Analyse und Entwicklung einer µ-Controller-basierten Echtzeitsteuerung für Sensorsysteme mit Ada-Raven

Dipl. Inf. (BA) Matthias Götz[1], Dr.-Ing. Jörg Matthes, Dr.-Ing. Hubert B. Keller, Dipl.-Math. Rolf Seifert, Institut für Angewandte Informatik, Forschungszentrum Karlsruhe GmbH, [1]ci-Tec GmbH Karlsruhe
Matthias.Goetz/, Joerg.Matthes/, Hubert.Keller/, Rolf.Seifert@iai.fzk.de

Für die Ansteuerung einer Elektronischen Nase wurde eine Echtzeitsteuerung mittels Ada und dem Ravenscar-Profil entwickelt. Ziele waren hierbei der Entwurf und die Umsetzung eines mehrstufigen abstrakten Softwaremodells und eine Abbildung der nebenläufigen Aktivitäten zur Steuerung der Elektronischen Nase. Letzteres wird durch die Verwendung von Ada erreicht, denn das Ada Laufzeitsystem ermöglicht den Einsatz von parallelen Tasks mit verschiedenen Ausführungssträngen. Um den Zuverlässigkeitsanforderungen für den Einsatz des Sensorsystems gerecht zu werden, wurde das Ravenscar-Profil, ein restriktives Subset von Ada, mit einem Tasking-Modell und preemptivem Scheduling verwendet.

1 Problemstellung

Die eingesetzte Elektronische Nase [1] (Abbildung *1*) ermöglicht es, Gasgemische qualitativ (Klassifikation der vorhandenen Gase) und quantitativ (Konzentrationsbestimmung) zu erfassen. Sie besteht aus einem Hubmagnet, einem Temperaturfühler, einem Feuchtigkeitsfühler und einem Gassensor mit integrierter Heizung. Der Hubmagnet verschließt eine Luftkammer in der sich der Gassensor befindet. Der Gassensor bildet den Kern der elektronischen Nase. Durch Variation der Sensorbeschichtung kann erreicht werden, dass der Sensor auf verschiedene Gase unterschiedlich sensitiv ist. Die Reaktion des Gases mit der Sensoroberfläche ist durch die Veränderung des Leitwertverhaltens des Gassensors messbar. Der Leitwert variiert auf Grund der Anlagerung von Gasmolekülen auf der Oberfläche. Für dieses Projekt werden Gassensoren verwendet, deren Oberfläche mit Zinndioxid beschichtet ist.

Um verschiedene Gasgemische unterscheiden zu können, wird der Gassensor zyklisch durch gezieltes Ansteuern der Sensorheizung aufgeheizt und abgekühlt. Parallel dazu wird der Leitwert des Gassensors gemessen. So kann ein Leitwert-Zeit/Temperatur-Profil erfasst werden. Durch eine Musteranalyse des kompletten Leitwert-Zeit-Profils kann eine Klassifikation des Gasgemisches [2] sowie anschließend eine Konzentrationsbestimmung durchgeführt werden. Da sowohl die Umgebungstemperatur als auch die Feuchtigkeit Einfluss auf die Analyse der Gase haben, werden diese über die beiden Fühler zusätzlich bestimmt.

Abbildung 1: Elektronische Nase

Den Aufbau eines typischen Messzyklus' mit den verschiedenen Aktivitäten zeigt Abbildung 2. Dieser Zyklus lässt sich sehr einfach in drei Phasen unterteilen. Die erste Phase beschreibt die ersten 15 Sekunden eines Zyklus und wird die Abkühlungsphase genannt. Die darauffolgenden 10 Sekunden bilden die Vorbereitungsphase und die letzten 40 Sekunden die Messphase**.**

Für die Steuerungssoftware des Sensors ergeben sich folgende Aufgaben:

- zyklische Ansteuerung des Hubmagneten zur Steuerung der einzelnen Phasen des Messzyklus,
- kontinuierliche Ansteuerung der Sensorheizung, zur Generierung eines Temperatur-Zeit-Profils (z.B. stetig ansteigende Sensortemperatur),

- kontinuierliche Erfassung des Sensorleitwertes,
- zyklische Erfassung der Umgebungstemperatur und –feuchte,
- Musteranalyse der Leitwert-Zeit-Profils am Ende oder bereits während eines Messzyklus',
- Kommunikation mit einem Sensor-Managementsystem (z.B. PC) zur zentralen Datenverarbeitung z.B. per TCP/IP

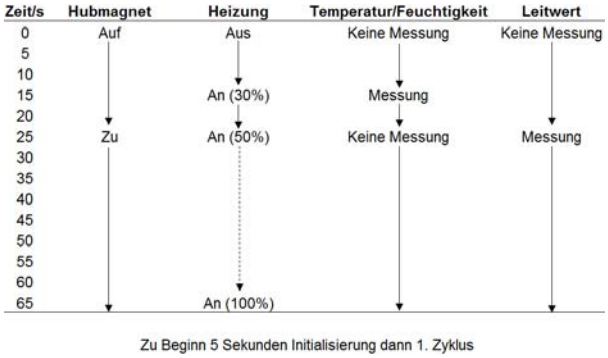

Abbildung 2: Nebenläufige Aktivitäten während eines Messzyklus'

Aus der Aufgabenstellung für die Steuerungssoftware wird ersichtlich, dass mehrere nebenläufige Aktivitäten realisiert werden müssen. Für diese Aktivitäten müssen feste Deadlines, die sich aus dem Aufbau eines Messzyklus ergeben, eingehalten werden. Es bietet sich daher an, die nebenläufigen Aktivitäten durch Ada-Tasks abzubilden. Zudem ist ein Datenaustausch zwischen einzelnen Tasks notwendig, bei dem die Konsistenz der Daten bei Schreib- und Lesezugriffen sichergestellt werden muss.

Als Hardware für die Steuerung kommt ein PHYTEC phyCORE Motorola MPC 565 zum Einsatz. Dieser bietet aufgrund seiner Leistungsfähigkeit die Möglichkeit, auch komplexe Verfahren zur Musteranalyse direkt auf dem µ-Controller zu implementieren. Zudem ist das Board mit einem Ethernet-Controller ausgestattet.

Architektur und Tasking-Modell

Um eine hohe Unabhängigkeit zwischen hardware- und anwendungsspezifischen Aspekten der Steuerungs-Software zu erreichen und damit auch eine hohe Wiederverwendbarkeit einzelner Programmteile zu erzielen, wurde ein Ebenenmodell mit 4 Ebenen gewählt (Abb.: 3).

Direkt auf der Hardware setzt das sog. Boardsupportpackage (BSP) auf, das von einer externen Firma bereit gestellt wurde. Es bietet grundlegende Funktionen zur Ansteuerung der einzelnen Hardware-Komponenten wie z.B. Analog-Digital-Umsetzer in Ada an. Jedoch ist für die Nutzung des BSP eine sehr gute Kenntnis der spezifischen Hardware notwendig, da durch das BSP noch keine wesentliche Abstraktion erfolgt.

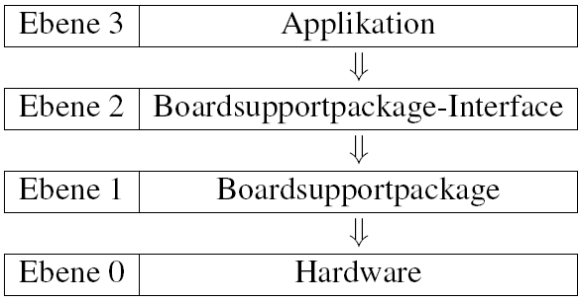

Abbildung 3: Ebenenarchitektur der Sensorsteuerung

Die auf dem BSP aufsetzende BSP-Interface-Schicht stellt die Verbindung zwischen der verstärkt hardware-spezifischen BSP-Schicht und der – idealerweise – hardware-unabhängigen Applikationsschicht her. Diese zusätzliche Ebene ist zum einen implementiert, um die Programmierung des µ-Controller zu vereinfachen und dem Applikationsentwickler leicht verständliche Funktionen beispielsweise zur Pulsweiten-Modulation und zur digitalen Ein- und Ausgabe anzubieten, sowie ihn möglichst von den technischen Details abzuschotten. Zum anderen erlaubt sie die Portierung von Applikationen auf andere Hardware-Plattformen mit den zugehörigen Boardsupportpackages. Bei einer Portierung wäre nur eine Anpassung der BSP-Interface-Schicht erforderlich.

Die Applikationsschicht nutzt die bereitgestellten Dienste der BSP-Interface-Schicht. In der Applikationsschicht wird die Nebenläufigkeit der verschiedenen Steuerungsaufgaben durch Ada-Tasks abgebildet. Für die Steuerung existieren die Tasks:

- Initialisierung (aller Komponenten),
- Leitwertmessung,
- Heizungssteuerung,
- Temperatur- und Feuchtigkeitsmessung sowie
- Hubmagnetsteuerung.

Gemäß dem Ada-Ravenscar-Profil kommt ein preemptives prioritätengesteuertes Scheduling zum Einsatz. Bei preemptiven Scheduling-Verfahren ist es möglich, dass die Abarbeitung eines Funktionsaufrufs der BSP-Interface-Schicht einer niederprioren Task durch eine höher priore Task unterbrochen wird. Dies würde bei der Analog-Digital-Umsetzung und beim digital E/A zu Fehlern führen (eine korrekte A/D-Wandlung darf nicht unterbrochen werden). Um dieses Problem zu verhindern, sind die Funktionen für die Nutzung des A/D-Wandlers bzw. der digitalen E/A jeweils durch ein protected object gekapselt. Als

Locking-Policy für gemeinsame Ressourcen kommt bei Ada-Ravenscar das Ceiling-Locking zum Einsatz. Als Ceiling-Priorität wird den protected objects in der BSP-Interface-Schicht standardmäßig die höchst mögliche Priorität zugewiesen. Genau genommen würde es genügen, als Ceiling-Priorität die maximale Priorität aller Tasks aus der Applikationsschicht zu wählen. Wichtig ist hier, wirklich alle Tasks zu berücksichtigen und nicht nur die, die auf das protected object zugreifen, da die A/D-Wandlung ja überhaupt nicht unterbrochen werden darf. Dies würde aber erfordern, dass der Applikationsentwickler die Ceiling-Prioritäten in der BSP-Interface-Schicht ändert, was durch die Schichten-Architektur nicht gewünscht ist. Durch Vergabe der maximal möglichen Priorität stellt sich diese Problematik nicht.

Ein Messzyklus der elektronischen Nase ist in drei verschiedene Phasen (Abkühlung, Vorbereitung, Messung) untergliedert. Der gesamte Messzyklus umfasst für den betrachteten Sensor 65 Sekunden. Abb. 4 zeigt die Aktivitäten der einzelnen Tasks während eines Messzyklus'. Aufgrund der Leistungsfähigkeit des µ-Controllers sind die maximalen Ausführungszeiten so klein, dass sie in der Abbildung kleiner als die minimale Linienstärke wären. Die Prozessorauslastung ist also sehr gering, was sich jedoch zukünftig durch die Erweiterung der Sensorsteuerung um eine Datenauswertungstask mit komplexen Musteranalyseverfahren und Remote-Anbindung ändern wird.

Für eine Prioritätenvergabe nach dem Rate Monotonic Scheduling (RMS) könnten alle Tasks vereinfacht als periodische Tasks mit konstanter

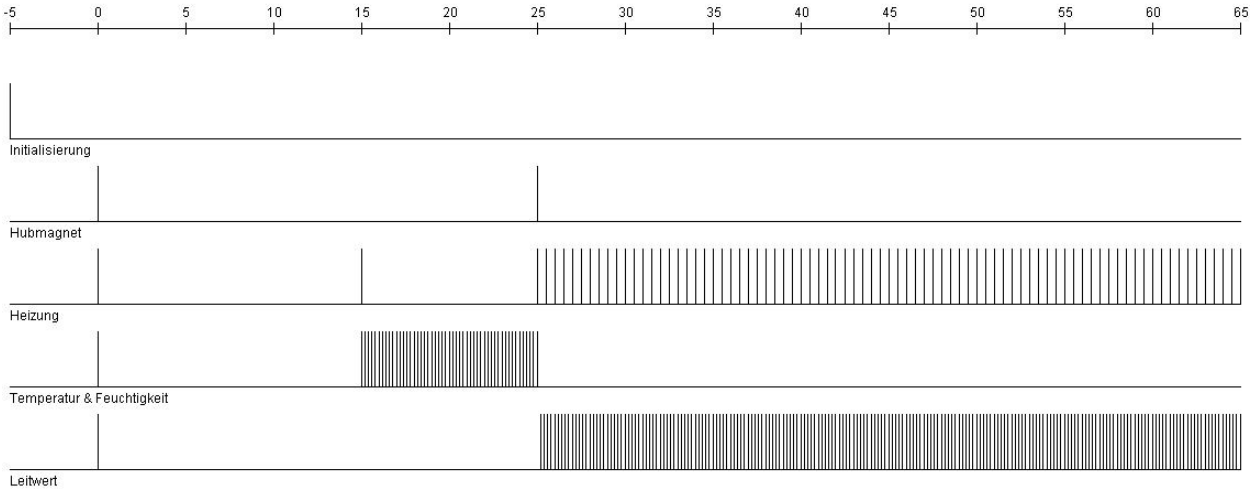

Abbildung 4: Aktivitäten der einzelnen Tasks während eines Messzyklus'

Nach der Initialisierung starten die Tasks Leitwertmessung, Heizungssteuerung, Temperatur- und Feuchtigkeitsmessung und Hubmagnetsteuerung ihren zyklischen Betrieb. In Ada wird dies jeweils durch ein Endlos-loop mit delay_until realisiert. Alle Zeitangaben, die für die Steuerung des Messzyklus benötigt werden, sind in einem gesonderten Paket abgelegt. Soll das Messverfahren (zeitlicher Ablauf) verändert werden, so müssen nur in diesem Paket Änderungen durchgeführt werden. Denkbar wäre später auch, diese Zeitkonstanten, und damit die Spezifikation des Messzyklus', beim Start per TCP/IP durch das Sensormanagementsystem zu empfangen.

Scheduling-Analyse

In der Basisrealisierung der Sensorsteuerung müssen bei der Scheduling-Analyse die vier zyklischen Tasks Leitwertmessung, Heizungssteuerung, Temperatur-/ Feuchtigkeitsmessung und Hubmagnetsteuerung berücksichtigt werden (Die Initialisierungstask wird nach Beenden der Initialisierung nicht wieder aktiviert.).

Periodendauer angesehen werden. Das RMS impliziert jedoch, dass die relative Deadline einer Task gleich der Periodendauer ist. Diese Voraussetzung ist hier nicht erfüllt: Z. B. soll die Ansteuerung der Heizung mit einer Periodendauer von 500 ms erfolgen. Es würde jedoch die Qualität des gewünschten Temperatur-Zeit-Profils negativ beeinflussen, wenn die Erhöhung einmal am Ende und einmal am Anfang einer Periode erfolgen würde. Aus technischer Sicht ist eine ausreichende Qualität der Steuerung für alle Tasks gewährleistet, wenn sie in jeder Periode ihre Aktivität innerhalb der in der nachfolgenden Tabelle angegebenen Deadline-Zeiten abgeschlossen haben.

Die Tabelle zeigt alle Tasks mit ihrer jeweiligen Periodendauern und maximalen Ausführungszeit pro Periode. Bei den maximalen Ausführungszeiten sind bereits die Blockierungszeiten, die sich bei Verwendung des Priority Ceiling Protokolls für gemeinsame Ressourcen ergeben, berücksichtigt. Weiterhin sind die relativen Deadlines bezogen auf den Beginn einer Periode aufgeführt; diese ergeben sich aus den technisch-physikalischen Anforderungen an den Sensor. Die letzte Spalte zeigt die Prioritäten, die sich

durch das Deadline Monotonic Verfahren ergeben. Eine Sonderstellung nimmt die (in der Tabelle nicht aufgeführte) Initialisierungstask ein: sie erhält die höchste Priorität 255, da sie vor dem ersten Start aller anderen Tasks abgeschlossen sein muss und deshalb nicht unterbrochen werden darf. (Durch eine delay-Anweisung am Anfang aller zyklischen Tasks wird sicher gestellt, dass die Initialisierungstask als erste arbeitet.)

Task-Name	Perioden-dauer T_i	max. Ausführungs-zeit C_i	Dead-line D_i	Prio-rität
Tempe-ratur/ Feuch-tigkeit	200 ms	0.06 ms	20 ms	254
Leitwert	200 ms	0.09 ms	20 ms	254
Heizung	500 ms	0.01 ms	50 ms	253
Hub-magnet	15000 ms	0.02 ms	100 ms	252

Um nachzuweisen, dass die Tasks unter allen Bedingungen ausführbar sind, kann der Rate Monotonic Schedulability Test [3] verwendet werden, wobei die Task-Perioden auf deren relative Deadlines verkleinert werden. Dabei handelt es sich jedoch um eine sehr pessimistische Abschätzung der Prozessor-Auslastung:

$$\sum_{i=1}^{n} \frac{C_i}{D_i} \leq n\left(2^{1/n} - 1\right)$$

Für die hier betrachtete Task-Menge mit n = 4 Tasks ergibt sich 0.0079 < 0.76. Die Einhaltung der Deadlines ist also garantiert.

Einsatz des Ravenscar-Profils

Die Entwicklung der Sensorsteuerungssoftware erfolgte zu einem großen Teil im Rahmen einer studentischen Arbeit. Der Einsatz des Sensorsystems war zunächst nicht für sicherheitskritische Anwendungen vorgesehen. Das Ada Ravenscar-Profil [4,5] wurde daher nicht in Hinblick auf bestimmte Zertifizierungen verwendet. Vielmehr war es das Ziel, Erfahrungen mit dem Ravenscar-Profil zu sammeln und gleichzeitig eine Software zu entwickeln, die erhöhten Zuverlässigkeitsanforderungen genügt.

Das Ravenscar-Profil legt Restriktionen für den Sprachumfang von Ada und für das Laufzeitsystem fest. Diese hatten bei der Entwicklung der Steuerungssoftware keine wesentlichen Einschränkungen zur Folge. Sie erhöhten im Gegenteil die Analysierbarkeit der Software deutlich.

Relevante Festlegungen im Ravenscar-Profil für die Sensorsteuerung und deren Konsequenzen sind:

- *FIFO_Within_Priorities, Ceiling_Locking*: Einfache Scheduling-Analyse zum Nachweis der Einhaltung aller Deadlines (vorhersagbares Verhalten).
- *No_Task_Allocators, No_Task_Hierarchy*: Alle Tasks der Sensorsteuerung stehen beim Programmstart fest und stehen auf einer Ebene.
- *Max_Task_Entries => 0*: Eine direkte Kommunikation zwischen den einzelnen Tasks (Rendezvous) wird für die Sensorsteuerung nicht benötigt; ein Datenaustausch kann über protected objects erfolgen.
- *No_Dynamic_Priorities*: Die Task- und Ceiling-Prioritäten werden statisch vergeben und ermöglichen den Nachweis wichtiger Eigenschaften wie Deadlock-Freiheit und Ausführbarkeit der Prozessmenge.
- *No_Relative_Delay*: Es werden nur delay_until-Statements benötigt.

Erweiterungen

Neben der grundsätzlichen Steuerung der Sensorhardware-Komponenten und dem Auslesen der Sensordaten soll die Software komplexe Datenauswertungen der Rohdaten direkt auf dem µ-Controller unterstützen. Für die Datenauswertung wird eine zusätzliche Task vorgesehen, die eine niedrigere Priorität als die Hardware-Steuerungs-Tasks zugewiesen bekommt. Dadurch wird der Messzyklus durch die Datenauswertung nicht beeinflusst. Die Auswertung soll immer nach Abschluss eines Messzyklus oder nach erfassen eines neuen Leitwert-Messwerts gestartet werden.

Die Bereitstellung dieser Daten für die Auswertung erfolgt über ein protected object, um immer einen konsistenten Datensatz durch einen gegenseitigen Ausschluss von Lese- und Schreibfunktionen zu garantieren. Für das Schreiben der Messdaten werden protected procedures „Write_Data“ genutzt, durch die die Mess-Tasks die Daten in das protected object schreiben. Das Lesen der Daten soll über einen protected entry „Read_Data“ realisiert werden, der eine Barrier „New_Data“ besitzt, über die der aufrufenden Auswerte-Task mitgeteilt werden kann, ob neue Daten vorliegen. Die Auswerte-Task wird dadurch also immer dann aktiviert, wenn neue Daten bereit stehen.

Zur Kommunikation wurde bereits eine TCP/IP-Task implementiert, wodurch eine einfache Kommunikation mit einem Server ermöglicht wird. Dies bietet die Grundlage zum Aufbau eines

Sensornetzes, welches weiterführende Auswertungen, wie z.B. Quellenlokalisierung umsetzen kann. Die konkreten Anforderungen an die Kommunikations-Task werden erst bei konkreten Einsatzzwecken bekannt.

Zusammenfassung

Im Rahmen einer Projektarbeit wurde eine Steuerungssoftware für ein intelligentes Sensorsystem (elektronische Nase) mittels Ada Raven entwickelt. Das dabei verwendete Ebenenmodell ermöglicht eine Abstraktion der Hardware für die Applikationsschicht, was eine komfortable Anwendungsentwicklung und gute Portierbarkeit auf andere Hardware-Plattformen gestattet. Die nebenläufigen Aktivitäten der Sensorsteuerung wurden auf ein preemptives prioritätengesteuertes Tasking-Modell abgebildet. Dies ermöglicht eine sehr übersichtliche und gut erweiterbare Softwarestruktur. Die Vergabe der Prioritäten erfolgt mit dem Deadline Monotonic Verfahren. Die Ausführbarkeit wird mit dem Rate Monotonic Scheduling Test nachgewiesen, wobei anstelle der Periodendauern die relativen Deadlines verwendet werden. Der Zugriff auf gemeinsame Ressourcen wie Daten oder A/D-Wandler wird über protected objects realisiert, denen Prioritäten nach dem Priority Ceiling Protokoll zugeordnet werden. Die Restriktionen, die sich aus der Verwendung des Ravenscar-Profils ergeben, schränken die Entwicklung nur unwesentlich ein, erleichtern jedoch die Analyse des Systems wesentlich und ermöglichen so eine Garantierung von Dead-Lock-Freiheit und Einhaltung der Deadlines der einzelnen Tasks. Dies bildet die Basis für die Anwendung der Sensorsteuerung unter sicherheitskritischen Anforderungen für den industriellen Dauereinsatz.

Literatur

[1] Armin Jerger, Heinz Kohler (Fachhochschule Karlsruhe), Hubert Keller, Rolf Seifert (Forschungszen-trum Karlsruhe): "A Novel and Economic Sensor System for Monitoring of Ammonia", Field Screening Europe 2001, Karlsruhe May 14 - 16, 2001

[2] K. Frank, V. Magapu, V. Schindler, H. Kohler, H. B. Keller, and R. Seifert: "Chemical Analysis with Tin Oxide Gas Sensors: Choice of Additives, Method of Operation and Analysis of Numerical Signal" Sensor Lett. 6, 908–911 (2008)

[3] Buttazzo, G. C.: "Hard real-time computing systems" Kluwer Academic Publishers Boston, Dordrecht, London (1997)

[4] Burns, A.; Dobbing, B., Vardanega T.: “Guide for the use of the Ada Ravenscar Profile in high integrity systems” , Technical Report YCS-2003-348, University of York (2003)

[5] Keller, Hubert B.; Matthes, Jörg; Kapitel: Ada, Handbuch Programmiersprachen; Hrsg. Henning, Peter; Vogelsang, Holger; Carl Hanser Verlag München (2006)