# Umweltmonitoring mit SensorScope

Stefan Stöckl
Betreuerin: Corinna Schmitt
Seminar: Sensorknoten - Betrieb, Netze & Anwendungen SS 2012
Lehrstuhl Netzarchitekturen und Netzdienste, Lehrstuhl Betriebssysteme und Systemarchitekturen
Fakultät für Informatik, Technische Universität München
Email: stefan-stoeckl@mytum.de

## KURZFASSUNG

Gerade in Zeiten der Klimaerwärmung versuchen immer mehr Wissenschaftler mit Hilfe von technischen Hilfsmitteln Umweltphänomene zu erklären. Dazu werden Messwerte an verschiedenen Orten und über einen längeren Zeitraum eingeholt. Dieses vorgehen bezeichnet man als Umweltmonitoring. Mit den gesammelten Daten wird dann anhand von Berechnungen versucht, Rückschlüsse auf die untersuchten Umweltphänomene zu ziehen. Ein System das zum Umweltmonitoring eingesetzt wird ist SensorScope. Es dient der Untersuchung von Klimabedingungen mit deren Hilfe Ereignisse, wie beispielsweise Schlammlawinen, vorhergesagt werden können. Seine Funktionsweise wird in dieser Arbeit ausführlich beschrieben. Anschließend werden noch zwei weitere Systeme kurz erklärt und es wird ein Vergleich hergestellt.

## Schlüsselworte

Umweltmonitoring, Umweltbeobachtung, Environmental Monitoring, SensorScope, Wireless Sensor Networks (WSN)

## 1. EINLEITUNG

Unter Umweltmonitoring versteht man die Erhebung und Auswertung von Daten über Umweltbedingungen. Zuerst werden hierbei Daten in Form von Messwerten erhoben und gesammelt. Anschließend wird versucht mit wissenschaftlichen Verfahren einen Zusammenhang zwischen Messwerten und Umweltphänomenen herzustellen, damit man sich diese erklären und vorhersagen kann. Die erhobenen Daten kommen vor allem aus den Bereichen Chemie, Physik und Biologie [8]. Bei der Auswertung der Daten kommen vor Allem Verfahren aus der Mathematik, insbesondere aus der Statistik, zum Einsatz. Hierbei versucht man herauszufinden, welche Umweltfaktoren einen Einfluss auf die untersuchten Phänomene haben. Typische Anwendungsgebiete von Umweltmonitoring sind die Beobachtung von Klima, Luftqualität, Wasserqualität, Natur-Phänomenen (wie z.B. Vulkane), Bakterien und Radioaktiver Strahlung.

Der Fokus dieser Arbeit liegt auf der Verwendung von SensorScope, einem Sensornetz (Wireless Sensor Network, WSN) das zur Erfassung von Daten im Bereich der Klimabeobachtung eingesetzt wird. Der Einsatz des WSNs bringt viele Vorteile im Vergleich zu herkömmlichen Verfahren. Zu erwähnen wäre hier, dass die Messwerte nicht manuell eingeholt werden müssen, sondern automatisch erfasst und an eine zentrale Datenbank gesendet werden, in der sie dann ausgewertet werden. Weiterhin ist es durch verteilte Knoten möglich, im Vergleich zu einer zentralen Messstation, eine gute räumliche Abdeckung des beobachteten Areals zu erreichen. Außerdem können durch den Verzicht auf teure Messstationen noch Kosten gespart werden.

SensorScope kommt hauptsächlich in Gebieten mit schwierigen Umweltbedingungen, wie zum Beispiel auf einem Gletscher, zum Einsatz. Daraus ergaben sich für die Entwickler des Systems besondere Herausforderungen. So muss ein reibungsloser Einsatz ohne Anschluss an ein Stromnetz sichergestellt werden. Außerdem muss gewährleistet werden, dass der Sensor die Umwelteinflüsse, wie Feuchtigkeit und extreme Temperaturen, ohne Schaden übersteht und weiterhin fehlerfrei funktioniert.

In Kapitel 2 wird SensorScope genau beschrieben. Hier wird in Abschnitt 2.1 der Aufbau einer Sensorstation, in Abschnitt 2.2 die Netzwerkfunktionalität und in Abschnitt 2.3 Anwendungen von SensorScope genau erläutert. Anschließend werden in Kapitel 3 weitere Umweltmonitoring-Systeme wie Macroscope (Abschnitt 3.1) und PermaSense (Abschnitt 3.2) kurz erklärt. Zuletzt werden diese Systeme noch in Kapitel 4 miteinander verglichen und Unterschiede und Gemeinsamkeiten hervorgehoben.

## 2. SENSORSCOPE

In diesem Kapitel wird SensorScope, ein WSN, das zum Umweltmonitoring verwendet wird, beschrieben. Es basiert größtenteils auf [7].

Sensorscope ist ein WSN, das für Umweltmonitoring verwendet wird. Es wurde in Zusammenarbeit von Umweltforschern, Hardware- und Software-Entwicklern der EPFL (École polytechnique fédérale de Lausanne; dt.: Eidgenössische Technische Hochschule Lausanne) entwickelt [3].

Abschnitt 2.1 geht auf die einzelnen Sensorstationen ein. Anschließend beschreibt Abschnitt 2.2 wie diese Stationen zu einem Netzwerk zusammengefügt werden und miteinander kommunizieren. Hierbei sollen auch mögliche Schwierigkeiten aufgezeigt werden. In Abschnitt 2.3 werden dann Anwendungen, die mit Hilfe von SensorScope umgesetzt wurden, erläutert und deren Ergebnisse präsentiert.

### 2.1 Sensorstation

Eine SensorScope-Messstation besteht aus einer Solarzelle, Sensoren und einer hermetisch abgeriegelten Box. In dieser Box befindet sich die Batterien und das Herzstück, der Sensorknoten. All dies ist auf eine 2 Meter hohe Stange montiert, da durch zu geringen Abstand zum Boden die Klimabedingungen verfälscht werden. Abbildung 1 zeigt eine Sensorstation. Solch eine Station kostet rund €1500.

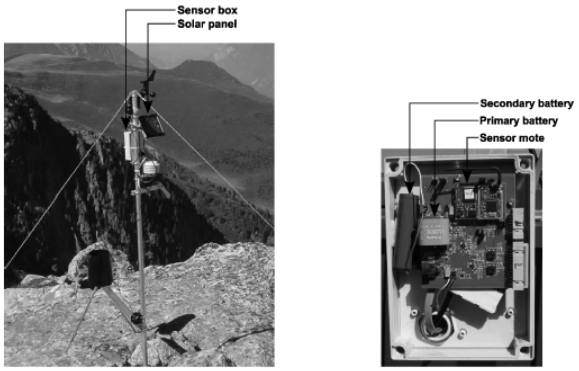


**Abbildung 1: Eine SensorScope Station (links) und die Box die den Microcontroller und die Batterien enthält (rechts) [7]**

### *2.1.1 Stromversorgung*

Die Stromversorgung von SensorScope besteht aus drei Komponenten und basiert auf [9].
**Solarzelle**: Zur Energiegewinnung wird eine MSX-01F Solarzelle von BP Solar verwendet. Laut Datenblatt ist sie 162x140 mm groß, hat eine zu erwartende Haltbarkeit von 20 Jahren und hat bei direkter Sonneneinstrahlung eine Ausgangsspannung von mindestens 1W [5].
**Primäre Batterie**: Als primäre Batterie wird, eine wiederaufladbare 150mAh NiMH Batterie verwendet, welche gegenüber dem vorgeschlagenen Superkondensators den Vorteil hat, dass sie billiger ist und auch noch mehr Energie speichern kann. Sie hält bis zu 5 Tage und befindet sich in der hermetisch abgeriegelten Box. Sie wird mit Hilfe der Solarzelle wiederaufgeladen und dient als Hauptstromquelle für den Sensorknoten.
**Sekundäre Batterie**: Als zweite Batterie wird eine LiIo-Baterie mit einer Kapazität von 2200mAh verwendet. Sie dient als Backup, falls die erste Batterie leer wird.
Mit Hilfe dieser drei Komponenten kann eine autonome Stromversorgung sichergestellt werden. In Abbildung 1 sieht man, wie die beiden Batterien in der Sensorstation untergebracht sind.
In zukünftigen Generationen von SensorScope wird vermutlich nur noch eine Batterie verbaut sein [7]. Diese Überlegung wird in Abschnitt 2.3.2 genauer beschrieben.

### *2.1.2 Sensoren*

Jeder Knoten verfügen über sieben Sensoren, mit denen es möglich ist, bis zu neun umweltbezogene Größen zu messen. Tabelle 1 fasst die hardwarespezifischen Informationen zu den Sensoren zuammen. Durch eine durchgeführten Erprobungen wurde jedoch festgestellt, dass es sehr nützlich wäre, noch eine Kamera zur Verfügung zu haben, um Bilder der Umgebung aufnehmen zu können (siehe Abschnitt 2.3.2).

**Tabelle 1: Von SensorScope verwendete Sensoren [7]**

| Sensor | Messgröße | Range | Präzision |
|---|---|---|---|
| Sensirion SHT75 | Luftfeuchtigkeit | 0-100% | ± 2% |
| Sensirion SHT75 | Lufttemperatur | -20-60°C | ±0,3°C |
| Davis Rain Collector | Niederschlagsintensität | 0-∞mm | ± 1 mm |
| Decagon EC-5 | Bodenfeuchtigkeit | 0-100% | ±0,1% |
| Davis Solar Radiation | Sonneneinstrahlung | 0 - 1800 W/m$^2$ | ± 90 W/m$^2$ |
| Zytemp TN901 | Oberflächentemperatur | -33-220°C | ± 0,6°C |
| Irometer Watermark | Bodenfeuchte | -200-0kPa | unbekannt |
| Davis Anemomenter | Windrichtung | 0-360° | ± 7° |
| Davis Anemomenter | Windgeschwindigkeit | 1,5-79m/s | ±1,5m/s |

### *2.1.3 Kerneinheit*

Als Kern von SensorScope wird eine TinyNode-Einheit verwendet. Sie besteht aus einem TI MSP430 Microcontroller und einer Xemics XE1205 Funkeinheit [1]. Es handelt sich um einen 16-Bit Microcontroller der aufgrund seindes geringen Energieverbrauchs für batteriebetriebene Geräte ausgelegt ist. Weitere Informationen kann man dem Datenblatt unter [12] entnehmen. Die Kerneinheit ist in Abbildung 1 zu sehen.

## 2.2 Netzwerk

In diesem Abschnitt wird die Netzwerkfunktionalität von SensorScope beschrieben. Abbildung 2 zeigt den Typischen Aufbau eines SensorScope Netzwerks. Die Sensorknoten sind über Funk miteinander verbunden. Einer von ihnen verfügt über GPRS, so dass er mit einem Datenbank-Server kommunizieren kann um diesem Daten aus dem Netzwerk zu senden. Dieser wiederum ist mit einem Webserver verbunden, der die Daten aufbereitet und den Benutzern zur Verfügung stellt.

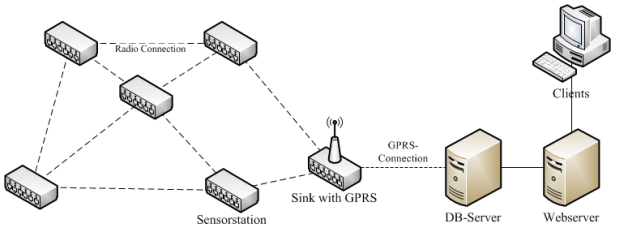


**Abbildung 2: Typischer Aufbau eines SensorScope Netzwerks**

Im weiteren Verlauf wird erklärt, wie das Netzwerkprotokoll aufgebaut ist, welche Herausforderungen beim Design des Netzwerkes es gab und wie diese gelöst wurden. Im Einzelnen wird Nachbarschaftsmanagement (2.2.2), Synchronisation
(2.2.3), Energiemanagement (2.2.4) und Routing (2.2.5) genauer erläutert.

### *2.2.1 Protokollimplementierung*

Bei der Entwicklung der Netzwerkarchitektur von SensorScope wurde darauf geachtet, dass alles möglichst einfach gestaltet wird, um das ganze robust zu gestalten. Hierbei wurde ein eigener Protokoll-Stack entwickelt, der in eine TinyOS-Nachricht eingebettet ist (vgl. [4]). Dass der Sensor-Scope-Header in den Payload der TinyOS-Nachricht und nicht in deren Header geschrieben wird hat den Vorteil, dass man an SensorScope nichts ändern muss, falls sich in neuen TinyOS-Distributionen etwas ändert.

Der Payload der TinyOS-Nachrichten beträgt 28 Bytes. Davon werden 4 Bytes als Header für Sensor-Scope und 24 Bytes als Daten genutzt. Der Header besteht aus folgenden vier Feldern (je 1 Byte)[4]:
Die **Sender ID** gibt an, von wem eine Nachricht ursprünglich gesendet wurde. In dem Feld **Cost to Sink** (dt. Kosten


doi: 10.2313/NET-2012-08-2_08

zur Senke) steht die Hop-Distanz zur Senke.
Der **Hop-Count** wird von der Transportschicht geschrieben. Für neue Pakete wird er auf 0 gesetzt, für Pakete die weitergeleitet werden wird er um 1 erhöht. Der Hop-Count ist nicht obligatorisch und wird nur für statistische Auswertungen benutzt.
Die **Sequenznummer** wird ebenfalls von der Transportschicht geschrieben. Sie wird jedes mal erhöht, wenn das Senden eines Pakets erfolgreich war. Anderenfalls wird das Paket nochmal mit der gleichen Sequenznummer geschickt. Dies dient der Auswertung der Link-Qualität.
In Abbildung 3 wird der Netzwerkstack von SensorScope gezeigt. Er besteht aus vier Schichten, die im Folgenden kurz charakterisiert werden.

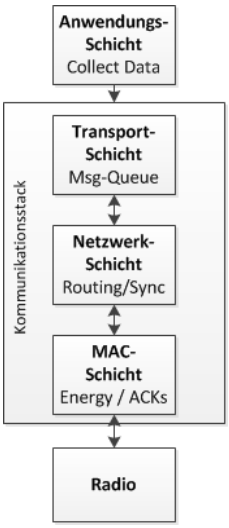


**Abbildung 3: Netzwerkstack von SensorScope**

Die **Anwendungsschicht** ist für das Sammeln von Daten zuständig. Sie fragt sowohl Sensordaten, als auch den Ladestand der Batterien ab.
Die **Transportschicht** abstrahiert den Rest des Kommunikationsstacks für die Anwendungsschicht und lässt dem Benutzer nur zwei Möglichkeiten, welche Befehle er versenden kann:
**Datenpakete** enthalten Daten, die zur Senke des Netzwerks geschickt werden sollen. Die Daten die gesendet werden sind gemessene Sensorwerte und Batteriestände.
**Kontrollpakete** richten sich an einen oder alle Nachbarn des Knotens. Sie enthalten Beacons (siehe Abschnitt 2.2.2) oder Synchronisationspakete (siehe Abschnitt 2.2.3). Die Synchronisation erfolgt nur lokal. Dies hat den Vorteil, dass das Netzwerk dezentralisiert bleibt.
In der Transportschicht werden die vom Benutzer übergenen Daten in Pakete gepackt und in eine Warteschlange (engl. Queue) eingereiht. In diese Warteschlange kommen auch empfangene Nachrichten, die weitergeleitet werden müssen. Kontrollpakete erhalten in der Warteschlange höhere Priorität gegenüber Datenpaketen. Das wird gemacht, da sie höhere zeitliche Anforderungen haben. Ausserdem werden hier die beiden Felder Hop-Count und Sequenznummer des Pakets geschrieben. Eine Überlastungskontrolle (engl. Congestion Control) ist aufgrund des geringen Datenaufkommens nicht nötig.
Die **Netzwerkschicht** ist für das Routing der Pakete zuständig. Sie routet Datenpakete in Richtung Senke. Kontrollpakete werden nicht weitergeleitet, da es nur lokale Broadcasts zu den direkten Nachbarn gibt und keine globalen. Wie genau bei SensorScope der nächste Hop ausgewählt wird, kann Abschnitt 2.2.5 entnommen werden. Die Netzwerkschicht schreibt die anderen beiden Felder des Paket-Headers: Sender ID und Kosten zur Senke. Wenn man einen neuen Routing-Algorithmus einführen will reicht es aufgund der Layer-Architektur, die Netzwerkschicht neu zu implementieren. Die übrigen Schichten können unverändert bleiben.
Die **MAC-Schicht** steuert den Funk und hat folgende Funktionalitäten: Ein- und Ausschalten des Funks, Nachrichten senden und empfangen und Zurücksenden eines Acknowledgements beim Empfang einer Datennachicht.
Wie bei Ethernet wurde ein Backoff-Mechanismus implementiert. Hierbei wird nach jedem Senden, das vom Empfänger nicht mit einem ACK bestätigt wurde, eine zufällige Zeit zwischen 0 und Max-Delay gewartet, bevor das Paket erneut gesendet wird. Bei jedem weiteren Senden, bei dem ein ACK ausbleibt, wird das Max-Delay exponentiell erhöht. Nachdem die Nachricht erfolgreich gesendet wurde wird das Max-Delay wieder zurückgesetzt. Der Einfluss dieses Verfahrens auf den Energieverbrauch der Sensor-Knoten wird in Abschnitt 2.2.4 genauer erläutert.
Der eigentliche Kommunikationsstack besteht aus der Transport-, Netzwerk- und MAC-Schicht.

### *2.2.2 Nachbarschaftsmanagement*

Jeder Sensorknoten pflegt eine eigene, lokale Nachbarschaftstabelle. Sie enthält einen Eintrag für jeden Knoten, von dem Nachrichten direkt empfangen werden können. Zusätzlich enthält die Tabelle auch noch für jeden Nachbarn dessen Entfernung zur Senke (Cost to Sink), seine Verbindungsqualität (Quality of Service, QoS) und ein Timestamp des letzten empfangenen Pakets das von dem Nachbarn gesendet wurde. Diese Informationen werden in der Netzwerkschicht durch Mithören und Auswerten der empfangenen Pakete erstellt.
**Timestamp des letzten empfangenen Pakets**: Dieser wird verwendet um Knoten, von denen man schon lange keine Pakete mehr empfangen hat (sog. Dead Neighbors) aus der Tabelle zu löschen. Dadurch wird sichergestellt, dass man keine Nachrichten mehr an Knoten schickt, die z.B. durch Hardware-Fehler, schon lange ausgefallen sind. In den in Abschnitt 2.3.2 bschriebenem Deployment wurde das Zeitinterval, nach dem inaktive Nachbarn gelöscht werden, auf 480 Sekunden festgelegt.
**Cost to Sink**: Als einfache Kosten-Metrik wird bei SensorScope die Hop-Distanz eines Knotens zur Senke gewählt. Am Anfang kennt außschließlich die Senke ihren eigenen Wert. Dieser ist nämlich per Definition null. Durch das Senden sogenannter Beacons an ihre direkten Nachbarn initiert sie die Berechnung der Kosten für alle Knoten des Netzwerks. Durch das Empfangen eines Beacons wissen die Nachbarn, dass Sie nur einen Hop von der Senke entfernt sind und beginnen mit dem Senden von Daten an die Senke. Da die Costs to Sink, wie in Abschnitt 2.2.1 beschrieben, im Header der gesendeten Daten-Pakete stehen, sind keine weiteren Beacons mehr nötig. Alle Nachbarn, die jetzt ein Datenpa-

ket mit Kosten 1 mithören wissen nun ebenfalls ihre Kosten (nämlich 2) und beginnen ebenfalls mit dem Senden von Daten-Paketen. So geht es weiter, bis alle Knoten ihre Hop-Distanz zur Senke kennen. Da Knoten normalerweise mehrere Nachbarn haben nehmen sie für sich immer den Wert der Kosten ihres kostengünstigsten Nachbarn und zählen 1 dazu. Zusammen mit der Service-Qualität (Quality of Service, QoS) sind die Costs to Sink ausschlaggebend für Routing-Entscheidungen (siehe Abschnitt 2.2.5).

**Service-Qualität**: Die QoS ist ein Maß sowohl für die Übertragungsqualität zu einem Nachbarn, als auch für die Fähigkeit des Nachbarn, Nachrichten weiterzuleiten. Zum berechnen dieser Metrik werden die Sequenznummern der letzten 16 von einem Nachbarn empfangenen Pakete betrachtet. Im Idealfall sind diese Sequenznummern 16 aufeinanderfolgende Zahlen. Wenn die Übertragungsqualität zwischen dem Knoten und dem Nachbarn aber schlecht ist wird es vorkommen, dass der Knoten nicht alle Pakete des Nachbarn empfangen kann und somit Sequenznummern fehlen. Wenn ein Nachbar seinerseits wieder schlechte Nachbarn hat und darum beim Senden von Nachrichten kein ACK zurückbekommt sendet er ein Paket erneut (Siehe Abschnitt 2.2.5) mit der gleichen Sequenznummer. Darum kann man aus dem Empfangen doppelter Seuquenznummern auf die schlechte Fähigkeit zum Weiterleiten von Paketen des Nachbarn schließen. Aufgrund dieser beiden Feststellungen wird die QoS mit Hilfe der Formel $QoS = \frac{16}{16+x+y}$ berechnet, wobei x für die Anzahl der fehlenden und y für die Anzahl der doppelten Sequenznummern steht [7]. Ein idealer Nachbar hat einen QoS-Wert von 1.

Anstatt über die Sequenznummern wurde auch noch in Betracht gezogen, die Übertragungsqualität mit Hilfe von RSSI (Received Signal Strength Indication) zu bestimmen. Jedoch wurde dieses Verfahren als zu ungenau bewertet. Außerdem ist es, wie man aus [2] entnimmt, viel komplexer als das simple Auswerten der Sequenznummern.

Es gibt aber auch noch Verbesserungspotential um Fehler zu vermeiden: Da alle Nachbarn, die ein Knoten hören kann, in dessen Nachbarschaftstabelle eingetragen werden, kann es auch vorkommen, dass Nachbarn, die man zwar hört, denen man aber keine Nachrichten schicken kann (asymmetrische Nachbarn) in die Nachbarschaftstabelle eingetragen werden. Wenn eine Nachricht an solche Nachbarn gesendet wird erhält man von ihnen kein ACK und die Nachricht wird erneut gesendet.

In regelmäßigen Abständen senden die Sensorknoten eine Liste ihrer Nachbarn und deren QoS zum Server. Dies ermöglicht es, Rückschlüsse über die Netzwerktopologie zu gewinnen und mögliche Schwachstellen im Netzwerk aufzuspüren [4].

### 2.2.3 Synchronisation

Da die Zeit, die es dauert, bis gesendete Nachrichten zum Server gelangen nicht genau berechnet werden kann, ist es nötig, dass Messwerte in Datenpaketen mit Zeitstempeln versehen werden. Außerdem ist für das in Abschnitt 2.2.4 beschriebene synchrone Duty Cycling notwendig, dass die Knoten im Netzwerk untereinander die gleiche Zeitbasis haben. Aus diesen Gründen ist es nötig, dass sich die Knoten untereinander und mit dem Server synchronisieren, was in den folgenden Abschnitten beschrieben wird.

#### 2.2.3.1 Synchronisation zwischen den Stationen.

Die Kristalle der in SensorScope verwendeten TinyNodes haben einen theoretischen Uhrenfehler (Clock Drift) von $\pm 20$ ppm [4], was 72 Milisekunden pro Stunde entspricht. Die Versuche aus [7] haben aber gezeigt, dass der Drift deutlich höher sein kann und auch noch von der Umgebungstemperatur abhängig ist. Dabei war der Fehler bei Zimmertemperatur bei etwa 120 ms/h und in einem Gefrierschrank sogar bei bis zu 375 ms/h. Um diese Driftraten zu kompensieren ist die Synchronisation notwendig.

Bei der Synchronisation zwischen einzelnen Sensorknoten dient die Zeit der Senke als Referenz-Zeit und es wird davon ausgegangen, dass Knoten die näher an der Senke sind (geringere Hop Distanz), eine genauere Zeit haben als welche, die weiter entfernt sind. Der Ablauf bei der Synchronisation ist in Abbildung 4 skizziert und wird im Folgenden erklärt:

Knoten **g** will die Zeit wissen und sendet einen *Sync Request* zu einem seiner Nachbarn, der näher an der Senke ist als er (**d**), da angenommen wird, dass dieser eine genauere Zeit hat. Dieser sendet einen *Sync Reply* mit seiner aktuellen Zeit als lokalen Broadcast an alle seine Nachbarn. Jeder Knoten, der den Reply empfängt setzt seine Zeit auf die aus dem Reply, falls er weiter von der Senke entfernt ist als der Sender des Reply. In der Abbildung ist dies bei den Knoten **b** und **g** der Fall. Die übrigen Empfänger (**c**, **f** und **h**) ignorieren den Reply, da sie näher oder gleich weit von der Senke entfernt sind wie **d**. Das senden eines lokalen Broadcasts hat den Vorteil, dass weniger *Sync Requests* gesendet werden, da alle Empfänger des Replys ihren Nächsten Request, je nach Synchronisationsmodus, aufschieben.

Es gibt zwei verschiedene Synchronisationsmodi. Der erste ist der *High-Frequency Mode*. Dieser wird verwendet, wenn ein Knoten die aktuelle Zeit nicht kennt, was beispielsweise der Fall ist, wenn er neu bootet. Der andere, der *Low-Frequency-Mode* ist dafür da, um die Driftrate der Kristalle der Sensorknoten zu kompensieren. Der Mode ist entscheidend dafür, wie oft sich ein Knoten mit dem Netzwrk Synchronisiert. Für die in dieser Arbeit beschriebenen Outdoor-Experiment (siehe Abschnitt 2.3.2) wurde ein Interval von 5 Sekunden für den *High-* und 1 Stunde für den *Low-Frequenzy Modus*, zwischen den Synchronisationen eines Knotens, verwendet.

#### 2.2.3.2 Synchronisation zwischen Senke und Server.

Da der Server wissen muss, zu welcher Zeit die Messungen der Daten, die in einer Nachricht stehen, durchgeführt wurden, muss er den Offset zwischen der Zeitbasis des Sensor-Knoten-Netzwerks und seiner Zeit (tatsächliche Zeit) kennen. Als Zeitbasis für das Netzwerk wird die lokale Zeit der Senke verwendet. Damit der Server den Offset zu dieser Zeit berechnen kann wird die lokale Zeit in regelmäßigen Abständen an den Server geschickt. Wenn die Senke neu gestartet wird versucht sie zuerst sich mit ihren Nachbarn zu synchronisieren (im *High-Frequency Mode*). Schlägt dies fehl geht sie davon aus, dass das Netzwerk neu gestartet wurde und ihre lokale Zeit wird als Basis für das Netzwerk verwendet.

Dieser Offset wäre nicht nötig, wenn sich Netzwerk und Server ebenfalls synchronisieren würden. Aufgrund der schlechten Erreichbarkeit der Senke über GPRS hat sich dies aber als zu schwierig herausgestellt [4].

In Zukunft ist geplant, die tatsächliche Zeit als Basis für

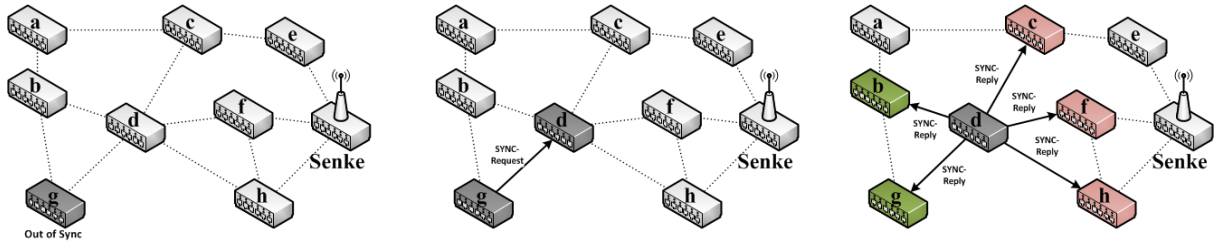


**Abbildung 4: Ablauf bei der Synchronisation**

das Netzwerk zu verwenden. Diese könnte man beispielsweise durch die Verwendung eines GPS Chips beziehen.

### 2.2.4 *Energiemanagement*

Ein Großteil des Energieverbrauchs des TinyNodes wird durch den Funk verbraucht. Bei abgeschalteten Funk verbraucht er gerade einmal 2 mA. Wenn man ihn anschaltet braucht er schon 15 mA und beim Senden verbraucht er je nach Stärke 25 - 58 mA (bei 0dBm bzw. 15 dBm). Darum ist es aufgrund der begrenzten Menge der verfügbaren Energie (siehe Abschnitt 2.1.1) in der Sensorstation nicht möglich, den Funk ununterbrochen laufen zu lassen. Eine Lösung hierfür bietet *Duty-Cycling*. Hierbei ist es entscheidend, dass alle Stationen ihren Funk gleichzeitig einschalten, um miteinander zu kommunizieren. Nach so einer Sende-Phase machen die Knoten dann eine Pause und schalten dabei ihren Funk ab, bis nach einem festen Intervall die nächste Sendephase eintritt. Für die in Abschnitt 2.3 beschriebenen Deployments wurden für die Sende-Phase 12 und für die Pause-Phase 108 Sekunden gewählt. Da sich die Stationen nur einmal in der Stunde synchronisieren laufen ihre Uhren in der Regel nicht exakt synchron. Darum passiert es, dass sie sich nicht zum genau gleichen Zeitpunkt einschalten. Darum warten sie am Anfang einer Sende-Phase 500 ms, bevor sie mit dem Senden von Nachrichten beginnen. Durch diese Wartezeit kann, auch unter Einbeziehung des Clock Drifts, sichergestellt werden, dass auch die Knoten, die etwas hinter der Referenz-Zeit sind, ihren Funk aktiviert haben.

### 2.2.5 *Routing*

Das Routing bei Sensorscope beruht auf Zufall. Wenn ein Knoten ein Datenpaket zur Senke senden will, wählt er zufällig einen geeigneten Nachbarn aus, an den er das Paket weiterleitet. Welche Knoten als geeignet angesehen werden richtet sich nach zwei Kriterien:

- Die Costs to Sink des Empfänger-Knotens müssen kleiner sein als die des Senders. So wird sichergestellt, dass das Paket der Senke mit jedem Hop näher kommt.
- Die Quality of Service des Empfängers muss gut sein. Um dies zu bestimmen werden zwei Thresholds festgelegt. Nachbarn, deren QoS über dem oberen Threshold liegt, werden als High Quality Neighbors bezeichnet. Falls es solche gibt, werden nur diese als geeignet angesehen. Solche, die zwischen den beiden Thresholds liegen, werden als Low Quality Neighbors bezeichnet. Diese werden als geeignet betrachtet, wenn es keine High Quality Neighbors gibt. Die restlichen, also diejenigen, die unter dem unteren Threshold liegen, werden gar nicht beachtet.

Eine weitere Möglichkeit wäre ein fester Backbone, der jeden Knoten mit der Senke verbindet. Dies hätte jedoch Nachteile gegenüber dem gewählten Verfahren. Einerseits wäre ein Overhead nötig, um Unterbrechungen (z.B. durch defekte Knoten) festzustellen. Zum anderen bestünde die Gefahr, dass es Knoten gibt, durch die sehr viele Routen gehen und es so zu einem Flaschenhals (engl. Bottleneck) kommen würde.
Ein Nachteil des verwendeten Verfahrens ist, dass das Netzwerk nur lokal betrachtet wird und so nicht zwingend die beste Route durch das gesamte Netzwerk genommen wird. Eine einfache Verbesserungsmöglichkeit wäre es, verstärkt an Knoten mit wenigen Nachbarn zu senden.
Genau wie bei der Synchronisation wurde auch beim Routing darauf geachtet, dass ein möglichst einfaches Verfahren ausgewählt wurde, um Fehler, die durch zu hohe Komplexität entstehen könnten, zu vermeiden.

## 2.3 Anwendungen und Ergebnisse

In diesem Abschnitt werden die bisherigen Einsätze von SensorScope erläutert und deren Ergebnisse aufgezeigt.

### 2.3.1 *Indoor Erprobung*

Das erste Experiment mit SensorScope führten die Entwickler in einem Bürogebäude der EPFL durch unter Verwendung von 17 Sensorstationen. Dabei ging es darum, das implementierte Netzwerkprotokoll zu testen, weshalb an die TinyNodes der Sensorstationen auch keine Sensoren angeschlossen wurden. Jeder Knoten sendete 5054 Nachrichten. Von allen bis auf einen Knoten kamen alle Nachrichten bei der Senke an. Von dem einen Knoten gingen etwa 200 verloren. Das lag wahrscheinlich daran, dass die Verbindung zu diesem Knoten temporär getrennt wurde und dadurch seine Nachrichten-Warteschlange überlief. Außerdem kam es insgesamt zu 6,5% doppelt gesendeter Pakete. Diese kommen zustande, wenn ein Acknowledgement für eine Nachricht verloren geht und diese darum erneut gesendet wird. Die Ergebnisse werden in [7] noch genauer erläutert.

### 2.3.2 *Einsatz im Freien: Das Projekt auf dem Genepi-Gletscher*

Nach diesem Schritt wurde damit begonnen, SensorScope auch für Outdoor-Projekte einzusetzen. Zuerst wurden Single-Hop Netzwerke getestet, um das Ganze möglichst einfach zu gestalten. Es wurden verschiedene Projekte durchgeführt. Eines davon auch auf dem 3000m hohen Genepi-Gletscher, um die Knoten unter rauhen Bedingungen zu testen.


doi: 10.2313/NET-2012-08-2_08

#### 2.3.2.1 *Projektbeschreibung.*

Als diese Tests nach beseitigung kleinerer Probleme, wie dem fehlerhaften Zusammenspiel von Software und den Hardware-Treibern (z.B. des Solarpanels) erfolgreich beendet waren, wurde damit begonnen, Sensorscope für Multi-Hop Deployments zu verwenden. Aus Platzgründen wird hier nur auf das wichtigste Projekt eingegangen[1]. Es fand auf dem 2500m hohen Genepi-Gletscher in der Schweiz statt und dauerte 60 Tage. Dieser Ort wurde ausgewählt, weil hier nach starken Regenfällen gefährliche Schlammlawinen auftreten. Sensorscope half dabei, die Klimabedingungen auf dem Gletscher besser zu verstehen und so bessere Vorhersagen treffen zu können, wann es zu Erdrutschen kommen könnte [10]. Es gelang, das Mikroklima auf dem Gletscher zu errechnen und so Schlammlawinen besser vorhersagen zu können.

Auf einer Fläche von 500 m x 500 m wurden 16 Sensorstationen und eine mit GPRS ausgestattete Senke aufgestellt. Obwohl GPRS aufgrund der schlechten Konnektivität auf dem Gletscher nur sehr eingeschränkt verfügbar war, war es für das Projekt ausreichend. Aufgrund der Bedeutung des Projekts wurde besonders darauf geachtet, dass möglichst viele Knoten nur einen Hop Abstand zur Senke hatten, um so die Anzahl an doppelten Nachrichten möglichst gering zu halten.

#### 2.3.2.2 *Gewonnene Erkenntnisse.*

In diesem Abschnitt wird genauer erläutert, welche Erkenntnisse durch das Genepi-Projekt für SensorScope gewonnen werden konnten.

**Stromversorgung**

Aus den Status-Nachrichten der Sensorknoten ging hervor, dass während des gesamten Projekts die sekundäre Batterie kein einziges mal verwendet wurde. Darum wurde geplant, in späteren Versionen von SensorScope nur noch eine Batterie einzubauen. Diese soll dafür größer sein als bisher.

**Datenauswertung**

Es ist wichtig, die gesammelten Messwerte zeitnah auszuwerten um Hardwarefehler schnell diagnostizieren zu können. Bei dem Genepi-Projekt kam es aufgrund eines durch Korrosion verursachten, Kurzschlusses der Interrupt-Leitung eines Regensensors zu fehlerhaften Messwerten. In einem solchen Fall ist es wichtig, den Fehler früh zu erkennen und zu beheben. Die Erkennung könnte zum Beispiel über den Vergleich mit Messwerten von anderen Sensoren geschehen.

**Testbedingungen**

Ein weiterer wichtiger Punkt ist es, unter möglichst realistischen Bedingungen zu testen. So gab es auf dem Genepi-Gletscher Phasen, in denen das GPRS-Signal so schwach war, dass die Senke zeitweise keine Nachrichten an den Server schicken konnte. Nachdem die Verbindung wieder hergestellt war, war die Senke ausschließlich damit beschäftigt, Nachrichten aus ihrem Speicher wegzuschicken, so dass sie momentan keine neuen Pakete mehr empfangen konnte, was zu deren Verlust führte. Dieses Fehlverhalten wurde während der Tests auf dem EPFL-Gelände nicht beobachtet, da hier eine stabile GPRS-Verbindung vorhanden war.

**Kameraeinsatz**

Die Sensoren von SensorScope können zwar die Niederschlagsmenge messen, aber nicht, um welche Art von Niederschlag es sich handelt. Sie können nicht zwischen Regen und geschmolzenem Schnee unterscheiden. Eine mögliche Lösung für dieses Problem wäre der Einsatz von Kameras, die Fotos von den Stationen machen. So könnte man sehen, ob Schnee liegt oder ob es regnet. Tatsächlich wurde auf dem Genepi-Gletscher bereits eine Kamera verwendet. Aufgrund des hohen Stromverbrauchs und Datenaufkommens war diese aber nicht in das SensorScope-Netzwerk integriert, sondern wurde autonom mit einer eigenen Autobatterie und einem eigenen GPRS-Gateway betrieben.

[1]Weitere Projekte sind unter [10] und [1] zu finden.

## 3. WEITERE UMWELTMONITORING SYSTEME

Natürlich gibt es neben SensorScope noch viele weitere Umweltmonitoring-Systeme. In diesem Kapitel werden zwei davon mit SensorScope verglichen. Zum einen Macroscope (Abschnitt 3.1), das benutzt wird um die Umwelt eines Baumes zu monitoren. Weiterhin wird noch PermaSense behandelt (Abschnitt 3.2). Es dient dem Monitoring von Temperatur und Feuchtigkeit in Felsen zur Vorhersage von Steinschlägen.

### 3.1 Macroscope

Die Inhalte dieses Abschnitts basieren auf den Angaben aus [13]. Macroscope wurde an der University of California in Berkeley entwickelt, um den Verlauf Mikroklimas im Umfeld eines 70m hohen Redwood-Bames zu beobachten und um Forschern zu ermöglichen, diesen Verlauf genauer zu untersuchen. Diese Daten sind relevant für Biologen, da man mit ihnen unter anderem Rückschlüsse auf das Wachstum von Bäumen ziehen kann. Mit den gewonnen Daten können sie nachvollziehen, ob ihre theoretischen Modelle auch in der Praxis zutreffen. Das Projekt hatte eine Laufzeit von 44 Tagen.

#### 3.1.1 *Aufbau und Funktionsweise von Macroscope*

Das Netzwerk von Macroscope besteht aus 33 Sensorknoten, die auf der Funktionsweise des in [6] vorgestellten Systems TASK basieren. Sie wurden in einer Höhe von 15 bis 70 m an dem Baum angebracht und haben zueinander einen Abstand von circa 2 m. Der Knotenaufbau ist schematisch in Abbildung 5 dargestellt.

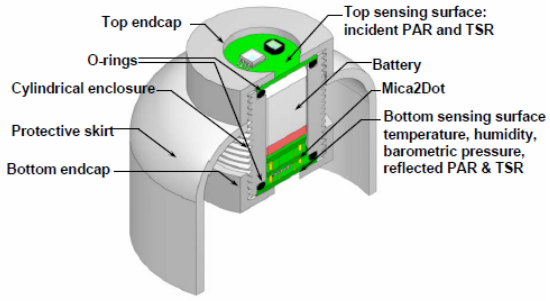


**Abbildung 5: Aufbau einer MacroScope-Station[13]**

Als Plattform wurde ein Mica2Dot von Crossbow verwendet, der einen Durchmesser von einem Inch hat. Außerdem verfügt die Plattform über einen Atmega128 Microprozessor


doi: 10.2313/NET-2012-08-2_08

von Atmega mit 4Mhz und einem 433Mhz-Funk von Chipcon. Weiterhin wurde ein Flash-Speicher mit einer Größe von 512 KB verbaut.
Vier verschiedene Sensoren dienen dazu, die Umgebungsbedingungen zu messen. Ein Sensor misst die Luft-Temperatur und -feuchtigkeit[2], ein weiterer den Luftdruck[3]. Weiterhin wird noch die photosynthetisch aktive Strahlung (engl. Photosynthetically Active Radiation; PAR) gemessen. Das ist der Anteil des Sonnenlichts, der für Photosynthese genutzt werden kann. Zudem wird noch das gesamte einfallende Sonnenlicht (engl. Total Solar Radiation; TSR) gemessen. Die Sensoren Für PAR[4] und TSR[5] sind je zweimal vorhanden. Einmal für das direkt einfallende Licht und einmal für das ambiente Licht, das von der Umgebung reflektiert wird. Dafür sind sie jeweils an der Oberseite des Gehäues und an der Unterseite angebracht. Zusätzlich zu Mikrokontroller und Sensoren verfügt eine Sensorstation auch noch über eine Batterie für die Energieversorgung.
Eine Solar-Vorrichtung wie bei SensorScope zum Laden der Batterie wurde nicht vorgesehen, da genung Energiesparmaßnahmen getroffen wurden um die gewünschte Laufzeit von 44 Tagen zu gewährleisten. So wurde genau wie bei SensorScope Duty-Cycling angewendet. Ein Zyklus dauerte 5 min, wovon nur 4 sec zum Austausch von Nachrichten genutzt wurden. Die hierfür nötige Synchronisation wurde von TASK erledigt. Wenn ein Knoten eine Nachricht von seinem Eltern-Knoten mithört übernimmt er dessen Zeit aufgrund eines Timestamps, der in allen Nachrichten mitgeschickt wird. So wird die Zeit des Root-Knotens Schritt für Schritt über das gesamte Netzwerk propagiert. Wenn ein Knoten über mehrere Duty-Cycles nichts von seinem Eltern-Knoten hört, bleibt er eine ganze Periode lang eingeschaltet, um sich wieder mit dem Eltern-Knoten zu synchronisieren. Mit diesem Verfahren wird eine Synchronität von 1 - 2 ms erreicht [6]. Im Gegensatz zu SensorScope sind hier keine expliziten Synchronisationsnachrichten notwendig.
Auch beim Routing gibt es Unterschiede zu Sensorscope. Bei Macroscope basiert das Routing auf MINTRoute [14]. Es werden Beacons verwendet, um für jeden Knoten den Nachbarn zu bestimmen, über den er die minimale Anzahl an Übertragungen benötigt, um eine Nachricht an die Senke zu schicken. Daraus wird ein Routing-Tree berechnet. Anders als bei SensorScope wird hier also ein Knoten immer den gleichen Weg zur Senke wählen, falls sich nichts an der Netzwerkstruktur ändert.
Genau wie Sensorscope benutzt Macroscope eine Basisstation (Senke), die die Daten per GPRS an einen Server weiterschickt.

### *3.1.2 Ergebnisse*

Aufgrund der räumlich dicht verteilten Messpunkte war es erstmals möglich die Dynamik des Mikroklimas in der Umgebung der Redwood-Bäume festzuhalten.
So konnten nicht nur qualitative, sondern auch quantitative Aussagen, über die biologischen Prozesse des Baumes, in Abhängigkeit seiner Umwelt, zu treffen [6].

[2]Sensiron SHT11 (Temperatur: ± 0.5 °C; Luftfeuchtigkeit: ± 3.5%)
[3]Intersema MS5534A
[4]Hamamatsu S1087
[5]TAOS TSL2550

## 3.2 PermaSense

Dieser Abschnitt beschreibt das Umweltmonitoring-System PermaSense und basiert größtenteils auf [11]. PermaSense dient der Erfassung von Daten über Permafrost in den schweizer Alpen. Es soll Geowissenschaftlern dabei helfen, ihr Wärmefluss-Modell von steilen Felshängen zu verbessern um deren Stabilität besser vorhersagen zu können. Das hier beschriebene Projekt fand am Jungfraujoch auf einer höhe von 3500 m statt. Es wurden 10 Sensorknotenverwendet. In Abbildung 6 wird der schematische Aufbau einer Station gezeigt. Genau wie bei SensorScope wurde GPRS verwendet

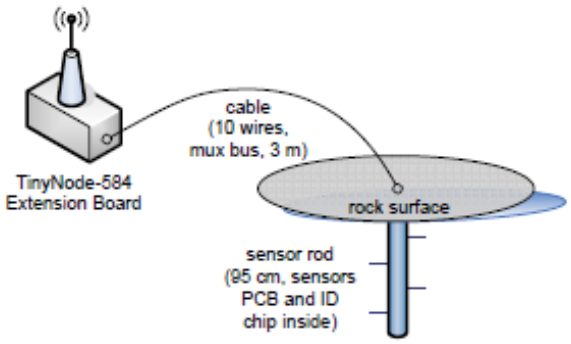


**Abbildung 6: Aufbau einer PermaSense-Station [11]**

um mit dem Server zu kommunizieren. Außerdem haben die Knoten auch einen TinyNode[6] und ein Semech Funk-Chip[7] verbaut, da diese einen geringen Energieverbrauch bei großer Funk-Reichweite bietet. Der geringe Energieverbrauch ist nötig, da die Knoten mit einer Batterie ohne sich wiederaufzuladen 4 bis 5 Jahre laufen sollen. Die hohe Funk-Reichweite ist nötig, weil die Knoten in Abständen von 250 bis 300 m aufgestellt wurden. Die Knoten sind mit einer 1 m langen Stange verbunden, an der Sensoren zum Messen der Temperatur und der Stromleitfähigkeit[8], angebracht sind. Diese Stange wird in ein 1 m tiefes Loch, das in den Fels gebohrt wurde, gesteckt. Das Loch hat einen Durchmesser von 14 mm. Die Sensoren sind in verschiedenen Tiefen an der Stange montiert. Die Stange ist über einen Bus mit dem TinyNode verbunden. Um robuster gegen Störungen zu sein werden immer 16 Messungn durchgeführt und dann gemittelt.
Da es vorkommen kann, dass die Knoten über 7 bis 8 Monate nicht für Menschen zugänglich sind gibt es einen Mechanismus, mit dem man die Abtastrate der Sensoren über das Netzwerk umstellen kann. Genau wie SensorScope verwendet PermaSense Duty-Cycling um Energie zu sparen. Die Länge eines Zyklus beträgt 30 Minuten und der Duty-Cycle beträgt 0,003%. Zur Synchronisation tauschen die Stationen bei jedem Zyklus ihre Zeiten aus und berechnen daraus ihre lokale Abweichung. Sie stellen jedoch daraufhin nicht ihre Uhr um, sondern passen nur das Zeitintervall bis zum nächsten Wake-Up an. Außerdem senden sie Nachrichten, in denen ihre lokale Abweichung steht. Um die Zeiten auf die tatsächliche Zeit umzurechnen bezieht die Senke die Zeit per

[6]TinyNode 585 von Stockfish
[7]Semtech XE1205
[8]Es wid die Leitfähigkeit zwischen zwei Messing-Ringen gemessen

NTP (Network Time Protocol) über GPRS.
Genau wie bei SensorScope wird Multi-Hop-Routing verwendet, aber der Routing-Mechanismus ist ein anderer. So gibt es einen sogenannten Transmission Corridor, der besagt, wie viele Nachrichten jeder Knoten in einem Zyklus senden darf. Dies sorgt dafür, dass Knoten, die über lange Zeit nich erreichbar waren (z.B. wegen Schnee), nicht das Netzwerk mit Nachrichten überfluten. Außerdem basiert das Routing auf einem Spanning Tree.
Gemessene Daten werden so lange in den Sensor-Knoten zwischengespeichert, bis sie ein Acknowledge vom Server erhalten, das besagt, dass die Daten erfolgreich übertragen wurden.

# 4. ZUSAMMENFASSUNG

Zum Schluss werden in diesem Kapitel noch die drei Vorgestellten Systeme miteinander verglichen. Eine Übersicht über die Unterschiede und Gemeinsamkeiten gibt Tabelle 2.

**Tabelle 2: Vergleich der drei in dieser Arbeit vorgestellten Systeme**

| | SensorScope | Macroscope | Permasense |
|---|---|---|---|
| Microcontroller | TinyNode | Mica2Dot | TinyNode |
| Stromversorgung | Solar + Batterie | Batterie | Batterie |
| Routing | randomisiert | fester Routing-Tree | fester Routing-Tree |
| Duty-Cycle | 120 s / 12 s | 5 min / 4 s | 30 min / 50 ms |
| Sync untereinander | einmal pro h | einmal pro 5min | einmal pro 30 min |
| Sample-Rate | 2 min | 5 min | 30 min |

Während SensorScope und Permasense TinyNode Microcontroller verwenden, verwendet Macroscope einen Mica2Dot. Bei der Stromversogung hat nur SensorScope die Möglichkeit siene Batterie mit Hilfe einer Solarzelle wieder aufzuladen. Bei den anderen beiden Systemen müsste die Batterie ausgetauscht werden, wenn sie leer ist. Was aber nicht vorkommen sollte, da die Lebensdauer der Batterie so ausgelegt ist, dass sie über die gesamte Dauer der Projekte hält. Wie in Abschnitt 2.2.5 beschrieben erfolgt bei SensorScope das Routing randomisiert, während Macroscope und Permasense zu beginn einen festen Routing-Tree festlegen. Alle drei Systeme verwenden Duty-Cycling um Energie zu sparen. Jedoch variieren die Sendeintervalle von 12 s alle 2 min bis zu 50 ms alle 30 min. Die Synchronisation erfolgt bei SensorScope einaml pro Stunde. Bei Macroscope wird mit jeder gesendeten Nachricht synchronisiert, was im Normalfall alle 5 Minuten der Fall ist. Bei Permasense erfolgt die Synchronisation zu Beginn jedes Sendeintervalls, was alle 30 min der Fall ist. Bei alle drei Systemen entspricht die Abtast-Rate der Sensoren der Länge des Duty-Cycles.

# 5. LITERATUR


[1] Sensorscope project homepage. Webseite. Online verfügbar unter http://sensorscope.epfl.ch/; besucht am 30.6.2012.

[2] C. Alippi and G. Vanini. A rssi-based and calibrated centralized localization technique for wireless sensor networks. In *Proceeding of Fourth annual IEEE International Conference on Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006.*, pages 5–pp. IEEE, 2006.

[3] G. Barrenetxea, M. Bystranowski, O. Couach, H. Dubois-Ferriere, S. Dufey, M. Krichane, J. Mezzo, S. Mortier, M. Parlange, G. Schaefer, et al. Demoabstract: Sensorscope, an urban environmental monitoring network. In *Proceeding of 4th European conference on wireless sensor networks (EWSN), Delft, Netherlands*, 2007.

[4] G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange. Sensorscope: Out-of-the-box environmental monitoring. In *Proceeding of International Conference on Information Processing in Sensor Networks. IPSN.*, pages 332–343. IEEE, 2008.

[5] BECOsolar, http://www.becosolar.com/DataSheets/MSX-01F & MSX-005F.pdf. *Datashet: Polycrystalline OEM Modules.*

[6] P. Buonadonna, D. Gay, J. Hellerstein, W. Hong, and S. Madden. Task: Sensor network in a box. In *Proceeedings of the Second European Workshop on Wireless Sensor Networks*, pages 133–144. IEEE, 2005.

[7] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange. Sensorscope: Application-specific sensor network for environmental monitoring. *ACM Transactions on Sensor Networks (TOSN)*, 6(2):17, 2010.

[8] I. L. P. Janick F. Artiola and M. L. Brusseau. *Environmental monitoring and characterization.* Elsevier Academic Press, Amsterdam Boston, 2004.

[9] X. Jiang, J. Polastre, and D. Culler. Perpetual environmentally powered sensor networks. In *Proceeding of Fourth International Symposium on Information Processing in Sensor Networks. IPSN.*, pages 463–468. IEEE, 2005.

[10] SensorScope. Portfolio. Webseite. Online verfügbar unter http://www.sensorscope.ch; besucht am 30.6.2012.

[11] I. Talzi, A. Hasler, S. Gruber, and C. Tschudin. Permasense: investigating permafrost with a wsn in the swiss alps. In *Proceedings of the 4th workshop on Embedded networked sensors*, pages 8–12. ACM, 2007.

[12] Texas Instruments, http://www.ti.com/lit/sg/slab034v/slab034v.pdf. *Datashet: TI MSP430.*

[13] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, et al. A macroscope in the redwoods. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 51–63. ACM, 2005.

[14] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 14–27. ACM, 2003.