Abschlussdokumentation Forschungsprojekt Sensornetze

Kay Förster

24. Februar 2015

Inhaltsverzeichnis

1 Projektmanagement

In diesem Kapitel sollen vor allem Arbeiten zusammengefasst werden, welche nicht primär mit der Entwicklung des Sensornetzes zu tun hatten, sondern eher als Projektmanagement / Qualitätsmanagement verstanden werden können. Diese Teilgebiete des software engineering werden meist etwas stiefmütterlich behandelt, da es für die meisten keinen sichtbaren Nutzen für ein Projekt bringt oder allgemein nur als Hindernis angesehen wird. Dabei beschäftigt sich das Gebiet unter anderem mit der Frage wie gewisse Abläufe verbessert und optimiert werden können.

1.1 Wissensdatenbank

Die Hauptaufgabe einer Wissensdatenbank ist, wie der Name bereits sagt, die Bereitstellung von Wissen für andere Personen. Als Wissen können dabei verschiedene Ideen, Problemlösungen, technische Dokumentationen und die Ergebnisse von Recherchen / Ausarbeitungen sein. Diese werden in aller Regel in schriftlicher Form digital gespeichert. Es ist dabei wichtig das dass gespeicherte Wissen schnell zu finden ist.

Für das Forschungsprojekt “Sensornetze” ist diese Wissensdatenbank dabei von zentraler Bedeutung, da im Gegensatz zu anderen Forschungsprojekten kontinuierlich an einem zentralem Thema gearbeitet wird. Ein weiterer wesentlicher Unterschied zu anderen Forschungsprojekten ist die Anzahl der Studenten die dieses Themengebiet bearbeiten. So ist dieses Modul sowohl im Bachelor / Diplom (1 Semester) als auch im Master (2 Semester) angeboten. Daraus resultiert eine hohe Anzahl von Studenten die sich in das Thema “Sensornetz” einarbeiten müssen.

1.1.1 Problemstellung

Seit Anfang 2012 kommt als Wissensdatenbank ein Wiki System zum Einsatz. Dabei handelt es sich speziell um das MediaWiki, welches vor allem durch die Nutzung von Wikipedia sehr bekannt ist.

In folgenden Semestern wurde das Wiki intensiv durch die Studenten genutzt. Mit der Zeit zeigte sich allerdings das Problem das im Wiki viele unterschiedliche Informationen eingetragen wurden, diese Informationen beim Nachschlagen nur schwer oder gar nicht fand. Vor allem Nutzer welche mit dem Wiki nicht regelmäßig arbeiteten, waren von diesem Problem betroffen.

Ein weiteres Problem, welche die Arbeit mit der Wissensdatenbank erschwerte, war die Tatsache das manche Informationen im Wiki verstreut, mehrfach in unterschiedlichen Artikeln beschrieben oder bereits veraltet waren. Des weiteren waren die Namen der Artikel meist zu lang und nicht aussagekräftig.

1.1.2 Lösungsmöglichkeiten

Die oben genannten Probleme haben dabei verschiedene Ursachen. Zum einen hat jeder Student ein anderes Verständnis dafür seine Artikel (bzw. seine Arbeit zu dokumentieren) zu Organisieren / Verwalten, was dadurch begünstigt wird das es keine genauen Vorgaben zur Erstellung von Artikeln gibt.

Ein weiter wichtiger Punkt ist, dass das Schreiben von Dokumentationen meist als unnötige Belastung wahrgenommen wird und somit meist sehr unregelmäßig erledigt wird. Mit dem Abschluss des Semestern verbleiben dann die unfertigen Artikel weiterhin im Wiki. Dies führte dazu das eine Vielzahl an Artikeln bzw. Abschnitte existierenden welche keinen Inhalt besaßen.

Im diesen Sinne soll auch erwähnt werden, das dass eingesetzte MediaWiki, nicht als klassische vollwertige Wissensdatenbank verstanden werden kann und somit nur einen begrenzten Funktionsumfang für diese Aufgabe besitzt.

Nützlichkeit

Im ersten Schritt, wurden die vorhanden Artikel nach ihrer Nützlichkeit für das Forschungsprojekt eingeteilt. Seiten mit keinen oder wenigen Inhalt wurden gelöscht bzw. auf andere Seiten verschoben.

Strukturierung

Im zweiten Schritt bzw. schon parallel zum ersten Schritt wurde eine mögliche neu Strukturierung des Wiki's erarbeitet. Im MediaWiki gibt es grundlegend folgende zwei Möglichkeiten Seiten zu strukturieren:

- Kategorien
- Unterseiten

Die meisten Seiten wurden durch die Studenten bereits kategorisiert. Diese Methode ist vor allem aus Wikipedia bekannt, sie besitzt allerdings den Nachteil das die Kategorie-Seiten, welche als Übersichtseite dann fungieren würden, bei der Suche nicht berücksichtigt werden.

Eine weitere Methode ein Wiki zu strukturieren ist die Verwendung von Unterseiten. Diese Unterseiten (engl. subpages) ist in der Wikipedia bei enzyklopädische Artikeln nicht erlaubt, da dies zu einen zu starken Hierarchiebaum führen würde. Die Entstehung eines Hierarchiebaum ist dagegen in der Wissensdatenbank erwünscht.

Unterseiten werden dabei einfach nur durch einen Slash im Namen von ihrer Hauptseite abgetrennt, siehe dazu das Beispiel im Listing. 1.1.

```
1 Hauptseite -> Hauptseite
2 Hauptseite/Subpage1 -> Subpage 1
3 Hauptseite/Subpage2 -> Subpage 2
4 Hauptseite/Subpage2/Subsubpahe -> Unterseite von Subpage 2
```

Listing 1.1: Einteilung in Subpages

Dadurch ist auch immer erkennbar zu welchen Themengebiet eine Seite gehört. Damit die entsprechenden Unterseiten auch auf den "Hauptseiten" angezeigt werden, wurde durch

Herrn Paul das Addon “SubPageList” installiert. Anschließend muss lediglich folgende Zeile in der Hauptseite eingefügt werden:

```
1   {{Vorlage:Subpages}}
```

Um diese Strukturierung zu erreichen mussten die meisten Artikel umbenannt oder verschoben werden. Neben der Strukturierung entstanden so auch eindeutig kurze zuordenbare Seitennamen.

1.1.3 Ausblick

Mit den hier kurz vorgestellten Maßnahmen konnte die Wissensdatenbank wieder etwas strukturiert und geordnet und ggf. aktualisiert werden. Es gibt allerdings noch Teilgebiete welche aufgrund von fehlenden Fachwissen nicht aktualisiert werden konnte. Hier sollte es in Zukunft immer einen Verantwortlichen aus dem jeweiligen Fachgebiet geben der die Seiten aktuell hält.

Das sich die Wissensdatenbank als Nachschlagewerk und Dokumentation der Arbeit bewährt hat, konnte man sehr gut im laufe des Semesters beobachteten. Allerdings sollte weiterhin darauf geachtet werden, dass die enthaltenen Informationen korrekt und aktuell sind. Auch wenn dies eine erhöhte Bearbeitungszeit benötigt erleichtert dies die Arbeit bei der Fehlersuche bzw. Einarbeitung in das Themas

1.2 Projekt-Repository

Ein weitere Aufgabe die in das Teilgebiet “Projektmanagement” zugeordnet werden kann, war die Aufgabe das interne Mercurial-Repository mit Contiki OS auf ein öffentliches Git-Repository zu verlagern. Dies hatte vor allem folgende Gründe:

- leichte Übernahme von Patches von Contiki OS oder anderen Repositorys
- Bessere Nachvollziehbarkeit von Änderungen (Webinterface, Logs, Commits)
- Erreichbarkeit von Außerhalb sowie Nutzung von aktuellen Techniken

2 Schnittstelle FHEM - CoAP

Für die automatisierte Bedienung und Steuerung der verschiedenen Sensoren / Aktoren im Sensornetz kommt das freie Serverprogramm FHEM zum Einsatz. Dieses besitzt allerdings noch keine Möglichkeit um auf ein 802.15.4-Sensornetz per CoAP zuzugreifen zu können. Aus diesen Grund wurde von unseren Vorsemestern ein FHEM-Modul sowie ein CoAP-Wrapper entwickelt, welche den Zugriff von FHEM auf verschiedene CoAP-Ressourcen ermöglicht.

Das FHEM-Modul wandelt die entsprechenden Abfragen für einen Sensor / Aktor in ein definiertes Datenaustauschformat um. Dieses wird dann an den CoAP-Wrapper geschickt, welcher eigenständig läuft und die Anfrage in einen CoAP-Request umwandelt und anschließend in das Sensornetz weiterleitet. Für eine genauere Beschreibung der Module sei hier auf die entsprechende Abschlussdokumentation verwiesen.

2.1 Problemstellung

Für die effiziente periodische Abfrage von Sensorwerten wurde in CoAP der sogenannt Observe-Befehl implementiert. Wird dieser Befehl an einen Sensor geschickt, sendet dieser anschließend regelmäßig und eigenständig eine Benachrichtigung (Wert) der jeweiligen zu beobachteten Ressource an den Anfragesteller. Diese muss anschließend den empfangen Wert bestätigen.

Von den jeweiligen Vorsemestern wurde diese Funktionalität bereits angefangen in den CoAP-Wrapper zu implementieren, allerdings konnte diese nicht fertiggestellt und getestet werden. Für das geplante Testnetz sollte diese Funktionalität vollständig bereitstehen.

2.2 TCP-Server

Unter dem Begriff TCP-Server wird nachfolgend der CoAP-Wrapper verstanden, welcher das definierte Datenaustauschformat vom FHEM-Modul in einen CoAP-Request umwandelt.

Für das Versenden der CoAP-Requests wird das Californium Framework genutzt. Californium ist eine Open-Source-Implementierung des CoAP Protokolls, welches direkt von der ETH Zürich entwickelt wird. Es kann als Art Referenzimplementierung angesehen werden. Aufbauend auf dieser Referenzimplementierung beziehungsweise den vorhandenem Beispiel (`ExampleClient.java`) wurde der TCP-Server entwickelt.

Da es sich bei CoAP zum aktuellen Zeitpunkt um noch keinen festen Standard handelt, werden noch regelmäßig Änderungen vorgenommen. So wurde auch seit dem letzten Erstellen des TCP-Server Californium auf einen neuen Stand gebracht. Dies hatte zur Folge das beim erstellen der Entwicklungsumgebung Californium nicht einfach eingebunden werden konnte sondern vorher die entsprechenden Quelltexte angepasst werden mussten. Erst nach Übernahme dieser Änderungen konnte mit der eigentlichen Anpassungen begonnen werden.

2.2.1 Implementierung Observe

Durch die vorherigen Semestern wurde bereits die Grundlagen für die Observe Funktion eines Sensors gelegt, allerdings waren diese nicht funktionstüchtig / richtig implementiert. So wurde der Observe Befehl richtig an den Sensor-Knoten weitergeleitet, allerdings wurden anschließend nicht auf weitere Pakete des Sensors gewartet. Aus diesen Grund konnten keine weiteren Pakete am TCP-Server empfangen werden und der Sensor entfernte den jeweiligen Observe Eintrag aus seiner Tabelle.

Der Californium-Client verbleibt bei einem Observe-Befehl in einer Endlosschleife in welcher er auf einen Response wartet und somit die Verbindung "gehalten" wurde. Um diese Verhalten auch beim TCP-Server zu erreichen wurde der TCP-Server so umgeschrieben, dass jeder Request in einem neuen Thread abgearbeitet wird. Somit kann für einen Observe-Befehl der Client in einer Endlosschleife verbleiben und die eingehenden Werte an das FHEM-Modul weiterleiten, während dessen andere Anfragen parallel in

ihrem jeweiligen Thread abgearbeitet werden. Dies hat den entscheidenden Vorteil das bei einem möglichen Timeout bzw. einem Observe nicht die ganze Schnittstelle zum erliegen kommt.

Um vorhandene Observe-Threads auch beenden zu können, erhält jeder Thread einen eindeutigen Namen bestehend aus jeweiligen `Methode` (DISCOVER, GET, PUT oder OBSERVE) und der `URL`. Die URL enthält neben der Adresse auch die angesprochene Ressource, womit die Zuordnung eindeutig wird. Wird nun durch das FHEM-Modul ein entsprechender DEOBSERVE Befehl für eine URL gesendet, so sucht der TCP-Server nach dem Thread-Namen (DISCOVER + URL) und beendet diesen. An den Sensor muss dabei kein gesonderter Befehl gesendet werden.

2.2.2 weitere Änderungen

Während der Implementierung des Observe-Funktionalität sind noch einige weitere kleine Probleme mit dem TCP-Server aufgefallen welche ebenfalls mit behoben wurden.

Dies betraff vorwiegend den Fall das FHEM beendet bzw. neu gestartet wurde. Der TCP-Server versuchte daraufhin ein neues TCP-Socket anzulegen, auf welches FHEM zugreifen kann. Allerdings ergaben sich dabei die folgenden zwei nennenswerte Probleme:

- Die Erstellung des TCP-Socket schlug jedes mal fehl. Die Ursache lag darin, das dass vorhergehende Socket durch die ausgelöste Exception nicht geschlossen wurde und dadurch das anlegen eines neuen Socket auf den gleichen Port nicht möglich war.

- Erhöhte Last durch das erzeugen der Sockets. Das sich das Programm, nach dem obigen beschrieben Szenario, nicht einfach terminiert, wurde das Programm von einer Endlosschleife umgeben. Im Fehlerfall führte diese einfache Konstruktion dazu das diese Schleife sehr häufig durchlaufen wird, was zu der herhöhten Last führte. Dies wird nun durch ein `Sleep` Befehl verhindert.

2.3 CoAP-Client

Wie bereits erwähnt, wurde eine neue Version der Californium-Bibliothek veröffentlicht. Aus diesen Grund wurde auch eine neue Version des CoAP-Clients kompiliert. Dieser wird über die folgende Syntax aufgerufen:

```
1 java -jar SampleClient.jar [-l] METHOD URI [PAYLOAD]
```

Listing 2.1: Californium CoAP-Client Syntax

```
1 java -jar CaliforniumClient.jar GET coap://[2002:8d38:831a:↩
    aaaa:21:2eff:ff00:3864]/sensors/intern_tmp
```

Listing 2.2: Californium CoAP-Client Beispiel-Abfrage eines Sensors

Dabei wurde auch festgestellt das es in letzter Zeit zu einer Verwechselung mit dem Programm `FHEMCoAPClient.jar` gekommen ist. Welches keine direkte CoAP-Syntax besitzt sondern nur dazu gedacht war, das Datenaustauschformat vom FHEM-Modul zu testen.

3 Analyse Stromverbrauch

Für ein Sensornetz ist der Stromverbrauch in den einzelnen Knoten (Sensoren) ein wesentlicher Parameter. So entscheidet der Stromverbrauch im wesentlichen über:

Akzeptanz des Users Wird vom Benutzer ein hoher administrativer Aufwand gefordert, wie zum Beispiel ein häufiger Wechsel der Batterien, sinkt beim Nutzer in der Regal die Bereitschaft das System zu nutzen.

Lebensdauer eines Sensors Je weniger Strom ein Knoten für seine Aufgabe benötigt desto länger kann er seiner Aufgabe im System nachgehen. Die Lebensdauer der einzelnen Knoten bestimmen daher auch die wesentliche Lebensdauer des Gesamtsystems.

mögliche Funktionalität des Systems Besitzt ein Sensorknoten eine starke Energie-Versorgung kann er weitere Aufgaben / Funktionen im System übernehmen. Als Beispiel sei hier der Rauchmelder der Firma Nest erwähnt, welcher zusätzlich zu seiner Aufgabe als Nachtlicht verwendet werden kann.

Eine genaue Berechnung des Stromverbrauchs ist nur schwer durchführbar, da die angegebenen Werte im Datenblatt nur theoretische maximal Werte des Herstellers sind, welche unter definierten Bedienungen ermittelt wurden. Weiterhin können wir keine genaue Aussagen treffen in welchen Zustand (Idle, Sleep, Senden / Empfangen) sich der Sensorknoten befindend. Dies wäre aber für eine einigermaßen Aussagekräftige Berechnung erforderlich.

Aus diesen Gründen soll der Stromverbrauch mittels eines Langzeit-Praxistest ermittelt werden. Neben den eigentlichen Stromverbrauch sollen im Anschluss auch ein Vergleich zu bereits existierenden Hausautomatisierungslösungen gezogen werden.

3.1 Monitoring der CoAP Ressource

Um den Stromverbrauch möglichst einfach zu erfassen wurde durch Gruppe 1 eine Möglichkeit geschaffen die Batteriespannung direkt mit dem Sensorknoten zu messen. Diese Messmethode hat eine ungefähre Hardware bedingte Toleranz von 0,05 V, welche für unsere Auswertung allerdings ausreichend ist.

Zur einfachen Abfrage der Messergebnisse wurden diese als CoAP Ressource bereitgestellt.

3.1.1 Einbindung in FHEM

Die Abfrage und Auswertung des Sensors sollte am Anfang direkt über die in FHEM integrierten Funktionen geschehen. Allerdings wurden zum Beginn des Tests noch durch andere Gruppen verschiedene Arbeiten an FHEM und dem TCP-Server vorgenommen, sodass eine sichere periodische Abfrage nicht gewährleistet werden konnte.

Die Nutzung der "observe"-Funktion konnte auch von vornherein ausgeschlossen werden, da diese aller 60 Sekunden einen Wert liefert. Für diesen Langzeittest ist eine solche hohe Abfragerate nicht notwendig und hätte eher die Messergebnisse verfälscht.

Zur einfachen Abfrage der aktuellen Spannung wurde der Sensor mit dem der entsprechenden Ressource in FHEM eingefügt. Eine automatische Aktualisierung wurde wegen der obrig genannten Gründe nicht eingestellt.

```
1 define BatSensor WSN [2002:8d38:831a:aaaa:21:2eff:ff00↩
    :3864]:5683/sensors/battery
2 attr BatSensor model temperature
3 attr BatSensor observable true
4 attr BatSensor room S311
5 attr BatSensor unit Voltage~V
```

Listing 3.1: Definition Sensor zur Abfrage der Batteriespannung

3.1.2 Realisierung als Cronjob

Da, wie bereits beschrieben, die Abfrage per FHEM nicht möglich war, wurde ein Bash-Script geschrieben welches den aktuellen Wert vom Sensor abfragt. Dazu nutzt das Script die Java Datei `FHEMCoAPClient.java` welche eine CoAP Anfrage an den, im Script definierten, Sensor schickt. Anschließend wird das Ergebnis geparst und in eine Textdatei geschrieben.

Das Script wird durch einen Cronjob (siehe 3.3) zu jeder vollen Stunde ausgeführt.

```
1  #/bin/bash
2  SENOR_IP="2002:8d38:831a:aaaa:21:2eff:ff00:3864"
3  SENOR_RESSOURCE="/sensors/battery"
4  FILE="/home/student/alex/values_3864"
5  CoAP_Client="/opt/fhem/WSN/FHEMCoAPClient.jar"
6
7  # Get the Result from the Sensor
8  RESULT=$(java -jar $CoAP_Client "get|[$SENOR_IP]:5683↩
       $SENOR_RESSOURCE" 2>/dev/null)
9
10 # Error?
11 echo $RESULT | grep -q "ERROR"
12 if [ $? -eq 0 ]
13 then
14         echo "[ERROR] Fehler bei der Abfrage"
15         exit -1
16 fi
17
18 # grep the value
19 VALUE=$(echo $RESULT | sed -n -e 's/.* \([0-9]*\.[0-9]*\) ↩
       \[.*\].*/\1/p')
20
21 # Save to file
22 date +"%Y-%m-%d %H:%M:%S ; $VALUE" >> $FILE
```

Listing 3.2: Definition Sensor zur Abfrage der Batteriespannung

```
1 # m h  dom mon dow   command
2 0 * * * * /home/student/alex/store_value.sh
```

Listing 3.3: Cronjob zum starten der Abfrage

Für die Auswertung der aufgezeichneten Werte wurde eine gnuplot-Konfiguration angelegt, welche die Werte in Abhängigkeit der Zeit grafisch darstellt.

```
1  #!/usr/bin/gnuplot
2
3  set xdata time
4  set timefmt '%Y-%m-%d %H:%M:%S'
5  set datafile sep ";"
6
7  set ylabel "Spannung"
8  set xlabel "Datum/Zeit"
9  set format x '%d.%m'
10 set autoscale x
11
12 set terminal svg size 1024,768 fname 'Verdana' fsize 10
13 set output 'output.svg'
14
15 plot 'values' using 1:2 title "Battery [ff00:225e]" with ↩
     lines
```

Listing 3.4: gnuplot Einstellungen für die grafische Auswertung

3.2 Auswertung

Die Daten für diese Auswertung wurden von dem Sensor `aaaa:21:2eff:ff00:3864` ermittelt. Der Sensor befand sich im Messzeitraum im Sensornetz-Schrank in der S311 auf der untersten Etage, der Borderrouter stand in der S311a am Server iblis.

3.2.1 Erwarteter Verlauf

Für den Test wurden spezielle Ni-MH-Akkus der Firma eneloop eingesetzt. Diese besitzen eine sehr geringe Selbstentladung (70% nach 5 Jahren) und sind speziell für den Einsatz in Geräten mit geringem Stromverbrauch und langen Bereitschaftszeiten vorgesehen. Die Akkus besitzen eine mindest Kapazität von 1900 mAh und eine Nennspannung von 1,2 V.

In der Abbildung 3.1 ist die Entladekurve beispielhaft für einen eneloop Ni-MH-Akkus mit 1400 mAh gezeigt, der von uns verwendete Akku verhält sich ähnlich. Es ist in der Abbildung sehr gut die relativ gleichbleibende Spannung während des Entladens zu sehen. Erst wenn die gesamte Kapazität des Akkus aufgebraucht ist, fällt sie Spannung rapide ab.

Dieses Verhalten ist für Verwendung als Energiequelle für einen Mikrocontroller gut, allerdings lässt anhand dieser Kurve die aktuelle Kapazität und somit eine Vorhersage der Lebensdauer des Akkus nur sehr schlecht bestimmen.

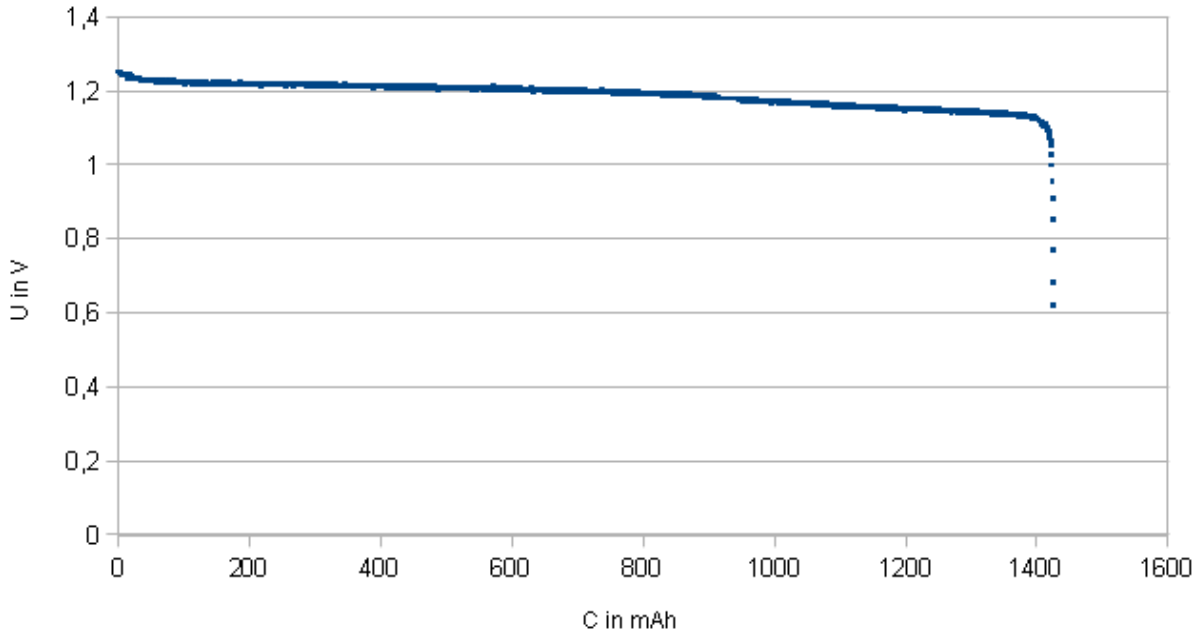

Abbildung 3.1: Entladekurve für eneloop Ni-MH-Akkus mit 1400 mAh

3.2.2 Auswertung Test 1

Ein erste Langzeittest wurde am 11.11.2014 um 12:00 Uhr gestartet. Vom 24.11.2014 bis zum 05.12.2014 konnten keine Daten des Sensors aufgezeichnet werden, da in diesem Zeitraum die Schnittstelle (tunslib) zum Sensornetz nicht erreichbar war.

Am 05.01.2015 wurde festgestellt das seit dem 21.12.2014 15:00 Uhr der Sensor mit einer Batteriespannung von 2,97 V nicht mehr reagiert. Da der Mikrocontroller für den Betrieb mindestens 3,0 V Spannung benötigt, können wir davon ausgehen das ein längere Betrieb nicht möglich war. Somit lief der Sensor 40 Tage ohne Unterbrechung.

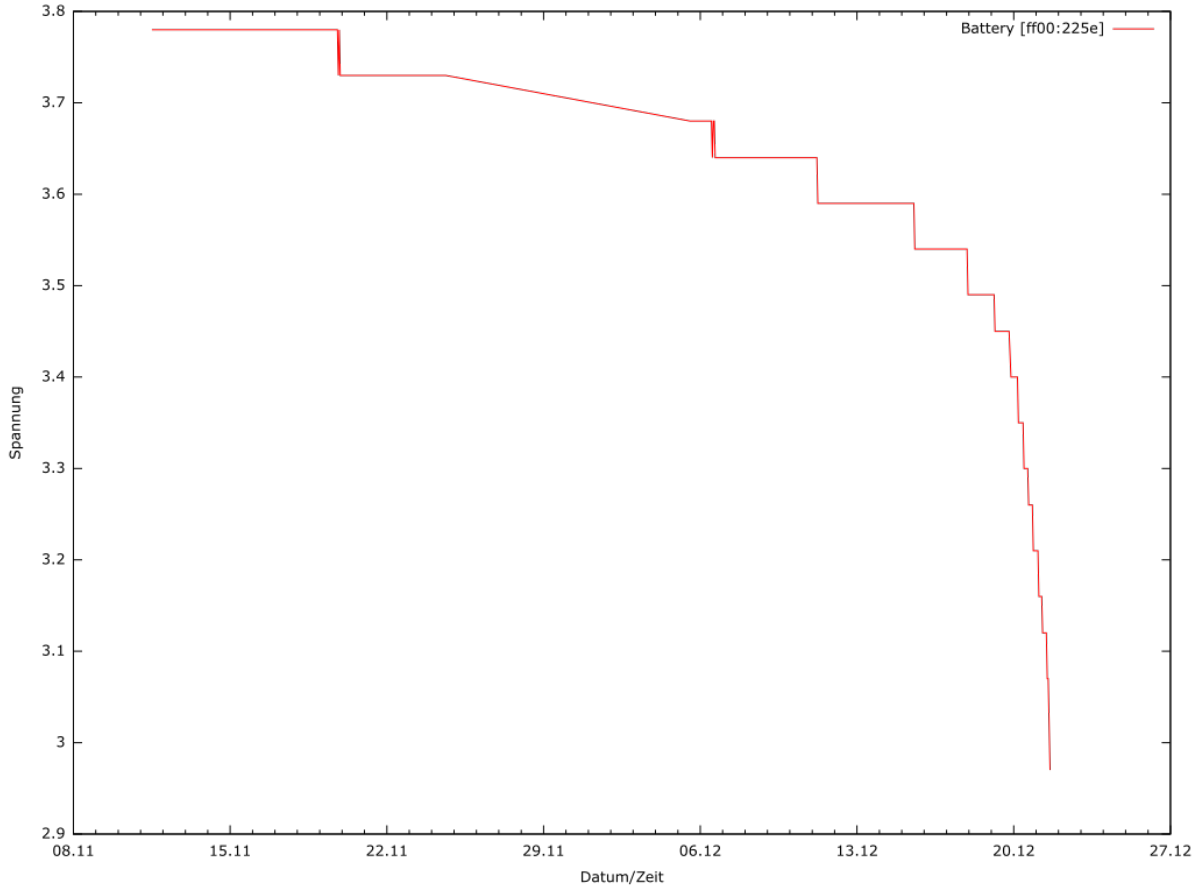

Abbildung 3.2: Grafische Darstellung der Spannung von Test 1

Trotz der geringen Laufzeit von 40 Tagen zeigt der grafische Verlauf grob den erwarteten Spannungsverlauf des Akkus. Nach genauer Betrachtung der Daten und Auswertung eines anderen Tests stellten wir fest, dass am 14.12.2014 ein Ping-Dauertest mit 1 Ping

pro Sekunde fälschlicherweise auf diesen Sensorknoten gestartet wurde. Durch diese hohe Belastung wurden die Akkus innerhalb von 10 Tagen vollkommen aufgebraucht. Der Test ist somit für eine genaue Auswertung nicht mehr brauchbar.

3.2.3 aktueller Verlauf

Am 12.01.2015 um 14:00 Uhr wurde ein weiterer Langzeittest gestartet, der zum aktuellen Zeitpunkt noch weiter lief. Aus diesem Grund sollen hier jetzt bloß die ersten Erkenntnisse und Vermutungen geäußert werden.

Der Test startete mit einer etwas höheren Batteriespannung (4,16 V / 3,78 V) als der erste Test. Wir vermuten das, bis die mindest Nennspannung erreicht ist, die Spannung weiter schnell abfällt und erst dann für längere Zeit schwach abfällt.

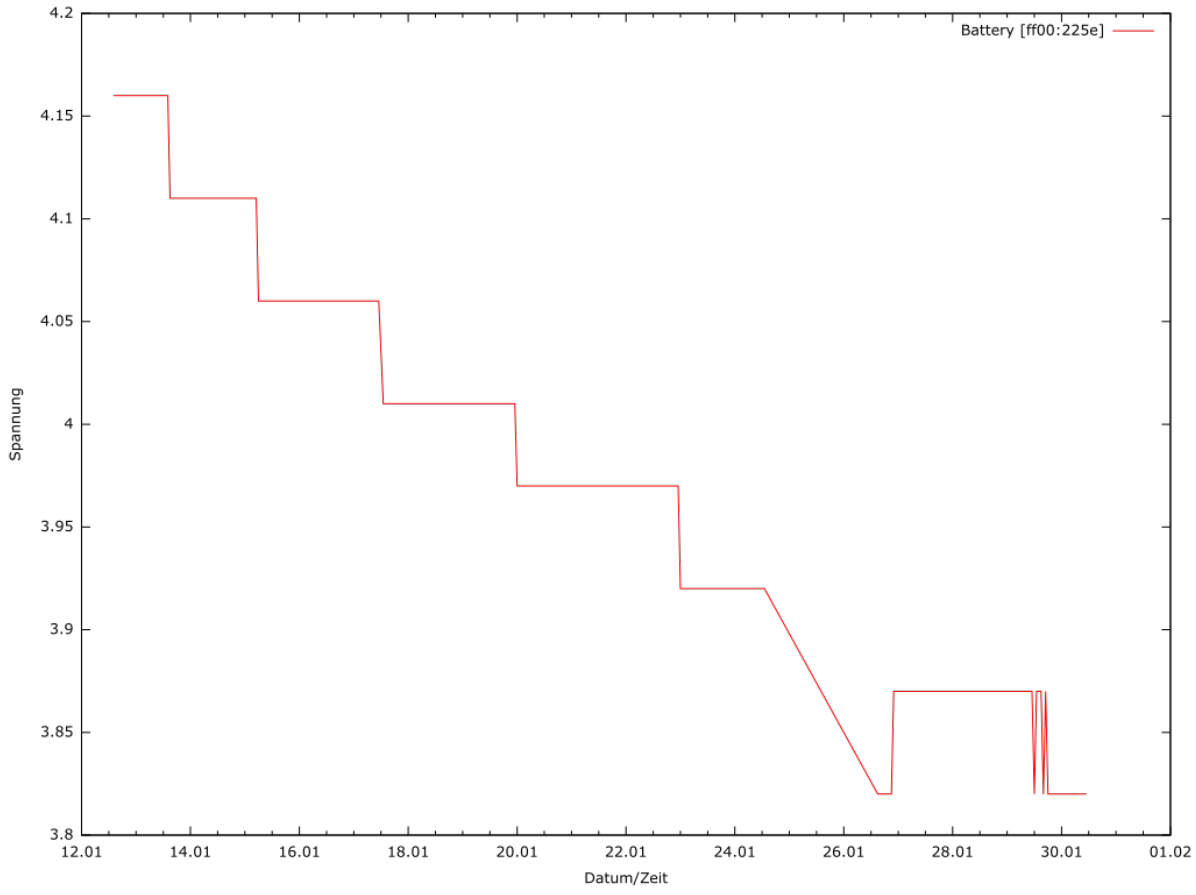

Abbildung 3.3: Grafische Darstellung der Spannung von Test 2

Der größte Unterschied zum ersten Test ist, dass der Sensor innerhalb von 19 Tage schon zweimal (24.01.2014 bis 26.01.2014 und ab dem 30.01.2014) komplett ausgefallen ist. Bei diesem “Komplett”-Ausfall war es nicht möglich den Sensorknoten durch einen Soft bzw. Hardreset zu reaktivieren, sondern der Sensor musste durch die Gruppe 1 komplett neu geflasht werden.

3.3 Ausblick

Aufgrund der Tatsache das die Langzeittests noch nicht erfolgreich abgeschlossen bzw. lange genug liefen ist es schwer verlässliche Aussagen über die Batterielebensdauer zu treffen und somit einen Ausblick zu wagen. Die bisherigen Tests, vor allem Test 2, haben gezeigt das es vor allem bei der Hardware noch einige unbekannte Probleme gibt die es zu lösen gilt.

Um weitere Erkenntnisse, auch unabhängig von der Messung der Batteriespannung, zu gewinnen sollte ein solcher Langzeittest weiter betrieben werden.