

Informationsverwaltung in Sensornetzen

Sensorknoten-Hardware und Software

Autor: Nikolaus Huber, s_huber[at]ira.uka.de
Betreuer: Markus Bestehorn, bestehorn[at]ipd.uka.de

Universität Karlsruhe
Institut für Programmstrukturen und Datenorganisation (IPD)
Lehrstuhl Prof. Böhm

Zusammenfassung Bei der Vielzahl unterschiedlicher Typen von Sensornetzwerken und ihren Sensorknoten ist es nicht leicht, die Übersicht zu behalten. Diese Arbeit, im Rahmen des Seminars Informationsverwaltung in Sensornetzwerken, will erklären, wie die Hardwarearchitektur von Sensorknoten aufgebaut ist und welche Software eingesetzt wird. Dabei werden grundlegende Aspekte erläutert und die Details einiger spezieller Knotentypen vorgestellt (MicaMote, BTnode, SunSPOT). Ebenso werden bei der Software die wichtigsten Merkmale hervorgehoben und anhand konkreter Software (TinyOS und Squawk VM) erklärt. Ziel ist es, einen Überblick über die aktuell bestehende Hardware und Software zu bekommen, um die Vor- und Nachteile der verschiedenen Entwicklungen im Bereich der Sensorknoten gegenüberstellen zu können.

1 Einleitung

Sensornetze sind drahtlose, mobile ad-hoc Netze, die aus vielen einzelnen Sensorknoten aufgebaut sind. Sensorknoten wiederum sind Kleinstcomputer, die mit ihren Sensoren die Umgebung überwachen und die gesammelten Daten über Funk weitergeben können. Ein Mikrocontroller und integrierter Speicher hilft den Knoten dabei, die Daten, wenn nötig, vor zu verarbeiten um danach diese Daten über die Kommunikationseinheit weiterzugeben.

Sensornetze und Sensorknoten können in vielen Bereichen des Alltags eingesetzt werden, doch findet man sie heute überwiegend in der Forschung. Biologen benutzen Sensornetze, um beispielsweise das Verhalten von Tieren zu beobachten. Das hat bei der Verhaltensforschung den Vorteil, dass der Mensch nicht mehr in den Lebensraum der Tiere eindringen muss und dadurch das natürliche Verhalten stört. Weiterer Pluspunkt ist die Kosteneinsparung bei einer länger andauernden Beobachtungen von Tieren. Als Beispiel: Auf einer kleinen Insel an der Küste von Maine (Great Duck Island [1]) wurde ein Sensornetz von 32 Knoten ausgebracht, um das Nistverhalten der extrem scheuen Sturmschwalbe zu beobachten. Dabei wurde auch das Sensornetz auf seine Praxistauglichkeit getestet.

Ein weiterer Vertreter der Sensornetze bei der Beobachtung von Tieren ist das ZebraNet [2] der Princeton University. In Kenia wurden Zebras mit in Halsbändern integrierten Sensorknoten ausgestattet (siehe Abbildung 1), um vom biologischen Standpunkt aus betrachtet, die Wanderung der Tiere zu beobachten. Auf der Seite der Entwickler der Sensorknoten lagen die besonderen Schwerpunkte auf der Erprobung von drahtlosen Protokollen und positionsgetreuen Berechnungen unter den Gesichtspunkten der Energieeffizienz.



Abbildung 1. Sensorknoten von ZebraNet [2]

Das Militär stellt sich vor, mit einer Gruppe von Sensorknoten, die über dem Zielgebiet verstreut werden, Feindbewegungen beobachten zu können, ohne selbst

entdeckt zu werden [3]. Durch den Einsatz der Sensorknoten soll auch die Gefahr für den Menschen gesenkt werden. Getestet wurde diese Methode mit einem Netz bestehend aus 70 Knoten, dabei stand besonders die Energieeffizienz und die Genauigkeit der Überwachung im Vordergrund.

Alle Anforderungen und Funktionen, die ein Sensornetz gewährleistet und zur Verfügung stellt, müssen von den einzelnen Knoten erbracht und erfüllt werden, aber diese haben, in ihre Umgebung eingebettet, meist nur stark begrenzt zur Verfügung stehende Ressourcen. Bestes Beispiel ist der knappe Energievorrat. So ist zum Beispiel ein Sensornetz nur so lange einsatzbereit, wie seine Knoten es sind. Fallen mehrere Knoten eines Sensornetzes aus, kann man das Netz in den meisten Fällen nicht mehr verwenden. Schwierigkeiten wie diese und weitere haben zu ähnlichen aber auch unterschiedlichen Ansätzen geführt, die Probleme zu lösen.

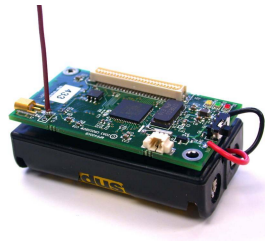


Abbildung 2. Ein Sensorknoten (Mica2 Mote) [13]

Diese Seminararbeit beschreibt im ersten Teil den schematischen Aufbau der Hardware und zeigt dann konkret die Hardwarearchitektur unterschiedlicher Sensorknoten. Der zweite Teil beschäftigt sich mit den allgemeinen Anforderungen, die an die Software von Sensorknoten gestellt werden, wie die einzelnen Implementierungen dies umsetzen und welche Unterschiede bzw. Gemeinsamkeiten zwischen den verschiedenen Ansätzen vorhanden sind. Abgerundet wird die Arbeit durch eine Gegenüberstellung der verschiedenen Knoten-Typen, um Struktur in die Vielzahl der unterschiedlichen Sensorknoten zu bringen und die Vor- und Nachteile einzelner Knoten unter Berücksichtigung des Einsatzbereichs aufzuzeigen.

2 Hardware von Sensorknoten

In diesem Kapitel wird insbesondere die Hardware von Sensorknoten näher erläutert. Dabei sollen die wesentlichen Bestandteile eines Sensorknotens beschrieben werden, also in welche Bereiche lässt sich die Hardware eines Sensorknotens unterteilen und welche Funktionen haben sie. Danach wird auf einige Knotentypen im Speziellen eingegangen. Der Schwerpunkt liegt dabei auf dem schematischen Aufbau bzw. auf den Bestandteilen eines Sensorknotens, um die Vor-/Nachteile einiger verschiedener Typen mit einander vergleichen zu können.

2.1 Allgemeine Architektur der Hardware

Allgemein lässt sich ein Sensorknoten in folgende schematische Komponenten (siehe Abbildung 3) unterteilen:

1. Prozesseinheit
2. Kommunikationseinheit
3. Sensoreinheit
4. Energiequellen

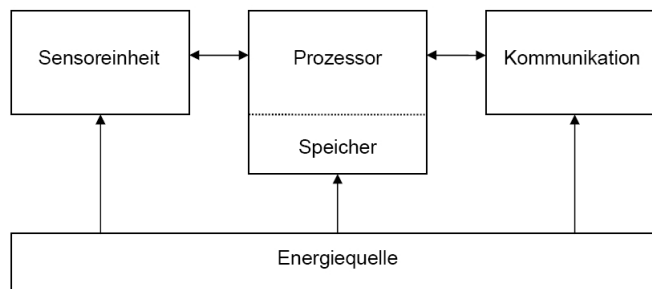


Abbildung 3. Schematischer Aufbau eines Sensorknotens

Meistens sind diese Komponenten der Sensorknoten als einzelnes Modul realisiert. Prozesseinheit, Kommunikationseinheit und Sensoreinheit findet man in der Regel auf einem gemeinsamen Board wieder, wobei man häufig auch zusätzliche Sensorboards kaufen kann. Die Prozesseinheit besteht aus einem Mikrocontroller mit internem Speicher und verarbeitet die von der Sensoreinheit empfangenen Daten. Alle Sensoren, die für die Messung von Daten zuständig sind, gehören zu der Sensoreinheit. Über die Kommunikationseinheit werden empfangene oder berechnete Daten an Nachbarknoten oder die Basisstation weitergeleitet. Die Kommunikation zwischen den verschiedenen Modulen ermöglicht ein Bus-System.

(1) Proessoreinheit Die Proessoreinheit besteht aus einem Mikrocontroller mit integriertem Speicher. Mikrocontroller sind kleinste Ein-Chip-Computer und vereinen CPU, Programmspeicher, Arbeitsspeicher und Ein-/Ausgabeschnittstellen in einem Chip. Heute sind sie überall vorzufinden, in MP3-Playern, Handys, Waschmaschinen usw.

Die Leistung eines Mikrocontrollers lässt sich über Faktoren wie den Taktzyklus und die Wortbreite sowie die Speichergröße beurteilen. Beispielsweise kann ein Mikrocontroller mit 8 Bit Wortbreite pro Takt ein Byte Daten verarbeiten. Die Leistung der Mikrocontroller ist nicht mit der normaler PC's vergleichbar, da den Sensorknoten nur sehr begrenzt Energie zur Verfügung steht und die Rechenleistung daher beschränkt ist.

Die Proessoreinheit verarbeitet die von den Sensoren oder, wenn nötig, die über die Kommunikationsschnittstelle erhaltenen Daten und verschickt sie anschließend weiter. Das Weiterleiten bzw. Versenden von Daten wird auch von der Proessoreinheit gesteuert.

(2) Kommunikationseinheit Zum Versenden von Daten ist die Kommunikationseinheit zuständig. Sie stellt eine Schnittstelle bereit, damit die gesammelten Daten an andere Sensorknoten weitergegeben werden können oder um Daten anderer Knoten zu empfangen. Pakete werden dabei entweder direkt zum Nachbarknoten gesendet oder über größere Strecken von Knoten zu Knoten weitergegeben (Multi-Hop), bis die Daten ihr Ziel erreichen.

Dabei gibt es unterschiedliche Wege, auf denen die Daten transportiert werden, wie z.B. Funk, Bluetooth oder Zigbee (siehe Abschnitt 2.2). Innerhalb eines Sensornetzes wird im Allgemeinen nur auf eine Art kommuniziert, da die Sensorknoten meistens nur einen Kommunikationstyp bereit stellen.

Bei allen Verfahren spielt der Energieverbrauch eine entscheidende Rolle, weshalb spezielle, energieeffiziente Netzwerkprotokolle entwickelt wurden (S-MAC, T-MAC, WiseMAC usw.), um z.B. Kollisionen beim Medienzugriff zu vermeiden oder den Kommunikationsaufwand zum initiieren einer Verbindung zu senken (siehe Abschnitt: 2.2).

(3) Sensoreinheit In der Sensoreinheit eines Knotens findet man einen oder auch mehrere unterschiedliche Sensoren. Es gibt Sensoren zur Messung der Lichtstärke, der Temperatur, der Luftfeuchtigkeit oder auch akustische Sensoren, Kameras oder Beschleunigungssensoren.

Die Auswahl eines zum gewünschten Einsatzbereich passenden Sensorknotens hängt somit unter Umständen auch von der Ausstattung des Sensorknotens mit Sensoren ab, jedoch kann häufig ein zusätzliches Sensorboard angeschlossen werden, um den Sensorknoten mit weiteren Sensoren auszurüsten. Zu beachten sind die verschiedenen Leistungsaufnahmen der einzelnen Sensortypen (siehe Tabelle 1, [4]). Die Wahl des richtigen Sensors hat also Einfluss auf die Lebensdauer des Knotens.

Tabelle 1. Leistungsaufnahme einiger Sensoren [4]

SENSORTYP	STROM
Lichtstärke	1.235 mA
Temperatur	0,150 mA
Luftdruck	0.010 mA

(4) **Energiequellen** Sensorknoten beziehen ihre Energie meistens aus Batterien, weshalb die zur Verfügung stehende Energie sorgfältig eingeteilt werden muss. Die Größe eines Sensorknotens wird maßgeblich von der Größe seiner Energiequelle beeinflusst (siehe Abschnitt 2.3). Es gibt auch Sensornetze und Sensorknoten wie z.B. das ZebraNet [2], die mit Solarzellen wieder Energie gewinnen können, doch vergrößert solch ein Ansatz den Sensorknoten meist erheblich.

Das Forschungsteam vom Bristol Robotics Laboratory geht mit dem EcoBot II [5] einen anderen Weg. Ausgestattet ist der Roboter mit sogenannten Microbial Fuel Cells (MFCs), die Mikroorganismen enthalten. Diese Mikroorganismen können mit Substrat (tote Fliegen oder verrottendes Obst) gefüttert werden und fungieren als Katalysator für chemische Gleichgewichtsreaktionen, mit denen man wiederum Energie aus dem Substrat gewinnen kann. Acht solcher MFCs in Reihe geschaltet produzieren genug Energie, um den Roboter schrittweise vorwärts zu bewegen.

Trotz dieser neuen Forschungsergebnisse bleibt die Energieversorgung das größte Problem der Sensorknoten. Aktuelle Sensorknoten verfügen meistens über zwei AA-Batterien, also etwa 4000 mAh, und können unter der Annahme eines durchschnittlichen Energieverbrauchs von $250 \mu\text{A}$ für etwa zwei Jahre betrieben werden, bevor man die Batterien austauschen muss. Hier zeigt sich deutlich, wie abhängig die Sensorknoten von der Energiequelle sind.

2.2 Kommunikations-Protokolle und Methoden

Kommunikationsmethoden: Im Anwendungsgebiet der Sensorknoten gibt es mehrere verbreitete Kommunikationsverfahren, die an dieser Stelle kurz erläutert werden sollen. Jede Kommunikation der später vorgestellten Sensorknoten läuft über ein ISM-Band, ein Frequenzbereich für Hochfrequenz-Sendegeräte. Der BT-node [16] verwendet beispielsweise den Bereich 433-915 MHz.

An anderes Verfahren, das der Sun SPOT verwendet und auch die Mica2 Motes können, ist Zigbee. Dieser Standard (IEEE 802.15.4) soll es ermöglichen, Geräte auf Kurzstrecken (1 - 100 m) zu verbinden und ist für den Betrieb im 868 MHz bzw. 2,4 GHz-Band definiert. Zigbee ist sehr ressourcen- und energiesparend, dafür fällt die Datenübertragungsrate mit 20-250 Kbit/s nicht so gut aus wie bei anderen Verfahren.

Eine weitere Kommunikationsmethode ist Bluetooth (IEEE 802.15.1) und arbeitet auch im ISM-Band zwischen 2,40 und 2,48 GHz und ist eigentlich eher als Ersatz für Kabel gedacht, benötigt daher relativ viel Ressourcen, bietet dafür eine hohe Bandbreite (720 Kbit/s) (vgl.: Tabelle 2, [6]).

Tabelle 2. Kommunikationsarten im Vergleich

NAME Standard	ZIGBEE 802.15.4	WI-FI 802.11.b	BLUETOOTH 802.15.1
Anwendungsziel	Überwachung und Steuerung	Web, eMail, Video	Kabelersatz
Systemressourcen	4 KB - 32 KB	1 MB+	250 KB+
Lebensdauer der Batterie in Tagen	100 - 1000+	0,5 - 5	1 - 7
Bandbreite in Kbit/s	20 - 250	11.000+	720+
Reichweite in Meter	1 - 100+	1 - 100	1 - 10+

Kommunikationsprotokolle: Zusätzlich zu den energiesparenden Kommunikationsmethoden gibt es für Sensornetze spezielle Protokolle, welche die Übertragung von Daten effizienter und energiesparender gestalten. Einige dieser Protokolle sollen hier vorgestellt werden.

S-MAC (Sensor-MAC) [7] ist ein Medienzugriffsprotokoll speziell für Sensornetzwerke und reduziert den Energieverbrauch dadurch, dass für die Knoten gleich große, feste Zeitfenster vergeben werden, die sich in zwei konstante Phasen unterteilen, in die *active-* und die *idle-Phase*.

Damit die Knoten alle zur gleichen Zeit aktiv sind und während dieser Zeit Nachrichten austauschen können, werden die Zeitfenster regelmäßig mit einem SYNC-Signal aneinander angeglichen. Dadurch werden auch neue Knoten in das bestehende Netz aufgenommen und es können alle Sensorknoten im gleichen, konstanten Zeitfenster kommunizieren. Bei diesem Verfahren entstehen sogenannte *virtuelle Cluster*, ein Nachteil, denn die Knoten am Rand von Clustern haben immer mehr Kommunikationsaufwand, da sie mit dem Nachbar-Cluster ihre Daten austauschen müssen.

Während der *idle-Phase* ist die Kommunikationseinheit inaktiv, um Energie zu sparen. Nur während der *active-Phase* versenden die Knoten Daten oder warten auf Pakete. Ein weitere Energieeinsparung wird dadurch erreicht, dass Knoten ihre Kommunikationsschnittstelle ausschalten, falls sie keine Daten versenden wollen und andere Knoten das Medium belegen. Daraus kann aber der Nachteil entstehen, dass es Knoten gibt, die noch Daten senden wollen, den Empfänger aber nicht mehr erreichen, weil dessen Kommunikationseinheit schon inaktiv ist.

Das **Timeout-MAC (T-MAC)** Protokoll [8] versucht diese Nachteile durch eine dynamische *active-Phase* auszugleichen. Hier soll jeder Knoten gleich am Anfang der *active-Phase* seine Daten versenden. Wenn ein Knoten keine Daten

zu versenden hat bzw. nach einer gewissen Zeitspanne auch keine Anfrage bei diesem Knoten eingegangen ist, verfällt er schon früher in den Schlafzustand als bei S-MAC.

Es gibt noch weitere eingesetzte Protokolle wie WiseMAC [9], XMesh ([10], [11]) sowie andere proprietäre Protokolle, deren Ziel auch die energie-effiziente Übertragung von Daten in Sensornetzen ist.

2.3 Verschiedene Sensorknoten

In den folgenden Abschnitten soll kurz auf einige Sensorknoten und ihre Architekturen eingegangen werden, die es zur Zeit zu kaufen gibt. Diese Auflistung von Sensorknoten ist keineswegs vollständig und soll nur einige unterschiedliche Typen und die damit verbundene Vielfalt von Sensorknoten vorstellen. Der Schwerpunkt liegt dabei auf den jeweiligen Besonderheiten des Sensorknotens.

Mica Motes Der Mica2 Mote ([12], [13]) wurde an der Universität Berkely in Kalifornien entworfen und wird von dem Unternehmen Crossbow Technologies [15] hergestellt und verkauft. Entwickelt wurde der Knoten für Sensornetze mit großem Umfang (mehr als 1000 Knoten).



Abbildung 4. Mica2Dot Mote

Der Mica2 Mote ist 58x32x7 mm groß und wiegt ohne Batterien 18 g (siehe Abbildung 2). Die verkleinerte Version des Mica2Motes, der Mica2Dot Mote [14] unterscheidet sich von dem Mica2 Mote vor allem durch Größe und Gewicht. Er ist gerade so groß wie eine Vierteldollar-Münze (25 mm Durchmesser) und wiegt 3 g (siehe Abbildung 4).

Der Prozessor beider Sensorknoten ist ein Atmel ATmega 128L mit 4MHz Takt und 8bit Wortbreite, 128 KByte Flash-Speicher und 4 KByte RAM. Beide Sensorknoten kommunizieren über Funk oder Zigbee, wobei an den größeren Mica2 Mote ein Interface oder Gateway Board angeschlossen werden kann, um ihn als Basisstation einzusetzen. Als Protokoll wird XMesh eingesetzt.

Das Betriebssystem, das auf beiden Knoten eingesetzt wird, ist TinyOS. Der Mica2Dot Mote verfügt nur über einen Temperatursensor, während der Mica2 Mote einen Temperatursensor, einen Lichtstärkesensor, einen Luftfeuchtigkeitsmesser, einen 2-Achsen-Beschleunigungssensor und ein Mikrofon besitzt. An beide Sensorknoten können noch weitere Sensorboards angeschlossen werden.

Der Mica2Dot Mote bezieht seine Energie aus einer 3 V Knopfzelle, weshalb er deutlich kleiner ist als der Mica2 Mote, der zwei AA-Batterien als Energiequelle nutzt. Vorteil des Mica2 Motes ist, dass er als Basisstation für die kleineren Mica2Dot Motes fungieren kann, dafür ist der Mica2Dot Mote aber wesentlich kleiner.

An diesen zwei Sensorknoten sieht man auch sehr deutlich den Größeneinfluß der Energiequelle (vgl. Abb. 2 und Abb. 4). Beide Sensoren haben die gleiche Leistungsfähigkeit mit Ausnahme der Sensoren, und trotzdem ist der Mica2Dot Mote ein vielfaches kleiner, nur aufgrund seiner kleineren, aber auch weniger leistungsfähigen Energiequelle.

BTnode Der BTnode Sensorknoten [17] wurde von der ETH Zurich entwickelt, ist ähnlich aufgebaut wie der Mica2 Mote und besitzt ebenfalls einen Atmel ATmega 128L Mikrocontroller, also auch 4 Mhz, 4 Bit Wortbreite, 128 KByte Falsh-Speicher und 4 KByte RAM. Jedoch ist der BTnode mit zusätzlichen 64 + 180 KByte SRAM ausgestattet. Auch das Gewicht und die Größe entsprechen dem Mica2 Mote.

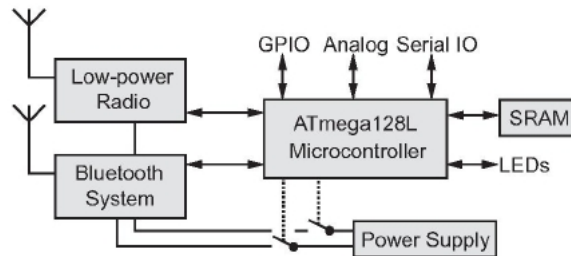


Abbildung 5. Schematischer Aufbau des BTnode

Die Besonderheit des BTnode ist seine zwei voneinander getrennt steuerbaren Wege zur Kommunikation, Bluetooth und Funk. Dadurch ist es möglich, den BTnode als Basisstation einzusetzen oder ihn als verbindenden Knoten zwischen zwei Sensornetztopologien, also Netze mit unterschiedlicher Kommunikationsart, zu benutzen (siehe Abbildung 6). Als Betriebssystem kann wahlweise TinyOS eingesetzt werden oder die „BTnut system Software“, die zusätzliche Unterstützung bei der Entwicklung bietet und im Gegensatz zu TinyOS die weiter verbreitete Programmiersprache C verwendet, so dass man keine neue Sprache erlernen muss.

An den BTnode können unterschiedliche Sensoren über eine dafür vorgesehene Schnittstelle angeschlossen werden. Die Energie des BTnode liefern wie bei den Mica2 Motes zwei AA-Batterien und seine Leistungsaufnahme liegt zwischen 9,9-198 mW. Allerdings macht sich die zusätzliche Kommunikationsschnittstelle nur bemerkbar, wenn man die benötigte Leistung beider Knoten unter maximaler Belastung betrachtet. Ohne Bluetooth braucht der Knoten nur 102,3 mW, mit Bluetooth dagegen 198 mW [17]. Dieser merklich größere Leistungshunger dürfte sich jedoch auf längere Zeit betrachtet in Grenzen halten, da man das Bluetooth-Modul nach belieben ausschalten kann und sicher nicht dauern einsetzt.

Die Veränderungen am BTnode sollen im Vergleich zum Mica2 mehr Flexibilität bieten, um einen Sensorknoten zu haben, der den Entwurf und das Testen von Anwendungen oder Protokollen erleichtern soll. Deshalb wurde zusätzlicher Speicher eingebaut und spezielle Software entwickelt, z.B. zum debuggen oder das „Sensor Network Maintenance Toolkit“.

Die zweite Kommunikationsschnittstelle macht den BTnode als Vermittler zwischen verschiedenen Sensornetzen besonders interessant bzw. hilft zusammen mit der Zusatzsoftware beim Entwickeln und Testen neuer Anwendungen, ist aber letztendlich beim realen Einsatz, wo meistens nur eine Kommunikationsschnittstelle eingesetzt wird, eher überflüssig [16].

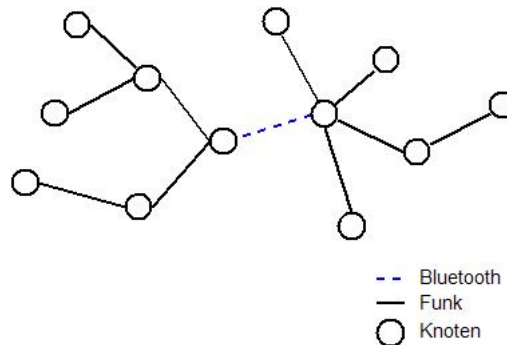


Abbildung 6. Verbinden zweier verschiedener Netze

Sun SPOT Die von Sun entwickelten Sun SPOTs (Smart Programmable Object Technology) [18] sind ein Versuch, die verfügbaren Sensorknoten, die hardwareorientiert und daher schwer zu programmieren sind, abzulösen. Dabei spielt die Squawk VM eine wichtige Rolle. Sie ist wie die Java VM eine virtuelle Maschine, die im Gegensatz zur Java VM direkt, also ohne Betriebssystem, auf den Knoten läuft und es möglich macht, die Sensorknoten mit Java zu programmieren (vgl. Abschnitt 3.3). Allerdings wird dafür eine leistungsstärkere Hardware benötigt.

Daher hat der aktuellste Sun SPOT einen ARM920T Prozessor mit 180 MHz Taktrate, 32 Bit Wortbreite, 4 MB Flashspeicher und 512 KByte SRAM. Kommunizieren kann der Sun SPOT über ein Zigbee-Interface (IEEE 802.15.4) bzw. über eine zusätzliche USB-Schnittstelle, um den Sun SPOT an den PC anzuschließen. Ausgestattet ist der Knoten mit einem Temperatur-, einem Licht- und einem Beschleunigungssensor.

Der Sun SPOT ist mit 35x25 mm ein bisschen kleiner als der Mica2 Mote. Das Standardsensorboard umfasst einen Temperatur- und einen Lichtsensor sowie einige Aktoren. Die Energie bezieht der Sun SPOT aus einem 3,7 V LiIo-Akku mit 750 mAh, den man über den USB-Anschluss laden kann. Er kann aber auch mit normalen 1,5V AA-Batterien betrieben werden. Der Stromverbrauch des Sun SPOT beträgt je nach Operation zwischen 25-90 mA, im Ruhezustand liegt er unter 35 μ A [19].



Abbildung 7. Sun SPOT (Small Programmable Object Technology)

3 Betriebssysteme und Software

3.1 Allgemeine Anforderungen an das Betriebssystem

Das Betriebssystem oder die Software, die auf einem Sensorknoten eingesetzt wird, unterscheidet sich stark von einem Betriebssystem wie Linux oder Windows. Es muss spezielle Anforderungen erfüllen, beispielsweise ist die Leistung von einem Sensorknoten stark beschränkt. Auch können nicht, wenn überhaupt, beliebig viele Tasks nebeneinander ausgeführt werden, da der Speicherplatz, den ein Betriebssystem einnimmt, nicht zu groß werden darf.

Weiter sollte das Betriebssystem dafür sorgen, dass möglichst wenig Energie verbraucht wird, weshalb verschiedene Betriebsmodi wie Active, Idle, PowerDown, PowerSave (siehe Abschnitt 3.2) unterstützt werden sollten. Trotz allem soll das Betriebssystem möglichst vielseitig eingesetzt werden können, sei es bei der Gebäudeüberwachung oder bei der Beobachtung von Tieren. Genauso sollten Programmierfehler nach Möglichkeit nicht zu einem Systemcrash führen, was

häufig vorkommt, weil in den Betriebssystemen für Sensorknoten Speicherschutzmechanismen fehlen.

Aus diesen Gründen liegt die Herausforderung bei der Entwicklung solcher Betriebssysteme darin, trotz dieser widrigen Umstände möglichst alle diese Probleme zu lösen und dem Anwender eine möglichst verwendbare und robuste Systemumgebung zu bieten. Im Folgenden sollen zwei dieser Ansätze vorgestellt werden.

3.2 TinyOS

Das Open-Source-Betriebssystem TinyOS [21] wurde an der Universität Berkeley entwickelt und ist in NesC geschrieben, eine wenig verbreitete, komponentenbasierte Form von C. Diese Sprache muss man lernen, wenn man Anwendungen für TinyOS schreiben will. Ein großer Vorteil von TinyOS: es verbraucht sehr wenig Ressourcen. Der Systemkern lässt sich in 432Byte implementieren, RAM-Verbrauch ab 46Byte.

Das Schichtenmodell TinyOS ist wie die meisten Betriebssysteme ereignisorientiert (*event-based*) und ist aus mehreren Schichten (*layer*) aufgebaut, die wiederum aus unabhängigen *components* bestehen. Dabei bildet die Hardware die unterste Schicht. Zwischen den Schichten wird über *commands* und *events* kommuniziert. Höhere Schichten schicken immer *commands* an tiefer gelegene Schichten. *Events* werden immer ausgehend von einer tieferen an eine höhere Schicht gesendet und können initial nur von der Hardware-Schicht ausgelöst werden (vgl. Abbildung 8).

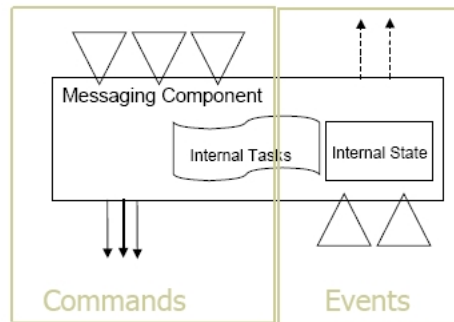


Abbildung 8. Aufbau einer *component*

Zwei-Level-Scheduling TinyOS verwendet ein Zwei-Level-Scheduling, das bedeutet: In jeder *component* gibt es eine Reihe von Aufgaben, die erledigt werden müssen (*tasks*). Diese *tasks* werden in eine Warteschlange gestellt und nach dem

FIFO-Prinzip abgearbeitet (First-In-First-Out). Für die *tasks* gibt es keine Priorisierung und die *tasks* können sich auch nicht gegenseitig unterbrechen. Durch Systemereignisse, signalisiert durch *events*, kann die Abarbeitung einer *task* angehalten werden. *Tasks* können also von *events* unterbrochen werden, deswegen werden *tasks* auch als zeitflexibel und *events* als zeitkritisch bezeichnet.

Sollten die Warteschlangen leer sein, schaltet TinyOS den Sensorknoten in den Schlafmodus um Energie zu sparen, bis ein neues Systemereignis ausgelöst wird. Dabei unterscheidet TinyOS folgende Betriebsmodi:

- **active**, alles an
- **idle**, Prozessor aus
- **power down**, Knoten ist ausgeschaltet
- **power save**, wie power down, ein noch aktiver Timer kann den Knoten wieder aufwachen lassen

Die Energie, die man durch das Ausschalten von einzelnen Komponenten eines Knotens einspart, ist enorm. Der BTnode braucht mit eingeschaltetem Prozessor 39,6 mW, schaltet man den Prozessor in den Schlafmodus, braucht der BTnode nur noch 9,9 mW. Die Differenz von fast 30mW macht mehr als 25% der maximalen Leistungsaufnahme (102,3 mW) aus.

3.3 Squawk VM

Mit der speziell für Sensorknoten entwickelten Squawk VM geht Sun einen anderen Weg als TinyOS. Die Squawk VM ist größtenteils in Java geschrieben, setzt direkt auf der Hardware auf („*Java on bare metal*“, [18]) und bildet ein CLDC- (Connected Limited Device Configuration [20]) und damit auch J2ME-konformes Betriebssystem. Es ist also kein dazwischenliegendes Betriebssystem wie bei der Java VM nötig. Trotzdem kann man die Squawk VM auf herkömmlichen Rechnern und Netzwerken einsetzen und deshalb können auch alle bekannten Werkzeuge zum entwickeln, debuggen, testen usw. von Java-Programmen verwendet werden. Sun stellt außerdem Tools zur Verfügung, die einen Sun SPOT auf dem Desktop visuell repräsentieren können. Das Ziel der Squawk VM ist, das Arbeiten mit Sensornetzen deutlich zu vereinfachen.

Codeoptimierung Obwohl der Sun SPOT über eine üppigere Hardwareausstattung im Vergleich zu anderen Sensorknoten verfügt, muss der Code, der auf einem Knoten ausgeführt wird, verkleinert werden, denn auch die Speichermenge des Sun SPOT ist begrenzt und Java-Code mit seinen Objekten braucht mehr Speicher als z.B. C-Code. Erreicht wird die Platzersparnis zum einen durch Bytecodeoptimierungen (da fast alles in Java geschrieben ist) und durch die Vorab-Verifikation der Klassen z.B. auf dem PC, bevor sie auf der VM eingesetzt werden (Verifier in der VM überflüssig, dadurch Platzersparnis).

Die verifizierten Klassen werden danach zu sogenannten *suites* gebündelt, eine speichereffiziente Repräsentation der Klassen, die in der *suite* enthalten sind.

Solche *suites* benötigen durchschnittlich nur 35% der Größe einer Klasse. Trotz der Verkleinerungen braucht die in [18] vorgestellte Version immer noch 80KByte RAM und 270KByte Flash-Speicher.

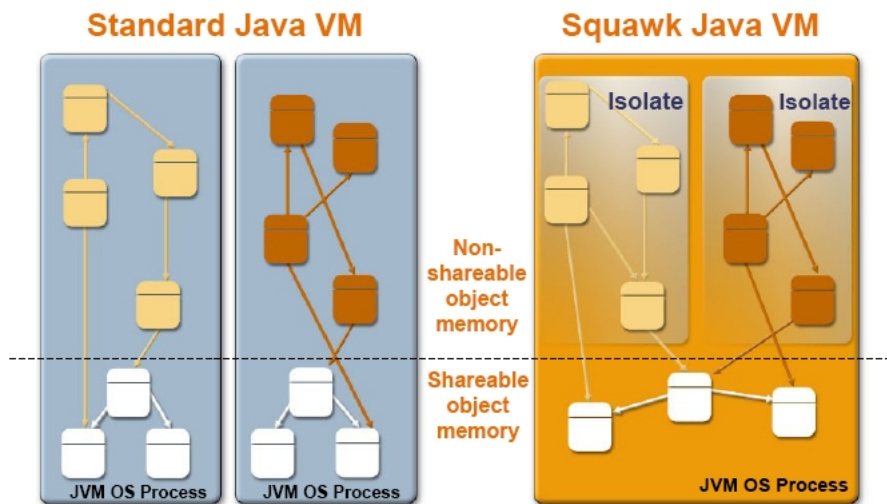


Abbildung 9. Die Squawk VM im Vergleich zur Standard Java VM

Isolate Application Model Eine weitere Besonderheit der Squawk VM ist, dass sie mehrere Anwendungen gleichzeitig ausführen kann. Ermöglicht wird das durch das *Isolate Application Model*. Jede Anwendung wird in der Squawk VM durch ein eigenes Java-Objekt repräsentiert und dieses Objekt ist eine Instanz der Isolate-Klasse, ein so genanntes *Isolate*. Jedes Isolate hat einen abruf- und steuerbaren Zustand (start, pause, resume, exit), der völlig unabhängig von anderen Isolates ist. So ist es auch möglich, mehrere Isolates auf der VM gleichzeitig auszuführen.

Aufgrund des *Isolate Application Model* ist es auch möglich, Isolates und damit Applikationen von einem Sensorknoten auf einen anderen Knoten oder auf einen PC zu migrieren, da sie komplett von einander unabhängig sind. Es ist auch möglich, dass sich z.B. eine Anwendung von selbst über das Sensornetz verteilt oder man ein Isolate zu Wartungsarbeiten einfach auf ein Notebook migriert.

Das *Isolate Application Model* bietet eine Funktionalität, die ganz besonders Wartungsarbeiten an Hardware und Software und insbesondere den Austausch von ganzen Knoten unterstützt, wenn z.B. die Batterie leer ist. Man muss nur die Software von einem Knoten auf den anderen migrieren und die Knoten austauschen.

3.4 Vergleich der beiden Betriebssysteme

Mit TinyOS und Squawk VM hat man zwei Betriebssysteme zur Auswahl, die sehr unterschiedliche Vor- und Nachteile aufweisen. Ersteres überzeugt durch seine geringe Größe und der Hardwarenähe (Systemkern ab 432Byte, nesC). Dafür bietet die Squawk VM deutlich mehr Unterstützung bei der Entwicklung (Java-Code, Standardwerkzeuge). Auch wird bei der Squawk VM der Aspekt der Nebenläufigkeit deutlicher als bei TinyOS und die bessere Portierbarkeit von Anwendungen erleichtert den Umgang mit den Sensorknoten. Dafür muss man aber einen hohen Preis in Sachen Lebensdauer und Praxistauglichkeit zahlen, denn die mit der Größe der Mica2Dot Motes kann der Sun SPOT nicht mithalten und die Leistungsaufnahme des Sun SPOT liegt mit ca. $35\mu\text{A}$ [19] deutlich höher als die des Mica2 Motes mit weniger als $1\mu\text{A}$ (vgl. Abschnitt 4).

4 Gegenüberstellung der Sensorknoten

Von den vielen unterschiedlichen Sensorknoten-Typen gibt es zwei unterschiedliche Herangehensweisen an das Thema „Sensorknoten - Hardware und Software“, die auffallen. Zum einen ist das der eher praxis-orientierte, für den realen Einsatz entworfene Ansatz mit dem Mica2 Mote und TinyOS als Vertreter. Auf der anderen Seite sind das der Sun SPOT mit der Squawk VM als Vertreter mit dem Schwerpunkt auf der Entwicklung von Sensornetzen. Beide Knoten-Typen weisen ihre Vor- und Nachteile auf, die an dieser Stelle nochmals kurz zusammengefasst werden sollen.

Im Bezug auf die Größe unterscheiden sich beide kaum, doch wenn man den Mica2Dot Mote hinzunimmt, wird der Vorteil offensichtlich. Auch die Lebensdauer der Mica2 Motes ist besser. Der Mica2 Mote braucht im Ruhezustand weniger als $1\mu\text{A}$, die Leistungsaufnahme des Sun SPOTs im Ruhezustand liegt bei unter $50\mu\text{A}$. Die Vorteile des Sun SPOTs liegen eher im Bereich der Entwicklungsmöglichkeiten. Während man bei TinyOS eine neue Programmiersprache lernen oder bei den BTnodes zumindest C können muss, kommt man bei den Sun SPOTs mit Java zurecht. Auch bekommt man hier viel mehr Unterstützung bei der Entwicklung, bei der Fehlersuche und auch bei der Wartung von Sensornetzen.

Zusammenfassend kann man sagen, dass für das Arbeiten mit Sensornetzen die Sun SPOTs besser geeignet sind. Die Vorteile, die man bei der Entwicklung hat, erkauft man sich allerdings wie bereits erwähnt durch geringere Lebensdauer und Einschränkungen in der Vielseitigkeit der Anwendungsmöglichkeiten. Andere Sensorknoten wie die Mica2 Motes überzeugen dafür im realen Einsatz, leider muss man sich deshalb aber mit speziellen Programmiersprachen und einer sehr hardwarenahen Implementierungen auseinandersetzen. Deshalb sind die Sun SPOTs eher geeignet, neue Technologien zu testen und zu optimieren um diese dann nachher auf andere Sensorknoten-Typen zu übertragen und in der realen Welt einzusetzen.

5 Fazit

Die noch junge Technologie der Sensornetze bietet eine Vielzahl an unterschiedlichen Sensorknoten für unterschiedliche Einsatzmöglichkeiten. Diese Technologien sind im Forschungsbereich alle praxistauglich, es gibt aber Stellen, die ausbaufähig sind. Das Hauptproblem stellt sicherlich die sehr begrenzte Energie dar. Würden hier neue Energiequellen entwickelt oder gefunden, würde sich die Größe der Sensorknoten reduzieren und somit die Einsatzmöglichkeiten vergrößern. Ein weiterer Vorteil besserer Energiequellen oder energie-effizienteren Entwürfe wäre die erhöhte Lebensdauer, momentan sicherlich ein Grund, Sensornetze noch nicht überall praktisch einzusetzen, weil der Wartungsaufwand sehr hoch ist.

Was die Zukunft angeht, so wird sich im Bereich der Sensorknoten mit Sicherheit noch einiges verändern. Es wird viel an Sensornetzen gearbeitet und geforscht und damit auch die Sensorknoten verbessert. Sun hat in dieser Richtung mit möglichst viel Unterstützung bei der Entwicklung einen wichtigen Schritt gemacht. Bei anderen Sensorknoten ist eine Verbesserung implementierter Verfahren und Algorithmen nur schwer möglich, nämlich nur, indem man komplett neu anfängt. Hier könnten die Sun SPOTs die Entwicklung beschleunigen, von der letzten Endes auch andere Knotentypen profitieren können.

Literatur

1. ALAN MAINWARING, JOSEPH POLASTRE, ROBERT SZEWCZYK, DAVID CULLER, JOHN ANDERSON, „*Wireless Sensor Networks for Habitat Monitoring*“, WSNA '02, 2002
2. ZEBRA NET (PRINCETON UNIVERSITY)
<http://www.princeton.edu/~mrm/zebranet.html>
3. TIAN HE, SUDHA KRISHNAMURTHY, JOHN A. STANKOVIC, TAREK ABDELZAHER, LIQIAN LUO, RADU STOLERU, TING YAN, LIN GU, JONATHAN HUI, BRUCE KROGH, „*Energy-Efficient Surveillance System Using Wireless Sensor Networks*“, Mo-biSYS 04, June 69, '2004,
4. JOSEPH ROBERT POLASTRE, „*Design and Implementation of Wireless Sensor Networks for Habitat Monitoring*“, Springer, 2003
5. BRISTOL ROBOTICS LABORATORY, *EcoBot II*, <http://www.ias.uwe.ac.uk/>
6. ZIGBEE ALLIANCE, <http://www.zigbee.org/en/about/faq.asp#7>
7. WEI YE, JOHN HEIDEMANN, DEBORAH ESTRIN, „*An Energy-Efficient MAC Protocol for Wireless Sensor Networks*“
8. TIJS VAN DAM, KOEN LANGENDOEN, „*An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks*“, SenSys '03, November 5-7, 2003
9. A. ELHOIYDI, J.D. DECOTIGNIE, „*WiseMAC: An Ultra Low Power MAC Protocol for the Downlink of Infrastructure Wireless Sensor Networks*“, ISCC '04, pages 244-251, June 2004
10. MIKE HORTON, JOHN SUH, „*A Vision for Wireless Sensor Networks*“, IEEE, 2005
11. CROSSBOW, „*XMesh, Mesh Networking*“, <http://www.xbow.com/Technology/MeshNetworking.aspx>
12. JASON L. HILL, DAVID E. CULLER, „*Mica: a wireless platform for deeply embedded networks*“, IEEE Micro, November-December 2002
13. CROSSBOW, *Mica2 Wireless Measurement System*, http://www.xbow.com/Products/Product_pdf_files/Wireless.pdf/MICA2_Datasheet.pdf
14. CROSSBOW, *Mica2Dot Wireless Microsensor Mote*, http://www.xbow.com/Products/Product_pdf_files/Wireless.pdf/MICA2DOT_Datasheet.pdf
15. CROSSBOW TECHNOLOGIES, <http://www.xbow.com/>
16. JAN BEUTEL, SWISS FEDERAL INSTITUTE OF TECHNOLOGY (ETH) ZURICH, „*Fast-prototyping Using the BTnode Platform*“, European Design and Automation Association, 2006
17. ETH ZURICH, *BTnodes - A Distributed Environment for Prototyping Ad Hoc Networks*, <http://www.btnode.ethz.ch/>
18. RANDALL B. SMITH, CHRISTINA CIFUENTES, DOUG SIMON, „*Enabling Java for Small Wireless Devices with Squawk and SpotWorld*“, www.ics.uci.edu/~lopes/bspc05/papers/smith.pdf
19. SUN, *SunSpotWorld*, <http://www.sunspotworld.com/>
20. CONNECTED LIMITED DEVICE CONFIGURATION (CLDC), <http://java.sun.com/products/cldc/>
21. JASON HILL, ROBERT SZEWCZYK, ALEC WOO, SETH HOLLAR, DAVID CULLER, KRISTOFER PISTER, „*System Architecture Directions for Networked Sensors*“, ASPLOS-IX 11/00 Cambridge, MA, USA