induktivität, was auf den ersten Blick sehr vielversprechend aussieht.

Die Messung der erzeugten Spannung über der Spule hat ergeben, dass die Spule von Premo mit der Bezeichnung 3015N 0238J 3206, welche bisher verwendet wurde, eine höhere Spannung erzeugt als die neue Spule von Würth. Die Spule von Premo hat bei den Geschwindigkeiten von 10 km/h und 20 km/h eine grössere induzierte Spannung. Die Spule von Würth hat eine höhere Spannung bei den Geschwindigkeiten von 15 km/h und 40 km/h. Jedoch sollte die Schaltung für 10 km/h optimiert werden, somit muss der Spule von Premo der Vortritt gewährt werden. Nachfolgend wird der Unterschied zwischen den induzierten Spannungen in den Spulen ersichtlich (siehe Abbildung 3.11). Der Unterschied ist nicht gross, jedoch muss hier die Spule von Premo bevorzugt werden, da die induzierte Spannung grösser ist.

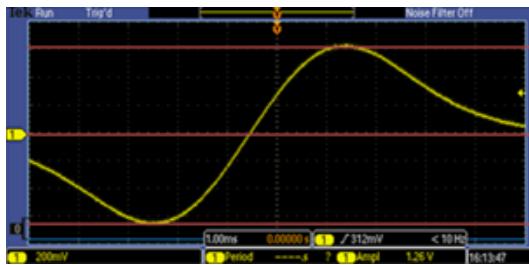


Abbildung 3.11: Spannung über der Spule von Premo, Geschwindigkeit 20 km/h

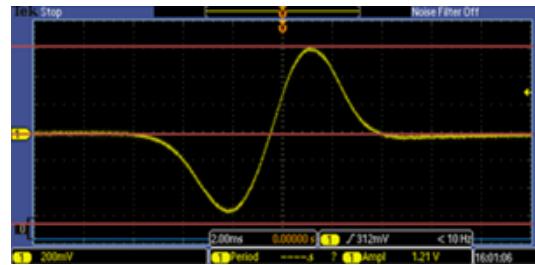


Abbildung 3.12: Spannung über der Spule von Würth, Geschwindigkeit 20 km/h

b. Der Gleichrichter

Der Gleichrichter aus der Machbarkeitsstudie bestand aus vier Dioden vom Typ 1N5819. Diese Dioden sind nicht für die LowPower-Anwendung ausgelegt, ausserdem sind die Dioden nicht in einem Gehäuse verbaut. Wichtig ist bei dem Gleichrichter, dass die Leckströme möglichst klein sind und die Schwellenspannung sollte ebenfalls möglichst klein sein, damit wenig Energie verbraucht wird.

Es wurde als erstes eine LowPower-Diode, mit der Bezeichnung HSMS-286P, getestet. Die Erwartungen waren entsprechend hoch, da diese Dioden für LowPower-Anwendungen spezialisiert sind. Die Spule von Premo wurde als Quelle verwendet, um zu sehen, wie die Spannung nach dem Gleichrichter aussieht. Die Spannung nach dem Gleichrichter bestehend aus den Dioden vom Typ 1N5819 ist bei allen getesteten Geschwindigkeiten, also 10 km/h, 15 km/h, 20 km/h und 40 km/h, höher als beim Gleichrichter bestehend aus den Dioden vom Typ HSMS-286P. Der Spannungsunterschied liegt im

Minimum bei ca. 40mV. Der grösste Unterschied ist bei der Geschwindigkeit von 15 km/h ersichtlich, was in der Abbildung 3.13 gezeigt wird.

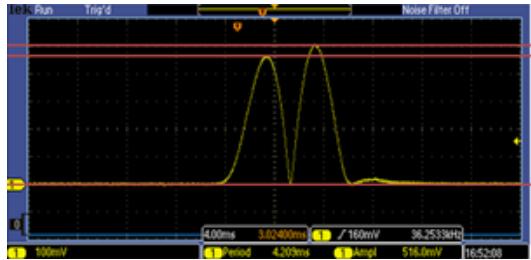


Abbildung 3.13: Gleichrichter 1N5819

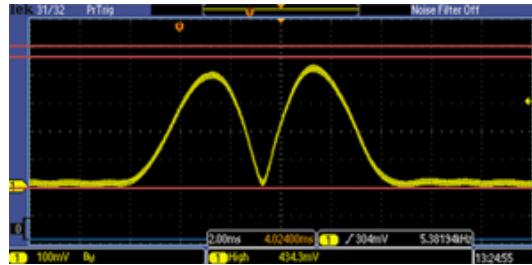


Abbildung 3.14: Gleichrichter aus HSMS-286P, 15 km/h

Als nächstes wurde ein Gleichrichter aus den Dioden vom Typ BAT54 getestet. Die Spannung nach dem Gleichrichter bestehend aus 1N5819 Dioden ist bei den Geschwindigkeiten von 15 km/h, 20 km/h und 40 km/h höher als nach dem Gleichrichter bestehend aus BAT54 Dioden. Der Spannungsunterschied liegt bei ca. 100 mVpp. Der Unterschied ist marginal, jedoch muss hier der Gleichrichter aus 1N5819 Dioden bevorzugt werden. Nur bei einer Geschwindigkeit von 10 km/h ist der Gleichrichter bestehend aus BAT54 Dioden besser als der Gleichrichter bestehend aus 1N5919 Dioden. Der Spannungsunterschied liegt hier bei ca. 10mVpp. Dieser Unterschied ist vernachlässigbar, in Angesicht dessen, dass der Gleichrichter bestehend aus 1N5819 Dioden in allen anderen getesteten Geschwindigkeiten besser ist.

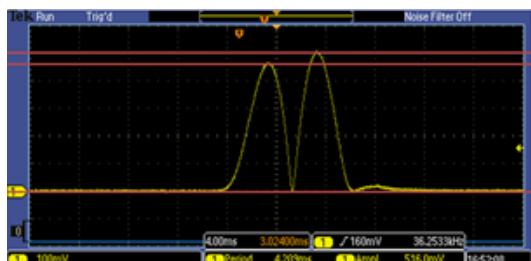


Abbildung 3.15: Gleichrichter 1N5819

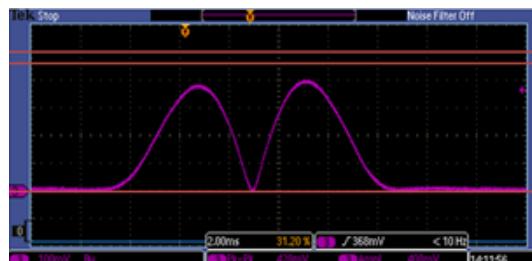


Abbildung 3.16: Gleichrichter aus BAT54, 15 km/h

c. Der Limiter

Die Spannungsbegrenzung, nachfolgend der Limiter genannt, ist ein sehr kritischer Teil der Harvesterschaltung, da die Spannung am EM-Chip Eingang nicht höher als 2 V sein darf, da ansonsten der EM-Chip beschädigt werden kann. Trotzdem soll möglichst wenig Energie verloren gehen, wenn die Spannungsbegrenzungsschaltung ihre Arbeit

verrichtet. Bisher wurden drei Dioden in Durchlassrichtung in Serie geschaltet, um die Spannung zu begrenzen.

Herr Erich Ruff hat eine Spannungsbegrenzungsschaltung entwickelt, welche er uns freundlicherweise zur Verfügung stellte. Diese Schaltung wurde mit dem Limiter verglichen, welcher aus drei Dioden bestand.

Die Spannung nach dem Dioden-Limiter ist bei 10 km/h, 15 km/h und 20 km/h höher, jedoch ist die Rippelspannung ebenfalls höher. Nur bei einer Geschwindigkeit von 40 km/h ist die Spannung nach dem Limiter von Herr Erich Ruff, nachfolgend FET-Limiter genannt, besser sowohl bei Spannungslevel als auch bei der Rippelspannung. Die Abbildung 3.17 zeigt, dass der Dioden-Limiter eine höhere Spannung liefert, jedoch ist die Rippelspannung ebenfalls höher.

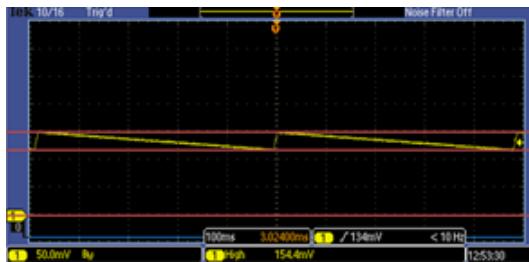


Abbildung 3.17: Links: Dioden-Limiter

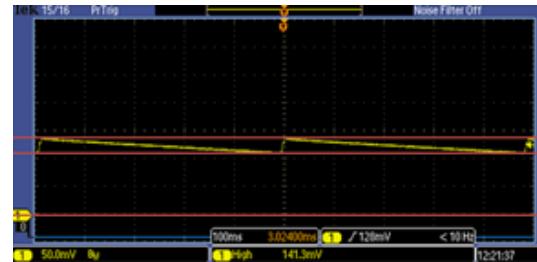


Abbildung 3.18: Rechts: FET-Limiter, 15 km/h

3.2.3 Layout

Schlussendlich musste aus den optimierten Teilen des Schemas eine Leiterplatte gestaltet werden. Die meisten Footprints waren bereits in den Bibliotheken des IneS vorhanden, einige mussten neu gezeichnet werden.

a. Positionierung

Die Positionierung der einzelnen Teile kann sehr wichtig sein, da mit einer guten Positionierung bereits unnötige Leiterbahnverläufe verhindert werden können. Ebenfalls können gut positionierte Bauteile die Spannungspegel stabilisieren. Es wurde darauf geachtet, dass die Bauteile, welche zu einem Block gehören, nahe beieinander zu platzieren, um unnötig lange Signallaufwege zu verhindern.

Die Bauteile der Harvesterschaltung wurden als Block so nahe wie möglich beieinander platziert und wo immer möglich wurden die Speisungsleitungen mit 20 Mil gezogen,

damit der Widerstand der Leiterbahn (?) möglichst klein gehalten wurde. Der Leiterwiderstand konnte so minimiert werden, was verhindert, dass die Energie in den Leiterbahnen verschwendet wird.

$$R = \rho \times \frac{l}{A} \quad (3.2)$$

Der Widerstand bei 10 Mil ist somit ca. 1.9 mΩ pro Meter, der Leiterwiderstand bei einer Leiterbahnbreite von 20 Mil ist ca. 1.0 mΩ pro Meter. Problematisch ist, dass sich der Block der Harvesterschaltung etwas entfernt vom Block des EM-Chips befindet. Das bedeutet, dass die Leiterbahn mit der Speisung des EM-Chips mehrere Zentimeter zurücklegen muss. Sicherlich ist der Unterschied im Widerstand nicht sehr gross, doch die Energie, welche in der Leiterbahn verloren gehen würde, konnte so halbiert werden.

Der wichtigste Aspekt der Platzierung des EM-Chips war, dass die Stützkondensatoren so nah wie möglich am EM-Chip platziert wurden, damit die Spannung am EM-Chip so konstant wie irgend möglich gehalten werden kann.

Ein weiterer wichtiger Punkt war die Platzierung des Steckers, welcher unsere Leiterplatte mit dem TI-SensorTag verbindet. Durch eine Falschplatzierung kann es hier passieren, dass die beiden Leiterplatten nicht korrekt übereinander ausgerichtet sind, wenn sie aufeinander gesteckt werden. Das ist eher ein ästhetisches Problem, jedoch kann das auch Probleme beim Einbauen in ein Gehäuse bereiten. Zu dem Stecker gehört auch die Platzierung der Testpunkte, welche direkt mit dem Stecker verbunden sind. Gemäss dem Wunsch der Betreuer wurde hier ein Rastermass von 2.5 mm der Testpunkte eingehalten, so dass eine Steckerleiste eingelötet werden könnte. Problematisch ist jedoch, dass die Testpunkte einen grossen Raum der Leiterplatte einnehmen, wie in Abbildung 3.19 ersichtlich.

b. Das erste Layout

Die erste Version der Leiterplatte war mit vielen Forderungen der Betreuer ausgestattet. Alle Netze wurden mit Testpunkten ausgestattet. Die Testpunkte des Steckers wurden im Rastermass 2.5 mm angeordnet und die Netze der Spannung nach dem Harvester. Die Spannung des Short- und Long-Term-Storage wurden mit Strommesspunkten ausgestattet. Es wurden ebenfalls Montagelöcher platziert, jedoch sind diese sehr minimalistisch, da nur eine M2-Schraube durchpasst. Besser wäre es, Montagelöcher für M3-Schrauben zu platzieren, doch der Platz auf der Leiterplatte ist sehr limitiert. Die Leiterplatte ist nur 33 x 42 mm gross, was ein Millimeter breiter ist als das TI-SensorTag. Der Anschluss der Energiespeicher wurde so realisiert, dass die Energiespeicher nicht auf der Leiterplatte Platz finden, da der Platz nicht ausreicht.

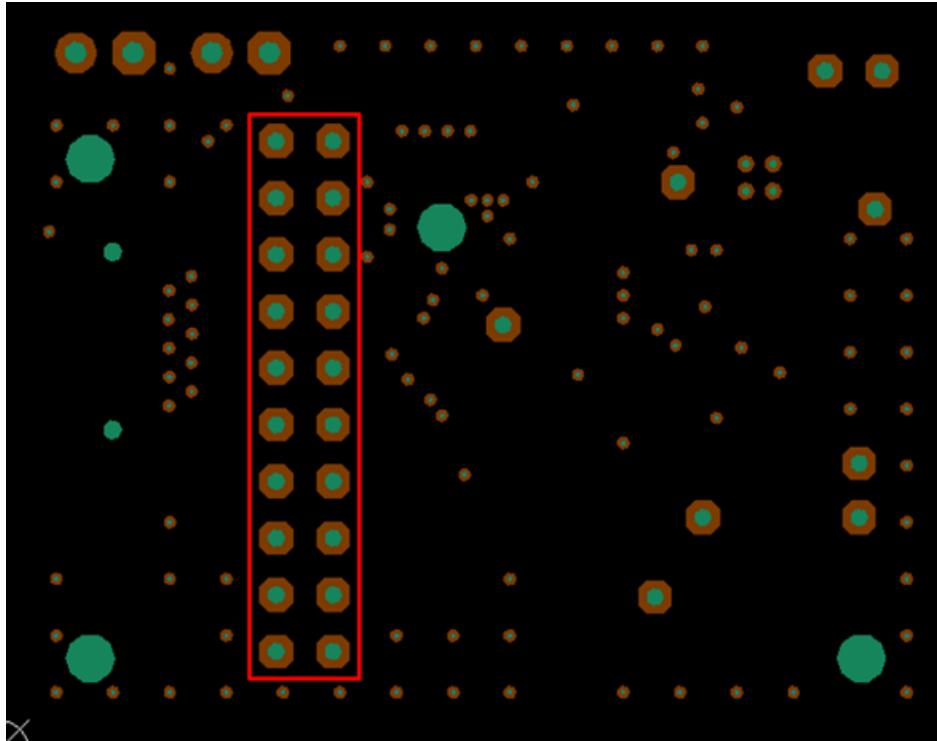


Abbildung 3.19: Pads der Leiterplatte, rot eingerahmt die Testpunkte des Steckers

c. Das Redesign

Das Layout wurde von Herr Olivier Rion begutachtet. Seine Kritikpunkte werden nachfolgend aufgelistet:

Kritikpunkte am Schema:

1. VCC sollte in Pfeil sein: Das Symbol für VCC und generell alle Speisungen sollten als Pfeil im Schema dargestellt werden.
2. Generell für eine erste Version sind viele TP gut.
3. Auf dem Schema fehlt noch ein Symbol für die GND Pads für den KO: Auf dem Top-Layer des Layouts wurde eine Fläche von Lötstopplack freigestellt, dies sollte auf dem Schema verzeichnet werden.
4. Die freien Pins auf X1 sollten mit IO angeschlossen sein: Die nicht verwendeten Pins vom EM8500-Chips sollten mit dem Stecker X1 verbunden werden, jedoch ist das problematisch, da eine direkte Verbindung zum Mikroprozessor Energie verbrauchen kann, solange der Mikroprozessor nicht gestartet ist.
5. Unten links auf dem Schema fehlt die Versionen, die Beschreibung, usw.
6. Eine kleine Beschreibung beim Harvester wäre gut, z.B. Prinzip mit T1, Spannungsregulierung usw.: Eine Beschreibung hilft die Schaltung schneller zu verstehen, dass macht eine Übergabe an die nachfolgenden Studenten einfacher.

Kritikpunkte am PCB:

1. Die Grenze vom Print sollte mit Mechanical und keepOutLayer gemacht werden.
2. Loch oben links: Die Leiterbahnen sind zu nah um die Löcher platziert, es sollte ein keepOutLayer um die Löcher platziert werden.
3. Die Leiterbahnen sollten immer in Gruppen platziert werden.
4. GND kann Kontakt mit den Schraubenköpfen haben: Es muss sichergestellt werden, dass die Schraube keinen Kontakt mit den Leitungen hat und dass die Flächen nicht unter dem Schraubenkopf platziert werden.

Wir sind dankbar für konstruktive Kritik, jedoch kann diese Kritik aus zeitlichen Gründen noch nicht umgesetzt werden. Neben den Kritikpunkten von Herr Rion wurden während der Arbeit noch weitere Kritikpunkte ersichtlich.

1. Die Testpunkte auf der Leiterplatte müssen neu platziert werden, dass sie Teilweise auf der Unterseite nicht angelötet werden können, da die Spule sie verdeckt.
2. Die Abstände zwischen den Testpunkten müssen vergrössert werden, damit eine KO-Sonde gut daran befestigt werden kann ohne einen Kurzschluss zu verursachen.
3. Der Footprint vom Stecker X1 muss überarbeitet werden, da dieser falsch erfasst wurde.
4. Die Position des Steckers X1 muss verändert werden, da die Position nicht mit dem Gegenstück des TI-SensorTags übereinstimmt.
5. Die Netze VCC_STS, VCC_LTS und VREG dürfen nicht mit dem Stecker X1 verbunden werden, da die Energie über diese Anschlüsse verloren geht.

Die Punkte c bis e, welche den Stecker X1 betreffen konnten vor Abschluss noch überarbeitet werden, jedoch wäre die Lieferzeit für die Leiterplatte zu lang, als dass die Leiterplatten vor Abgabe der Arbeit eintreffen würden. Darum wurden alle Messung mit der ersten Version der Leiterplatte durchgeführt.

nensklärung:
IRV =
C

sprotokolle
benennen

3.3 Inbetriebnahme des Prototypen

Nach der Entwicklung der neuen Hardware musste diese getestet werden, um die Funktionsweise mit den bisherigen Messungen zu verifizieren. Es werden generell zwei wichtige Messstellen in der Abbildung 3.20 ersichtlich. Die erste befindet sich nach dem Harvester und vor dem EM8500-Chip, die zweite wichtige Messstelle befindet sich zwischen

dem EM8500-Chip und dem TI-SensorTag. Es ist wichtig die Leistungs- oder Energiemessung durchzuführen, da diese Werte essentiell für die Einstellung des EM8500-Chips und die Entwicklung der Firmware für das TI-SensorTags sind (siehe Abbildung 3.20).

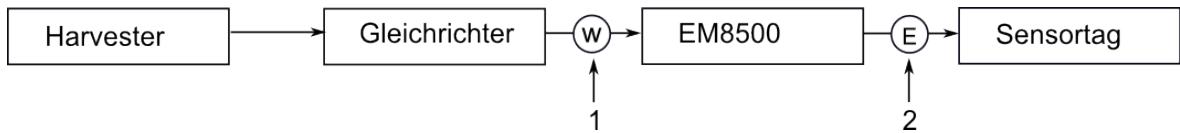


Abbildung 3.20: Messstellen am Prototypen

3.3.1 Testen der Harvesterschaltung

a. Leistungsverifizierung

Es wurde erneut eine Leistungskennline der Harvesterschaltung aufgenommen, um die verfügbare Leistung zu verifizieren. Wichtig ist, dass die verfügbare Leistung im gleichen Bereich ist, wie beim fliegenden Aufbau.

Tabelle 3.1: Leistung des fliegenden Aufbaus

Geschwindigkeit	maximale Leistung	MPP-Ratio
10 km/h	12.87 μ W	43.23 %
20 km/h	43.35 μ W	45.51 %
40 km/h	250.26 μ W	48.44 %

Tabelle 3.2: Leistung der neuen Leiterplatte

Geschwindigkeit	maximale Leistung	MPP-Ratio
10 km/h	10.17 μ W	56.99 %
20 km/h	48.57 μ W	61.82 %

Die maximal zur Verfügung stehende Leistung wurde etwas kleiner, bei dem fliegenden Aufbau betrug die maximale Leistung bei 10 km/h ca. 12 μ W, die maximale Leistung bei der neuen Leiterplatte beträgt ca. 10 μ W. Jedoch ist bei einer Geschwindigkeit von 20 km/h die Leistung der neuen Leiterplatte besser und der MPP wurde ebenfalls verschoben, was ein grosser Vorteil ist, da die Einstellung des MPP-Ratio auf dem EM8500-Chip konfiguriert werden kann. Bisher konnte das MPP-Ratio nicht korrekt auf die Hardware eingestellt werden, da die Einstellung nur Werte zwischen 50 und 80 % akzeptiert.

b. Testen der Spule

Es wurde klar, dass die Energie, welche die Harvesterschaltung mit der aktuellen Spule von Premo mit einer Induktivität von 2.38 mH liefert zu klein ist und verbessert werden muss. Ideen wurden gesammelt und Herr Marcel Meli bemerkte, dass noch eine Spule von Premo mit einer Induktivität von 4.77 mH vorhanden wäre, diese sollte getestet werden. Die Spule hatte den exakt gleichen Aufbau wie die Spule, welche bis an hin verwendet wurde, nur die Induktivität war höher. Gemäss der Formel ? ist die Induktivität von der Wicklungszahl abhängig. Die Wicklungszahl wiederum beeinflusst die induzierte Spannung.

Tabelle 3.3: Leistung mit unterschiedlichen Spulen

Spule	maximale Leistung
Premo 2.38 mH	48.57 μW
Premo 4.77 mH	66.82 μW

Die Messung hat ergeben, dass die Energie bei einer Verdoppelung der Induktivität eine Leistungssteigerung von ca. 37 % zur Folge hat. Dies lässt sich auch mathematisch nachweisen:

$$L = \mu_0 \times \frac{N^2 \times A}{l} \quad (3.3)$$

$$N = \sqrt{\frac{L \times l}{\mu_0 \times A}} \quad (3.4)$$

$$\text{neue Wicklungszahl} = \sqrt{2} \times \text{bisherige Wicklungszahl} \quad (3.5)$$

Da die Leistung von der induzierten Spannung abhängt und die induzierte Spannung von der Wicklungszahl abhängt, steigt die Leistung in Abhängigkeit zur Wicklungszahl. Ein Zuwachs der Leistung von bis zu 41 % war zu erwarten. Es wurde entschieden, dass die stärkere Spule mit einer Induktivität von 4.77 mH verwendet werden soll.

c. Testen Magnete in Serie

Herr Dario Dündar machte uns auf ein Produkt von Reelight aufmerksam. Die Firma Reelight stellt Lichter für das Fahrrad her welche über Energy Harvesting betrieben werden, dafür werden neben der Wirbelstromvariante (siehe Kapitel theoretische Grundlagen) ebenfalls Produkte mit speziellen Magneten verwendet. Leider war kein Datenblatt für die Magnete mit der Bezeichnung RE-10200 vorhanden und auch auf Nachfrage wurde kein Datenblatt oder Ähnliches herausgegeben. Jedoch konnte man herausfinden, dass in dem Gehäuse drei Magnete verbaut sind, welche stärker sind als unsere Magnete. Unsere Magnete haben eine N42 Magnetisierung, was bedeutet, dass

sie eine Remanenz von 1.29 – 1.32 Tesla haben. Die Magnete, welche im Gehäuse von Reelight verbaut sind, haben eine N45 Magnetisierung, was bedeutet sie haben eine Remanenz von 1.37 – 1.42 Tesla. Der Durchmesser der Magnete beträgt 2.5 mm. Unsere bisher verwendeten Magnete wiesen einen Durchmesser von 20 mm auf.

Es wurde eine Leistungskennlinie mit dem Magneten von Reelight aufgenommen. Die maximale zur Verfügung stehende Leistung betrug $135.50 \mu\text{W}$, was einem Zuwachs von über 100 % entsprach.

Tabelle 3.4: Leistung mit unterschiedlicher Anzahl an Magneten

Konfiguration	maximale Leistung
Spule Premo 4.77 mH, 1 Magnet in Serie	$66.82 \mu\text{W}$
Spule Premo 4.77 mH, 3 Magnete in Serie (Reelight)	$135.50 \mu\text{W}$

Diese Messung brachte uns auf die Idee einen weiteren Versuch zu realisieren. Wir modifizierten den Testaufbau, damit zwei Magnete direkt hintereinander montiert werden konnten. Es sollte untersucht werden, ob die beiden Magnete die Spule besser anregten als nur ein Magnet. Die induzierten Spannungswerte unterschieden sich nicht drastisch, jedoch wurde die Spule länger angeregt. Die Abbildung 3.21 zeigt den Unterschied zwischen einer Spule, welche nur von einem Magneten angeregt wurde und einer Spule, welche von zwei direkt aufeinander folgenden Magneten angeregt wurde. Es ist zu sehen, dass die Spannung in der Spule, welche nur mit einem Magneten angeregt wurde, höher ist, jedoch wesentlich kürzer anliegt. Es wurde, trotzdem Potential für mehr Energie mit zwei Magneten, dagegen entschieden zwei Magnete in Serie einzusetzen, da man dieses Prinzip wieder ins Extreme treiben könnte und das ganze Rad mit Magneten ausstatten könnte. Sicherlich würde das jegliches Energieproblem unserer Arbeit lösen, jedoch wäre dies unpraktisch, wegen der verstärkten Bremswirkung und es wäre unästhetisch.

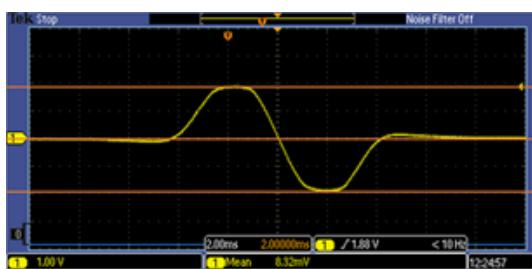


Abbildung 3.21: Anregung der Spule mit einem Magneten

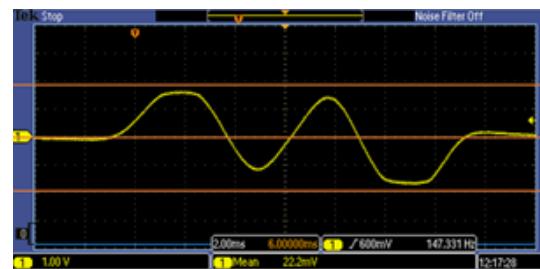


Abbildung 3.22: Anregung der Spule mit zwei Magneten in Serie

3.3.2 Ausmessen der Leistung endgültige Hardware

Gemäss dem vorangegangen Kapitel b. wurde ersichtlich, dass die stärkere Spule verwendet werden sollte. Wir haben uns jedoch gegen die Variante mit zwei Magneten in

Serie entschieden, trotzdem könnte dies später verwendet werden. Es musste erneut die Leistung, welche die Harvesterschaltung zur Verfügung stellt, aufgenommen werden.

Tabelle 3.5: Leistung des Prototypen

Geschwindigkeit	maximale Leistung
10 km/h	24.77 μ W
15 km/h	61.07 μ W
20 km/h	116.69 μ W
40 km/h	472.61 μ W

Die maximal verfügbare Leistung konnte durch die stärkere Spule erneut deutlich verbessert werden.

3.3.3 Ausmessen der Energieabgabe EM8500-Chip

Es war nun bekannt, wie viel Leistung die Harvesterschaltung zur Verfügung stellte, jedoch musste die Leistungsabgabe nach dem konfigurierten EM8500-Chip noch erfasst werden. Diese Angabe war sehr wichtig, um die Entwicklung der Firmware zu optimieren. Diese Werte gaben einen Rahmen, wie viel Energie das TI-SensorTag verbrauchen durfte.

Da es sich beim Ausgang des EM8500-Chips um eine Spannungsquelle handelt, welche bei uns auf 2 V konfiguriert wurde, konnte man die Energie, welche zur Verfügung stand, messen. Der Ausgang des EM8500-Chips wurde für diesen Zweck mit einem 10Ω Widerstand belastet. Anschliessend wurde gemessen, wie lange die Spannung am Ausgang aufrechterhalten werden konnte. Die Abbildung ?? zeigt den Spannungsverlauf über der Last. Da der Spannungsverlauf annähernd ein Rechteck war, konnte die Energie anhand der Zeit, welche der Ausgang aktiv war, berechnet werden.

$$E = \frac{U^2}{R} \times \Delta t \quad (3.6)$$

Tabelle 3.6: Leistungsabgabe Ausgang EM8500

Geschwindigkeit	abgegebene Energie
10 km/h	261.2 μ Ws
15 km/h	267.6 μ Ws
20 km/h	273.2 μ Ws
40 km/h	373.2 μ Ws

3.4 Energy Management

Das Ziel des Energy Managements ist es, dass die zur Verfügung stehende Energie bei 10 km/h genügt, um BLE-Pakete zu versenden. Die sekundäre Aufgabe ist es, dass sich

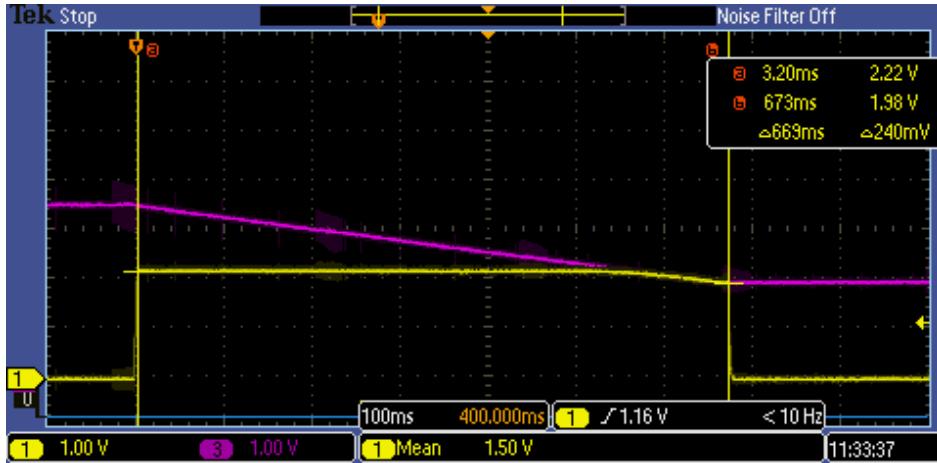


Abbildung 3.23: rot: VCC_STS, gelb: VSUP

der Long Time Storage lädt und entlädt. Ein Laden des LTS ohne dass die Energie im LTS verwertet werden kann ist Energieverschwendung. Die zwei gestellten Aufgaben sollen durch möglichst optimale Speichergrößen und intelligente Schwellwerte erreicht werden.

Damit man die Kondensatorenwerte berechnen kann, braucht es Energiedaten. Deshalb stehen als erstes im Unterkapitel 3.4.1 die Energie-Messergebnisse, danach folgt im zweiten Unterkapitel 3.4.2 die Dimensionierung der Kondensatoren und dann das Berechnen der Schwellwerte in Unterkapitel 3.4.3. Als letzter Punkt werden die Energiezustände innerhalb des Bicycle Computers definiert. Dies, weil die letzte offene Aufgabe der Optimierungsliste (3.1.3) das Senden nach 10 s dem Energiezustand angepasst werden soll.

3.4.1 Energiemessungen

Die Entwicklung ist konstant begleitet durch Energiemessungen. Sei dies durch Leistungsmessungen bei der Hardware (Harvester und EM8500-Chip) oder sei dies als Energiemessung der Software (TI-SensorTag). Für die Energiemessungen wird der Power Analyser von Agilent Electronic N6705B mit der Serienummer MY50000795 gebraucht. Dieses Messgerät misst gleichzeitig Strom, Spannung und Leistung im Zeitverlauf. Annäherungen aus den KO-Messungen sind nicht mehr notwendig.

In diesem Unterkapitel werden die Resultate den TI-SensorTag-Energiemessungen dargestellt. Mehr Messgraphiken und Erklärungen sind in der Datei ? abgelegt. Den Überblick über die Versionen ist in der untenstehenden Tabelle aufgelistet.

a. Überblick Energiemessungen

Um einen ersten Anhaltspunkt über den Energieverbrauch des TI-SensorTags zu erhalten, werden drei BLE-Pakete im Advertising Mode per Knopfdruck gesendet (siehe Abbildung 3.24). Dies entspricht der Messung zur TI-SensorTagversion V0. Nach der Weiterentwicklung des Programms zum Auslesen der Sensoren, folgte die Messung V1. In dieser Version wird die GPIO ausgelesen. Die GPIO ist für das Einlesen der Reed Switch-Impulse unumgänglich. Die Version V2, der Versuch mehr Sleep-Time einzuhaltende

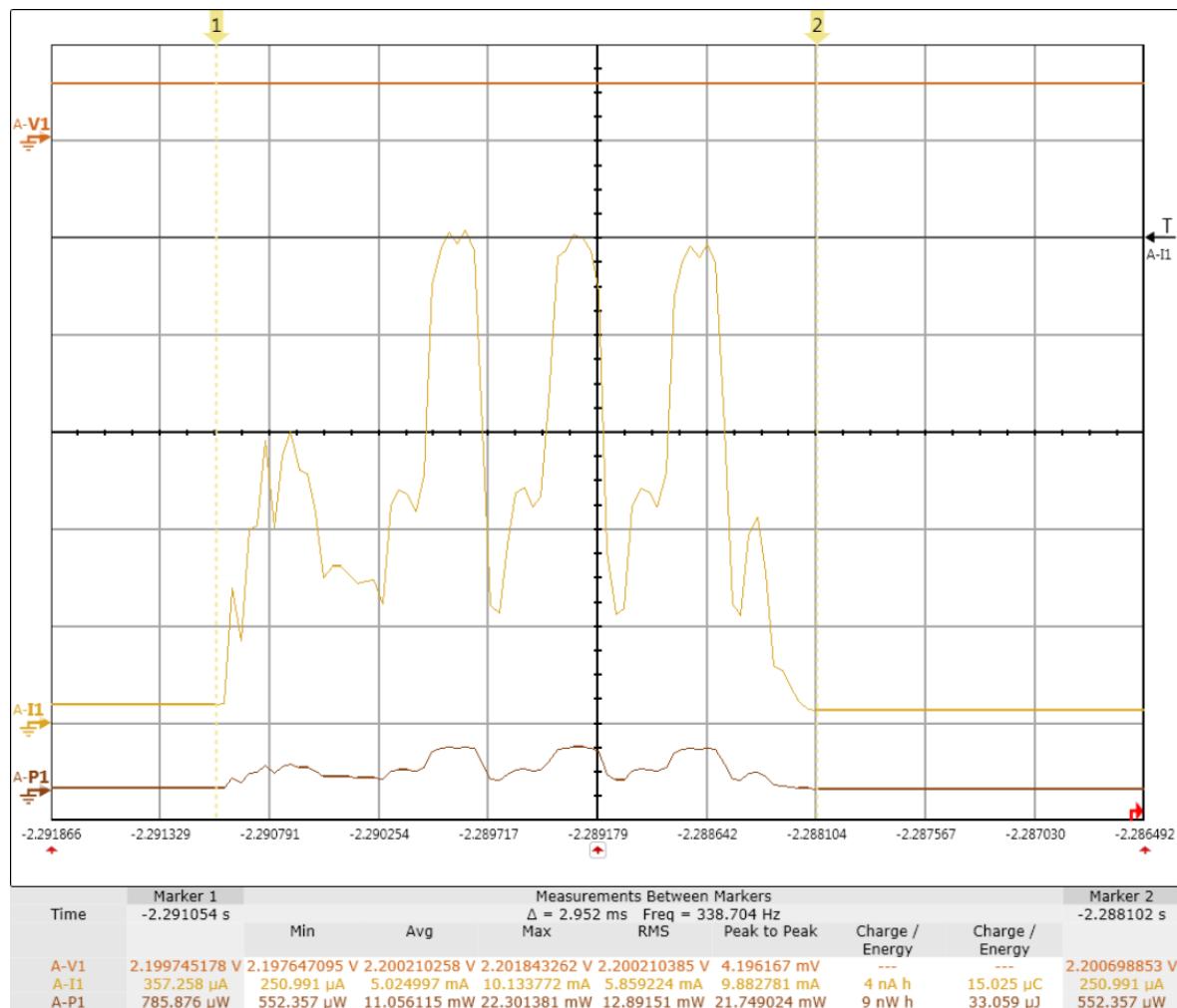


Abbildung 3.24: Minimalster Energieverbrauch: 3 BLE Pakete über Advertising Mode senden

bauen, scheitert am Zusammenspiel des Timings der BLE-Radio-Interrupts mit den GPIO-Interrupts. Durch das Neuaufsetzen des Programms entsteht die Version V3. Diese sendet power-optimiert Geschwindigkeitspaket.

Die untenstehende Tabelle stellt die Energieresultate dar. Unterschieden wird zwischen dem Energieverbrauch für die Initialisierung, kurz Init, und der Energie zum Senden von drei BLE-Paketen. Die Diskussion über Energieoptimierungen und die Deutung der Resultate finden sich in den Sitzungsprotokollen vom XXXXX - XXXX.

Tabelle 3.7: Messresultate nach TI-SensorTag-Versionen

Version	Datum	Aufgabe	Energie Init	Energie Senden
V0	10.3.16	Nur BLE Paket	unbekannt	33 μ J
V1	16.3.16	mit Geschwindigkeit	87 μ J	32 μ J
V3	22.4.16	Schlafmodus optimiert	40 μ J	29 μ J
V4	03.6.16	3 Sensore, kein Schlafmodus optimiert	77 μ J	43 μ J
V5	ongoing	3 Sensore mit optimiertem Schlafmodus	unbekannt	unbekannt

Bei den ersten drei Messungen werden die Sensoren noch nicht ausgelesen. Es zeigt sich, dass das Auslesen der Sensoren über I₂C und das Warten, bis dass die Sensoren gestartet sind, mehr Energie verbraucht als erwartet. Der Energieverbrauch für das Auslesens eines Sensors ist ohne Optimierungsmassnahmen so gross, dass kein Senden der Daten möglich ist. Die Speisung der Applikation bricht zusammen (siehe Abbildung 3.25).

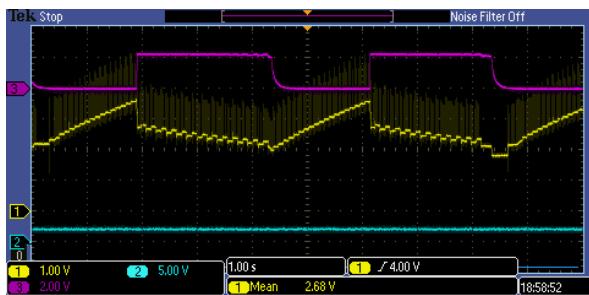


Abbildung 3.25: Auslesen der Sensoren reisst Energie zusammen

Aus diesen Grund wurde die Poweroptimierung innerhalb des Programms zur zentralen Herausforderung in dieser Arbeit. Details darüber sind im nächsten Unterkapitel (3.5 Power Management) zusammengefasst. Bei der Version V4 zeigt sich, dass die Funktion BLE-Send, zu der auch das Einlesen der Sensordaten gehört, ein Zuwachs von 50 % an Energiebedarf bewirkt.

b. Resultate Energieverbrauch TI-SensorTag Applikation V4

Als Überblick über den konkreten Energieverbrauch bei der Version V4 (siehe [xxx](#))
folgt exemplarisch ein Bild zur Initialisierung (Abbildung 3.29), eines zum Senden der drei BLE-Pakete (Abbildung 3.30) und dann je eine Abbildung zum Energieverbrauch beim Starten der drei Sensoren. Die Abbildung 3.26 zeigt den Überblick eines erstmaligen Sendens. Ganz links ist das einmalige Aufladen der Kondensatoren auf dem TI-SensorTag zu sehen. Dieser Vorgang ist in Abbildung 3.27 vergrössert dargestellt. Aus der Abbildung 3.26, die den Überblick des Sendens, kann man die durchschnittliche Zeit, nach der ein Refresh erfolgt ungefähr herauslesen. Im Durchschnittlich folgt ca. alle 0.1 s folgt ein Refresh-Peak. Auch dieser Vorgang ist als einzelne Energiemessung aufgenommen und in Abbildung ?? dokumentiert. Am Schluss dieses Unterkapitels

werden alle Energiewerte in einer Tabelle zusammengefasst.

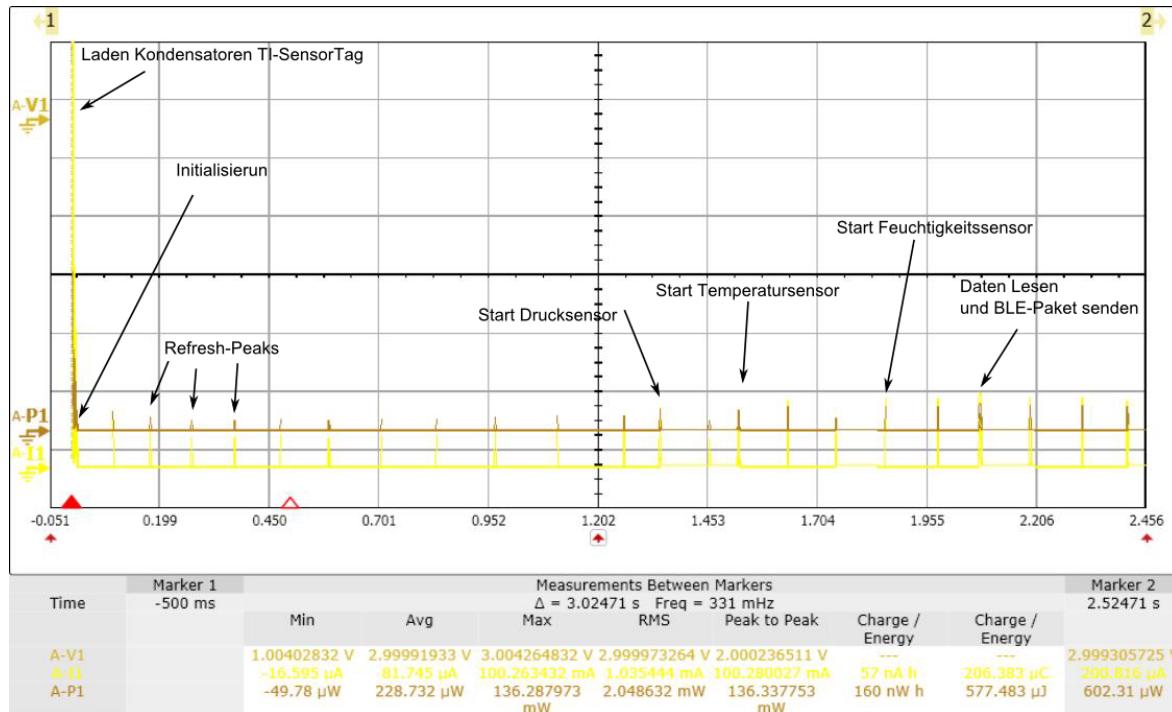


Abbildung 3.26: Überblick Energieverbrauch mit Refreshzyklen

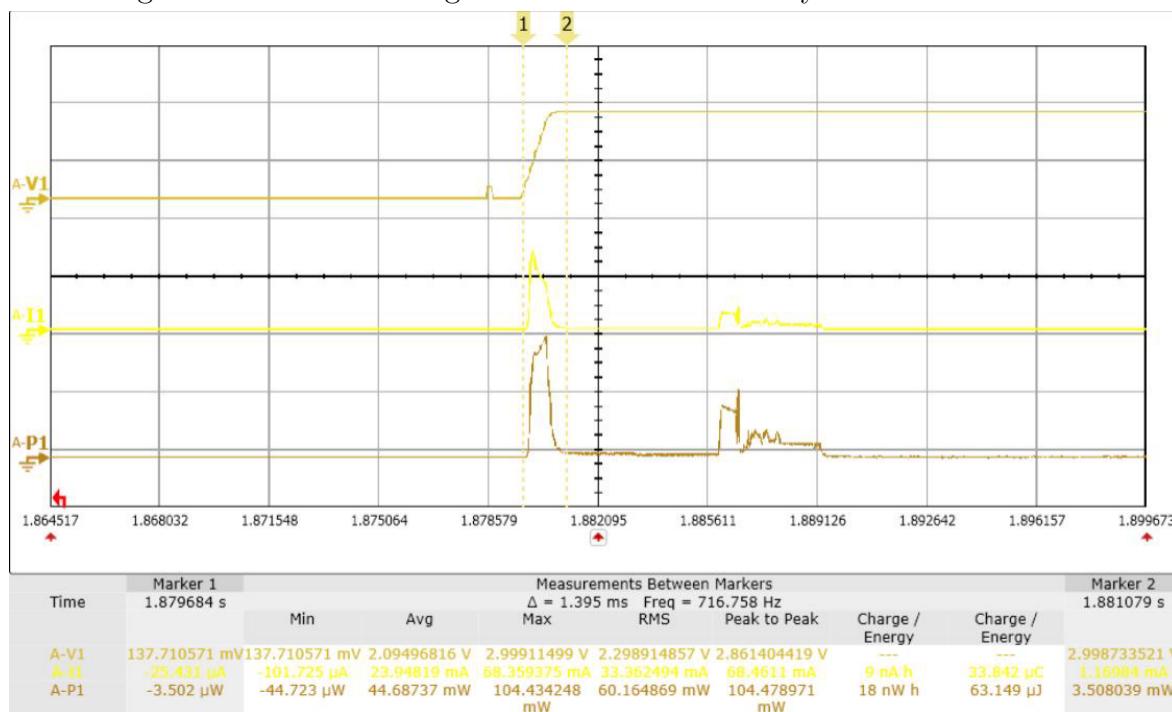


Abbildung 3.27: Einmaliges Aufladen der Kondensatoren TI-SensorTag

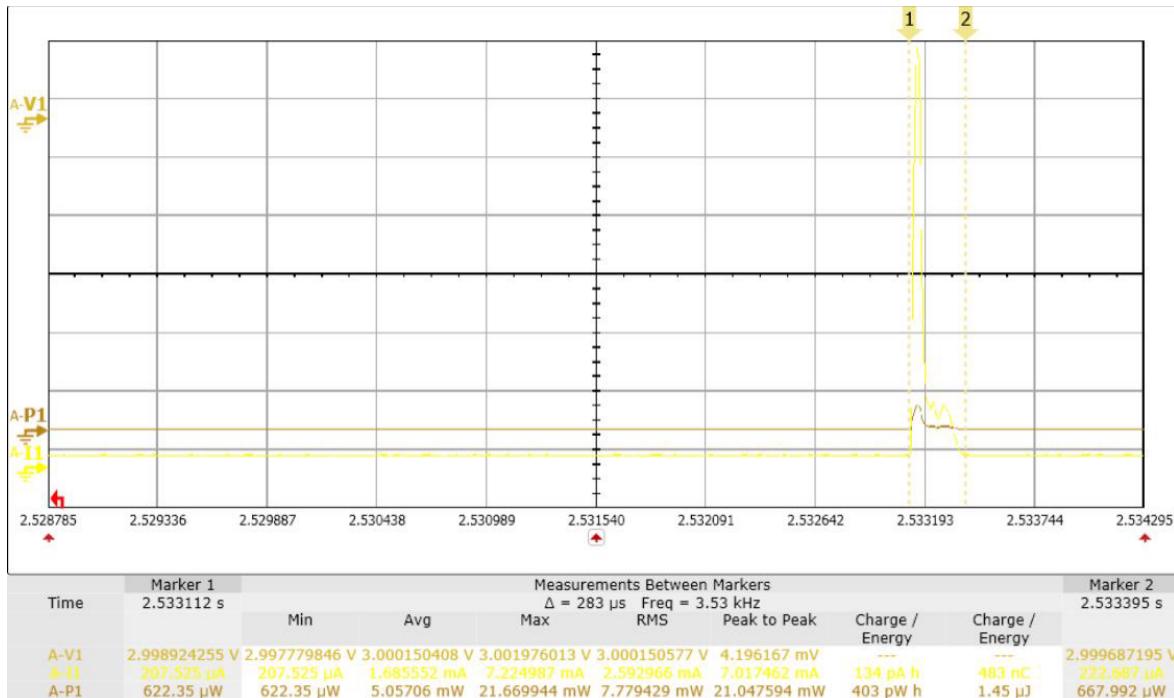


Abbildung 3.28: Energieverbrauch Refreshzykus

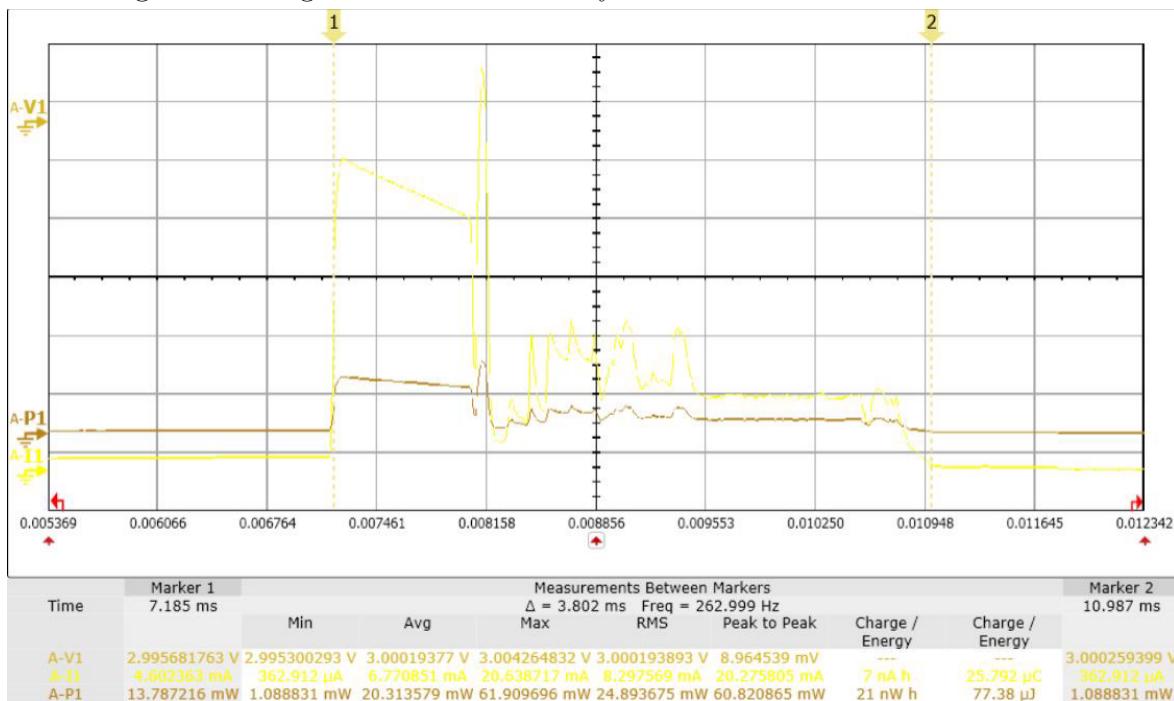


Abbildung 3.29: Energieverbrauch für Initialisierung

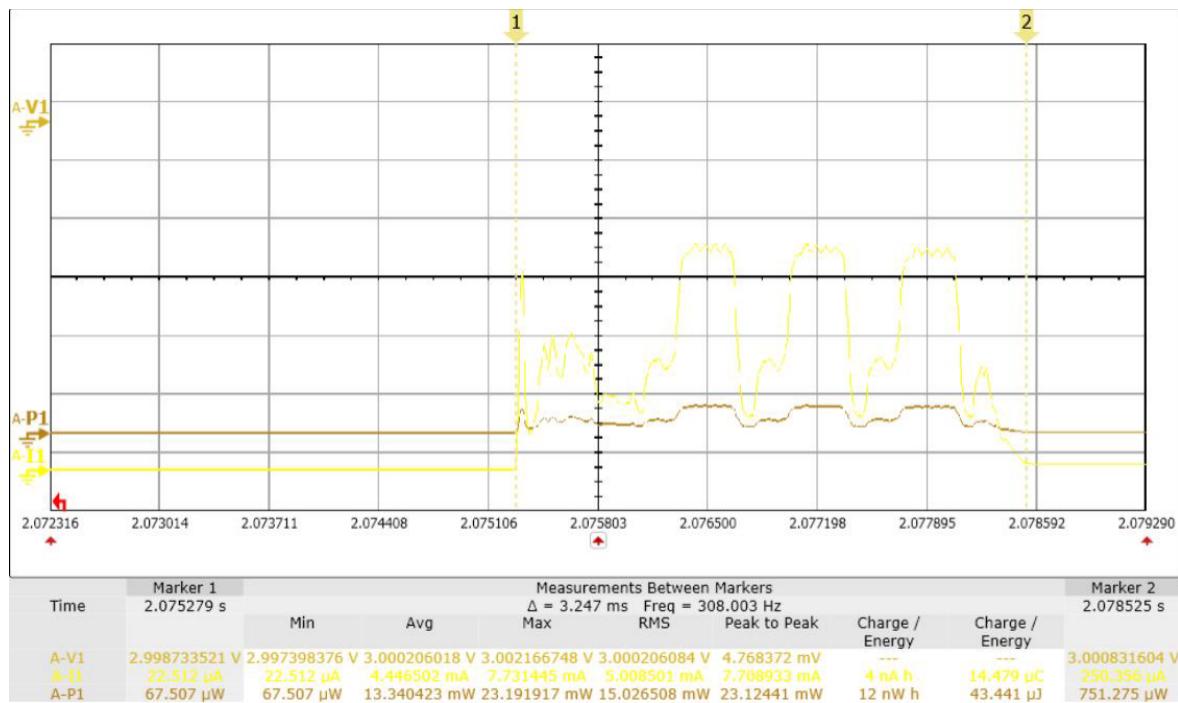


Abbildung 3.30: Energieverbrauch senden eines BLE-Paketes mit Sensordaten

Die Initialisierung ist vergrössert in der Abbildung 3.29 dargestellt. Im Überblick geht sie unter, da die Initialisierung direkt ans Laden der TI-SensorTag-Kondensatoren folgt. Da der Peak des Ladens viel höher ist, erscheinen die nachfolgenden Energieverbrauche nicht deutlich. Die Energiemessungen einzelner Stellen zeigen, dass für die Initialisierung $77.38 \mu\text{J}$, für das Refreshen ca. $1.4\mu\text{J}$ (Abbildung 3.28) und für das Senden des BLE-Pakets (Abbildung 3.30) $43.44 \mu\text{J}$ verbraucht werden. Das Senden des BLE-Paketes ist bei der Zeitmarke siehe Zeitmarke bei 2.07 s.

Energieverbrauch Starten des Drucksensors

Auf dem TI-SensorTag wird der Drucksensor BMP 208 von Bosch Sensortec verwendet. Bei der Zeitmarke 1.34 s wird der Sensor gestartet. Die Abbildung 3.31 dokumentiert einen Energieverbrauch von $5.753 \mu\text{J}$ zum Starten des Sensors.

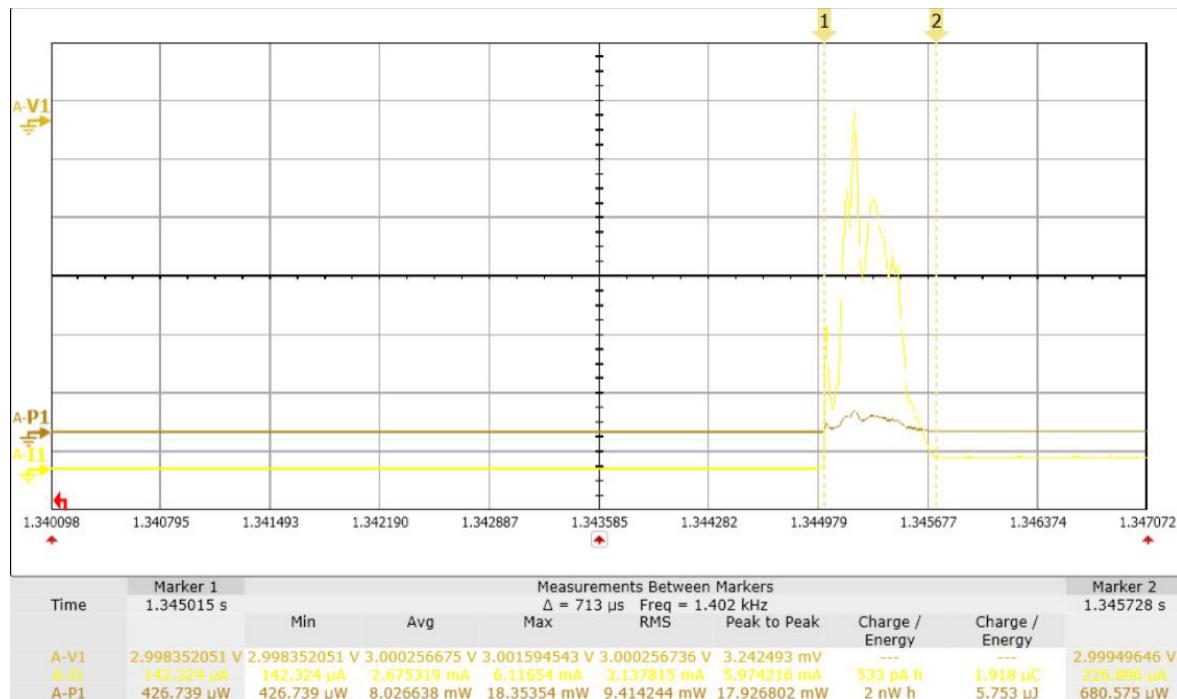


Abbildung 3.31: Energieverbrauch des Drucksensors BMP 280

Energieverbrauch beim Starten des Temperatursensors

Der Temperatursensor xxx der Marke yyy verbraucht für das Starten $10.635 \mu\text{J}$. Dies findet bei der Zeitmarke $t = 1.54 \text{ s}$ statt.

tempsensor
daten

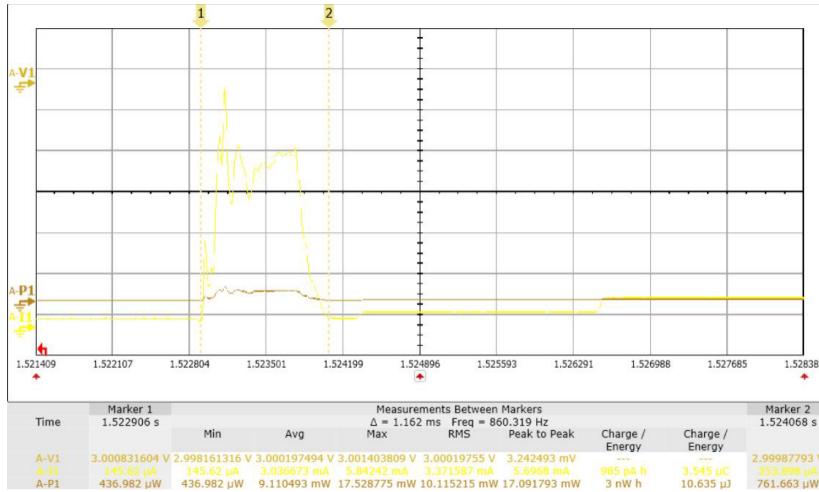


Abbildung 3.32: Energiemessung Temperatursensor auf TI-SensorTag

Energieverbrauch Feuchtigkeitssensor

Der Feuchtigkeitssensor xxx der Marke yyy verbraucht $6.358 \mu\text{J}$, siehe Abbildung 3.33.

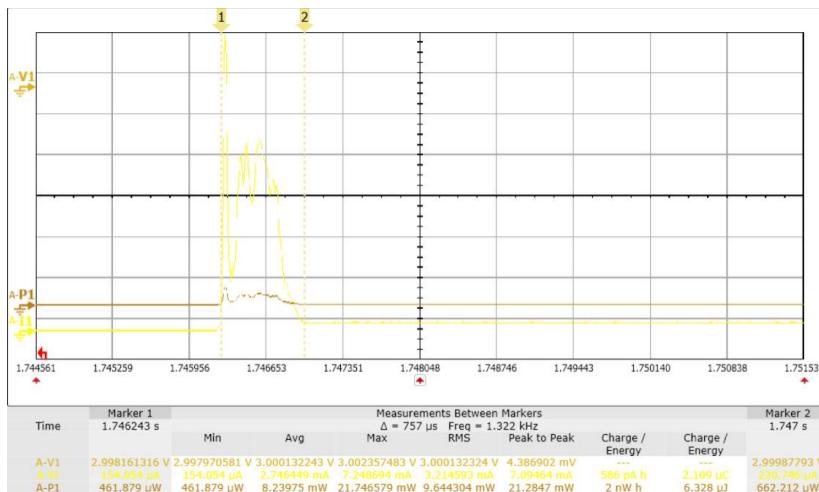


Abbildung 3.33: Energieverbrauch Feuchtigkeitssensor

Um die verschiedenen Energiewerte in Bezug zueinander zustellen, werden in der Tabelle 3.8 die Energiewerte zusammengesfasst und in der Abbildung 3.34 der Verbrauch aller Aktionen in Bezug zueinander graphisch dargestellt.

Tabelle 3.8: Zusammenfassung Energieverbrauch nach Aktion

Aktion	Energie
Laden C TI-Sensortag	63 μJ
Initialisierung TI-Sensortag	77 μJ
Refresh-Peaks während 10 s	14 μJ
Starten Drucksensor	6 μJ
Starten Temperatursensor	11 μJ
Starten Feuchtigkeitssensor	6 μJ
Senden BLE-Paket nur Geschwindigkeit	33 μJ
Senden BLE-Paket mit Sensordaten	43 μJ

fügen
ce in Ta-
e

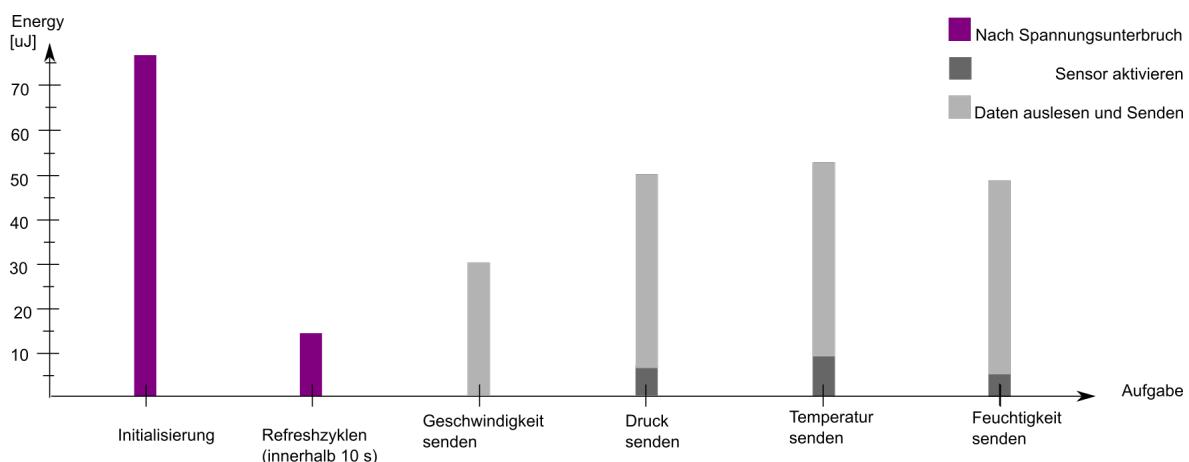


Abbildung 3.34: Energieverbrauch pro Aufgabe

Es wird deutlich, dass die Initialisierung und das notwendige Refreshen des Speichers zusammen (die Balken sind violett) fast gleich viel Energie brauchen, wie das Senden zweier BLE-Pakete mit Sensorwerten. Aus diesem Grund hat erste Priorität beim Power Management, dass die Speisung V_SUP nicht abstellt. Denn wenn bei der Datenverarbeitung zu viel Energie verbraucht wird und dadurch die Kondensatoren unter den minimalen Schwellwert fallen, stellt die Speisung ab. Jedes Mal wenn die Speisung unterbrochen wird, ist eine neue Initialisierung notwendig. Diese verbraucht unnötig viel Energie.

3.4.2 Energiekalkulation

Die Energiekalkulation dient der Bestückung des Kondensators STS und der Definition von Schwellwerten für die Register im EM8500-Chip. Es wird vom schlechtesten

Fall, also einer Geschwindigkeit von 10 km/h, und dem noch nicht optimalen Energieverbrauch des TI-Sensortags V4 ausgegangen. Laut Datenblatt des EM8500 (? , S. 9, Abschnitt Operating Conditions) soll STS zwischen 10 - 100 μF gross sein. Typisch ist 47 μF . Die Grundlagen zur Energiekalkulation sind im Theorieteil 2.2.4 abgebildet. Als oberer Schwellwerte beim STS wird 3 V angenommen. Die Berechnung dazu findet sich in der Gleichung 3.10.

Gleichungen
als Referenz
einbauen

$$\begin{aligned} E_{Applikation} &= E_{LadenC} + E_{Init} + E_{Refreshen} + E_{StartDrucksensor} + E_{SendenBLE} \\ &= 63 \mu\text{J} + 77 \mu\text{J} + 14 \mu\text{J} + 6 \mu\text{J} + 43 \mu\text{J} = 203 \mu\text{J} \end{aligned} \quad (3.7)$$

$$C_{STS} = \frac{2 \times 203 \mu\text{J}}{3^2 - (2.1)^2} \quad (3.8)$$

$$\begin{aligned} &= \frac{406 \mu\text{J}}{9 - 4.41} \\ &= \frac{406 \mu\text{J}}{4.59} \\ &= 88.45 \mu\text{F} \end{aligned} \quad (3.9)$$

Der Wert wird aufgerundet auf 100 μF , um genügend Energie zu haben.

MPP einstellen

Während der Entwicklung des Harvesters wurde die Leistungskurve aufgenommen (siehe Messprotokoll ?). Wie im Theorieteil erklärt 2.1.3 unterscheidet sich das Leistungsmaximum nach Geschwindigkeit. Weil das Ziel ein funktionstüchtiger Prototyp bei 10 km/h ist, bezieht sich das Einstellen auf dieses Leistungsmaximum. Das Leistungsmaximum liegt beim entwickelten Bicycle Computer bei 21.87 μW . Die MPP-Ratio liegt beim MPP bei 24.87 % (siehe). Die MPP-Ratio liegt somit unter 50 %. In der Umsetzung eines Energy Managements mit dem EM8500-Chip ergibt sich das Problem, dass nur Konfigurationen von MPPT-Ratios von 50 - 80 % erlaubt sind (siehe Tabelle unten). Zudem sind die Einstellungsschritte der tieferen MPP-Ratio von 50 % bis 67 % gröber, als die von 71 % bis 88 %. Dies, weil der EM8500-Chip für Harvester vom Typ TEG (ein konstantes MPP bei 50 %) und Solarzellen konzipiert ist. Bei der Bewegungsinduktion ist dieser vorgegebene Range nicht ideal.

Messprotokol
einträge

Werte
überprüfen,
aus welchem
Messprotoko
kommen die
Werte?

Referenz ein
bauen

Tabelle 3.9: MPPT-Ratio Einstellungen EM8500

Registerwert	MPPT-Ratio
0x00	50 %
0x01	60 %
0x02	67 %
0x03	71 %
0x04	75 %
0x05	78 %
0x06	80 %
0x07	82 %
0x08	83 %
0x09	85 %
0x0A	86 %
0x0B	87 %
0x0C	88 %

Registerwert	MPPT-Ratio
0x00	50 %
0x01	60 %
0x02	67 %
0x03	71 %
0x04	75 %
0x05	78 %
0x06	80 %
0x07	82 %
0x08	83 %
0x09	85 %
0x0A	86 %
0x0B	87 %
0x0C	88 %

ungsprotokoll
um

3.4.3 Einstellen der Schwellwerte

Das Konzept der Schwellwerte von VSTS und VLTS sind im Theorieteil bei der Erklärung des Speicherkonzepts des EM8500-Chips im Unterkapitel 2.2.1 beschrieben. Das Ziel ist, dass bei 10 km/h sich LTS entlädt. Die Berechnung der Schwellwerte stammt aus dem Theorieteil 2.2.4.

Aus der Vorgängerarbeit sind in der Tabelle unten notierte Konfigurationswerte überliefert. Bei der Inbetriebnahme fiel der hohe Schwellwert von v_bat_min_hi_dis auf (3.6 V), ab der erst die Applikation gespiesen wird. Der hohe Wert beruht auf dem Versuch, möglichst viel Energie vor dem Datensenden zu sammeln. Ein hoher Schnellwert verzögert die Zeit bis zum ersten Datensenden. Die Vorgänger wählten in ihren Einstellungen eine MPPT-Ratio von 88 %. Die Vermutung liegt nahe, dass keine Leistungskurve des Harvesters im Voraus aufgenommen wurde.

Tabelle 3.10: Konfiguration Vorgängermodell

Register	Wert
v_bat_max_hi	4.2 V
v_bat_max_lo	4.1 V
v_bat_min_hi_dis	3.6 V
v_bat_min_hi_con	2.1 V
v_bat_min_lo	1.8 V
v_appl_max_hi	3.8 V
v_appl_max_lo	3.7 V
mppt_ratio	88 %

Nach der Energiemessungen der Version V3 des TI-SensorTags (siehe a.), in dem die

Geschwindigkeit per BLE gesendet werden kann, entstanden folgende Schwellwerte:

Tabelle 3.11: Konfiguration aufgrund Geschwindigkeitsmessung

Register	Wert
v_bat_max_hi	4.8 V
v_bat_max_lo	4.7 V
v_bat_min_hi_dis	2.8 V (siehe Berechnungen unten)
v_bat_min_hi_con	2.1 V
v_bat_min_lo	2.0 V (vorgegeben von TI-SensorTag)
v_appl_max_hi	3.8 V
v_appl_max_lo	3.7 V
mppt_ratio	50 % (aus MPP-Kurve Harvester)

Die Grundlage bildete die Gleichung $E_{Applikation} = E_{STS_HIGH} - E_{STS_LOW}$, wobei $E_{Applikation}$ aus der Messung von V4 mit total 189 μJ für die Initialisierung und das Senden der Daten gebraucht werden. Als Kondensatorwert für STS wird 100 μF genommen. So ergibt sich folgender minimaler oberer Schwellwert:

$$(v_{bat_min_low})^2 = \frac{2 \times E_{Applikation}}{C_{STS}} + (V_{STS_LOW})^2 \quad (3.10)$$

$$= \frac{2 \times 189, \mu J}{100 \mu F} + (2.0)^2 \quad (3.11)$$

$$v_{bat_min_low} = \sqrt{\frac{378}{100} + 4} = \sqrt{7.8} = 2.79$$

Referenz Th
rie

Wichtigster Wert in der Konfiguration ist jedoch der korrekte MPPT-Wert. Er wird auf das Minimum (50 %) gesetzt. Das Maximum des Ladezustands wird auf den maximal erlaubten Wert gesetzt: v_bat_max = 4.8 V bzw. 4.7 V. Denn solang Energie geerntet werden kann, soll sie gespeichert werden. v_appl_max spielt in unserer Anwendung keine wichtige Rolle. Der Wert wird von den Vorgängern übernommen.

Das Zusammensetzen der Last (TI-SensorTag) mit der konfigurierten Hardware erstaunte (siehe Messprotokoll ? und Sitzungsprotokoll yyy). Ab 25 km/h funktionierte die Schaltung gut. Doch darunter reagiert der EM8500-Chip nicht. Weder STS noch LTS konnten sich laden. Nachmessungen am Eingang vom EM8500-Chip ergaben, dass bei 10 km/h und einer MPPT-Ratio von 50 % eine Spannung von 0.2 V anliegt. Diese ist zu tief, um den Booster zu starten. Erst bei höherer Geschwindigkeit wird eine Eingangsspannung von mehr als 0.3 V erreicht, und das System beginnt zu funktionieren. Um eine Minimalspannung von 0.3 V bei niedrigen Geschwindigkeit zu garantieren wird die MPPT-Ratio zukünftig auf den zweitiefsten Wert (60 %) eingestellt. Ein unerwartetes Verhalten des Drucksensors zeigt sich bei der Implementation des Auslesen. Der BLE-Sniffer empfängt die Druckdaten korrekt, doch beim Anschlussen des TI-SensorTags an die Harvesterschaltung, riss die Spannung sofort zusammen. Innert kürzester Zeit entlädt sich der STS sowie der parallel geschaltene LTS und un-

Messproto
und Sitzungs
protokoll
einfügen

terschreitet der STS den Schwellwert von V_SUP von 2 V, so stellt die Speisung der Applikation ab (siehe Abbildung 3.25 im Unterkapitel a.). Das Ausmessen des Energieverbrauchs bringt Klarheit: Die I2C-Kommunikation und das Aufstarten des Sensors brauchen xxx-mal mehr Energie als das Berechnen der Geschwindigkeit aus den Reed-Switch Impulsen (siehe Abbildung 4.15 im Resultat-Teil).

Durch den deutlich höheren Energieverbrauch wird eine Power-Optimierung im Softwareteil notwendig. Es zwingt aber auch dazu, den Schwellwert von v_bat_min_hi_dis wieder zu erhöhen. Die finale Konfiguration ist im Resultate-Teil abgebildet.

3.4.4 Energiezustand des Systems

Die letzte der vier Aufgaben aus der Optimierungsliste nach der Inbetriebnahme (siehe Unterkapitel 3.1.3) ist das variable Anpassen des BLE-Paketsendens aufgrund des Energiezustandes. Im ersten Unterkapitel 3.4.4 wird das Einteilen des Systems in Energiezustände beschrieben.

Definition von Energiezuständen

Da die produzierte Energie von der Fahrgeschwindigkeit abhängt, werden die Energiezustände aufgrund der Fahrgeschwindigkeit definiert. Welche Aufgaben in welchem Energiezustand getan werden, folgt in den Abbildungen 3.35, 3.36 und 3.37, direkt nach der Definition der Zustände.

Tabelle 3.12: Energiezustände aufgrund der Geschwindigkeit

Geschwindigkeit	Systemzustand
0 - 15 km/h	LOW_ENERGY
15 - 20 km/h	MIDDLE_ENERGY
> als 20 km/h	HIGH_ENERGY

Bei wenig Energie lädt sich nur der STS. Erreicht er den Schwellwert für die Speisung der Applikation, so erhält TI-SensorTag Spannung. Die Reed-Impulse werden ausgelesen, der zweite Prozessor für das Senden der BLE-Pakete gestartet und die Daten übertragen. Bei wenig Energie verbrauchen diese drei Aktionen alle zur Verfügung stehende Energie. Denn LTS hat sich nicht geladen. Die Energie genügt nicht für weitere Aktionen.

Bei etwas mehr Energie befindet sich das System in einem Zwischenzustand (Abbildung 3.36). Zu jeder Geschwindigkeitsmessung wird ein Sensor ausgelesen. Im Idealfall unterstützt LTS die Stabilität. Wenn nicht, schaltet V_SUP bei zu wenig Energie ab. Im hohen Energiezustand wird mit jedem Geschwindigkeitspaket die aktualisierten Werte der drei Sensoren mitübertragen.

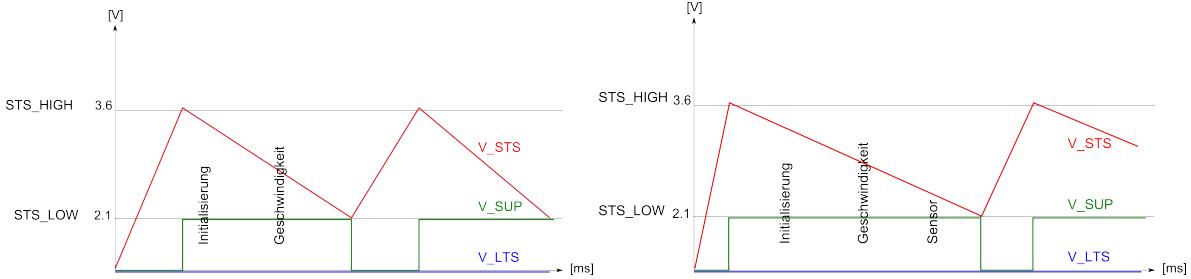


Abbildung 3.35: Wenig Energie

Abbildung 3.36: Mittlere Energie

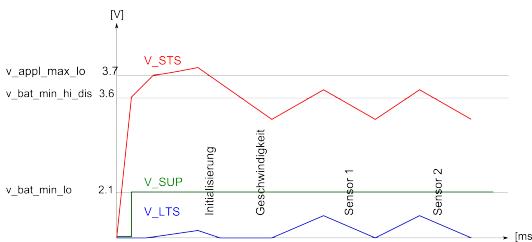


Abbildung 3.37: Viel Energie

3.5 Power Management

Das Power Management wird mit dem TI-SensorTag umgesetzt. Im ersten Unterkapitel 3.5.1 wird beschrieben, auf welche Bibliotheken und Hilfen man von Texas Instruments zurückgreifen kann. In nächsten Unterkapitel 3.5.2 werden die Power-Domains des TI-SensorTags erklärt und im letzten Unterkapitel die Umsetzung, wie Schlafenszeiten eingebaut werden 3.5.3.

3.5.1 Programmieren für Low Power Applikation im Advertising Mode

Das ausgewählte TI-SensorTag ist für Low-Power-Anwendungen ausgelegt. Die Low-Power-Programmbeispiele von TI basieren auf dem TI-Real Time Operating System (kurz RTOS). Da bei 10 km/h Energie im μ J-Bereich zur Verfügung steht, läuft die Applikation im Advertising Mode. Dieser braucht weniger Energie als der Connected Mode und der Overhead eines Betriebssystems (RTOS) und der Initialisierung des BLE-Stacks sind nicht notwendig. Der Nachteil dieser schlanken Programmierung ist, dass die TI-SensorTag Beispiele ausschliesslich auf RTOS aufbauen, und somit nicht 1:1 übernommen werden können. Für nachkommende Entwicklerinnen und Entwickler sind zwei Vorteile und zwei Nachteile dieser Wahl festgehalten:

1. Vorteil ohne RTOS: Ein sehr schlanker Code entsteht. Die Register werden direkt gesetzt, und die Funktionsaufrufe sind sehr schnell.
2. Vorteil ohne RTOS: Man muss sich nicht in das TI-RTOS-Konzept einarbeiten. Die Message-Queue und das Thread-Handling bei TI muss man nicht kennen.
3. Nachteil ohne RTOS: TI implementierte über die Library die Hauptaktivitäten wie Power Domains einschalten oder Kommunikation zu Peripheriegeräten aufbauen. Die Zugriffe werden vom OS übernommen. Ohne RTOS muss man sich selber z.B. ums Aufwachen oder das Abhandeln der Interrupts kümmern.
4. Eine Dokumentation zur umfangreichen Library (cc26xxware/driverLib) fehlt, da TI davon ausgeht, dass man über das OS auf die Bibliothek zugreift. Die internen Abhängigkeiten zu verstehen, ist nicht einfach. Unerwünschte Nebeneffekte treten auf.

3.5.2 Power Domains beim TI-SensorTag

Bei einer Low Power Applikation stellt das Ein- und Ausschalten von sogenannten Power Domains eine zentrale Rolle. Das TI-SensorTag unterscheidet 10 Power Domains. Ein Überblick ist in der Abbildung 3.38 abgebildet. Das Konzept der Verbrauchsoptimierung besteht im Abschalten unnötigen Peripherie- und Prozessorteilen. Nur absolut notwendigen Power Domains laufen weiter. Der Teil, der nicht abstellt wird AON (Always On) genannt. Zu dieser Domain gehört ein Clock, ein Timer, gewisse Interrupts und Schnittstellen. Im Anhang D ist in der Abbildung D.1 detailliert dargestellt, wie die einzelnen Power Domains sich beim Ausschalten verhalten. So braucht z.B. das RAM und die GPIO Refreshzyklen. Beim RAM bedeutet dies eine Speicherauffrischung, bei der GPIO eine Kontrolle der Pinzustände.

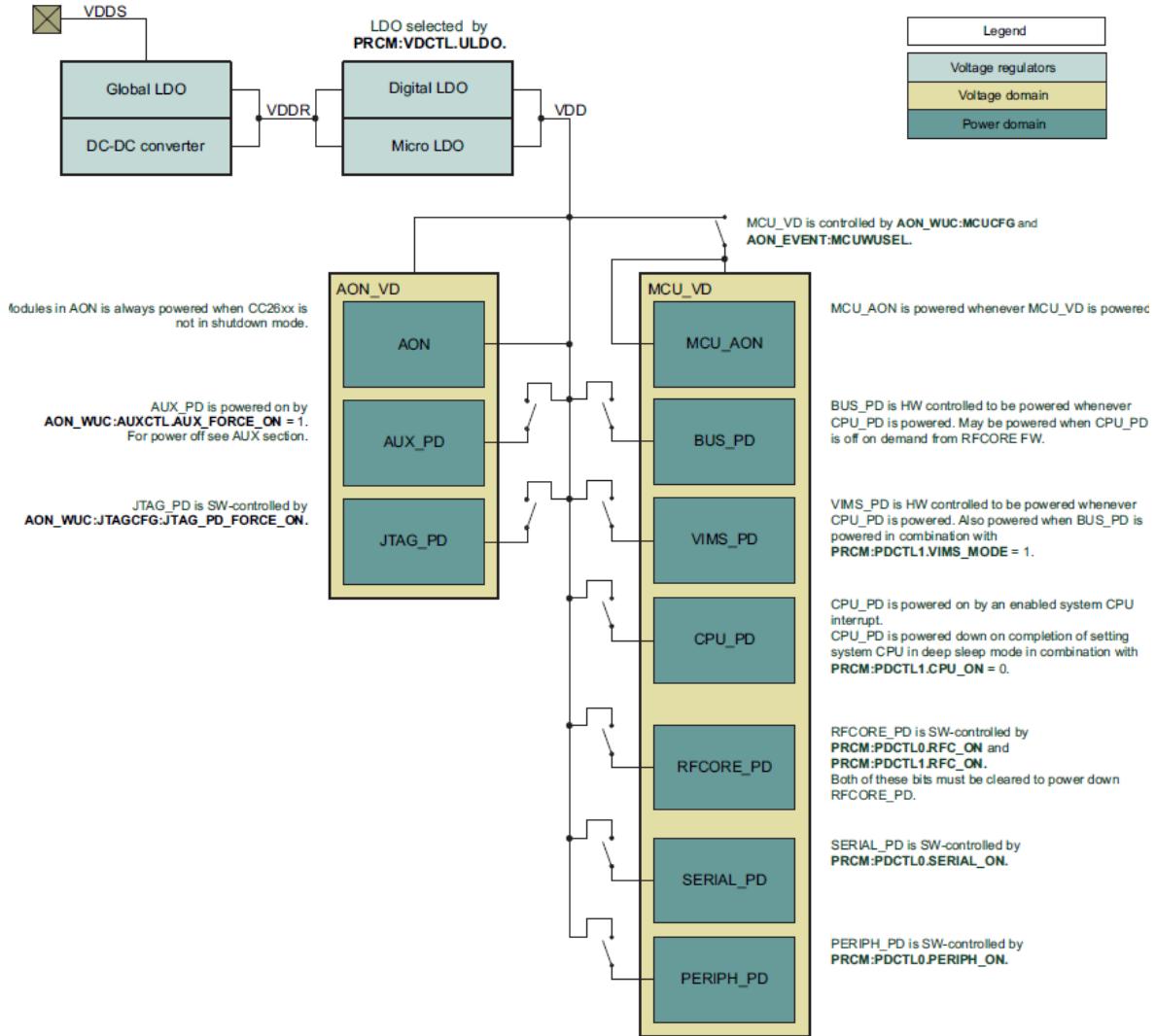


Abbildung 3.38: Supply System TI-SensorTag aus: ?, S. 416

3.5.3 Schlafenszeiten implementieren

In der Theorie wird das Schlafen zwischen Funktionsblöcken beschrieben 2.3.1. Der Energy Managements-Teil fasst zusammen, welche Aktionen bei welcher Energie gemacht werden können (siehe Abbildungen 3.39). Das Auslesen der Sensoren ist nur möglich, weil die Aktionsblöcke aufgeteilt werden und das System dazwischen schlafen geht. Die Abbildung 3.39 zeigt der implementierte Codeablauf der TI-SensorTag Version V4, bei einer Geschwindigkeit unter 50 km/h . Der Fall von einer Geschwindigkeit von über 50 km/h, bei der V_SUP nicht mehr abstellt, ist im letzten Unterkapitel illustriert.

Ist eine Aufgabe getätig, wie z. B. das Starten eines Sensors, kann das System schlafen gehen. Dieses Prozedere funktioniert einwandfrei und sicher. Das System wacht aufgrund eines Interrupts wieder auf. Nachfolgend sind die zwei Hauptfunktionen: Vorbereiten zum Standby und ausschalten des Hauptprozessor abgebildet.

referenz
überprüfen

Geschwindig
überprüfen

Geschwindig
überprüfen

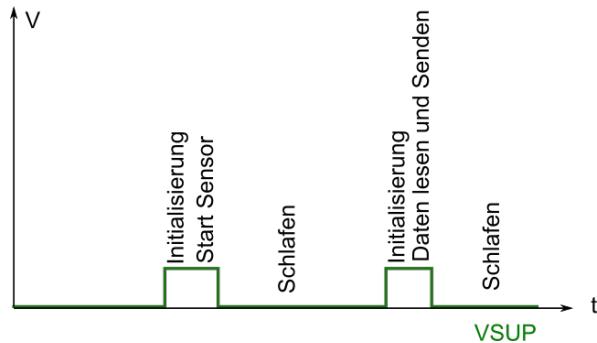


Abbildung 3.39: Schlafenszeiten zwischen Aktionen

a. Standby-Prozedur

Die abgebildete Funktion verdeutlicht exemplarisch, welche Schritte bei einer Programmierung ohne RTOS durch die Programmiererin bzw. den Programmierer ausgeführt werden. Zum Verständnis werden einige Zeilen des Codes kurz kommentiert:

- Zeile 1: Der Christal-Oszillator wird abgeschaltet,
- Zeile 2: Das Radio-Modul ebenfalls,
- Zeile 3: Ein niedriger Takt wird eingestellt und
- Zeile 4: unnötige Periphariespeisung abgestellt.
- Zeile 5: Dem Prozessor wird erlaubt, seine notwendigen Register zu refreshen
- und
- Zeile 6: das Cache wird abgeschaltet.
- Zeile 7: Wobei ein Refreshzyklus aktiviert wird.
- Zeile 11: Das Intervall zwischen den Refreshzyklen wird berechnet
- Zeile 12: Es wird gewartet, bis alle Anweisungen abgeschlossen sind

Zeile ein-
nen

```

1 void go_to_standby(){
2     powerDisableXtal();
3     powerDisableRFC();
4     OSCHfSourceSwitch(); // activates lower clock
5     powerDisableAuxForceOn();
6     powerEnableMcuPdReq();
7     powerDisableCache();
8     powerDisableCacheRetention();
9
10    // calculate next recharge time
11    SysCtrlSetRechargeBeforePowerDown(XOSC_IN_HIGH_POWER_MODE);
12    SysCtrlAonSync();
13 }
```

b. Abschalt-Prozedur

Nach der Vorbereitung zum Abschalten wird der Prozessor zum Schlafen geschickt.

Zeile 1: Cortex M3 wird zum Schlafen geschickt
 Zeile 2: Das Power-Controll-Register erhält das Schlafens-Flag
 Zeile 3: Die Always-On-Power-Domain muss die Konfigurationen speichern
 Zeile 4: Den Refrehzyklus-Wert setzen
 Zeile 5: Warten, dass alle Prozesse fertig abgehandelt sind.

```

1 void go_to_deep_sleep(){
2     powerDisableCPU();
3     PRCMDDeepSleep();
4     SysCtrlAonUpdate();
5     SysCtrlAdjustRechargeAfterPowerDown();
6     SysCtrlAonSync();
7 }
```

c. Sleep Time-Übersicht bei Version V4

Bei den ersten Messungen von Sensoren riss V_SUP so schnell zusammen, dass kein BLE-Paket gesendet wird. Das Senden wird möglich, indem zwischen einzelnen Aufgaben geschlafen wird. Denn während dem das TI-SensorTag nur die minimale Energie verbraucht, kann der Bicycle Computer Energie sammeln. Während der Schlafenszeit steigt der Spannungswert an den Kondensatoren STS und LTS. Die Abbildung 3.40 zeigt, dass bei der Version V4 (siehe xxxx die Aufgaben aufteilt werden. Dies entspricht dem Konzept des vorangehenden Unterkapitels mit der Abbildung 3.39, nur dass in Abbildung 3.40 der Fall von einer Geschwindigkeit von über 40 km/h gezeigt wird, und V_SUP deshalb nicht abstellt.

Link

Link zu SensorTag versiegen einbaute

Wert prüfen

Beim Senden von Sensordaten sind zwei Funktionen kritisch: das Starten eines Sensors und das Senden der Daten. Diese zwei Aufgaben werden getrennt und dazwischen geht das System schlafen (siehe vereinfacht dargestellt in der Abbildung 3.40).

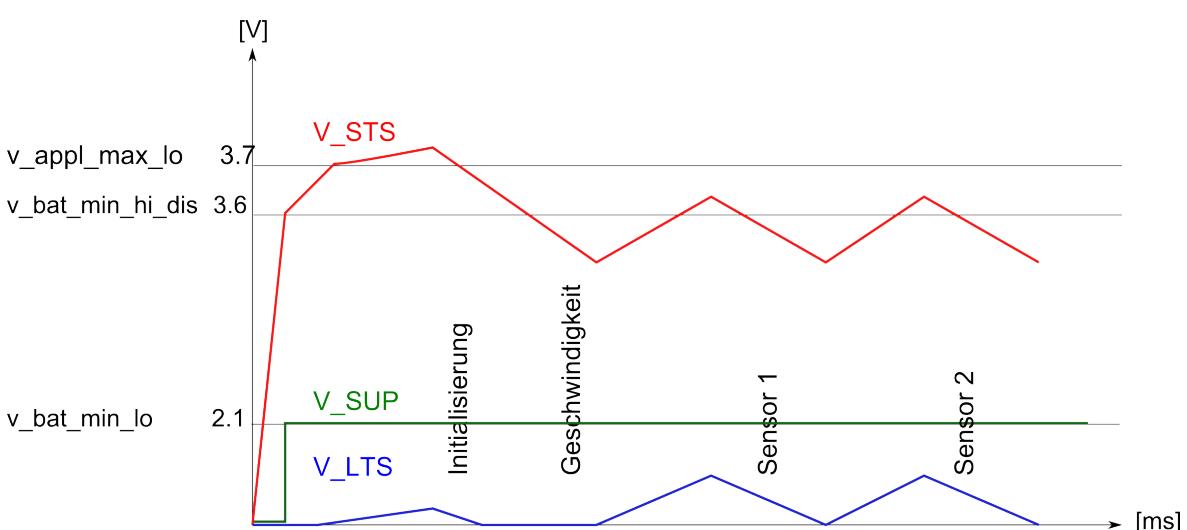


Abbildung 3.40: Auf trennen der Arbeit durch Schlafen dazwischen

Die Schlafenszeit wird experimentell ermittelt. Innerhalb des GPIO-Handlers zählt ein Counter die Reed Switch-Interrupts, also die Radumdrehungen. Da am Fahrrad zwei Magnete befestigt sind, entspricht der Counterwert der doppelten Anzahl Radumdrehungen. Ist das Zählermaximum erreicht, wird der Code rund um eine energiestarke Aktion ausgeführt. Danach geht das System schlafen. Das System soll so lang schlafen, dass die Spannung von V_STS genug gross ist, dass die energiestarke Aktion ausgeführt werden kann. In der unten aufgeführten Tabelle findet sich eine grobe Count-Statistik der Version V4 für verschiedene Geschwindigkeiten.

Tabelle 3.13: Zählerstände für Schlafenszeit

Geschwindigkeit	Counter-Wert	Ladezeit STS	Radumdrehungen
13 - 16 km/h	100	80 s	50
20 km/h	50	17 s	25
31 - 32 km/h	50	11 s	25

Das Setzen der Schlafenszeit ist in einer globalen Variable gespeichert. Abhängig von der Geschwindigkeit, wird das Schlafensintervall (das Counter-Maximum) angepasst.

3.6 Applikationsentwicklung

Nachfolgend wird der Aufbau von TI SensorTag zusammen mit unserem selbst entwickelten Board als Sensor benannt und die Android Applikation auf dem Smartphone wird einfach als Applikation benannt.

3.6.1 Aufbau der Applikation

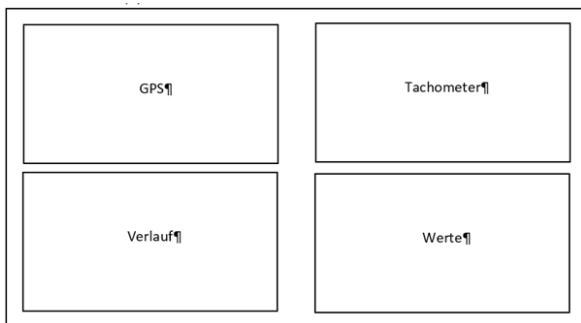


Abbildung 3.41: Aufteilung des Bildschirms

Der Aufbau der Applikation sollte am Anfang in vier Teile des Bildschirms geteilt werden (siehe Abbildung 3.41). Im Bereich GPS soll eine Karte angezeigt werden. Außerdem soll es möglich sein die Route aufzuzeichnen und abzuspeichern. Der Verlauf zeigt die Entwicklung der Geschwindigkeit über einen gewissen Zeitraum, wobei es wählbar sein soll welche Daten angezeigt werden. Die Geschwindigkeit soll anschaulich

in einem Tachometer angezeigt werden und die Tachonadel soll im besten Fall animiert sein. Im Bereich der Werte sollen die aktuellen Werte in Zahlen und den entsprechenden Einheiten dargestellt werden. Schnell wurde klar, dass dieser Aufbau wenig Sinn macht, da die Bereiche auf einem Smartphone viel zu klein wären und nicht mehr leserlich sein würden. Darum wurden die Teile des Bildschirms als einzelne Activities geplant, somit wäre der ganze Bildschirm als Anzeige für jede einzelne Aufgabe vorhanden.

a. Home-Screen

Als erstes soll der Benutzer den sogenannten Home-Screen sehen, hier werden der Tachometer, die Werte und Buttons zur Bedienung angezeigt. Je nach Anzahl der Funktionen sollen neue Buttons implementiert werden. So soll beispielsweise für die Sensorwahl ein Button vorhanden sein. Während der Entwicklung der Applikation wurde die Idee von einer GPS-Karte und einem Verlauf verworfen, da diese Funktionen aus zeitlichen Gründen nicht mehr realisierbar waren.

b. Sensorwahl

Schnell wurde klar, dass die Auswahl eines spezifischen Sensors wichtig werden würde, für den Fall, dass einmal mehrere Sensoren vorhanden wären. Ein Beispiel: Würde unsere Applikation bei einem Fahrradrennen eingesetzt werden, so müsste die Applikation auf dem Smartphone mit dem Sensor am Fahrrad gepaart werden. Ansonsten würde man die Daten von allen Sensoren empfangen und die Anzeige würde nicht die Daten des eigenen Sensors anzeigen.

c. Einheiten und Einstellungen

Es kam ebenfalls der Wunsch auf, die Einheiten der Werte einstellbar zu machen, da die Applikation bestenfalls auch in anderen Ländern verwendet werden soll. Also wäre es optimal, wenn die Geschwindigkeit beispielsweise auch in Miles per Hour dargestellt werden könnte. Es müsste eine Einstellungsmöglichkeit bereitgestellt werden, damit die Temperatur kalibriert werden kann.

d. BLE-Kommunikation

Der wichtigste Teil der Applikation ist die Bluetooth-Kommunikation. Hier werden die Daten vom Sensor empfangen. Die Daten müssen nach dem Empfangen gefiltert werden, damit sichergestellt wird, dass die empfangenen Daten von einem unserer Sensoren kommen. Sobald sichergestellt ist, dass die empfangenen Daten zu unserem Sensor gehört, müssen diese umgerechnet werden.

e. Inbetriebnahme des Codes der PA

Als erster Schritt wurde der Code der vorangegangenen Arbeit analysiert und die wichtigen Teile zur Bluetoothkommunikation übernommen. Es wurden folgende Funktionen aus dem Code übernommen: onStart, onActivityResult, scanLeDevice, BluetoothAdapter.LeScanCallback.

Die Funktion onStart wurde in BLE_init umbenannt und der Mechanismus zum Wachhalten des Geräts wurde entfernt. Der Mechanismus wurde entfernt, da dieser nicht mehr funktionierte. Ebenfalls wurde die Funktion in die Funktion onCreate eingefügt, um den Code übersichtlicher zu gestalten und da diese Funktion nur einmalig aufgerufen werden musste.

Die Methoden onActivityResult und scanLeDevice wurden komplett übernommen. Die Callback-Funktion des Bluetoothadapters BluetoothAdapter.LeScanCallback wurde strukturell übernommen, jedoch wurde die Codierung zur Verarbeitung der Daten entfernt.

f. Filter einbauen

Der nächste logische Schritt war, dass die empfangenen Daten gefiltert werden mussten. Dafür wurde folgende Struktur definiert, damit festgestellt werden kann, ob die Daten von einem unserer Sensoren versendet wurden.

elle zu lang.
zen

Tabelle 3.14: Filter

Length	Type	UUID	Package ID	Speed	Pressure	Temperature	Humidity	Checksum
23	0x03	0xDEBA2 bytes		4 bytes	4 bytes	4 bytes	2 bytes	

Aufgrund dieser Struktur können alle empfangenen Daten, welche nicht die definierte Länge, Typ und UUID besitzen, ignoriert werden.

Wenn sichergestellt ist, dass die Daten von unserem Sensor stammen, musste noch die Adresse vom Sender gespeichert werden. Alle Adressen werden in einer Liste gespeichert, vor dem Speichern wird jedoch noch geprüft, ob die Adresse bereits in der Liste ist, falls dies so ist wird die Adresse nicht noch einmal gespeichert.

g. Berechnung der Daten

Der Sensor sendet Werte wie den Zeitunterschied zwischen zwei Magnetdurchläufen, den Luftdruck, die Temperatur und die relative Luftfeuchtigkeit. Diese Daten müssen erst noch in die korrekten Werte umgerechnet werden, da alle Werte in mehreren Bytes verteilt vorliegen.

Der Zeitunterschied zwischen zwei Magnetdurchläufen wird in vier Bytes dargestellt. Die ersten beiden Bytes enthalten die Anzahl Sekunden. Folgende Formel zeigt die Umrechnung von zwei Bytewerten in eine reelle Zahl:

$$t = ((byte_0 << 8) + byte_1) \quad (3.12)$$

Die erhaltene Zahl muss noch mit einem Typecast in den Datentyp float umgewandelt werden, um die weiteren Berechnungen zu erleichtern.

Das dritte und vierte Byte des übermittelten Zeitunterschieds enthält die Sekundenbruchteile. Folgende Formel zeigt, wie die Bytes verrechnet werden müssen, um die korrekten Werte zu erhalten:

$$t = \frac{((byte_2 << 8) + byte_3)}{2^{16}} \quad (3.13)$$

Der erhaltene Sekundenbruchteil kann nun zu den ganzen Sekunden dazu addiert werden, um die ganze Zeit zwischen zwei Magnetdurchläufen zu erhalten.

Ein Beispiel: Wird also der hexadezimale Wert 0x00041000 empfangen sind das 1.25 Sekunden zwischen zwei Magnetdurchläufen. Die genaue Berechnung des Beispiels kann in den nachfolgenden Formeln verfolgt werden:

$$t_{ganzeSekunden} = ((0x01 << 8) + 0x00) = 1s \quad (3.14)$$

$$t_{Sekundenbruchteile} = \frac{((0x40 << 8) + 0x00)}{2^{16}} = 0.25s \quad (3.15)$$

$$t_{Total} = t_{ganzeSekunden} + t_{Sekundenbruchteile} = 1s + 0.25s = 1.25s \quad (3.16)$$

Um die Geschwindigkeit zu erhalten muss der Radumfang durch die erhaltene Zeit geteilt werden. Jedoch befinden sich an unserem Rad zwei Magnete, deswegen darf nur der halbe Radumfang verrechnet werden. Ausserdem ist das Resultat in m/s und muss deshalb noch in km/h umgerechnet werden, was nichts anderes als eine Multiplikation mit dem Faktor 3.6 ist.

Der Druck wird in Pascal (Pa) übertragen, hier müssen die verschiedenen Bytes miteinander addiert werden und in hPa umgerechnet werden. Es werden jedoch nur die letzten drei Bytes mit Werten befüllt sein, da das TI-SensorTag nur drei Bytes aus dem Drucksensor erhält. Das erste Byte kann ignoriert werden. Trotzdem wurden vier Byte übertragen, um später die Daten zum Druck mit weiteren Informationen zu bestücken. Folgende Formel zeigt die Umrechnung der Bytes in eine float Zahl:

$$p = (float)((byte_2 << 16) + (byte_1 << 8) + byte_0) \times 0.01 \quad (3.17)$$

Die barometrische Höhenformel (? , Seite 284) zeigt die Abhängigkeit des Druckes von der aktuellen Höhe.

$$p(h) = p_0 \times e^{-\frac{h}{7991m}} \quad (3.18)$$

Durch Umformen ergibt sich die folgende Formel, womit sich die Höhe aus dem Druck berechnen lässt.

$$h = \ln\left(\frac{p(h)}{p_0}\right) \times (-7991m) = \ln\left(\frac{p(h)}{1013.25hPa}\right) \times (-7991m) \quad (3.19)$$

Die Temperatur wird in zwei Bytes übertragen. Jedoch wird die Temperatur nicht in °C übertragen, sondern in Tausendsteln Grad Celsius. Das bedeutet, dass die Temperatur, welche empfangen wird, zusammengesetzt werden muss und durch 1000 dividiert werden muss. Die Formel sieht folgendermassen aus:

$$T = (\text{float})((\text{byte}_1 \ll 8) + \text{byte}_0) \times 0.001 \quad (3.20)$$

Der letzte Wert, welcher vom Sensor versendet wird, ist die relative Luftfeuchtigkeit. Die Formel zur Berechnung der relativen Luftfeuchtigkeit sieht folgendermassen aus:

$$H = (\text{float})((\text{byte}_3 \ll 24) + (\text{byte}_2 \ll 16) + (\text{byte}_1 \ll 8) + \text{byte}_0) \times 0.01 \quad (3.21)$$

3.6.2 Einstellungsmöglichkeiten

Als nächstes wurde die Aufgabe in Angriff genommen, die Applikation als User einzustellen zu können. Es musste genau überlegt werden, welche Einstellungen ein User machen darf und welche dem Programmierer bzw. der Applikation überlassen werden.

a. Adressauswahl

Die Adressauswahl oder auch Sensorwahl ist ein Bedürfnis, da der User die Applikation mit seinem eigenen Sensor paaren können sollte. Es sollte sichergestellt werden, dass die Applikation nach der Sensorwahl nur noch Daten von dem gewählten Sensor anzeigt und die Daten der anderen Sensoren ignoriert werden. Die Adressen von Sensoren in der Nähe sollten aber weiterhin gespeichert werden, damit man gegebenenfalls auch einen anderen Sensor auswählen konnte.

Immer wenn Daten empfangen werden und die Struktur mit der definierten Struktur übereinstimmen soll die Adresse gespeichert werden. Dafür wurde folgende Funktion entwickelt:

Der adressCounter wird mit dem Wert -1 initialisiert. Werden die ersten Daten mit der vorgegeben Struktur empfangen wird die Funktion automatisch die Adresse in der adressList an der Stelle 0 speichern. Werten erneut Daten von einem unserer Sensoren empfangen wird zuerst geprüft, ob die Adresse bereits ein Teil der adressList ist. Falls die Adresse noch nicht in der Liste enthalten ist, wird der Index bzw. der adressCounter, erhöht und die Adresse in der Liste gespeichert.

Sobald eine Adresse ausgewählt wurde, müssen die zukünftigen Datenpakete überprüft werden, ob sie vom ausgewählten Sensor stammen. Die Funktion checkAdress überprüft, ob die ausgewählte Adresse der Adresse des Sensors entspricht, welche die neuen Daten gesendet hat. Sollte keine Adresse ausgewählt sein, so wird dies behandelt, als ob der Sensor die richtige Adresse hat.

Der User muss noch die Möglichkeit haben eine Adresse auszuwählen, dafür wird eine neue Benutzeroberfläche (Activity) erstellt. Die neue Activity ist eine List-Activity. Das bedeutet, dass es wird eine Liste von möglichen Adressen angezeigt wird, welche mit einem Klick auf die gewünschte Adresse ausgewählt wird. Anschliessend wird die ausgewählte Adresse an die Haupt-Activity zurückgegeben. In Abbildung 3.44 ist die List-Activity zu sehen, welche eine Adresse zur Auswahl hat. Damit die ausgewählte

```
// Speichert die Adresse falls sie noch nicht gespeichert
private void saveAdress(){
    if (adressCounter >= 0) {
        boolean isListed = false;

        for (int i = 0; i <= adressCounter; i++) {
            if (adressList.get(i).compareTo(adress) == 0) {
                isListed = true;
            }
        }
        if (!isListed) {
            adressCounter++;
            adressList.add(adressCounter, adress);
        }
    } else {
        adressCounter = 0;
        adressList.add(adressCounter, adress);
    }
}
```

Abbildung 3.42: Sourcecode saveAdress

```
// Prüft, ob die Adresse des Senders dem Filter entspricht bzw. kein Filter gesetzt ist.
private boolean checkAdress(){
    if (selectedAdress == noAddressFilter){
        return true;
    }
    else if (adressList.get(selectedAdress).compareTo(adress) == 0){
        return true;
    }

    return false;
}
```

Abbildung 3.43: Sourcecode checkAdress

Adresse von der Haupt-Activity überhaupt empfangen werden kann, musste die List-Activity mit den auszuwählenden Adressen mit folgendem Befehl gestartet werden: `startActivityForResult`. Die gestartete Activity kann somit ein Resultat, also die ausgewählte Adresse, zurückgeben und die Haupt-Activity kann diese in der Funktion `onActivityResult` empfangen und abspeichern. Die genaue Ausführung der Übergabe der Adressen sowie die Rückgabe der ausgewählten Adresse kann dem Quellcode entnommen werden.

b. Einheiten

Die Einheiten der empfangenen Daten sollten ebenfalls einstellbar gestaltet werden. Für diesen Zweck wurden mehrere Enumeration definiert. Die Enumeration für die Temperatur sieht folgendermassen aus:

Es wurden der Enumeration drei Elemente hinzugefügt, welche alle einen Faktor, einen Offset und einen String mit der geschriebenen Einheit enthalten. Anhand dieser Enumeration kann der Faktor und der Offset einfach zugänglich gemacht werden. (?)?(?)? Eine neue Activity wurde codiert, welche drei Spinner zur Auswahl der Einheiten,

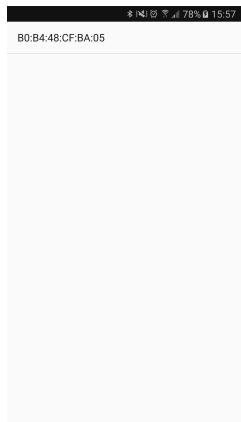


Abbildung 3.44: Adressauswahl mit nur einer Adresse

```
// Enumerationen
private enum tempSetting{
    celsius(1, 0, " °C"),
    fahrenheit(1.8, 32, " °F"),
    kelvin(1, -273.15, " K");

    private double factor, summand;
    private String unit;

    private tempSetting(double factor, double summand, String unit){
        this.factor = factor;
        this.summand = summand;
        this.unit = unit;
    }
}
```

Abbildung 3.45: Sourcecode Enumeration

ein Eingabefeld (EditText) für die Eingabe des Radumfangs und ein EditText für die Kalibrierung der Temperatur zur Verfügung stellt. Die eingestellten Einheiten und die eingetragenen Werte werden über den Speichern-Button an die Haupt-Activity zurückgegeben. In der Haupt-Activity werden diese Daten empfangen und abgespeichert und anschliessend zur Darstellung der Werte verwendet. Erst bei der Anzeige der Werte werden die eingestellten Faktoren und Summanden der Enumerationen verwendet.

Abschliessend wurde die Activity zur Einstellung der Einheiten noch dahingehend verbessert, dass die eingestellten Einheiten und die eingetragenen Werte bei erneutem Aufrufen der Activity dargestellt werden. Wird die Activity zur Einstellung der Einheiten gestartet, wird ihr die aktuelle Einstellung übergeben und die Spinner und EditText zeigen die zuvor abgespeicherten Werte an. Ein Beispiel: Wenn man in der Activity den Spinner für die Temperatur auf "Fahrenheit (°F)" einstellt und diese Einstellung abspeichert und dann die Activity dann erneut über einen Druck auf den Button startet, wird beim Spinner für die Temperatur die Einstellung "Fahrenheit (°F)" angezeigt, da diese Einstellung zuvor abgespeichert wurde.



Abbildung 3.46: Datenübergabe zwischen den Activities

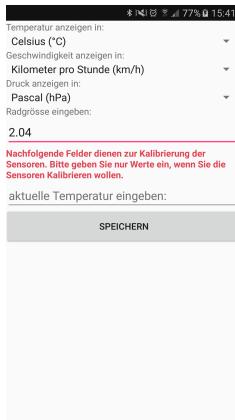


Abbildung 3.47: Bildschirm der Einheiteneinstellung

c. Kalibrierung

Während der Entwicklung wurde festgestellt, dass der Wert der Temperatur von der Referenztemperatur abweicht. Aus diesem Grund wurde eine Möglichkeit zur Kalibrierung der Temperatur geschaffen.

Die Kalibrierungsmöglichkeit befindet sich in der gleichen Activity, wie die Einstellungen der Einheiten und der Eingabemöglichkeit für den Radumfang. Wird bei dem Textfeld ein Wert eingegeben, so wird diese Temperatur an die Haupt-Activity zurückgegeben. Die Referenztemperatur wird gespeichert und beim Empfang der nächsten Daten wird die automatisch ein Offset berechnet. Dieser Offset fließt automatisch in die Berechnung der Temperatur ein.

3.6.3 Anmierter Tachometer

Es sollte ein Tachometer entwickelt werden, welches eine animierte Tachonadel hat, die die aktuelle Geschwindigkeit anzeigt.

a. Funktionsweise

Bereits in der PA von Manuel König wurde mit der Anzeige von Bilddateien, genauer Bitmaps (.bmp-Datei), experimentiert. Es lag nahe, diese Erfahrung zu nutzen und für dieses Problem eine möglichst einfache Lösung zu generieren. In Android ist es möglich,

ein Bitmap in ein Zahlenarray zu laden, dieses Array anzupassen und schlussendlich das Array wieder in ein Bitmap umzuwandeln.

Die Funktion drawTacho liest die Pixel eines Bildes eines Tachometers ohne Tachonadel ein, dreht es und fügt die rote Tachonadel direkt in das Bild ein.

war
scht. ist
ichtig hier?



Abbildung 3.48: Tachometer ohne Tachonadel

Das Bild konnte mit folgender Funktion eingelesen werden:

```
// Pixel in Array einlesen
bitmap.getPixels(pixels, 0, width, 0, width, height);
```

Abbildung 3.49: Sourcecode getPixel

Die Bitmap wird in ein eindimensionales Integer-Array eingelesen, die Werte im Array stellen den Farbwert des jeweiligen Pixels dar. Diese Farbwerte können nun verändert werden. Als erstes wurde das ganze Bild gespiegelt, da der rote Bereich des Tachometers auf der rechten Seite angezeigt werden sollte, wie es von anderen Tachometern bekannt ist.

Nach dem Spiegeln des Bildes musste noch die Tachonadel direkt in das Integer-Array eingefügt werden. Die genaue Vorgehensweise wird im nachfolgenden Punkt (b.) genauer beschrieben.

Aus dem Bearbeiten Integer-Array konnte anschliessend eine Bitmap erstellt werden, welches von der Funktion zurückgegeben wurde.

```
// Bitmap aus dem Array erstellen
bitmap = bitmap.createBitmap(pixels, width, height, Bitmap.Config.ARGB_8888);
return bitmap;
```

Abbildung 3.50: Sourcecode createBitmap

b. Mathematik der Tachonadel

Die Tachonadel wurde in mehreren Schritten animiert, erst wurde ein Strich mit der Breite von einem Pixel animiert. Die Tachonadel kann als ein Vektor angesehen werden,

```
// Zeiger einfügen
int zeigerlength = width/3, zeigerthickness = height/50, middle = (height/2)*width + (width/2);
for (int j = -(zeigerthickness / 2); j < (zeigerthickness / 2); j++){
    for (int i = 0; i < zeigerlength; i++){
        int x, y;
        x = width / 2 + (int)((i * Math.cos(angleShown / 180 * Math.PI)) - (j * Math.sin(angleShown / 180 * Math.PI)));
        y = height / 2 + (int)((i * Math.sin(angleShown / 180 * Math.PI)) + (j * Math.cos(angleShown / 180 * Math.PI)));
        pixels[(y * width) + x] = Color.RED;
    }
}
```

Abbildung 3.51: Sourcecode Mathematik des Zeigers

der im Ursprung des Bitmaps (linke untere Ecke) gedreht und anschliessend verschoben wurde. Die X- bzw. Y-Koordinaten im Bild konnten folgendermassen berechnet werden:

$$x = \frac{\text{Breite des Tachometers}}{2} + \text{Länge der Tachonadel} \times \cos(\text{Winkel der Tachonadel}) \quad (3.22)$$

$$y = \frac{\text{Höhe des Tachometers}}{2} + \text{Länge der Tachonadel} \times \sin(\text{Winkel der Tachonadel}) \quad (3.23)$$

Anschliessend sollte die Tachonadel dicker werden, da ein Strich mit der Breite von nur einem Pixel nicht leicht auf einem hochauflösenden Display zu erkennen ist.

Die Betrachtungsweise, dass die Tachonadel ein Vektor ist, musste angepasst werden. Jeder einzelne Pixel hatte einen Ortsvektor, der angepasst werden musste. Nachfolgende Formel zur Berechnung der Koordinaten nach einer Drehung des Vektors konnte angewendet werden:

$$x_{\text{neu}} = x_{\text{aktuell}} \times \cos(\alpha) - y_{\text{aktuell}} \times \sin(\alpha) \quad (3.24)$$

$$y_{\text{neu}} = x_{\text{aktuell}} \times \sin(\alpha) + y_{\text{aktuell}} \times \cos(\alpha) \quad (3.25)$$

Diese Formeln mussten noch mit der Verschiebung in den Mittelpunkt des Tachometers ergänzt werden. Die Koordinaten jedes Pixels der Tachonadel konnten nun berechnet werden und wurden anschliessend mit dem Farbwert für Rot in dem Integer-Array überschrieben.

$$x_{\text{neu}} = \frac{\text{Breites des Tachometers}}{2} + x_{\text{aktuell}} \times \cos(\alpha) - y_{\text{aktuell}} \times \sin(\alpha) \quad (3.26)$$

$$y_{\text{neu}} = \frac{\text{Höhe des Tachometer}}{2} + x_{\text{aktuell}} \times \sin(\alpha) + y_{\text{aktuell}} \times \cos(\alpha) \quad (3.27)$$



Abbildung 3.52: Tachometer mit animierter Nadel

3.6.4 Modularität der Applikation

Die Applikationsentwicklung basiert auf einer modularen Struktur. Dadurch wird eine Weiterentwicklung der Applikation möglich. Ebenfalls wurde beachtet, dass die Applikation sogar in andere Sprachen übersetzt werden können muss, dafür wurden alle Texte, die ersichtlich sind, in einem File aufgenommen und können zentral abgeändert werden, ohne einen Eingriff in den aktuellen Code. Die verwendeten Texte werden in einem XML-File gespeichert und mit einer ID versehen, die Applikation greift auf die ID zu. Somit kann der Inhalt des Textes angepasst werden, ohne dass der Sourcecode der Applikation angepasst werden muss.

4

Resultate

Im Kapitel 3 ist die Umsetzung ausführlich erklärt. In diesem Kapitel geht es um die relevanten Resultate. Beim Harvester geht es um den produzierten Print, die Leistungsfähigkeit der Schaltung, und die Qualität des Signals am Harvesterausgang. Beim Energy Management wird das Verhalten mit den finalen Schwellwerten dokumentiert. Einerseits geht es um die Kontroller der eingestellten Schwellwert sowie um die erreichte Ladezeit des Primärspeichers STS. Es wird zusammengefasst, bei welcher Geschwindigkeit wie oft BLE-Pakete gesendet werden. Das Verhalten wird im Unterkapitel durch die Energiebilanz gedeutet. Der Zusammenhang zwischen der Geschwindigkeit und dem realen Verbrauch des TI-Sensortags wird für ein erstes Anfahren und für das Weiterfahren beschrieben. Der Vorteil des Weiterfahrens ist, dass die Kondensatoren in diesem Fall bereits geladen sind. Als letztes wird das User Interface der neuen BLE-Applikation gezeigt.

4.1 Harvesterschaltung

Nachfolgend werden die Resultate, der Harvesterschaltung beschrieben. Die Energie, welche die Harvesterschaltung zur Verfügung stellt liegt im μJ -Bereich, das nachfolgende Energymanagement muss mit dieser Energie arbeiten können.

4.1.1 Der Print

Die erstellte Leiterplatte ist in den Abbildungen 4.2 und 4.1 zu sehen. Auf der Ansicht von oben ist der Top-Layer inklusive der Bestückung zu sehen. Die Bestückung des Top-Layers beinhaltet die Harvesterschaltung (unten rechts), den EM8500 (oben rechts) und den Stecker (mittig links), welcher als Verbindung zum TI-SensorTag dient. Ebenfalls sind die vielen Testpunkte zu sehen. Jedes Netz wurde mit einem Testpunkt ausgestattet. Oben links sind die Anschlüsse für die Energiespeicher zu sehen. Hier wurden die beiden Kondensatoren über Litzen angeschlossen. Die Elkos können nicht auf der Oberseite der Leiterplatte platziert werden, da der Platz zwischen TI-SensorTag und dem Prototypen-PCB nicht ausreicht und auf der Unterseite ist kein Platz, da die Speiche des Fahrrads in einem Abstand von ein bis zwei Zentimeter vorbei schnellt. Auf der Ansicht von unten ist der Bottom-Layer inklusive der Bestückung zu sehen. Die Unterseite der Leiterplatte beherbergt die Spule und den Reed-Switch, welche auf

der oberen Seite der Leiterplatte keinen Platz mehr gefunden haben. Der Reed-Switch hätte noch Platz auf der oberen Seite gehabt, jedoch muss der Reed-Switch in der Nähe der Spule platziert werden, da hier der Magnet durchläuft.

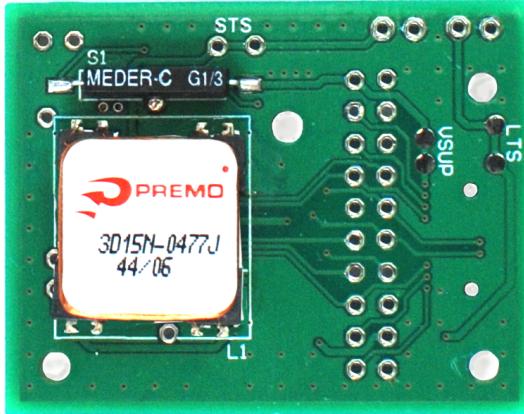


Abbildung 4.1: Print Ansicht von unten

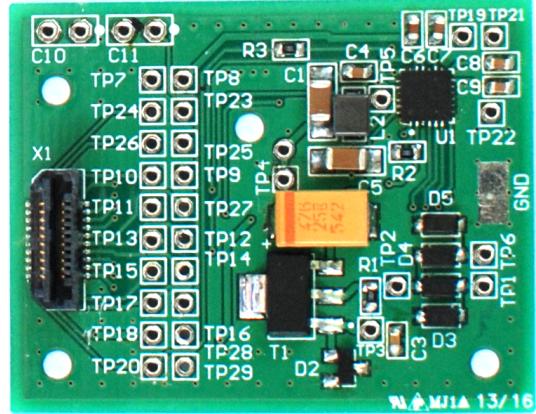


Abbildung 4.2: Print Ansicht von oben

4.1.2 Leistung am Harvesterausgang

Wie erwartet (siehe theoretische Grundlagen b.) ändert sich bei einer Hardware mit einer Spule das Leistungsmaximum. Die Abbildung 4.3 zeigt die reale MPP-Kurve des Prototypen. Bei verschiedenen Geschwindigkeiten ist der MPP an einem anderen Punkt. Problematisch ist, dass die Einstellung im EM8500 nicht so genau eingestellt werden kann. Es sind nur grobe Schritte im Bereich von 50 - 70 % verfügbar.

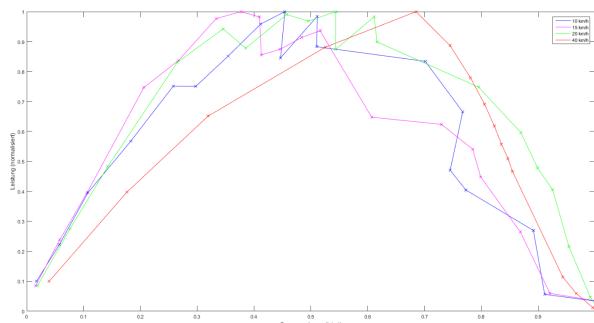


Abbildung 4.3: Leistungskurve Harvesterausgang (normalisiert)

Die Leistung, welche der Harvester zur Verfügung stellen kann, liegt bereits bei einer Geschwindigkeit von 10 km/h bei $24.77 \mu\text{W}$. Dies ist die Durchschnittsleistung bei optimaler Belastung. Es handelt sich nicht um die Spitzenleistung. In der Tabelle 4.1 ist es ersichtlich, dass die Leistung mit der steigender Geschwindigkeit zunimmt.

Tabelle 4.1: Leistung Harvesterschaltung Bicycle Computer

Geschwindigkeit	Leistung Harvester
10 km/h	24.77 μ W
15 km/h	61.07 μ W
20 km/h	116.69 μ W
40 km/h	472.61 μ W

Es ist zu beachten, dass diese Werte die Leistung des MPP sind, somit können diese Leistungen nur erreicht werden, wenn die Einstellung des EM8500-Chips dementsprechend eingestellt werden.

4.1.3 Verhalten des Harvesterausgangs

Die Abbildung 4.4 zeigt das Verhalten des Harvesterausgangs mit einem 47 μ F Elko bei der Belastung durch den EM8500-Chip. Die Spannung ist über einen längeren Zeitraum betrachtet sehr instabil. Diese Problematik wurde bereits im Kapitel ?? beschrieben. Die Abbildung 4.5 zeigt das reelle Verhalten des Harvesterausgangs mit einem 100 μ F Elko. Es ist zu sehen, dass die Spannung am Harvesterausgang über einen längeren Zeitraum relativ konstant bleibt. Genaue Messdaten zu dieser Problematik sind im Messprotokoll ? zu finden.

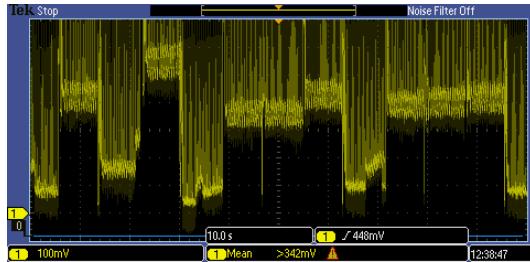


Abbildung 4.4: Spannung VCC beim Harvesterausgang bei 15 km/h

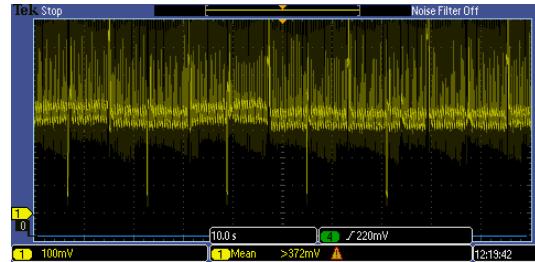


Abbildung 4.5: Spannung VCC beim Harvesterausgang bei 15 km/h

4.1.4 Energie am EM8500-Chipausgang

Die Abbildung 4.6 zeigt die berechnete Energie, die am Ausgang des EM-Chip abgegeben wird in Bezug zur Geschwindigkeit. Die Berechnung, wie aus dem Puls über die Zeit, bis dass VSUP wieder eingeschalten wird, ist im Messprotokoll ? dokumentiert. Die gewonnene Energie, die dem TI-SensorTag zur Verfügung steht ist:

Tabelle 4.2: Leistung EM8500-Chip-Ausgang

Geschwindigkeit	Leistung EM8500_out
10 km/h	$5.44 \mu\text{W}$
15 km/h	$20.91 \mu\text{W}$
20 km/h	$41.39 \mu\text{W}$
40 km/h	$170.75 \mu\text{W}$

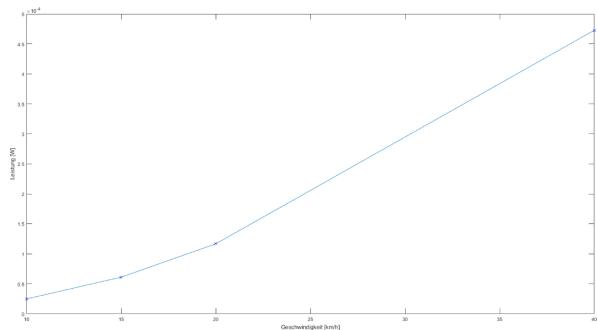


Abbildung 4.6: Maximale Leistung vs. Geschwindigkeit

4.1.5 Wirkungsgrad des Bicycle Computers

Aus den zwei Leistungsmessungen wird graphisch die Energiegewinnung im Vergleich dargestellt (Abbildung 4.7). Die Werte beziehen sich auf die gemittelte Leistung über 10 s. Es ist zu sehen, dass die Energiegewinnung bei der Harvesterschaltung mit der Geschwindigkeit deutlich schneller zunimmt als die Energie am Ausgang des EM-Chips. Aus den zwei Leistungsmessungen wurde der Wirkungsgrad bei den gemessenen Geschwindigkeiten berechnet (siehe Tabelle 4.3). Der Wirkungsgrad ist bei tiefen Geschwindigkeiten sehr schlecht (24.87%). Dies ist eine der zwei Ursachen, dass trotz genug geernteter Energie das TI-SensorTag für das Senden der Sensorwerte bei tiefer Geschwindigkeiten nicht wie erwartet genug Energie hat. Mit mehr Geschwindigkeit erhöht sich nicht nur die geerntete Leistung rapide, sondern auch der Wirkungsgrad (siehe Tabelle 4.3). Der Wirkungsgrad des EM8500 innerhalb des entwickelten Prototypen liegt bei 40 km/h bei 41.01%.

Tabelle 4.3: Wirkungsgrad Bicycle Computer nach Geschwindigkeiten

Geschwindigkeit	Leistung Harvester	Leistung EM8500_out	Wirkungsgrad
10 km/h	$21.87 \mu\text{W}$	$5.44 \mu\text{W}$	24.87 %
15 km/h	$57.19 \mu\text{W}$	$20.91 \mu\text{W}$	36.56 %
20 km/h	$114.67 \mu\text{W}$	$41.39 \mu\text{W}$	36.09 %
40 km/h	$416.29 \mu\text{W}$	$170.75 \mu\text{W}$	41.01 %

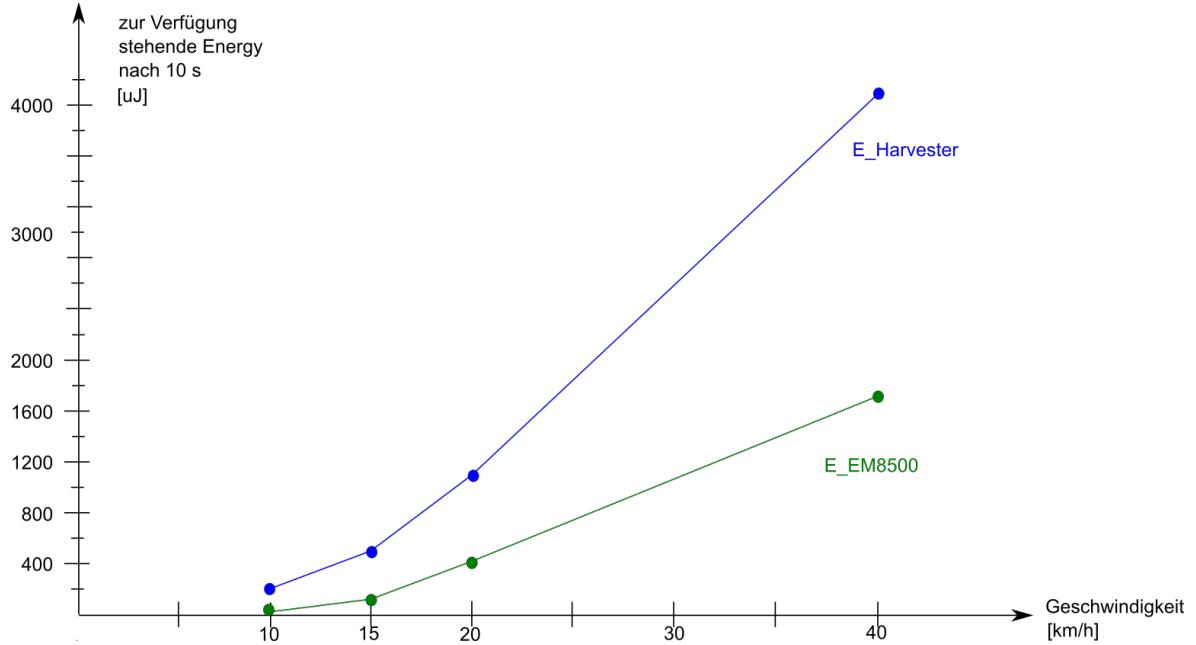


Abbildung 4.7: Energiegewinn Zusammengefasst nach Stelle in der Schaltung

4.2 Energy Management

Bei dem Vorgehen ist die Einteilung der Energiezustände in LOW, MIDDLE und HIGH ENERGY beschrieben (3.4.4). Die folgenden Abbildungen im Unterkapitel 4.2.1 zeigen diese Verhalten. Zuerst wird das Verhalten bei LOW und MIDDLE ENERGY beschrieben. Da interessiert die Ladezeit bis dass wieder ein Paket gesendet wird. Bei beiden Zuständen kann sich der LTS nicht laden, da das Senden über BLE alle Energie verbraucht. Erst bei höherer Energie kann das Ladeverhalten von LTS angeschaut werden. Diese Abbildungen sind im Unterkapitel 4.2.2. Im letzten Unterkapitel stehen die finalen Schwellwerte und Kondensatorenwerte, auf denen diese Messresultate basieren.

4.2.1 Verhalten bei tiefen und mittleren Geschwindigkeiten

Das Primärziel, dass der Energy Harvesting Powered Bicycle Computer bei 10 km/h BLE-Pakete erhält ist gemäss Abbildung 4.8 gelungen. Nach rund 40 Sekunden sendet der Fahrrad-Computer neue Daten. Das rote Signal ist die Speisung V_SUP, die nach dem erreichen des Schwellwertes von V_STS freigeschaltet wird. LTS lädt sich bei niedriger Geschwindigkeit nicht. Die Zeit für das Laden beträgt rund eine Minute. Die Software dieser Version beinhaltet, dass bei der ersten Aktivierung der Speisung der Drucksensor geweckt wird und bei der zweiten Aktivierung der Speisung das Paket versendet wird. So erhält die Fahrerin oder der Fahrer jede zweite Minute ein BLE-Paket mit aktuellen Werten.

Bei 16 km/h verkürzt sich die Ladezeit auf 10 s (siehe Abbildung 4.9). Die Fahrerin bzw. der Fahrer erhält alle 20 s einen aktuellen Wert. Trügerisch in diesem Bild ist, dass LTS bereits geladen ist. Dies entsteht nicht durch 16 km/h, sondern entstand durch vorangehende Messungen mit höheren Geschwindigkeiten. Auf die Ladezeit des STS hat dies keine Auswirkung.

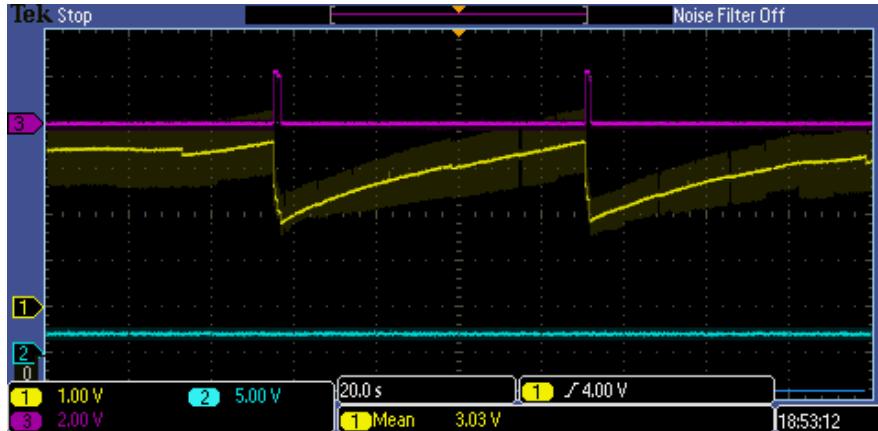


Abbildung 4.8: Senden von Paketen bei 9 - 10 km/h; rot: V_SUP, gelb: V_STS

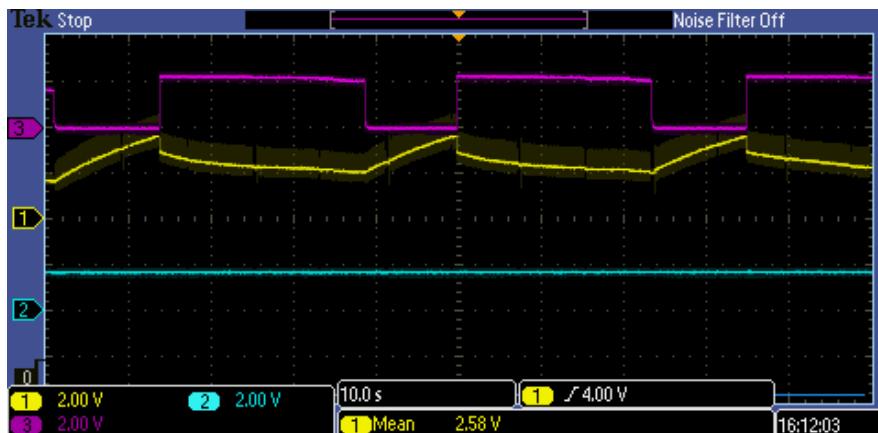


Abbildung 4.9: Senden von Paketen bei 16 km/h; rot: V_SUP, gelb: V_STS, blau: V_LTS

Bei 40 km/h werden rund 15 BLE-Pakete mitgesendet, bis dass V_STS (gelb) unter den Schwellwert fällt und V_SUP (violett) abstellt (Abbildung 4.10). Danach dauert es nicht ganz 2 Sekunden, bis dass wieder nonstop gesendet wird und zwar für die nächsten 2.5 Sekunden. Bei 40 km/h werden alle Sensoren im Ringbufferprinzip ausgelenzen: beim ersten Paket wird der Druck aktualisiert, danach die Temperatur, dann die Luftfeuchtigkeit. LTS lädt sich bei dieser Geschwindigkeit nicht, da V_STS nur bis zum Schwellwert zur Speisung von V_SUP gelangt und nie höher steigt.

4.2.2 Laden und Entladen des LTS

Das zweite Ziel, dass die vom Long Time Storage gespeicherte Energie für das Versenden der BLE-Pakete genutzt wird, wird in diesem Unterabschnitt erklärt. Die Bedingung nach Konzept des EM8500, dass VSTS den Schwellwert von v_appl_max_lo von 3.7 V erreicht (siehe Schwellwerte graphisch dargestellt in 4.11). In der Konfiguration wurde dieser Schwellwert so nah wie möglich an der Schwelle zur Speisung V_SUP mit 3.6 V gewählt. Die Idee ist, sobald einmal V_SUP konstant gespiesen ist, beginnt LTS unmittelbar mit dem Laden. Die Abbildung 4.11 zeigt die finalen Einstellungen, zum

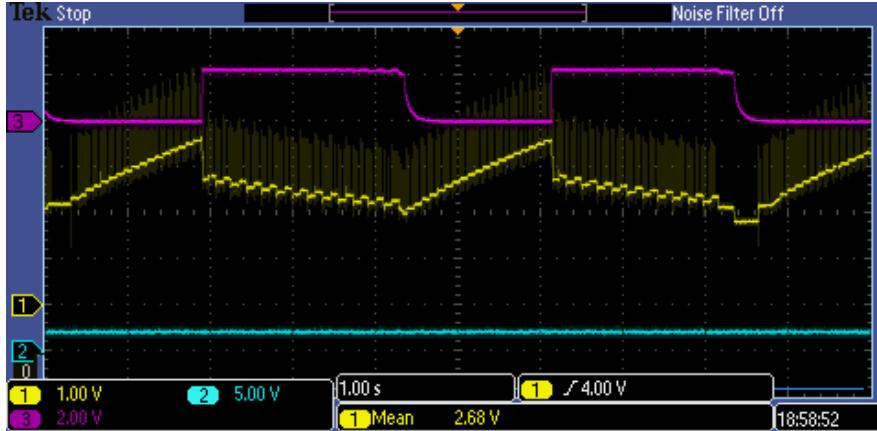


Abbildung 4.10: Senden von vielen Paketen bei 40 km/h

möglichst raschen Laden von LTS.

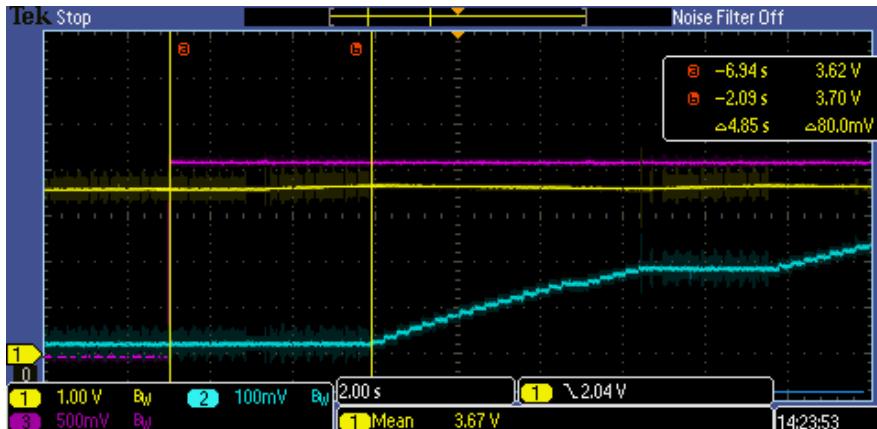


Abbildung 4.11: Einschalten von LTS nach Schwellwert von 3.7 V

Dies ist erst möglich, wenn VSUP konstant gespiesen wird. Denn nur in diesem Fall kann VSTS über den Schwellwert zur Speisung der Applikation von 3.6 V steigen und die Schwelle für die Speisung des LTS bei 3.7 V aktivieren. Dies gelingt ohne Sleep-optimierung ab einer Geschwindigkeit von 50 km/h. Die Abbildung 4.12 zeigt dies bei 60 km/h. LTS hat sich bereits auf die Höhe des STS von 3 V geladen. Ab einer Höhe von 3.6 V laden und entladen sich die beiden Kondensatoren parallel. Sie befinden sich im Connected Mode (siehe Datenblatt S.). Der Connected Mode ist in Abbildung 4.13 dokumentiert.

Die letzte Abbildung (4.14) zum Verhalten des LTS zeigt, wie STS und LTS im Parallelbetrieb sich entladen, V_SUP sich halten kann und beide sich wieder parallel laden. Diese Bilder belegen, dass das Laden und Entladen von LTS gut funktioniert. Das Problem liegt darin, dass bei der (noch nicht sleep-optimierten) TI-SensorTag Version V4 die Sensoren zu viel Energie verbrauchen und V_SUP abfällt. Dadurch kommt es nicht zum Laden des LTS. Erst bei 50 km/h ist das Verhalten exemplarisch demonstriert. Auszugehen ist, dass bei einer sleep-optimierten TI-SensorTag-Version das Laden und

Seite Connected Mode Beschreibung EM8500 Datenblatt

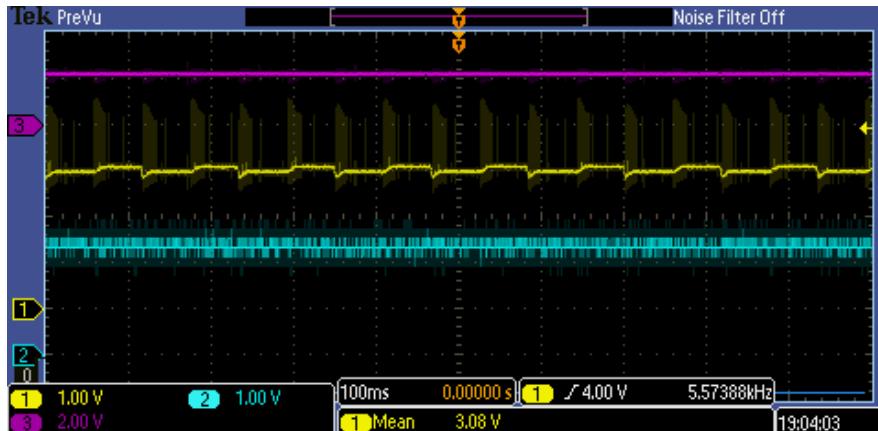


Abbildung 4.12: VSUP wird konstant gespiesen. LTS lädt sich.

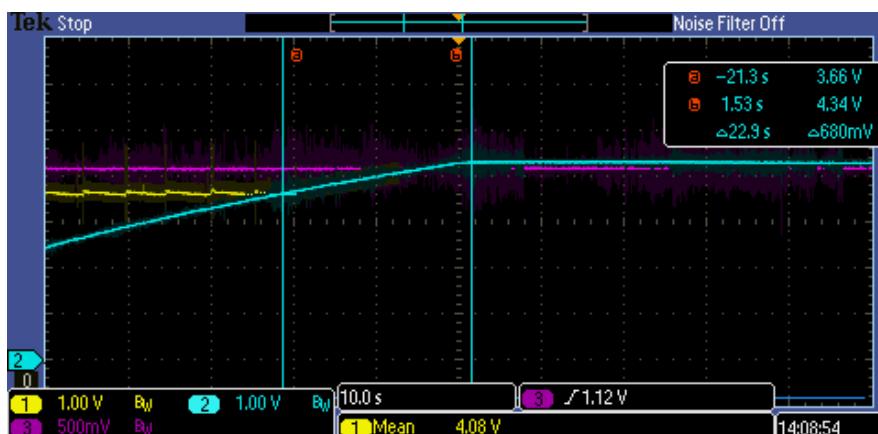


Abbildung 4.13: Connected Mode der zwei Speicher

Entladen von LTS bereits bei tiefen Geschwindigkeiten funktioniert.

4.2.3 Verwendete Werte

Die oben aufgeführten Messwerte sind mit der Software-Version V4 für das TI-SensorTag und den nachfolgenden, genannten Werten umgesetzt. Die Herleitung der Werte steht in den Unterkapiteln Energiekalkulation 3.4.2 und Schwellwerte. Zwei Änderungen der Schwellwerte werden aufgrund letzter Messungen vorgenommen. Die Applikationsspeisung (V_SUP) wird um 0.1 V auf 2.1 V erhöht. Der Grund ist, dass die Speisung bei 2 V, dem Minimum für das TI-SensorTag nicht 100 % funktioniert. Das TI-SensorTag stellte teils selbstständig ab, da angeblich zu wenig Spannung am Eingang anliegt. Deshalb die Spannungserhöhung. Um die gleiche Spannungsdifferenz von $\Delta 0.8$ V zwischen STS_HIGH und STS_LOW zu haben, wird parallel dazu auch v_{bat_min_hi_dis} erhöht. Der neue Wert ist nun 3.0 V, um sicher genügend Energie zur Verfügung zu stellen.

$$\text{STS} = 100 \mu\text{F} \quad (\text{siehe Gleichung 3.8})$$

$$\text{LTS} = 3300 \mu\text{F} \quad (\text{Wert von Vorgängermodell behalten})$$

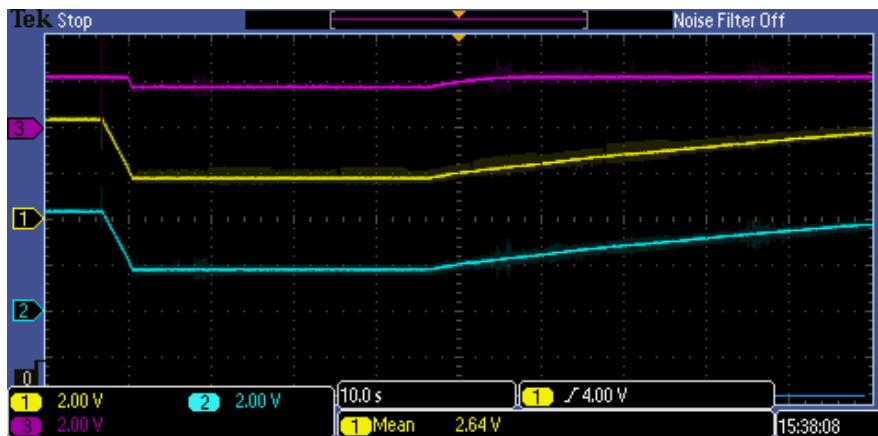


Abbildung 4.14: Paralleles Entladen und Wiederaufladen der Kondensatoren

Tabelle 4.4: Schwellwerte Energy Management Bicycle Computer

Register	Spannungswert
v_bat_max_hi	3.8 V
v_bat_max_lo	3.7 V
v_bat_min_hi_dis	3.0 V (STS_HIGH)
v_bat_min_hi_con	2.2 V
v_bat_min_lo	2.1 V (STS_LOW)
v_appl_max_hi	3.8 V
v_appl_max_lo	3.7 V
mpp-ration	60 %

4.3 Energiebilanz

Das vorangehende Unterkapitel Energy Management zeigte, die erzielten Resultate. Diese Kapitel liefert die Deutung dazu. Dazu werden aus der Tabelle 3.8 im Unterkapitel Energiemessungen den Energieverbrauch nach Aufgaben übernommen und in Zusammenhang mit der gelieferten Energie am EM8500-Ausgang gestellt. Die Energiedaten für den Ausgang des EM8500 stammt aus den Messungen 3.3.3, die in der Tabelle 3.6 zusammengefasst sind.

4.3.1 Energiebilanz Bicycle Computer

Da der Sockelbeitrag an Energie für ein erstes Aufladen der Kondensatoren des TI-SensorTags wie auch des Energiespeichers STS gross ist (siehe 4.15), wird zwischen den Startbedingungen, in denen dieses einmalige Aufladen getan werden muss, und der Nutzung des Bicycle Computers nach einer Minute Fahrzeit unterschieden.

Welche Referenz?? Woran auf?

a. Energiebilanz beim Beginn der Fahrradnutzung

Die Abbildung 4.15 zeigt in horizontalen Balken den Pegel des Energieverbrauchs des TI-SensorTags dar. Unterschieden wird zwischen dem einmaligen Laden (violett), der

Initialisierung (rot) und dem Lesen und Senden der Daten (grau). Total ergibt sich eine Energiebedarf von $203 \mu\text{J}$ wie dies in der Gleichung 3.7 berechnet wird. Die eingetragene grüne Kurve stellt die zur Verfügung stehende Energie am Em8500-Chipausgang dar. Die Datenquellen und die exakten Werte stehen in der Tabelle 4.5 unterhalb der Abbildung 4.15.

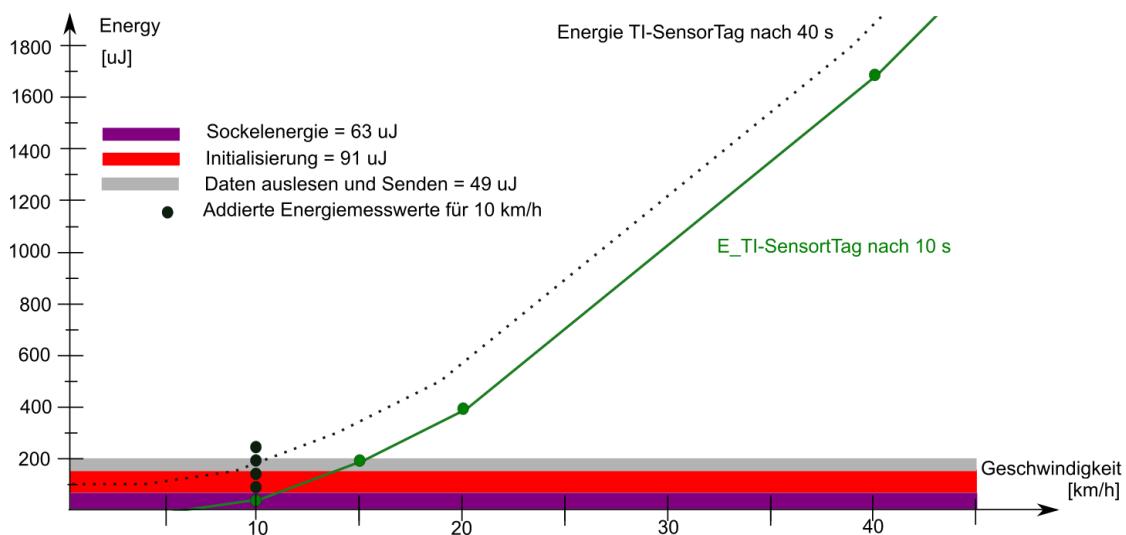


Abbildung 4.15: Energieverbrauch gemäss Verarbeitungsaufwand für CPU

Tabelle 4.5: Energieverbrauch nach Aufgaben

Aufgabe	Farbe	Energieverbrauch	Quelle
Laden C TI-SensorTag	violett	$63 \mu\text{J}$	Tabelle xxx
Initialisierung + Refresh-Zyklen	rot	$77 \mu\text{J} + 14 \mu\text{J} = 91 \mu\text{J}$	Tabelle xxx
Start Sensor + Geschwindigkeit, Daten BLE	grau	$6 \mu\text{J} + 43 \mu\text{J} = 49 \mu\text{J}$	Tabelle xxx
Gesamtverbrauch TI-SensorTag		$203 \mu\text{J}$	Gleichung 3.7

renzen
ellen eing-
gen

Die Abbildung 4.15 zeigt, dass bei der TI-Sensortag-Applikation V4 (siehe) innerhalb von 10 s nicht genug Energie für den Beginn der Fahrradnutzung geht. Wartet die Fahrerin bzw. der Fahrer jedoch länger so addiert sich die gesammelte Energie (siehe Tabelle 4.6). Bei einem minimalen Energieverbrauch von $203 \mu\text{J}$, sollte bei einem Neubeginn der Nutzung des Fahrrad-Computers nach 40 s das erste Datenpaket gesendet werden. Die Wiederholungsrate, mit der neue Paket bei Aufgeladenem System wird im nächsten Unterkapitel dokumentiert.

Tabelle 4.6: Gesammelter Energieverbrauch für 10 km/h

Zeit	Energievorrat	Quelle
10 s	54.4 μ J	Messung xxxx
20 s	108.8 μ J	
30 s	163.2 μ J	
40 s	217.6 μ J	
50 s	272.0 μ J	

Referenz ein
bauen

4.3.2 Energiebilanz nach 1 min Fahrzeit

Fällt das Aufladen der TI-SensorTag-Kondensatoren weg, so sieht die Energiebilanz besser aus. Die Abbildung 4.16 zeigt, dass bei einer Geschwindigkeit von 10 km/h nach 30 s genügend Energie zum Senden eines Paketes mit Sensordaten zur Verfügung steht. Zudem zeigt sich, dass mit zunehmender Geschwindigkeit kein Energieproblem mehr besteht.

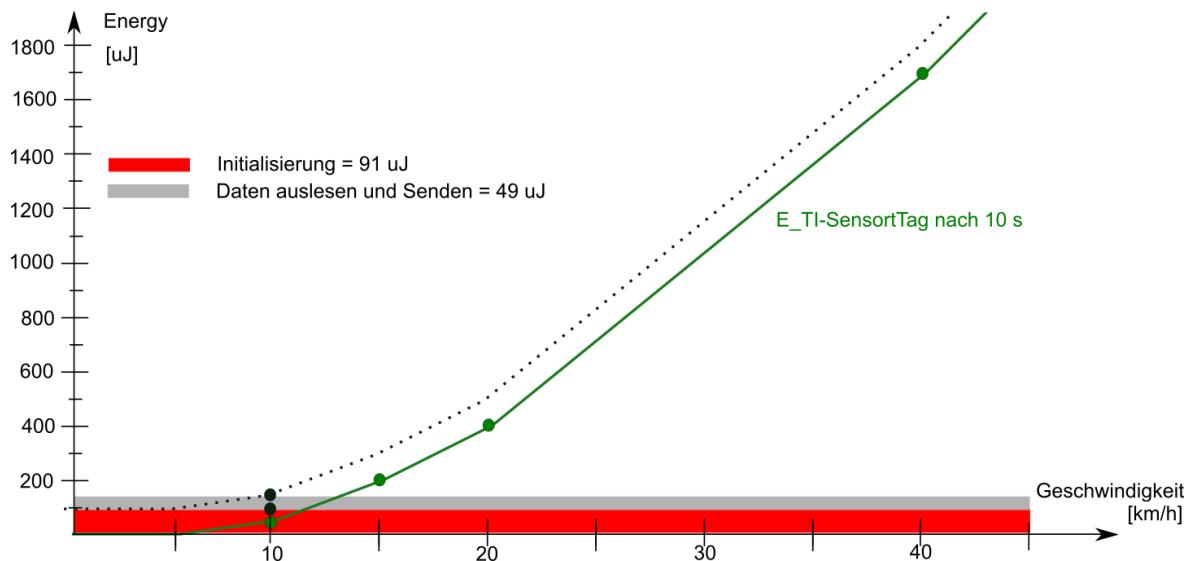


Abbildung 4.16: Energieverbrauch gemäss Verarbeitungsaufwand für CPU

Die zur Verfügung stehende Energie ändert sich mit zunehmender Geschwindigkeit schnell.

Zusammenfassend lassen sich über den Prototypen eines Fahrrad-Computers sagen:

Tabelle 4.7: Resultate Bicycle Computer

1. Das Übermitteln der Geschwindigkeit und Sensordaten ist ab 10 km/h möglich
2. Das Übermitteln erster Daten braucht bei 10 km/h rund 1.5 Minuten.
3. Bei einer Geschwindigkeit von 20 km/h werden die Daten alle 20 s aktualisiert
4. Bei einer Geschwindigkeit von 40 km/h werden die Daten alle 2 s aktualisiert

4.4 Ergebnisse BLE-Applikation

Der Aufbau des TI-SensorTags zusammen mit dem selbst entwickelten Board wird “Sensor” genannt. Dies, weil aus der Sicht einer Benutzerin oder eines Benutzers keine detaillierte Hardware besteht, sondern “ein Sensor”. Die Vereinfachung dient der Benutzerfreundlichkeit. Die Android-Applikation ist bewusst einfach aufgebaut, um die Benutzerin bzw. den Benutzer nicht zu verwirren.

Der animierte Tachometer stellt die aktuelle Geschwindigkeit schnell und übersichtlich dar. Weitergehende Funktionen sind durch prägnante Namen selbsterklärend und der User sollte keine Mühe haben, die App ohne Lesen einer Anleitung zu verstehen.

4.4.1 Applikationsstruktur

Beim Öffnen der Applikation wird geprüft, ob Bluetooth aktiviert ist (siehe Abbildung 4.17). Sollte Bluetooth nicht aktiviert sein, wird der User gefragt, ob Bluetooth aktiviert werden darf. Sollte der User die Aktivierung ablehnen schliesst sich die Applikation sofort.

Wird die Aktivierung der Bluetooth-Schnittstelle erlaubt, erscheint der Startbildschirm. Auf diesem befindet sich zentral der Tachometer (siehe Abbildung 4.18). Dieser zeigt Geschwindigkeiten von 0 – 90 km/h mit einer animierten Tachonadel an. Unterhalb des Tachometers werden die einzelnen Messwerte angezeigt und im untersten Teil des Startbildschirms befinden sich zwei Buttons, über die man Einstellungen vornehmen kann. Die Kontextmenüs zu diesen Einstellungen werden in den nächsten zwei Abbildungen 4.19 und 4.20 ersichtlich.

Wählt man auf dem Startbildschirm “Sensor wählen”, erscheint ein neuer Bildschirm. Auf diesem erscheinen nur die aktiven Bluetooth Geräte mit dem implementierten Prototypen-Filter. Der Bildschirm bildet die TI-SensorTag-Adresse ab. Der Verbund aus unserer neuen Leiterplatte und dem TI-SensorTag wird absichtlich als Sensor bezeichnet, da es sich für den User um ein Gerät handelt, welches Umgebungsdaten misst. Der User sieht das Gerät nicht in seinen Einzelteilen, sondern nimmt es als einen Sensor war. Es wird nicht der Sensor (Temperatursensor, Drucksensor, etc.) auf dem TI-SensorTag ausgewählt wird, sondern die Adresse des Bluetooth-Chips. Jedes TI-SensorTag hat eine eigene Adresse und bei mehreren Prototypen im Raum, kann das entsprechende Gerät ausgewählt werden, da die Adresse nicht sofort mit einem spezifischen TI-SensorTag in Verbindung gebracht werden kann.

Auf dem Startbildschirm befindet sich auch ein Konfigurationsknopf. In das Untermenü gelangt man, indem man den Button “Einheiten + Einstellungen” auswählt.

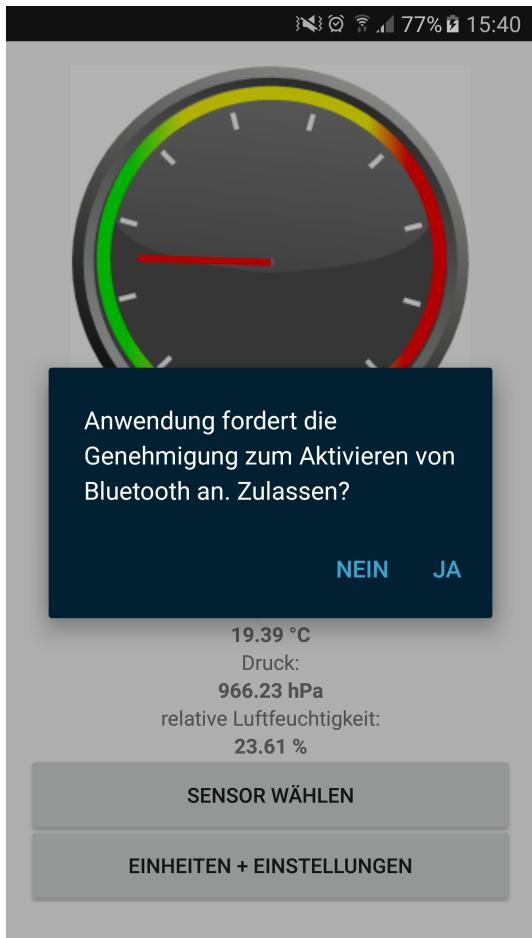


Abbildung 4.17: Bluetooth Permission

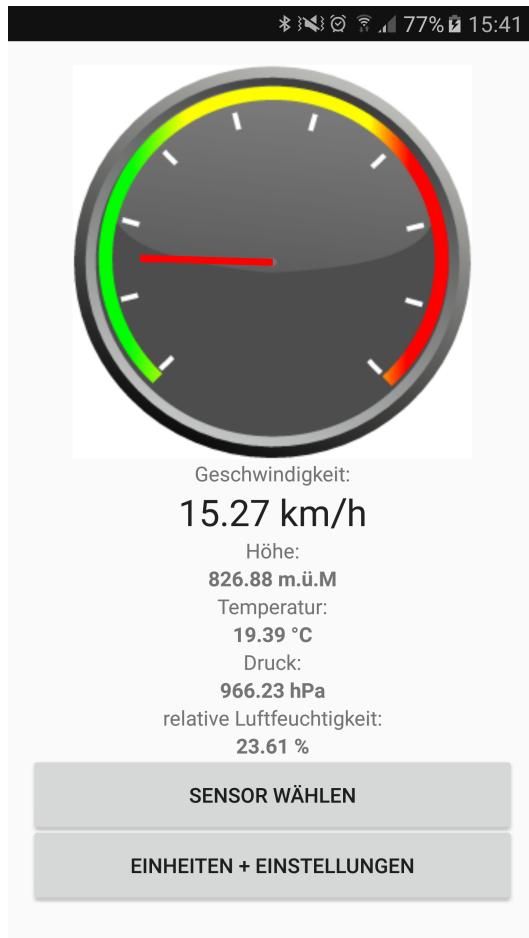


Abbildung 4.18: Startbildschirm der Applikation

Die Applikation stellt mehrere Konfigurationsmöglichkeiten zur Verfügung (siehe Abbildung 4.20). Zu jedem Sensor gehört ein Drop Down-Element, in dem die Einheiten ausgewählt werden können. Neben der Einheitenauswahl verfügt die App über die Möglichkeit, den Radumfang anzupassen und die Temperatur zu kalibrieren.

Die Zahl des Radumfangs kann über die Tastatur in das Feld eingegeben, es können nur Zahlen eingegeben werden. Dies wurde implementiert, um zu verhindern, dass die Eingabe von der Applikation nicht interpretiert werden kann.

Bei der Temperaturanzeige ist eine Kalibrierung eingebaut. Dies, weil alle drei Temperatursensoren auf dem TI-SensorTag einheitlich zu hohe Werte ausgeben. Der zu hohe Temperaturwert liegt am Aufbau des Prototypen. TI-SensorTag und Print liegen nahe aufeinander und es entsteht Wärme im Zwischenraum. Der Benutzer oder die Benutzerin kann den korrekten Wert im Kalibrationsfeld eingeben. Die nächste Temperatur, die empfangen wird, wird mit dem eingetragenen Wert verglichen und ein Offset wird eingestellt. Wird keine Kalibration eingetragen, wird die Temperatur nicht kalibriert und der Offset bleibt unverändert.

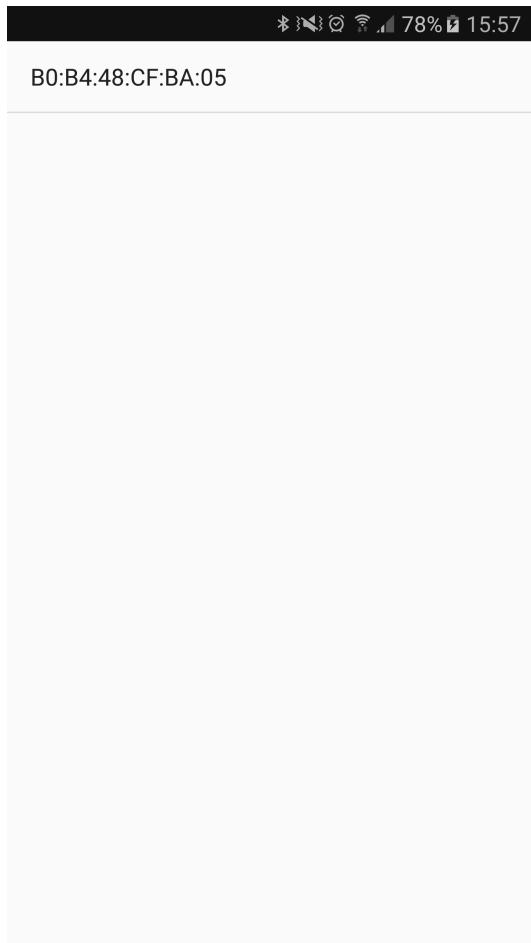


Abbildung 4.19: TI-SensorTagauswahl

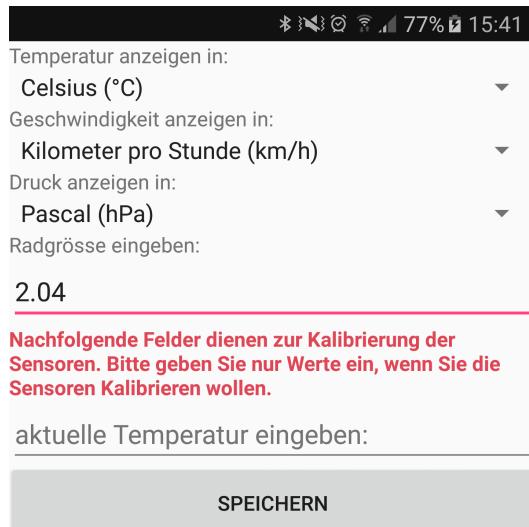


Abbildung 4.20: Einheiten und Einstellungen

Tabelle 4.8: Auswählbare Einheiten

Temperatur	Geschwindigkeit	Druck
Celsius (°C)	Kilometer pro Stunde (km/h)	Pascal (haP)
Fahrenheit (°F)	Miles per hour (mph)	Bar (bar)
Kelvin (K)		Atmosphäre (atm)
		Pound-Force per square inch (psi)
		Millimeter Quecksilber (mmHG)

Die vorgenommenen Einstellungen werden über den Button “Speichern” gesichert. Danach werden die eingelesenen Sensordaten in den ausgewählten Einheiten dargestellt.

4.4.2 Paketverlust

TI-SensorTag sendet im Advertising Mode drei Pakete pro Sendevorgang. Gemessen wird der Paketverlust bei einer Geschwindigkeit von 20 km/h (siehe Tabelle 4.9 und ?).

Tabelle 4.9: Paketverlust BLE-Applikation

Messperiode	Gesendete Pakete	Empfangene Pakete	Paketverlust
1 min	38	35	7.9 %
1 min	37	31	16.2 %
2 min	75	64	14.7 %
2 min	77	65	15.6 %
5 min	193	168	13.0 %
5 min	190	158	16.8 %
10 min	345	301	12.8 %
10 min	349	302	13.5 %

4.4.3 Korrektheit der Daten

Die Korrektheit der empfangenen Daten auf der Applikation wird mit einem visuellen Vergleich der Datenpakete im BLE-Sniffer von TI verglichen (siehe Abbildung 4.21). Die Daten im Sniffer entsprechen exakt den Daten, welche die Applikation empfängt. Der Inhalt der Daten wird somit unverfälscht übermittelt.

P.nbr.	Time [us]	Channel	Access Address	Adv PDU Type	Adv PDU Header			AdvA	AdvData			CRC	RSSI [dBm]	FCS
1	+0 =0	0x25	0x8E89BED6	ADV_NON_CONN_IND	Type	TxAdd	RxAdd	PDU-Length	0xB0B448CFBA05	17 03 DE BA 00 8C 00 00 56 DC 00 01 77 BB 00 00 00 00 00 00 17 2A 00 00	0xD908C	-41	OK	
2	+771575 =771575	0x25	0x8E89BED6	ADV_NON_CONN_IND	Type	TxAdd	RxAdd	PDU-Length	0xB0B448CFBA05	17 03 DE BA 00 8E 00 00 5E 3A 00 01 77 BB 00 00 00 00 00 00 15 C8 00 00	0x1C1AA9	-43	OK	
3	+829577 =-1601152	0x25	0x8E89BED6	ADV_NON_CONN_IND	Type	TxAdd	RxAdd	PDU-Length	0xB0B448CFBA05	17 03 DE BA 00 90 00 00 67 A8 00 01 77 BD 00 00 00 00 00 00 15 C8 00 00	0xAFB5DB	-43	OK	
4	+769719 =-2370871	0x25	0x8E89BED6	ADV_NON_CONN_IND	Type	TxAdd	RxAdd	PDU-Length	0xB0B448CFBA05	17 03 DE BA 00 92 00 00 60 50 00 01 77 BD 00 00 00 00 00 00 15 C8 00 00	0xC48FA	-44	OK	
5	+835185 =3206056	0x25	0x8E89BED6	ADV_NON_CONN_IND	Type	TxAdd	RxAdd	PDU-Length	0xB0B448CFBA05	17 03 DE BA 00 94 00 00 63 F2 00 01 77 BD 00 00 00 00 00 00 16 36 00 00	0xDDE3B0	-45	OK	

Abbildung 4.21: Empfangende Daten des Sniffers

Video zeigen
(auf CD)

<code>receiveddata[0] = 0x17</code>				
<code>receiveddata[1] = 0x3</code>				
<code>receiveddata[2] = 0xDE</code>				
<code>receiveddata[3] = 0xBA</code>				
<code>receiveddata[4] = 0x0</code>				
<code>receiveddata[5] = 0x8C</code>	<code>receiveddata[5] = 0x8E</code>	<code>receiveddata[5] = 0x90</code>	<code>receiveddata[5] = 0x92</code>	<code>receiveddata[5] = 0x94</code>
<code>receiveddata[6] = 0x0</code>				
<code>receiveddata[7] = 0x0</code>				
<code>receiveddata[8] = 0x56</code>	<code>receiveddata[8] = 0x5E</code>	<code>receiveddata[8] = 0x67</code>	<code>receiveddata[8] = 0x60</code>	<code>receiveddata[8] = 0x63</code>
<code>receiveddata[9] = 0xDC</code>	<code>receiveddata[9] = 0x3A</code>	<code>receiveddata[9] = 0xA8</code>	<code>receiveddata[9] = 0x50</code>	<code>receiveddata[9] = 0xF2</code>
<code>receiveddata[10] = 0x0</code>				
<code>receiveddata[11] = 0x1</code>				
<code>receiveddata[12] = 0x77</code>				
<code>receiveddata[13] = 0xBB</code>	<code>receiveddata[13] = 0xBB</code>	<code>receiveddata[13] = 0xBC</code>	<code>receiveddata[13] = 0xBD</code>	<code>receiveddata[13] = 0xBD</code>
<code>receiveddata[14] = 0x0</code>				
<code>receiveddata[15] = 0x0</code>				
<code>receiveddata[16] = 0x0</code>				
<code>receiveddata[17] = 0x0</code>				
<code>receiveddata[18] = 0x0</code>				
<code>receiveddata[19] = 0x0</code>				
<code>receiveddata[20] = 0x17</code>	<code>receiveddata[20] = 0x15</code>	<code>receiveddata[20] = 0x15</code>	<code>receiveddata[20] = 0x15</code>	<code>receiveddata[20] = 0x16</code>
<code>receiveddata[21] = 0x2A</code>	<code>receiveddata[21] = 0xC8</code>	<code>receiveddata[21] = 0xC8</code>	<code>receiveddata[21] = 0xC8</code>	<code>receiveddata[21] = 0x36</code>
<code>receiveddata[22] = 0x0</code>				
<code>receiveddata[23] = 0x0</code>				
<code>actVelocity = 10.822458</code>	<code>actVelocity = 9.9762945</code>	<code>actVelocity = 9.068744</code>	<code>actVelocity = 9.760228</code>	<code>actVelocity = 9.405463</code>
<code>actPressure = 961.87</code>	<code>actPressure = 961.87</code>	<code>actPressure = 961.88995</code>	<code>actPressure = 961.88995</code>	<code>actPressure = 961.88995</code>
<code>actAltitude = 415.8433</code>	<code>actAltitude = 415.8433</code>	<code>actAltitude = 415.67773</code>	<code>actAltitude = 415.67773</code>	<code>actAltitude = 415.67773</code>
<code>actTemperatur = 19.39</code>				
<code>actHumidity = 59.3</code>	<code>actHumidity = 55.76</code>	<code>actHumidity = 55.76</code>	<code>actHumidity = 55.76</code>	<code>actHumidity = 56.86</code>

Abbildung 4.22: Empfangene Daten der Applikation

5

Diskussion

Am Ende der Arbeit steht ein funktionstüchtiger Prototyp eines Bicycle Computers zur Verfügung. Die Leistungsgewinnung ist um xxx optimiert, der Aufbau auf eine Leiterplatte miniaturisiert, das Energy Management auf die minimale Fahrgeschwindigkeit von 10 km/h optimiert sowie neu werden auch Sensordaten an die benutzerfreundliche Android-Applikation gesendet.

Der Prototyp ist so entwickelt, dass zukünftige Teams den Code schnell verstehen, die Leiterplatte leicht aufteilen und weiterentwickeln können und die modulare Android-Applikation ist nach Belieben ausbaubar.

Die Minimalanforderungen der definierten Aufgabenstellung wurden alle erreicht. Von den optionale Aufgaben sind nachfolgende Anforderungen eingebettet worden: Die Harvesterschaltung wurde optimiert, das Energy Management ist auf verschiedene Geschwindigkeiten angepasst, zusätzliche Sensoren (Temperatur und Feuchtigkeit) und das Auftrennen der Verarbeitungsschritte in "Daten speichern" und bei genügend Energie "senden".

Als mögliche Weiterentwicklungen stehen insbesondere Energieverbrauchsoptimierungen an erster Stelle. Das Ziel ist, dass bis zur Präsentation des Bicycle Computer an der Nacht der Technik, die Sensoren mit weniger Energie ausgelesen werden, sodass die Versorgungsspannung bei Geschwindigkeiten unter 40 km/h nicht abstellt. Interessant ist die Auswirkung des Connected Modes bei höherer Geschwindigkeit. Bei tieferen Geschwindigkeiten als 40 km/h ist der Verbindungsauflauf nicht realistisch. Doch bei höheren Geschwindigkeiten könnten über den Connected Mode sicher Daten versendet werden.

Die entwickelte Leiterplatte kann für Schülerinnen und Schüler zum Experimentieren mit BLE verwendet werden. Bewusst wurden viele Testpunkte und ein Stecker für das Abgreifen der Signale implementiert. Montagelöcher für eine Befestigung sind vorhanden und Steckplätze für die Kondensatoren gewähren einen flexiblen Einsatz der Kondensatoren. Das TI-SensorTag zusammen mit dem EM8500 eignen sich gut zum Kennenlernen des Energy Managements.

Faktor

Für die Vorführung des fertigen Produkts ist ein Gehäuse in Planung. Dieses wird an der Verstrebung zwischen Lenker und Sattel montiert. Die Distanz zum Rad kann flexibel eingestellt werden, sodass der Prototyp sich nicht auf ein Fahrradmodell limitiert. Zum Endprodukt zählt auch eine professionelle Montage der Doppelmagnete an den Speichen. Als Letztes wird der Hohlraum zwischen der Leiterplatte und dem TI-SensorTag durch zwei Verstrebungen verstärkt, damit das Gerät Bodenschläge aushält. Das Endprodukt ist somit ein voll anwendungsfähiger Bicycle Computer.

Auf das Produkt sind wir Stolz. Dies nicht zu Letzt, da die Umsetzung nicht ganz einfach war. Der EM8500-Chip ist ein neues Produkt und verhält sich teilweise nicht stabil. Konkret musste Manuel König acht EM8500-Chips auf die Leiterplatten anlöten, weil immer wieder einer ausstieg. Das heisst, entweder konnte die Kommunikation über SPI nicht mehr stattfinden, Slave-Address-Error, oder V_SUP startete trotz grosser Energie am Eingang nicht. Da der Chip nicht zuverlässig funktionierte, war es in der Entwicklung schwierig zu unterscheiden, ob in der Harvesterschaltung ein Fehler auftrat oder der Chip einen Defekt aufwies. Zu oft suchten wir den Fehler in der Hardware und nicht im Chip. Die zweite Herausforderung war die Benutzung des TI-SensorTags ohne RTOS. Diese rudimentäre Programmierung machte Spass. Mit Hilfe des Know-Hows am InES konnten die Probleme längerfristig gelöst werden. Da es Pionierarbeit ist, ging die Entwicklung des Codes nicht so schnell wie gewünscht vorwärts. Sinnvoll ist z.B. bei mehr Energie per SPI das Status Register des EM8500 auszulesen. In diesen 8 Bits steht der Zustand des Energiezustands der Speicher und welche Schwellwerte überschritten sind.

Abschliessend möchten wir sagen, dass die Unterstützung bei der Bachelorarbeit sowohl von Prof. Dr. Meli wie auch von Research-Assistent Dario Dündar sehr gut und zeitnah war. Die Probleme, die auf uns zukamen, waren viel grösser als erwartet. Da wir es super im Team hatten, ergänzten wir uns auf eine gute Art und Weise. Die Freude blieb trotz teilweiser technischen Schwierigkeiten nie aus. Durch diese Arbeit haben wir sehr viel gelernt und freuen uns, zukünftig als Ingenieurin und als Ingenieur zu arbeiten.

6

Verzeichnisse

6.1 Glossar und Abkürzungen

Clock Domain

Ein Bereich der Hardware, der mit demselben Takt läuft.

GPIO

General Purpose Input and Output bezeichnet die externe Schnittstelle zum Sensor-
tag. Alle Zustände, die das EM8500 sendet, gelangen über die GPIO zum Sensor-
tag. Für den Bicycle Computer ist der Reed Switch-Eingang der wichtigste, da aus diesem
die Geschwindigkeit berechnet werden kann.

InES

Ist der Institutsname eines Instituts der ZHAW. InES steht für Institut of Embedded
Systems.

KO

Steht für Kathodenstrahl-Oszilloskop und bezeichnet ein Messgerät, dass Spannung
im Zeitverlauf aufzeichnet.

MCU

MCU steht für Microkontroller Unit und bezeichnet eine Einheit aus einem Pro-
zessor und den Peripheriebausteinen. Eine MCU ist ein lauffähiges System um einen
MicroProzessor herum.

Power Domain

Basiert auf der Fähigkeit eines Prozessor Speisungsgebiete zur Verfügung zu stel-
len. Der Prozessor teilt seine Funktionalitäten in Gebiete ein, die separat ein- und
ausgeschalten werden können.

MPP

Maximum Power Point (MPP) bezeichnet in einer Leistungskurve den höchsten
Punkt, also das Leistungsmaximum.

MPPT

Versucht ein System, einen Input stets auf das Leistungsmaximum zu regeln, spricht
man von Maximum Power Point Tracking. Tracking steht für Einfangen.

MPP-Ratio

Bezeichnet die Auswertung des MPP auf Spannungsachse. Liegt das Leistung maxi-
mum beim Kurzschluss, so ist die MPP-Ratio bei 0 %, liegt sie bei Leerlauf, dann liegt

die MPP-Ratio bei 100 %. Üblicherweise liegt die MPPT-Ratio dazwischen.

MPPT-Ratio

Einstellungswert eines Registers im EM8500-Chip. Durch den Wert gibt man vor, auf welchen MPP das System sich einstellen (tracken) soll.

RTC

Der Real Time Clock kann im Cortex M3 bei ausgeschaltener CPU weiterlaufen.

State Machine

Heisst korrekt Finite State Machine und bezeichnet eine Konzept, bei dem aufgrund einer Kombination von Eingangssignalen, sich das System in einem bestimmten Zustand befindet. In jedem Zustand sind nur gewisse Inputs zulässig, ansonsten verbleibt das System in diesem Zustand. Folgt ein korrekter Input, wechselt das System in den entsprechenden Zustand.

UML

Die Unified Modeling Language (UML) ist ein Quasistandard, wie Prozesse abgebildet werden können. Die Sprache definiert Formen, aufgrund deren man weiß, ob es sich um eine Initialisierung, eine Entscheidung oder um eine Verarbeitung, etc. handelt.

6.2 Abbildungsverzeichnis

Abbildung 1.1	Arbeitsblöcke	16
Abbildung 2.1	Theoretische Abbildung einer Gleichspannung am Ausgang eines TEG	20
Abbildung 2.2	Wechselspannung bei Bewegungsinduktion	20
Abbildung 2.3	Rippelspannung aufgrund der gepulsten Eingangsenergie	21
Abbildung 2.4	MPP TEG (?)	21
Abbildung 2.5	MPP Solarzelle (?)	21
Abbildung 2.6	MPP Spule (?)	22
Abbildung 2.7	MPP Harvester (?)	22
Abbildung 2.8	Grundprinzip Applikationsspeisung	23
Abbildung 2.9	Applikationsspeisung EM8500	23
Abbildung 2.10	Sicheres Betreiben durch Long Term Storage	24
Abbildung 2.11	Konzept Hersteller (?)	24
Abbildung 2.12	Leistungsmessung des Harvesters	25
Abbildung 2.13	In- und Outputs EM8500 (? , p.11)	27
Abbildung 2.14	Schlafen zwischen Ausführungen	29
Abbildung 2.15	Schlafen innerhalb des Codes	30
Abbildung 2.16	BLE Paketstruktur	32
Abbildung 3.1	Funktionsblöcke Bicycle Computer	36
Abbildung 3.2	Spannungswerte Modell der Machbarkeitsstudie	37
Abbildung 3.3	Rippelspannung über dem $47 \mu\text{F}$ Kondensator	40
Abbildung 3.4	Rippelspannung über dem $10 \mu\text{F}$ Kondensator	40
Abbildung 3.5	Spannung am EM8500-Eingang über einem $47 \mu\text{F}$ Kondensator	40
Abbildung 3.6	Spannung am EM8500-Eingang über einem $100 \mu\text{F}$ Kondensator	40

Abbildung 3.7 Harvesterschaltung der PA15	42
Abbildung 3.8 Schema des EM-Chips mit der benötigten Peripherie	43
Abbildung 3.9 Schema der Energiespeicher (Elkos sind nur Platzhalter)	44
Abbildung 3.10 Schema Umlauferfassung	44
Abbildung 3.11 Spannung über der Spule von Premo, Geschwindigkeit 20 km/h	45
Abbildung 3.12 Spannung über der Spule von Würth, Geschwindigkeit 20 km/h	45
Abbildung 3.13 Gleichrichter 1N5819	46
Abbildung 3.14 Gleichrichter aus HSMS-286P, 15 km/h	46
Abbildung 3.15 Gleichrichter 1N5819	46
Abbildung 3.16 Gleichrichter aus BAT54, 15 km/h	46
Abbildung 3.17 Links: Dioden-Limiter	47
Abbildung 3.18 Rechts: FET-Limiter, 15 km/h	47
Abbildung 3.19 Pads der Leiterplatte, rot eingerahmt die Testpunkte des Steckers	49
Abbildung 3.20 Messstellen am Prototypen	51
Abbildung 3.21 Anregung der Spule mit einem Magneten	53
Abbildung 3.22 Anregung der Spule mit zwei Magneten in Serie	53
Abbildung 3.23 rot: VCC_STS, gelb: VSUP	55
Abbildung 3.24 Minimalster Energieverbrauch: 3 BLE Pakete über Advertising Mode senden	56
Abbildung 3.25 Auslesen der Sensoren reisst Energie zusammen	57
Abbildung 3.26 Überblick Energieverbrauch mit Refreshzyklen	59
Abbildung 3.27 Einmaliges Aufladen der Kondensatoren TI-SensorTag	59
Abbildung 3.28 Energieverbrauch Refreshzykus	60
Abbildung 3.29 Energieverbrauch für Initialisierung	60
Abbildung 3.30 Energieverbrauch senden eines BLE-Paketes mit Sensordaten	61
Abbildung 3.31 Energieverbrauch des Drucksensors BMP 280	62
Abbildung 3.32 Energiemessung Temperatursensor auf TI-SensorTag	63
Abbildung 3.33 Energieverbrauch Feuchtigkeitssensor	63
Abbildung 3.34 Energieverbrauch pro Aufgabe	64
Abbildung 3.35 Wenig Energie	69
Abbildung 3.36 Mittlere Energie	69
Abbildung 3.37 Viel Energie	69
Abbildung 3.38 Supply System TI-SensorTag aus: ?, S. 416	71
Abbildung 3.39 Schlafenszeiten zwischen Aktionen	72
Abbildung 3.40 Auf trennen der Arbeit durch Schlafen dazwischen	73
Abbildung 3.41 Aufteilung des Bildschirms	74
Abbildung 3.42 Sourcecode saveAdress	79
Abbildung 3.43 Sourcecode checkAdress	79
Abbildung 3.44 Adressauswahl mit nur einer Adresse	80
Abbildung 3.45 Sourcecode Enumeration	80
Abbildung 3.46 Datenübergabe zwischen den Activities	81
Abbildung 3.47 Bildschirm der Einheiteneinstellung	81
Abbildung 3.48 Tachometer ohne Tachonadel	82
Abbildung 3.49 Sourcecode getPixel	82
Abbildung 3.50 Sourcecode createBitmap	82

Abbildung 3.51	Sourcecode Mathematik des Zeigers	83
Abbildung 3.52	Tachometer mit animierter Nadel	84
Abbildung 4.1	Print Ansicht von unten	86
Abbildung 4.2	Print Ansicht von oben	86
Abbildung 4.3	Leistungskurve Harvesterausgang (normalisiert)	86
Abbildung 4.4	Spannung VCC beim Harvesterausgang bei 15 km/h	87
Abbildung 4.5	Spannung VCC beim Harvesterausgang bei 15 km/h	87
Abbildung 4.6	Maximale Leistung vs. Geschwindigkeit	88
Abbildung 4.7	Energiegewinn Zusammengefassst nach Stelle in der Schaltung	89
Abbildung 4.8	Senden von Paketen bei 9 - 10 km/h; rot: V_SUP, gelb: V_STS	90
Abbildung 4.9	Senden von Paketen bei 16 km/h; rot: V_SUP, gelb: V_STS, blau: V_LTS	90
Abbildung 4.10	Senden von vielen Paketen bei 40 km/h	91
Abbildung 4.11	Einschalten von LTS nach Schwellwert von 3.7 V	91
Abbildung 4.12	VSUP wird konstant gespiesen. LTS lädt sich.	92
Abbildung 4.13	Connedted Mode der zwei Speicher	92
Abbildung 4.14	Paralleles Entladen und Wiederaufladen der Kondensatoren	93
Abbildung 4.15	Energieverbrauch gemäss Verarbeitungsaufwand für CPU .	94
Abbildung 4.16	Energieverbrauch gemäss Verarbeitungsaufwand für CPU .	95
Abbildung 4.17	Bluetooth Permission	97
Abbildung 4.18	Startbildschirm der Applikation	97
Abbildung 4.19	TI-SensorTagauswahl	98
Abbildung 4.20	Einheiten und Einstellungen	98
Abbildung 4.21	Empfangende Daten des Sniffers	99
Abbildung 4.22	Empfangene Daten der Applikation	100
Abbildung A.1	Offizielle Ausschreibung der Arbeit	I
Abbildung B.1	Blockschema Sensortag	III
Abbildung C.1	Blockschema TI-SensorTag aus ?, S. 3	V
Abbildung D.1	Abbildung aus ?, S. 418	VII
Abbildung E.1	Messaufbau während der Inbetriebnahme des Prototypen .	X

6.3 Tabellenverzeichnis

Tabelle 3.1	Leistung des fliegenden Aufbaus	51
Tabelle 3.2	Leistung der neuen Leiterplatte	51
Tabelle 3.3	Leistung mit unterschiedlichen Spulen	52
Tabelle 3.4	Leistung mit unterschiedlicher Anzahl an Magneten	53
Tabelle 3.5	Leistung des Prototypen	54
Tabelle 3.6	Leistungsabgabe Ausgang EM8500	54
Tabelle 3.7	Messresultate nach TI-SensorTag-Versionen	57
Tabelle 3.8	Zusammenfassung Energieverbrauch nach Aktion	64

Tabelle 3.9	MPPT-Ratio Einstellungen EM8500	66
Tabelle 3.10	Konfiguration Vorgängermodell	66
Tabelle 3.11	Konfiguration aufgrund Geschwindigkeitsmessung	67
Tabelle 3.12	Energiezustände aufgrund der Geschwindigkeit	68
Tabelle 3.13	Zählerstände für Schlafenszeit	74
Tabelle 3.14	Filter	76
Tabelle 4.1	Leistung Harvesterschaltung Bicycle Computer	87
Tabelle 4.2	Leistung EM8500-Chip-Ausgang	88
Tabelle 4.3	Wirkungsgrad Bicycle Computer nach Geschwindigkeiten	88
Tabelle 4.4	Schwellwerte Energy Management Bicycle Computer	93
Tabelle 4.5	Energieverbrauch nach Aufgaben	94
Tabelle 4.6	Gesammelter Energieverbrauch für 10 km/h	95
Tabelle 4.7	Resultate Bicycle Computer	96
Tabelle 4.8	Auswählbare Einheiten	98
Tabelle 4.9	Paketverlust BLE-Applikation	99

Anhang A

Ausschreibung Bachelorarbeit

Bachelorarbeit 2016 – FS: BA16_mema_1

Titel

Energy harvesting powered bicycle computer

Beschreibung

Im Rahmen dieser Bachelorarbeit soll ein batterieloses Geschwindigkeitsmessgerät für Fahrräder entwickelt werden. Das Messgerät wird an der Gabel des Fahrrades befestigt und misst die Anzahl Durchgänge eines an den Speichen befestigten Magneten. Der Tachometer gewinnt die gesamte benötigte Energie aus den Magnetdurchläufen. Die gemessenen Daten werden per Bluetooth an einen Fahrradcomputer (in diesem Fall ein Mobiltelefon) gesendet.

Der Machbarkeitsbeweis wurde durch die vorangegangene Projektarbeit „Bicycle computer and sensoric powered with harvested energy“ (PA15_mema_1) bereits erbracht. Der hier entwickelte Fahrradcomputer wird sich durch ein intelligentes Energieverwaltungssystem vom Prototypen abheben. Dies bedeutet, der Fahrradcomputer soll wissen, wie viel Energie bereits gespeichert wurde und er soll prognostizieren, wie viel Energie in näherer Zukunft geerntet werden kann. Aufgrund dieser und möglicherweise weiteren Informationen berechnet der Fahrradcomputer den optimalen Zeitpunkt und Zeitabstand, um die gemessenen Geschwindigkeitsdaten zu versenden. Sobald genug Energie zur Verfügung steht, beginnt der Fahrradcomputer außerdem Werte wie den Luftdruck (Höhe), die Temperatur und die Luftfeuchtigkeit zu messen. Optional kann auch ein Bewegungssensor in Betrieb genommen werden.

Es soll außerdem eine Smartphone App entwickelt werden, welche die gemessenen Geschwindigkeitsdaten, Sensordaten und aktuelle Werte der Energieverwaltung empfängt sowie darstellt. Optional kann ein Kommunikationskanal vom Smartphone zum Fahrradcomputer implementiert werden, um Parameter wie Radumfang oder Sicherheitseinstellungen auszutauschen. Selbstverständlich müsste der Fahrradcomputer auch dann noch ohne Batterie auskommen.

Firmware: 40%
Hardware: 30%
App: 30%

Abbildung A.1: Offizielle Ausschreibung der Arbeit

Anhang B

Blockdiagramm EM8500

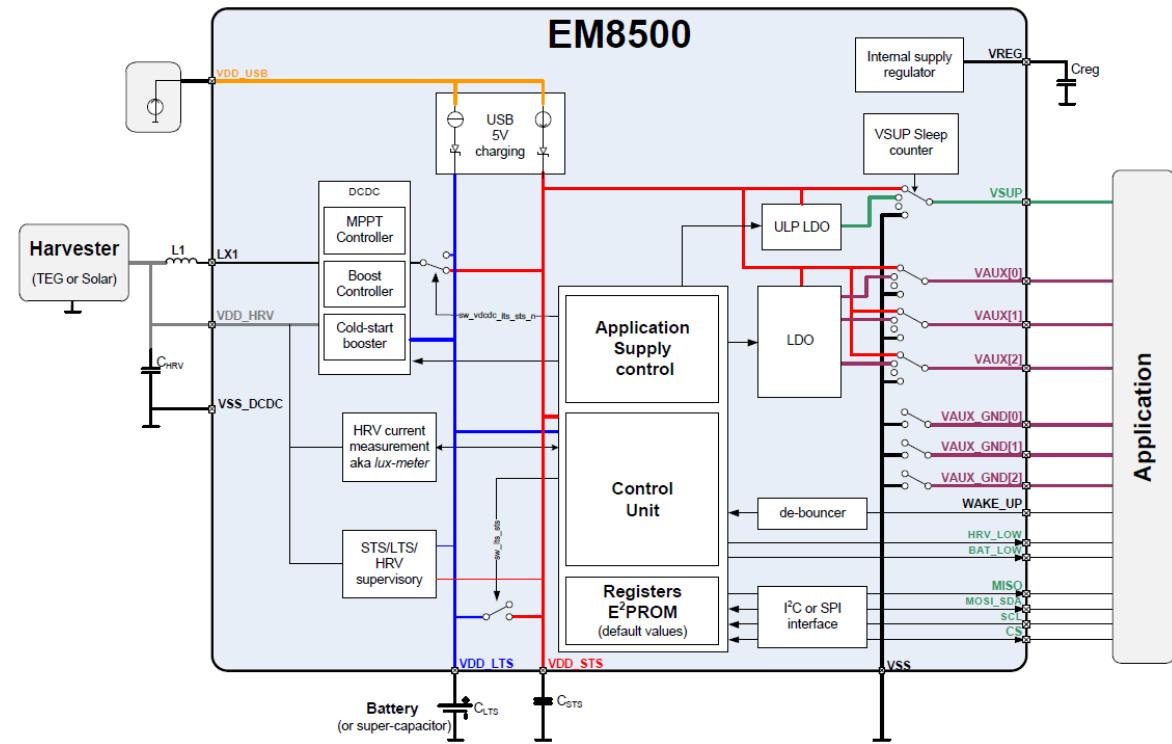


Abbildung B.1: Blockschema Sensortag

Anhang C

Funktionsblöcke des TI-SensorTags

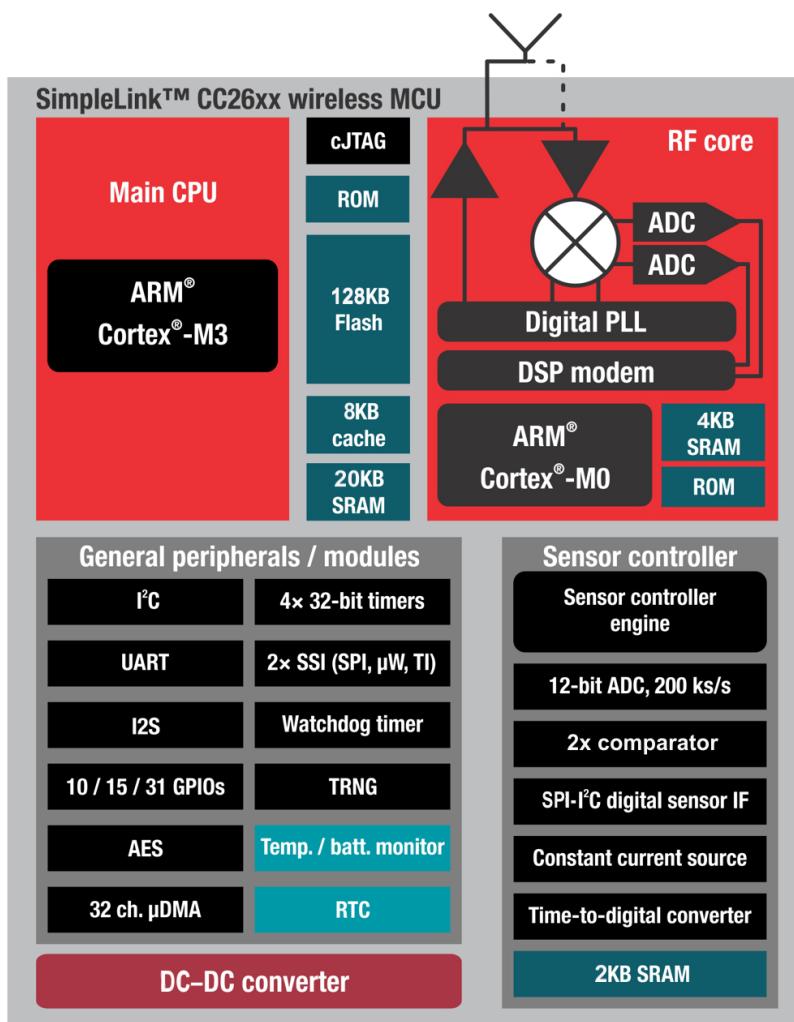


Abbildung C.1: Blockschema TI-SensorTag aus ?, S. 3

Anhang D

Digital Power Partitioning beim TI-SensorTag

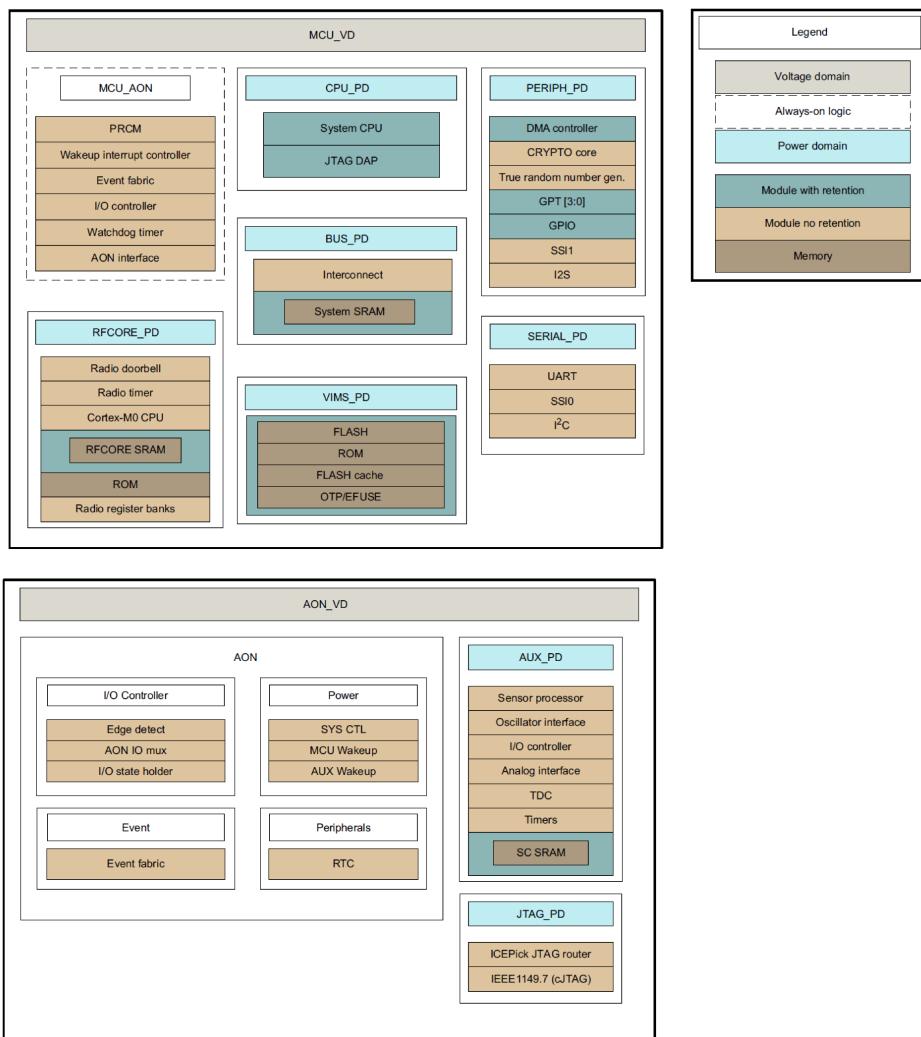


Abbildung D.1: Abbildung aus ?, S. 418

Anhang E

Messaufbau

Problematisch war, dass die Messresultate mit dem Fahrrad nicht reproduzierbar waren, daher wurde eine Radimitation erarbeitet. Herr Erich Ruff hat einen Aufbau entwickelt, welcher über einen Elektromotor angetrieben wurde, damit konnte die Geschwindigkeit relativ konstant gehalten werden. Die Toleranz der Geschwindigkeit lag bei +/- 1 km/h, was eine grosse Verbesserung gegenüber der bisherigen Vorgehensweise war. Die Radimitation bestand aus nur einer Speiche, welche nur eine einfache Alustange war. An dieser Alustange waren mehrere Löcher zur Befestigung des Topfmagneten vorhanden.

Der Radimitation ist ein Tachometer angehängt, der die aktuelle Geschwindigkeit anzeigt. Der Tachometer ist vom Hersteller Sigma Sport, die genaue Bezeichnung lautet Sigma Sport Baseline 500. Dieser Tachometer wurde im Jahr 1999 hergestellt, funktioniert jedoch nach wie vor einwandfrei. Im Verlauf der Entwicklung wurde entschieden, dass zwei Magnete im Abstand von 180 Grad angebracht werden. Diese Entscheidung wurde getroffen, um die Energiegewinnung zu maximieren, es wurde jedoch ebenfalls entschieden, dass nicht mehr als zwei Magnete montiert werden sollen. Das Erscheinungsbild des Fahrrads würde damit sehr beeinträchtigt und die Bremswirkung würde mit vielen Magneten sicherlich zum Tragen kommen.

Alle Messungen werden mit einem Radumfang von 2.04 m ausgeführt, der Magnet befand sich 25 cm vom Zentrum der Radsimulation entfernt. Den Einfluss des Abstands kann aus dem Messprotokoll vom 06. Mai 2016 entnommen werden.

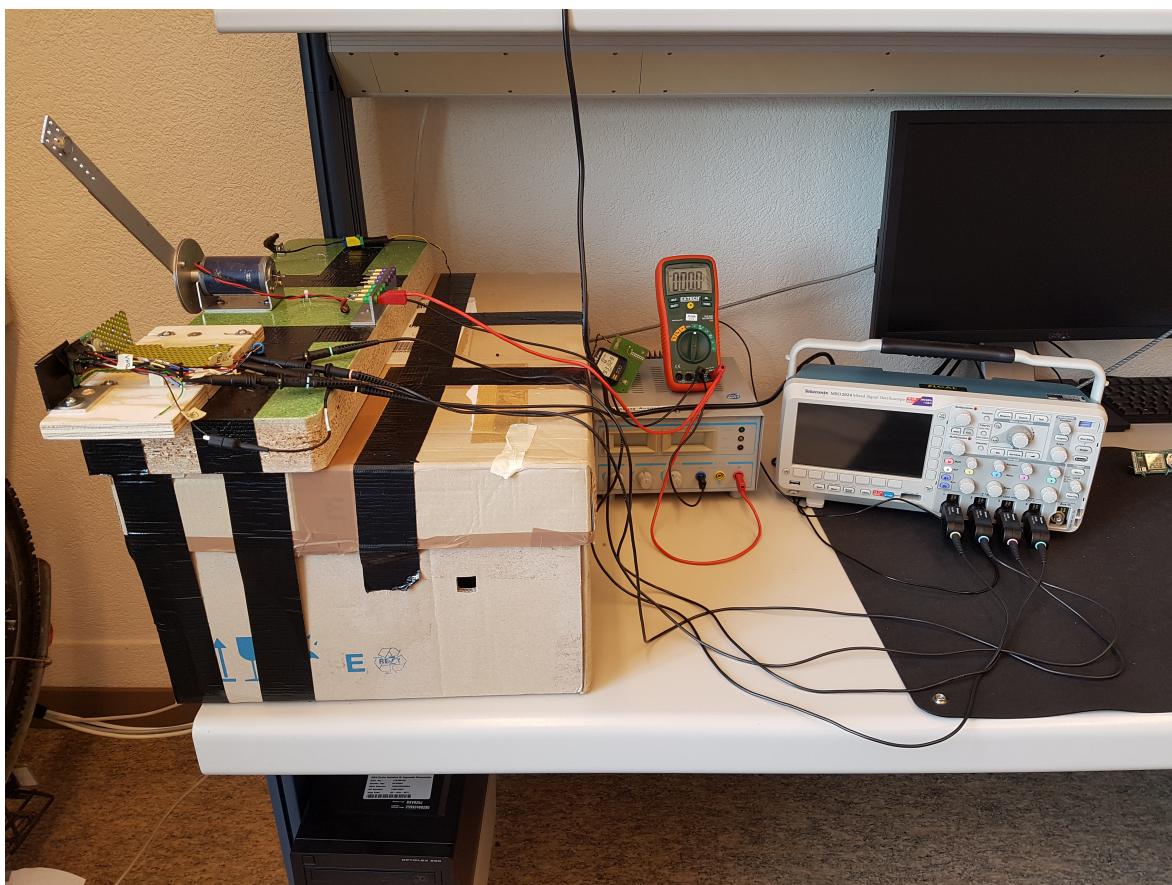


Abbildung E.1: Messaufbau während der Inbetriebnahme des Prototypen

Anhang F

Messprotokolle per Datum und Messobjekt

- 26. Februar 2016, *Harvesterausgang Elko*
Messprotokoll_Kondensator_Ausgang_Harvesterschaltung.pdf
- 14. März 2016, *Gleichrichter*
Messprotokoll_Optimierung_Gleichrichter_1.pdf
- 26. Februar 2016, *Harvesterausgang Elko*
Messprotokoll_Kondensator_Ausgang_Harvesterschaltung.pdf
- 14. März 2016, *Spule*
Messprotokoll_Optimierung_Spule.pdf
- 14. März 2016, *Gleichrichter*
Messprotokoll_Optimierung_Gleichrichter_1.pdf
- 14. März 2016, *Gleichrichter*
Messprotokoll_Optimierung_Gleichrichter_2.pdf
- 14. März 2016, *Spannungsbegrenzung*
Messprotokoll_Optimierung_Limiter.pdf
- 19. März 2016, *Harvesterausgang*
Messprotokoll_Leistungskennlinie_Harvester_versch_Harvester.pdf
- 30. März 2016, *EM8500-Chip-Ausgang*
Messprotokoll_Leistungskennlinie_Harvester_fliegender_Aufbau.pdf
- 14. April 2016, *Spule und Magnete*
Messprotokoll_Optimierung_Spule_versch_Spulen_und_Magnete.pdf
- 14. April 2016, *Harvesterausgang*
Messprotokoll_Leistungskennlinie_Harvester_Prototypenhardware.pdf
- 06. Mai 2016, *Harvesterausgang*
Messprotokoll_Leistungskennlinie_Position_Magnet.pdf
- 16. Mai 2016, *Prototyp*
Messprotokoll_Inbetriebnahme_Prototyp.pdf
- 16. Mai 2016, *Harvesterausgang*
Messprotokoll_Leistungskennlinie_Harvesterausgang_Prototyp.pdf

- 18. Mai 2016, *EM8500-Chip-Ausgang*
Messprotokoll_Energiemessung_EM-Chip_Inbetriebnahme.pdf
- 19. Mai 2016, *Prototyp*
Messprotokoll_Inbetriebnahme_Prototyp_Energiemanagment.pdf
- 21. Mai 2016, *Harvesterausgang Elko*
Messprotokoll_Harvesterausgang_Elko.pdf
- 21. Mai 2016, *Harvesterausgang*
Messprotokoll_Leistungskennlinie_Harvesterausgang_endgültige_Hardware.pdf
- 22. Mai 2016, *BLE-Kommunikation*
Messprotokoll_BLE_Kommunikation_Paketverlust.pdf
- 28. Mai 2016, *EM8500-Chip-Ausgang*
Messprotokoll_Energiemessung_EM-Ausgang_endgültige_Hardware.pdf
- 03. Juni 2016, *TI-SensorTag*
Messprotokoll_Energieverbrauch_Sensortag.pdf