Hussein Hammoud, 230768

Aufgabenstellung

Das Gebiet der Sensornetzwerke nimmt zunehmend an Bedeutung zu [1][2][3]. Ein Sensornetzwerk besteht aus einzelnen ressourcenarmen Sensorknoten, die in beliebigen Umgebungen eingesetzt und verschiedene physikalische Größen messen können, sowie aus Basisstationen, die die gemessenen Daten der Sensorknoten sammeln, vorverarbeiten und anschließend an höheren Schichten, in der Regel dem Benutzer, weiterleiten. Der Datenfluss kann aber auch in die andere Richtung gehen. Softwareupdates oder Erteilung von Kommandos für einzelne Sensorknoten bzw. das gesamte Netzwerk sind entsprechende Anwendungsfälle.

Als Ausgangslage für diese Arbeit dient ein System aus einem Sensorknoten und einem Notebook, die über die Bluetooth- Technologie miteinander kommunizieren können. Der Sensorknoten stellt eine Basisstation eines Sensornetzwerks dar und übernimmt daher unter anderem die Kommunikation mit der Außenwelt, dem Anwender mit dem Notebook. Auf dem Sensorknoten läuft eine Anwendungssoftware, implementiert in der Programmiersprache C, die verschiedene Funktionalitäten implementiert und diese als Dienste zur Verfügung stellt, die vom Anwender angefordert bzw. gestartet werden können. Diese Funktionalitäten erlauben unter anderem die Verwaltung von (Mess-) Daten auf dem Sensorknoten, den Zugriff auf der SD-Speicherkarte als dauerhaftes Speichermedium dieser Basisstation, die Übertragung von Dateien sowie die Steuerung einer angebrachten Kamera. Die entsprechenden Anforderungen können mit Hilfe einer Konsolen- Applikation, die auf dem Notebook läuft und ebenfalls in der Programmiersprache C implementiert ist, gestellt werden. Die Kommunikation zwischen dem Sensorknoten und dem Notebook erfolgt wie bereits erwähnt drahtlos durch Einrichtung einer virtuellen seriellen Verbindung mit Hilfe entsprechender Bluetooth- Module auf beiden Seiten. Für die einfache Programmierung und Debugging liegt der Sensorknoten mit den anderen Komponenten wie Kamera, SD-Karte und Bluetooth-Modul als Entwicklungsplatine vor. Für die Programmentwicklung dienen zugehörige Entwicklungswerkzeuge der Firma Microchip, Hersteller des zur Grunde liegenden 16-Bit Signalcontrollers dsPICxxxxx. Diese beinhalten unter anderem eine integrierte Entwicklungsumgebung (MPLAB IDE) sowie einen speziell für die eigenen Signalcontroller optimierten C-Compiler, den MPLab C30.

Im Rahmen dieser Diplomarbeit soll zum einen eine funktional gleiche Software wie die beschriebene Konsolen-Applikation als ein User-Interface für ein Nokia N70 Handy implementiert werden. Anders als die existierende Variante, wird die neue Implementierung für das Nokia Handy aus Gründen der Benutzerfreundlichkeit natürlich als eine grafische Benutzeroberfläche realisiert. Die Implementierung der neuen Kommunikationsschnittstelle mit dem Sensorknoten wird vorteilhaft in Java erfolgen. Die spezifisch für das Handy und andere ressourcenbeschränkte (mobile) Geräte angepasste Java-Plattform, Java Platform Micro Edition (Java ME), stellt unter anderem nützliche Klassen für die Erstellung grafischer Anwendungen bereit. Ein weiterer wichtiger Vorteil ist die geschätzte Plattformunabhängigkeit von Java. Das zu entwickelnde User-Interface wird damit auch auf andere mobile Geräte ausführbar sein, die eine Java virtuelle Maschine haben, die die Java ME implementiert. Bei dem Kommunikationspartner am anderen Ende, dem Sensorknoten, sollen die Implementierungen der oben genannten Dienste in der Anwendungssoftware möglichst weitgehend unverändert bleiben.

Des Weiteren soll mittels Datenkompression die Menge der in beiden Richtungen zu übertragenden Daten reduziert werden. Die Datenkompression soll also sowohl auf dem Sensorknoten als auch auf das Nokia Handy implementiert werden. Die von den Sensorknoten über einen Zeitraum ermittelten Messwerte oder aufgenommenen Bilder sollen dadurch vor Übertragung an dem Benutzer, dem Handy, komprimiert werden können. Dies ist nicht zuletzt für die Lebensdauer der üblicherweise batteriebetriebenen Sensorknoten mit dem relativ hohen Energieverbrauch auf seitens des Kommunikationsmoduls vorteilhaft oder gar notwendig [3]. Auf der anderen Seite erfordern Anwendungsfälle wie Softwareupdates für die Sensorknoten ebenfalls Datenkompression auf dem Nokia Handy vor Übertragung an das Sensornetzwerk. Die zu übertragenden und damit zu komprimierenden Daten lassen sich grob in Text- und Bilddaten unterscheiden. Der einzusetzende Kompressionsalgorithmus soll also in Abhängigkeit des Typs der zu komprimierenden Daten jeweils so angepasst werden bzw. sich so verhalten, dass die Ausgabe stets optimal klein ist. Aufgrund der erwähnten Ressourceneinschränkungen bei dem Sensorknoten, aber auch bei dem Handy, soll das Kompressionsverfahren ferner hinsichtlich Speicherverbrauchs möglichst optimal sein.

Die Datenkompression soll zunächst mit Hilfe des Verfahrens der *arithmetischen Kodierung* realisiert werden. Vereinfacht gesagt kodiert dieses Verfahren einen Strom aus Eingangssymbolen durch eine Zahl, die zwischen 0 und 1 liegt [8]. Aus einer so kodierten Zahl lassen sich durch Dekodierung als entgegengesetzte Operation die Originaldaten verlustfrei wieder gewinnen. Bei geeigneter Wahl des sogenannten statistischen Modells lassen sich mit der arithmetischen Kodierung optimale Kompressionsraten erzielen [8]. Darauf aufbauend soll die Datenkompression danach auch mit dem Verfahren *Range Coder* implementiert werden. Dieses läuft im Detail nur etwas anders ab als die arithmetische Kodierung, ist aber dafür schneller. Anschließend sollen die beiden Verfahren anhand der erzielten Ergebnisse miteinander verglichen werden.

Die Kommunikation zwischen dem Nokia Handy und dem Sensorknoten soll auch hier mit der Bluetooth- Technologie erfolgen, wobei analog zur existierenden Lösung ebenfalls eine virtuelle serielle Schnittstelle eingerichtet wird. Das zu entwickelnde User-Interface ist also um eine zusätzliche (Software-) Komponente zur Kommunikation über das bereits im Handy integrierte Bluetooth-Modul zu erweitern. Mit Hilfe des (optionalen) Pakets *javax.Bluetooth* von Java ME erlangt man Zugang zu dem Bluetooth-Stack auf dem Handy. Die Implementierung der Bluetooth-Komponente für das User-Interface wird dadurch wesentlich vereinfacht.

Arbeitsplan

- Das bereits existierende System aufbauen und testen, so dass wir sicherstellen können, einen funktionierenden Sensorknoten als Kommunikationspartner fürs Handy zu haben. Geschätzter zeitlicher Aufwand: 1-2 Wochen.

- Entwurf (UML-Klassendiagramm, Statecharts etc.) des zu entwickelnden User-Interfaces und Bluetooth- Komponente. Geschätzter zeitlicher Arbeitsaufwand: bis 1 Woche.

- Implementierung des zuvor entworfenen Software-Modells. Geschätzter zeitlicher Arbeitsaufwand: 4 Wochen.

- Tests und eventuelle Verbesserungen/Korrekturen. Geschätzter zeitlicher Arbeitsaufwand: 1-2Wochen.

- Schriftliche Dokumentation der erledigten Arbeitsschritte. Geschätzter zeitlicher Arbeitsaufwand: 4-5 Wochen.

Wenn die genannten Arbeitsschritte erfolgreich abgeschlossen sind, wäre der zuvor existierende Stand mit unserer neuen Implementierung funktional erreicht. Man kann also dann mit dem Handy Kommandos an den Sensorknoten schicken, die er ausführt und die Resultate an das Handy zurück schickt. Anschließend folgen die unten aufgelisteten Arbeitsschritte.

- Entwurf & Implementierung der Datenkompression mit dem Verfahren der arithmetischen Kodierung. Geschätzter zeitlicher Arbeitsaufwand: 3-4 Wochen.

- Entwurf & Implementierung der Datenkompression mit dem Verfahren des Range Coders. Geschätzter zeitlicher Arbeitsaufwand: 2 Wochen.

- Kompression von Bildern mit der arithmetischen Kodierung.

- Systemtests und eventuelle Verbesserungen/Korrekturen. Geschätzter zeitlicher Arbeitsaufwand: 2 Wochen.

- Schriftliche Dokumentation der erledigten Arbeitsschritte. Geschätzter zeitlicher Arbeitsaufwand: 4-5 Wochen.

Literatur

[1] Hahn und Neukum, Ludwig Maximilians Universität München, Institut für Informatik: „Sensornetzwerke“, Hauptseminar „Dienste & Infrastrukturen mobiler Systeme“. http://www.nm.ifi.lmu.de/teaching/Seminare/2003ws/itiletom/DIMS/Ausarbeitung_Hahn_Neukum.pdf, Stand: 09.05.2008.

[2] Golatowski et al., Universität Rostock, Institut f. Angewandte Mikroelektronik und Datentechnik: Artikel „Softwarearchitektur für Sensornetzwerke“. http://www-md.e-technik.uni-rostock.de/veroeff/Softwarearchitektur_fuer_Sensornetzwerke.pdf, Stand: 07.05.2008.

[3] David Culler et al., University of California, Berkeley: „Overview of Sensor Networks“, IEEE-Fachartikel, 2004, Ausgabe 8. http://compilers.cs.ucla.edu/emsoft05/CullerEstrinSrivastava04.pdf, Stand: 07.05.2008.

[4] Adam Wolisz & Andreas Willig, Technische Universität Berlin, Fachgebiet Telekommunikationsnetze: Vorlesung „Ad-hoc und Sensornetzwerke“. http://www.tkn.tu-berlin.de/curricula/ws0708/vl-adsen/papers/AdHoc-Sensor-Introduction.pdf, Stand: 02.05.2008.

[5] Thorsten Stetter, Universität Mannheim, Lehrstuhl für Praktische Informatik: „Smart Dust“, Seminararbeit. http://www.informatik.uni-mannheim.de/pi4.data/content/courses/2001-ws/ubiquitouscomputing/smartdust_arbeit.pdf, Stand: 09.05.2008.

[6] Mark Weiser: „The Computer for the 21st Century “, Aufsatz. http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html, Stand: 10.05.2008.

[7] Matthias Handy, Universität Rostock, Institut für Angewandte Mikroelektronik und Datentechnik: Vorlesung „Hardwarenahe Programmierung“. http://www-md.e-technik.uni-rostock.de/ma/hm13/lehre/v_wsn.ppt, Stand: 10.05.2008.

[8] Mark Nelson: „Arithmetic Coding + Statistical Modeling = Data Compression“ Part 1 & Part 2, Fachartikel für Dr.Dobb’s Journal, Feb. 1991. , http://dogma.net/markn/articles/arith/part1.htm bzw. http://dogma.net/markn/articles/arith/part2.htm, Stand: 01.05.2008.