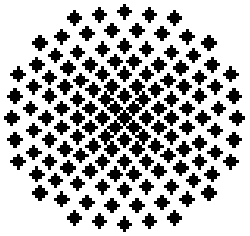


Universität Stuttgart

Fakultät Informatik, Elektrotechnik und Informationstechnik

Diplomarbeit Nr. 2487

Gruppierung von Sensorknoten auf Basis von Sensordaten und Konnektivität

Andreas Heydlauff

Studiengang:	Informatik
Prüfer:	Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Betreuer:	Dipl.-Inf. Matthias Gauger
begonnen am:	18. 04. 2006
beendet am:	18. 10. 2006
CR-Klassifikation:	C.2.4, C.3, I.5.3

Institut für Parallele und
Verteilte Systeme
Abteilung Verteilte Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Zusammenfassung

Sensornetze bestehen aus vielen Sensorknoten, die mit Sensoren die Umwelt aufzeichnen und mit einem speziellen Mikroprozessor verarbeiten. Im Normalfall besitzen sie eine geringe Menge von Arbeits- und Programmspeicher, einer Möglichkeit zur drahtlosen Kommunikation untereinander und Batterien zur Stromversorgung. Oft ist es sinnvoll mehrere solcher Sensorknoten in Gruppen zusammenzufassen, um deren Fähigkeiten zu kombinieren. Ziel dieser Arbeit ist eine Gruppeneinteilung der Sensorknoten, die auf Gemeinsamkeiten in den gemessenen Sensordaten basiert. Die so gewonnenen Gruppen reflektieren eine Verteilung der Knoten in geographisch begrenzten Regionen.

Es werden sowohl bestehende Ansätze untersucht als auch neue Methoden erarbeitet. Eine Evaluation der Methoden erfolgt durch Messreihen, die ebenfalls im Rahmen dieser Arbeit erfasst werden. Am Ende resultiert daraus eine einsatzfähige prototypische Implementierung.

Inhaltsverzeichnis

Inhaltsverzeichnis ii

Abbildungsverzeichnis iv

Tabellenverzeichnis vii

1 Einführung 1
- 1.1 Motivation . . . 1
  - 1.1.1 Lokalisierung . . . 1
  - 1.1.2 Clustering . . . 2
  - 1.1.3 Gruppierung auf Basis von Sensordaten . . . 2
- 1.2 Problemstellung . . . 3
- 1.3 Gliederung . . . 3

2 Grundlagen 5
- 2.1 Sensornetze . . . 5
- 2.2 TMote Sky . . . 5
- 2.3 Mica2Dot . . . 6
- 2.4 TinyOS . . . 7

3 Verwandte Arbeiten 10
- 3.1 Clusteringalgorithmen . . . 10
- 3.2 Lokalisierungsalgorithmen . . . 11
- 3.3 Einordnung . . . 13

4 Konzeption 15
- 4.1 Definition . . . 15
  - 4.1.1 Systemmodell . . . 15
  - 4.1.2 Terminologie . . . 16
- 4.2 Analyseverfahren . . . 16
  - 4.2.1 Verarbeitungsstufen . . . 16
  - 4.2.2 Stufe 1: Signalverarbeitung . . . 18
  - 4.2.3 Stufe 2: Vergleich . . . 26
  - 4.2.4 Signalstärke . . . 34
  - 4.2.5 Stufe 3: Gruppenbildung . . . 35
  - 4.2.6 Stufe 4: Gesamtergebnisse . . . 37
- 4.3 Verfahren aus der Biologie . . . 40

5 Analyse **43**
5.1 Datenerfassung 43
5.2 Intels Berkeley-Forschungslabor 45
5.3 Auswertung 47
5.3.1 Stufe 1: Signalverarbeitung 47
5.3.2 Stufe 2: Vergleich 65
5.3.3 Signalstärke 81
5.3.4 Stufe 3: Gruppenbildung 83
5.3.5 Stufe 4: Gesamtergebnis 86
5.3.6 Verfahren aus der Biologie 90
5.3.7 Daten aus Intels Berkeley-Forschungslabor 94
5.4 Analyseergebnis 98

6 Prototyp **100**
6.1 Entwurf 100
6.2 Implementierung 105
6.3 Ergebnis 107

7 Ausblick **109**

8 Zusammenfassung und Fazit **111**

Literaturverzeichnis **112**

Abbildungsverzeichnis

2.1 TMote Sky [Mot] 6
2.2 Mica2Dot [Cro] 7

4.1 Verarbeitungsstufen 17
4.2 Beispiel für die Signalverarbeitung 19
4.3 Beispiel für den Filter „Änderung" 19
4.4 Beispiel für den Filter „prozentuale Änderung 20
4.5 Beispiel für den Filter „Betrag" 21
4.6 Beispiel für den Filter „Mittelwert" 21
4.7 Beispiel für den Filter „Integral" 22
4.8 Beispiel für den Filter „Normierung" 23
4.9 Beispiel für den Filter „Fourier-Transformation" 24
4.10 Beispiele für den Filter „Bandpass" 24
4.11 Beispiel für die Erkennung absoluter Ereignisse 25
4.12 Beispiel für die Erkennung prozentualer Ereignisse 26
4.13 Beispiele für den Komparator „Differenz" 27
4.14 Beispiel für den Komparator „Produkt" 28
4.15 Beispiel für den Komparator „Prozent" 28
4.16 Beispiel für Komparator „Kohärenz" 29
4.17 Beispiele für den Komparator „übereinstimmende Ereignisse" 30
4.18 Beispiele für den Komparator „verpasste Ereignisse" 31
4.19 Beispiel für den Komparator „prozentual übereinstimmende Ereignisse" auf Mote 07 32
4.20 Beispiel für den Modifikator „Betrag" auf Mote 07 32
4.21 Beispiel für den Modifikator „Quadrat" auf Mote 07 33
4.22 Beispiel für den Modifikator „Integral" auf Mote 05 34

5.1 Moteverteilung in der Fakultät für Informatik 45
5.2 Moteverteilung im Intel-Berkeley-Forschungslabor [Int] 46
5.3 Sensordaten des Luftfeuchtigkeitssensors 47
5.4 Sensordaten des Temperatursensors 48
5.5 Sensordaten des Lichtsensors PAR 48
5.6 Sensordaten des Lichtsensors TSR 49
5.7 Änderung der Luftfeuchtigkeit 50
5.8 Änderung der Temperatur 51
5.9 Änderung des Lichts PAR 51
5.10 Änderung des Lichts TSR 52

5.11 Mittelwert der Luftfeuchtigkeitssensordaten 53
5.12 Mittelwert der Temperatursensordaten 53
5.13 Änderungen in der Luftfeuchtigkeit 54
5.14 Änderungen in der Temperatur 54
5.15 Anhäufung von Änderungen in der Luftfeuchtigkeit 55
5.16 Anhäufung von Änderungen in der Temperatur 55
5.17 Betrag der Änderung 56
5.18 Ereignisse in der Luftfeuchtigkeit 56
5.19 Ereignisse in der Temperatur 57
5.20 Normierte Lichtsensordaten 58
5.21 Aufsummierung der Änderung der unnormierten Lichtdaten TSR 58
5.22 Aufsummierung der Änderung der normierten Lichtdaten TSR 59
5.23 Prozentuale Änderung des Lichts PAR 59
5.24 Prozentuale Änderung des Lichts TSR 60
5.25 Aufsummierung der prozentualen Änderung des Lichts TSR 60
5.26 Ereignisse basierend auf der prozentualen Änderung des Lichts PAR . . . 61
5.27 Ereignisse basierend auf der prozentualen Änderung des Lichts TSR . . . 61
5.28 Fouriertransformation 62
5.29 Rücktransformation der Luftfeuchtigkeit 63
5.30 Rücktransformation der Temperatur 63
5.31 Rücktransformation des Lichts PAR 64
5.32 Rücktransformation des Lichts TSR 64
5.33 Integral der Sensordaten 65
5.34 Differenz der Luftfeuchtigkeitssensordaten von Mote 15 66
5.35 Differenz der Temperatursensordaten von Mote 15 66
5.36 Differenz der Lichtsensordaten PAR von Mote 15 67
5.37 Differenz der Lichtsensordaten TSR von Mote 15 67
5.38 Betrag der Differenz der Luftfeuchtigkeitssensordaten von Mote 15 68
5.39 Betrag der Differenz der Temperatursensordaten von Mote 15 68
5.40 Quadrat der Differenz der Luftfeuchtigkeitssensordaten von Mote 15 . . . 69
5.41 Quadrat der Differenz der Temperatursensordaten von Mote 15 69
5.42 Aufsummierte absolute Differenz der Luftfeuchtigkeitssensordaten von Mote 15 70
5.43 Aufsummierte absolute Differenz der Temperaturdaten von Mote 15 . . . 70
5.44 Prozentuale Unterschiede der Luftfeuchtigkeitsdaten von Mote 15 71
5.45 Prozentuale Unterschiede der Temperaturdaten von Mote 15 71
5.46 Prozentuale Unterschiede der Lichtsensordaten PAR von Mote 15 72
5.47 Prozentuale Unterschiede der Lichtsensordaten TSR von Mote 15 72
5.48 Kohärenz der Sensordaten von Mote 15 73
5.49 Aufsummiertes Produkt der Tendenzen der Sensordaten von Mote 15 . . 74
5.50 Übereinstimmende Ereignisse in der Luftfeuchtigkeit von Mote 15 75
5.51 Übereinstimmende Ereignisse in der Temperatur von Mote 15 75
5.52 Übereinstimmende Ereignisse im Lichtbereich PAR von Mote 15 76
5.53 Übereinstimmende Ereignisse im Lichtbereich TSR von Mote 15 76
5.54 Verpasste Ereignisse in der Luftfeuchtigkeit von Mote 15 77
5.55 Verpasste Ereignisse in der Temperatur von Mote 15 77

5.56 Verpasste Ereignisse im Lichtbereich PAR von Mote 15 78
5.57 Verpasste Ereignisse im Lichtbereich TSR von Mote 15 78
5.58 Prozentual übereinstimmende Ereignisse von Mote 15 79
5.59 Aufsummierte gewichtete Ereignisse der Luftfeuchtigkeit von Mote 15 . . 80
5.60 Aufsummierte gewichtete Ereignisse der Temperatur von Mote 15 80
5.61 Aufsummierte gewichtete Ereignisse im Lichtbereich PAR von Mote 15 . . 81
5.62 Aufsummierte gewichtete Ereignisse im Lichtbereich TSR von Mote 15 . . 81
5.63 Statistische Verteilung der Signalstärke von Mote 15 empfangener Nachrichten 82
5.64 Aufsummierte gewichte Signalstärken von Mote 15 empfangener Nachrichten 82
5.65 Adjazenzfehler der Lösungsverfahren 89
5.66 Adjazenzfehler der Lösungsverfahren mit prozentualen Daten 90
5.67 Fehler in der Adjazenzmatrix bei der äußeren Erweiterung 94
5.68 Sensordaten des Luftfeuchtigkeitssensors 95
5.69 Sensordaten des Temperatursensor 95
5.70 Sensordaten des Lichtsensors 96

6.1 Entwurf der prototypischen Implementierung 101

Tabellenverzeichnis

4.1 Beispiel für die Entscheidungsfindung eines Separators 35
4.2 Ergebnis des Beispiels für die Separatoren „Minimum-Maximum" und „Mittelwert" 36
4.3 Ergebnis des Beispiels für den Separator „Prozent" in Prozent 37
4.4 Beispiel für die allgemeine Vorgehensweise zur Lösungsbildung 38
4.5 Beispiel für eine Gruppenkonfiguration nach der Maximumsuche 40

5.2 Gruppenkonfiguration von Mote 15 durch Separator „Minimum-Maximum" 83
5.1 Ergebnisse der Kanäle von Mote 15 84
5.3 Gruppenkonfiguration von Mote 15 durch Separator „Mittelwert" 85
5.4 Gruppenkonfiguration von Mote 15 durch Separator „Positiv" 85
5.5 Gruppenkonfiguration in Prozent Zugehörigkeit von Mote 15 durch Separator „Prozent" 86
5.6 Allgemeine Vorgehensweise bei Mote 15: Summen des Separators „Mittelwert" 86
5.7 Allgemeine Vorgehensweise bei Mote 15: Mittelwerte des Separators „Prozent" 86
5.8 Lösung „Minimum-Maximum" für Mote 15 87
5.9 Prozent-basierte Lösung „Minimum-Maximum" für Mote 15 87
5.10 Lösung „Mittelwert" für Mote 15 87
5.11 Prozent-basierte Lösung „Mittelwert" für Mote 15 87
5.12 Summen aller Motes 88
5.13 Korrekte Gesamtkonfiguration 88
5.14 Gruppeneinteilung nach 36 h 91
5.15 Gruppeneinteilung nach 60 h 91
5.16 Gruppeneinteilung nach 72 h 92
5.17 Zeitliche Gruppeneinteilung durch die Berechnungsvorschrift Pearson Korrelationskoeffizient 92
5.18 Gruppeneinteilung und Fehler bei der inneren Erweiterung 93
5.19 aufsummierte Konfigurationen 97
5.20 Adjazenzfehler bei den Intel-Berkeley-Daten 97

6.1 Speicherverbrauch 108

1 Einführung

Ein Sensornetzwerk besteht aus einer Menge von mit Sensoren bestückten Computern, *Motes* genannt. Diese Motes besitzen einen speziellen Mikroprozessor, werden von Batterien betrieben und können untereinander drahtlos kommunizieren. Sie können mit einer breiten Vielfalt von Sensoren bestückt sein, wie z. B. Licht-, Temperatur-, Luftfeuchtigkeits-, Luftdruck-, Beschleunigungs-, Seismik-, Akustik-, Magnetfeld- oder GPS-Sensoren. Manche Knoten können auch als Aktoren, wie zum Beispiel Regler, oder als Basisstationen fungieren. Dadurch werden die gemessenen Sensordaten entweder direkt verarbeitet oder zentral gesammelt.

Einsatzgebiete von Sensornetzen liegen in der Überwachung im zivilen oder militärischen Bereich. Zum Beispiel kann die Bewegung von Gletschern aufgezeichnet werden, das Wanderverhalten von Steppentieren untersucht werden oder einfach die Heizung im Haus gesteuert werden. Für das Militär ist das Sammeln von Informationen über den Feind, wie Truppenposition und -stärke, für die Ausarbeitung einer optimalen Strategie besonders wichtig.

Diese Motes werden für ihren speziellen Einsatz meist in großer Anzahl in der vorgesehenen Umgebung verteilt um Daten zu sammeln. Oft ist eine nötige Wartung in diesen Umgebungen wegen der großen Anzahl oder einer großflächigen oder unzugänglichen Verteilung schwer durchzuführen. Aus diesem Grund müssen die Motes autonom arbeiten und Wartungsintervalle durch energiesparende Arbeitsweise vergrößern.

1.1 Motivation

Die Gruppierung von Sensorknoten auf Basis von Sensordaten und Konnektivität ist mit zwei Themengebieten verwandt: Lokalisierung und Clustering. Im nächsten Abschnitt wird allgemein eine Lokalisierung von Sensorknoten motiviert. Danach werden Gründe für ein Clustering angeführt. Im Abschnitt 1.1.3 wird abschließend darauf eingegangen, wie sich das Gruppierung von Sensorknoten auf Basis von Sensordaten und Konnektivität in beide verwandte Themengebiete einfügt.

1.1.1 Lokalisierung

Das Wissen, wo sich ein Sensorknoten befindet, lässt sich auf viele verschiedene Weisen verwenden. Ein Beispiel ist die Lokalisierung von gemessenen Ereignissen. Da Sensornetze zur Überwachung der Umwelt eingesetzt werden, ist es selbstverständlich, dass bei einem Ereignis nicht nur die Art des Ereignisses, sondern auch dessen Position interessant ist. Dazu müssen natürlich die einzelnen Sensoren auch ihre Position kennen. Aus den Positionen der Sensoren, die das Ereignis detektieren, lässt sich das Ereignis lokalisieren.

Ein weiteres wichtiges Anwendungsgebiet von Positionsinformationen liegt im Geographischen Routing. Hier werden Koordinaten als Adressen für Netzwerkknoten verwendet. Durch die Kenntnis der Positionen muss ein Knoten eine Nachricht nicht über alle möglichen Datenpfade senden, sondern kann eine Richtung zum Empfänger berechnen. Im drahtlosen Fall heißt das, dass Knoten, die für das Weiterleiten der Nachricht zuständig sind, nur dann aktiv werden müssen, wenn sie sich geographisch zwischen Sender und Empfänger befinden. Dadurch wird versucht, dass die Nachricht einen möglichst zeit- und energieoptimalen Weg zum Ziel nimmt.

Ebenfalls besteht überall dort eine Notwendigkeit die Position zu kennen, wo Messdaten mit dem Ort ihrer Erhebung verknüpft werden. So lassen sich Karten erstellen, die z. B. für Vorhersagen genutzt werden können.

Des Weiteren lassen sich aufgrund von Positionsinformationen Rückschlüsse auf Schwachstellen im Sensornetz ziehen und diese beseitigen.

1.1.2 Clustering

Da Sensorknoten nur über sehr wenig Energie verfügen, können sie nur mit den direkten Nachbarn kommunizieren. Um aber eine weit entfernte Basisstation zu erreichen, ist es nötig, dass Messwerte weitergeleitet werden. Um Energie zu sparen, werden manche Sensorknoten in der näheren Umgebung ausgewählt, um Daten zwischenzuspeichern, zusammenzufassen und dann gesammelt weiterzuleiten. Dadurch bilden sich lokale Cluster mit je einem Clusterhead.

Oft ist es auch sinnvoll Cluster zu bilden, in denen verschiedene Funktionalitäten redundant vorhanden sind. Dadurch ist es möglich, Fehler in Messwerten zu erkennen und so die Zuverlässigkeit des Sensornetzes zu erhöhen.

Eine andere Möglichkeit die Redundanz auszunutzen, ist die Lebensdauer des Sensornetzes zu verlängern. Jeder Cluster kann entscheiden, welcher Sensorknoten welche Funktion übernimmt und Sensoren mit gleicher Funktion zeitweise abschalten und so deren Energie sparen.

1.1.3 Gruppierung auf Basis von Sensordaten

Eine Gruppierung von Sensorknoten in geographisch beschränkte Regionen auf Basis von Sensordaten ist ein vielversprechender Ansatz für einen intelligenten Algorithmus. Dazu werden Merkmale aus Sensordaten extrahiert, die nur für eine Region zutreffen. So kann für jeden Mote automatisch entschieden werden, welcher Mote in derselben oder in einer anderen Region installiert ist. Durch diese Gruppierung ergeben sich automatisch dieselben Vorteile wie bei einem normalen Clustering. Durch die Voraussetzung Motes anhand von Ähnlichkeiten der Sensordaten zu gruppieren folgt sofort, dass damit auch redundante Motes gefunden werden. Diese Motes können sich untereinander absprechen, um so Fehlmessungen zu erkennen. Aber auch eine Abschaltung einzelner redundanter Motes ist denkbar, da sie in einer Region dieselben Sensordaten liefern. Dies ist natürlich nicht möglich, wenn auch die Unterschiede in einer Region ausschlaggebend für Anwendungen sind.

Andererseits kann ein Mote, der neu installiert wird und nicht über eine teure Ausstattung zur Positionserkennung, wie GPS, verfügt, sich mit bereits vorhandenen Motes

gruppieren und sich damit in einer Region lokalisieren. Bei der Installation von sehr vielen Sensorknoten wird so der Aufwand für eine individuelle Aufspielung der Information über die entsprechende Region durch die automatisierte Gruppierung erheblich verringert. Damit lässt sich selbstverständlich nur eine Lokalisierung mit der Genauigkeit der Größe einer Region bewerkstelligen. Oft ist aber diese Lokalisierung schon ausreichend. So fängt zum Beispiel ein Wartungsingenieur, der einen defekten Mote in einem Büro reparieren muss, mit einer Raumnummer mehr an als mit den GPS-Koordinaten des Motes.

Diese Positionsinformationen lassen sich durch Abbilden der Gruppen auf einen Nachbarschaftsgraph der Regionen für die Erstellung von Routing-Tabellen oder für Source-Routing verwenden. Dabei wird davon ausgegangen, dass die Kommunikation von einer Region zur nächsten möglich ist, aber eine größere Reichweite nicht unbedingt gegeben ist. Eine Adressierung erfolgt hier auf Ebene der Regionen, so dass ein Routing-Pfad aus einer Folge von Regionaladressen besteht. Sobald in jeder Region mehrere Netzwerkknoten vorhanden sind, kann eine Redundanz eingeführt werden oder die Bandbreite vergrößert werden, wenn unterschiedliche Frequenzen benutzt werden.

1.2 Problemstellung

Bevor eine Gruppierung von Sensorknoten auf Basis von Sensordaten stattfinden kann, müssen grundlegende Fragen angegangen werden:

- Kann man Regionen mit den auf Sensorknoten vorhandenen Sensoren unterschiedliche charakteristische Eigenschaften zusprechen?
- Unterscheiden sich die Messwerte zwischen verschiedenen Regionen deutlich genug für eine klare Trennung?
- Korrelieren Sensordaten innerhalb einer Region genug für eine Erkennung einer Gruppe?
- Welche vorhandenen Verfahren lassen sich anwenden oder übertragen?
- Wie lassen sich speziell für die Gruppierung von Sensorknoten auf Basis von Sensordaten und Konnektivität konzipierte Verfahren strukturieren?
- Lassen sich passende Verfahren auf Motes übertragen und parallel ausführen oder ist eine zentrale Sammlung und Verarbeitung nötig?

1.3 Gliederung

Im folgenden Kapitel werden die bereits erwähnten Motes genauer betrachtet und weitere Grundlagen vorgestellt.

Kapitel 3 beschäftigt sich mit veröffentlichten Beiträgen, die sich entweder mit dem verwandten Themengebiet „Lokalisierung von drahtlosen Geräten" oder mit „Gruppierung in Cluster" beschäftigen.

Da in keiner bisherigen Publikationen hinreichend auf die Gruppierung auf Basis der Sensordaten eingegangen wird, werden in Kapitel 4 neue Methoden erarbeitet, die hauptsächlich auf die Verwendung auf Sensorknoten ausgelegt sind. Kapitel 5 analysiert diese Methoden mit Hilfe einer Sammlung von Messwerten.

Kapitel 6 beschreibt schließlich eine prototypische Anwendung, die zur Gruppierung von Sensorknoten in einer Büroumgebung zum Einsatz kommt.

Basierend auf den Ergebnissen dieser Arbeit werden im Kapitel „Ausblick" mögliche weiterführende Themen vorgeschlagen, um dann in Kapitel 8 diese Arbeit zusammenfassend abzuschließen.

2 Grundlagen

Dieses Kapitel führt den Leser in das Gebiet der Sensornetze ein. Als wichtiger Bestandteil von Sensornetzen wird detaillierter auf die Hardware zwei verschiedener Sensorknoten eingegangen: TMote Sky und Mica2Dot. Abschließend wird in Abschnitt 2.4 das für diese Sensorknoten konzipierte Betriebssystem TinyOS vorgestellt.

2.1 Sensornetze

Sensornetze werden von vier grundlegenden Eigenschaften charakterisiert. Die erste bezieht sich auf die verwendeten Geräte, die *Motes*. Diese Motes besitzen einen oder mehrere Sensoren. Um die Sensoren zu nutzen, sind sie mit einem Mikroprozessor ausgerüstet. Die zweite Eigenschaft ist die Autonomie von Motes. Gewöhnlich existiert keine Infrastruktur im Einsatzgebiet der Motes. Das heißt, sie sind auf eine eigene Stromversorgung angewiesen, die sparsam benutzt werden muss. Es ist ebenfalls keine feste Infrastruktur zur Kommunikation zwischen Motes oder zu einer Basisstation. Deshalb müssen drahtlose Kommunikationswege genutzt werden, die als dritte charakterisierende Eigenschaft gelten. Die letzte Eigenschaft beschreibt die Anzahl der Motes. Interagieren zwischen 10 und 100 Motes handelt es sich um ein kleines Netzwerk. Zwischen 100 und 10 000 Motes ist ein mittleres Netzwerk anzusiedeln und Netzwerke mit über 10 000 werden als groß angesehen und sind zur Zeit noch visionär.

Aus diesen vier Charakteristika leiten sich Anforderungen an die Hardware und an die Software ab. Die Hardware zeichnet sich durch eine preiswerte, kleine, autonome und energiesparende Bauweise aus. Auf der Software-Seite müssen diese Hardwarebeschränkungen akzeptiert werden und mit autonomen, energiesparenden Algorithmen unterstützt werden. Daraus folgen in vielen Fällen enorme Herausforderungen an den Programmierer, der die meist komplexen Algorithmen mit oft trickreichen Techniken umsetzen muss.

2.2 TMote Sky

Die von Moteiv [Mot] hergestellten TMote Sky arbeiten mit einem MSP430 Mikroprozessor von Texas Instruments, einem 16 Bit RISC Prozessor, der mit 8 MHz getaktet wird. Er verfügt über 10 KB flüchtigen Arbeitsspeicher und 48 KB Flashspeicher, der das ausführbare Programm beherbergt. Für die drahtlose Kommunikation steht ein 250 kbps IEEE 802.15.4 (ZigBee) CC2420 Radio Transceiver von Chipcon zur Verfügung. Damit ist er in der Lage mit allen IEEE 802.15.4 konformen Geräten zu kommunizieren. Zusätzlich besitzt der TMote Sky drei frei verwendbare LEDs, eine USB Schnittstelle, über die er komfortabel programmiert werden kann oder die auch zur Kommunikation mit einem PC verwendet werden kann.

Abbildung 2.1: TMote Sky [Mot]

Die verwendeten TMote Sky sind mit einem SHT11 Sensor der Firma Sensirion ausgestattet, der die relative Luftfeuchtigkeit und die Temperatur misst. Die Luftfeuchtigkeit kann damit auf zwischen 0,5 % und 0,03 % relative Luftfeuchtigkeit genau bestimmt werden. Die Genauigkeit der Temperatur liegt zwischen 0,04 ° C und 0,01 ° C in einem Temperaturbereich von -40 ° C bis 123,8 ° C. Außerdem besitzen sie einen S1087 und einen S1087-1 Lichtsensor von der Firma Hamamatsu, deren Empfindlichkeitsbereich im kurzwelligen sichtbaren Licht von 320 nm bis 730 nm mit einer Empfindlichkeitsspitze bei 560 nm (Photosynthetisch Aktive Strahlung - PAR) bzw. im langwelligen sichtbaren Licht und infraroten Bereich von 320 nm bis 1100 nm mit einer Empfindlichkeitsspitze bei 960 nm (gesamte Sonnenstrahlung - TSR) liegt.

Die einzelnen Sensoren des TMote Sky wandeln die jeweilige Messgröße in ein analoges Signal um, das zwischen 0 V und der Betriebsspannung von 1,5 V liegt. Durch einen Analog-Digital-Wandler stehen der Software auf dem TMote Sky die digitale Abtastung der Spannung zur Verfügung.

Die TMote Sky sind mit einem 1 MB großer externer Flashspeicher ausgerüstet. Dieser besteht aus 16 unabhängigen Seiten zu je 64 KB, die von Programmen zu unterschiedlichen Zwecken für eine persistente Speicherung von Daten benutzt werden können. Dabei ist zu beachten, dass das byteweise Schreiben von Daten nur durch das Kippen der entsprechenden Bits von 1 nach 0 möglich ist. Deshalb muss darauf geachtet werden, dass sich ein Speicherbereich in einem zurückgesetzten, also gelöschten Zustand befindet, der dadurch charakterisiert ist, dass der Speicher nur auf eins gesetzte Bits enthält.

Angetrieben werden die TMote Sky von zwei Mignon-Batterien. Dadurch sind sie speziell auf einen äußerst flexiblen Einsatz ausgelegt. Aber natürlich ist auch eine lange Lebensdauer ohne intensive Wartung begrenzt.

2.3 Mica2Dot

Die Mica2Dot werden von Crossbow Technology Inc. [Cro] hergestellt. Sie sind mit einem Atmel ATmega128L 8 Bit RISC Mikroprozessor ausgestattet, der mit 4 MHz getaktet ist. Dieser Mikroprozessor verfügt über 4 KB internen SRAM zur Ablage von Variablen während des Programmlaufs und 128 KB internen Flash für die Ablage des Programmcodes. Als Zusatz enthalten sie eine frei verwendbare LED, 512 KB externen seriellen Flashspeicher für die persistente Speicherung von Daten und eine UART-Schnittstelle

zur Kommunikation mit einem PC über ein Programmiermodul. Zur drahtlosen Kommunikation besitzen sie einen 192 kbps CC1000 Radio Transceiver von Chipcon. Die Mica2Dot werden durch eine Knopfzelle mit Strom versorgt und sind auch nicht wesentlich größer. Zur Messung wird zusätzliche Hardware mit Sensoren benötigt. Eine Erweiterung ist das „Weather Board". Mit ihm kann die Luftfeuchtigkeit, Temperatur, Luftdruck und Lichtstärke gemessen werden.

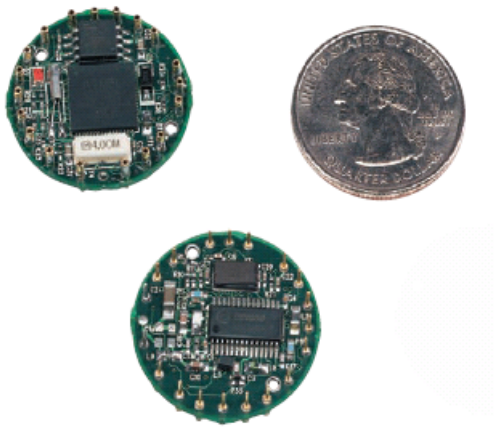


Abbildung 2.2: Mica2Dot [Cro]

2.4 TinyOS

Speziell für die Anforderungen von Sensornetzen wurde das Open-Source-Betriebssystem TinyOS [Tin, Mot] entwickelt. Implementierungen im Rahmen dieser Arbeit verwenden die Version 2. TinyOS zeichnet sich durch einen kleinen Betriebssystemkern aus, bietet aber auch modulare Erweiterungen für komplexe Anwendungen. Es existieren Umsetzungen für verschiedene Architekturen: Atmega8, Avrmote, Mica, Mica128, Mica2, Mica2Dot, MicaZ, Rene2, Telos, TelosB, TMote Sky und TMote Invent. Als Zusatz wird Möglichkeit geboten, mehrere Motes auf einem PC zu simulieren. Für alle Architekturen wird eine Abstraktionsstufe angeboten, die eine plattform-unabhängige Entwicklung von Anwenderprogrammen gestattet. So ist ein Zugriff auf die Sensoren, die Zeitgeber, die Alarme, die LEDs und den Flash der Motes möglich. Zur Abfrage der Sensoren wird vom Anwenderprogramm eine Funktion aufgerufen, die einen Messauftrag stellt. Das Anwenderprogramm muss nur noch auf die Lieferung des Sensorwertes warten. Zeitgeber können durch Benutzung einer Schnittstelle abgefragt werden. Genauso können Alarme durch das Anwenderprogramm mittels eines vorhandenen Services gestartet werden und nach Ablauf der Zeit eine Meldung bekommen. Auch für das An- bzw. Ausschalten von LEDs stehen abstrahierte Funktionen zum Aufruf bereit. Der Flash wird von einem Service in 16 Seiten aufgeteilt, die einen jeweils exklusiven Zugriff für verschiedene Anwendungen erlauben. Basieren auf dieser Virtualisierung arbeitet ein Logger, der die Seiten einer Anwendung für einen seriellen Lese- und Schreibzugriff abstrahiert. Des Weiteren wird die Kommunikation mit anderen Motes über eine drahtlose

Schnittstelle oder mit einem PC über eine drahtgebundene Schnittstelle jeweils bis zur Schicht 2 abstrahiert. Darauf wird eine weitere Abstraktion angeboten, die ein transparente Verwendung beider Kommunikationsmöglichkeiten erlaubt.

Das komplette Betriebssystem liegt als Quellcode in *nesC* vor. Dabei handelt es sich um eine Spezialisierung der Programmiersprache C für TinyOS. Durch das Kompilieren entsteht eine monolithische Datei, die sowohl Bootloader, Betriebssystem, Bibliotheken und Anwendungsprogramme enthält. Mit dieser Datei können Motes direkt programmiert werden.

NesC bietet die Möglichkeit einer objektorientierten Programmierweise in C, die eine einfache Anpassung an verschiedene Situationen erlaubt. Dazu existiert das Konzept von Modulen, Schnittstellen und Konfigurationen. In Schnittstellen können bidirektionale Methoden definiert werden. Die Module enthalten den eigentlichen auszuführenden Quellcode, geben aber auch an, welche Schnittstellen von der Implementierung des Moduls angeboten werden und welche Schnittstellen benutzt werden. Durch benutzende Schnittstellen ist die Kommunikation mit Modulen aus niedrigeren Abstraktionsstufen möglich. Angebotene Schnittstellen stellen den implementierten Service Modulen in höheren Abstraktionsstufen zur Verfügung. Dadurch, dass in Schnittstellen bidirektionale Methoden, also sowohl Kommandomethoden als auch Signalmethoden, definiert werden können, ist auch eine Meldung von Vorfällen an höhere Abstraktionsschichten möglich. Für sich alleine ist ein Modul nicht lauffähig. Durch Angaben in einer Konfiguration werden die von Modulen benutzten Schnittstellen mit angebotenen Schnittstellen anderer Module verdrahtet oder durch die Konfigurationsdatei selbst als benutzend oder anbietend deklariert. So können sowohl Module als auch andere Konfigurationen in einer Konfigurationsdatei instanziert werden. Dabei werden Module, die in mehreren Konfigurationen instanziert werden, in Wirklichkeit nur einmal instanziert. TinyOS 2 bietet als Neuerung generische Konfigurationen und Module an. Nutzt man dies trickreich aus, lässt sich die Beschränkung auf nur eine Instanz pro Modul in gewisser Weise aufheben.

TinyOS besitzt ein auf die Motes angepasstes Prinzip um Nebenläufigkeit für Programme zu erlauben. Es basiert auf *Tasks*, die einen Programmablauf darstellen und von Anfang bis Ende ausgeführt werden. Nach Abschluss eines Tasks sorgt ein Scheduler dafür, dass die Rechenzeit einem wartenden Task übertragen wird. Tasks können nur durch Hardware-Interrupts unterbrochen werden, die selbst blockierend sind und deshalb so schnell wie möglich ihre Arbeit erledigen sollten. Größere Aufgaben sollten generell in mehrere Tasks unterteilt werden, um sich fair und kooperativ zu verhalten und das System nicht zu blockieren. Ein Mechanismus, der ein solches Paradigma unterstützt, ist der *Split-Phase*-Mechanismus. Er besteht aus zwei grundsätzlichen Methoden: eine Kommandomethode, die einen Auftrag entgegen nimmt, und eine Signalmethode, die das Ende der Durchführung des Auftrags meldet. Die beauftragte Aufgabe bedarf in der Regel einer längeren Ausführung, die durchaus verzögert abgearbeitet werden kann. Es kann sich auch um einen Warteprozess handeln. Sobald das Ergebnis vorliegt, wird es dem Auftraggeber gemeldet. Dieser hat zwischenzeitlich seine Arbeit eingestellt und wird so wieder aufgeweckt, um mit dem Ergebnis weiter zu arbeiten.

Basierend auf den Konfigurationen lassen sich auch Bibliotheken bereitstellen. Durch das Einbinden von Bibliotheken werden Konfigurationen so überschrieben, dass sich die Bibliothek nahtlos in das Betriebssystem einbettet. Eine dieser Bibliotheken ist TinyDB [MHH03]. Sie stellt eine Abstraktion eines Sensornetzes auf eine Datenbank her. Dazu

bietet sie eine SQL-ähnliche Sprache zum Abrufen gewünschter Informationen von Motes an. Damit werden periodische oder einmalige Anfragen zugelassen. Die Antworten werden von TinyDB automatisch von Motes an den anfragenden PC geschickt. Dabei findet eine Optimierung, z. B. Sammlung und Filterung, der Daten schon im Netzwerk statt.

3 Verwandte Arbeiten

Die Gruppierung von Sensorknoten auf Basis von Sensordaten und Konnektivität weist eine Verwandtschaft zu zwei Themengebieten: Lokalisierung und Clustering. Im Folgenden werden Beiträge aus beiden Gebieten vorgestellt. In Abschnitt 3.3 wird beschrieben, wie diese Arbeit in beide Themengebiete eingeordnet werden kann.

3.1 Clusteringalgorithmen

Die meisten Applikationen berichten ihre Sensordaten an eine zentrale Station, von wo aus Aktionen initiiert werden, um auf die Sensordaten zu reagieren. Das Ziel der Clusteringalgorithmen besteht darin das Sensornetzwerk in Gruppen aufzuteilen, um die Sensordatensammlung energieeffizient zu vollziehen, indem vorgegebene Wege für die Nachrichten definiert werden. Für jede Gruppe wird ein sogenannter Clusterhead gewählt, der für alle anderen Sensorknoten in der Gruppe als unmittelbarer Ansprechpartner agiert, die übermittelten Nachrichten optimierend zusammenfasst und gesammelt an die Basisstation weiterleitet.

Ein einfacher Algorithmus zum Clustering mit flacher und tiefer Hierarchie wird von S. Bandyopadhyay [BC03] konzeptioniert. Die Funktionsweise basiert auf einem Wahrscheinlichkeitsparameter p, mit dem jeder Sensor Clusterhead wird und diese Entscheidung kundtut. Diese wird k Hops weitergegeben. Beim hierarchischen Ansatz wählen die Clusterheads einen Clusterhead höherer Stufe aus ihren Reihen mit demselben Algorithmus, aber unterschiedlichen Parametern p und k. Weitergehend zeigt S. Bandyopadhyay, wie diese Parameter optimal berechnet werden.

Ein ähnlicher Algorithmus zum Clustering, der Energie sparen soll, stammt von W. Heinzelman [HCB00]. Er nennt sich *LEACH* und basiert auf einem Rundensystem. In jeder Runde wird ein Knoten nach dem Zufallsprinzip Clusterhead und gibt dies bekannt. Knoten, die nicht Clusterheads wurden, melden sich bei Clusterheads an. Die Clusterheads erstellen dann für ihre Knoten einen TDMA Plan (Zeitmultiplexverfahren - Time Division Multiple Access), in dem sie die Verteilung der Sendezeiten ihrer untergeordneten Knoten festlegen.

Auf der Basis von *LEACH* schlägt A. Manjeshwar [MA01] mit *Teen* ein Routing Protokoll vor, um beim Berichten von Sensordaten Energie zu sparen. Dabei geht er von einer bestehenden hierarchischen Clusterstruktur, wie sie *LEACH* bietet, aus. Sensorknoten berichten ihre gemessenen Werte nicht in regelmäßigen Intervallen, sondern nur, wenn der Messwert einen bestimmten Pegel übersteigt und die Änderung über einem bestimmten Niveau liegt.

O. Younis [YF04a, YF04b] stellt mit *HEED* einen anderen Algorithmus vor, der ein dezentrales Clustering in konstanter Zeit und mit konstantem Nachrichtenaufwand realisiert. Für diesen Algorithmus werden drei verschiedene Zustände für Sensorknoten unterschieden: normaler Knoten, vorläufiger Clusterhead, endgültiger Clusterhead. Jeder

Knoten beginnt als normaler Knoten. Er wählt eine niedrige Wahrscheinlichkeit für die Erhebung zum Clusterhead aus, die proportional zur Verfügung stehenden Energie ist. In mehreren Runden wird geprüft, ob der Knoten als Clusterhead besser als andere geeignet ist. Ist dies der Fall bietet er sich als vorläufiger Clusterhead durch eine Nachricht an. Anschließend wird die Wahrscheinlichkeit verdoppelt. Steigt die Wahrscheinlichkeit über eins und hat sich kein anderer Knoten gemeldet, so wird er endgültiger Clusterhead. Andernfalls tritt er einem Cluster bei.

O. Younis [YFS04] erweitert *HEED* auch um eine Ansatz zur größeren Fehlertoleranz. Dafür führt er mehrere Clusterhead-Overlays ein, wobei in jedem der *HEED* Algorithmus ausgeführt wird. Zusätzlich wird noch darauf geachtet, dass ein Knoten nicht in zwei Overlays zum Clusterhead wird. Dadurch stehen einem Knoten mehrere redundante Clusterheads zur Auswahl.

Als Entwicklung und Implementierung bietet T. Nagata [NOY05] eine Middleware für Clustering auf Basis von ähnlichen Sensordaten. Dabei triggern detektierte Sensorereignisse ein Clusterbeitritt, bzw. -wechsel, wobei der Sensorknoten mit der meisten verfügbaren Energie sich früher als andere Sensorknoten zum Clusterhead erhebt und dies allen mitteilt. Die anderen Knoten, die ebenfalls dieses Ereignis registriert haben, schließen sich dem Cluster an. Auf die Methode der Erkennung von Ereignissen wird nicht eingegangen.

Ein weiteres Verfahren, das Sensordaten benutzt, stammt von J. Lester [LHB04]. Er verwendet die Messdaten von Beschleunigungssensoren, um festzustellen, ob zwei Geräte von derselben Person getragen werden. Dazu lässt er mehrere Personen mit verschiedenen Beschleunigungssensoren einen gewissen Weg entlang gehen. Die gesammelten Sensordaten werden in 8 sek Intervallen mit Hilfe einer Fourieranalyse ausgewertet. Um festzustellen, dass zwei Geräte von einer Person mitgetragen wurden, wird die Übereinstimmung der Fourierdaten durch der Kohärenz überprüft. J. Lester kommt zu dem Ergebnis, dass die Art der Messung 100 % exakt ist.

3.2 Lokalisierungsalgorithmen

Lokalisierungsalgorithmen werden im Bereich der Sensornetze für zwei Anwendungen eingesetzt. Die erste Anwendung lokalisiert gemessene Ereignisse in der Umwelt, für die die entsprechende Applikation ausgelegt ist. Dazu ist in der Regel eine genaue Kenntnis der Sensorknoten notwendig. Dies definiert das zweite Anwendungsgebiet. Da in gängigen Szenarien Sensorknoten selten mit einem GPS Empfänger oder Ähnlichem ausgestattet sind, müssen Algorithmen angewendet werden, die die Lokalisierung bewerkstelligen.

Dazu definiert A. Kröller [FKB+05] Klassifikationen für solche Algorithmen und untersucht jede Klasse: Ein anker-basiertes, ankerfreies und abstraktes Lokationsbewusstsein. Bei Ankern handelt es sich um einen Bruchteil der Knoten in einem Sensornetzwerk, die mit speziellen Erweiterungen ausgestattet sind, die eine genaue Positionsbestimmung ermöglichen. Alle übrigen Knoten müssen mit Algorithmen lokalisiert werden. Ankerbasierte Verfahren können in 3 Phasen aufgeteilt werden: Bestimmung der Distanzen zwischen den Ankern, Bestimmung der Distanzen von Sensorknoten zu Ankern, lokale Heuristiken zur Verbesserung der Position. Ankerfreie Verfahren müssen ohne bekannte

Koordinaten auskommen und definieren sich hierzu ein eigenes Koordinatensystem. Danach werden heuristische Verfahren zur Lokalisierung eingesetzt. Im Gebiet „abstraktes Lokationsbewusstsein" greift er symbolische Lokalisierung auf. Dabei werden Cluster aus benachbarten Sensorknoten erstellt und auf ein Graphen abgebildet, der so statt der Knotennachbarschaften die Clusternachbarschaften wiedergibt.

Viele Algorithmen beschäftigen sich mit einem anker-basierten oder ankerfreien Ansatz. Ein viel zitiertes Werk in diesem Bereich stammt von P. Bahl [BP00]. Er untersucht mit seiner Software-Entwicklung *RADAR* die Möglichkeit, ein mobiles Gerät innerhalb eines Gebäudes anhand der WLAN Signalstärke zu lokalisieren. Dazu sendet das mobile Gerät kontinuierlich Nachrichten aus, die von mehreren fest installierten Basisstationen registriert werden. Basierend auf der Signalstärken erfolgt die Lokalisierung durch Triangulation.

N. Bulusu [BHE00] schlägt ein schwerpunkt-basiertes Lokalisierungsverfahren vor. Ausgehend von einem Sensornetzwerk mit Ankern, die regelmäßig ihre Position in Funknachrichten preisgeben, schätzen alle anderen Knoten ihre Position. Dafür bilden sie den Schwerpunkt aus allen Positionen, die sie empfangen können.

Da dieses Verfahren sehr ungenau ist stellt J. Blumenthal [BRT05] eine Erweiterung für die schwerpunkt-basierte Lokalisation vor. Die Erweiterung führt Gewichte ein, die von der Signalstärke oder der Anzahl der Hops abhängen. Die von normalen Knoten empfangbaren Ankerpositionen werden bei der Schwerpunktbestimmung durch diese Gewichte verfeinert.

Ein weiteres Verfahren, das auf der Verwendung der Signalstärke beruht, stell T. He [HHB+03] vor. Dafür vergleicht er die Signalstärke von drei Ankerknoten mit Nachbarn. Aus diesen Vergleichen kann mittels eines Point-In-Triangulation-Tests geschlossen werden, ob der Sensorknoten innerhalb oder außerhalb eines durch die drei Ankerknoten aufspannenden Dreiecks sitzt. Diese Tests werden für mehrere Kombinationen von Ankerknoten durchgeführt, um aus der Schnittmenge der Dreiecke eine Position ableiten zu können.

Mit einem ähnlichen Prinzip versucht S. Guha [GMS05] Sensorknoten oder detektierte Ereignisse zu lokalisieren. Er grenzt die Regionen durch Bezier Flächen ein. Dabei können Flächen eingrenzend oder ausgrenzend wirken. Für die Lokalisierung werden zwei Radien definiert: ein großer Radius R für die maximale Übertragungsreichweite, der zum Tragen kommt, wenn ein Sensor in Reichweite eines anderen ist und deshalb seine Position auf einen Kreis mit diesem Radius um die Position des anderen einschränken kann; ein kleiner Radius r für die minimale Übertragungsreichweite, der zum Tragen kommt, wenn ein Sensor nicht in Reichweite eines anderen ist und deshalb seine Position auf Gebiete außerhalb dieses Kreises beschränken kann.

H. Chan [CLP05] stellt drei verschiedene Algorithmen zur näherungsweisen Berechnung der Position von Clusterheads vor. Alle drei gehen von Ankerknoten mit bekannten Positionen aus. Basierend auf bekannten Positionen von Nachbar-Clusterheads und einer Länge l können alle Clusterheads ihre eigene Position schätzen. Diese Position kann durch ein Federmodell in nachfolgenden Schritten präzisiert werden. Für den zweiten Algorithmus werden für einen Clusterhead alle Nachbar-Clusterheads systematisch im Gegen-/Uhrzeiger aufgelistet. Ausgehend von diesen Listen können Winkel zwischen zwei Knoten und deren Position geschätzt werden und an diese übermittelt werden. Diese Technik kann iterativ für eine Präzisierung der Schätzung eingesetzt werden. Der

dritte Algorithmus erweitert die ersten beiden, für den Fall, dass die Ankerknoten ebenfalls keine absolute Position kennen. Dazu werden für jeden Ankerknoten Transformationen auf ein gemeinsames Koordinatensystem bestimmt und als Transformation für die vorhergehenden zwei Algorithmen benutzt.

Eine weitere Art Entfernungen zu schätzen besteht neben der Verwendung von Signalstärken in der Messung der Laufzeitdifferenz zweier unterschiedlich gearteter Signale. M. Broxton [BLP05] entwickelt dazu ein Lokalisierungsalgorithmus für ein dichtes Netzwerk aus Motes, den *Pushpins*, die über einen IR-Empfänger und ein Mikrofon verfügen. Durch ein externes Gerät werden drei IR-Blitze und Klänge erzeugt, die von den Pushpins registriert werden. Durch einen Leader-Algorithmus wird jeweils der am nächsten zum Ereignis liegende *Pushpin* bestimmt. Durch diese drei Ereignisse werden Ankerknoten für ein Koordinatensystem bestimmt. Die restlichen Pushpins können darauf basierend ihre Position berechnen.

Mit *Spotlight* entwickelt und realisiert R. Stoleru [SHSL05] ein innovatives Prinzip für die Lokalisierung in offenem Gelände. Dieses System ist so entworfen, dass keine teure Hardware an den Sensorknoten zusätzlich angebracht werden muss, da es mit den bereits vorhandenen Lichtsensoren auskommt. Zusätzlich zu den Sensoren werden sogenannte *Spotlight-Geräte* verwendet, mit denen künstliche Ereignisse, wie punktuelle Lichterscheinungen und Muster erzeugt werden. Diese Ereignisse werden von den Sensorknoten detektiert und an das *Spotlight-Gerät* gemeldet. Dieses berechnet aus den Sensordaten die Position des Sensors.

Eine andere Art der Lokalisierung hat K. Kunze [KLJT05] mit seinem Experiment, in dem er die Position von Geräten mit Beschleunigungssensoren am Körper bestimmt, zum Ziel. Dazu teilt er die Messwerte in Intervalle zu 1 sek und 10 sek mit 5 sek Überlappung ein und berechnet daraus, wann eine Testperson läuft. Zu diesen Zeitpunkten wird eine Frame-by-Frame-Analyse für die Positionsbestimmung durchgeführt.

3.3 Einordnung

In beiden Themengebieten lassen sich Verwandtschaften zu dieser Arbeit feststellen. Dennoch ergeben sich einige fundamentale Unterschiede, die eine eindeutige Einordnung nicht zulassen.

Lokalisierung

Lokalisierungsalgorithmen lassen sich nach A. Kröller [FKB+05] durch die Art der verwendeten Koordinatenmodelle in drei Klassen unterteilen:

- Absolute Koordinaten basieren auf einem globalen Koordinatensystem, in dem sich die Netzwerkknoten lokalisieren müssen. Voraussetzung ist, dass mindestens drei Netzwerkknoten ihre Position kennen, zum Beispiel mit Hilfe von GPS. Durch verschiedene Methoden, wie Laufzeitbestimmungen oder Triangulation von Nachbarknoten, können weitere Knoten ihre Koordinaten bestimmen.
- Relative Koordinaten unterscheiden sich von den absoluten Koordinaten nur in der Tatsache, dass kein globales Koordinatensystem verwendet wird, sondern ein für das Netzwerk individuelles Koordinatensystem definiert wird.

- Symbolische Koordinaten drücken sich durch die Beschreibung der Umgebung aus.

Ohne Umstände können den jeweiligen Gruppen symbolische Koordinaten zugewiesen werden. Durch die Gruppierung von Knoten entsprechend den geographischen Regionen entsteht eine Verwandtschaft zu den symbolischen Lokalisierungsalgorithmen.

Clustering

Genau wie bei den vorgestellten Clustering-Verfahren befasst sich diese Arbeit mit der Bildung von Gruppen aus mehreren Netzwerkknoten. Im Gegensatz dazu wird weder ein Clusterhead nötig sein, noch eine Optimierung des Energieverbrauchs oder des Nachrichtenaufwands das primäre Ziel sein. Diese Arbeit beschränkt sich nur mit der Untersuchung, ob Sensorwerte für das Erkennen von Raumstrukturen ausreichen, und mit der Erforschung entsprechender Werkzeuge. Natürlich können die Ergebnisse leicht für ein Clustering verwendet werden.

4 Konzeption

Die Gruppierung von Sensorknoten auf Basis von Sensordaten und Konnektivität gehört zu einer bisher wenig erforschten Möglichkeit zur Gruppierung oder Lokalisierung. Um die Herausforderungen, Motes entsprechend den messbaren Gemeinsamkeiten einer regionalen Verteilung zu gruppieren, anzugehen, besteht eine Notwendigkeit in diesem Bereich weiterführende Forschung zu betreiben, welche Inhalt dieser Arbeit ist. Nach einer genauen Definition des verwendeten Systemmodells und nötiger Begriffe werden in Abschnitt 4.2 bestehende und neue Ansätze für eine Konzeption (weiter-)entwickelt. Es wird hier ein vierstufiges Verfahren eingeführt, das Sensordaten sukzessiv in eine fertige Gruppenaufteilung verwandelt. Dazu werden die gemessenen Sensorwerte aufbereitet, verglichen und zu Gruppenaufteilungen benutzt. Dieses vierstufige Modell zielt auf den unmittelbaren Einsatz auf Motes. In Abschnitt 4.3 wird ein verwandtes Verfahren aus dem Bereich der Biologie vorgestellt. Dieses ist für die Gruppierung von DNA-Analysen entworfen worden. Zur Verarbeitung nutzt es einen Algorithmus, der zwei Schritte ausführt.

4.1 Definition

Für das Verständnis dieser Arbeit ist es wichtig, das geplante Einsatzgebiet zu kennen. Dies wird im nächsten Abschnitt als futuristisches Büroszenario definiert. Dabei werden Büroräume durch Sensorknoten aufgewertet, um die Umwelt aufzunehmen und möglicherweise Aktoren zu steuern. Außerdem werden danach nötige Begriffe eingeführt, die eine bessere Verständigung ermöglichen.

4.1.1 Systemmodell

Im Folgenden wird von einer Umwelt ausgegangen, in der durch massive Wände getrennte Büroräume die geographisch beschränkten Regionen darstellen. Diese Büroräume werden von Personen normal genutzt, das heißt, es wird Licht an- bzw. ausgeschaltet, Türen oder Fenster geöffnet oder auch eine Klimaanlage/Heizung benutzt.

Sensorknoten vom bereits vorgestellten Typ TMote Sky bilden mit ihren Fähigkeiten und Einschränkungen ein Sensornetz, das sich über mehrere Büroräume erstreckt. Die Sensorknoten sind willkürlich in den Räumen verteilt, aber fest installiert und mit Sensoren für die Temperatur-, Luftfeuchtigkeits- und Lichtmessung ausgestattet. Eine drahtlose Kommunikation in einem Büroraum ist vollständig gegeben. Eine Kommunikation durch Wände zu anderen Büroräumen kann eventuell möglich sein.

4.1.2 Terminologie

Um eine bessere Verständigung in der restlichen Kapiteln zu fordern, werden hier einige Begriffe definiert und deren Bedeutung erklärt.

- Eine **Gruppenzugehörigkeit** bezieht sich auf einen Mote. Sie bezeichnet das Ergebnis des Vorgangs, wie ein Mote die Gruppe mit der größten Ähnlichkeit in den Sensordaten findet.
- Eine **Gruppenkonfiguration** beschreibt die Gruppenzugehörigkeit eines Motes als Datenobjekt.
- Als **Kanal** wird eine Sequenz von angewandten Analysemethoden ausgehend von einer Messreihe eines Sensors bis hin zur daraus resultierenden Entscheidung über die Gruppenzugehörigkeit bezeichnet.

4.2 Analyseverfahren

Am Anfang seines Einsatzes besitzt ein Mote noch keinerlei Kenntnis, in welcher Region er sich befindet und ob in seiner Region noch weitere Motes installiert sind. Um diese eventuell vorhandenen Motes zu identifizieren und sich mit ihnen zu einer Gruppe zusammen zu schließen, kann er sich nur auf Sensordaten stützen, die signifikante Ähnlichkeiten in einer Region oder Unterschiede zwischen Regionen aufweisen. Um diese Sensordaten in eine Gruppenzugehörigkeit zu verwandeln, wird im Folgenden ein vierstufiges Verarbeitungsmodell mit unterschiedlichen Analyseverfahren vorgestellt. Dabei werden aus den Sensordaten die Ähnlichkeiten oder Unterschiede als Erkennungsmerkmale für eine Region extrahiert. Diese Erkennungsmerkmale können einzelne Ereignisse oder Charakteristika im zeitlichen Verlauf der Sensorwerte sein. Dabei sind Ereignisse spontane, ausgeprägte und zeitlich beschränkte Erscheinungen, die im Verlauf deutlich heraustreten.

Mit den TMote Sky lassen sich Sensorwerte für die Temperatur, Luftfeuchtigkeit und zwei Empfindlichkeitsbereiche im Licht messen. Im restlichen Kapitels wird der Einfachheit halber nicht zwischen den einzelnen Sensortypen unterschieden. Stattdessen werden allgemeine Verfahren vorgestellt, um sich nicht auf das Vorhandensein dieser Sensoren verlassen zu müssen und auch andere Arten von Sensoren, wie z. B. Akustiksensoren, zu unterstützen. Zusätzlich zu den Sensoren besitzen Motes die Fähigkeit die Signalstärke für eingehende Nachrichten zu erfassen. Die Signalstärke kann auch zur Gruppenbildung verwendet werden, unterscheidet sich aber in der Erfassung und Bedeutung von den Sensordaten und wird deshalb in Abschnitt 4.2.4 gesondert betrachtet.

4.2.1 Verarbeitungsstufen

Durch die Beschaffenheit der Motes, die Messwerte bereits digitalisiert liefern, ergibt sich eine kanonische Vorgehensweise, um von den erfassten Sensordaten zu einer Einteilung in Gruppen der Sensorknoten zu gelangen. Diese fordert eine Abstraktion der Daten in vier aufeinander aufbauenden logischen Stufen, wie sie in Abbildung 4.1 dargestellt sind.

- Die erste Stufe bereitet die gesammelten Messdaten auf jedem Mote auf. Entsprechend einer digitalen Signalverarbeitung werden für diese Stufe spezielle Filter für die Vorverarbeitung entworfen.
- Die zweite Stufe führt den essentiellen Vergleich der gefilterten Daten von allen Motes aus. Dieser bewertet bereits die Ähnlichkeiten oder Unterschiede zwischen den Motes.
- Die dritte Stufe versucht auf Basis der Vergleiche eine Einteilung in Gruppen zu finden. Diese Stufe ist nötig, um die Resultate von unterschiedlichen Typen von Kanälen in ein homogenes Format zu fassen. Sie ist eng mit der zweiten Stufe verknüpft, da deren Parameterwahl starke Auswirkungen zeigen kann.
- Die vierte Stufe fasst die homogenisierten Ergebnisse der einzelnen Kanäle aus der dritten Stufe zu einem Gesamtergebnis zusammen. Damit liegt am Ende eine vollständige Einteilung in Gruppen vor, die die Region reflektieren.

Je weiter die Daten abstrahiert werden, desto wichtiger ist es, mit Messungenauigkeiten und Messfehlern umzugehen. Die Verarbeitung der Sensordaten auf den untersten Stufen ist zu wenig abstrahiert, um die Auswirkungen der Sensordaten auf die Gruppenbildung erkennen und eine vollständige Fehlerbehebung durchführen zu können. Deshalb müssen die dritte und vierte Stufe in der Lage sein, Messfehler in den Sensordaten zu tolerieren.

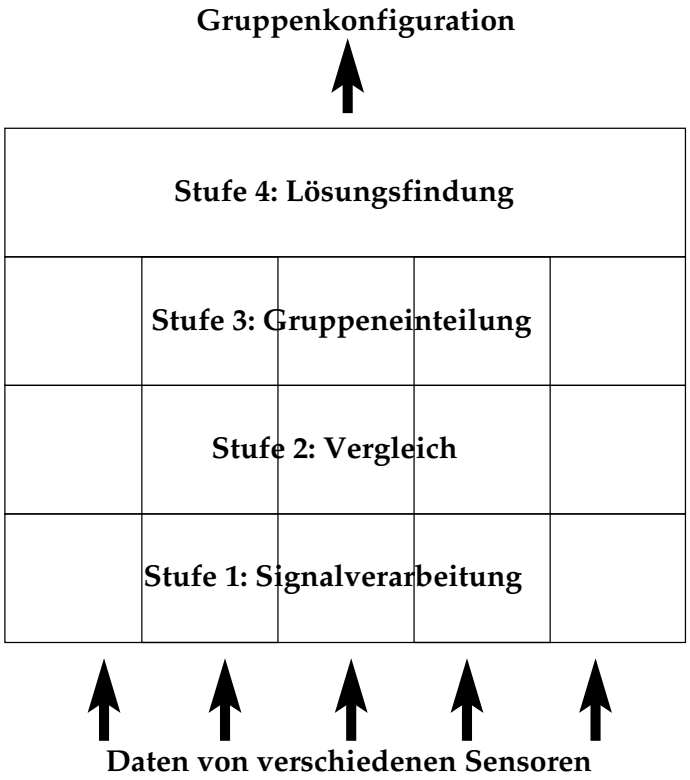


Abbildung 4.1: Verarbeitungsstufen

4.2.2 Stufe 1: Signalverarbeitung

In der ersten Stufe werden die Messwerte einzelner Sensoren für einen späteren Vergleich aufbereitet. Die analogen Messsignale werden von den Motes digitalisiert und können somit durch den Prozessor einer digitalen Signalverarbeitung unterzogen werden. Dazu wird der zeitliche Verlauf von Sensorwerten als Kurve der Messwerte über die Zeit aufgefasst. Alle Verfahren in dieser Stufe sind wichtige Hilfsmittel für die nächste Stufe.

Durch die Verarbeitung der Messwerte werden virtuelle Messreihen eingeführt, die sich in den weiteren Stufen nicht von realen Messreihen unterscheiden. Ein Beispiel für eine solche virtuelle Messreihe wäre die „Ableitung der Temperatur", die aus der realen Messreihe „Temperatur" durch Ableiten der Kurve entsteht.

Im folgenden Abschnitt werden Filter aufgelistet. Jeder einzelne führt eine für ihn charakteristische Signalverarbeitung auf dem Verlauf einer Kurve durch. Danach werden Filter vorgestellt, die Ereignisse im Verlauf einer Kurve erkennen können.

4.2.2.1 Verlauf

Der Verlauf von Messwerten betrachtet eine kontinuierliche Messung über einen längeren Zeitraum. Auf den Motes ist wie für jede digitale Signalverarbeitung ein Sampling in diskreten Zeitschritten nötig. Die Sampling-Periode muss insbesondere auf den Motes so angepasst werden, dass ein Kompromiss zwischen Messgenauigkeit und Energie, die für jede einzelne Messung verbraucht wird, gefunden wird.

Die Messwerte der Sensoren werden mit unterschiedlichen Filtern analysiert und ergeben so unterschiedliche Ausgangskurven für die nächste Stufe. Ein Filter bekommt eine Eingabekurve $x(t)$ aus Sensorwerten in diskreten Zeitschritten $t = 0, 1, 2, \ldots$, modifiziert diese und gibt das Ergebnis als Kurve $y(t)$ aus. Alle Filter können in beliebiger Reihenfolge (auch mehrfach) angewendet werden, indem die Ausgabe eines Filters als Eingabe eines weiteren Filters dient. Dadurch werden sogenannte *Filterketten* gebildet.

In Abbildung 4.2 ist ein einfaches Beispiel dargestellt. Die gezeigten Kurven sind von Hand generiert und so gestaltet, dass sie gewisse Charakteristika aufweisen, um als Eingangskurve $x(t)$ folgende Erklärungen zu veranschaulichen. Dabei sollen sie realen Sensormessungen möglichst ähnlich sein. Die Abbildung zeigt den Messverlauf von vier unterschiedlichen Motes mit den Namen 05, 06, 07 und 08. Dabei werden die Werte, die der Digital-Analog-Wandler vom Sensor liest, über der vergangenen Zeit abgetragen. Die Kurven sind so gewählt, dass die Motes 05 und 06 und die Motes 07 und 08 jeweils zueinander Ähnlichkeiten aufweisen.

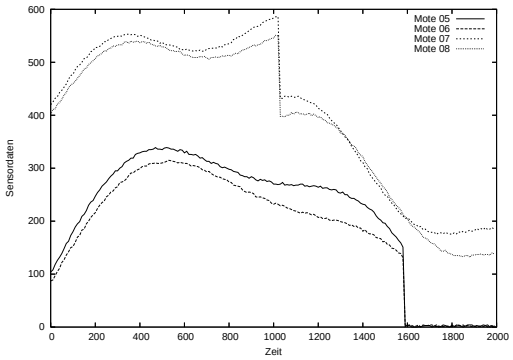


Abbildung 4.2: Beispiel für die Signalverarbeitung

Der Filter „Änderung" spiegelt im kontinuierlichen Fall eine Ableitung wider. Er erzeugt die Differenz aus zwei zeitlich aufeinander folgenden Werten $y(t) = \Delta x(t) = x(t) - x(t-1)$. Erwartet wird, dass sich große Änderungen in den Messwerten als Ereignis erkennen lassen. So zeigen sich die großen Sprünge in der Beispielkurve als große Ausschläge in Abbildung 4.3. Leider ist schwer zu erkennen, dass es sich dabei um die Ausschläge von je zwei Motes handelt, weil sie ungefähr von gleicher Größe sind und sich deshalb in der Grafik überdecken. Da bei den Sprüngen die Kurve fällt, erzeugt dies negative Ausschläge. Die kleinen Ausschläge in der Abbildung stellen ein sogenanntes *Rauschen* dar. Dies resultiert aus natürlichen Schwankungen in der Umgebung und Ungenauigkeiten der Sensoren.

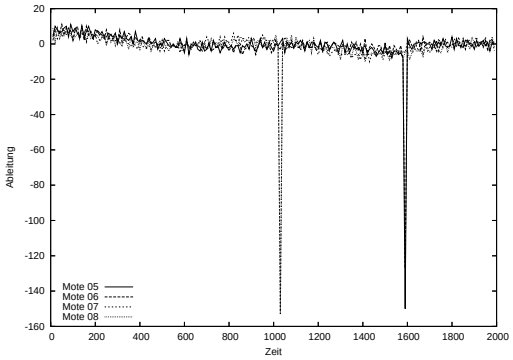


Abbildung 4.3: Beispiel für den Filter „Änderung"

Der Filter „prozentuale Änderung" erzeugt einen Wert, der die absolute Differenz aus zwei zeitlich aufeinander folgenden Werten im Verhältnis zum Maximum beider Werte

wiedergibt.

$$y(t) = \frac{|\Delta x(t)|}{\max(x(t), x(t-1))}$$

Dadurch wird eine Änderung prozentual zum tatsächlichen Wert ausgedrückt. Genauso wie beim Filter „Änderung" ist dieser Filter ein Kandidat für das Aufspüren von Ereignissen. Um aber Rauschen nahe bei Null keine großen Änderungen zuzuweisen und später daraus auf Ereignissen zu schließen, sollte mit einem Schwellenwert für die Änderung $|\Delta x(t)|$ gearbeitet werden. Ist diese Änderung zu klein, wird sie zu Null abgeschnitten. So zeigt Abbildung 4.4 das Ergebnis des Filters, wenn er auf die Beispielkurven angewendet wird. Auch hier sind die Sprünge als Ausschläge erkennbar und auch hier sind an diesen Stellen wieder die Kurven von je zwei Motes ungefähr deckungsgleich.

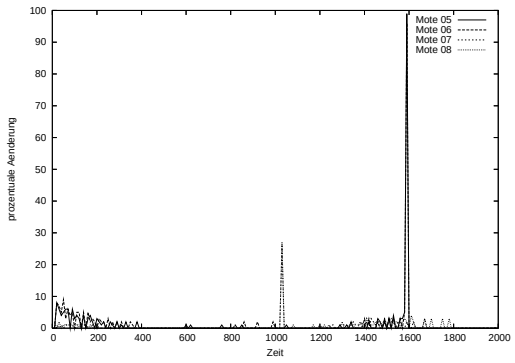


Abbildung 4.4: Beispiel für den Filter „prozentuale Änderung

Der Filter „Betrag" bildet den Absolutwert $y(t) = |x(t)|$. Er ist hauptsächlich in Verbindung mit dem Filter „Änderung" sinnvoll, da dort negative Werte entstehen können, auf die dieser Filter eine Auswirkung hat. Abbildung 4.5 zeigt das Ergebnis dieses Filters, wenn er auf die Signale von Abbildung 4.3 angewendet wird.

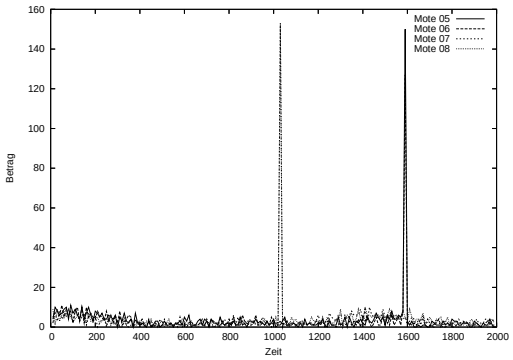


Abbildung 4.5: Beispiel für den Filter „Betrag"

Der Filter „Mittelwert" mittelt über eine bestimmte Zeit N die Werte $y(t) = \frac{1}{N} \sum_{i=t-N}^{t} x(i)$. Als Parameter wurde $N = 5$ in Abbildung 4.6 gewählt. Dadurch erscheint die Messkurve glatter, aber die Sprünge sind trotzdem noch sichtbar.

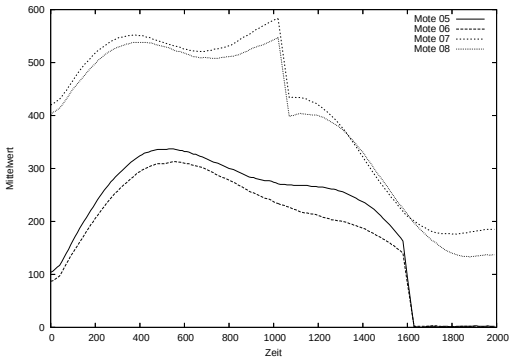


Abbildung 4.6: Beispiel für den Filter „Mittelwert"

Der Filter „Integral" summiert alle in der Vergangenheit liegenden Werte auf $x(t) = \sum_{i=0}^{t} x(i)$. Dadurch bekommen die Messwerte eine „Erinnerung" an die Vergangenheit, was in der nächsten Stufe ausgenutzt werden kann. In Abbildung 4.7 sind die Integrale der Beispielkurven dargestellt.

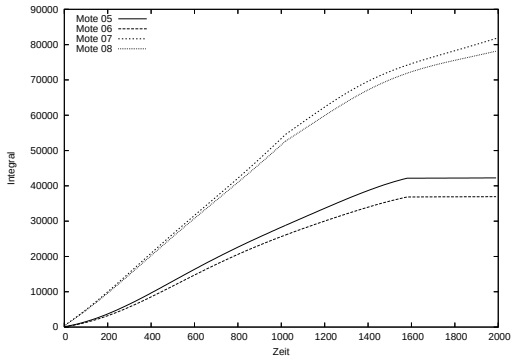


Abbildung 4.7: Beispiel für den Filter „Integral"

Der Filter „Integral vorzeichengleicher Werte" summiert nur die Werte auf, die in einer beschränkten Anzahl von Schritten in der Vergangenheit ein gleiches Vorzeichen aufweisen. Er dient ebenfalls dazu, einen Erinnerungseffekt an die Vergangenheit zu schaffen, der zur Ereigniserkennung nützlich sein kann. Geht man zum Beispiel von einem „abrupten" Anstieg der Temperatur als Ereignis aus, ist eine signifikante Wertentwicklung über mehrere Messschritte ausgedehnt.

Der Filter „Normierung" sucht das Minimum und Maximum der Kurve in einem zeitlichen Bereich N und normiert danach die Messwerte auf diesen Bereich.

$$y(t) = \frac{x(t) - \min(x(i))}{\max(x(i)) - \min(x(i))}, i = t - N \ldots t$$

Dadurch wird versucht die Ähnlichkeiten zweier Kurven für andere Filter herauszuarbeiten, indem in denselben Wertebereich transformiert wird. Nützlich kann dies sein, wenn sie sich wegen Kalibrierungsfehler nur durch ein Offset voneinander unterscheiden oder wenn deren Maximalwerte unterschiedlich weit auseinander liegen, weil sie mit einer Konstante skaliert sind. Vorstellbar ist letztere Situation für zwei Motes, die in einem Raum unterschiedlich intensiv mit Licht bestrahlt werden, wodurch sich die Sensorwerte selbstverständlich unterscheiden. Trotzdem können deren Kurven Ähnlichkeiten besitzen, die ohne diesen Filter nicht automatisiert erkennbar sind. Für Abbildung 4.8 wird das Minimum und Maximum auf dem ganzen Zeitbereich gesucht. Man sieht, dass nach der Transformation durch den Filter der Wertebereich aller Kurven übereinstimmt.

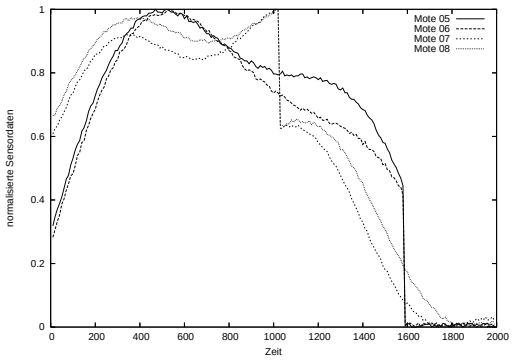


Abbildung 4.8: Beispiel für den Filter „Normierung"

Der Filter „Steigung" gibt die Tendenz (auf- oder abwärts) einer Kurve wider.

$$y(t) = \begin{cases} 1 & \Delta x(t) \geq 0 \\ -1 & \Delta x(t) < 0 \end{cases}$$

Dies ist eine weitere Möglichkeit Kalibrierungsfehler oder weit auseinander liegende Maximalwerte zu behandeln, da keine Messwerte berücksichtigt werden und trotzdem ein Verlauf der Kurve beschrieben wird.

Der Filter „Fourier-Transformation" transformiert eine Kurve in den Frequenzraum.

$$y(\omega) = \mathcal{F}\{x(t)\} = \frac{1}{\sqrt{2\pi}} \int x(t) e^{-i\omega t} dt$$

Durch die Transformation entsteht eine komplexe Funktion. Zusammen mit der Rücktransformation

$$x(t) = \mathcal{F}^{-1}\{y(\omega)\} = \frac{1}{\sqrt{2\pi}} \int y(\omega) e^{i\omega t} d\omega$$

stellt er die Grundlage für eine weitere Signalverarbeitung durch einen Bandpassfilter dar. Für den Fall, dass $x(t)$ reell ist, ist auch $\mathcal{F}^{-1}\{F\{x(t)\}\}$ wieder eine reelle Funktion. Abbildung 4.9 zeigt die vier Beispielkurven, wie sie im Frequenzraum nach einer Fourier-Transformation aussehen. Für die Darstellung der komplexen Funktion bietet sich deren Betrag an. Ein Hauptteil der transformierten Kurve liegt hier im niederfrequenten Bereich.

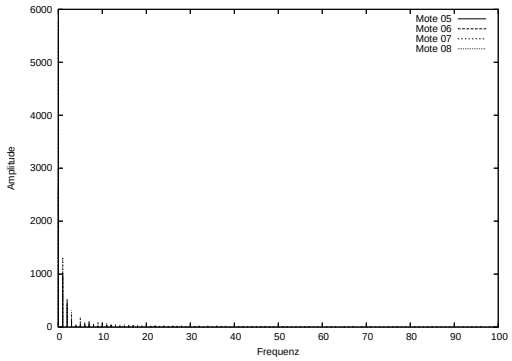
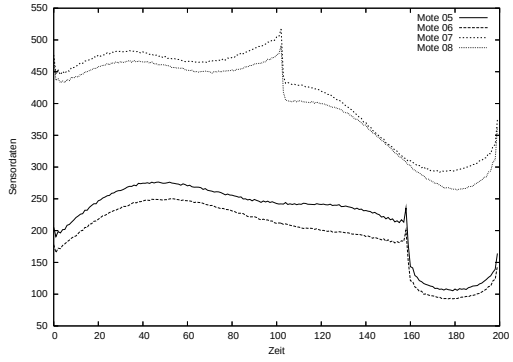
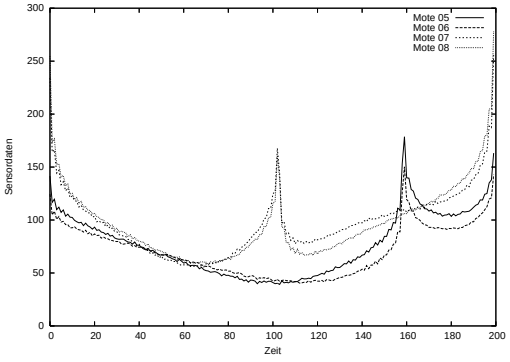


Abbildung 4.9: Beispiel für den Filter „Fourier-Transformation"

Der Filter „Bandpass" filtert ein Frequenzband aus einer fourier-transformierten Kurve heraus. Je nach Einstellung kann er auch als Tief- oder Hochpassfilter verwendet werden. Der Tiefpassfilter kann genau wie der Filter „Mittelwert" zum Glätten der Kurve verwendet werden. Der Hochpassfilter dagegen stellt die hochfrequenten Änderungen dar. Während Abbildung 4.10 a) einen Tiefpass für die unteren 50 % der Kurve darstellt, zeigt Abbildung 4.10 b) einen Hochpass für die oberen 50 % der Frequenzen.



(a) Ruecktransformation nach einer Filterung durch den Tiefpass



(b) Ruecktransformation nach einer Filterung durch den Hochpass

Abbildung 4.10: Beispiele für den Filter „Bandpass"

4.2.2.2 Ereignisse

Neben der Betrachtung des Verlaufs von Sensordaten wird vor allem für die Lichtsensoren erwartet, dass sich Ereignisse gut verwenden lassen. In einer Büroumgebung ist das künstliche Licht ein entscheidender Faktor. So sollte ein An- bzw. Ausschalten der

Bürobeleuchtung detektierbar sein und vor allem sich im ganzen Raum einheitlich, in anderen Büroräumen dagegen nicht, ausbreiten.

Da die Sensoren der TMote Sky keine Ereignisse hardware-basiert signalisieren, müssen dieselben Messwerte wie aus dem Verlauf verwendet werden und per Software analysiert werden. Dazu wird ähnlich wie beim Verlauf folgender Filter vorgeschlagen, mit dem Unterschied, dass seine Ausgabe nur binär ist und somit ein Vorhandensein eines Ereignisses signalisiert.

Der Filter „Ereignis" signalisiert ein Ereignis, wenn der Eingabewert $x(t)$ einen bestimmten Schwellenwert s überschreitet.

$$y(t) = \begin{cases} 1 & x(t) \geq s \\ 0 & x(t) < s \end{cases}$$

Durch entsprechende Wahl der Eingangsmessreihe (real oder virtuell) und des Schwellenwerts s lassen sich alle spontanen, ausgeprägten und zeitlich beschränkten Erscheinungen als Ereignis klassifizieren. Man kann sich durchaus vorstellen, Maxima und Minima in den Messwerten zu suchen. Aber das größte nutzbare Potential liegt in Ereignissen, die durch eine signifikante Änderung der Messwerte entstehen. Entsprechend den zwei Verlaufsfiltern, die mit Änderung arbeiten, existieren zwei Möglichkeiten hierfür.

Absolute Ereignisse sind große Pegeländerungen zwischen zwei aufeinander folgenden Messwerten. Diese können große Schwankungen in der Temperatur oder in der Luftfeuchtigkeit sein, aber auch Wolken, die die Sonne verdunkeln. Realisierbar ist diese Ereigniserkennung durch die Kombination aus dem Verlaufsfilter „Änderung" und dem Filter „Ereignis" . Ist die Änderung größer als der Schwellenwert, wird der Änderung ein Ereignis zugeordnet. Dadurch werden die Änderungen, die durch das Rauschen, also Ungenauigkeiten der Sensoren, hervorgerufen werden, vernachlässigt. Im Beispiel 4.11 ist dies dargestellt. Die Grundlage bilden die Kurven aus Abbildung 4.5. Eine Markierung zeigt ein Ereignis zu einer bestimmten Zeit an. Die Zeile der Markierung gibt den Mote an, der das Ereignis detektiert.

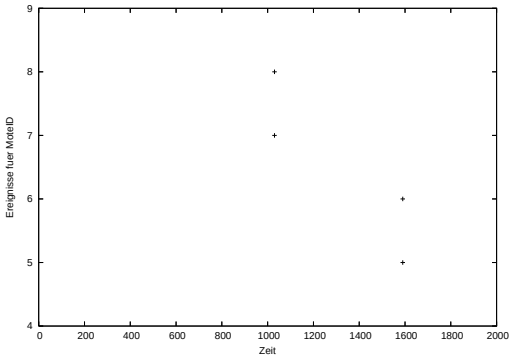


Abbildung 4.11: Beispiel für die Erkennung absoluter Ereignisse

Prozentuale Ereignisse unterscheiden sich von absoluten Ereignissen dadurch, dass die prozentuale Änderung zwischen zwei aufeinander folgenden Messwerten betrachtet wird. In diese Kategorie fällt beispielsweise das Ein- und Ausschalten der Bürobeleuchtung. Dies erzeugt bei wenig Tageslicht eine große prozentuale Änderung und eignet sich so als Ereignisse. Dem entsprechend lässt sich diese Ereigniserkennung durch den Verlaufsfilter „prozentuale Änderung" und den Filter „Ereignis" realisieren, um so nur prozentuale Änderungen mit genügend Ausschlag als Ereignisse zu interpretieren. Analog zu den absoluten Ereignissen ist in Beispiel 4.12 die Ereigniserkennung von prozentualen Ereignissen dargestellt. Die Basis bilden dabei die Kurven aus Abbildung 4.4.

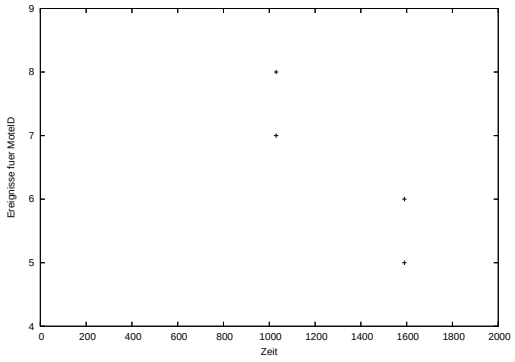


Abbildung 4.12: Beispiel für die Erkennung prozentualer Ereignisse

4.2.3 Stufe 2: Vergleich

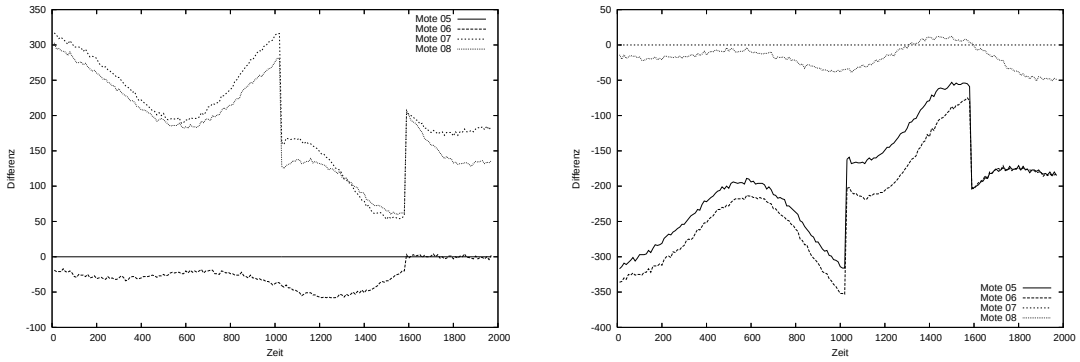
In der zweiten Stufe erfolgt ein Vergleich entsprechender Messreihen von je zwei Motes $a(t)$ und $b(t)$ zu jedem Zeitschritt t durch Vergleichsoperatoren, sogenannte *Komparatoren*. Die Daten ergeben sich aus den Filtern, die in der ersten Stufe für bestimmte Analysen entsprechend aufbereitet worden sind, oder direkt aus den unveränderten Sensordaten (reale Messreihen). Auch hier ist das Ergebnis $y_{ab}(t)$ eine Kurve über die Zeit. Dieses Ergebnis kann durch verschiedene Modifikatoren zu $\hat{y}_{ab}(t)$ weiter aufbereitet werden, um schließlich den Grad der Verwandtschaft zwischen je zwei Motes zu symbolisieren. Damit ist das Ziel dieser Stufe - das Prognostizieren einer wahrscheinlichen Verwandtschaft - erreicht.

Die folgenden Grafiken zeigen die Berechnungen von nur einem Mote. Der betrachtete Mote, *Referenzmote* genannt, ist jeweils zur Abbildung aufgeführt. Die Motes, mit denen die Vergleiche angestellt werden, werden *Vergleichsmotes* genannt.

4.2.3.1 Verlauf

Mit den Verlaufskomparatoren werden Verlaufsmessreihen miteinander verglichen. Diese Vergleiche $y_{ab}(t)$ geben Auskunft darüber, wie ähnlich zwei (gefilterte) Kurven $a(t)$ und $b(t)$ zu einer bestimmten Zeit t zueinander sind.

Der Komparator „Differenz" bildet die Differenz zwischen den zwei Eingangskurven, also $y_{ab}(t) = a(t) - b(t)$. Damit lassen sich zwei Kurven auf sehr einfache Weise vergleichen. Dies ist besonders für die Verarbeitung auf den ressourcen-beschränkten Motes ausgelegt. Das Beispiel weiterführend zeigt Abbildung 4.13 a) die jeweilige Differenz der Sensordaten von Mote 05 zu den anderen Motes abgetragen über die Zeit. Man sieht, dass die Differenz zu Mote 06 kleiner ist als die zu den Motes 07 und 08. Durch die geringere Differenz lässt sich auf eine größere Ähnlichkeit schließen. Abbildung 4.13 b) zeigt die entsprechenden Kurven, wie sie Mote 07 berechnet. Hier ist deutlich erkennbar, dass die Differenz von Mote 08 näher an der Nulllinie liegt, also eine größere Ähnlichkeit aufweist.



(a) Differenz von Mote 05

(b) Differenz von Mote 07

Abbildung 4.13: Beispiele für den Komparator „Differenz"

Der Komparator „Produkt" multipliziert die Werte beider Kurven miteinander $y_{ab}(t) = a(t) \cdot b(t)$. Dies stellt somit ebenfalls eine einfache Art dar, einen Vergleich anzustellen. Als Haupteinsatz ist der Vergleich von Daten, die durch den Filter „Steigung" bearbeitet wurden, vorgesehen. Dieser Filter liefert ein zweiwertiges Ergebnis, das 1 ist für eine steigende Kurve und -1 für eine fallende Kurve. Dort eingesetzt erzeugt gleiche Tendenz in $a(t)$ und $b(t)$ ($a(t) = b(t)$) ein positives Ergebnis $y_{ab}(t) = 1$, unterschiedliche Tendenz ($a(t) \neq b(t)$) dagegen ein negatives Ergebnis $y_{ab}(t) = -1$. Abbildung 4.14 zeigt den Vergleich von Mote 05 mit den anderen. Man sieht, dass sich die Tendenzen der Motes 07 und 08 an mehr Stellen unterscheiden als die Tendenzen von Mote 06. Daraus lässt sich eine Ähnlichkeit zwischen den Motes 05 und 06 ableiten.

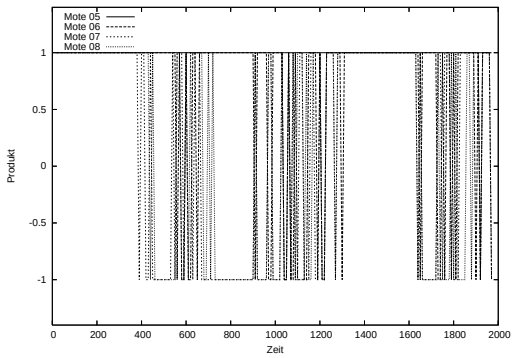


Abbildung 4.14: Beispiel für den Komparator „Produkt"

Der Komparator „Prozent" teilt die Werte der einen Kurve durch die der anderen $y_{ab}(t) = a(t)/b(t)$. Die Idee hinter dieser Vergleichsweise ähnelt der des Komparators „Differenz". Es ist eine einfach, aber leistungsstarke Vergleichsmöglichkeit. Die Tatsache, dass die Division nicht auf allen Motes in Hardware vorhanden ist und somit kompliziert nachgebildet werden muss, darf bei dieser Methode nicht außer Acht gelassen werden. Abbildung 4.15 stellt das Ergebnis dieses Komparators für Mote 07 dar. Der Vergleich ergibt, dass Mote 08 zu jedem Zeitschritt näher als die anderen Motes an der 100 % Übereinstimmungslinie liegt.

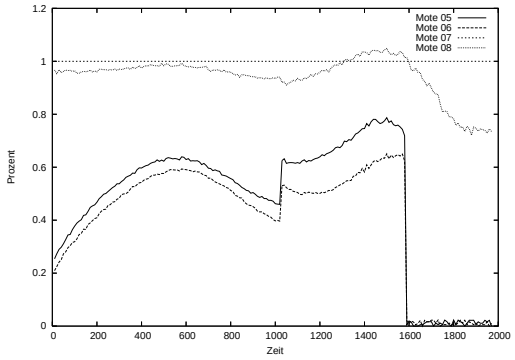


Abbildung 4.15: Beispiel für den Komparator „Prozent"

Der Komparator „Kohärenz" berechnet nach der Methode von P. Welch [Wel67] die Kohärenz zweier Kurven. Dazu teilt er die Kurven in N gleich große, überlappende Intervalle a_i bzw. b_i auf, die mit einer Fensterfunktion $w(t)$ multipliziert werden. Als Fensterfunktion wird zum Beispiel eine Hammingfunktion verwendet. Für jedes Intervall werden die Kurven $A_i = \mathcal{F}(a_i \cdot w)$ und $B_i = \mathcal{F}(b_i \cdot w)$ durch Fourier-Transformation erzeugt und diese zu $A(f) = \frac{1}{N}\sum_{i=1}^{N} A_i(f)$ bzw. $B(f) = \frac{1}{N}\sum_{i=1}^{N} B_i(f)$ gemittelt. Die-

se Daten werden frequenzweise miteinander multipliziert und durch den Betrag beider Transformierten geteilt $k(f) = \frac{|A(f) \cdot B(f)|^2}{|A(f)| \cdot |B(f)|}$. Durch Integrieren dieses Ergebnisses berechnet sich der Verwandtschaftsgrad $y_{ab}(t) = \int k(f) df$. Es ist zwar ein viel versprechender Ansatz, der aber auch durch seine extreme Komplexität eine Herausforderung für eine effiziente Implementierung auf den Motes darstellt. In Abbildung 4.16 sind die Kohärenzkurven $k(f)$ zwischen Mote 07 und den anderen Motes dargestellt. Dabei sieht man, dass die Kohärenz zu Mote 08 bei allen Frequenzen bei fast 100 % liegt. Zu den Motes 05 und 06 ist dagegen die Kohärenz deutlich geringer.

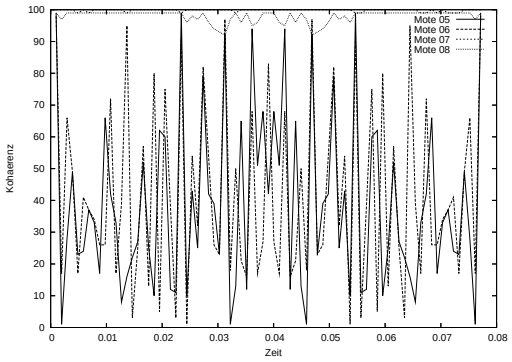


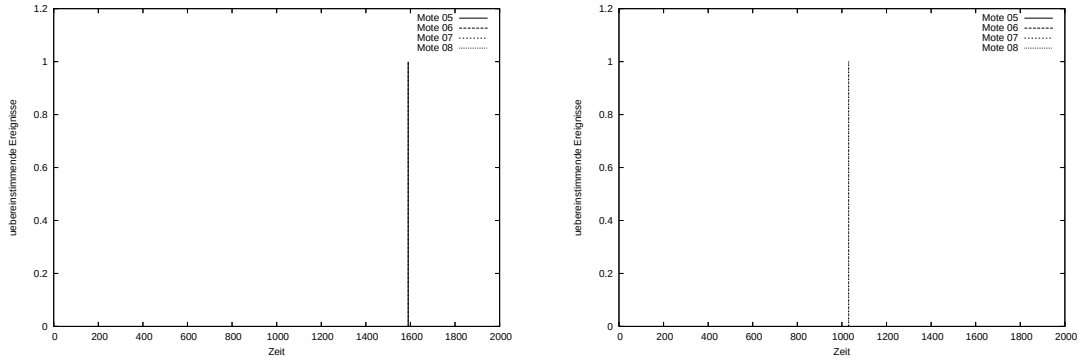
Abbildung 4.16: Beispiel für Komparator „Kohärenz"

4.2.3.2 Ereignisse

Die binären Ergebnisse der Ereignisfilter benötigen spezielle Komparatoren. Es gibt hauptsächlich zwei verschiedene Methoden Ereignisse zu vergleichen. Einmal kann ein Mote versuchen seine detektierten Ereignisse in den detektierten Ereignissen anderer Motes wiederzufinden und so Übereinstimmungen zu erkennen oder auf möglicherweise verschiedene Gruppenzugehörigkeiten aus fehlenden Ereignissen zu schließen. Im Gegensatz zu den bisher vorgestellten Komparatoren ist bei den Ereignis-Komparatoren keine Symmetrie bezüglich Vertauschung der Motes vorhanden, da der Referenzmote nur aktiv wird, wenn er ein Ereignis erkennt, und dann erst die Suche nach Ereignissen eines Vergleichsmotes anstrengt. Das Erkennen eines Ereignisses wiederum hängt nur von den eigenen Messwerten ab. Wenn ein Mote nur bei erkannten Ereignissen Nachrichten versendet, hilft ein solches Vorgehen bei der Einsparung von Energie. Durch den Verlust der Symmetrie kann es vorkommen, dass sich inkonsistente Gruppenkonfigurationen auf verschiedenen Motes bilden. Um diese Gefahr zu vermindern oder ganz zu verhindern, werden im Folgenden an geeigneter Stelle tolerante Algorithmen vorgestellt.

Der Komparator „übereinstimmende Ereignisse" untersucht die Messreihen zweier unterschiedlicher Motes auf das Auftreten eines Ereignisses zur selben Zeit. Erkennen zwei oder mehr Motes Ereignisse zur selben Zeit, wird davon ausgegangen, dass es sich um dasselbe Ereignis handelt. Das Ergebnis dieses Komparators lässt sich als Kurve mit

binärem Wertebereich darstellen. Die logische Eins symbolisiert ein übereinstimmendes Ereignis, also eines, das von beiden verglichenen Motes erfasst wird. In allen anderen Fälle ergibt sich die logische Null. Da auch dieser Komparator wie alle Verlaufskomparatoren eine Ausgangskurve erzeugt, lässt er sich mit Modifikatoren verbinden. Es ist zum Beispiel sinnvoll, einen weiter unten vorgestellten Integral-Modifikator zu verwenden, um die Übereinstimmungen nicht nur zu detektieren, sondern auch zu zählen. Hier gilt: je höher die Zahl der übereinstimmenden Ereignisse ist, desto enger ist die Verwandtschaft. Dadurch fallen vereinzelte Ereignisse, die Asymmetrien verursachen, weniger ins Gewicht und dieser Komparator wird dafür toleranter. Abbildung 4.17 a) zeigt die erkannten Übereinstimmungen für Mote 05. Für Mote 06 (und natürlich auf für Mote 05) liegt eine Übereinstimmung vor, wobei Mote 07 und 08 keine Übereinstimmung aufweisen. In Abbildung 4.17 b) verhält es sich für den Referenzmote 07 genau umgekehrt.

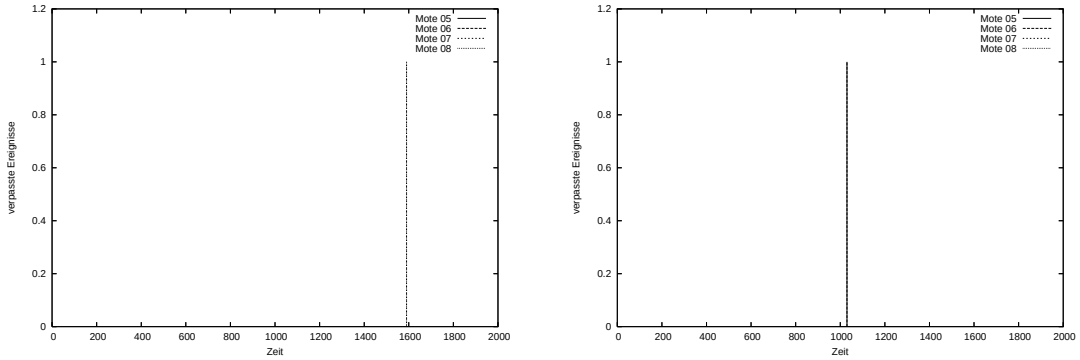


(a) Uebereinstimmende Ereignisse von Mote 05

(b) Uebereinstimmende Ereignisse von Mote 07

Abbildung 4.17: Beispiele für den Komparator „übereinstimmende Ereignisse"

Der Komparator „verpasste Ereignisse" stellt das duale Pendant zum Komparator „übereinstimmende Ereignisse" dar. Er erkennt die Ereignisse eines Motes, die von anderen Motes nicht erfasst werden. Der Wertebereich ist ebenfalls binär, wobei eine logische Eins ein Ereignis kennzeichnet, dass von dem Vergleichsmote nicht erfasst wurde. Für alle anderen Fälle steht die logische Null. Durch die Verwendung eines Integral-Modifikators drückt sich eine nahe Verwandtschaft durch einen kleinen Wert aus. Auch hier hilft diese Verwendung, Asymmetrien verursachende Ereignisse zu tolerieren. In Grafik 4.18 a) erzeugt das Ereignis von Mote 05 eine logische Eins, die für die Motes 07 und 08 nicht stattfanden. Dies bedeutet, dass Mote 05 und Mote 07 bzw. 08 nicht in derselben Region installiert sind. Auf dasselbe Ergebnis kommt Mote 07 nach Abbildung 4.18 b). Hier fehlt das von Mote 07 detektierte Ereignis bei den Motes 05 und 06.



(a) Verpasste Ereignisse von Mote 05

(b) Verpasste Ereignisse von Mote 07

Abbildung 4.18: Beispiele für den Komparator „verpasste Ereignisse”

Der Komparator „prozentual übereinstimmende Ereignisse” arbeitet wie der Komparator „übereinstimmende Ereignisse” mit einem Integral-Modifikator, nur dass zusätzlich noch der prozentuale Anteil zu der Gesamtanzahl der bisherigen Ereignisse des Referenzmotes berechnet wird. Dies bedeutet, dass sowohl die übereinstimmenden als auch alle aufgetretenen Ereignisse gezählt werden und das Verhältnis zueinander bestimmt wird. Dieser Komparator weist ebenfalls keine Symmetrie bezüglich der Vertauschung der Motes auf. Der Verwandtschaftsgrad lässt sich direkt am prozentualen Ergebnis ablesen. Dual zu diesem Komparator ist ein Komparator „prozentual verpasste Ereignisse” denkbar, der statt der übereinstimmenden Ereignisse die verpassten Ereignisse auf die Gesamtzahl abbildet. Dieser würde aber keine neuen Erkenntnisse liefern, da diese Information schon durch den Komparator „prozentual übereinstimmende Ereignisse” erzeugt wird und als 100 % minus „prozentual übereinstimmende Ereignisse” berechnet werden kann. In Abbildung 4.19 sieht man das Ergebnis dieses Komparators für den Referenzmote 07. Solange noch kein Ereignis stattgefunden hat, wird hier von einer hundertprozentigen Übereinstimmung ausgegangen. Genauso gut kann man den Anfang zu nullprozentiger Übereinstimmung definieren. Beim ersten Ereignis teilen sich die Graphen der Motes entsprechend ihrer prozentualen Übereinstimmung auf. Hier behält Mote 08 eine hundertprozentige Übereinstimmung, weil er das Ereignis ebenfalls erfassen konnte. Die Motes 05 und 06 hingegen konnten das Ereignis von Mote 07 nicht messen und weisen deshalb eine nullprozentige Übereinstimmung auf.

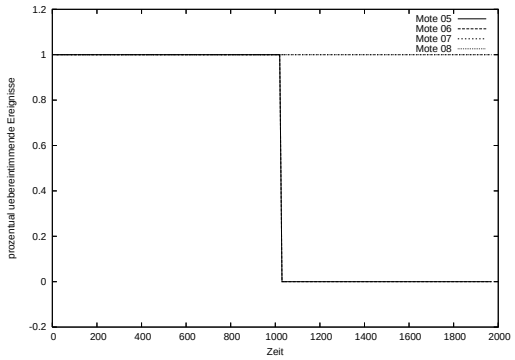


Abbildung 4.19: Beispiel für den Komparator „prozentual übereinstimmende Ereignisse" auf Mote 07

4.2.3.3 Modifikatoren

Bei Modifikatoren handelt es sich um Hilfsmittel der Komparatoren. Sie werten das Ergebnis der Komparatoren $y_{ab}(t)$ durch zusätzliche Verarbeitung zu $\hat{y}_{ab}(t)$ auf. Diese Ausgabe kann erneut als Eingabe $y_{ab}(t)$ eines weiteren Modifikators benutzt werden, um *Modifikatorketten* zu bilden. In der Funktionsweise arbeiten sie wie Filter, nur dass sie für die Verarbeitung der Daten auf dieser Stufe konzipiert sind.

Der Modifikator „Betrag" bildet den Absolutwert $\hat{y}_{ab}(t) = |y_{ab}(t)|$. Da nur beim Komparator „Differenz" negative Werte auftreten können, auf die alleine dieser Modifikator Auswirkungen hat, ist er ausschließlich bei deren Aufbereitung sinnvoll anwendbar. Im Beispiel 4.20 wurden die Differenzen von Mote 07 zu den anderen Motes aus Abbildung 4.13 b) modifiziert. Die Interpretationen bleiben dieselben, nur sind sie jetzt deutlicher für Menschen erkennbar und auch algorithmisch leichter verarbeitbar.

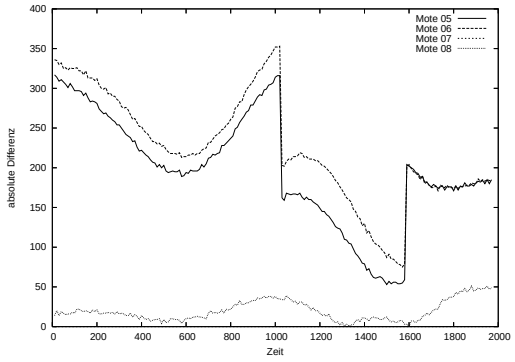


Abbildung 4.20: Beispiel für den Modifikator „Betrag" auf Mote 07

Der Modifikator „Quadrat" multipliziert jeden Wert mit sich selbst $\hat{y}_{ab}(t) = (y_{ab}(t))^2$. In der Funktionsweise erzeugt er ein ähnliches Resultat wie der Modifikator „Betrag", da dadurch ebenfalls nur positive Werte erzeugt werden. Bei einer Nebeneinanderstellung von Abbildung 4.20 und 4.21 ist der Unterschied beider Modifikatoren sichtbar. Beiden Abbildungen liegen dieselben Ausgangsdaten (Abbildung 4.13 b)) zugrunde. Beim Modifikator „Quadrat" treten große Differenzen deutlicher hervor und kleine Differenzen verschwinden beinahe. Eine spätere Entscheidung über die Ähnlichkeiten zu verschiedenen Motes lässt sich damit leichter treffen.

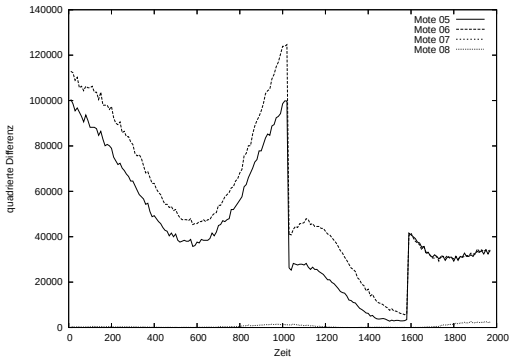


Abbildung 4.21: Beispiel für den Modifikator „Quadrat" auf Mote 07

Der Modifikator „Integral" dient zur Aufsummierung aller in der Vergangenheit liegenden Werte $\hat{y}_{ab}(t) = \sum_{i=0}^{t} y_{ab}(i)$. Dadurch erhalten die modifizierten Ergebnisse von Komparatoren einen Erinnerungseffekt. Vergangene Ergebnisse fließen genau wie aktuelle Ergebnisse in das Resultat dieser Verarbeitungsstufe ein. Dies führt zu einer soliden, fehlertoleranten Grundlage für die spätere Entscheidung über die Gruppenzugehörigkeit. Durch Aufsummieren der Vergleiche zwischen den Motes divergieren die Resultate von verschiedenen Motes unterschiedlich stark auseinander. Im Beispiel 4.22 ist das Integral des Produkts der Steigungen von Mote 05 mit den anderen Motes dargestellt. Da die Steigungen der Motes 05 und 06 öfter übereinstimmen als die Steigungen von Mote 05 im Vergleich zu Mote 07 oder 08, ergibt sich ein größere Integralwert am Schluss der Messung.

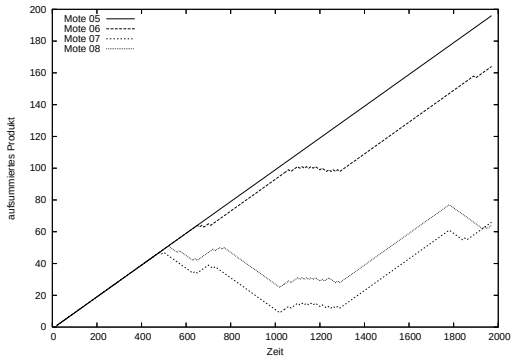


Abbildung 4.22: Beispiel für den Modifikator „Integral" auf Mote 05

Der Modifikator „Negativ" ändert das Vorzeichen für jeden Wert $\hat{y}_{ab}(t) = -y_{ab}(t)$. Dadurch werden große Werten zu kleinen Werte und umgekehrt. Aus technischen Design-Gründen in der nächsten Verarbeitungsstufe kann diese Inversion für die Gruppenzugehörigkeitsprognose notwendig sein.

4.2.4 Signalstärke

Zusätzlich zu der Auswertung ihrer Sensoren besitzen die meisten Motes die Möglichkeit, die Signalstärke für eine eingehende Nachricht zu erfassen. Grob gesagt, gilt dabei: Je weiter weg der Sender ist, desto schwächer ist das Signal. Das Verhältnis zwischen Entfernung und gemessener Signalstärke ist aber nicht linear und lässt sich somit schwer in einer exakten Formel ausdrücken. Effekte wie Interferenzen, Reflexionen besitzen ein großen Einfluss auf die gemessene Signalstärke. Außerdem müssen zusätzliche Effekte bei der Abstrahlung beachtet werden, die durch die Aufbauten neben der Antenne auf den Motes entstehen. Dies äußert sich in einer nicht radial-symmetrischen Empfangsleistung um den Sendemote herum [GKW+02]. Setzt man geographisch begrenzte Regionen voraus, kann man davon ausgehen, dass ein Signal zwischen zwei Regionen merkbar abgeschwächt ist. Dies kann entweder aus der geographischen Entfernung oder von Hindernissen, wie z. B. Wänden, zwischen zwei Regionen resultieren. Als zusätzliches Kriterium kann also auch die Signalstärke von Nachrichten benutzt werden.

Um die Signalstärke für das vierstufige Verarbeitungsmodell nutzbar zu machen, müssen sie entsprechend aufbereitet werden. Bei der empfangenen Signalstärke handelt es sich um eine Kennzahl, die für jede Nachricht ermittelt wird und umso größer ist, je besser das Nachrichtensignal empfangen wurde. Nachrichten, die mit einer niedrigeren Leistung versendet werden, sind besonders von einer Abschwächung betroffen. Aber gerade diese Nachrichten sind für die Einschätzung der Gruppenzugehörigkeit wichtig. Deshalb muss ihnen bei der Berechnung der Gruppenkonfiguration eine größere Bedeutung zugeschrieben werden. Dies geschieht durch eine Gewichtung, die invers zur Sendeleistung ist. Damit bekommen Motes, die selbst mit einer niedrigen Sendeleistung noch gut empfangen werden eine hohe, gewichtete Signalstärke und somit eine gute Verwandtschaft bescheinigt. Dagegen erreichen Motes, die mit hoher Sendeleistung zwar

genauso gut empfangen werden, nur einen entsprechend niedrigeren Verwandtschaftsgrad. Dieses Modell funktioniert nur, wenn alle Motes regelmäßig Nachrichten senden. Ist dies nicht der Fall, können Motes aus anderen Regionen das Netz mit Nachrichten fluten und durch die Aufsummierung trotz einer geringen Signalstärke einen großen Verwandtschaftsgrad erlangen. Verloren gegangene Nachrichten hingegen können durch die Aufsummierung toleriert werden.

Auf Basis dieser gewichteten Signalstärken können in den Verarbeitungsstufen ab der dritten Stufe die erzeugten Verwandtschaftsbeziehungen parallel zu denen der Sensordaten für eine Gruppeneinteilung genutzt werden.

4.2.5 Stufe 3: Gruppenbildung

Basierend auf den errechneten Verwandtschaftsgraden entscheidet in der dritten Stufe jeder Mote über seine Gruppenzugehörigkeit. Voraussetzung aller folgenden Algorithmen ist das Vorhandensein von mindestens zwei Regionen. Für ein gutes Ergebnis müssen sich die Verwandtschaftsverhältnisse zwischen einem Mote mit anderen in derselben Region deutlich von denen anderer Regionen abheben. Vergleiche mit Erinnerungseffekt aus der vorherigen Verarbeitungsstufe können dabei hilfreich sein. Je nach betrachteter Messreihe ist möglicherweise eine Messung über einen längeren Zeitraum erforderlich. Das Ergebnis dieser Stufe ist eine Gruppenkonfiguration, in der der Grad der Gruppenzugehörigkeit zu jedem Mote einheitlich ausgedrückt wird.

In diesem Abschnitt wird das bekannte Beispiel aus Abbildung 4.6 a) fortgesetzt. Dessen Messwerte werden direkt mit dem Komparator „Differenz" verglichen und mit den Modifikatoren „Betrag", „Negativ" und „Integral" aufgewertet. Das nummerische Ergebnis ist in Tabelle 4.1 aufgeführt. Dabei gibt je eine Zeile die Berechnungen eines Motes wieder. In den Spalten stehen die Resultate bezüglich eines Motes.

	05	06	07	08
05	0	-5373	-38805	-35315
06	-5373	0	-44112	-40622
07	-38805	-44112	0	-3910
08	-35315	-40622	-3910	0

Tabelle 4.1: Beispiel für die Entscheidungsfindung eines Separators

Der Separator „Minimum-Maximum" sucht sich die zwei Motes mit minimalem und maximalem Verwandtschaftsgrad heraus. Als zugehörig zur eigenen Gruppe werden alle Motes betrachtet, deren Verwandtschaftsgrad über der Mitte zwischen Minimum und Maximum liegt. Einen Erfolg verspricht diese Methode, wenn Verwandtschaftsgrade auf zwei verschwommenen, aber nicht überlappenden Niveaus liegen. Ein Niveau mit hoher Übereinstimmung entspricht den Motes in derselben Region und ein Niveau mit niedriger Übereinstimmung entspricht den Motes in anderen Regionen. Durch das zusätzliche Einführen eines Parameters, der statt der Mitte einen Schwellenwert angibt, kann diese Methode verfeinert werden. Dieser Separator hat den großen Vorteil, dass er sich leicht auf Motes implementieren lässt, da der Speicherverbrauch gering und konstant ist. Der

Algorithmus kann auch einfach umgesetzt werden. Probleme können allerdings entstehen, wenn es zusätzlich zu den Regionen mit maximalem und minimalem Verwandtschaftsgrad noch weitere Regionen gibt, deren Übereinstimmung sich nicht in eines der beiden Niveaus zuordnen lässt.

In der Beispieltabelle 4.1 ergibt sich für Mote 05 das Minimum zu -38805 bei Mote 07 und das Maximum zu 0 bei Mote 05 (der Mote selbst). Die Mitte zwischen diesen zwei Werten liegt bei -19402,5. Nur bei den Motes 05 und 06 liegen die Werte über dieser Mitte, also werden sie in die Gruppe aufgenommen, was in Tabelle 4.2 durch eine Eins dargestellt wird. Für die anderen Motes wird analog verfahren.

Der Separator „Mittelwert" betrachtet die Verwandtschaftsgrade zu allen Motes. Aus diesen wird ein Mittelwert berechnet, der als Schwellenwert für eine Entscheidung dient. Motes mit größerem Verwandtschaftsgrad werden als zugehörig zur eigenen Gruppe gezählt, Motes mit kleinerem Verwandtschaftsgrad dagegen nicht. Das Einsatzszenario ähnelt dem für „Minimum-Maximum", nur dass durch die Einbeziehung aller Motes keine ausgeprägten Niveaus nötig sind. Ziel dieser Methode ist eine größere Toleranz, die durch einen leicht komplexeren Algorithmus erkauft wird. Problematisch ist der Einsatz mit vielen Motes entweder innerhalb oder außerhalb der eigenen Region, da dadurch der Schwellenwert zu weit in die eine oder andere Richtung beeinflusst werden kann und folglich Fehlentscheidungen getroffen werden.

Für Mote 05 aus der Beispieltabelle 4.1 liegt der abgeschnittene Mittelwert bei -19873. Die Motes 05 und 06 liegen mit ihren Werten über diesem Schwellenwert und werden deshalb als zugehörig zu Gruppe erklärt, was in Tabelle 4.2 durch eine Eins für beide Motes symbolisiert wird.

	05	06	07	08
05	1	1	0	0
06	1	1	0	0
07	0	0	1	1
08	0	0	1	1

Tabelle 4.2: Ergebnis des Beispiels für die Separatoren „Minimum-Maximum" und „Mittelwert"

Der Separator „Positiv" ist nur in Verbindung mit „negativem" Verwandtschaftsgrad sinnvoll. Ein Einsatz ist zum Beispiel die Kombination aus übereinstimmenden und verpassten Ereignissen, in denen die übereinstimmenden positiv und die verpassten negativ gewichtet werden. Dadurch kommt es bei mehr verpassten als übereinstimmenden Ereignissen zu einem negativen Verwandtschaftsgrad, bei mehr übereinstimmenden zu einem positiven Verwandtschaftsgrad. Dieser Separator gruppiert die Motes mit einem positiven Verwandtschaftsgrad. Die Umsetzung kann sehr effizient und ohne zusätzlichen Speicher realisiert werden, besitzt aber aufgrund der Ansprüche auf die darunter gelegene Verarbeitungsstufe ein eingeschränktes Einsatzgebiet.

Der Separator „Prozent" berechnet den prozentualen Gruppenzugehörigkeitsgrad. Im Gegensatz zu den anderen Separatoren versucht er keine endgültige Einteilung in Gruppen sondern überlässt dies der nächsten Verarbeitungsstufe. Dazu werden alle Verwandtschaftsgrade in einen prozentualen Gruppenzugehörigkeitsgrad umgerechnet. Dieser Grad der Zugehörigkeit berechnet sich als Projektion jedes Verwandtschaftsgrades in den Bereich zwischen Minimum und Maximum der Verwandtschaftsgrade. Die Implementierung ist genauso einfach umsetzbar wie bei den anderen Separatoren, nur muss bei den Projektionen die nötigen Divisionen beachtet werden, die auf den Motes ohne Hardwareunterstützung einen vergleichsweise höheren Rechenaufwand bedeuten. Aus einem anderen Blickwinkel können die Ergebnisse der anderen Separatoren als Extremfall mit hundertprozentigem und nullprozentigem Gruppenzugehörigkeitsgrad gesehen werden. Für die Beispieltabelle 4.1 ergeben sich die prozentualen Zugehörigkeiten in Tabelle 4.3.

	05	06	07	08
05	100	87	0	9
06	88	100	0	8
07	13	0	100	92
08	14	0	91	100

Tabelle 4.3: Ergebnis des Beispiels für den Separator „Prozent" in Prozent

4.2.6 Stufe 4: Gesamtergebnisse

Bis hin zur dritten Verarbeitungsstufe werden jeweils einzelne Messreihen analysiert. Es ist die Aufgabe der vierten Stufe diese Zwischenergebnisse zusammenzuführen. Die hier vorgestellten Algorithmen haben als ersten Schritt eine gemeinsame Vorgehensweise, in der die Gruppenkonfigurationen jedes einzelnen Kanals zusammenaddiert werden. Im zweiten Schritt ist aber jeder Lösungsalgorithmus individuell. Die Lösungen können optional durch Einbeziehung von gespeicherten, vorherigen Lösungen oder Lösungen von anderen Motes noch stabilisiert werden.

Allgemeine Vorgehensweise Die einzelnen Kanäle liefern möglicherweise inkonsistente Gruppenkonfigurationen, die aus Unzulänglichkeiten oder Messfehlern einzelner Kanäle resultieren. Es gilt nun daraus eine konsistente Lösung zu erstellen. Dazu werden alle Konfigurationen aufsummiert, in denen der Referenzmote selbst zur Gruppe gehört. Jeder Kanal liefert seine Gruppenkonfiguration als eine Zeile für eine Tabelle, deren Spalten den Grad der Gruppenzugehörigkeit aller Motes bezüglich des Referenzmotes repräsentieren. Die Grade sind 1 für „zugehörig zur Gruppe" und 0 für „nicht zugehörig zur Gruppe" bei allen Separatoren bis auf den Separator „Prozent", wo die Zugehörigkeit prozentual ausgedrückt wird. Durch spaltenweises Aufsummieren wird die Anzahl der Kanäle bestimmt, die für jeden Mote eine Zugehörigkeit zum Referenzmote signalisieren.

In der Beispieltabelle 4.4 sind einzelne Kanäle aufgeführt, wie sie von Mote 05 stammen könnten. Kanal 1 kann durch den Separator Mittelwert aus dem bisherigen Beispiel

entstanden sein. Kanal 2 kann von einem anderen Sensor stammen, der ungenauere Daten liefert. Bei Kanal 3 kann es sich um einen Ereignis-erkennenden Kanal handeln, der bisher keine Entscheidung treffen konnte. Die Kanäle 4 bis 6 dienen als Beispiele für die später vorgestellten Erweiterungen zu mehr Stabilität in der Lösung. Bei Kanal 6 fehlt Mote 05 in der Gruppenkonfiguration. Aus diesem Grund wird die komplette Zeile nicht mitsummiert.

	05	06	07	08
Kanal 1	1	1	0	0
Kanal 2	1	1	0	1
Kanal 3	1	1	1	1
Kanal 4	1	1	0	0
Kanal 5	1	1	0	0
~~Kanal 6~~	~~0~~	~~1~~	~~1~~	~~0~~
Summe	5	5	1	2

Tabelle 4.4: Beispiel für die allgemeine Vorgehensweise zur Lösungsbildung

Die Lösung „Minimum-Maximum" fasst alle Motes zusammen, die von mehr Kanälen als der Mitte zwischen dem Minimal- und Maximalwert der Summen als zugehörig gemeldet werden. Es wird hier davon ausgegangen, dass es nur zu vereinzelten falschen Gruppenkonfigurationen durch Fehler kommt. Das bedeutet, dass durch die Aufsummierung zwei Niveaus entstehen: ein hohes Niveau, das allen Motes entspricht, die zur Gruppe gehören, und ein niedriges Niveau, das allen Motes entspricht, die nicht zur Gruppe gehören. Diese Niveaus können durch Fehleinschätzungen eine gewisse (kleine) Breite besitzen. Durch die Mitte zwischen dem niedrigen Niveau (Minimalwert) und dem hohen Niveau (Maximalwert) werden beide Niveaus getrennt. Dadurch wird eine Fehlertoleranz gegenüber den Entscheidungen der Separatoren eingebaut. Die Anzahl der tolerierten Fehler berechnet sich zu der halben Differenz zwischen Minimal- und Maximalwert der Summen. Je mehr Kanäle vorhanden sind, desto höher kann der Maximalwert der Summe sein und desto höher ist auch die Fehlertoleranz. In der Praxis hängt aber die Anzahl der Kanäle von der Anzahl der vorhandenen Sensoren ab, was nur eine beschränkte Fehlertoleranz zulässt. Die Implementierung dagegen lässt sich für Motes effizient umsetzen.

Im Beispiel liegt das Minimum bei 1 und das Maximum bei 5. Die Mitte dazwischen ist 3. Also gehören nur die Motes 05 mit 5 Kanälen und 06 mit ebenfalls 5 Kanälen zur Gruppe. Gäbe es die Kanäle 4 bis 6, die zur Stabilisierung vorgesehen sind, nicht, läge das Maximum bei 3 und die Mitte bei 1,5. was zur Folge hätte, dass Mote 08 fälschlicherweise in die Gruppe aufgenommen würde.

Die Lösung „Hälfte" gruppiert alle Motes, die in mehr Kanälen zugehörig sind als die Hälfte des Maximalwerts der Summen. Dabei wird ebenfalls von zwei Niveaus als Ergebnis der Aufsummierung ausgegangen: das obere entspricht den Motes in der Gruppe und das untere entspricht den Motes außerhalb der Gruppe. Im Gegensatz zur Lösung „Minimum-Maximum" wird das untere Niveau auf Null festgelegt, was absolut keiner

Zugehörigkeit entspricht. Dies wäre im fehlerfreien Fall auch das gefundene Minimum. Die Fehlertoleranz lässt sich genau wie bei der Lösung „Minimum-Maximum" berechnen, ist aber durch die Festlegung des Minimums auf Null größer (oder gleich). Trotzdem ergeben sich ähnliche Probleme mit einer geringen Anzahl von Kanälen.

Das Maximum im Beispiel liegt bei 5, dessen Hälfte bei 2,5. Also gehören auch hier die Motes 05 und 06 zur Gruppe. Auch hier würde das Fehlen der Stabilisierungskanäle 4 bis 6 zu einer Fehlentscheidung für Mote 08 führen.

Die Lösung „Mittelwert" bildet den Mittelwert der Summen. Gruppiert werden alle Motes, deren Summe größer als dieser Wert ist. Diese Lösung stellt ein sehr fehlertolerantes Verfahren dar, das durch die Einbeziehung aller Werte keine ausgeprägten Niveaus benötigt. Es besteht jedoch die Gefahr, dass sich der Mittelwert von einer extrem ungleichen Anzahl von Motes innerhalb oder außerhalb der eigenen Region zu sehr beeinflussen lässt, so dass zu wenige bzw. zu viele Motes in die Gruppe aufgenommen werden. Dieser Algorithmus lässt sich beinahe so effizient implementieren, wie die bisherigen Lösungsverfahren.

Aus den Beispieldaten ergibt sich ein Mittelwert von 3,25. Nur die Motes 05 und 06 weisen mehr Kanäle auf und gehören deshalb zur Gruppe von Mote 05.

Die Lösung „Maximale Zugehörigkeit" erfolgt in zwei Schritten. Im ersten Schritt werden alle Motes identifiziert, deren Summe der maximalen Summe entspricht, darunter auch der analysierende Mote selbst, der selbstverständlich diese Forderung erfüllt. Im Beispiel ist dies der Referenzmote 05, aber auch der Mote 06. Im zweiten Schritt wird aus den von allen Motes so identifizierten Motes die transitive Hülle mit dem Warshall Graphalgorithmus [War62] gebildet, wobei die Motes Knoten darstellen. (Ungerichtete) Kanten werden vom Referenzmote zu allen identifizierten Motes gezogen. Durch Bildung der transitiven Hülle werden für jeden Knoten Kanten zu anderen Knoten so eingefügt, dass direkte Verbindungen entstehen, wenn es Verbindungen über mehrere Knoten bereits gibt. Dadurch entstehen Zusammenhangskomponenten im Graph, die die Gruppen reflektieren. Zu beachten ist, dass das Bilden der transitiven Hülle wesentlich komplexerer als die bisherigen Lösungsalgorithmen ist und dazu noch einen großen Kommunikationsaufwand zwischen den Motes verursacht. Weil der Referenzmote die maximale Summe festlegt, kann außerdem das Problem entstehen, dass kein anderer Mote identifiziert wird. Falls dies bei allen Motes einer Region oder in einer nachteiligen Konstellation geschieht ist die Folge, dass sich diese Problemmotes nicht in die Gruppe integrieren, im Extremfall keine Gruppe in der Region bildet. Als Problemlösung wird vorgeschlagen, den Referenzmote bei der Bestimmung der maximalen Summe außer Acht zu lassen. Dadurch ist garantiert, dass ein Vergleichsmote identifiziert wird. Diese so modifizierte Lösung eignet sich besonders für Netzwerke, in denen zwei oder drei Motes pro Region installiert sind. Durch die unbedingte Wahl eines verwandten Motes versagt er aber bei Regionen mit nur einem Mote. Auch bei einer Region mit mehr als drei Motes ist eine fehlerhafte Arbeitsweise nicht auszuschließen. Zusätzliche Fehlertoleranz kann durch die Einführung eines Parameters, der beim Identifizieren der Motes eine niedrigere Summe zulässt, realisiert werden. Ergibt sich durch die Maximumsuche eine Gruppenkonfiguration wie in der Beispieltabelle 4.5, so werden durch die Bildung

der transitiven Hülle die richtigen Gruppenkonfigurationen zu allen Motes propagiert.

	05	06	07	08
05	1	1	0	0
06	0	1	0	0
07	0	0	1	0
08	0	0	1	1

Tabelle 4.5: Beispiel für eine Gruppenkonfiguration nach der Maximumsuche

Verbesserung der Lösung Jeder Mote berechnet auf die vorgestellte Weise nur eine lokale Lösung. Um eine global gültige Lösung zu erlangen müssen alle lokal gefundenen Lösungen ausgetauscht werden und daraus eine Lösung errechnet werden, die für alle Motes gilt. Dazu kann ebenfalls eines der vorgestellten Lösungsverfahren angewendet werden. Diese operieren statt auf den Gruppenprognosen einzelner Kanäle auf den Gruppenprognosen von Motes. Dies ist eine zuverlässige Methode, um inkonsistente Gruppenkonfigurationen zu vermeiden, die von Asymmetrien bei der Ereigniserkennung stammen.

Um eine Lösung zu finden, die eine größere Fehlertoleranz aufweist, können statt nur der lokalen Prognosen der eigenen Kanäle auch die Prognosen von Kanälen anderer Motes einbezogen werden. Doch dies bedeutet einen erheblichen Kommunikationsaufwand zwischen den Motes. In der Tabelle 4.4 ist diese Erweiterung beispielhaft dargestellt. Kanal 5 und Kanal 6 können zwei Kanäle von Mote 06 darstellen, die an Mote 05 übermittelt wurden. Kanal 6 kann bei der Aufsummierung nicht berücksichtigt werden, da Mote 05 nicht als zur Gruppe zugehörig gezählt wird.

Für eine zeitlich stabile Lösung besteht die Möglichkeit, ähnlich wie beim Finden einer globalen Lösung, eine größere Anzahl von Kanälen zur Lösungsbildung zu verwenden. Dazu werden regelmäßig gespeicherte Gruppenkonfigurationen von Kanälen zusätzlich zu den aktuellen Gruppenkonfigurationen als Summanden verwendet. So liegen den Lösungsverfahren mehr Informationen für eine Entscheidung zugrunde, die durch die Aufsummierung über einen längeren Zeitraum Fehler tolerieren können. Kanal 4 in der Beispieltabelle 4.4 könnte eine solche Erweiterung darstellen.

Da beide Verbesserungsverfahren unabhängig voneinander sind, lassen sie sich in beliebiger Reihenfolge kombinieren.

4.3 Verfahren aus der Biologie

Auch in fachfremden Bereichen besteht die Notwendigkeit, experimentelle Messungen in Gruppen zu unterteilen. In der Biologie werden zum Beispiel Gruppierungsverfahren eingesetzt, um ein untersuchtes Gen oder einen DNA-Strang mit bereits bekannten Analysen zu vergleichen [ESBB98, SCSS05]. Dies wird zur Funktionsanalyse des untersuchten Gens verwendet oder auch um Tumore schnell zu klassifizieren.

Mit dem Programm Cluster 3.0 [HINM04] lassen sich Gen- oder DNA-Analysen in Cluster zusammenfassen. Jedem Gen wird ein Vektor $\vec{x}$ zugeordnet, der die Resultate

von zeitlich aufeinander folgenden Experimenten x_i wiedergibt. Um Verwandtschaftsverhältnisse zwischen zwei Genen herzustellen, werden Distanzen zwischen ihren beiden repräsentierenden Vektoren berechnet. Dabei arbeitet das Programm mit einem von den acht unten beschriebenen Berechnungsvorschriften.

Das Grundprinzip des Programms besteht darin, die einzelnen Gene zufällig einer durch den Benutzer vorgegebenen Anzahl von Clustern zuzuordnen. Danach werden wiederholt die zwei Schritte

1. Bilden des mittleren Vektors aller Gene in jedem Cluster;
2. Neuzuordnung jedes Gens in den Cluster, dessen mittlerer Vektor am nächsten ist;

ausgeführt. Für die Berechnung der paarweisen Entfernung zwischen zwei Vektoren können unterschiedliche Algorithmen eingesetzt werden. Da sich das Verfahren auf Zufallsordnungen stützt, sind mehrere Ausführungen notwendig, um falsche Lösungen zu vermeiden. Je öfter eine Lösung gefunden wird, desto wahrscheinlicher handelt es sich um die optimale Lösung.

Die Berechnungsvorschriften der Entfernung r zwischen zwei Vektoren $\vec{x}$ und $\vec{y}$ der Länge n stammen teilweise aus der Statistik aber auch aus der Geometrie.

1. Der Pearson-Korrelationskoeffizient [Con71] „ist ein dimensionsloses Maß für den Grad des linearen Zusammenhangs [...] zwischen zwei mindestens intervallskalierten Merkmalen" [Wik]. Er berechnet sich mit $r_{Pearson} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i-\bar{x}}{\sigma_x}\right)\left(\frac{y_i-\bar{y}}{\sigma_y}\right)$, wobei $\bar{x}$ bzw. $\bar{y}$ die Mittelwerte und σ_x bzw. σ_y die Standardabweichung von x_i bzw. y_i sind. Der Wertebereich liegt zwischen 1 und -1, wobei 1 komplette Übereinstimmung bedeutet und -1 bei komplett gegenläufigen Variablen auftritt. Dies ist der Fall, wenn die eine Variable eine Spiegelung der anderen ist.
2. Der unzentrierte Pearson-Korrelationskoeffizient $r_{uncentered}$ stellt ein Spezialfall des Pearson-Korrelationskoeffizienten dar. Es wird die Annahme getroffen, dass $\bar{x} = \bar{y} = 0$ gilt, auch wenn dies nicht gilt. Damit führt der unzentrierten Korrelationskoeffizient zu anderen Ergebnissen.
3. Der absolute Pearson-Korrelationskoeffizient mit $r = |r_{Pearson}|$. Dadurch werden auch gegenläufige Variablen, also Spiegelungen, ebenfalls mit einer hohen Korrelation ausgezeichnet.
4. Der absoluter unzentrierter Pearson-Korrelationskoeffizient mit $r = |r_{uncentered}|$.
5. Der Spearmans Rangkorrelationskoeffizient [Con71] ist ein parameterloses Maß für den Zusammenhang zwischen zwei Variablen. Er wird mit $r = 1 - \frac{6\sum_i d_i}{n\cdot(n^2-1)}$ berechnet, wobei $d_i = rang(x_i) - rang(y_i)$ ist. Der Rang des kleinsten Elements beträgt 1. Der Rang des nächstgrößeren, also des zweitkleinsten Elements beträgt 2, usw...
6. Kendall's τ [Con71] ist eine Besonderheit des Spearman Rangkorrelationskoeffizienten. Kendall's τ betrachtet statt des Ranges einfach die relative Stellung zweier Komponenten. Es wird berechnet, indem alle $\frac{1}{2}n\cdot(n-1)$ verschiedenen Paare (x_i, y_i)

auf Übereinstimmung in der Größenrelation zwischen $x_i > x_j$ und $y_i > y_j$ getestet werden. Übereinstimmungen werden als n_c, Unstimmigkeiten als n_d, Fälle mit $x_i = x_j$ als n_y, Fälle mit $y_i = y_j$ als n_x und Fälle mit $x_i = x_j$ und $y_i = y_j$ nicht gezählt. Der Korrelationskoeffizient berechnet sich dann als $r = \frac{n_c - n_d}{\sqrt{n_c+n_d+n_x}\sqrt{n_c+n_d+n_y}}$.

7. Die Euklidische Distanz gibt den Abstand zweier Punkte in einem n-dimensionalen euklidischen Raum mit $r = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2$ an.

8. Die Manhattan-Distanz beschreibt die Entfernung auf einem n-dimensionalen Raster mit $r = \sum_{i=1}^{n}|x_i - y_i|$.

Da dieses Programm mit seinen Berechnungsvorschriften für das Clustering von Genanalysen ausgelegt ist, müssen Änderungen vorgenommen werden, um es für ein Clustering von Motes auf Basis der Sensordaten nutzbar zu machen. Dazu wird jeder Mote als ein Gen aufgefasst. Seine Messwerte der verschiedenen Sensoren werden als die Ergebnisse der Experimente interpretiert und füllen somit einen Vektor $\vec{x}$.

Ein großer Nachteil dieses Verfahrens beruht auf der Tatsache, dass es einen zentralen Algorithmus besitzt, der sich nicht auf Motes verteilen lässt. Außerdem sind die Berechnungsvorschriften teilweise sehr komplex, was durch die geforderte mehrfache Ausführung wegen dem Einfluss von Zufall einen erheblichen Rechenaufwand darstellt. Dies spricht abermals für eine zentrale Ausführung, was einen dafür notwendigen Nachrichtenaufwand zur Folge hat. Bei entsprechend großen Netzen und der begrenzten Sendereichweite eines Motes heißt dies, dass Nachrichten über mehrere Stationen zu einer Basisstation weitergeleitet werden müssen.

Dadurch, dass die Anzahl der Cluster vorgegeben werden muss, muss eine genaue Kenntnis über die Anzahl der Regionen vorhanden sein. Diese Anzahl ist dann auch über die komplette Einsatzzeit des Sensornetzes fest und kann nicht dynamisch variieren.

5 Analyse

Dieses Kapitel beschreibt die Analyse der vorgestellten Gruppierungsverfahren an Hand realer Messdaten und die Erforschung von geeigneten Kombinationen aus Filtern, Komparatoren, Modifikatoren, Separatoren und Lösungsverfahren. Die verwendeten Messdaten stammen aus eigener Sammlung, wobei das Büroszenario aus Kapitel 4.1.1 als Umgebung dient. Als unabhängige Datenquellen werden die öffentlich zugänglichen Messreihen von Intels Berkeley-Forschungslabor [Int] herangezogen.

5.1 Datenerfassung

Um die Gruppierungsverfahren aus Kapitel 4 auf ihre Anwendbarkeit in einer realen Umgebung zu untersuchen, müssen Messwerte gesammelt werden. Zur Erfassung der eigenen Daten werden TMote Sky Sensorknoten eingesetzt. Für die darauf ausgeführte Anwendung wird ein Software-Design gewählt, das Sensordaten periodisch misst und im vorhandenen Flash-Speicher ablegt. Zusätzlich werden Signalstärken eingehender Nachrichten im Flash gespeichert. Nach der Erfassungsphase werden diese Daten von einem PC ausgelesen und dort entsprechend ausgewertet.

Auf den TMote Sky kommt eine von Moteiv angepasste Version von TinyOS zum Einsatz, die bereits Funktionalität für das Ansprechen der Sensoren und den Flash bietet. Zusätzlich virtualisiert es den Flash durch Aufteilung in mehrere Sektionen. So ist es möglich, mehreren unterschiedlichen Applikationen exklusiven Zugriff auf ihren Teil zu gewähren. Diese besondere Funktionalität wird zwar nicht benötigt, aber darauf aufbauend arbeitet ein Logger, der das serielle Schreiben und Lesen abstrahiert. Dies lässt sich für das periodische Hinzufügen von Messdaten ausnutzen. Für die speziellen Anforderungen der Evaluation muss dieser Logger erweitert werden, so dass er mehrere Datenströme behandeln kann, um sowohl Sensordaten als auch die Signalstärken eingehender Nachrichten verarbeiten zu können. Dafür wird ein paket-basiertes Design gewählt, das den vorhandenen Logger virtualisiert. Diese Abstraktion nimmt Datenpakete von verschiedenen Datenstromquellen entgegen, versieht sie mit einem Paketkopf, der die Quelle identifiziert und verwendet den Logger zur tatsächlichen Speicherung.

Zusätzlich wird im Paketkopf die Netzwerkzeit zum Zeitpunkt der Speicherung vermerkt, so dass die Analyse am PC erleichtert wird. Bei der Netzwerkzeit handelt es sich um eine Uhrzeit, die für das gesamte Netzwerk gültig ist. Da die Motes nur ihre Systemzeiten kennen, müssen sie beide Zeiten ineinander umrechnen. Dazu ist eine Synchronisation der einzelnen Uhren notwendig. Die bestehende Bibliothek TimeSync [MKSÁL04] übernimmt diese Aufgabe, indem jeder Mote regelmäßig Nachrichten mit der Netzwerkzeit, die aus seiner eigenen Systemuhrzeit berechnet wird, aussendet. Der Mote mit der kleinsten Mote-ID wird zum Referenzmote gewählt und seine Systemzeit als Vorgabe für die Zeitskala gesetzt, die für alle Motes gültig ist. Aus den Gangunterschieden der einzelnen Uhren berechnet jeder Mote die Parameter zur Umrechnung seiner Systemuhrzeit

in die Netzwerkzeit. Dadurch wird eine Genauigkeit von einigen Millisekunden erreicht, was für die spätere Auswertung ausreicht. Als Nebenprodukt der Zeitsynchronisation können die dafür versendeten Nachrichten zur Bestimmung der typischen Signalstärken zwischen Nachbarn herangezogen werden. Für eine differenzierte Betrachtung der Signalstärke variiert der Sender die Sendeleistung vor dem Senden einer Nachricht.

Beim Auslesen der Sensorwerte sollte die Sampling-Periode einerseits möglichst hoch sein, um eine gute Auflösung für das Erkennen von Ereignissen zu erzielen. Andererseits stellt die Kapazität des Flashs eine Begrenzung dar, die eine untere Schranke für die Sampling-Periode vorgibt. Zieht man von der gesamten Kapazität die Größe der benötigten Verwaltungsinformationen für den Logger und die Flashvirtualisierung ab, so bleiben 1 040 302 Byte übrig, welche die Pakete mit einer Größe von 14 Byte pro Messung, die sich aus dem Kopf und den Sensordaten zusammensetzen, aufnehmen müssen. Legt man eine maximale Laufzeit von 7 Tagen zu Grunde, so ergibt sich eine minimale Sampling-Periode von 8,14 s, um den Flash komplett zu füllen. Allerdings müssen zusätzlich die Pakete berücksichtigt werden, die die Signalstärke der Nachrichten von Nachbarn speichern. Wählt man eine Sampling-Periode von 10 s bleibt Platz für 8065 Nachrichtenpakete mit je 24 Byte, was aufgeteilt auf die 30 möglichen Sendeleistungen 268,5 speicherbaren Nachrichten entspricht. Dieses Kontingent muss nochmals auf mehrere Sender aufgeteilt werden. Bei 15 Sendern bedeutet dies, dass ungefähr alle 1200 s eine Nachricht gesendet werden kann.

Für die Sammlung von Messdaten wurden 12 TMote Sky in drei Büroräumen des Informatikgebäudes der Universität Stuttgart verteilt (Abbildung 5.1). Die Motes 13 bis 16 befanden sich zusammen in einem Raum an der Südseite des Gebäudes, der im Folgenden als *blaues Büro* bezeichnet wird und für dessen Motes in Grafiken blau dargestellt sind. Die Motes 05 bis 08 wurden in dem östlich anschließenden Büro platziert, das als *rotes Büro* bezeichnet wird und in Grafiken mit rot dargestellt wird. Diese beiden Räume besitzen eine Klimaanlage. In einem unklimatisierten Raum, dem *grünen Büro*, nordöstlich davon wurden die Motes 09 bis 12 angebracht, die in Abbildungen grünen dargestellt werden. Dieses Zimmer befindet sich auf der Gegenseite des Gangs. Um eine konstante, sichere Energieversorgung zu gewährleisten, wurden die TMote Sky über ihre USB-Schnittstelle von einem zentralen Verteiler pro Raum mit Strom versorgt. Installiert wurden die Motes an Wänden oder auch auf Tischen. Auf eine möglichst großflächige Verteilung wurde Wert gelegt. Die vertikale Verteilung war zweitrangig, reichte aber trotzdem von 70 cm bis 2,5 m über dem Zimmerboden. Die Sammlung begann am 16. Juni 2006 um 18.10 Uhr und dauerte 5 Tage und 16 Stunden und enthält somit Merkmale, die typisch für Sommer sind. Eine Auswertung zu anderen Jahreszeiten kann die Ergebnisse beeinflussen. Nach Abschluss der Erfassungsphase werden die Motes über die USB-Schnittstelle an einen PC angeschlossen, um die gespeicherten Daten auszulesen.

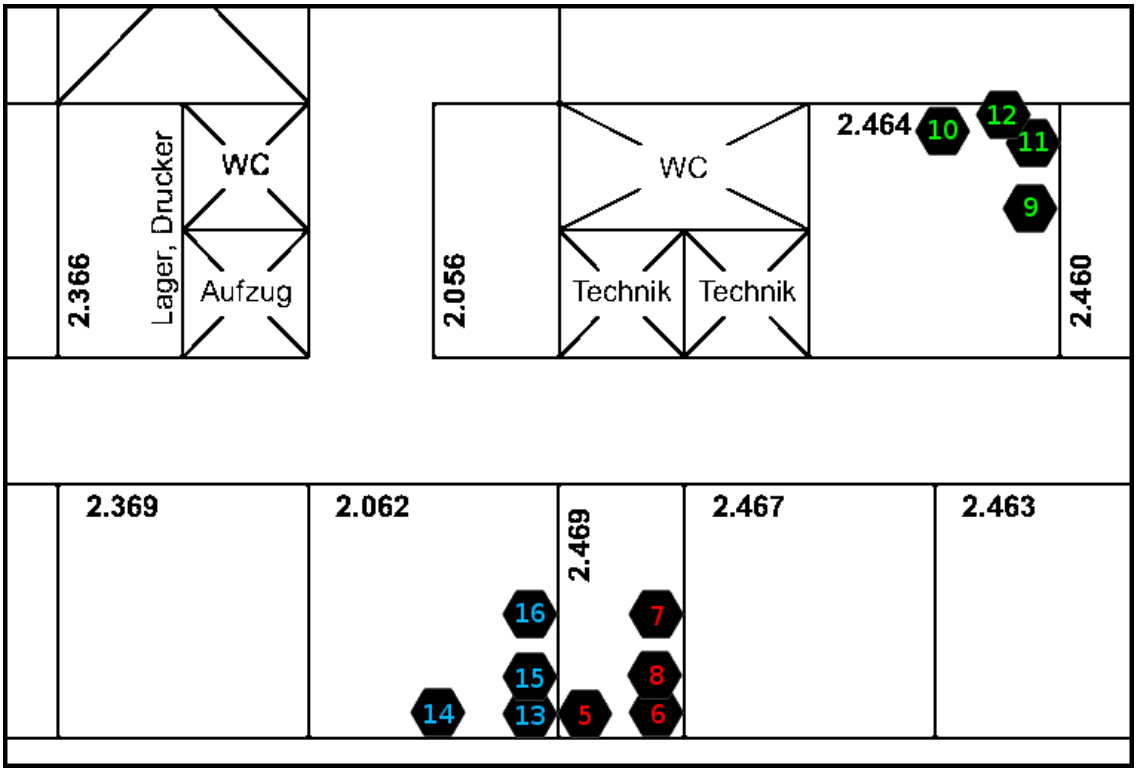


Abbildung 5.1: Moteverteilung in der Fakultät für Informatik

5.2 Intels Berkeley-Forschungslabor

Das Intel-Berkeley-Forschungslabor stellt eine Sammlung von Messdaten zur Verfügung [Int]. Diese Daten werden ebenfalls zur Untersuchung der in Kapitel 4 vorgestellten Analyseverfahren verwendet. Da die Messdaten öffentlich zugänglich sind, können sie weiteren Forschungsarbeiten als Vergleichsgrundlage dienen.

In Intels Berkeley-Labor wurden Temperatur, Luftfeuchtigkeit und Licht gemessen. Für die Messung wurden 54 Mica2Dot mit Weather Boards gemäß Abbildung 5.2 verteilt. Die Messung erfolgte zwischen dem 28. Februar 2004 und dem 5. April 2004. Als Betriebssystem wurde TinyOS mit der Erweiterung TinyDB eingesetzt. TinyDB ist eine Bibliothek für TinyOS, die es einem Benutzer zum Beispiel über einen PC ermöglicht, SQL-ähnliche Anfragen an das gesamte Sensornetzwerk zu stellen. Das Netzwerk wird dadurch zu einer Datenbank der Messwerte abstrahiert. Darüber hinaus bietet TinyDB viele Automatismen, welche die Antworten auf eine Anfrage bereits im Netz auf jedem Sensor optimieren. Mit einer regelmäßigen Abfrage wurde alle 31 s die Umgebung gemessen.

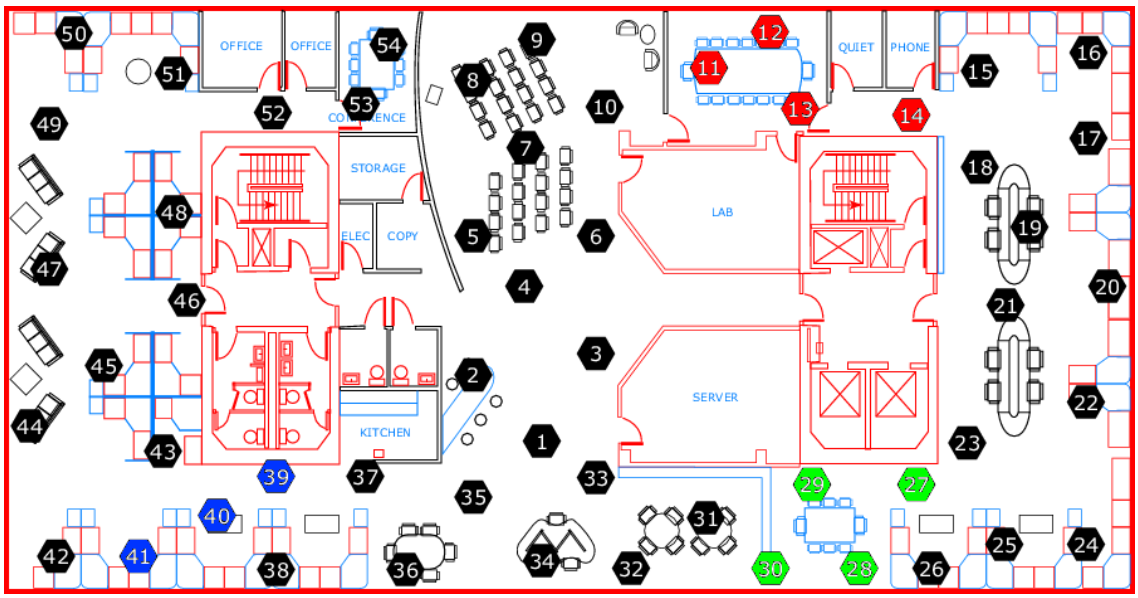


Abbildung 5.2: Moteverteilung im Intel-Berkeley-Forschungslabor [Int]

Für die Analyse der Gruppierungsverfahren müssen die folgenden drei Einschränkungen hingenommen werden.

1. In dem in Abbildung 5.2 gezeigten Szenario sind keine klar voneinander getrennten Regionen vorhanden, wie es vom Systemmodell verlangt wird. Um näherungsweise die geforderten Bedingungen zu schaffen, werden einzelne Gruppen von Motes ausgewählt (Motes 11, 12, 13, 14 & Motes 27, 28, 29, 30 & Motes 39, 40, 41 - in der Abbildung 5.2 rot), die einen ausreichend großen Abstand zueinander besitzen, so dass mehrere Regionen entstehen.

2. Das Datenformat der angebotenen Sammlung liefert Informationen über den messenden Mote, den Zeitpunkt sowie die Epoche der Messung und die gemessenen Daten in einem lesbaren Format. Die Epoche der Messung wird durch einen Zähler realisiert, der sich bei jeder Messung um eins erhöht. Die Temperaturmessungen sind in $^\circ$ C, die Luftfeuchtigkeit in Prozent absolute Luftfeuchtigkeit und das Licht in Lux angegeben.
   Die Werte müssen in die Einheiten der TMote Sky umgerechnet werden, um mit denselben Schritten analysiert werden zu können, wie die Daten aus eigener Sammlung. Dies sind digitalisierte Werte, die aus der analogen Ausgabe der einzelnen Sensoren umgewandelt werden.

3. Die Zeitangaben der Messdaten sind teilweise inkonsistent und können deshalb nicht direkt verwendet werden. Als Ersatz werden die Epochen verwendet und daraus die Zeit interpoliert. Dies kann durch eine einfache Multiplikation des Epochenzählers mit dem Messintervall von 31 s geschehen.

4. Die Daten enthalten nur Informationen über die Wahrscheinlichkeit des Empfangs einer Nachricht von einem bestimmten Absender. Sie können nicht zur Analyse in Signalstärken umgerechnet werden.

5.3 Auswertung

Die Auswertung der in Kapitel 4 vorgeschlagenen Verfahren findet am externen PC statt. Die Implementierung arbeitet mit Daten, die entweder aus eigenen Messungen oder aus den Sammlungen von Intels Berkeley-Forschungslabor stammen. Die Daten aus dem Forschungslabor können nicht direkt übernommen werden, sondern müssen vorher gemäß 5.2 angepasst werden.

5.3.1 Stufe 1: Signalverarbeitung

Analyse der unveränderten Sensordaten In der ersten Stufen werden Rohdaten von den Sensoren für die Weiterverarbeitung aufbereitet. Die Abbildungen 5.3, 5.4, 5.5 und 5.6 zeigen die Sensordaten, wie sie von den verteilten Motes aufgezeichnet wurden. Diese Daten stellen die Grundlage für die folgenden Analysen dar. Beim Betrachten der Luftfeuchtigkeit und der Temperatur fällt sofort ein sehr großer Ausreißer auf, bei dem die Luftfeuchtigkeit rapide sinkt und die Temperatur steigt. Beide Lichtsensoren erreichen zu diesem Zeitpunkt auch ihr Maximum. Diese ungeklärte Anomalie wird nur von Mote 13 gemessen, findet tagsüber statt und dauert etwa eine halbe Stunde. Dieser Mote befindet sich auf einem Schreibtisch in der Nähe eines Fensters. Mote 05 im Nachbarzimmer weist ähnliche, weniger ausgeprägte Merkmale dieser Anomalie auf, die aber im Rahmen seiner Messwerte liegen. Daraus kann man schließen, dass es sich um ein natürliches Ereignis handelt.

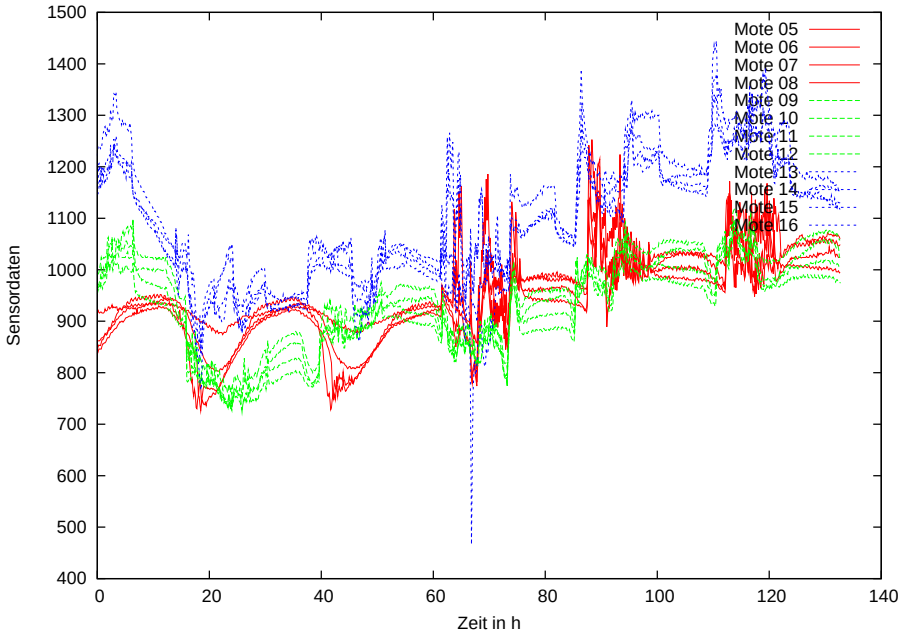


Abbildung 5.3: Sensordaten des Luftfeuchtigkeitssensors

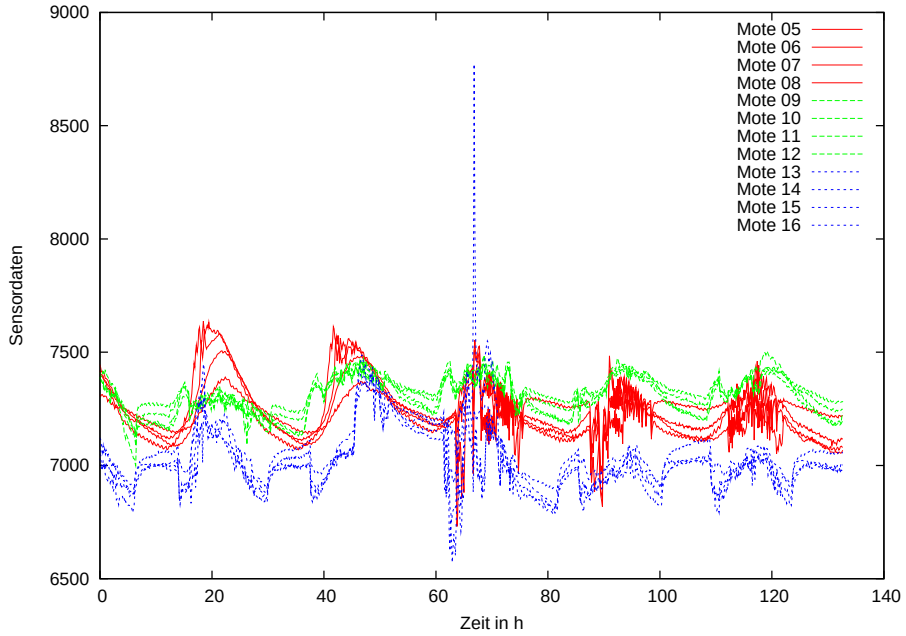


Abbildung 5.4: Sensordaten des Temperatursensors

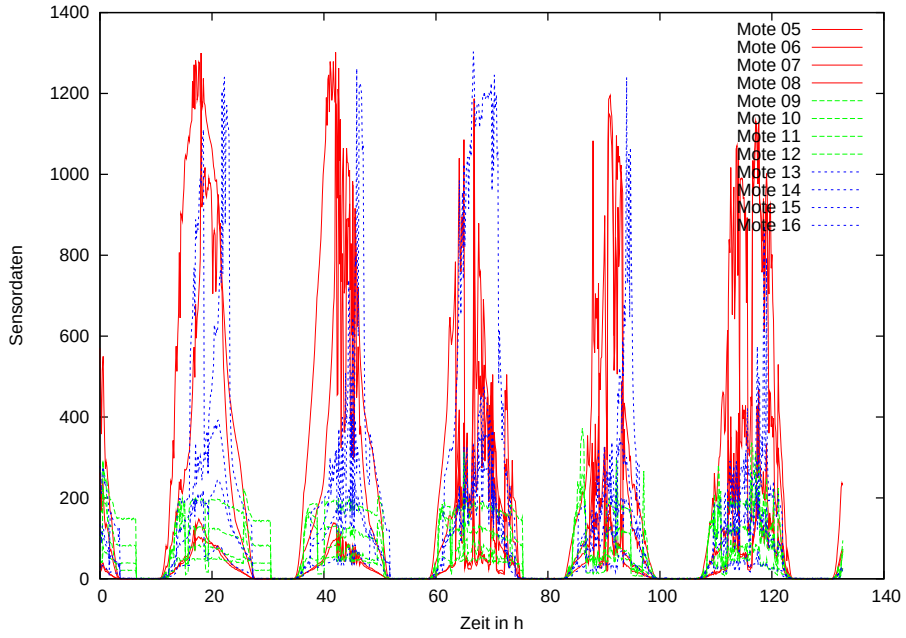


Abbildung 5.5: Sensordaten des Lichtsensors PAR

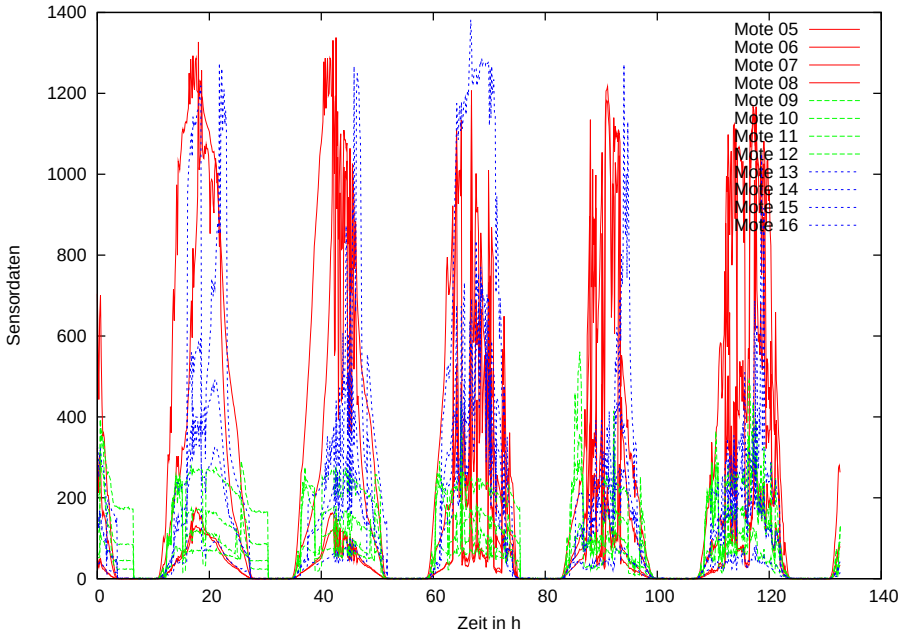


Abbildung 5.6: Sensordaten des Lichtsensors TSR

Besonders gut erkennt man in den Temperaturkurven des roten Büros, dass es bis zum Mittag wärmer wird und es sich danach wieder langsam abkühlt. Die Luftfeuchtigkeit hat einen entsprechenden Verlauf: bis zum Mittag trocknet die Luft, danach wird sie wieder feuchter. Die entsprechenden Kurven des grünen Büros sind deutlich flacher. Ein Grund dafür ist die fehlende Sonneneinstrahlung an der Nordseite, an der sich das Büro befindet. Im blauen Büro ist eine Klimaanlage eingebaut, die jeden Tag um 8 Uhr in Betrieb geht und den Raum kühlt. In den Betriebspausen der Klimaanlage ab etwa 0 Uhr erwärmt sich der Raum wieder. Auch hier folgt die Luftfeuchtigkeit den Temperaturänderungen. Insgesamt steigt die Luftfeuchtigkeit während des Messzeitraums leicht. Beide Lichtsensoren weisen ähnliche Kurven auf. Die Tag- und Nachtphasen sind deutlich erkennbar. In der etwa 8 Stunden dauernden Nacht sind sämtliche Messwerte bei Null. Am Tag fallen zwei rote Motes und ein blauer Mote durch ihre hohen Werte auf. Sie befinden sich an einem Südfenster und werden intensiver von Licht bestrahlt. An den ersten zwei Tagen kann man einen stark verlängerten Arbeitstag im grünen Büro feststellen, denn offensichtlich ist die Bürobeleuchtung eingeschaltet. Die Lichtsensorwerte aller grünen Motes sinken gegen Mitternacht abrupt ab.

Auf Grund dieser Beobachtungen kann man eine Klassifizierung der Sensoren in zwei Klassen unternehmen. Zur ersten Klasse gehören die Temperatur und Luftfeuchtigkeit. Die Kurven „schweben im Graph", das heißt, sie weisen keinen absoluten Nullpunkt auf, sondern bewegen sich periodisch zwischen zwei Extremwerten $\gg 0$. Die Kurven sind höchstens durch eine Konstante in den Graphen 5.3 und 5.4 horizontal zueinander verschoben und variieren leicht in der Steigung.

Die zweite Klasse bilden die Lichtsensoren. Sie sind nach unten durch den Wert Null beschränkt. Die Höhe ihrer Maximalausschläge hängt alleine von ihrem Installationsort und der damit verbundenen Lichteinstrahlung ab. Als generelles Ergebnis dieser Un-

tersuchung kann festgehalten werden, dass die Kurven der Temperatur und der Luftfeuchtigkeit gemeinsame charakteristische Eigenschaften aufweisen. Dasselbe gilt wie erwartet bei den Kurven beider Lichtsensoren. Zwischen diesen beiden Klassen besteht dagegen kein Zusammenhang.

Ereignisse in der Luftfeuchtigkeit und Temperatur Auf der Suche nach geeigneten virtuellen Messreihen für eine spätere Ereigniserkennung zeigen die Abbildungen 5.7, 5.8, 5.9 und 5.10 die Ausgangskurven des Filters „Änderung". Dieser Filter bildet die Differenz zweier aufeinander folgenden Messwerte und gibt sie aus. Als Ereignisse lassen sich große Änderungen besser verwenden als das Erreichen eines Extremwerts. Große Änderungen in den Messwerten sind lokale Erscheinungen. Extremwerte dagegen sind erst in einem globaleren Zusammenhang definiert. So benötigt das Erkennen eines solchen eine Unmenge an Informationen über die Werte in einem Zeitintervall vor und nach dem Extremum.

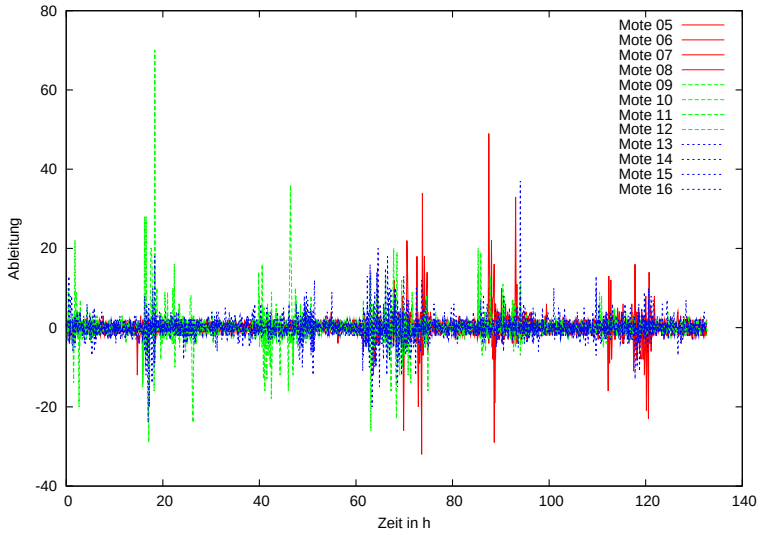


Abbildung 5.7: Änderung der Luftfeuchtigkeit

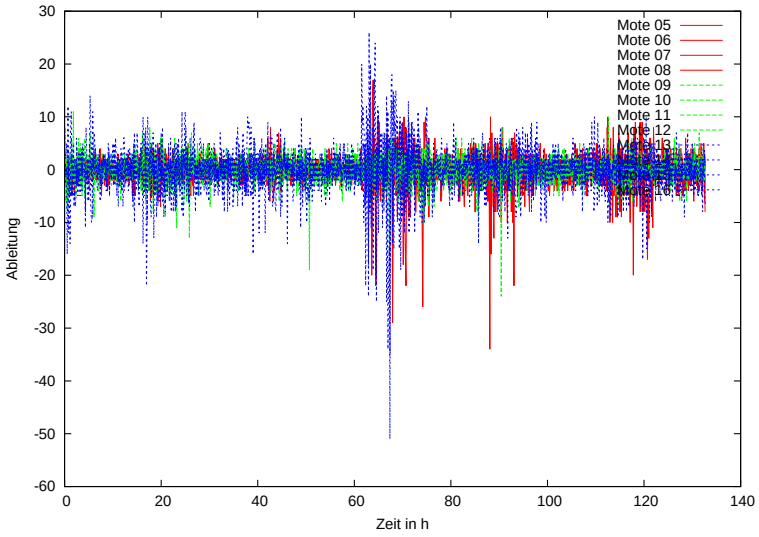


Abbildung 5.8: Änderung der Temperatur

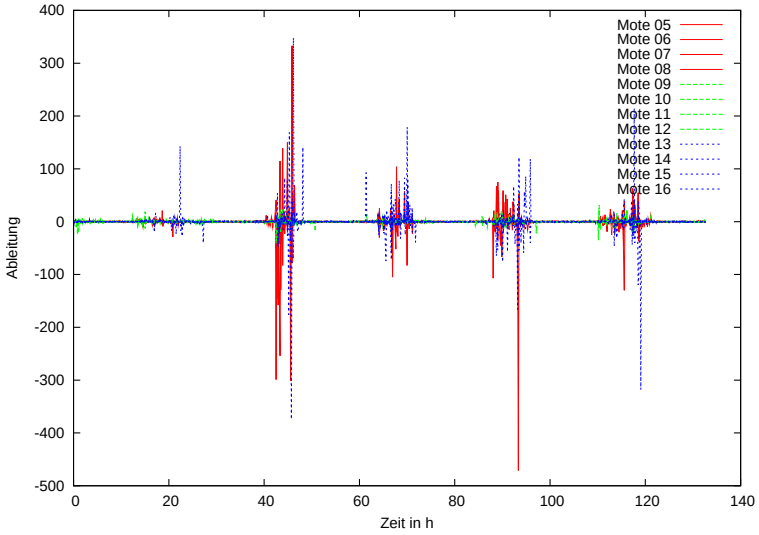


Abbildung 5.9: Änderung des Lichts PAR

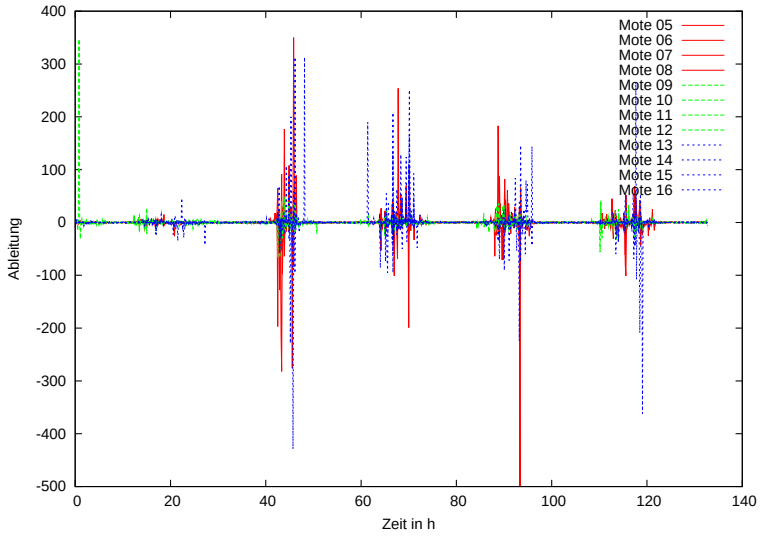


Abbildung 5.10: Änderung des Lichts TSR

Bei der Temperatur- und der Luftfeuchtigkeitskurve in den Abbildungen 5.7 und 5.8 werden zahlreiche Ausschläge durch Rauschen verursacht, wobei sich die Zeitspannen, an denen sich die Temperatur ändert, nicht deutlich herausheben. Trotzdem kann man (als menschlicher Betrachter) erahnen, zu welchem Zeitpunkt es wärmer oder kälter wird. Im Gegensatz dazu sind die Änderungen in der Luftfeuchtigkeit ausgeprägter. Man erkennt auch, dass sich die Änderungen in unterschiedlichen Räumen zu verschiedenen Zeitpunkten abspielen. Dies führt später zu einer leichten Unterscheidbarkeit der Büroräume. Erneut sehen die Änderungen der beiden Lichtsensorwerte in den Abbildungen 5.9 und 5.10 ähnlich aus. In diesem Fall fällt aber der Unterschied in den verschieden ausgeprägten Ausschlägen zwischen beiden Sensoren deutlicher auf. Die teilweise enormen Ausschläge werden von Motes am Südfenster erzeugt, die auch die größten Messwerte liefern. Im Gegensatz zur Temperatur oder Luftfeuchtigkeit, verschwindet hier das Rauschen beinahe ganz. Deshalb sind sie gut in Verbindung mit dem Filter „Ereignis" für die Erkennung von Ereignissen in der Änderung geeignet. Solche Ereignisse sind einschneidende Sprünge im Verlauf der Sensordaten.

Um den unterschiedlichen Problematiken jedes einzelnen Sensors entgegenzutreten, sind jeweils unterschiedliche Ansätze nötig. Ein einfach zu lösendes Problem ist der Effekt des Rauschens. Das Rauschen beruht auf der natürlichen Unstetigkeit der Umgebung, in der sich ein Sensor befindet und auf der Messgenauigkeit des Sensors. Da es sich dabei um kleine statistisch verteilte Schwankungen handelt, hat man mit einer Mittelung der Werte aus der Vergangenheit über ein bestimmtes Zeitintervall Erfolg. Man erhält durch Anwenden des Filter „Mittelwert" so die Abbildungen 5.11 und 5.12 für die Luftfeuchtigkeit bzw. Temperatur.

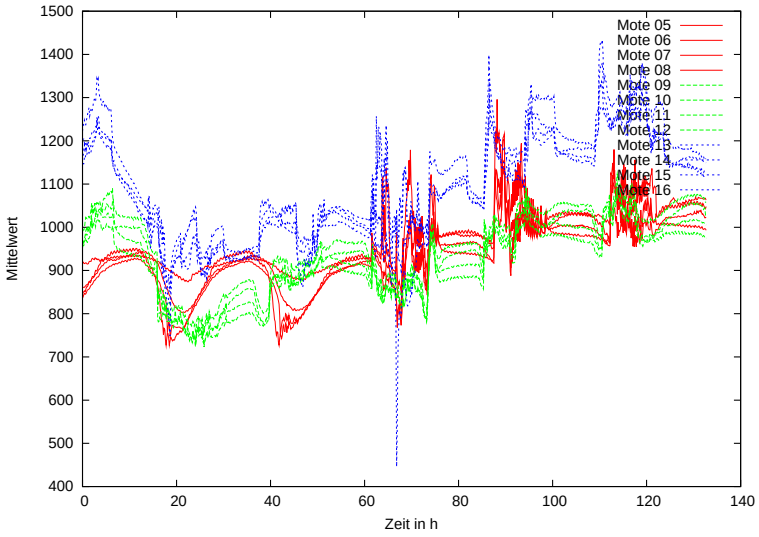


Abbildung 5.11: Mittelwert der Luftfeuchtigkeitssensordaten

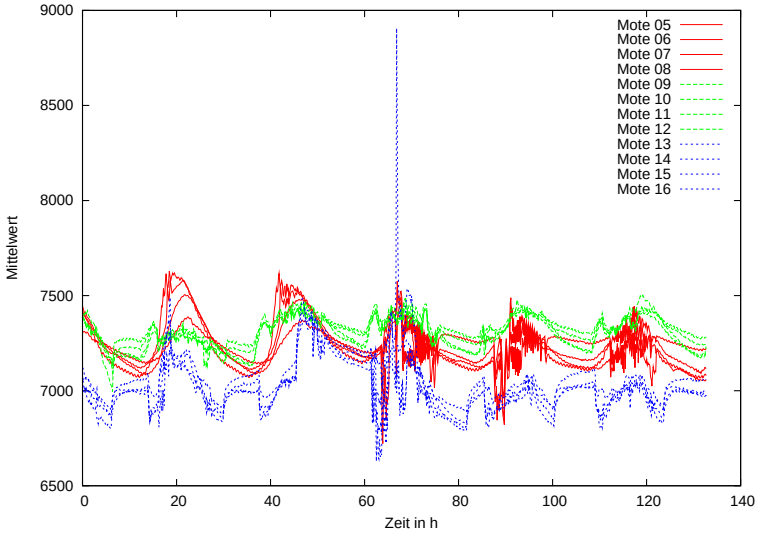


Abbildung 5.12: Mittelwert der Temperatursensordaten

Da das Rauschen hauptsächlich bei der Luftfeuchtigkeit und Temperatur eine problematische Rolle spielen, ist hier der glättende Effekt, der durch die Mittelung entsteht, am deutlichsten. In den Abbildungen werden jeweils über 30 Werte gemittelt, was einem Zeitraum von 5 Minuten entspricht. Bis auf das Rauschen werden durch das Glätten keine Charakteristika dieser Kurven zerstört. Vielmehr hilft es bei der Analyse der Änderung, wie sie in Abbildung 5.13 und 5.14 abgetragen ist. Dort ergeben sich in Werte, die meistens zwischen -1 und 1 liegen. Da die Sensordaten diskret sind, erzeugt der Filter „Änderung" auch nur ganzwertige Zahlen.

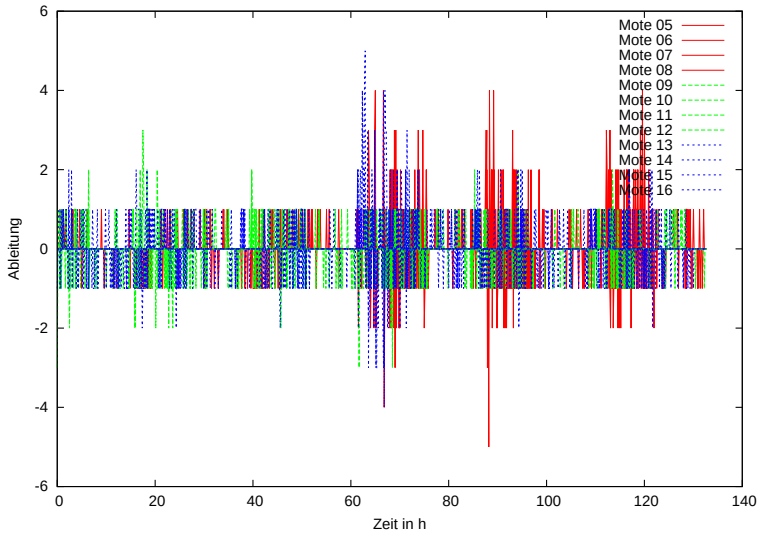


Abbildung 5.13: Änderungen in der Luftfeuchtigkeit

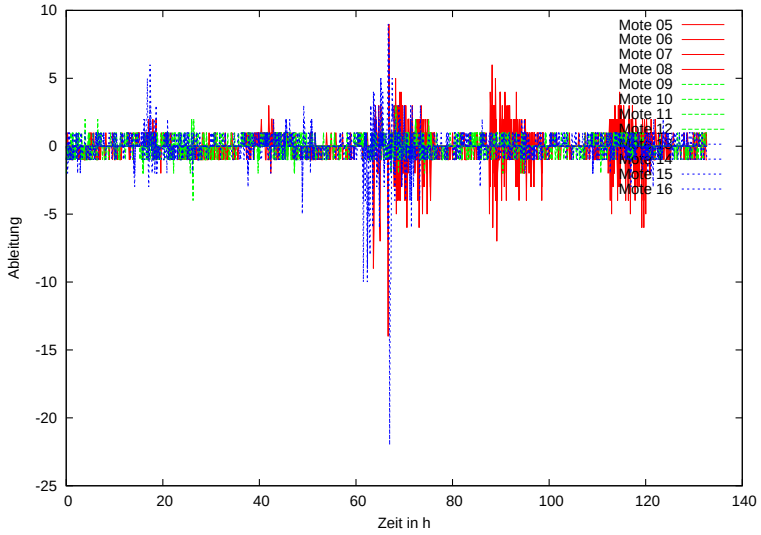


Abbildung 5.14: Änderungen in der Temperatur

Sowohl bei der Luftfeuchtigkeit als auch der Temperatur existieren Bereiche, in denen die Änderung Null beträgt, was ein Indiz dafür ist, dass durch die Wahl des Intervalls das Rauschen beseitigt wurde. Das häufige Auftreten von Änderungen von genau -1 oder 1 zeigt andererseits, dass die Wahl des Intervalls nicht den Verlauf der Kurve ändert. Große Steigungen oder Gefälle in der Kurve werden durch entsprechend große zeitliche Häufungen von 1 bzw. -1 repräsentiert. Mit Hilfe des Filters „Integral vorzeichengleicher Werte" lassen sich diese zeitlichen Häufungen in Zahlen formulieren, was in Abbildung 5.15 und 5.16 dargestellt ist.

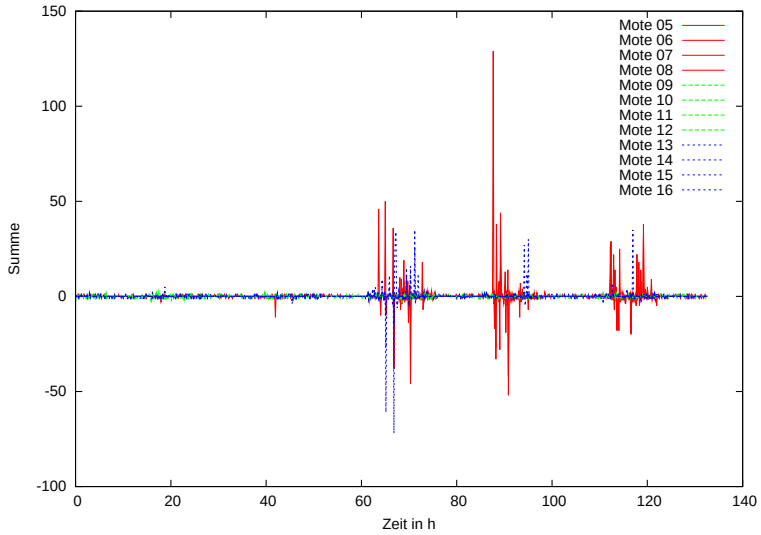


Abbildung 5.15: Anhäufung von Änderungen in der Luftfeuchtigkeit

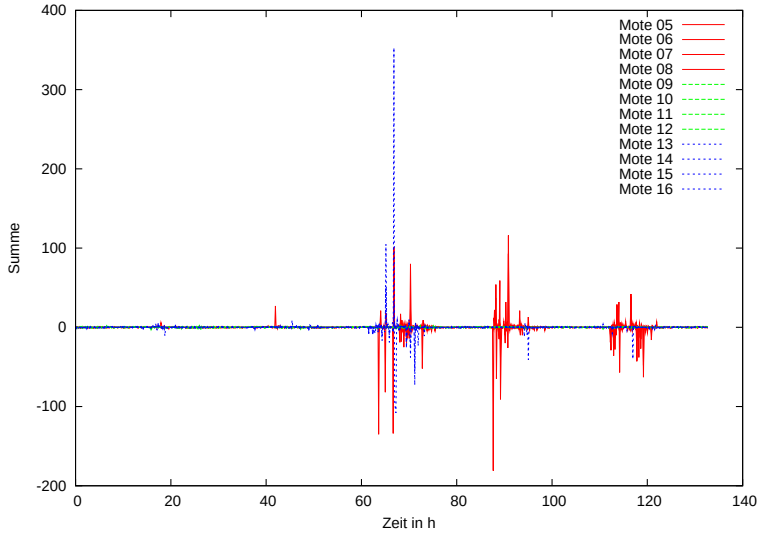
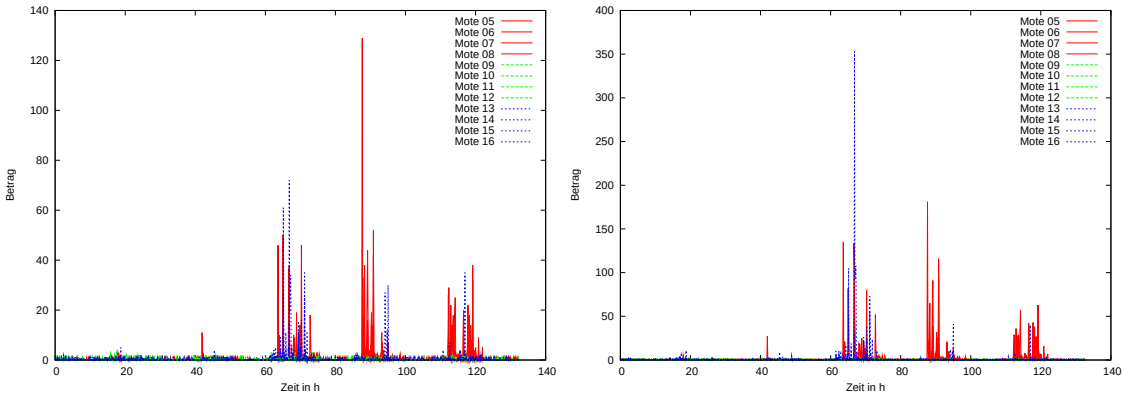


Abbildung 5.16: Anhäufung von Änderungen in der Temperatur

Für eine spätere Ereigniserkennung ist es irrelevant, ob die Kurve einen Sprung nach oben oder einen Sprung nach unten vollzieht. Deshalb wird durch den Filter „Betrag" dieser Unterschied vereinheitlicht. Die Abbildungen 5.17 a) und b) zeigen dies für die betrachteten Daten.



(a) Luftfeuchtigkeit (b) Temperatur

Abbildung 5.17: Betrag der Änderung

Auf Basis der auf solcher Weise gefilterten Sensordaten können schließlich mit dem Filter „Ereignis" die Stellen herausgefunden werden, deren Werte größer als ein angegebener Schwellenwert sind und so ein Ereignis repräsentieren. Abbildungen 5.18 und 5.19 zeigen die gefundenen Ereignisse für die gesammelten Messdaten in der Temperatur und in der Luftfeuchtigkeit.

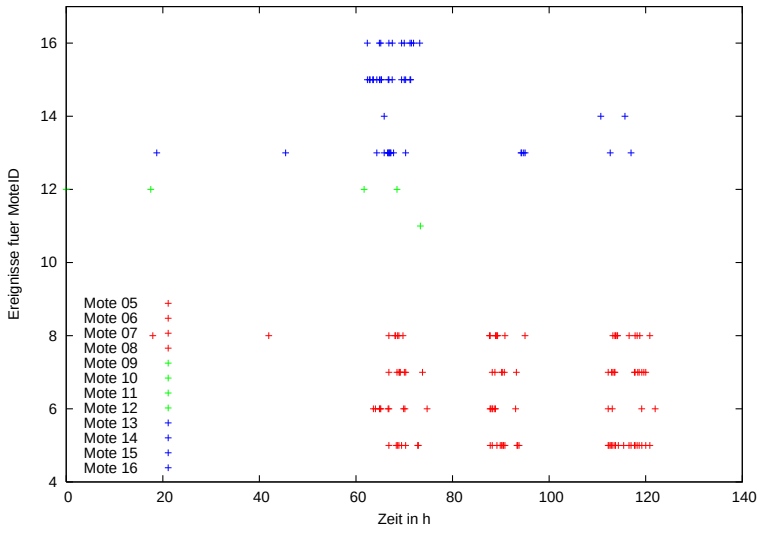


Abbildung 5.18: Ereignisse in der Luftfeuchtigkeit

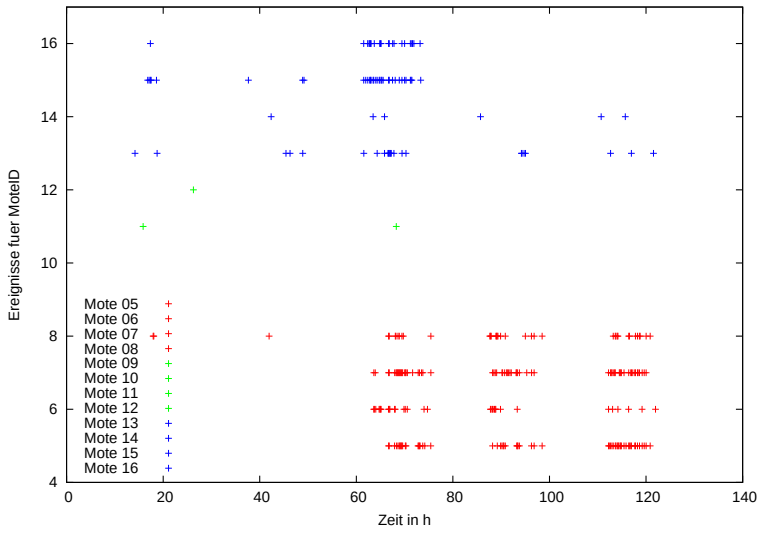
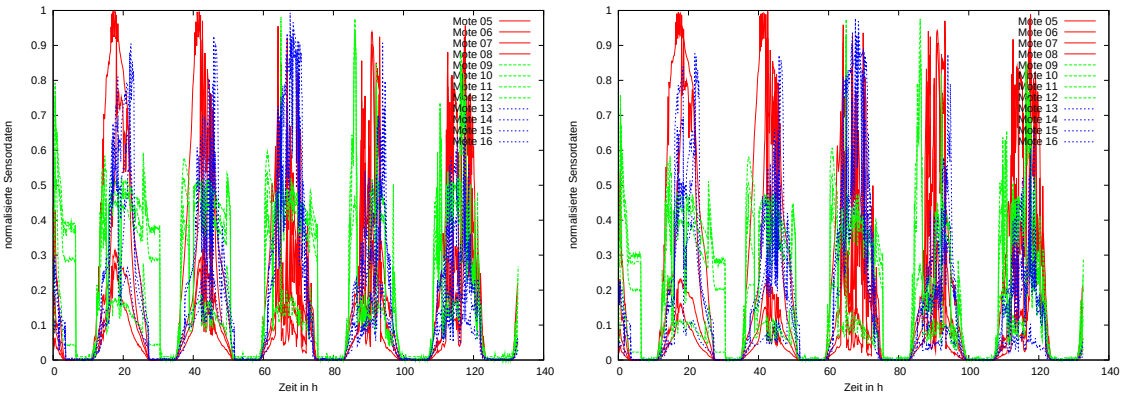


Abbildung 5.19: Ereignisse in der Temperatur

Ereignisse in den Daten des Lichts Für beide Lichtsensoren eignet sich die für Temperatur und Luftfeuchtigkeit vorgestellte Vorgehensweise nicht, da die Größe der Ausschläge in den Änderungen beider Lichtsensorwerte (Abbildungen 5.9 und 5.10) sich in der Amplitude von einem Mote zum andern zu sehr unterscheiden. Sensoren, die intensiver von Licht bestrahlt werden, erzeugen große Sensorwerte und damit auch große Werte in deren Analyse mit dem Filter „Änderung". Sensoren, die im gleichen Raum an einem schattigeren Ort aufgestellt sind, erzeugen eine vergleichbare Kurve, die aber deutlich kleiner in den Sensorwerten ausfällt und somit auch kleinere Änderungen aufweist. Aber selbst ein Versuch, die Kurven aller Motes auf einen gleichen Wertebereich zu stauchen, wie es durch den Filter „Normierung" geschieht und in den Abbildungen 5.20 a) und b) für beide Lichtsensoren dargestellt ist, bringt nicht das erhoffte Resultat.



(a) Lichtbereich PAR

(b) Lichtbereich TSR

Abbildung 5.20: Normierte Lichtsensordaten

Im optimalen Fall wäre eine näherungsweise Überdeckung der Kurven für einen Büroraum zu erkennen. Das Ergebnis ist jedoch von diesem Fall zu weit entfernt, um eine Verbesserung darzustellen. Als Gütegrad kann man das Verhältnis zwischen Maximum und Minimum der Aufsummierung der Änderung am Ende der Messung definieren. Abbildung 5.21 zeigt diese Aufsummierung der Änderung der Lichtdaten TSR aus Abbildung 5.10 stellvertretend. Dagegen ist die Änderung der normierten Sensordaten in Abbildung 5.22 gestellt. Für die unnormierte Grafik gilt ein Verhältnis von 34,94, während es für die normierte Grafik 4,60 beträgt. Dies stellt immerhin eine 7,60-fache Verbesserung dar. Aber bei weiterführenden Analysen stellte sich dies als ungenügend heraus.

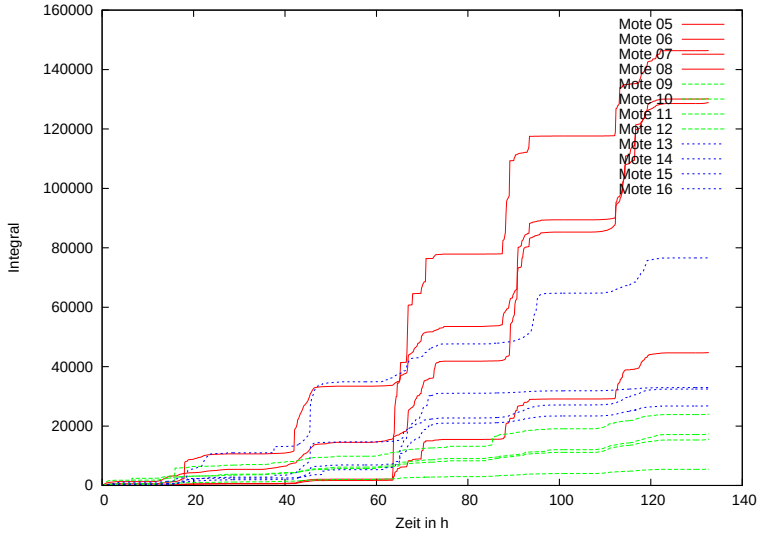


Abbildung 5.21: Aufsummierung der Änderung der unnormierten Lichtdaten TSR

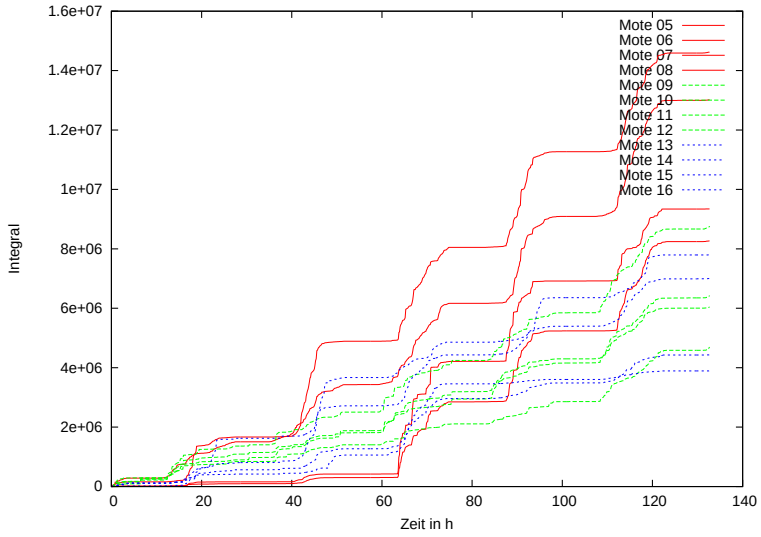


Abbildung 5.22: Aufsummierung der Änderung der normierten Lichtdaten TSR

Ein anderer Ansatz, um eine bessere Vergleichbarkeit der Kurven herzustellen, liegt im Filter „prozentuale Änderung", der die Änderungen zweier aufeinander folgenden Sensorwerte auf das Maximum der beiden projiziert und so die Änderung prozentual ausdrückt. In den Abbildungen 5.23 und 5.24 sind dessen Ergebnisse für beide Lichtsensoren dargestellt.

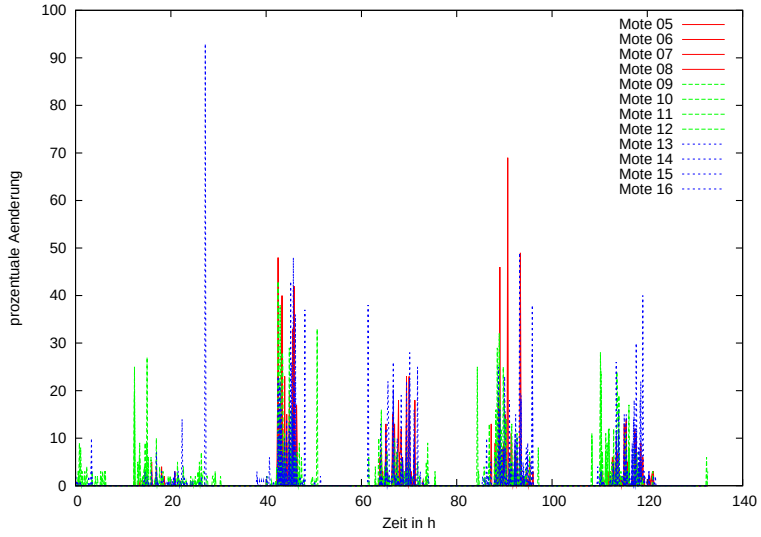


Abbildung 5.23: Prozentuale Änderung des Lichts PAR

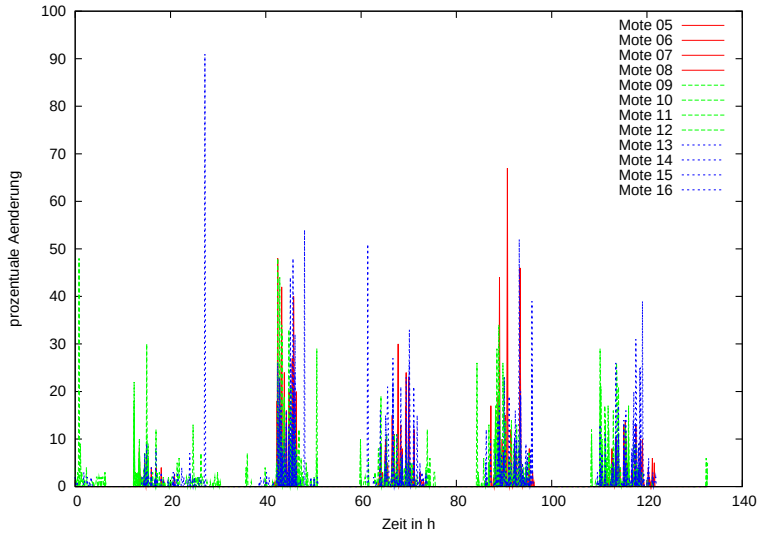


Abbildung 5.24: Prozentuale Änderung des Lichts TSR

Dieser Ansatz versucht nicht die Kurven selbst ähnlicher zu machen, sondern deren Änderungen. Dadurch, dass die Änderung einer Kurve auf das Maximum der zwei Messwerte unmittelbar vor und nach der Änderung relativiert wird, stimmen die Ausschläge in ihrer Höhe besser überein. Die Gütequalität ist in Abbildung 5.25 dargestellt.

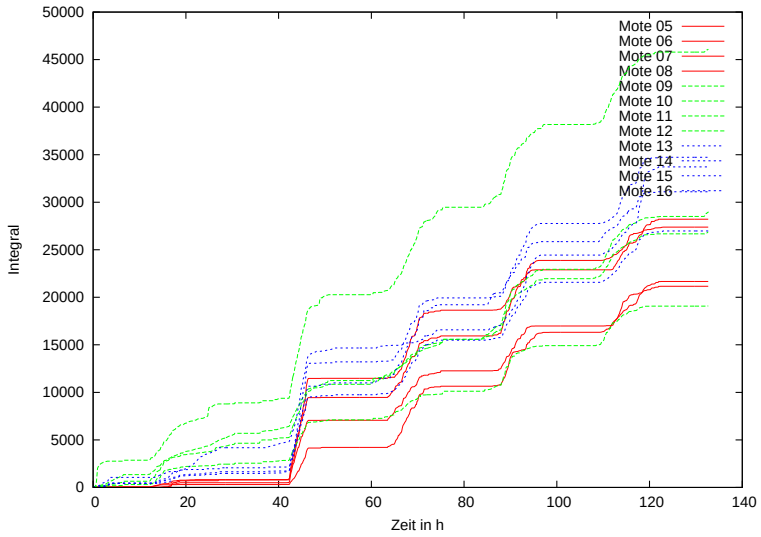


Abbildung 5.25: Aufsummierung der prozentualen Änderung des Lichts TSR

Das Verhältnis zwischen Maximum und Minimum liegt hier bei nur 2,41. Dies bedeutet, dass die prozentuale Änderung eine 14,5-fache Verbesserung in der Ähnlichkeit gegenüber der absoluten Änderung erreicht und nochmal fast einen Faktor von 2 besser als der Ansatz der Normierung. Dies ist ausreichend genug, um Ereignisse daraus Ereignisse ableiten zu können. Abbildungen 5.26 und 5.27 zeigen die Ereignisse, die von einer

Änderung von mehr als 15 % stammen.

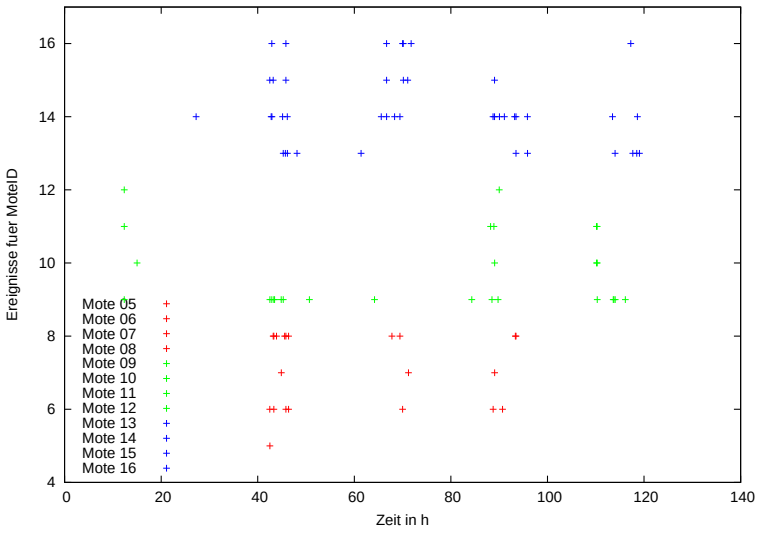


Abbildung 5.26: Ereignisse basierend auf der prozentualen Änderung des Lichts PAR

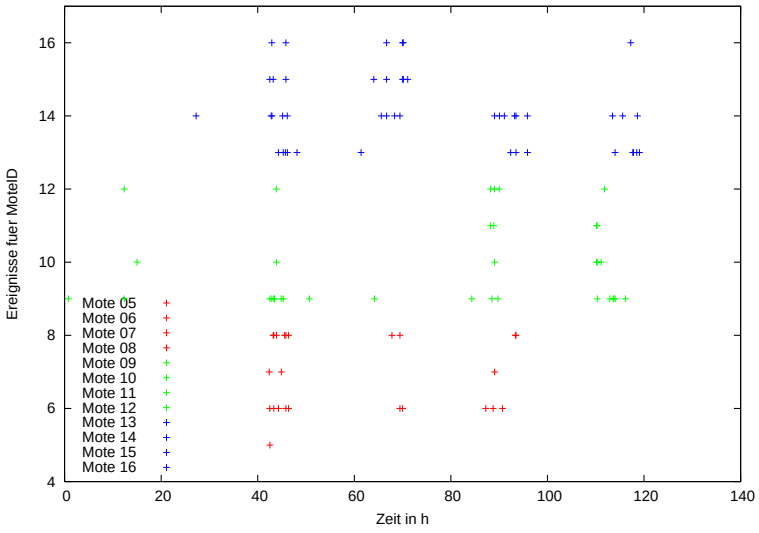
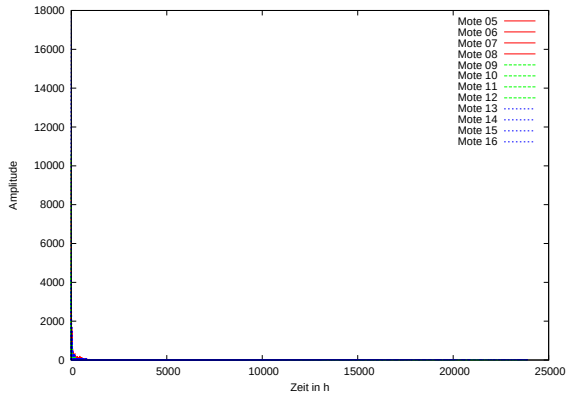
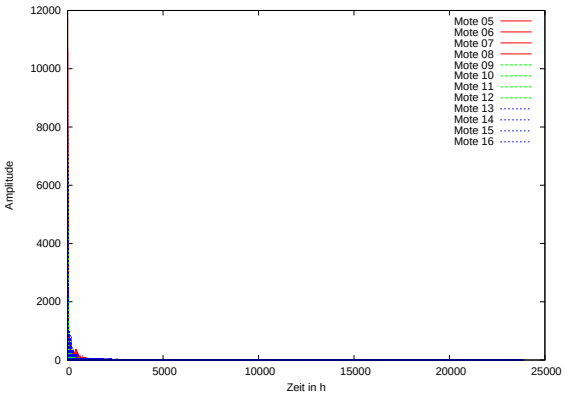


Abbildung 5.27: Ereignisse basierend auf der prozentualen Änderung des Lichts TSR

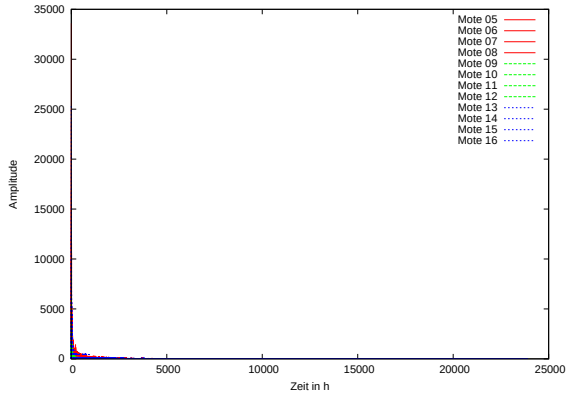
Ungeeignete Analyseverfahren Weitere Analyseverfahren für die Signalverarbeitung schaffen keine Basis für einen Vergleich, der Verwandtschaftsbeziehungen zwischen Motes aus deren Resultaten ableiten zu kann. In den Abbildungen 5.28 a), b), c) und d) sind die Fouriertransformationen der einzelnen Sensordaten dargestellt.



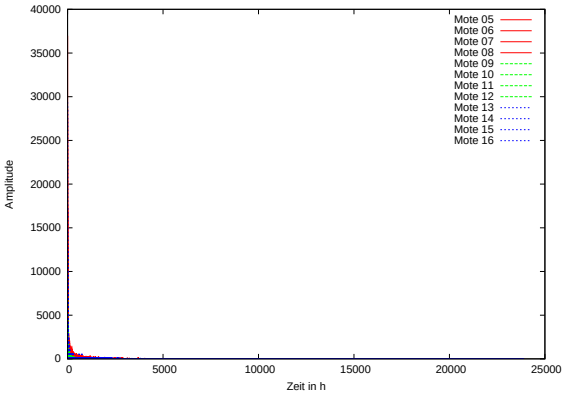
(a) Luftfeuchtigkeit



(b) Temperatur



(c) Lichtbereich PAR



(d) Lichtbereich TSR

Abbildung 5.28: Fouriertransformation

Bei allen Kurven ist deutlich zu erkennen, dass der Hauptteil der Frequenzen im niederen Bereich liegt. Durch diese Kompaktifizierung der ursprünglichen Sensordaten auf einen so geringen Bereich ist kein signifikanter Unterschied vorhanden, der eine Gruppierung von Motes zulassen würde. Eine Rücktransformation mit nur 25 % der Information ergibt, wie die Abbildungen 5.29, 5.30, 5.31 und 5.32 zeigen, eine vollständige Rekonstruktion der Kurven. Zur Originalkurve bestehen keine Unterschiede. Dies kann von Vorteil sein, wenn Kurven gespeichert oder übermittelt werden, da sich die Informationen dadurch auf mindestens 25 % reduzieren lassen.

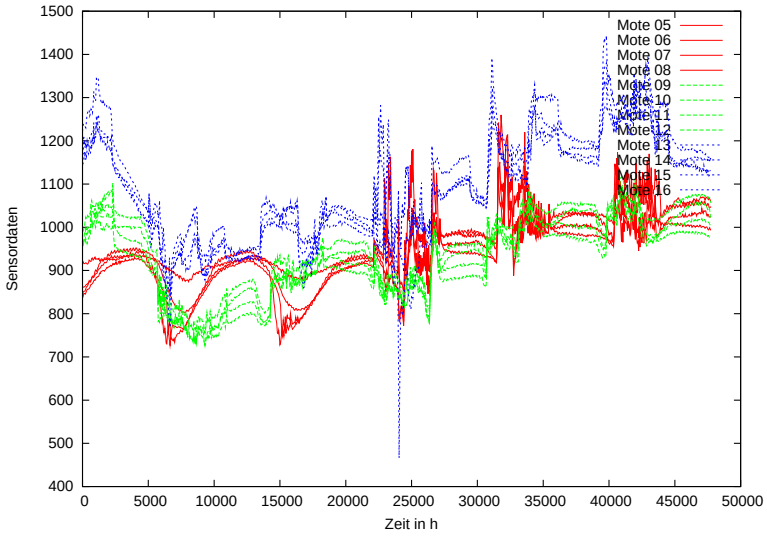


Abbildung 5.29: Rücktransformation der Luftfeuchtigkeit

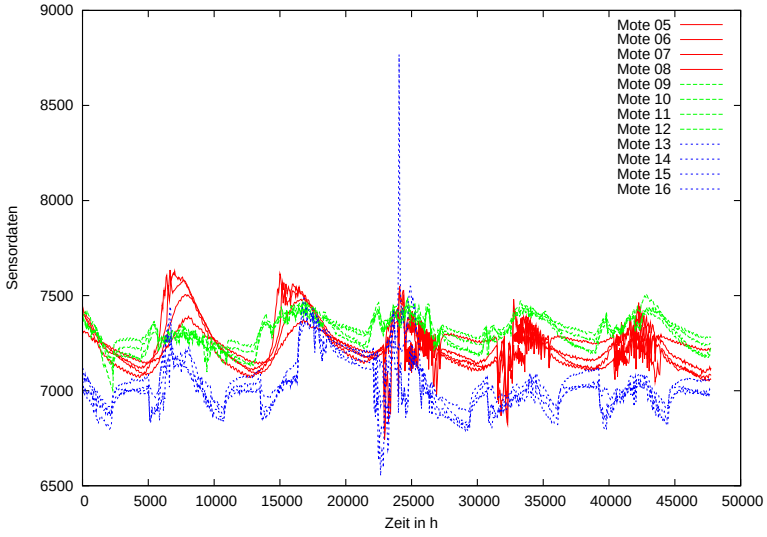


Abbildung 5.30: Rücktransformation der Temperatur

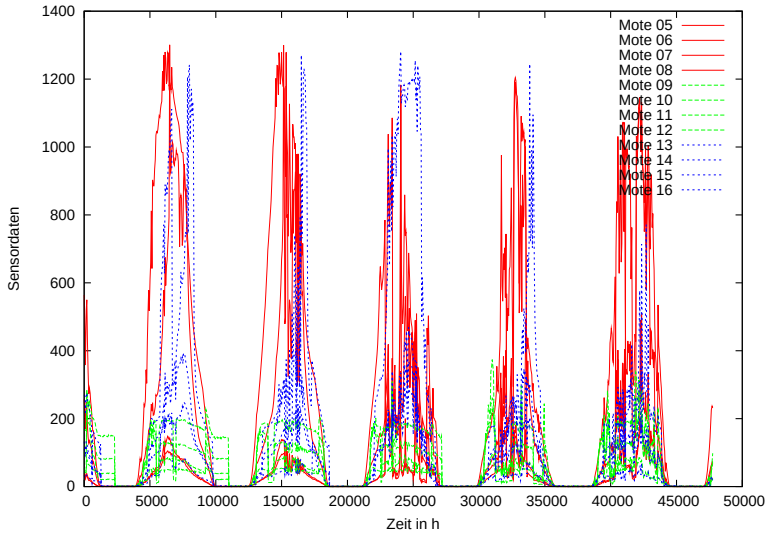


Abbildung 5.31: Rücktransformation des Lichts PAR

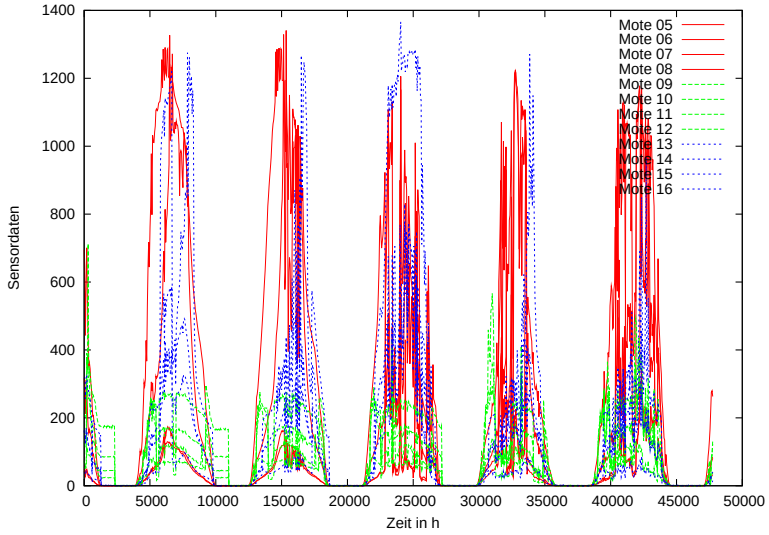
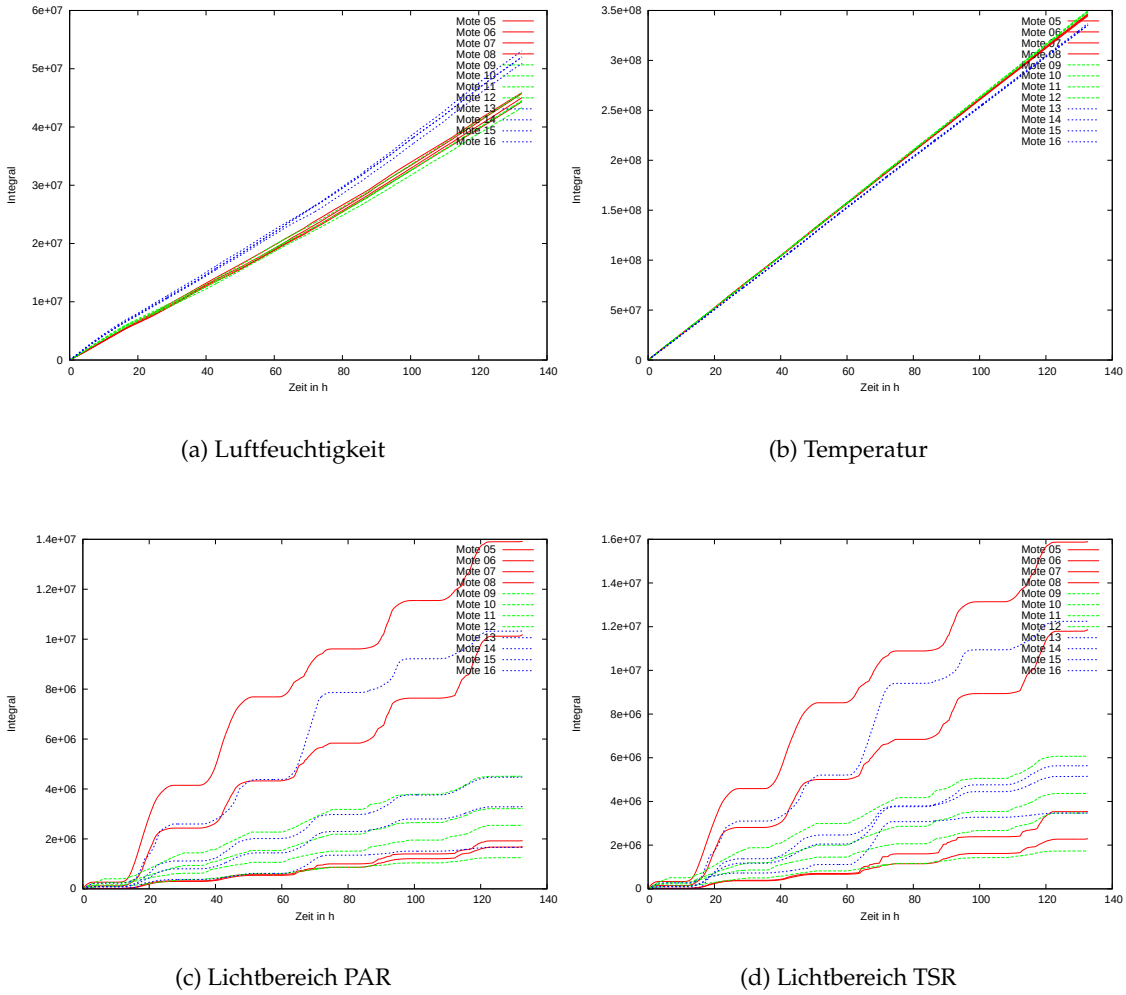


Abbildung 5.32: Rücktransformation des Lichts TSR

Bei den Ergebnissen des Filters „Tendenz", die eine Steigung der Kurve durch eine 1 und ein Abfallen der Kurve durch eine -1 repräsentieren, ist das Rauschen störend. Es muss durch den Filter „Mittelwert" wie schon bei den Änderungen entfernt werden. Danach sind trotzdem keine besonderen Merkmale, die für einen späteren Vergleich hilfreich oder störend sind, erkennbar. Ein Urteil auf Verwendbarkeit wird aus diesem Grund bis in die nächste Stufe vertagt.

Die Anwendung des Filters „Integral" ist in den Abbildungen 5.33 a), b), c) und d) dargestellt. Weder bei der Temperatur, noch bei der Luftfeuchtigkeit, noch bei beiden Lichtbereichen ergeben sich daraus erkennbare Vorteile.



(a) Luftfeuchtigkeit

(b) Temperatur

(c) Lichtbereich PAR

(d) Lichtbereich TSR

Abbildung 5.33: Integral der Sensordaten

5.3.2 Stufe 2: Vergleich

Differenz In der zweiten Stufe werden die in Stufe 1 aufgewerteten Sensordaten mit der Intention einer Verwandtschaftsprognose verglichen. Die einfachsten Daten, die verglichen werden können, sind die Sensordaten direkt. Die Abbildungen 5.34, 5.35, 5.36 und 5.37 bilden das Ergebnis des Vergleichs mit dem Komparator „Differenz" für Mote 15 (ein blauer Mote) repräsentativ für alle 12 Motes ab. Sowohl bei der Luftfeuchtigkeit als auch bei der Temperatur in den Abbildungen 5.34 und 5.35 ist zu erkennen, dass die Differenzen zu den blauen Motes näher an der Nulllinie liegen als zu den anderen Motes. Bei den Differenzen beider Lichtsensordaten in den Abbildung 5.36 und 5.37 ist ein solches Verhalten nicht zu erkennen. Deshalb eignet sich ein solcher Vergleich nur für die Temperatur- und Luftfeuchtigkeitssensordaten.

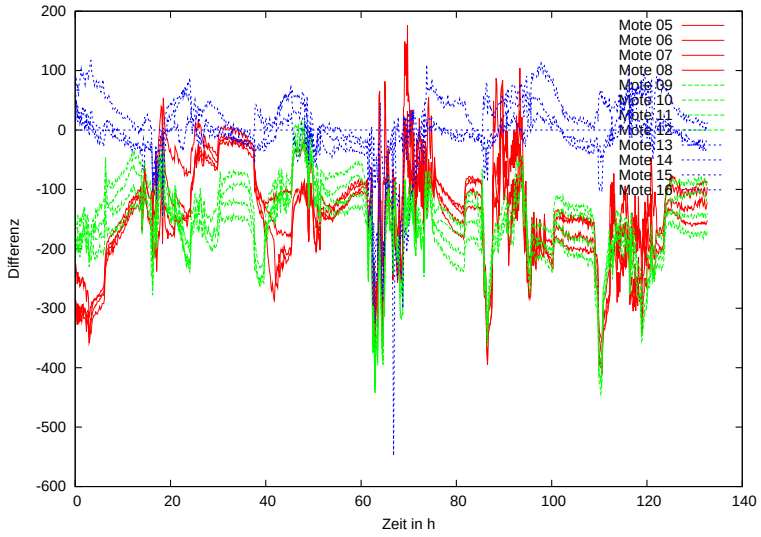


Abbildung 5.34: Differenz der Luftfeuchtigkeitssensordaten von Mote 15

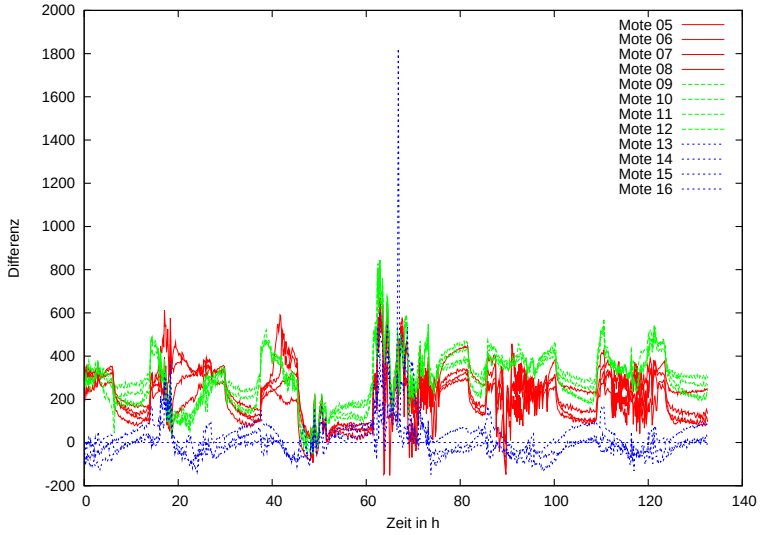


Abbildung 5.35: Differenz der Temperatursensordaten von Mote 15

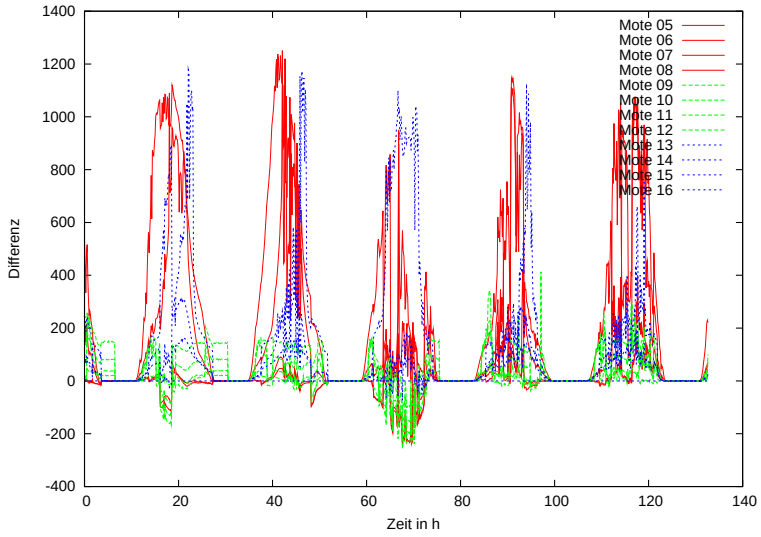


Abbildung 5.36: Differenz der Lichtsensordaten PAR von Mote 15

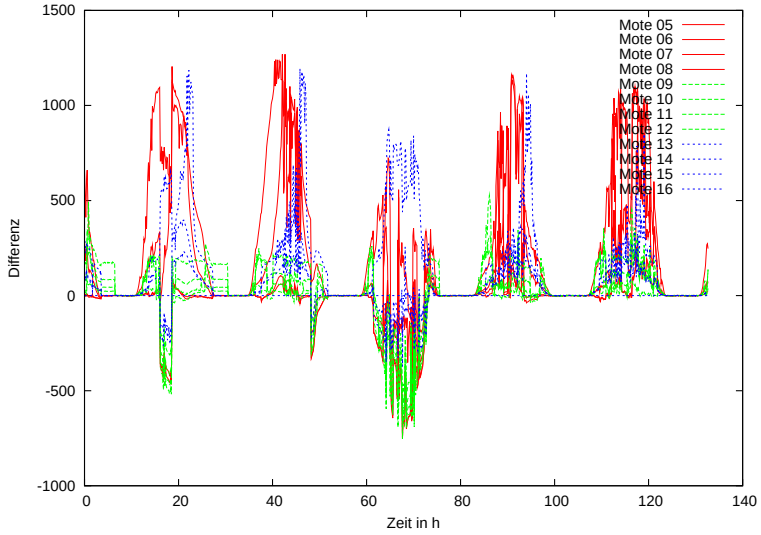


Abbildung 5.37: Differenz der Lichtsensordaten TSR von Mote 15

Die Abbildungen 5.38 und 5.39 zeigen die absoluten Werte der Differenzen in Temperatur und Luftfeuchtigkeit und die Abbildungen 5.40 und 5.41 deren quadrierte Werte. Diese entstehen durch die Anwendung des Modifikators „Betrag" bzw. „Quadrat" auf die vorherigen Daten. Hier sieht man die häufig geringere Differenz der blauen Motes zu Motes 15 - besonders gut bei den quadrierten Werten. Wie man unschwer erkennt, erreichen die quadrierten Werte sehr leicht enorme Ausmaße. Für die Implementierung auf den Motes ist dies mit Problemen verbunden, da der große Wertebereich eine entsprechend große Datenbreite für Variablen erfordert, also speicheraufwändiger ist als die Implementierung mit Betrag. Aus diesem Grund wird weitgehend mit der Betrag-Variante

gearbeitet. So zeigt die Aufsummierung durch den Modifikator „Integral" in den Abbildungen 5.42 und 5.43, dass diese Entscheidung ausreichend ist, um am Ende der Messung deutliche Verwandtschaftsverhältnisse zwischen den Motes (hier repräsentativ für Mote 15) zu prognostizieren. Dies erkennt man an dem ausgeprägten Unterschied zwischen allen blauen Motes und den restlichen Motes, der sowohl bei der Temperatur als auch bei der Luftfeuchtigkeit gemessen werden kann.

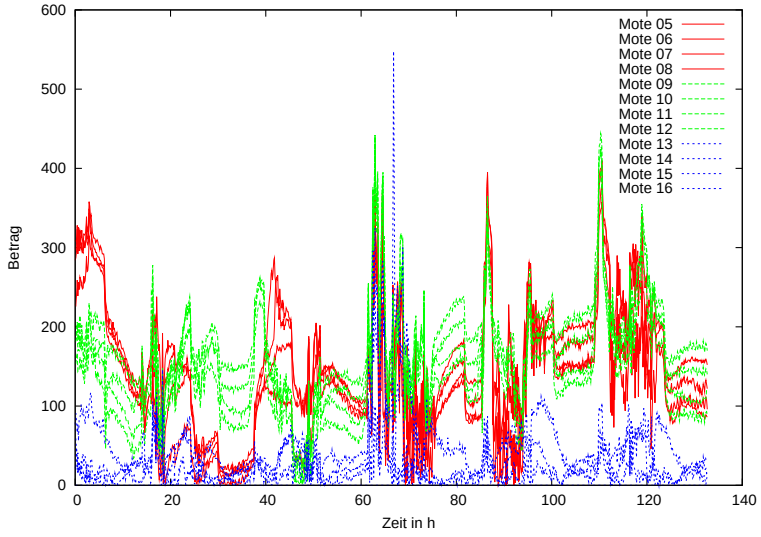


Abbildung 5.38: Betrag der Differenz der Luftfeuchtigkeitssensordaten von Mote 15

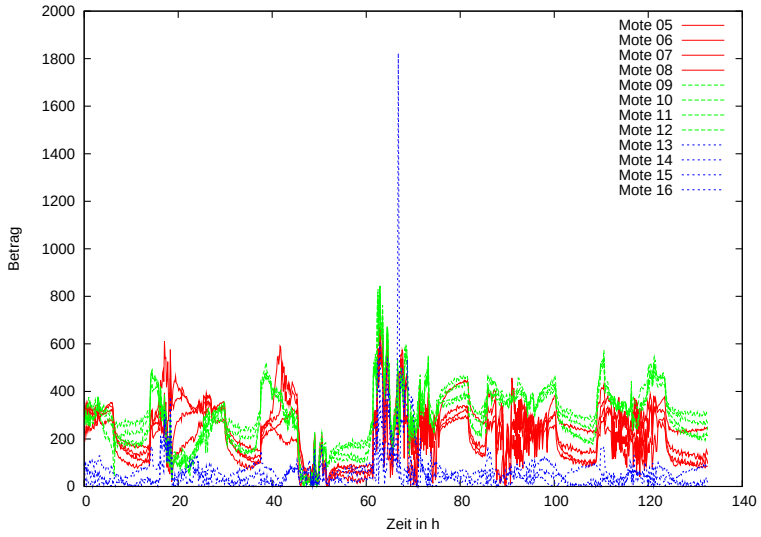


Abbildung 5.39: Betrag der Differenz der Temperatursensordaten von Mote 15

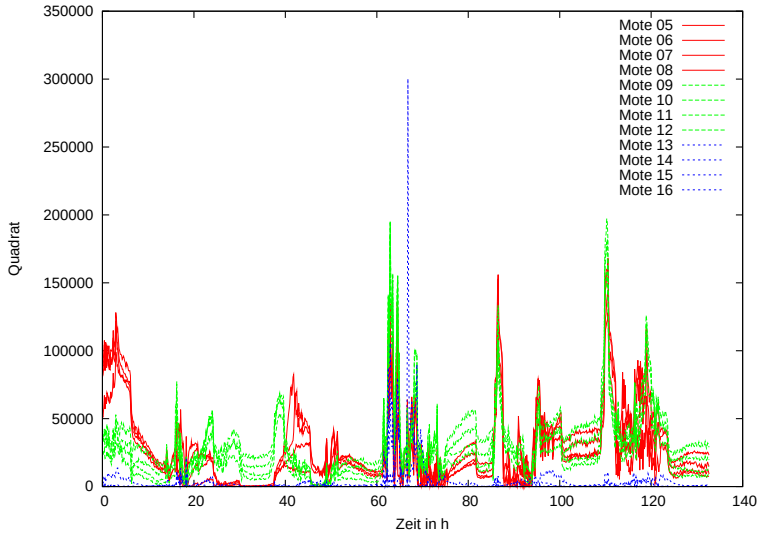


Abbildung 5.40: Quadrat der Differenz der Luftfeuchtigkeitssensordaten von Mote 15

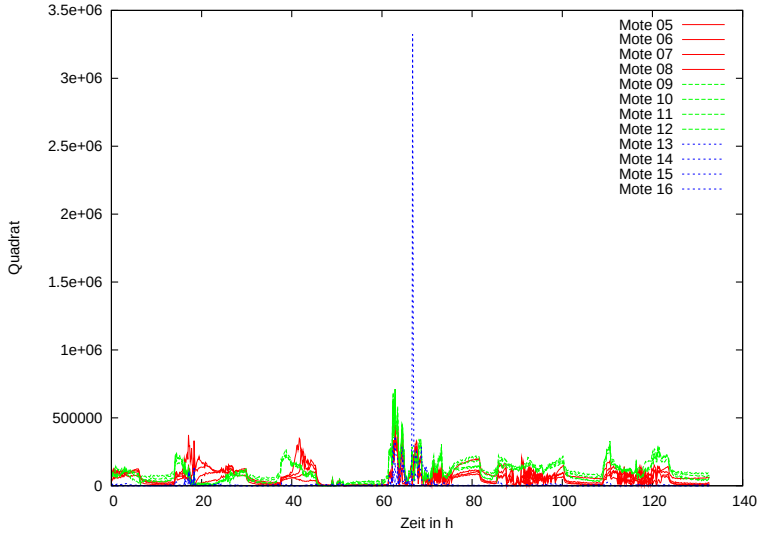


Abbildung 5.41: Quadrat der Differenz der Temperatursensordaten von Mote 15

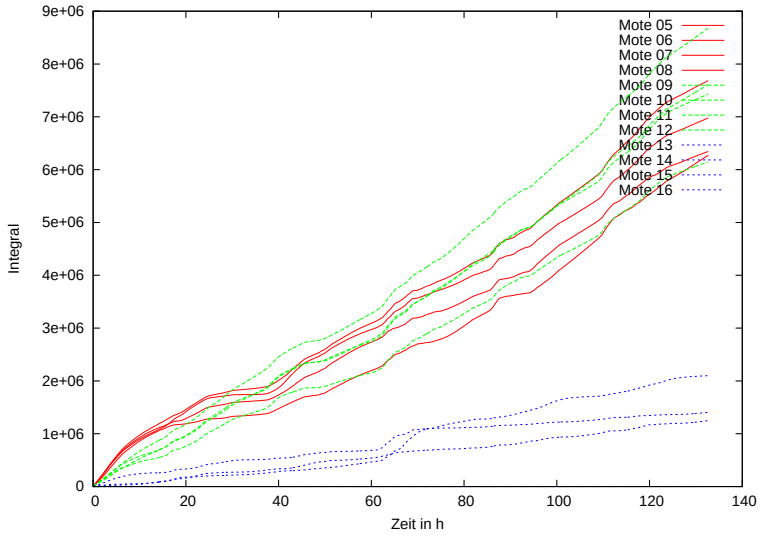


Abbildung 5.42: Aufsummierte absolute Differenz der Luftfeuchtigkeitssensordaten von Mote 15

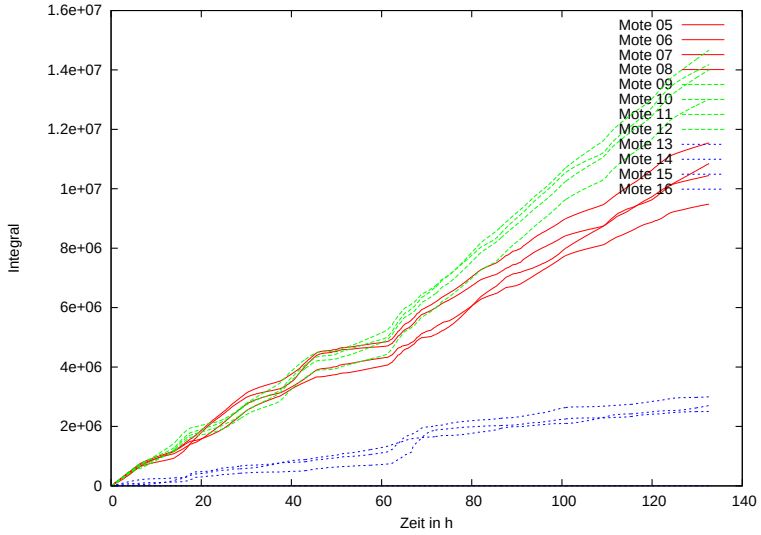


Abbildung 5.43: Aufsummierte absolute Differenz der Temperaturdaten von Mote 15

Prozent Die nächsten Abbildungen 5.44, 5.45, 5.46 und 5.47 zeigen den Komparator „Prozent". Er wird ebenfalls auf die unverarbeiteten Sensordaten angewendet. Die Ergebnisse entsprechen denen der Differenz: Sowohl bei der Temperatur als auch bei der Luftfeuchtigkeit lassen sich die Unterschiede erkennen. Jedoch lassen die Ergebnisse beider Lichtsensoren keinen Rückschluss auf Verwandtschaftsbeziehungen zu. Da die Ergebnisse keine neuen Erkenntnisse bringen und die spätere Verarbeitung aufwändiger als bei der Differenz ist, wird von einem weiteren Einsatz dieses Komparators abgesehen.

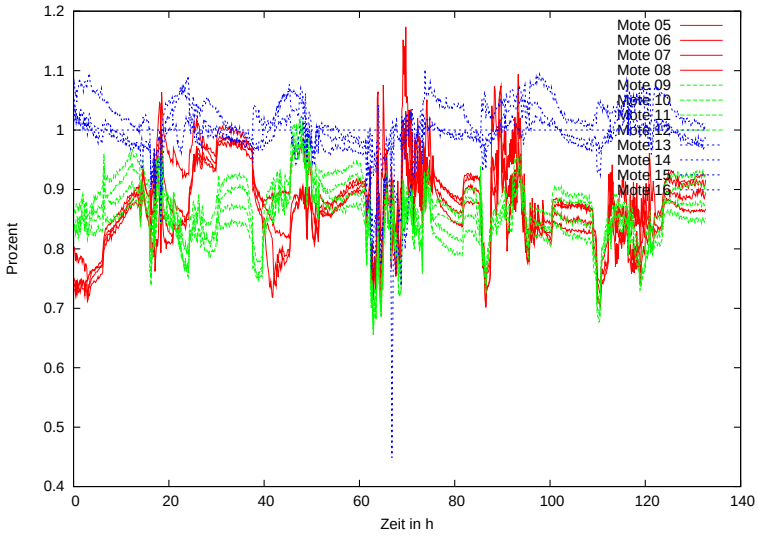


Abbildung 5.44: Prozentuale Unterschiede der Luftfeuchtigkeitsdaten von Mote 15

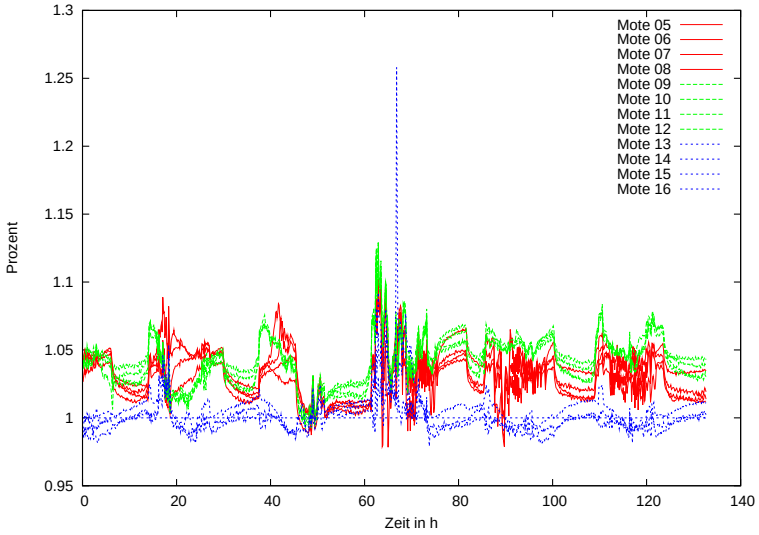


Abbildung 5.45: Prozentuale Unterschiede der Temperaturdaten von Mote 15

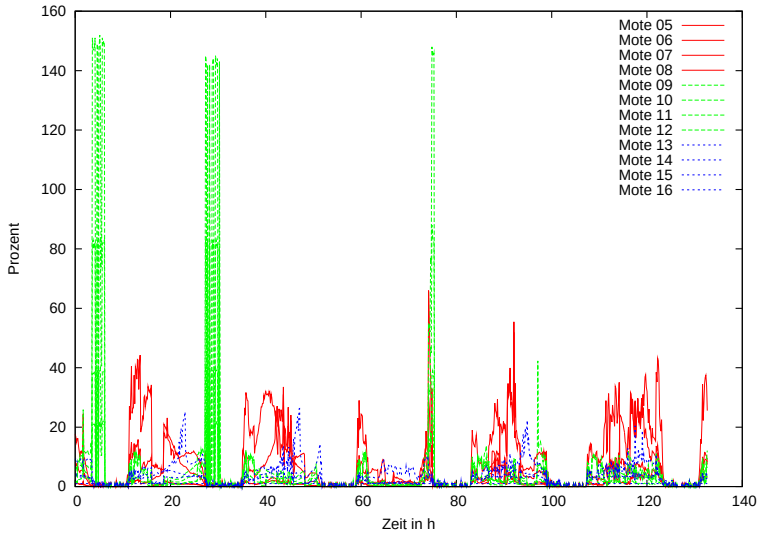


Abbildung 5.46: Prozentuale Unterschiede der Lichtsensordaten PAR von Mote 15

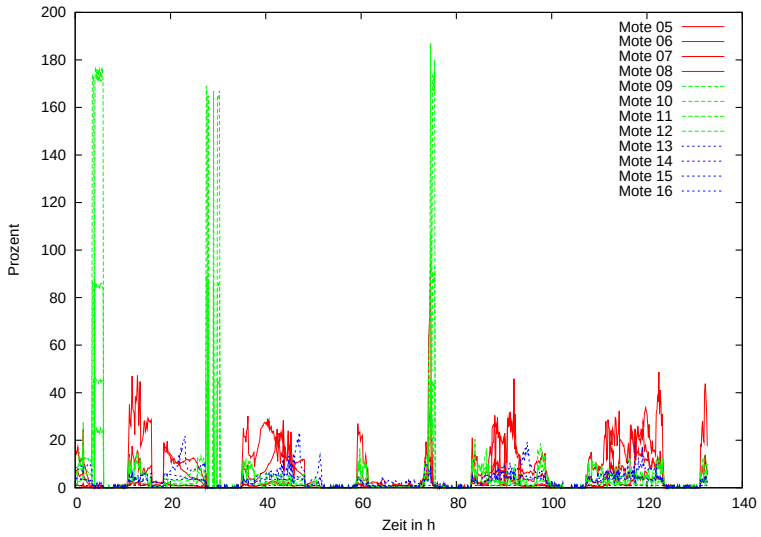
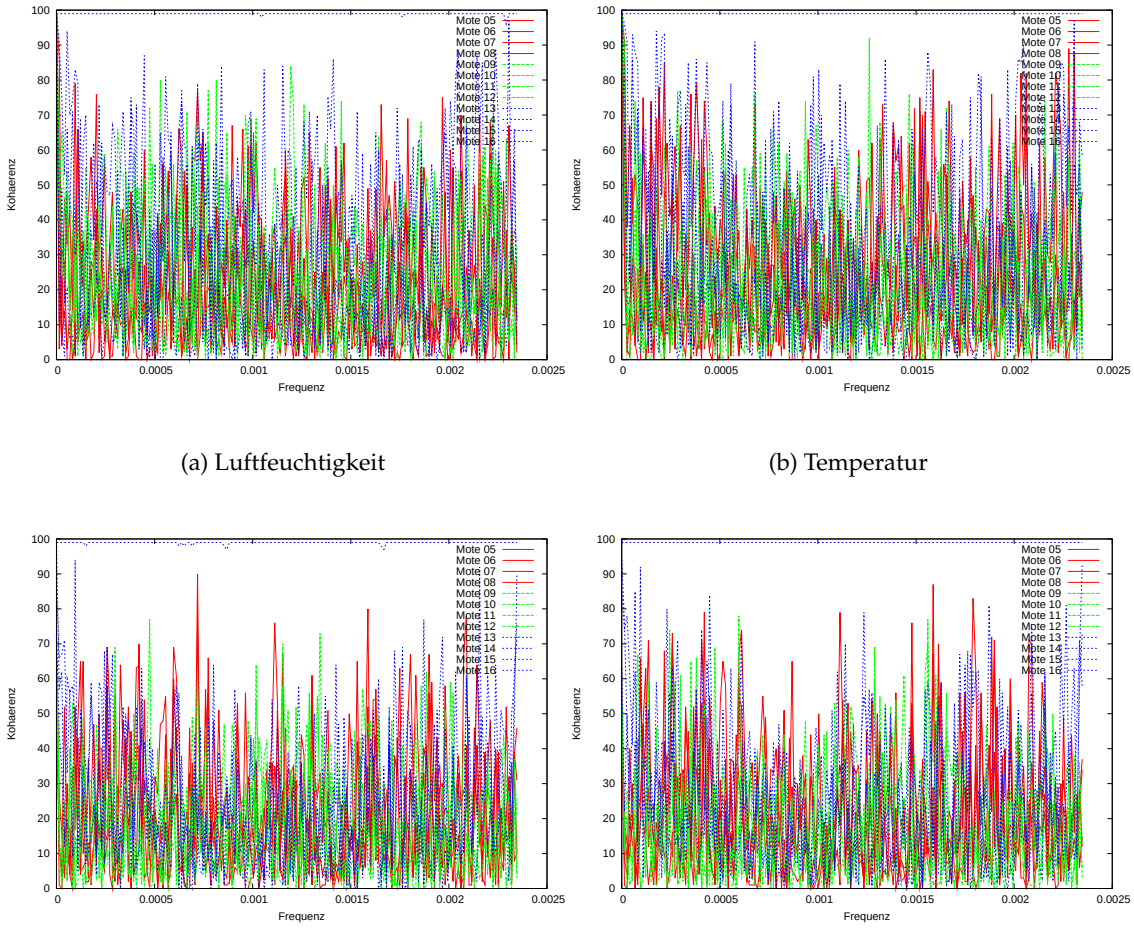


Abbildung 5.47: Prozentuale Unterschiede der Lichtsensordaten TSR von Mote 15

Kohärenz Die Anwendung des Komparators „Kohärenz" auf die direkten Sensordaten ist in den Abbildungen 5.48 a), b), c) und d) gezeigt. Dabei wurden die Eingangskurven in Zeitintervalle der Länge von einem Tag mit einer Überlappung von 5 Minuten eingeteilt. Für keine Sensordatenquelle ergibt sich eine so bestechend gute Lösung, dass sich ein Einsatz dieses komplexen Algorithmus auf Motes rechtfertigen lässt.



(a) Luftfeuchtigkeit

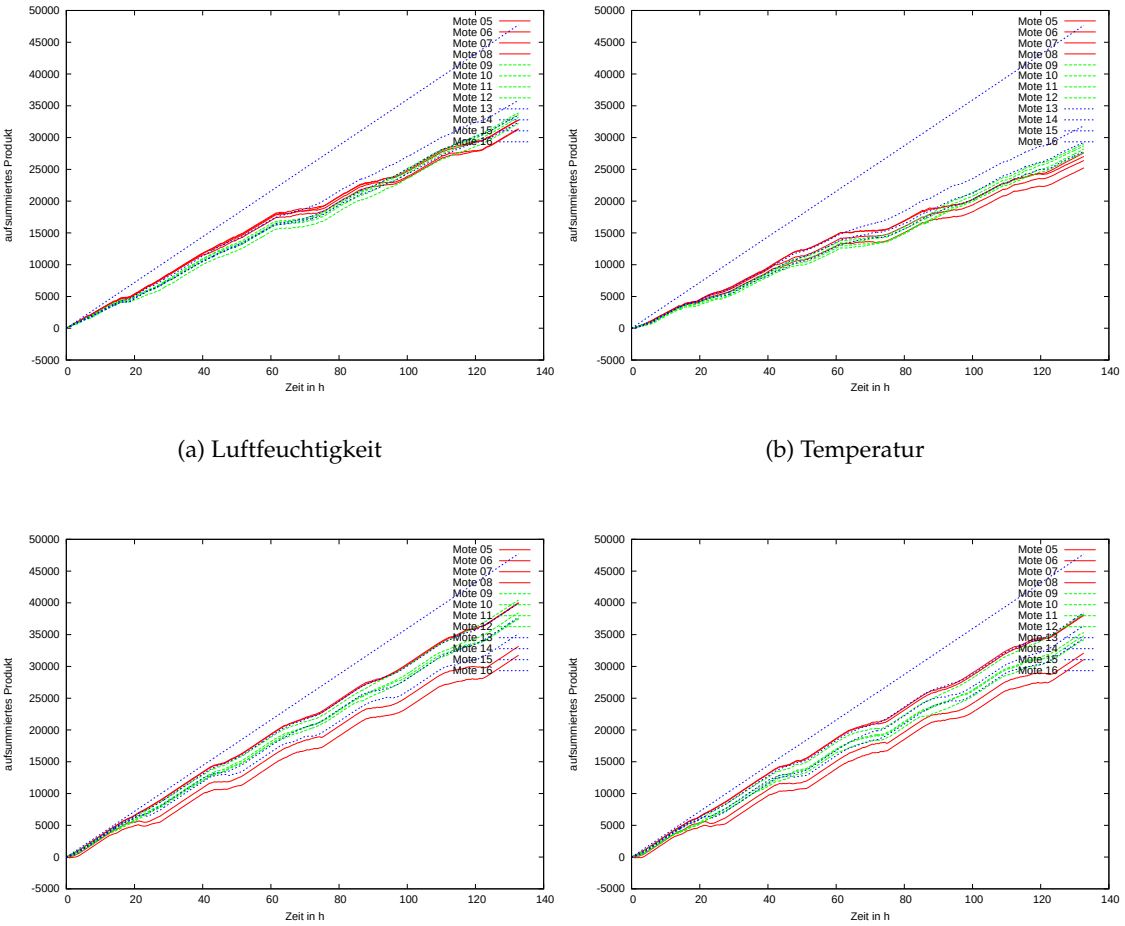
(b) Temperatur

(c) Lichtbereich PAR

(d) Lichtbereich TSR

Abbildung 5.48: Kohärenz der Sensordaten von Mote 15

Tendenz Ein anderer Ansatz, eine Verwandtschaft zwischen Motes zu messen, besteht darin, die Kurven auf die Übereinstimmung ihrer Tendenz, fallend oder steigend, zu untersuchen. Dazu sind mit dem Filter „Tendenz" vorverarbeitete Sensordaten notwendig, die durch Anwendung des Filters „Mittelwert" von Rauschen befreit sind. Die Daten für den Anstieg, repräsentiert durch 1, und das Abfallen, repräsentiert durch -1, werden auf gleiche Tendenz mit dem Komparator „Produkt" verglichen. In den Abbildungen 5.49 a), b), c) und d) wurde der Übersicht halber noch ein Modifikator „Integral" im Anschluss an den Vergleich des blauen Motes 15 angewendet. Wie sich herausstellt, haben alle Kurven einen ähnlichen Verlauf, so dass sich keine Abhebungen für die blauen Motes ergibt. Aus diesem Grund lässt sich dieser Ansatz nicht zur Prognose von Verwandtschaftsbeziehungen heran ziehen.



(a) Luftfeuchtigkeit

(b) Temperatur

(c) Lichtbereich PAR

(d) Lichtbereich TSR

Abbildung 5.49: Aufsummiertes Produkt der Tendenzen der Sensordaten von Mote 15

Ereignisse Die Abbildungen 5.50, 5.51, 5.52 und 5.53 zeigen die übereinstimmenden Ereignisse, wie sie mit dem Komparator „übereinstimmende Ereignisse" als Vergleich erzeugt werden. Als Punkte sind die Ereignisse zu einer gewissen Zeit exemplarisch für den blauen Mote 15 im Vergleich zu jedem anderen Mote dargestellt. Die Datenbasis der Vergleiche stellen die Abbildungen 5.18 und 5.19 für die Luftfeuchtigkeit (5.50) und die Temperatur (5.51) dar. Diese entstehen durch die Filterkette *„Mittelwert"* → *„Änderung"* → *„Integral vorzeichengleicher Werte"* → *„Betrag"* → *„Ereignis"*. Für die Ereignisse der Lichtbereiche bilden die Abbildungen 5.26 und 5.27 die Datenbasis, die von der Filterkette *„prozentuale Änderung"* → *„Ereignis"* erzeugt werden. Bei allen Diagrammen ist eine Übereinstimmung von Ereignissen sowohl in der eigenen Gruppe (blaue Motes) als auch bei Motes aus anderen Gruppen erkennbar. Schon eine wenig größere Anzahl von Übereinstimmungen in der eigenen Gruppe genügt für eine gute Erkennung von Verwandtschaftsbeziehungen.

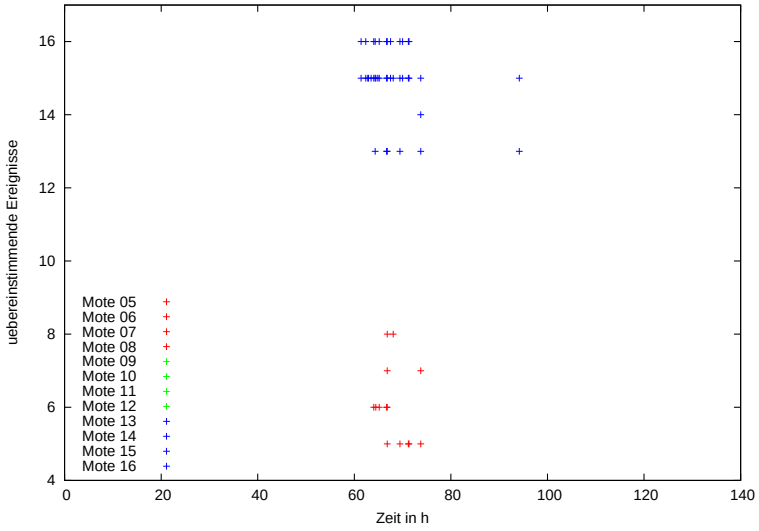


Abbildung 5.50: Übereinstimmende Ereignisse in der Luftfeuchtigkeit von Mote 15

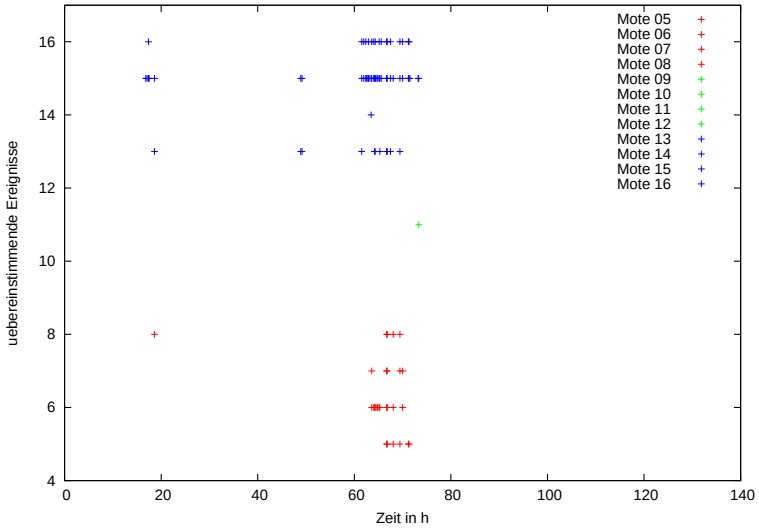


Abbildung 5.51: Übereinstimmende Ereignisse in der Temperatur von Mote 15

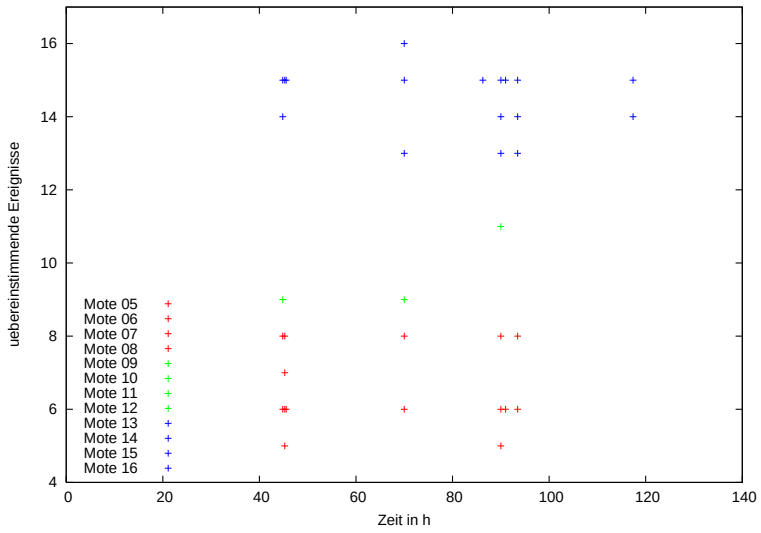


Abbildung 5.52: Übereinstimmende Ereignisse im Lichtbereich PAR von Mote 15

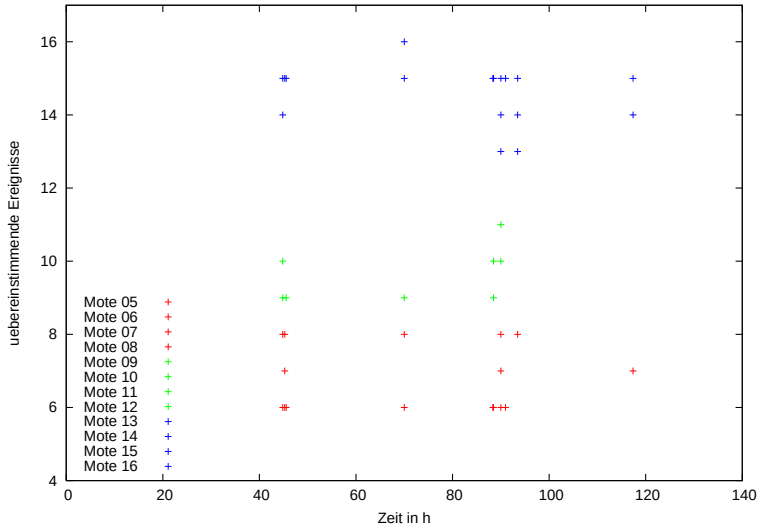


Abbildung 5.53: Übereinstimmende Ereignisse im Lichtbereich TSR von Mote 15

Im Gegensatz zu den „übereinstimmenden Ereignissen" werden in den Abbildungen 5.54, 5.55, 5.56 und 5.57 die Graphen des Komparators „verpassten Ereignisse" dargestellt. Als Datenbasis dienen dieselben Ausgangskurven. Es zeigt sich, dass mehr Ereignisse von anderen Motes verpasst werden als erfasst werden. Auch hier reicht schon eine wenig geringere Anzahl von verpassten Ereignissen für die Zusicherung einer guten Verwandtschaftsbeziehung.

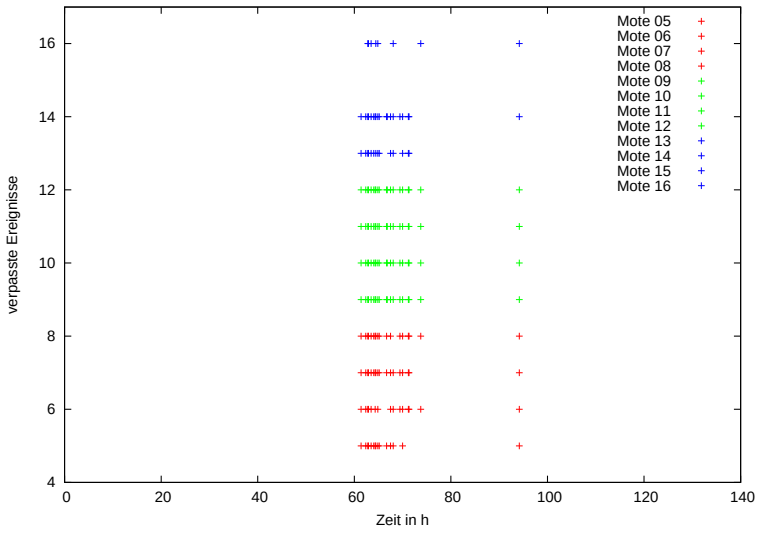


Abbildung 5.54: Verpasste Ereignisse in der Luftfeuchtigkeit von Mote 15

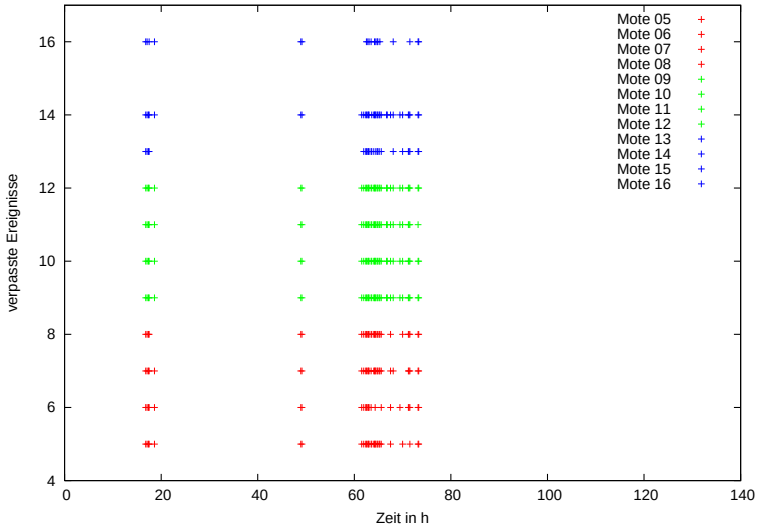


Abbildung 5.55: Verpasste Ereignisse in der Temperatur von Mote 15

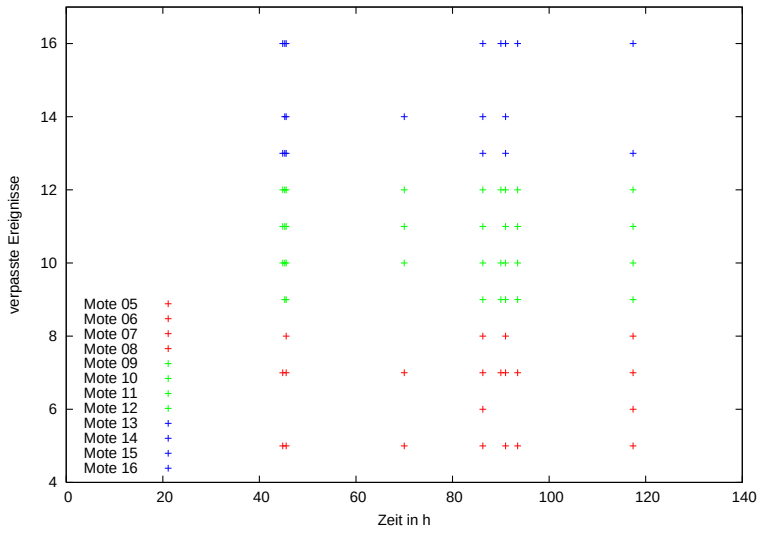


Abbildung 5.56: Verpasste Ereignisse im Lichtbereich PAR von Mote 15

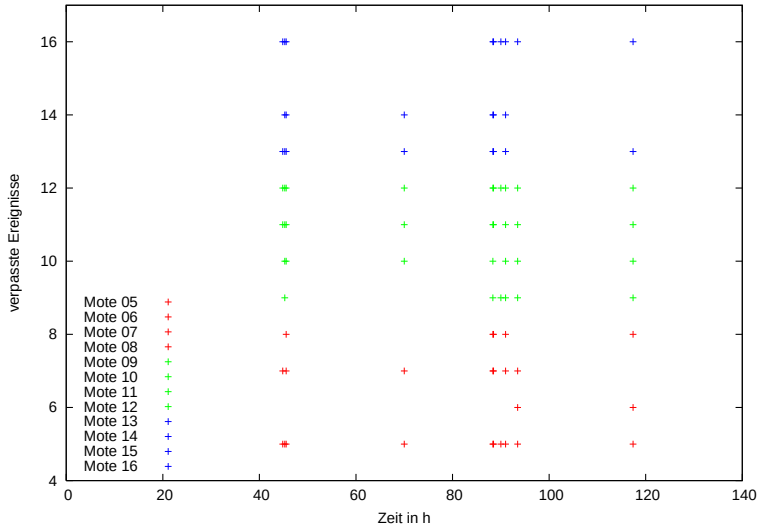
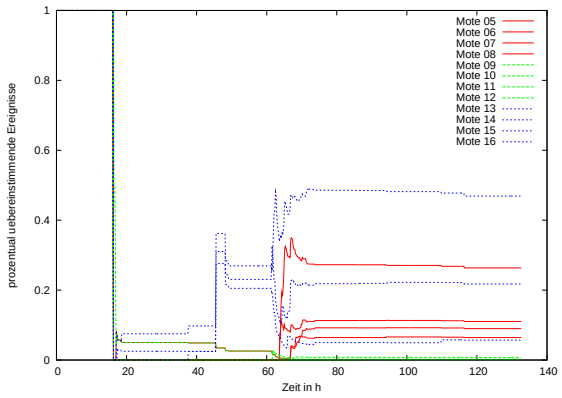
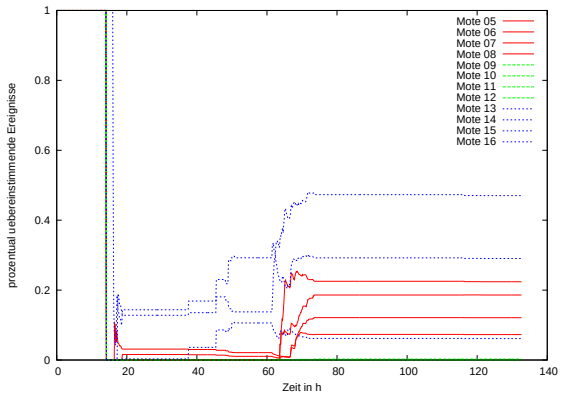


Abbildung 5.57: Verpasste Ereignisse im Lichtbereich TSR von Mote 15

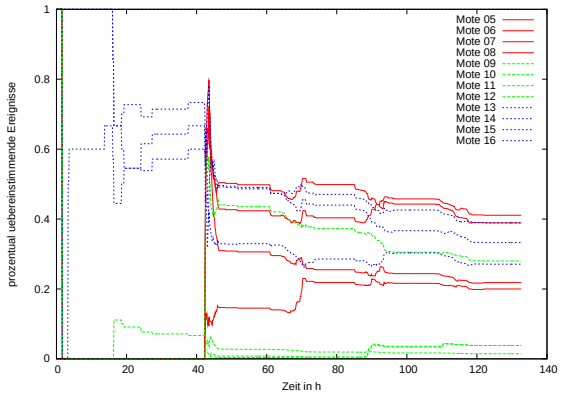
Eine Kombination aus beiden Ereignis-Komparatoren ist der Komparator „prozentual übereinstimmende Ereignisse". Er zählt sowohl die übereinstimmenden als auch alle Ereignisse und bildet daraus das Verhältnis. In den Abbildungen 5.58 a), b), c) und d) ist jeweils sichtbar, wie sich eine wenig größere Anzahl an übereinstimmenden Ereignissen auf den Verwandtschaftsgrad auswirkt.



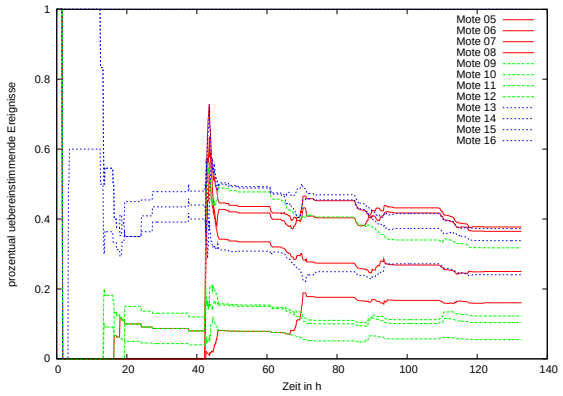
(a) Luftfeuchtigkeit



(b) Temperatur



(c) Lichtbereich PAR



(d) Lichtbereich TSR

Abbildung 5.58: Prozentual übereinstimmende Ereignisse von Mote 15

Dieselben Ergebnisse liefert eine generische Kombination aus dem Modifikator „Integral" mit den Daten der Komparatoren „übereinstimmende Ereignisse" und „verpasste Ereignisse". Bei unterschiedlicher Gewichtung der übereinstimmenden Ereignisse zu verpassten Ereignissen bildet der Modifikator die Summe. Die Abbildungen 5.59, 5.60, 5.61 und 5.62 zeigen eine Gewichtung von 2 für übereinstimmende und -1 für verpasste Ereignisse. Mit einer dem Einsatzszenario angepassten Wahl der Gewichtungsparameter lassen sich so negative Werte erzeugen, die von Motes angenommen werden, die nicht zur Gruppe gehören.

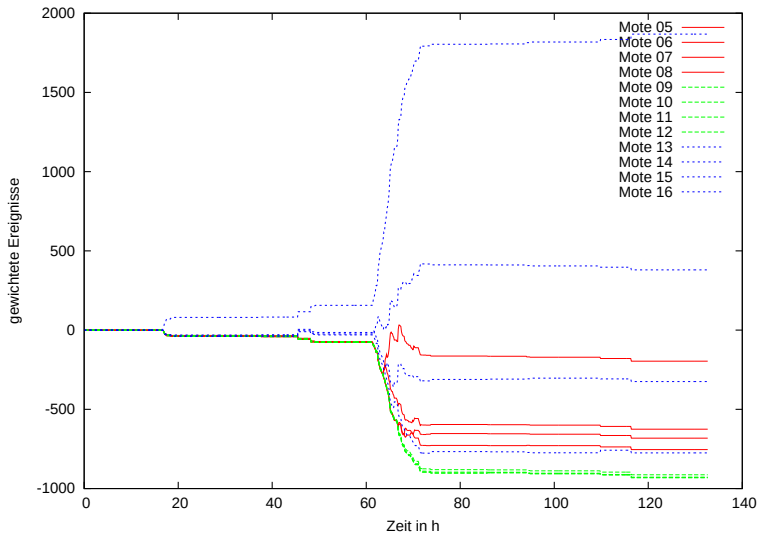


Abbildung 5.59: Aufsummierte gewichtete Ereignisse der Luftfeuchtigkeit von Mote 15

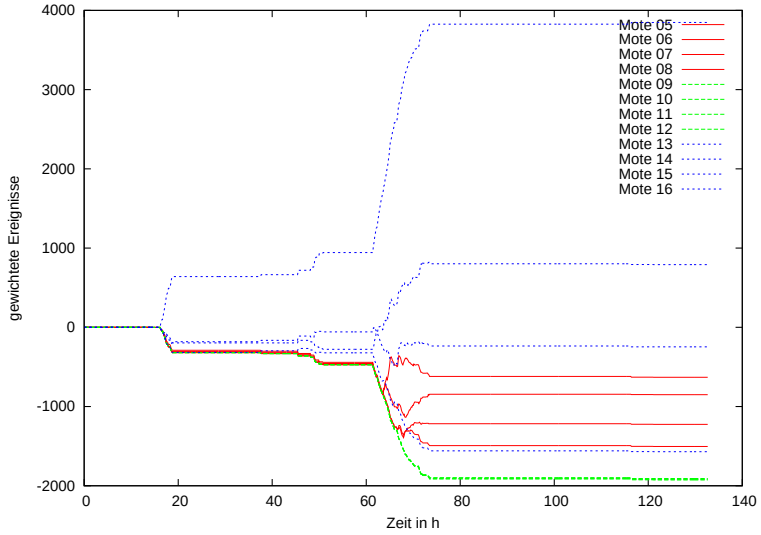


Abbildung 5.60: Aufsummierte gewichtete Ereignisse der Temperatur von Mote 15

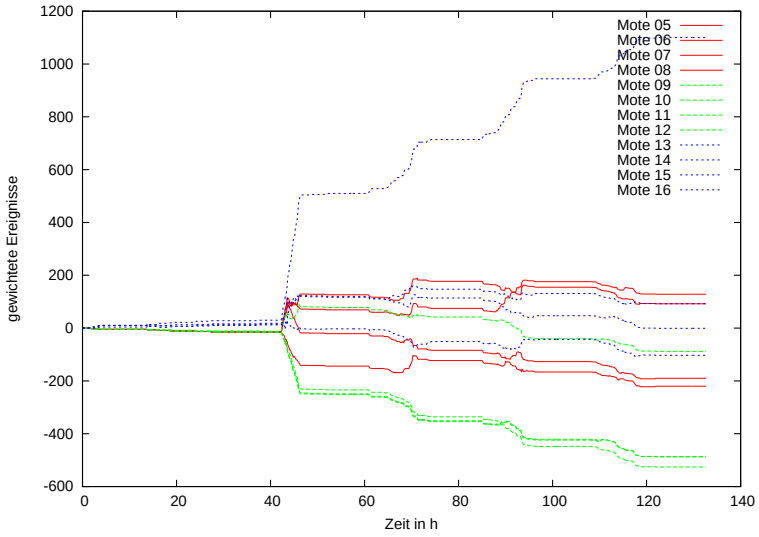


Abbildung 5.61: Aufsummierte gewichtete Ereignisse im Lichtbereich PAR von Mote 15

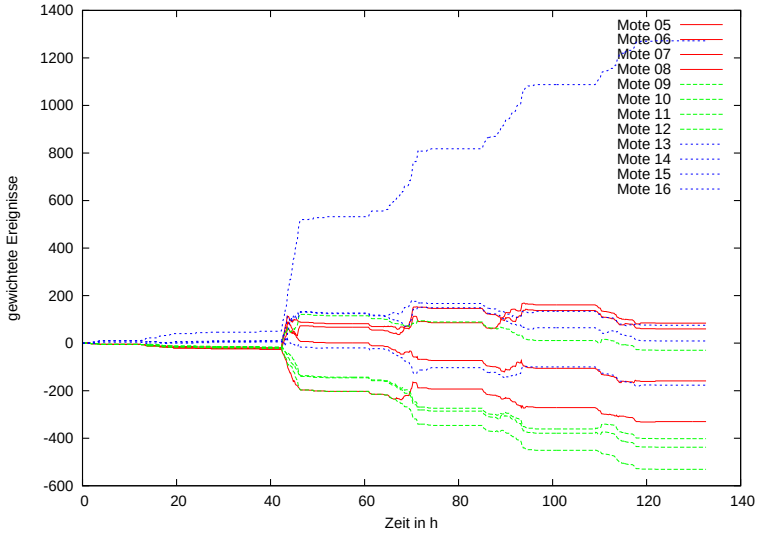


Abbildung 5.62: Aufsummierte gewichtete Ereignisse im Lichtbereich TSR von Mote 15

5.3.3 Signalstärke

Die Signalstärke von Nachrichten, die aus der Nachbarschaft zu einem Mote eintreffen, kann auch als grobe Unterscheidungshilfe von Regionen dienen. In Abbildung 5.63 sind die gemittelten Signalstärken in Abhängigkeit der Sendeleistung für alle Motes aufgetragen, wie sie von Mote 15 gemessen werden. Für eine größere Übersichtlichkeit wurde von allen Signalstärken ein Wert von 180 abgezogen, wodurch die Unterschiede zwischen den Motes für das Auge deutlicher hervortreten. Die Signale von Nachrichten aus der eigenen Region sind in der Regel stärker. Dies äußert sich in Abbildung 5.63 in den

höheren Balken, die sich für die blauen Motes ergeben.

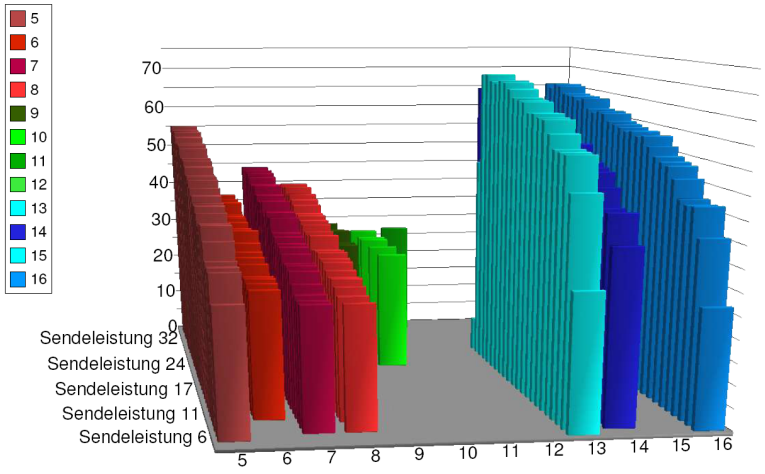


Abbildung 5.63: Statistische Verteilung der Signalstärke von Mote 15 empfangener Nachrichten

Um eine gleiche Signalstärke beim empfangenden Mote hervorzurufen, wie es Motes aus derselben Region mit niederer Sendeleistung erreichen, müssen weiter entfernte Motes aus anderen Regionen ihre Sendeleistung erhöhen. Um diesen Effekt mit einzubeziehen, werden Signalstärken von Nachrichten invers zu ihrer Sendeleistung gewichtet. Abbildung 5.64 zeigt die aufsummierten, gewichteten Signalstärken von dem blauen Mote 15 über die Zeit verteilt.

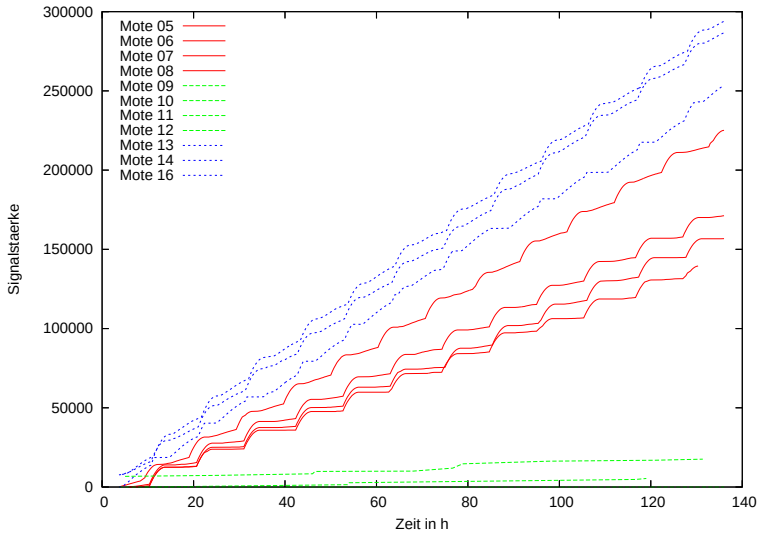


Abbildung 5.64: Aufsummierte gewichte Signalstärken von Mote 15 empfangener Nachrichten

Es ist klar, dass Mote 15 in dieser Abbildung fehlt, da er keine Nachrichten von sich selbst empfangen kann. Man kann feststellen, dass Mote 15 sowohl von alle blauen als

auch von alle roten Motes erreichbar ist. Nachrichten von grünen Motes hingegen werden selten empfangen, was sich in einer flachen Kurve niederschlägt. Die blauen Motes sind nur durch ihre größere Signalstärke bei kleinen Sendeleistungen gegenüber den roten Motes ausgezeichnet und heben sich dadurch stärker ab. So ist den blauen Motes ein größerer Verwandtschaftsgrad zuzuschreiben als den anderen Motes.

5.3.4 Stufe 3: Gruppenbildung

In den vergangen Stufen konnten an Hand von verarbeiteten Messwerten Verwandtschaftsgrade in verschiedenen Kanälen ermittelt werden. Diese Verwandtschaftsgrade werden in der dritten Stufe des Verarbeitungsmodells in eine Gruppeneinteilung umgesetzt. Als erfolgreich haben sich die folgenden Kanäle herausgestellt:

1. Differenz der Luftfeuchtigkeit aus Abbildung 5.42 (abgekürzt: DL)
2. Differenz der Temperatur aus Abbildung 5.43 (abgekürzt: DT)
3. Ereignisse in der Luftfeuchtigkeit aus Abbildung 5.59 (abgekürzt: EL)
4. Ereignisse in der Temperatur aus Abbildung 5.60 (abgekürzt: ET)
5. Ereignisse im Lichtbereich PAR aus Abbildung 5.61 (abgekürzt: ELP)
6. Ereignisse im Lichtbereich TSR aus Abbildung 5.62 (abgekürzt: ELT)
7. Signalstärke aus Abbildung 5.64 (abgekürzt: Sig)

Dazu zeigt Tabelle 5.1 die letzten Werte aus den jeweiligen Kanälen von Mote 15 exemplarisch. Für die verbliebenen Motes kann die folgende Analyse analog ausgeführt werden, wobei sich entsprechende Resultate ergeben, wie sich im nächsten Abschnitt zeigt.

Da die Ergebnisse der Kanäle unterschiedliche Einheiten besitzen, müssen sie zunächst in ein gemeinsames Format umgerechnet werden. Dies geschieht mit Hilfe der Separatoren.

Der Separator „Minimum-Maximum" erzeugt das Ergebnis aus Tabelle 5.2. Eine Zugehörigkeit eines Motes zur Gruppe von Mote 15 wird durch eine 1 symbolisiert.

	05	06	07	08	09	10	11	12	13	14	15	16
Kanal DL	0	0	0	0	0	0	0	0	1	1	1	1
Kanal DT	0	0	0	0	0	0	0	0	1	1	1	1
Kanal EL	0	0	0	0	0	0	0	0	0	0	1	0
Kanal ET	0	0	0	0	0	0	0	0	0	0	1	0
Kanal ELP	0	0	0	0	0	0	0	0	0	0	1	0
Kanal ELT	0	0	0	0	0	0	0	0	0	0	1	0
Kanal Sig	1	0	1	1	0	0	0	0	1	1	1	1

Tabelle 5.2: Gruppenkonfiguration von Mote 15 durch Separator „Minimum-Maximum"

	05	06	07	08	09	10	11	12	13	14	15	16
Kanal DL	6,98M	6,35M	6,28M	7,69T	7,44M	6,15M	8,68M	7,61M	1,41M	2,10M	0	1,25M
Kanal DT	10,45M	9,49M	10,86M	11,55M	14,19M	13,02M	14,68M	14,02M	2,51M	3,00M	0	2,71M
Kanal EL	-682	-196	-625	-754	-928	-931	-931	-913	-325	-775	1868	380
Kanal ET	-1225	-631	-850	-1504	-1924	-1924	-1909	-1915	-247	-1570	3848	791
Kanal ELP	-220	92	-190	128	-88	-487	-487	-526	-103	92	1100	-1
Kanal ELT	-330	84	-159	60	-30	-402	-438	-531	-177	75	1272	9
Kanal Sig	225164	139503	171232	156722	17563	5408	414	414	253576	286696	-	293991

Tabelle 5.1: Ergebnisse der Kanäle von Mote 15

Man sieht, dass sich dieser Separator nicht für alle Kanäle gleich gut eignet. Gut geeignet ist er für die zwei Kanäle DL und DT, die die Differenz aus den Sensordaten der Luftfeuchtigkeit bzw. Temperatur bilden und aufsummieren. Der Grund liegt in den zwei Niveaus, die sich deutlich für Motes im selben Büro und für Motes in anderen Büros bilden. Ein nicht so ausgeprägtes Verhalten besitzt der Kanal Sig in der Signalstärke. Deshalb unterlaufen diesem Separator Fehler bei den Motes 05, 07 und 08. Gänzlich ungeeignet ist dieser Separator für alle Kanäle, die auf Ereignissen beruhen - dies sind die Kanäle EL, ET, ELP und ELT. Dort werden je drei Fehler gemacht.

Die Tabelle 5.3 stellt das Ergebnis des Separators „Mittelwert" dar.

	05	06	07	08	09	10	11	12	13	14	15	16
Kanal DL	0	0	0	0	0	0	0	0	1	1	1	1
Kanal DT	0	0	0	0	0	0	0	0	1	1	1	1
Kanal EL	0	1	0	0	0	0	0	0	1	0	1	1
Kanal ET	0	1	0	0	0	0	0	0	1	0	1	1
Kanal ELP	0	1	0	1	0	0	0	0	0	1	1	1
Kanal ELT	0	1	0	1	0	0	0	0	0	1	1	1
Kanal Sig	1	0	1	1	0	0	0	0	1	1	1	1

Tabelle 5.3: Gruppenkonfiguration von Mote 15 durch Separator „Mittelwert"

Die Zugehörigkeit zur gleichen Gruppe wie Mote 15 wird durch eine 1 ausgedrückt. Genau wie der Separator „Minimum-Maximum" teilt dieser Separator die Motes der Kanäle DL und DT - also die Kanäle, die mit dem Komparator „Differenz" arbeiten - perfekt ein. Ein besseres, jedoch nicht vollständig perfektes Ergebnis wird für die ereignisvergleichenden Kanäle EL, ET mit einem Fehler und für die Kanäle ELP, ELT mit drei Fehlern berechnet. Das Ergebnis für den Kanal der Signalstärke deckt sich mit dem des Separators „Minimum-Maximum".

Das Ergebnis des Separators „Positiv" ist in Tabelle 5.4 aufgelistet, wobei eine 1 eine Zuteilung zur Gruppe von Mote 15 repräsentiert.

	05	06	07	08	09	10	11	12	13	14	15	16
Kanal DL	1	1	1	1	1	1	1	1	1	1	1	1
Kanal DT	1	1	1	1	1	1	1	1	1	1	1	1
Kanal EL	0	0	0	0	0	0	0	0	0	0	1	1
Kanal ET	0	0	0	0	0	0	0	0	0	0	1	1
Kanal ELP	0	1	0	1	0	0	0	0	0	1	1	0
Kanal ELT	0	1	0	1	0	0	0	0	0	1	1	1
Kanal Sig	1	1	1	1	1	1	1	1	1	1	1	1

Tabelle 5.4: Gruppenkonfiguration von Mote 15 durch Separator „Positiv"

Wie erwartet versagt dieser Separator bei den Kanälen DL, DT und Sig, die nur positive Werte aufweisen können. Auch bei den Kanälen EL, ET, ELP und ELT, für die dieser Separator konzipiert wurde, bringt es dieser Separator mit den gewählten Parametern auf zwei bzw. drei Fehler und erzielt damit nicht die erhofften Resultate.

Der Separator „Prozent" stellt eine Homogenisierung der Daten her, indem die Verwandtschaftsgrade prozentual ausgedrückt werden. Tabelle 5.5 zeigt diese Ergebnisse.

	05	06	07	08	09	10	11	12	13	14	15	16
Kanal DL	20	27	28	12	15	30	0	13	84	76	100	86
Kanal DT	29	36	26	22	4	12	0	5	83	80	100	82
Kanal EL	8	26	10	6	0	0	0	0	21	5	100	46
Kanal ET	12	22	18	7	0	0	0	0	29	6	100	47
Kanal ELP	18	38	20	40	26	2	2	0	26	38	100	32
Kanal ELT	11	34	20	32	27	7	5	0	19	33	100	29
Kanal Sig	76	47	58	53	5	1	0	0	86	86	100	100

Tabelle 5.5: Gruppenkonfiguration in Prozent Zugehörigkeit von Mote 15 durch Separator „Prozent"

Die weitere Interpretation dieser Daten wird dann der nächsten Verarbeitungsstufe überlassen.

5.3.5 Stufe 4: Gesamtergebnis

In Stufe 3 wurde gezeigt, wie alle Kanäle in homogene Gruppenkonfigurationen umgewandelt werden können. Auf deren Basis muss ein Gesamtergebnis gefunden werden. Für jeden Kanal kann ein geeigneter Separator gewählt werden. Es hat sich aber in allen Kanälen der Separator „Mittelwert" als der beste herausgestellt. Parallel zur Analyse der Lösungsverfahren auf den Ergebnissen des Separators „Mittelwert" werden auch die Lösungsverfahren auf den Ergebnissen des Separators „Prozent" untersucht.

Eine allgemeine Vorgehensweise ist allen Lösungsverfahren gemeinsam. Diese besteht darin, die Gruppenkonfiguration aller Kanäle aufzusummieren. In der Tabelle 5.3 entspricht dies einer spaltenweisen Summe. Die Summe des Separators „Mittelwert" ist in Tabelle 5.6 gebildet. Parallel dazu zeigt die Tabelle 5.7 statt der Summe den spaltenweise gebildeten Mittelwert des Separators „Prozent", der anschaulicher ist, aber im weiteren Verlauf keine weiteren Auswirkungen hat, da diese Skalierung für die folgenden Verfahren irrelevant ist.

	05	06	07	08	09	10	11	12	13	14	15	16
Summe	1	4	1	3	1	0	0	0	5	5	7	7

Tabelle 5.6: Allgemeine Vorgehensweise bei Mote 15: Summen des Separators „Mittelwert"

	05	06	07	08	09	10	11	12	13	14	15	16
Summe	28,9	26,1	31,7	31,3	10,3	7,3	1,0	2,6	51,7	48,3	100	63,1

Tabelle 5.7: Allgemeine Vorgehensweise bei Mote 15: Mittelwerte des Separators „Prozent"

Für die Lösung „Minimum-Maximum" liegt das Minimum bei 0 und das Maximum bei 7. Daraus folgt die Mitte beider Werte zu 3,5. Gruppiert man alle Motes mit größeren Summen dem Verfahren entsprechen, entsteht die endgültige Gruppenkonfiguration in Tabelle 5.8.

	05	06	07	08	09	10	11	12	13	14	15	16
Konfiguration	0	1	0	0	0	0	0	0	1	1	1	1

Tabelle 5.8: Lösung „Minimum-Maximum" für Mote 15

Dabei werden die Motes 13, 14, 15 und 16 korrekt und Mote 06 fälschlicherweise zur Gruppe von Mote 15 gezählt. Für Mote 06 signalisieren zu viele Kanäle (hier vier) eine falsche Zugehörigkeit. In dieser Konstellation toleriert dieses Verfahren nur bis zu drei fehlerhafte Kanäle. Dagegen liegt das Maximum bei 100 % und das Minimum bei 1 %, die Mitte also bei 50,5 % im Fall des Separators „Prozent". Tabelle 5.9 führt die entstehende Konfiguration auf.

	05	06	07	08	09	10	11	12	13	14	15	16
Konfiguration	0	0	0	0	0	0	0	0	1	0	1	1

Tabelle 5.9: Prozent-basierte Lösung „Minimum-Maximum" für Mote 15

Hier werden zwar keine Motes fälschlicherweise zur Gruppe von Mote 15 hinzugefügt, aber Mote 14 gehört nach dieser Lösung nicht zur Gruppe.

Da das Minimum bei der Lösung „Minimum-Maximum" bei 0 liegt, ändert sich nichts im Ergebnis der Lösung „Hälfte", die das Minimum per Definition auf 0 festlegt. Bei der prozent-basierten Summe liegt die Mitte bei 50 %, was aber am Ergebnis auch nichts verändert.

Der von der Lösung „Mittelwert" berechnete Mittelwert liegt bei 2,8. Das heißt, dass zusätzlich zu den korrekten Motes 13, 14, 15 und 16 fälschlicherweise die Motes 06 und 08 der Gruppe von Mote 15 zugeordnet werden.

	05	06	07	08	09	10	11	12	13	14	15	16
Konfiguration	0	1	0	1	0	0	0	0	1	1	1	1

Tabelle 5.10: Lösung „Mittelwert" für Mote 15

Für die auf dem Separator basierende Summe ergibt sich ein Mittelwert von 33,2 %, was zu einer perfekten Gruppierung, wie sie in Tabelle 5.11 dargestellt ist, führt.

	05	06	07	08	09	10	11	12	13	14	15	16
Konfiguration	0	0	0	0	0	0	0	0	1	1	1	1

Tabelle 5.11: Prozent-basierte Lösung „Mittelwert" für Mote 15

Für den Lösungsansatz „Maximale Zugehörigkeit" werden die Summen aller Motes als Zeilen einer Adjazenzmatrix eines gewichteten, gerichteten Nachbarschaftsgraphen

aufgefasst. Diese ist in Tabelle 5.12 abgebildet.

	05	06	07	08	09	10	11	12	13	14	15	16
05	**7**	6	3	**7**	2	1	1	1	1	2	0	0
06	5	**7**	**6**	5	2	3	1	1	0	3	4	2
07	3	**7**	**7**	**7**	2	2	2	2	0	2	2	0
08	5	**6**	5	**7**	3	3	3	2	1	2	3	0
09	2	2	2	3	**7**	**5**	**5**	**5**	0	2	1	1
10	3	3	3	3	**7**	**7**	**7**	5	0	0	0	0
11	2	2	2	2	6	**7**	**7**	6	0	0	0	0
12	3	2	2	2	6	6	**7**	**7**	0	0	0	0
13	1	0	0	2	0	0	0	0	**7**	5	**7**	5
14	1	2	0	2	0	0	0	0	5	**7**	**7**	4
15	1	4	1	3	1	0	0	0	5	5	**7**	**7**
16	1	3	2	2	2	0	0	0	6	5	**7**	**7**

Tabelle 5.12: Summen aller Motes

Die identifizierten Motes mit maximaler Zugehörigkeit sind rot eingefärbt. Dabei wurde die Variante des Lösungsverfahrens „Maximale Zugehörigkeit" verwendet, die den Referenzmote (in der Tabelle in der Diagonale zu finden) bei der Suche nach dem Maximum nicht betrachtet. Diese identifizierten Konfigurationen werden mit allen Motes ausgetauscht und daraus eine transitive Hülle gebildet. Das Ergebnis ist als Adjazenzmatrix in Tabelle 5.13 zu sehen.

	05	06	07	08	09	10	11	12	13	14	15	16
05	1	1	1	1	0	0	0	0	0	0	0	0
06	1	1	1	1	0	0	0	0	0	0	0	0
07	1	1	1	1	0	0	0	0	0	0	0	0
08	1	1	1	1	0	0	0	0	0	0	0	0
09	0	0	0	0	1	1	1	1	0	0	0	0
10	0	0	0	0	1	1	1	1	0	0	0	0
11	0	0	0	0	1	1	1	1	0	0	0	0
12	0	0	0	0	1	1	1	1	0	0	0	0
13	0	0	0	0	0	0	0	0	1	1	1	1
14	0	0	0	0	0	0	0	0	1	1	1	1
15	0	0	0	0	0	0	0	0	1	1	1	1
16	0	0	0	0	0	0	0	0	1	1	1	1

Tabelle 5.13: Korrekte Gesamtkonfiguration

Die Gruppenkonfigurationen stimmen perfekt mit der realen Verteilung aller Motes überein. Ein Grund dafür ist in jeder Gruppe ein Mote, der durch eine hohe Übereinstimmungsquote eine starke Attraktivität auf die anderen Motes ausübt. Für die rote Gruppe ist dies Mote 08, Mote 11 in der grünen Gruppe und Mote 15 stellt den attraktivsten Mote in der blauen Gruppe dar.

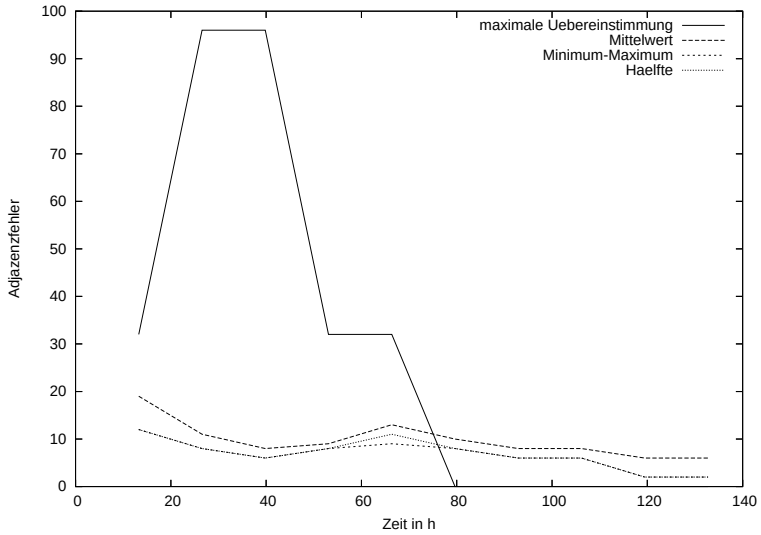


Abbildung 5.65: Adjazenzfehler der Lösungsverfahren

Abbildung 5.65 zeigt die Fehler aller Lösungsverfahren an mehreren Stellen in der Messung. Zu erkennen ist, dass das Lösungsverfahren „maximale Übereinstimmung" am Anfang große Schwierigkeiten besitzt, dann aber nach 3 Tagen eine vollständig korrekte Lösung liefert. Alle anderen Verfahren haben anfänglich kleinere Probleme, aber nach etwa drei Tagen ist eine sinkende Tendenz in den Fehlern zu erkennen. Beim Verfahren „Mittelwert" sind am Ende der Messung nur noch 6 Adjazenzfehler, bei den Verfahren „Minimum-Maximum" und „Hälfte" nur noch 2 Adjazenzfehler vorhanden. Selbst Stabilisierungsverfahren, wie das Austauschen von Gruppenkonfigurationen zwischen den Motes oder das Speichern und zusätzliche Verwenden von alten Gruppenkonfigurationen, können dieses Ergebnis nicht weiter verbessern. Parallel dazu gibt Abbildung 5.66 die Fehler für die Lösungsverfahren wieder, wenn sie auf die prozentualen Daten angewendet werden.

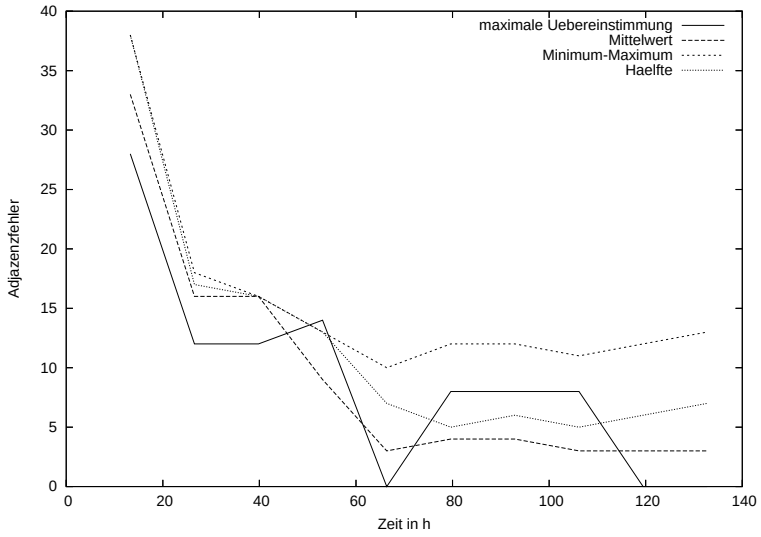


Abbildung 5.66: Adjazenzfehler der Lösungsverfahren mit prozentualen Daten

Als erstes fällt auf, dass das Lösungsverfahren „maximale Übereinstimmung" nicht kontinuierlich eine perfekte Lösung produziert. Auch die Lösungsverfahren „Minimum-Maximum" und „Hälfte" schneiden schlechter ab. Nur das Lösungsverfahren „Mittelwert" kann sich verbessern. Insgesamt werden am Anfang der Messung mehr Fehler gemacht.

Als Endergebnis steht jetzt eine Gruppenkonfiguration zur Verfügung. Die besten Lösungsverfahren dafür sind „maximale Übereinstimmung" auf binären Daten, „Mittelwert" auf prozentualen Daten und „Minimum-Maximum" auf binären Daten.

5.3.6 Verfahren aus der Biologie

Das Verfahren aus der Biologie arbeitet grundsätzlich direkt mit den Sensordaten, wie sie in Abbildung 5.3 für die Luftfeuchtigkeit, in Abbildung 5.4 für die Temperatur, in Abbildung 5.5 für den Lichtbereich PAR und in Abbildung 5.6 für den Lichtbereich TSR dargestellt sind. Es wird zu einem beliebigen Zeitpunkt aus den Sensordaten eines Motes ein Experiment-Vektor erstellt. Die Gesamtheit aller Experiment-Vektoren wird zufällig in eine vorbestimmte Anzahl von Gruppen verteilt und danach mit einem der Berechnungsvorschriften aus Abschnitt 4.3 in die Gruppen mit geringster Distanz umgeordnet. Die Tabellen 5.14, 5.15 und 5.16 zeigen die aus den Ergebnissen der einzelnen Berechnungsvorschriften interpretierten Gruppen in eine rote (r), grüne (g) und blaue (b) Gruppe bei einer Einteilung nach 36 Stunden, 60 Stunden bzw. 72 Stunden. Bei 36 Stunden sind bei allen Berechnungsvorschriften mittlere Fehler, bei 60 Stunden große Fehler und bei 72 Stunden keine Fehler zu finden. Eine Ausnahme machen die rang-basierten Berechnungsvorschriften 5 (Spearmans Rangkorrelationskoeffizient) und 6 (Kendall's τ). Durchgehend sind bei ihnen höhere Fehler zu beobachten. Besonders markant tritt dies bei 72 Stunden hervor, wo die höchste Fehlerquote verzeichnet wird und alle anderen Berechnungsvorschriften keine Fehler aufweisen.

	1	2	3	4	5	6	7	8
05	r	r	r	r	g	r	r	r
06	g	g	g	g	b	b	g	g
07	g	g	g	g	b	b	g	g
08	b	g	b	g	r	r	g	g
09	g	g	g	g	g	g	g	g
10	g	g	g	g	g	g	g	g
11	g	g	g	g	r	r	g	g
12	g	g	g	g	g	g	g	g
13	b	g	b	g	b	b	b	b
14	b	b	b	b	g	g	b	b
15	b	b	b	b	b	b	b	b
16	b	b	b	b	g	g	b	b
Fehler	3	4	3	4	6	5	3	3

Tabelle 5.14: Gruppeneinteilung nach 36 h

	1	2	3	4	5	6	7	8
05	r	r	r	r	g	g	b	b
06	g	g	g	g	r	r	g	g
07	g	g	g	g	b	b	g	g
08	g	g	g	g	b	b	r	r
09	g	g	g	g	g	g	g	g
10	g	g	g	g	g	g	g	g
11	g	g	g	g	g	g	g	g
12	g	g	g	g	r	r	g	g
13	r	r	r	r	b	b	b	b
14	g	g	g	g	b	b	r	r
15	b	b	b	b	g	g	b	b
16	b	b	b	b	g	g	b	b
Fehler	5	5	5	5	6	6	5	5

Tabelle 5.15: Gruppeneinteilung nach 60 h

	1	2	3	4	5	6	7	8
05	r	r	r	r	r	r	r	r
06	r	r	r	r	r	r	r	r
07	r	r	r	r	r	r	r	r
08	r	r	r	r	g	g	r	r
09	g	g	g	g	b	b	g	g
10	g	g	g	g	r	r	g	g
11	g	g	g	g	g	g	g	g
12	g	g	g	g	b	b	g	g
13	b	b	b	b	r	r	b	b
14	b	b	b	b	r	r	b	b
15	b	b	b	b	b	b	b	b
16	b	b	b	b	r	r	b	b
Fehler	0	0	0	0	7	7	0	0

Tabelle 5.16: Gruppeneinteilung nach 72 h

Tabelle 5.17 zeigt die Gruppeneinteilung und die damit verbunden Fehlentscheidungen an mehreren Zeitpunkten für die Berechnungsvorschrift 1 (Pearson Korrelationskoeffizient). Eine Unstetigkeit in den Fehlern kennzeichnet diese Gruppeneinteilungen.

	0 h	12 h	24 h	36 h	48 h	60 h	72 h	84 h	96 h	108 h	120 h
05	r	r	r	r	r	r	r	g	r	r	r
06	r	g	r	g	r	g	r	r	r	g	r
07	r	g	r	g	r	g	r	g	r	g	r
08	r	b	r	b	r	g	r	r	r	r	r
09	g	g	g	g	r	g	g	g	r	g	r
10	g	g	g	g	g	g	g	g	r	g	r
11	g	g	g	g	r	g	g	g	r	g	r
12	g	g	g	g	r	g	g	g	r	g	r
13	b	b	b	b	b	r	b	b	b	b	b
14	b	b	b	b	b	g	b	b	g	b	g
15	b	g	b	b	b	b	b	b	b	b	b
16	b	b	b	b	b	b	b	b	b	b	b
Fehler	0	4	0	3	3	5	0	2	5	2	5

Tabelle 5.17: Zeitliche Gruppeneinteilung durch die Berechnungsvorschrift Pearson Korrelationskoeffizient

Aus den bisherigen Ergebnissen ist ersichtlich, dass sich dieses Verfahren aus der Biologie nicht ohne weitere Anstrengungen auf unsere Bedürfnisse im Bereich Sensornetze übertragen lässt. Dazu sind zwei Ansätze denkbar:

1. Statt einer immer wiederkehrenden Einzelmessung werden an mehreren Zeitpunkten Sensorwerte gespeichert. Diese werden gesammelt ausgewertet. Dabei handelt es sich um eine zeitliche Erweiterung im Verfahren selbst.

2. Eine zeitliche Erweiterung außerhalb des Verfahrens lässt sich durch die Bildung eines Mittelwerts der Ergebnisse der immer wiederkehrenden Einzelmessungen verwirklichen. Dazu lassen sich die Lösungsverfahren aus Kapitel 4.2.6 verwenden.

Die innere Erweiterung liefert bei einem 12 h Intervall für die Einzelmessungen am Ende eine Gruppeneinteilung, wie sie in Tabelle 5.18 für jede Berechnungsvorschrift dargestellt ist. Zusätzlich sind noch die Fehlentscheidungen und die daraus resultierenden Fehler in der Adjazenzmatrix angegeben. Man sieht, dass die Fehler bei keiner Berechnungsvorschrift verschwinden.

	1	2	3	4	5	6	7	8
05	r	r	r	r	r	r	r	r
06	g	g	g	g	r	r	g	g
07	g	g	g	g	r	r	g	g
08	r	r	r	r	r	r	b	b
09	g	g	g	g	g	g	g	g
10	g	g	g	g	g	g	g	g
11	g	g	g	g	g	g	g	g
12	g	g	g	g	g	g	g	g
13	b	b	b	b	b	b	b	b
14	b	b	b	b	b	b	b	b
15	g	g	g	g	g	g	g	g
16	b	b	b	b	b	b	g	b
Fehler	3	3	3	3	1	1	5	4
Adjazenzfehler	42	42	42	42	14	14	62	34

Tabelle 5.18: Gruppeneinteilung und Fehler bei der inneren Erweiterung

Abbildung 5.67 zeigt die Fehler bei der äußeren Erweiterung in der Adjazenzmatrix für alle Berechnungsvorschriften zu mehreren Zeitpunkten. Wie man leicht sieht, steigt die Zahl der Fehler in den ersten 12 Stunden deutlich an. Danach sinkt sie leicht, bis sie sich nach etwa 3 Tagen stabilisiert.

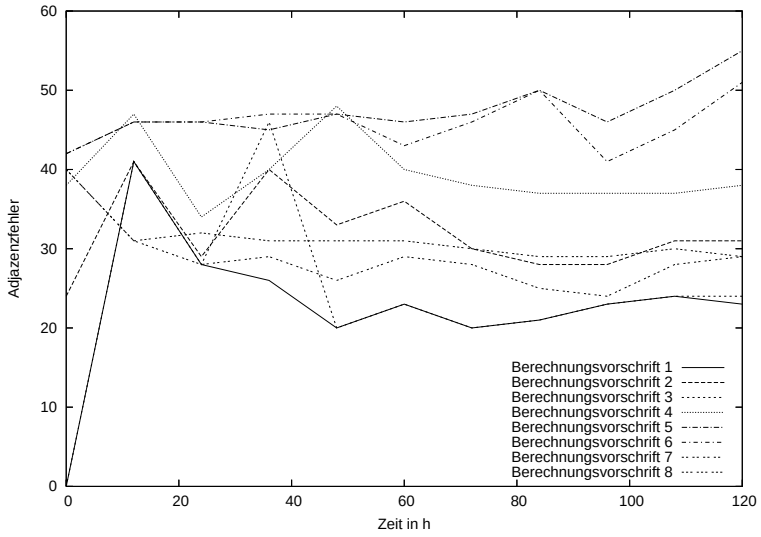


Abbildung 5.67: Fehler in der Adjazenzmatrix bei der äußeren Erweiterung

Zum größten Teil liegen die Fehler der äußeren Erweiterung unter denen der inneren Erweiterung. Aber dennoch sind sie weit höher als die Fehler des vierstufigen Verarbeitungsverfahrens. Diese liegen je nach Lösungsverfahren am Ende der Messung sogar bei unter 6, wie man aus Abbildung 5.65 ablesen kann.

5.3.7 Daten aus Intels Berkeley-Forschungslabor

Die Daten aus Intels Berkeley-Forschungslabor dienen zur Verifikation der gefundenen Resultate. An ihnen können die einzelnen Schritte des vierstufige Verfahrens, die sich zur Gruppeneinteilung eignen, nachgeprüft werden. In den Abbildungen 5.68, 5.69 und 5.70 sind die Daten für die Luftfeuchtigkeit, Temperatur und das Licht dargestellt. Alle Sensordaten weisen gewisse Ähnlichkeiten zu denen aus eigener Sammlung auf. So zeigen die Luftfeuchtigkeit und die Temperatur eine Korrelation. Bei den Daten des Lichts ist eine gute Unterscheidung zwischen Tag- und Nachtphasen möglich. Es sind auch die Effekte von unterschiedlich intensiver Bestrahlung der Motes erkennbar. Bei etwa 20 Stunden Messung treten in den grünen und roten Motes unerklärbare Erscheinungen auf, die kurzzeitig Werte wie tagsüber nicht nur bei den Sensordaten des Lichts, sondern auch in der Luftfeuchtigkeit und Temperatur erzeugen. Verglichen mit den eigenen Dateien scheinen insbesondere die Kurven der Luftfeuchtigkeit, aber auch der Temperatur ähnlicher zueinander zu sein.

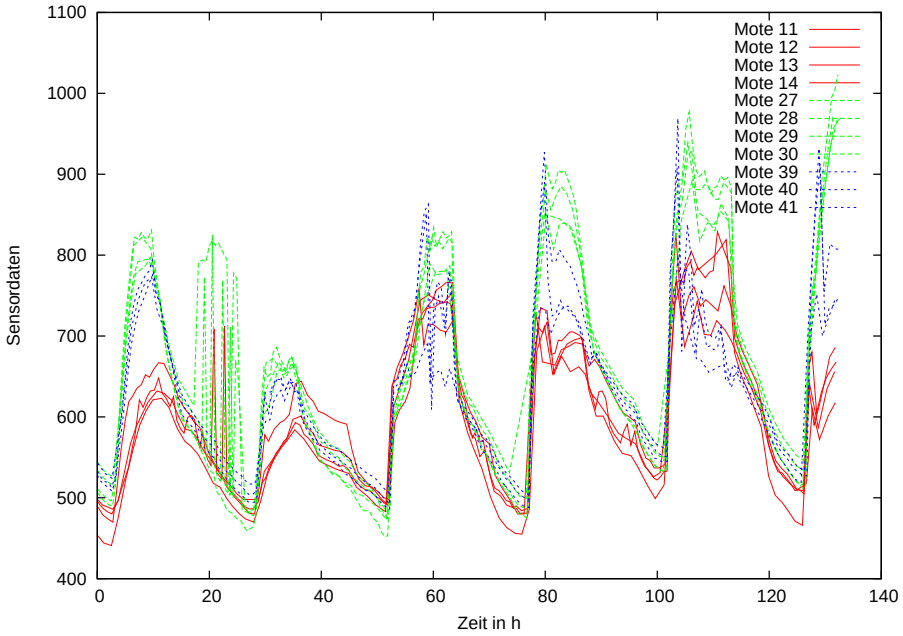


Abbildung 5.68: Sensordaten des Luftfeuchtigkeitssensors

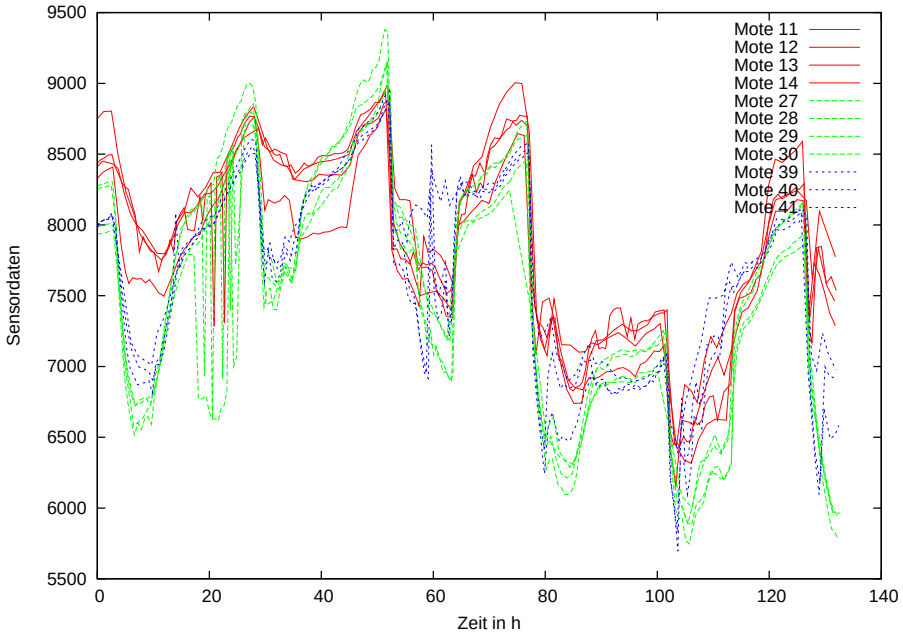


Abbildung 5.69: Sensordaten des Temperatursensor

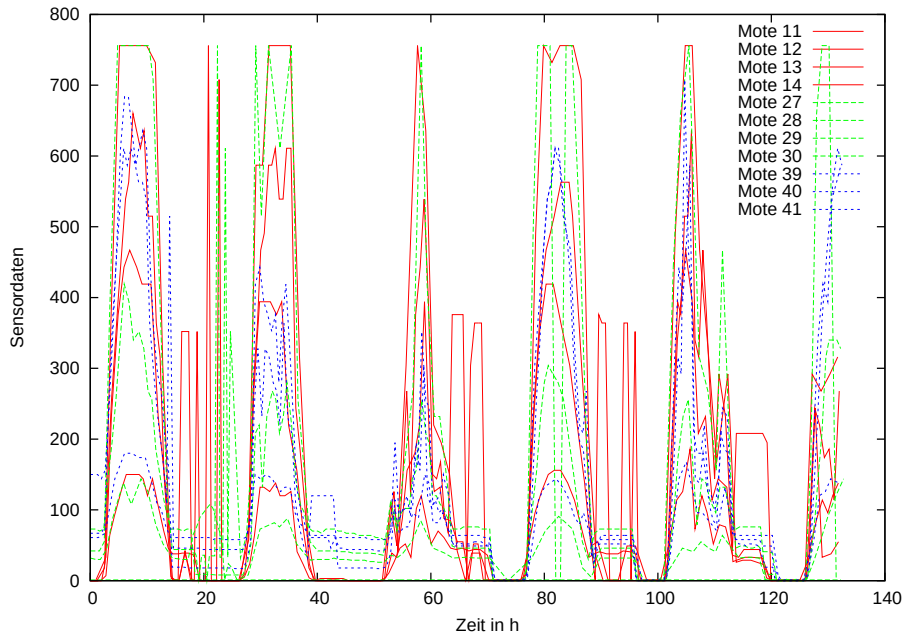


Abbildung 5.70: Sensordaten des Lichtsensors

Will man die erfolgreichen Analysemethoden auf die Sensordaten von Intel anwenden, stehen dafür fünf Kanäle zur Verfügung:

1. Differenz der Luftfeuchtigkeit
2. Differenz der Temperatur
3. Ereignisse in der Luftfeuchtigkeit
4. Ereignisse in der Temperatur
5. Ereignisse im Licht

Benutzt man auf den Ergebnissen dieser Kanäle den Separator „Mittelwert" zur Bestimmung der Gruppenzugehörigkeiten und bildet deren Summen, ergibt sich die Tabelle 5.19 für alle Motes.

	11	12	13	14	27	28	29	30	39	40	41
11	**5**	**4**	**2**	**2**	**3**	0	0	**3**	0	0	0
12	**2**	**5**	**4**	**2**	**2**	0	0	**2**	0	1	1
13	**3**	**5**	**5**	2	2	0	0	1	2	2	2
14	2	**4**	**3**	**5**	2	0	0	**3**	2	**3**	**3**
27	1	1	0	0	**5**	**2**	**2**	**5**	0	1	1
28	1	1	0	0	**3**	**4**	**4**	**4**	0	1	**3**
29	0	1	0	1	**4**	**3**	**5**	**5**	1	**4**	**4**
30	1	1	0	0	**4**	**2**	**3**	**5**	0	0	**2**
39	0	2	0	2	**3**	0	2	**3**	**5**	**5**	**5**
40	0	2	0	2	2	0	2	2	**4**	**5**	**5**
41	0	2	0	0	**3**	0	2	2	**3**	**5**	**5**

Tabelle 5.19: aufsummierte Konfigurationen

Unerwarteterweise kann man feststellen, dass bei Mote 28 nur vier Kanäle eine Übereinstimmung mit sich selbst signalisieren. Der fehlende Kanal ist Kanal 5, der die Ereignisse im Licht vergleicht. Während der ganzen Messung kann der Mote 28 kein Ereignis detektieren. Ansonsten weist die Tabelle insgesamt mehr Fehler auf als die eigenen Messungen. Mote 14 ist zum Beispiel für seine korrekte Gruppe zu unattraktiv. Dagegen sind die Motes 27 und 30 wegen Ereignissen in der Luftfeuchtigkeit und Temperatur sehr attraktiv für alle anderen Motes. Insbesondere äußert sich die Asymmetrie der Ereignisse bei Mote 27 klar, denn er erkennt seine eigene Gruppe sehr schlecht. Dies schlägt sich natürlich in den Ergebnissen der Lösungsverfahren nieder. Tabelle 5.20 zeigt die Anzahl deren Fehler in der Adjazenzmatrix.

Lösungsverfahren	Fehler
Maximale Übereinstimmung	0
Mittelwert	16
Minimum-Maximum	20
Hälfte	20

Tabelle 5.20: Adjazenzfehler bei den Intel-Berkeley-Daten

Als einziges Lösungsverfahren liefert „maximale Übereinstimmung" ein perfektes Ergebnis. Die sonstigen schlechten Ergebnisse resultieren hauptsächlich aus teilweise gehäuften Fehlern und der geringen Möglichkeit bei nur 5 Kanälen diese Fehler zu tolerieren. In Tabelle 5.19 sind die Lösungen des Verfahrens „Mittelwert" rot markiert. Leider sind die Gruppen nicht klar ausgebildet. Auch eine Erweiterung zur Stabilisierung bringt keine Besserung in das Ergebnis.

Ein Grund für das schlechte Abschneiden der Daten von Intels Berkeley-Forschungslabor ist sicherlich die nicht erfüllte Voraussetzung von Büroräumen. Dies schlägt sich besonders in die Temperatur und Luftfeuchtigkeit nieder, die im ganzen Großraumbüro annähernd einen gleichen Verlauf hat und deshalb keine lokalen Ereignisse erfasst werden können. Ein weiter Grund ist die geringe Anzahl von Kanälen, die nicht in der Lage ist, ausreichend Fehler zu tolerieren. Außerdem lässt sich keine Aussage über die Vorgänge

in dem Berkeley-Forschungslabor von Intel treffen.

5.4 Analyseergebnis

Basierend auf der Analyse der eigenen Daten ergeben sich einige teilweise unerwartete Ergebnisse.

Ein direkter Vergleich der Daten mit dem Komparator „Differenz" bringt erstaunlich gute Verwandtschaftsbeziehungen für die Temperatur und nochmals leicht bessere für die Luftfeuchtigkeit. Durch betragsweise Aufsummierung der Differenzen lassen sich diese festigen. Die beiden Lichtbereiche lassen sich auf diese Weise jedoch nicht vergleichen.

Für die Erfassung von Ereignissen muss für die Lichtsensoren anders vorgegangen werden als für Ereignisse in der Luftfeuchtigkeit und Temperatur. Bei der Luftfeuchtigkeit und Temperatur ist das Rauschen störend. Es muss durch Mittelwertbildung über einige Messschritte heraus gefiltert werden. Danach erst kann die Änderung aus den Daten herausgearbeitet werden. Da sich durch die Mittelung auch Ereignisse in die Länge ziehen, müssen häufig aufeinander folgende Änderungen wieder aufsummiert werden. Für die leichtere Verarbeitung wird der Betrag der Summe gebildet und so nicht mehr zwischen steigenden oder fallenden Kurven unterschieden. Betragswerte über einem gewissen Schwellenwert können jetzt als Ereignis aufgefasst werden.

Für die Erfassung von Ereignissen in den Lichtsensordaten werden prozentuale Änderungen betrachtet. Bei diesen Änderungen muss das Rauschen nur bei kleinen Werten, die meist in der Nacht auftreten, heraus gefiltert werden. Dies kann vorher mit einem Schwellenwert für die Sensordaten geschehen. Mit Hilfe eines weiteren Schwellenwerts werden große prozentuale Änderungen als Ereignisse aufgefasst.

Der Vergleich von Ereignissen gestaltet sich als Suche nach übereinstimmenden oder verpassten Ereignissen. Da beides komplementär zueinander ist, ist eine Kombination mit einer positiven Gewichtung für übereinstimmende und einer negativen Gewichtung für verpasste Ereignisse sinnvoll. Auch hier ist eine Aufsummierung aller Ereignisse zu einer Fehler tolerierenden Verwandtschaftsbeziehung angebracht, da Ereignisse recht selten sind und im Fall eines einzelnen Ereignisses auch keine klare Gruppeneinteilung zustande kommen kann.

Die darüber hinaus untersuchten Filter „Normierung", „Fourier-Transformation" und „Integral" enthalten keine zum Vergleich verwertbaren Eigenschaften. Für den Filter „Tendenz" gilt dasselbe, was aber erst bei der Auswertung des Komparators „Produkt" deutlich wird. Und auch die Komparatoren „Prozent" und „Kohärenz" erzeugen keine verwertbaren Verwandtschaftsbeziehungen.

Die stärkeren Signale bei niedriger Sendeleistung von Motes aus der gleichen Region sind für eine gute Verwandtschaftsbeziehung brauchbar. Dazu werden die Signalstärken invers zur Sendeleistung gewichtet. Auch hier ist ein Erinnerungseffekt in Form von Aufsummierung der einzelnen gewichtete Signalstärken hilfreich.

In der dritten Verarbeitungsstufe hat sich der Separator „Mittelwert" in allen Kanälen durchgehend bewährt. Der Separator „Minimum-Maximum" kann als Alternative bei der Gruppeneinteilung auf den Differenzen der Luftfeuchtigkeit und Temperatur und auf der Signalstärke genutzt werden. Als Alternative für die ereignis-basierten Kanäle

existiert der etwas schlechtere Separator „Positiv". Parallel dazu wandelt der Separator „Prozent" die Verwandtschaftsbeziehungen in homogene Verwandtschaftsgrade um.

Auf dem Separator „Mittelwert" oder alternativen Separatoren aufbauend erzeugen am Anfang der Messung die Lösungsverfahren „Minimum-Maximum" und „Hälfte" gleich gute Gruppenkonfigurationen. Das Lösungsverfahren „maximale Übereinstimmung" besticht nach einer Anlaufzeit von etwa drei Tagen durch seine perfekte Gruppenkonfiguration. Der parallel dazu laufende Ansatz, der auf den homogenen Daten des Separators „Prozent" basiert, favorisiert das Lösungsverfahren „Mittelwert". Die richtige Wahl eines Lösungsverfahrens hängt von den entsprechenden Gegebenheiten des Einsatzgebietes ab. Stehen wenige Kanäle zur Verfügung, sind die Verfahren „Minimum-Maximum" und „Hälfte" nicht in solchem Maße fehlertolerant, wie das Verfahren „maximale Übereinstimmung". Dagegen kann das Verfahren „Mittelwert" mit den Fehlern in den prozentualen Verwandtschaftsgraden gut umgehen.

Das zusätzlich untersuchte Verfahren aus der Biologie reicht in der Fehlerquote nicht an das vierstufige Verarbeitungsmodell heran. Dies gilt auch für alle untersuchten Erweiterungen diesen Verfahrens. Deshalb wird es als ungeeignet eingestuft.

Mit der Datensammlung des Berkeley-Forschungslabors von Intel können die als geeignet kategorisierten Methoden des vierstufigen Verarbeitungsmodells verifiziert werden. Die hier gefundenen Resultate weisen leider etwas mehr Fehler auf als die aus eigener Sammlung. Ein Grund könnte in der Beschaffenheit der Umgebung liegen, die sich nicht gänzlich mit den Voraussetzungen deckt.

6 Prototyp

Als Ergebnis dieser Arbeit steht eine prototypische Implementierung zur Gruppierung von Sensorknoten in einer Umgebung mit geographisch beschränkten Regionen wie die in Kapitel 5 untersuchten Büroräume. Diese Implementierung findet in der Sprache *nesC* statt und basiert auf den durch TinyOS geschaffenen Grundlagen. Das vierstufige Verarbeitungsverfahren mit den in Kapitel 5 gewonnenen Erkenntnissen wird die Gruppeneinteilung vornehmen.

6.1 Entwurf

Das vierstufige Verarbeitungsmodell eignet sich hervorragend für einen modularen Ansatz, der in Abbildung 6.1 aufgeführt ist. In der Implementierung erhält jede Stufe eine eigene Abstraktionsebene. Diese sind in der Abbildung als Kästen dargestellt, wobei die Stufen von unten nach oben angeordnet sind. Der Daten- und auch der Programmfluss laufen dementsprechend von unten ausgehend von den Sensordaten nach oben, bis die Daten zu einer Gruppenkonfiguration abstrahiert sind.

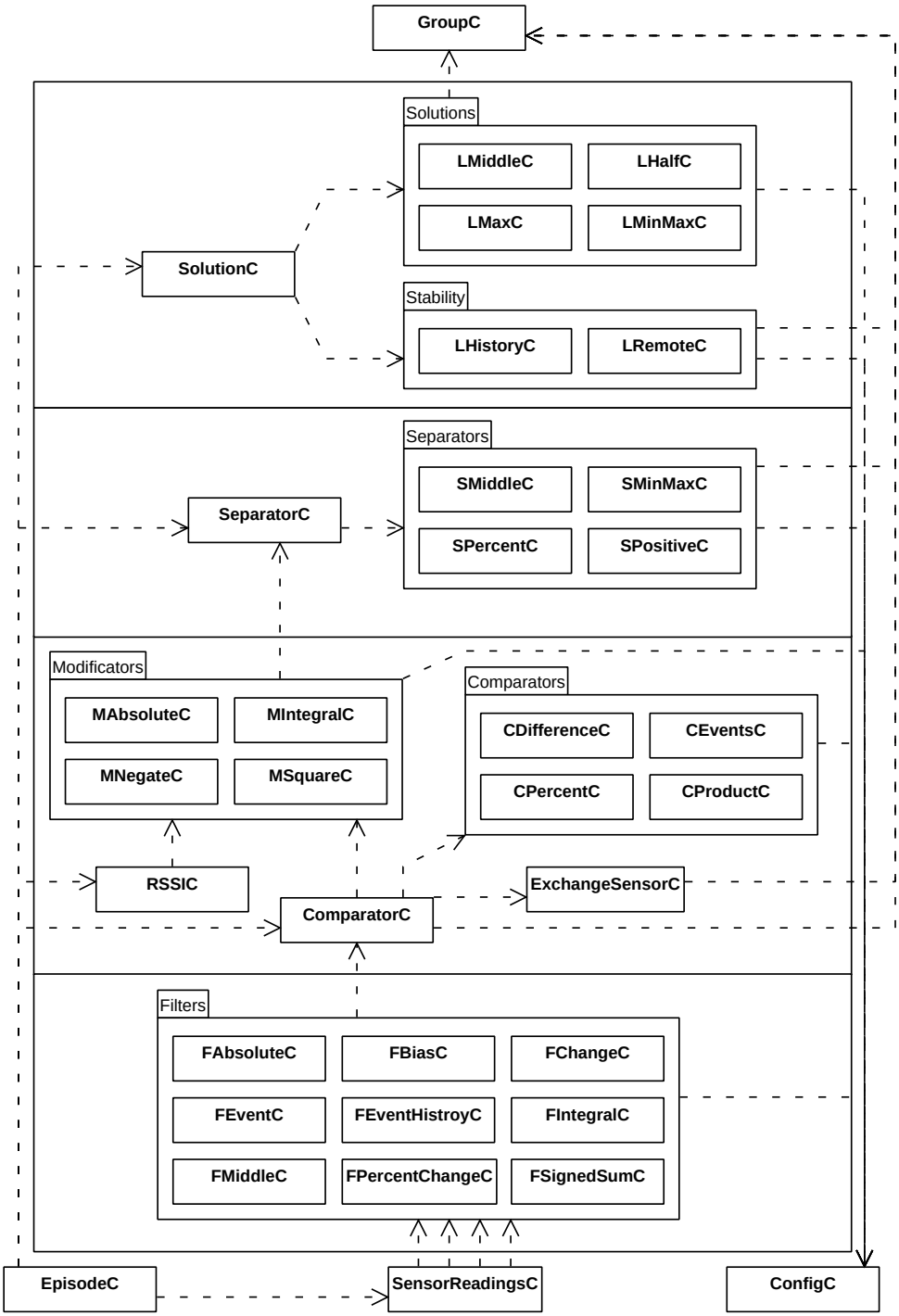


Abbildung 6.1: Entwurf der prototypischen Implementierung

Ein wichtiger konzeptioneller Punkt ist das Modul **`GroupC`**. Es stellt nicht nur am Schluss des Verarbeitungsprozesses die Gruppenkonfiguration zur Verfügung, sondern

dient als zentraler Speicher für Informationen, die mit fremden Motes zu tun haben. Durch die zentrale Datenhaltung wird die Speicherung redundanter Informationen, die andernfalls anfallen würde, vermieden. Dadurch wird dieses Modul parallel von allen Verarbeitungsstufen genutzt. Dies stellt einen Kompromiss zwischen einem modularen Software-Konzept, bei dem die Daten separat, direkt in jeder Verarbeitungsstufe gehalten werden, und einem speicher-sparenden Konzept dar, der auf Grund der Gegebenheiten der Motes eingegangen werden muss.

Auf unterster Ebene ist der Taktgeber **`EpisodeC`** zu finden. Er gibt die Sampling-Periode und damit auch den Takt für alle Verarbeitungsschritte an. Die dadurch definierten Intervalle sind nur für einen Mote gültig. Eine Synchronisation der Takte zwischen Motes ist nicht nötig. Jeder Stufe wird von unten nach oben der Beginn einer neuen Periode signalisiert. Angefangen wird beim Modul **`SensorReadingsC`**. Dieses Modul setzt die Sampling-Periode in Abtastungen der Sensoren um und propagiert diese Werte in die erste Verarbeitungsstufe. Dort sind zahlreiche Filtermodule zur Signalverarbeitung zu finden.

- **`FAbsoluteC`** stellt den Filter „Betrag" dar. Dazu benötigt es eine zyklische Schlange für die geschichtlichen Aufzeichnungen.
- **`FBiasC`** hat den Filter „Tendenz" als Vorbild. Bei positiven Werten erzeugt es die Ausgabe 1 und bei negativen -1.
- **`FChangeC`** implementiert den Filter „Änderung".
- **`FEventC`** setzt entsprechend dem Filter „Ereignis" Werte über einem gewissen Schwellenwert in binäre Ereignisse um.
- **`FEventHistoryC`** speichert die letzten drei Ergebnisse des Moduls **`FEventC`**, um einen zeitlich späteren Vergleich zu ermöglichen. Dazu werden die binären Ereignisse in einer Variable als Bits gespeichert. In jedem Takt werden die Bits geschoben um einem neuen Ereignis Platz zu schaffen.
- **`FIntegralC`** stellt den Filter „Integral" dar.
- **`FMiddleC`** gibt den Filter „Mittelwert" wieder.
- **`FPercentChangeC`** realisiert den Filter „Prozentuale Änderung".
- **`FSignedSumC`** bildet den Filter „Integral vorzeichengleicher Werte" nach.

Diese Filter bieten und benutzen eine Schnittstelle **`Filter`**, die eine einzige Signalmethode **`process()`** besitzt. Über die benutzende Schnittstelle eingehende Daten werden in dieser Methode verarbeitet und anschließend über die angebotene Schnittstelle an nachfolgende Module signalisierend weitergegeben. Dadurch können die Filter in beliebiger Reihenfolge miteinander kombiniert werden.

In der zweiten Verarbeitungsstufe ist **`ComparatorC`** das zentrale Modul. Es stellt den Abschluss der Filterkette dar und nimmt den Vergleich der Daten vor. Dazu werden zum einen die eigenen Daten durch das Modul **`ExchangeSensorC`** an alle Nachbarn verteilt und zum anderen von Nachbarn eingehende Daten zum Vergleich im Modul **`GroupC`**

gespeichert. Befindet sich der Sender der Daten in der eigenen Gruppe und sind die Daten neu, leitet der Mote die Daten einmal weiter. Dadurch werden Daten innerhalb einer Gruppe geflutet.

Auf das Taktsignal von **EpisodeC** hin, beschafft sich das Modul **ComparatorC** die aktuellsten eingegangenen Daten aus dem Modul **GroupC**. Der eigentliche Vergleich dieser Daten wird für jeden Kanal individuell von einem der Komparatoren übernommen. Diese Komparatoren implementieren eine Schnittstelle **Compare**, die eine Kommandomethode **compare()** besitzt um zwei Werte entgegen zu nehmen und ein Resultat des Vergleichs zurückzugeben.

- **CDifferenceC** implementiert die Funktionalität des Komparators „Differenz".
- **CEventsC** ist ein generisches Modul. Es stellt den Vergleich von Ereignissen an, gewichtet entsprechend ein übereinstimmendes oder verpasstes Ereignis und summiert diesen Wert auf.
- **CProductC** stellt die Funktionalität des Komparators „Produkt" bereit.
- **CPercentC** realisiert den Komparator „Prozent".

Das Ergebnis dieses Vergleichs wird zur weiteren Verarbeitung an die Modifikatoren gegeben.

Parallel zum Vergleich von Sensordaten läuft in der zweiten Verarbeitungsstufe die Erfassung der Signalstärken. Das Modul **RSSIC** wird von **LinkRadio**, einem TinyOS Modul zur Nachrichtenverarbeitung, über eingehende Nachrichten informiert. Aus diesen Nachrichten wird der Sender und die Sendeleistung ausgelesen und die Signalstärke bestimmt. Von der Signalstärke wird das Minimum aller mit derselben Sendeleistung gesendeten Nachrichten abgezogen, um die Unterschiede deutlicher herauszuarbeiten und den Wertebereich klein zu halten. Diese Werte werden invers zur Sendeleistung gewichtet und an Modifikatoren weitergegeben:

$$Vergleich = (32 - Sendeleistung) \cdot (Signalstaerke - Minimum[Sendeleistung])$$

Um Motes, die das Netz mit Nachrichten fluten, nicht zu tolerieren, werden die gewichteten Signalstärken nur einmal pro Takt ausgewertet. Das hat zu Folge, dass nur die letzte empfangene Nachricht für die weitere Berechnung von Bedeutung ist. Zusätzlich simuliert der Mote einmal pro Takt eine Nachricht von sich selbst. Diese weist die höchstmögliche Signalstärke auf und berechnet sich zu $(32 - geringsteSendeleistung) \cdot (256 - Minimum[geringsteSendeleistung])$.

Modifikatoren bieten und benutzen die Schnittstelle **Modificator** mit der Signalmethode **process()**. Die Modifikatoren ähneln sehr den Filtern. Dadurch ist eine Benutzung wie bei den Filtern möglich, die es zulässt, Modifikatorketten zu bilden. Die verfügbaren Modifikatormodule sind:

- **MAbsoluteC** implementiert den Modifikator „Betrag".
- **MIntegralC** bildet dem Modifikator „Integral" entsprechend die Summe aus Vergleichswerten. Dieser Modifikator greift auf seine alten Werte im zentralen Speicher **GroupC** zurück und sollte deshalb als letzter in der Modifikatorkette stehen.

- **MNegateC** negiert die Vergleichswerte. Dadurch werden große Werte zu kleinen Verwandtschaftsgraden. Dies erleichtert eine Implementierung der dritten Verarbeitungsstufe.
- **MSquareC** realisiert den Modifikator „Quadrat".

In der dritten Verarbeitungsstufe bildet das Modul **SeparatorC** den Mittelpunkt. Diese Modul stellt zum einen den Abschluss der Modifikatorkette dar, indem es die Vergleichswerte im zentralen Speicher **GroupC** zwischenspeichert. Zum anderen wird es vom Taktgeber regelmäßig dazu angestoßen, eine Gruppeneinteilung aller Kanäle zu unternehmen. Dazu delegiert es die Entscheidung pro Kanal an eines der Separatormodule ab:

- **SMiddleC** stellt die Funktionalität des Separators „Mittelwert" zur Verfügung.
- **SMinMaxC** teilt die Motes mit Vergleichswerten oberhalb der Mitte zwischen Minimum und Maximum dem Separator „Minimum-Maximum" entsprechend ein.
- **SPercentC** bildet die Vergleichswerte nach dem Separator „Prozent" auf einen prozentualen Bereich ab.
- **SPositiveC** gruppiert alle Motes mit positiven Vergleichswerten.

Alle Separatormodule implementieren die Schnittstelle **Separate** mit der Kommandomethode **separate()** und der Signalmethode **group()**, die das Module **SeparatorC** veranlasst einen Mote als zugehörig zur Gruppe zu erfassen. An dieser Stelle implementiert **SeparatorC** auch schon die „Allgemeine Vorgehensweise" aus der vierten Verarbeitungsstufe (Abschnitt 4.2.6), indem die Summe der Gruppenzugehörigkeit signalisierenden Kanäle gebildet wird.

Der vierten Verarbeitungsstufe verbleibt die Aufgabe, die Summe der Gruppenzugehörigkeit in eine endgültige Gruppenkonfiguration zu verwandeln. Dazu muss eines aus mehreren Lösungsmodulen ausgewählt werden, die die Schnittstelle **Solution** mit der Kommandomethode **solve()** implementieren. Diese wird in regelmäßigen Takten aufgerufen.

- **LMiddleC** realisiert die Lösung „Mittelwert".
- **LHalfC** implementiert die Lösung „Hälfte".
- **LMinMaxC** stellt die Lösung „Minimum-Maximum" dar.
- **LMaxC** setzt die Lösung „Maximale Übereinstimmung" mit der Erweiterung, den Referenzmote bei der Suche nach dem Mote mit der größten Summe nicht zu berücksichtigen, um. Diese Wahl wird an alle Nachbarn gesendet. Außerdem ist es für die Speicherung der eingehenden Wahlen anderer Motes verantwortlich, die bei der Lösungsfindung zur Bildung der transitiven Hülle verwendet werden.

Zur Stabilisierung der Lösung stehen folgende zwei Module zur Verfügung. Diese implementieren ebenfalls die Schnittstelle **Solution**, können aber nicht als Ersatz für die Lösungsmodule eingesetzt werden, da sie ausschließlich zur Stabilisierung dienen.

- **SHistoryC** speichert vergangene Summen aus der „Allgemeinen Vorgehensweise". Bevor die Lösungsmodule ihre Arbeit aufnehmen, wird die aktuelle Summe mit den gespeicherten Informationen erhöht.
- **SRemoteC** tauscht die Summen aus der „Allgemeinen Vorgehensweise" mit anderen Motes aus. Dazu versendet es eine Nachricht mit den Mote IDs, die die größten Summen aufweisen. Andersherum werden eingehende Summen überprüft, ob sich der Mote selbst unter den empfangenen Mote IDs befindet, und dann gespeichert. Vor der Lösungsfindung durch die Lösungsmodule werden die alle empfangenen Summen auf die aktuelle Summe addiert.

Am oberen Ende des Verarbeitungsmodells steht das Modul **GroupC**. Es dient als zentraler Speicher aller in allen Verarbeitungsstufen anfallenden Ergebnisse, die verschiedenen Motes zugeordnet sind. Da auf Motes unter TinyOS keine dynamische Speicherverwaltung verwendet wird, wird eine vorgegebene Menge Speicher reserviert. Sobald eine Nachricht von einem bisher unbekannten Mote empfangen wird, wird dieser Mote registriert. Ist nicht mehr genügend Platz vorhanden, wird ein Mote mit einer kleinen Übereinstimmungssumme ersetzt. Da es möglich ist, Verarbeitungsschritte mit einem Erinnerungseffekt auszustatten, werden zusätzlich noch alle Vergleichsinformationen im nächsten Takt zurückgesetzt, um den neuen Mote durch sein spätes Einsteigen nicht zu benachteiligen. Außerdem werden alle Informationen von Motes, die sich über eine lange Zeit nicht melden, aus dem Speicher entfernt. Dazu werden Epochen eingeführt, in denen sich ein Mote melden muss. Zusätzlich wird ein Request/Response Prinzip angeboten, dass dem Benutzer erlaubt, eine Anfrage nach der aktuellen Gruppenkonfiguration zu stellen. Auf eine Anfrage hin werden Nachrichten mit den Mote IDs gefüllt, die zur Gruppe gehören, und an den Anfragesteller als Antwort zurückgesendet.

Als Zusatz steht das Module **ConfigC** dem Benutzer zur Abfrage und zum Setzen von Parametern und dem Status einzelner Verarbeitungsmodule zur Verfügung. Dazu implementiert es ein Request/Response Prinzip, das über die drahtlose Schnittstelle genutzt werden kann. Die Verarbeitungsmodule implementieren für ihre Parameter und Statusvariablen die Schnittstelle **Parameter** mit den Methoden **get()** zur Abfrage und **set()** zum Setzen und melden sie bei **ConfigC** an.

6.2 Implementierung

Die Implementierung der meisten Module ist ohne unlösbaren Probleme möglich. Nur bei den Modulen in der untersten Abstraktionsebene sind die Charakteristika von TinyOS ausschlaggebend für die Umsetzung und bilden Schwierigkeiten, die es zu lösen gilt.

Eine Schwierigkeit für das Modul **SensorReadingsC** liegt in der Art und Weise, wie Daten von den Sensoren abzufragen sind. Dies geschieht nach dem Mechanismus der *Split-Phase*: Durch einen Funktionsaufruf wird ein Auftrag erteilt. Das Ergebnis des Auftrags wird verzögert ausgeliefert, indem eine andere Rückruf-Funktion ausgeführt wird. Dadurch ist es nicht möglich, sequentiellen Code zu schreiben, der einen Sensor nach dem anderen abfragt. Aus diesem Grund wird ein Zustandsautomat verwendet, der sich die Stelle im Programmablauf merkt, wenn ein Sensor nach seinen Daten gefragt wird.

Nachdem alle aktuellen Sensorwerte vorliegen, werden durch diesen Automat die Daten zu den Filtern propagiert.

Ein Problem für den Taktgeber **`EpsiodeC`** stellen Module da, die Tasks benutzen. Durch sie ist es dem Taktgeber nicht möglich, in einer Schleife etwa alle Module von unterster Verarbeitungsstufe an nach oben durchgehend zu signalisieren. Da obere Verarbeitungsstufen von den Daten unterer Verarbeitungsstufen abhängen und untere Verarbeitungsstufen durchaus von Tasks Gebrauch machen und dadurch verzögert ausgeführt werden, kann es eine Dateninkonsistenz zur Folge haben. Um den Anwendungsprogrammierer von dieser Gefahr zu schützen, werden den Verarbeitungsmodulen, die vom Taktgeber zu signalisieren sind, Prioritäten vergeben. Zu Beginn jeden Taktes wird das Verarbeitungsmodul mit höchster Priorität gestartet. Dieses teilt dem Taktgeber kurz vor Ende den Abschluss seiner Operation mit. Daraufhin wird das Verarbeitungsmodul mit nächstniedrigerer Priorität gestartet und so weiter. Dadurch wird ein serieller Datenfluss erzwungen, der Inkonsistenzen vermeidet.

Für die Messung der Signalstärke im Modul **`RSSIC`** ist eine Änderung an TinyOS notwendig. Der Nachrichtenkopf muss um zwei Felder erweitert werden. Das erste zusätzliche Feld enthält den Absender der Nachricht. Dies wird für die Zuordnung von eingehenden Nachrichten zu einem Mote benötigt. Das zweite Feld wird vor dem Senden mit der Sendeleistung gefüllt, die beim Empfänger zur Gewichtung der Signalstärke verwendet wird. Außerdem wird die Sendeleistung vor dem Senden entsprechend angepasst. Als Anmerkung sei hier erwähnt, dass die regelmäßig gesendeten Nachrichten, die zum Vergleich aufbereitete Sensordaten enthalten, auch für die Bestimmung der Signalstärke herangezogen werden können. Daher ist das Verfahren nicht unbedingt auf andere Nachrichten angewiesen, die ausschließlich den Zweck erfüllen, Signalstärken bestimmen zu können.

Da der Algorithmus von Warshall [War62] zur Berechnung der transitiven Hülle als zentraler Algorithmus ausgelegt ist, muss eine Anpassung vorgenommen werden, um eine parallele Ausführung auf mehreren Motes zu erlauben. Für einen Mote ist nur der Teil der kompletten transitiven Hülle nötig, der seine Gruppenzugehörigkeit angibt. Algorithmus 1 zeigt, wie ein Mote ausgehend von sich als einziges Gruppenmitglied andere Gruppenmitglieder findet. Im ersten Schleifendurchlauf der Schleife über k werden die Motes zur Gruppe hinzugefügt, die er oder die ihn als „maximale Übereinstimmung" identifiziert haben. Im zweiten Schleifendurchlauf werden die Motes zur Gruppe hinzugefügt, die die bereits hinzugefügten oder die einen der bereits hinzugefügten Motes als „maximale Übereinstimmung" identifiziert haben, also über zwei Ecken verwandt sind, usw.

```
Gruppenmitglied(Selbst) = Wahr
für k = 1 bis #Motes {
  für j = 1 bis #Motes {
    maxMote = maximaleÜbereinstimmung(Motes[j])
    falls Gruppenmitglied(Motes[j]) dann
      Gruppenmitglied(maxMote) = Wahr
    falls Gruppenmitglied(maxMote) dann
      Gruppenmitglied(Motes[j]) = Wahr
  }
}
```

Algorithmus 1: angepasster Warshall-Algorithmus zur parallelen Ausführung auf Motes

TinyOS bietet mit Hilfe der *nesC*-Konfigurationen eine gute Möglichkeit Kanäle zu definieren. Dabei kann ein Kanal in einer einzigen Konfigurationsdatei alle benötigten Module von **SensorReadingsC** über die Filter, Komparatoren, Modifikatoren und den Separator bis **SolutionC** instanzieren und miteinander verbinden. Da das Lösungsverfahren unabhängig von einem Kanal ist, wird es zentral in **SolutionC** festgelegt und instanziert. Eine so zusammengestellte Konfiguration kann einfach in die Konfiguration **ChannelsC** eingetragen werden, die dann den Kanal selbst instanziert. Die komplette Implementierung wurde als Bibliothek angelegt. So kann ein Anwendungsprogrammierer die Bibliothek durch einfache Instanzierung der Konfiguration **GroupingC** einbinden. Diese instanziert den Taktgeber **EpisodeC**, die Kanäle in **ChannelsC**, das Konfigurationsmodul **ConfigC** und das Gruppenmodul **GroupC**.

Um die Implementierung nachvollziehbar und wiederholbar zu testen, wurde der Simulator von TinyOS so modifiziert, dass er die gesammelten Daten aus einer Datei einliest und als simulierte Sensordaten zur Verfügung stellt. Dazu reicht es aus, die Module, die von **SensorReadingsC** zur eigentlichen Ansprache der Sensoren benutzt werden, und das Modul **LinkRadio**, das das Module **RSSIC** über den Empfang Nachrichten informiert, zu überschreiben. Für die Sensormodule muss der *Split-Phase*-Mechanismus nachgebildet werden, so dass bei einem Auftrag Sensordaten aus der Datei ausgelesen werden und dem Modul SensorReadingsC durch eine Methode zugeschoben werden. Bei dem Nachrichtenmodul reicht ein Zeitgeber, der periodisch Nachrichten mit Sender, Sendeleistung und Signalstärke nach eine Vorlage aus Datei generiert.

6.3 Ergebnis

Bei der Implementierung für Motes lassen sich die gleichen Ergebnisse feststellen, wie sie sich in Kapitel 5.3.5 ergeben. Am Anfang produziert das Lösungsverfahren „maximale Übereinstimmung" mehr Fehler als die Lösung „Mittelwert". Jedoch verringern sich die Fehler beider Verfahren, wie erwartet, im Verlauf der Simulation.

Der Speicherverbrauch, der sich durch die Einbindung der implementierten Bibliothek zur Gruppierung von Sensorknoten auf Basis von Sensordaten und Konnektivität ergibt, ist für die TMote Sky in Tabelle 6.1 aufgelistet. Dabei bezeichnet ein leeres Programm

eine Anwendung ohne die Einbindung der Bibliothek. Bestandteil dieses Programms ist lediglich ein Modul, das periodisch Sensordaten abfragt und in Nachrichten versendet. Die zweite Zeile gibt eine Konfiguration wieder, die für alle Gruppeneinteilungen mit dem Separator „Prozent" und zur Bestimmung der Gesamtlösung das Verfahren „Mittelwert" benutzt. Die Konfiguration, die mit dem Separator „Mittelwert" und dem Lösungsverfahren „maximale Übereinstimmung" eine perfekte Lösung erzeugt, ist in der dritten Zeile aufgelistet.

Programm	ROM (Byte)	RAM (Byte)
Leeres Programm	22 066	681
`LMiddleC` mit **`SPercentC`**	27 786	1 681
`LMaxC` mit **`SAvergeC`**	27 994	1 755

Tabelle 6.1: Speicherverbrauch

Es zeigt sich, dass das Lösungsverfahren „maximale Übereinstimmung" sowohl leicht mehr Programmspeicher als auch leicht mehr Arbeitsspeicher als das Lösungsverfahren „Mittelwert" erfordert. Der Verbrauch an Arbeitsspeicher hängt bei beiden Verfahren linear von der Anzahl der Motes ab, die potentiell eine Gruppe bilden. Dieser Verbrauch lässt sich mit der Berechnungsvorschrift $\#Motes \cdot (10 + 6 \cdot \#Kanaele)$ Bytes angeben. In der Tabelle 6.1 sind im Arbeitsspeicher 12 Motes bei 7 Kanälen enthalten. Aus diesen Werten ergibt sich eine Bibliotheksgröße von nur 5,78 KB und ein geringer Arbeitsspeicherverbrauch von 450 Byte zuzüglich des Bedarfs pro Gruppenmote, der von der jeweiligen Einsatzumgebung abhängt.

Als Ergebnis kann festgehalten werden, dass sich das vierstufige Verarbeitungsmodell hervorragend für ein strukturiertes Software-Design eignet. Dieses Design kann in eine effiziente Implementierung umgesetzt werden. Die Implementierung lässt sich durch den Simulator wiederholbar testen und erfüllt auch die beschränkten Speicheranforderungen der Motes.

7 Ausblick

Aufbauend auf der Möglichkeit Gruppen von Sensorknoten auf Basis von Sensordaten und Konnektivität zu bilden, die geographische Regionen reflektieren, existieren zahlreiche Anwendungen. Diese Anwendungen zählen einerseits zum Themengebiet Lokalisierung, zum andern zum Themengebiet Clustering.

Um die Gruppen für eine Lokalisierung zu nutzen, muss beides enger miteinander verschmelzen. Eine Möglichkeit sind symbolische Koordinaten, die eine Gruppe oder Region beschreiben. Dies kann durch Abbilden der Gruppen auf die entsprechenden Regionen geschehen. Zum Beispiel wäre dies für die in Kapitel 5 zur Datensammlung benutzten Verteilung eine Abbildung auf einen Bürogrundriss, wie er in Grafik 5.1 skizziert ist. Dort bieten sich die Raumnummern als symbolische Koordinaten an. Hat man erst eine solche Lokalisierung, lässt diese sich für Routingverfahren nutzen. Ein Routing-Pfad besteht dabei aus Raumnummern. Entweder gibt ein Anwender, der Informationen aus einem Raum haben will und Kenntnisse über den Gebäudegrundriss besitzt, den Pfad als Source-Route vor oder benutzt die Routingtabellen, die auf Motes vorhanden sind und die Büronachbarschaften wiedergeben. Möglicherweise lassen sich aber auch solche Nachbarschaften aus Messungen ableiten. Dazu muss das vierstufige Verarbeitungsmodell erweitert werden, um statt nur Gruppenzugehörigkeiten auch Gruppennachbarschaften zu erkennen. Als Vision lässt sich an eine Lokalisierung von Büroräumen in einem Gebäude denken, die zum Beispiel aufgrund der direkten Sonneneinstrahlung und den damit verbundenen höheren Temperaturen erkennt, dass Büroräume an der Süd- oder Nordseite liegen. Dann lässt sich vermutlich auch ein Unterschied zwischen Ost- und Westseite feststellen, der auf den Sonnenverlauf zurückzuführen ist und zeitlich versetzte Licht- und Temperaturmessungen im Lauf eines Tages hervorruft.

Selbstverständlich lassen sich Gruppen zu Clustern erweitern, indem ein Clusterhead aus der Mitte einer Gruppe gewählt wird. Die resultierenden Vorteile, wie Redundanz, Kombination von Fähigkeiten unterschiedlich ausgestatteter Motes oder Möglichkeit zur Energiesparung durch Abschaltung einzelner Motes, folgen daraus.

Im Anschluss an diese Arbeit ist eine großflächige Messung nahe liegend. Diese sollte klären, ob eventuell Skalierungsprobleme in einer Stufe des Verarbeitungsmodells auftreten. Auch kann eine solche Messung bei der Optimierung der zahlreichen Parameter, die miteinander verschränkt sind, helfen. Als weiteres Forschungsthema ist eine Untersuchung, ob sich Parameter dynamisch bestimmen lassen, denkbar. Dabei geht man von einer kleinen Anzahl von Motes in einer Region aus, die vertrauenswürdige, aufgespielte Informationen über ihre Gruppenmitglieder besitzen. Diese können untereinander die Parameter dynamisch so optimieren, dass die aufgespielten Gruppeninformationen zutreffen. Eine Verwendung der Parameter findet dann bei Motes statt, die später ohne Gruppeninformationen in das Netzwerk integriert werden.

Wie sich in dieser Arbeit herausgestellt hat, lassen sich Ereignisse im Licht erkennen und auf die zeitliche Übereinstimmung prüfen. Da sich die künstlichen Lichtereignisse

An- und Ausschalten von Bürobeleuchtung im gesamten Raum gleichzeitig ausbreiten, ist eine Zeit-Synchronisierung auf diesen Ereignissen möglich. Um eine hohe Genauigkeit zu erreichen, sind wesentlich kürzere Sampling-Perioden notwendig als die, die in Kapitel 5 zur Langzeit-Datensammlung dienten. Da eine so häufige Abfrage sehr viel Energie benötigt, muss über eine frühzeitige Ankündigung eines solchen Ereignisses nachgedacht werden. Als Lösung könnte der Lichtschalter eine Alarmnachricht senden und erst leicht verzögert das Licht schalten.

8 Zusammenfassung und Fazit

In Sensornetzen mit einer großen Menge von Sensorknoten ist in vielen Anwendungen eine Gruppierung oder eine Lokalisierung der Knoten notwendig. Unterschiedliche Möglichkeiten wurden bereits erforscht und einige davon in dieser Arbeit vorgestellt. Über die Verwendbarkeit von Sensordaten und Konnektivität zur Gruppierung von Sensorknoten war wenig bekannt. Zu diesem Gebiet wurden deshalb verschiedene Verfahren untersucht. Zum einen wurde ein vierstufiges Verarbeitungsmodell entwickelt, nach dem Sensordaten zuerst mit Filtern aufbereitet, dann verglichen und anschließend zu Gruppenkonfigurationen umgerechnet werden. Die vierte Stufe fasst mehrere Gruppenkonfigurationen zu einer Gesamtlösung zusammen. Ein weiteres Verfahren stammt aus der Biologie, wo es zur Gruppierung von Gen- oder DNA-Analysen benutzt wird. Um diese Verfahren an realen Sensordaten zu verifizieren, wurden in einer Büroumgebung Messungen durchgeführt und aufgezeichnet. Auf Basis der gesammelten Daten konnte das vierstufige Verarbeitungsmodell entwickelt werden. Dabei stellte sich heraus, dass sich die Luftfeuchtigkeit und Temperatur durch Bildung der Differenz zwischen den Sensordaten von Motes vergleichen lassen. Um signifikante Änderungen in den Messwerten als Ereignisse zu erkennen und vergleichen zu können, müssen die Sensordaten von Luftfeuchtigkeit und Temperatur von Rauschen befreit werden. Bei den Lichtbereichen PAR und TSR hingegen spielt die unterschiedlich intensive Einstrahlung von Sonnenlicht an verschiedenen Aufstellungsorten von Motes eine große Rolle. Dies verhindert einen direkten Vergleich. Auch bei der Ereigniserkennung müssen Änderungen prozentual ausgedrückt werden. Um die Signalstärke zur Gruppeneinteilung verwenden zu können, werden sie invers zur Sendeleistung gewichtet. Die gefundenen Vergleichsdaten werden aus Gründen der Fehlertoleranz durch Aufsummierung mit einem Erinnerungseffekt versehen. Für die Erstellung einer Gesamtlösung existieren zwei Kombinationsmöglichkeiten, die Verwandtschaftsbeziehungen zu verarbeiten. Die erste nimmt eine vorläufige Gruppeneinteilung der einzelnen Kanäle durch Mittelwertbildung vor, kombiniert mit einer Suche nach den Motes mit maximaler Übereinstimmung und einer auf der Wahl aller Motes basierenden Bildung der transitiven Hülle. Dies führte zu einer perfekten Gesamtlösung. Die zweite Möglichkeit homogenisiert die Verwandtschaftsbeziehungen lediglich und erstellt durch Mittelwertbildung die Gesamtlösung. Diese Lösung wies vernachlässigbar wenige Fehler auf. Beide Gesamtlösungen stellten sich im direkten Vergleich mit dem Verfahren aus der Biologie als weit überlegen heraus. Die gefundenen Resultate konnten an öffentlich zugänglichen Datensammlungen des Berkeley-Forschungslabor von Intel mit kleinen Einschränkungen nachvollzogen werden. Als Fazit der Untersuchungen kann gesagt werden, dass eine Gruppierung von Sensorknoten auf Basis von Sensordaten und Konnektivität automatisiert durchführbar ist. Zu diesem Zweck steht auch eine Implementierung für Motes zur Verfügung.

Literaturverzeichnis

[BC03] S. Bandyopadhyay and E. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom 2003)*, 2003.

[BHE00] Nirupama Bulusu, John Heidemann, and Deborah Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28–34, October 2000.

[BLP05] Michael Broxton, Joshua Lifton, and Joseph Paradiso. Localizing a Sensor Network via Collaborative Processing of Global Stimuli. In Erdal Cayirci, Sebnem Baydere, and Paul Havinga, editors, *Proceedings of the Second European Workshop on Wireless Sensor Networks*, pages 321–332. IEEE, January 2005.

[BP00] P. Bahl and V. N. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775–784 vol.2, 2000.

[BRT05] J. Blumenthal, F. Reichenbach, and D. Timmermann. Position estimation in ad hoc wireless sensor networks with low complexity. In *Joint 2nd Workshop on Positioning, Navigation and Communication 2005 (WPNC 05) & 1st Ultra-Wideband Expert Talk 2005*, pages S.41–49, March 2005.

[CLP05] Haowen Chan, Mark Luk, and Adrian Perrig. Using clustering information for sensor network localization. In *The International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 109–125, 2005.

[Con71] W. J. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, New York, December 1971.

[Cro] *Crossbow Technology, Inc*. URL: www.xbow.com.

[ESBB98] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 95, pages 14863–14868. Proceedings of the National Academy of Sciences of the United States of America, December 1998.

[FKB+05] Sándor P. Fekete, Alexander Kröller, Carsten Buschmann, Stefan Fischer, and Dennis Pfisterer. Koordinatenfreies Lokationsbewusstsein (Localization without Coordinates). *it - Information Technology*, 47(2):70–78, 2005.

[GKW+02] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. *UCLA Computer Science Technical Report UCLA/CSD-TR 02-0013*, 2002.

[GMS05] Saikat Guha, Rohan Murty, and Emin Gün Sirer. Sextant: A Unified Node and Event Localization Framework Using Non-Convex Constraints. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 205–216, New York, NY, USA, May 2005. ACM Press.

[HCB00] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd International Conference on System Sciences (HICSS)*, January 2000.

[HHB+03] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, and Tarek F. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 81–95, New York, NY, USA, September 2003. ACM Press.

[HINM04] M.J.L. De Hoon, S. Imoto, J. Nolan, and S. Miyano. Open source clustering software. *Bioinformatics*, 20(9):1453–1454, 2004.

[Int] *Intel Berkeley Research Lab*. URL: http://berkeley.intel-research.net/labdata/ visited 10.10.2006.

[KLJT05] Kai S. Kunze, Paul Lukowicz, Holger Junker, and Gerhard Tröster. Where am I: Recognizing On-body Positions of Wearable Sensors. In *International Workshop on Location- and Context-Awareness*, pages 264–275, May 2005.

[LHB04] Jonathan Lester, Blake Hannaford, and Gaetano Borriello. "Are You with Me?" - Using Accelerometers to Determine If Two Devices Are Carried by the Same Person. In Alois Ferscha and Friedemann Mattern, editors, *Pervasive Computing, Second International Conference, PERVASIVE 2004, Vienna, Austria, April 21-23, 2004, Proceedings*, volume 3001 of *Lecture Notes in Computer Science*, pages 33–50. Springer, 2004.

[MA01] Arati Manjeshwar and Dharma P. Agrawal. Teen: A routing protocol for enhanced efficiency in wireless sensor networks. In *IPDPS '01: Proceedings of the 15th International Parallel & Distributed Processing Symposium*, page 189, Washington, DC, USA, 2001. IEEE Computer Society.

[MHH03] Samuel R. Madden, Joseph M. Hellerstein, and Wei Hong. TinyDB: In-network query processing in TinyOS. http://telegraph.cs.berkeley.edu/tinydb, September 2003.

[MKSÁL04] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 39–49, New York, NY, USA, 2004. ACM Press.

[Mot] *Moteiv Corporation*. URL: www.moteiv.com.

[NOY05] Tomohiro Nagata, Hisashi Oguma, and Kenichi Yamazaki. A sensor networking middleware for clustering similar things. In *Proceedings of the Workshop on Smart Object Systems, held in conjunction with UbiComp 2005*, 2005.

[SCSS05] Alexander Schliep, Ivan G. Costa, Christine Steinhoff, and Alexander Schonhuth. Analyzing gene expression time-courses. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 2(3):179–193, 2005.

[SHSL05] Radu Stoleru, Tian He, John A. Stankovic, and David Luebke. A high-accuracy, low-cost localization system for wireless sensor networks. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 13–26, New York, NY, USA, 2005. ACM Press.

[Tin] *TinyOS Operating System*. URL: www.tinyos.net.

[War62] Stephen Warshall. A theorem on boolean matrices. *J. ACM*, 9(1):11–12, 1962.

[Wel67] P. D. Welch. The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging Over Short, Modified Periodograms. *IEEE Trans. Audio & Electroacoust., Volume AU-15, p. 70-73*, 15:70–73, 1967.

[Wik] *Wikipedia.de: Korrelationskoeffizient*. URL: de.wikipedia.org visited 10.10.2006.

[YF04a] Ossama Younis and Sonia Fahmy. Distributed clustering in ad-hoc sensor networks: A hybrid energy-efficient approach. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOMM)*, Hong Kong, March 2004.

[YF04b] Ossama Younis and Sonia Fahmy. HEED: A hybrid, energy-efficient, distributed clustering approach for ad-hoc sensor networks. *IEEE Transactions on Mobile Computing*, 3(4):366–379, Oktober 2004.

[YFS04] Ossama Younis, Sonia Fahmy, and Paolo Santi. Robust communications for sensor networks in hostile environments. In *Proc. 12th IEEE Workshop on Quality of Service (IWQoS)*, pages 10–19. IEEE, June 2004.

Erklärung

Hiermit versichere ich, diese Arbeit selbstständig verfasst und nur die angegebenen Quellen benutzt zu haben.

Jettingen, den 18. Oktober 2006

Andreas Heydlauff