Vom Smart Dust zum Smart Phone: Verfügbare Sensing-Plattformen

Adrian Friedli

Departement für Informatik, ETH Zürich

adrianfriedli@student.ethz.ch

Zusammenfassung

Für Sensing-Applikationen stehen verschiedene Geräte mit Sensoren zur Verfügung. In Sensornetzen kommen meist sowohl spezialisierte, extra für eine bestimmte Aufgabe produzierte Sensorknoten, als auch generische Sensorknoten, welche durch Erweiterungsmöglichkeiten mit eigenen Sensoren erweitert werden können, zum Einsatz. Spezialisierte Sensorknoten sind aufgrund der beschränkten Anforderungen sehr klein und nicht sehr leistungsfähig. Die generischen Sensorknoten sind deutlich grösser, bieten dafür aber mehr Leistung und Speicherplatz, sodass darauf komplexere Programme ausgeführt werden können. Bei Mobiltelefonen eignen sich vor allem Smartphones für Sensing-Applikationen, da sie im Vergleich zu gewöhnlichen Mobiltelefonen mehr Sensoren integriert haben, deutlich leistungsfähiger sind und auch die Programmierung nicht so stark eingeschränkt ist.

Abbildung 1: Ein Smartphone mit dem Android Betriebssystem

1 Einführung

Die Bandbreite der programmierbaren Geräte, welche über Sensoren die reale Welt mit der virtuellen Welt verknüpfen können, ist gross. Vom „Smart Dust“, sehr kleinen Sensorknoten in Sensornetzen, über Massenmarkt-Mobiltelefone bis zu aktuellen Smartphones gibt es verschiedene Sensing-Plattformen mit unterschiedlichem Leistungsvermögen.

Angefangen mit den ortsgebundenen Sensornetzen betrachten wir die Hardwareeigenschaften, die Sensing-Fähigkeiten und die Programmierbarkeit der Sensorknoten. In einem zweiten Teil widmen wir uns dann den Mobiltelefonen, welche in den letzten Jahren verstärkt auch als allgemeine Computing und Sensing-Plattform eingesetzt werden.

2 Sensorknoten in Sensornetzen

2.1 Sensornetze

Ein Sensornetz (engl.: wireless sensor network) besteht aus mehreren selbstständigen Sensorknoten, welche über Funk miteinander kommunizieren und dadurch ein Ad-hoc-Netz aufbauen können. Die Sensornetze können zur Überwachung ihrer Umgebung genutzt werden, um zum Beispiel vor Gefahren wie Erdbeben, Luftverschmutzung oder Feuer zu warnen, oder das Verhalten von Menschen oder Tieren zu beobachten [9]. Üblicherweise bestehen solche Netze aus verschiedenen Klassen von Sensorknoten [6]:

- Spezialisierte Sensorknoten werden für eine spezielle Aufgabe hergestellt und sind dadurch sehr klein und verbrauchen wenig Energie. Ein solcher Knoten hat zum Beispiel nur einen Temperatursensor, ein Funkmodul und sehr wenig Speicher und Leistung.
- Generische Sensorknoten können durch verschiedene Sensoren erweitert werden und sind dadurch flexibel einsetzbar. Sie sind leistungsfähiger als spezialisierte Sensorknoten und können dadurch komplexere Programme ausführen und sind daher nicht auf wenige Spezialaufgaben beschränkt.
- Für die Kommunikation mit der Aussenwelt werden Gateways genutzt, welche sehr leistungsfähig sind und oft als Senke im Sensornetz fungieren.

Da die Sensornetze oft über längere Zeit ohne menschliche Hilfe funktionieren müssen, ist ein tiefer Energieverbrauch der speziellen und generischen Sensorknoten wichtig, denn im Gegensatz zu Mobiltelefonen ist ein regelmässiges Aufladen der Batterien oft nicht möglich oder unerwünscht. Dies hat nicht nur auf die Konstruktion der Hardware Auswirkungen, sondern auch auf die Programmierung der Geräte, welche auf die begrenzten Energieressourcen Rücksicht nehmen muss. Das Open-Source-Betriebssystem TinyOS [17] wurde speziell für drahtlose Sensornetze entwickelt und ermöglicht, die knappen Ressourcen optimal zu nutzen. Für die speziellen Anforderungen der Programmierung von Sensorknoten wurde eine eigene Programmiersprache mit dem Namen nesC als Erweiterung von C entwickelt und das Betriebssystem komplett in dieser implementiert. Die meisten generischen Sensorknoten verwenden dieses Betriebssystem. Alternativen sind zum Beispiel Java ME, das bei den Sun SPOTs verwendet wird, und BTnut, welches bei den BTnodes Anwendung findet.

2.2 Sensorknoten

Die spezialisierten Sensorknoten haben eine Grösse von wenigen Kubikmillimetern. Das Schlagwort Smart Dust kommt von der Grösse dieser Knoten. Als Beispiel dient der Spec Chip, wie er in Abbildung 2 zu sehen ist. Er wurde an der University of California von J. Hill entwickelt [5]. Er besitzt keinen Sensor und kann für eine periodische Anwesenheitskontrolle verwendet werden. Er hat eine Grösse von 2 x 2.5 mm und besitzt 3 kByte Speicher. Über den eingebauten Funksender kann der Spec über wenige Meter mit anderen Knoten kommunizieren. Dafür benötigt er nur wenige externe Komponenten wie eine Antenne und eine Stromversorgung.

Der Spec besitzt eine SPI Programmierschnittstelle [15] über die der C Code auf den Chip geladen werden kann. Der Energieverbrauch liegt im Ruhezustand in der Grössenordnung von $1\ \mu W$ und im aktiven Zustand

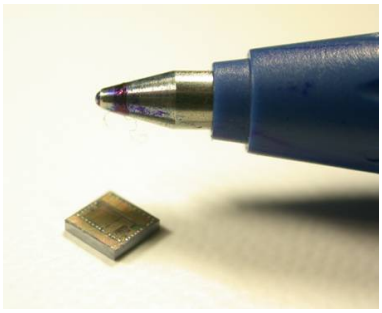

Abbildung 2: Der Spec Chip im Vergleich mit einem Kugelschreiber. [5]

(d.h. der Prozessor und der Funksender sind aktiv) bei wenigen mW. Mit einer kleinen Batterie ausgestattet kann er über Jahre laufen und z.B. periodisch seine Anwesenheit bestätigen, womit Gegenstände vor Diebstahl gesichert werden können. [6]

Die Generischen Sensorknoten zeichnen sich durch die Möglichkeit aus, dass sie durch verschiedene Sensoren erweitert und so individuell den Bedürfnissen angepasst werden können. Für diese Flexibilität benötigen die Sensorknoten leistungsfähigere Hardware als spezialisierte Knoten. Im Folgenden werden zwei Geräte genauer betrachtet: die BTnodes [1] und die Sun SPOTs von der Firma Sun [12].

Die BTnode Plattform wurde an der ETH Zürich für Forschungs- und Demonstrationszwecke entwickelt. Das in Abbildung 3 gezeigte Gerät kann sowohl über Bluetooth als auch über einen Chipon-Funksender mit anderen Knoten kommunizieren und kann so mit anderen Geräten, wie beispielsweise den Mica2 Sensorknoten in einem Netz zusammenarbeiten. Die beiden Funkarten können zusammen verwendet oder einzeln abgeschaltet werden, um den Energieverbrauch zu reduzieren. Mit Strom versorgt wird das Gerät mit zwei AA Batterien oder durch den Anschluss an das Stromnetz. Es besitzt einen Atmel Mikrocontroller mit 8 MIPS (Millionen Instruktionen pro Sekunde), 64 kByte SRAM und 128 kByte Flash Speicher. Es gibt zwei externe Anschlüsse, einen um das Gerät mit Sensoren zu erweitern und einen zum Debuggen der Software. Die Grösse beträgt 58.15 x 33 mm.

Abbildung 3: Der BTnode rev3 wurde an der ETH Zürich entwickelt und kann als generischer Sensorknoten in Sensornetzen eingesetzt werden. [1]

Auf dem BTnode läuft ein Open-Source-Betriebssystem für eingebettete Systeme mit dem Namen Nut/OS. Programmieren kann man das Gerät sowohl auf der Hardwareebene, um diese möglichst effizient und energiesparend zu nutzen, als auch auf der Betriebssystemebene mit diversen Libraryfunktionen. Die Programmiersprache ist C. An einem Support für TinyOS wird noch gearbeitet. [1]

Einen neuen Ansatz verfolgt Sun mit den Sun SPOTs (Abbildung 4). Diese werden fast komplett in Java programmiert, was es erlaubt, solche Sensorknoten ohne spezielles Wissen über eingebettete Systeme zu programmieren. Die Java VM läuft direkt auf der Hardware, sodass kein Betriebssystem benötigt wird. Die verwendete Java Implementation heisst Squawk und ist kompatibel zu Java ME und bietet dazu noch grundlegende Betriebssystemfunktionalität und unterstützt CLDC 1.1 (Connected Limited Device Configuration) und MIDP 1.0 (Mobile Information Device Profile).

Abbildung 4: Ein Sun SPOT im Grössenvergleich mit einem amerikanischen Viertel-Dollar.

Die Sun SPOTs besitzen einen 32-Bit ARM920T Prozessor mit 180 Mhz, 512 kByte RAM, 4 MByte Flash Speicher und können mittels Funk mit anderen Sun SPOTs kommunizieren. Als Energieversorgung kommt ein 750 mAh Akku zum Einsatz, welcher über den USB Anschluss aufgeladen wird. Eine Akkuladung reicht für eine Laufzeit von sieben Stunden, wenn der Prozessor und der Funksender aktiv sind, und bis zu 900 Tagen im Ruhezustand. Die Sun SPOTs haben folgende Sensoren bereits eingebaut:

- 3D Beschleunigungssensor
- Temperatursensor
- Lichtsensor

Weitere Sensoren können über die Erweiterungsstecker angeschlossen werden.

Plattform-name	Programmierung/ Betriebssystem	Energieverbrauch aktiv	Energieverbrauch im Ruhezustand	Standardmässig vorhandene Sensoren
Spec	C / -	wenige mW	$1\,\mu W$	-
Tmote Sky [10]	nesC / TinyOS	$60\,mW$	$15\,\mu W$	Feuchtigkeit Temperatur
MICAz [4]	nesC / TinyOS	$84\,mW$	$100\,\mu W$	-
BTnode	nesC / BTnut	$400\,mW$	$30\,mW$	-
SUN SPOTs	Java ME / -			3D Beschleunigung Temperatur Helligkeit

Tabelle 1: Übersicht über ausgewählte aktuelle Sensorknoten

3 Mobiltelefone

Mobiltelefone sind im Gegensatz zu Sensornetzen ortsunabhängig und die Energieversorgung ist aufgrund der Möglichkeit, den Akku regelmässig aufzuladen, weniger kritisch. Im Folgenden wird zwischen Massenmarkt-Mobiltelefonen und Smartphones unterschieden. Smartphones sind leistungsfähiger als gewöhnliche Mobiltelefone und besitzen teilweise berührungsempfindliche Bildschirme. Das Betriebssystem der meisten Smartphones stammt von einem Drittanbieter, und es ermöglicht es dem Benutzer, einfach eigene Programme zu schreiben und auf dem Gerät zu installieren. Auf gewöhnlichen Mobiltelefonen können meist nur, wenn überhaupt, Java Anwendungen hinzugefügt werden. [16] Eine Übersicht ber aktuelle Mobiltelefone wird in Tabelle 3 am Ende des Kapitels gegeben.

3.1 Massenmarkt-Mobiltelefone

Die meisten Massenmarkt-Mobiltelefone besitzen heutzutage eine Kamera. Diese ist aber vom Leistungsumfang nicht mit einer Digitalkamera zu vergleichen. Die Kameras sind meist auf gute Lichtverhältnisse angewiesen und besitzen keine Zoomfunktionalität. Meist kann auch von den Java-Programmen nicht auf die Kamera zugegriffen werden. Für Sensig-Applikationen stehen daher oft nur der Bluetoothadapter und das Mikrofon des Mobiltelefons zur Verfügung.

Als Beispiel eines gewöhnlichen Mobiltelefons dient hier das Nokia 6230. Das in Abbildung 5 gezeigte Telefon läuft mit einem Nokia-eigenen Betriebssystem und für Benutzerprogramme stehen höchstens 3 Mbyte zur Verfügung, wobei ein JAR File maximal 125 kbyte gross sein darf. Es unterstützt Bluetooth- und Infrarot-Verbindungen und besitzt eine 1.3 Megapixel Kamera ohne Zoommöglichkeit.

Abbildung 5: Das Nokia 6230 [8]

Als Programmierplattform kommt das Series 40 SDK zum Einsatz. Mit dieser können mittels Java und Flash Lite, einer abgespeckten Version von Adobe Flash [2], Anwendungen programmiert werden. Unter anderem werden die Java APIs für Bluetooth und Mobile Media unterstützt, wobei Mobile Media auf Audio- und Videowiedergabe und Audioaufnahme beschränkt ist. Ein Zugriff auf die Bilder der Kamera ist daher nicht möglich.

3.2 Smartphones

Smartphones sind noch nicht so weit verbreitet wie gewöhnliche Mobiltelefone. Aber die Verbreitung nimmt laufend zu: im Jahr 2008 stieg der Verkauf um etwa 28%. Vor allem Apple konnte mit dem iPhone die

Verkaufszahlen markant steigern. [3]

Auf Smartphones kommt meist ein Betriebssystem eines Drittanbieters zum Einsatz. Dabei ist Symbian mit 46.6% momentan der klare Marktführer. Eine Ausnahme bildet hier das iPhone, auf dem das Apple-eigene Betriebssystem „OS X iPhone“ zum Einsatz kommt. In Tabelle 2 sind die aktuellen Marktanteile der verschiedenen Betriebssystemanbieter aufgeführt.

Betriebssystem-Anbieter	Verkaufszahlen 3. Quartal 2008	Marktanteil
Symbian	18’583’060	46.6%
Apple	6’899’010	17.3%
RIM	6’051’730	15.2%
Microsoft	5’425’470	13.6%
Linux	2’028’490	5.1%
Andere	862’340	2.2%
Total	39’850’100	100%

Tabelle 2: Marktanteile der Betriebssystem-Anbieter im Jahr 2008 [3]

Das in Abbildung 6 gezeigte iPhone 3G von Apple besitzt je nach Ausführung 8 oder 16 GByte Flash Speicher und bietet damit genügend Speicherplatz für zusätzliche Programme. Diese werden in der Programmiersprache Objective-C [14] geschrieben, welche eine Objekt-Orientierte Erweiterung von C ist. Andere Sprachen wie zum Beispiel Java oder Python werden nicht unterstützt. Dazu ist der Zugriff auf die 2.0 Megapixel Kamera nur beschränkt möglich und für die Programmierung benötigt man Apple Hardware.

Abbildung 6: Das Apple iPhone 3G

Neben GPS und Bluetooth bietet das iPhone für Sensing-Applikationen eine Reihe von weiteren Sensoren:

- Beschleunigungssensor
- Annäherungssensor
- Umgebungslichtsensor
- Mikrofon

Als weiteres Beispiel für ein Smartphone betrachten wir das Nokia N95 8G (Abbildung 7). Es sind 8 Gbyte Speicher für Daten vorhanden und für die Ausführung von Programmen stehen 90 Mbyte Ram zur Verfügung.

Neben GPS und Bluetooth besitzt das N95 noch einen 3D-Beschleunigungssensor. Auf dem Gerät läuft das Symbian Betriebssystem in der Version 9.2. Die verwendete Entwicklungsplattform ist S60 SDK for Symbian OS. Dabei werden die folgenden Programmiersprachen unterstützt: [7]

- Symbian C++ und Open C/C++
  Zusätzlich zu den Symbian C++ APIs können Entwickler auf diverse C und C++ Bibliotheken wie die STL und IOStream zurückgreifen. Symbian C++ eignet sich dann, wenn eine schnelle Ausführung des Programms entscheidend ist, oder wenn der Zugriff auf bestimmte Hardware des Smartphones in einer anderen Programmiersprache nicht möglich ist. Der Nachteil von Symbian C++ ist vor allem die lange Einarbeitungszeit und die fehlende Dokumentation.

- Java
  Mit Java haben Entwickler Zugriff auf CLDC 1.1 (engl. Connected Limited Device Configuration), der kleinstmöglichen Konfiguration einer Java ME Umgebung. Zusätzlich stehen diverse APIs bereit, wie zum Beispiel Bluetooth, Advanced Multimedia Supplements, Wireless Messaging und weitere. Java ist eine gute Alternative für Entwickler, die sich nicht in Symbian C++ einarbeiten wollen. Die Palette an Tools und Dokumentation ist sehr umfangreich. Der Nachteil von Java ist, dass nicht immer auf alle Hardware zugegriffen werden kann, so ist zum Beispiel der Zugriff auf 3D Beschleunigungssensoren mit Java ME momentan noch nicht möglich.

- Python
  Python ist eine einfach zu erlernende Script-Sprache. Python for S60 bietet eine Vielzahl von APIs und lässt sich im Gegensatz zu Java durch C++ Module ergänzen. Der Nachteil ist die Geschwindigkeit der Programmausführung, insbesondere im Vergleich mit Symbian C++.

- Weitere Sprachen wie Web Runtime und Flash Lite

Abbildung 7: Das Nokia N95 8G

Android-Plattform

Android ist ein Betriebssystem für Smartphones welches von Google entwickelt wurde und zum grössten Teil Open-Source ist. Um die Weiterentwicklung kümmert sich die Open Handset Alliance, welcher momentan 47 Firmen angehören. Als Grundlage für Android dient ein Linux Kernel 2.6, welcher sich um alle Betriebssystemaufgaben kümmert.

Alle Programme werden in Java geschrieben, und diese laufen dann in der von Google entwickelten Dalvik VM, welche ähnlich wie die VM von Sun arbeitet. Der grösste Unterschied liegt in der Prozessorarchitektur der virtuellen Maschine. Die Java VM hat eine Stapelmaschine als Grundlage und Dalvik eine Registermaschine, wie die meisten aktuellen Prozessoren auch. Das übersetzen der Java-Programme erfolgt dadurch in zwei Schritten. Zuerst wird der Programmcode mit dem Java-Compiler in Java Bytecode, und danach vom Dalvik Cross-Assembler in Dalvik Bytecode übersetzt. Für die Entwicklung der Java-Programme wurde Java ME durch weitere Android-spezifische Bibliotheken erweitert. In geschwinigkeitskritischen Bereichen greifen diese auf nativ, in C++ geschriebenen Code zurück.

Alle Java Anwendungen des Systems können durch eigene Programme ersetzt werden, wodurch das System fast komplett den eigenen Bedürfnissen angepasst werden kann. Das erste Smartphone mit Android wurde von T-Mobile unter dem Namen G1 im Jahr 2008 in den USA zum Verkauf angeboten. [13][11]

Mobiltelefon	Betriebssystem	Programmiersprachen	Sensoren / GPS / Kamera
Nokia 6230	Nokia OS	Java ME Flash Lite	1.2 Megapixel Kamera
iPhone	OS X iPhone	Objective-C	3D Beschleunigungssensor Annäherungssensor Umgebungslichtsensor GPS 2.0 Megapixel Kamera WLAN
Nokia N95 8G	Symbian	Symbian C++ Java CLDC Python und mehr	3D Beschleunigungssensor GPS 5.0 Megapixel Kamera WLAN
T-Mobil G1	Android 1.0	Java	3D Beschleunigungssensor GPS 3.2 Megapixel Kamera WLAN

Tabelle 3: Eigenschaften von aktuellen Mobiltelefonen / Smartphones. Bluetooth und Mikrophon sind bei allen aufgezählten Geräten vorhanden.

4 Schlussfolgerungen

Sensornetze bieten durch den Einsatz von verschiedenen Klassen von Sensorknoten eine flexible Lösung für Sensing-Applikationen, welche eine fixe Umgebung überwachen sollen. Die drei Klassen von Sensorknoten decken dabei die ganze Bandbreite von möglichen Leistungsanforderungen ab. Von spezialisierten Knoten mit kleiner Grösse und Leistung über generische Knoten, die mehrere Sensoren steuern können, bis zu den Gateways, die sehr leistungsfähig sind und als Schnittstelle zur Aussenwelt (Internet oder Benutzer) dienen und im Netz meist als Senke fungieren, ist alles erhältlich. Da auf den meisten generischen Sensorknoten TinyOS als Betriebssystem läuft, erfolgt die Programmierung der Geräte häufig in nesC, einer speziell für Sensorknoten entwickelten, auf C aufsetzenden Programmiersprache. Mit den SUN SPOTs verfolgt die Firma Sun den Ansatz, die Programmierung zu vereinfachen, indem die Knoten mit Java programmiert werden.

Bei den Mobiltelefonen eignen sich die Massenmarkt-Mobiltelefone nur begrenzt für Sensing-Applikationen, da sie meist keine Sensoren enthalten und nicht auf die Kamera zugegriffen werden kann. Nur Bluetooth und das Mikrophon bleiben für die Interaktion mit der realen Welt. Die Smartphones hingegen bieten meist diverse Sensoren, GPS und wesentlich leistungsfähigere Hardware. Der Marktführer von Betriebssystemen für Smartphones ist Symbian. Dieses System erlaubt es dem Entwickler zwischen mehreren Programmiersprachen auszuwählen. Auch ist der Zugriff auf sämtliche Hardware mittels Symbian C++ möglich. Nicht mit Symbian, sondern mit dem Betriebssystem von Apple ausgestattet ist das iPhone. Es konnte seinen Markanteil startk ausbauen und bietet einen grosse Palette an integrierten Sensoren. Die Programmierung ist aber wegen der restriktiven Lizenzpolitik von Apple momentan auf Objective-C eingeschränkt. Mit dem im Jahr 2007 von Google vorgestellten Betriebssystem Android, welches auf Linux basiert und zum grössten Teil Open-Source ist, kommt ein Betriebssystem für Smartphones auf den Markt, welches den Entwicklern viele Anpassungsmöglichkeiten bietet.

Literatur

[1] Dokumentation zu der BTnode Plattform. http://www.btnode.ethz.ch/. [Stand: 13. Feb. 2009].

[2] Adobe. Flash Lite. http://www.adobe.com/products/flashlite/.

[3] Canalys. Marktanteile von Smartphones im Jahr 2008. http://www.canalys.com/pr/2008/r2008112.htm.

[4] Crossbow. Produktdetails MICAz. http://www.xbow.com/Products/productdetails.aspx?sid=164.

[5] J. Hill. Spec takes the next step toward the vision of true smart dust. http://www.jlhlabs.com/jhill_cs/spec/. [Stand: 13. Feb. 2009].

[6] R. K. Jason Hill, Mike Horton and L. Krishnamurthy. The Platforms Enabling Wireless Sensor Networks. *Communications of the ACM, SPECIAL ISSUE: Wireless sensor networks*, 47(6):41–46, Juni 2004.

[7] Nokia. Die Enwicklerplattform S60 für das Symbian Betriebssystem. http://www.forum.nokia.com/Resources_and_Information/Explore/Software_Platforms/S60/.

[8] Nokia. Geräte Details für das Mobiltelefon 6230. http://www.forum.nokia.com/devices/6230.

[9] K. Römer. Einführung in Sensornetze. Fachseminar Sensornetze im Sommersemester 2003 an der ETH Zürich.

[10] Sentilla. Tmote Sky. http://www.sentilla.com/moteiv-endoflife.html.

[11] M. Stäuble. Googles mobile Plattform Android. http://www.heise.de/developer/artikel/print/120124.

[12] Sun. Sun SPOTs Documentation. http://www.sunspotworld.com/.

[13] Wikipedia. Android Plattform. http://de.wikipedia.org/wiki/Android_(Plattform). [Stand 16. Feb. 2009].

[14] Wikipedia. Objective-C. http://de.wikipedia.org/wiki/Objective-C.

[15] Wikipedia. Serial Peripheral Interface. http://de.wikipedia.org/wiki/Serial_Peripheral_Interface. [Stand 13. Feb. 2009].

[16] Wikipedia. Smartphone. http://de.wikipedia.org/wiki/Smartphone. [Stand: 16. Februar 2009].

[17] Wikipedia. TinyOS. http://de.wikipedia.org/wiki/Tinyos. [Stand 13. Feb. 2009].