PEARL-News

Ausgabe 2/2008

Mitteilungen
der GI-Fachgruppe Echtzeitsysteme

ISSN 1437-5966

Impressum

Herausgeber	GI-Fachgruppe Echtzeitsysteme (RT) URL: http://www.real-time.de
Sprecher	Dr. P. Holleczek Universität Erlangen-Nürnberg, Regionales Rechenzentrum Martensstraße 1, D-91058 Erlangen Telefon: 09131/85-27817 Telefax: 09131/30 29 41 E-Mail: peter.holleczek@rrze.uni-erlangen.de
Stellvertreter	Prof. Dr. Dr. W. Halang FernUniversität in Hagen Universitätsstraße 27 - PRG D-58084 Hagen Telefon: 02331/987-372 Telefax: 02331/987-375 E-Mail: wolfgang.halang@fernuni-hagen.de
Redaktion	Prof. Dr. R. Müller FH Furtwangen, Fachbereich Computer- & Electrical Engineering Robert-Gerwig-Platz 1, 78120 Furtwangen Telefon: 07723/920-2416 Telefax: 07723/920-2610 E-Mail: mueller@hs-furtwangen.de
ISSN	1437-5966

Redaktionell abgeschlossen am 15. November 2008

Einreichung von Beiträgen
Diese Zeitschrift soll nicht nur Mitteilungsblatt sein, sondern auch eine Plattform für den Informations- und Meinungsaustausch zwischen allen an den Fragen der Echtzeitprogrammierung Interessierten bilden. Diskussionsstoff bzw. offene Fragen gibt es auf unserem Gebiet reichlich.

Wir möchten Sie, liebe Leserinnen und Leser, daher ausdrücklich ermuntern, auch in Zukunft die PEARL-News durch Ihre Beiträge mit zu gestalten. Für ein ausgewogenes Bild der News sollten Beiträge nicht länger als 5 Seiten sein.

Der Titel „PEARL-News“ wird trotz geändertem Fachgruppennamen aus rein traditionellen Gründen beibehalten.

Rainer Müller (Furtwangen)

Inhaltsverzeichnis

1 Organisation der Fachgruppe

Der Fachbereich (FB) Technische Informatik (TI) der Gesellschaft für Informatik hat in seiner Sitzung vom 25.2.2008 beschlossen, die Fachgruppe (FG) Echtzeitsysteme als Fachausschuß (FA) aufzunehmen. Ein Grund dafür liegt darin, dass „Echtzeitsysteme“ zwar mit den anderen FAs gleichermaßen etwas zu tun haben, aber keinem direkt zuzuordnen sind. Das ist insofern eine Ausnahmekonstruktion, als wir als FG persönliche Mitglieder haben. Wir werden also innerhalb des FB Technische Informatik als FA geführt. Auf der GI-Homepage werden wir als FA/FG bezeichnet. Als Kürzel gilt „RT“ für Real-Time.

In der weiteren Darstellung wird zur leichtere Lesbarkeit nur von der Fachgruppe gesprochen. Auf der Sitzung der Fachgruppenleitung am 14.5.2008 wurde die anfallende Arbeit auf mehrere Schultern verteilt. Damit alle Mitglieder schnell und direkt die jeweils zuständigen Personen erreichen erfolgt hier eine kurze Darstellung des Aufgabengebietes zusammen mit der Kontakadresse.

Dr. Holleczek hat weiterhin die Rolle des Sprechers der Fachgruppe inne.
Kontakt: `peter.holleczek@rrze.uni-erlangen.de`

Prof. Halang übernimmt die Rolle des Stellvertreters.
Kontakt: `wolfgang.halang@fernuni-hagen.de`

Prof. Zöbel hilft mit bei der Organisation des jährlichen Workshops in Boppard. Darübe hinaus ist er der Schriftführer der Fachgruppe.
Kontakt: `zoebel@uni-koblenz.de`

Prof. Schiedermeier übernimmt den Bereich Öffentlichkeitsarbeit. Dazu zählt auch die Wartung der Homepage http://www.real-time.de und Verwaltung der Mailingliste.
Kontakt: `gschied@fh-landshut.de`

Prof. Benra übernimmt die Organisation der Wettbewerbe.
Kontakt: `benra@fbe.fh-wilhelmshaven.de`

Prof. Heitmann übernimmt die Organisation des Arbeitskreises Echtzeit-Netze
Kontakt: `heitmann@informatik.haw-hamburg.de`

Prof. Gumzej ist nun Leiter Arbeitskreise Ausbildung und Sprachpflege
Kontakt: `roman.gumzej@uni-mb.si`

Prof. Müller ist wieder Herausgeber der News
Kontakt: `mueller@hs-furtwangen.de`

2 Neuer Arbeitskreis „Echtzeitfähige Kommunikationssysteme“

Innerhalb der GI-Fachgruppe „Echtzeitsysteme“ soll ein neuer Arbeitskreis entstehen. Das erste Treffen dieses Arbeitskreises ist im Rahmen des diesjährigen Workshops „Echtzeit 2008“ in Boppard geplant.

2.1 Motivation

Bei der Festlegung der Internetprotokolle wurden Forderungen bezüglich eines deterministischen Zeitverhaltens wenig berücksichtigt. Durch die starke Zunahme der Internet-basierten Multimediaanwendungen (z. B. VoIP und Video-on-Demand) werden aber Aspekte der Echtzeitfähigkeit des Internets immer wichtiger. Dies spiegelt sich wieder in der Entwicklung etlicher neuer Protokolle. So wurde z. B. in IPv6 das Konzept der „flows“ eingeführt, von dem heute aber noch nicht klar ist, wie es in der Praxis umgesetzt werden soll.

Ein weiterer Aspekt ist die Unterstützung von mobilen Internetanwendungen. Neben dem Zeitverhalten der eigentlichen funkbasierten Übertragung ist hier ein schnelles Roaming zwischen den Funkzellen notwendig.

Auf der anderen Seite ist in der Industrie ein aktueller Trend die Verdrängung der klassischen Feldbusse durch echtzeitfähiges Ethernet. In der Automatisierungstechnik sind vielerlei Methoden bekannt, wie man die Echtzeitproblematik im klassischen Ethernet umgehen kann. Daraus hat sich eine Vielzahl von Standards entwickelt, die heute von diversen Herstellern kommerziell angeboten werden (siehe www.real-time-ethernet.de). Mittlerweile werden Zykluszeiten unter $25\mu s$ mit einem Jitter kleiner als $1\mu s$ erreicht. Damit können sogar Antriebe direkt über Ethernet angesteuert werden.

Allgemein ist zu beobachten, dass Netzwerke auch verstärkt in eingebettete Systeme eingesetzt werden. Eine wichtige Rolle spielt dabei die Fahrzeugtechnik und der Flugzeugbau. In neueren Flugzeugen werden Ethernet-basierte Komponenten für die Kommunikation zwischen den Systemen innerhalb des Flugzeuges eingesetzt (AFDX). Auch die Autoindustrie denkt verstärkt über den Einsatz von Ethernet nach. Möglich wird dieser Trend durch die Verfügbarkeit sehr preiswerter Mikrocontroller mit integrierten Netzwerkschnittstellen.

Die Durchdringung des täglichen Lebens mit Funkanwendungen ist allgegenwärtig. Es haben sich viele Standards etabliert (WLan, Bluetooth, ZigBee), die sich verstärkt in der Industrie durchsetzen. Zunehmend werden auch hier Anwendungen entwickelt, die ein deterministisches Zeitverhalten erfordern. Erwähnt werden sollen in diesem Zusammenhang die verschiedenen Initiativen zum Aufbau der Car-to-Car-Kommunikation. Wesentliches Ziel hierbei ist es, durch Informationsaustausch zwischen den Autos eine Verbesserung der Sicherheit im Straßenverkehr zu erreichen. Dies gelingt aber nur dann, wenn Protokolle zur Verfügung stehen, die in Echtzeit den Aufbau eines Ad-hoc-Netzwerkes zwischen den Autos ermöglichen und die Kommunikation unter Einhaltung strenger Zeitvorgaben durchführen.

2.2 Zielsetzung und Themenvorschläge

Die oben aufgeführten Beispiele zeigen ein breites Spektrum von Themen, die sich im Bereich der Echtzeitkommunikation ergeben. Eine erste Aufgabe des Arbeitskreis ist es sicherlich, diese Themenbereiche zu sondieren und einen Schwerpunkt der eigenen Arbeit zu definieren. Allgemein können folgende Ziele des Arbeitskreises formuliert werden:

- Forum zum Austausch aktueller Forschungsergebnisse
- Förderung von Aktivitäten im Bereich vernetzter Echtzeitanwendungen
- Diskussion der Sicherheit und Zuverlässigkeit

Prof. Dr. Hans Heinrich Heitmann
Hochschule für Angewandte Wissenschaften
Fakultät Technik und Informatik - Department Informatik
Berliner Tor 7
20099 Hamburg
Telefon: 040 42875-8153
EMail: heitmann@informatik.haw-hamburg.de

3 Preisträger 2008

Die Fachgruppe Echtzeitsysteme veranstaltet seit 2007 einen Wettbewerb für Nachwuchsarbeiten im Bereich Echtzeitsysteme. Die Preisverleihung findet im Rahmen des diesjährigen Workshops „Echtzeit 2008“ statt. Die Preisträger stellen ihre Arbeiten auch in der Frühsession am 28.November vor.

Die diesjährigen Preisträger sind:

Andreas Hollmann wurde für seine Diplomarbeit **„Virtualization Concepts for AIDASS/MaTE“** an der Fachhochschule Landhut prämiert.

Abstract:
In this thesis various virtualization approaches are analyzed in order to figure out the best suited solution for the real-time application area and particularly for the usage with AIDASS/MaTE.
Throughout this paper a great number of available virtualization techniques are introduced and explained to the reader. This work focuses particularly on the preservation of real-time requirements in virtualized environments.
Accurate timer devices are of vital importance for real-time systems. The timer subsystem of modern x86 systems is, therefore, analyzed and two fine grained timer devices are selected for the task of replacing PowerPC's timer hardware used by AIDASS.
Afterwards five preselected virtualization solutions are presented in detail. All solutions are assessed and eventually one is selected as the preferred solution. Further investigation of the selected product is done in the following chapter.
A short overview of the installation and configuration of the selected Hypervisor is shown. Several VxWorks tests are explained and some certain limitations of recent x86 machines are discussed. Additionally, some x86 system characteristics and new technologies are introduced.
Finally, a number of important points that need attention when porting PowerPC AIDASS to x86 are stated in the last chapter.

Andrey Kolesnikow wurde für seine Diplomarbeit **„Konzeption und Entwicklung eines echtzeitfähigen Lastgenerators für Multimedia-Verkehrsströme in IP-basierten Rechnernetzen“** an der Universität Hamburg prämiert.

Abstract:
Experimente zur Leistungsanalyse von dienstintegrierten Rechnernetzen sind stets unter Last durchzuführen. Der Einsatz von künstlichen (synthetischen) Lasten bringt hier signifikante Vorteile. Demzufolge wächst auch der Bedarf nach entsprechenden geeigneten Spezialwerkzeugen zur Lastgenerierung in existierenden Netzen. In der vorliegenden Arbeit wird eine Architektur für einen echtzeitfähigen Lastgenerator basierend auf einer dedizierten Lastspezifikationstechnik skizziert und einige zu berücksichtigende Realisierungsaspekte werden aufgezeigt. Ausgehend von einem existierenden Prototyp eines Lastgenerators (UniLoG) wird eine echtzeitfähige Version des bestehenden Lastgenerierungswerkzeuges erstellt, die ein breites Anwendungsspektrum besitzt. Der realisierte Lastgenerator wird überdies auf seine Realitätsnähe hin überprüft.

Sven Bünte fertigte seine Masterthesis der Universität des Saarlandes in Saarbrücken mit dem Thema **„Fault Tolerance Analysis on the FlexRay Startup Procedure Using Model Checking“** an.

Abstract:
The FlexRay protocol is intended to facilitate safety-critical, hard real-time applications in the automotive domain. Fault tolerance and time-triggered communication are key features to assure an adequate degree of dependability. FlexRay's wakeup and startup procedures aim at establishing a consistent, synchronized slot schedule with respect to a TDMA methodology. To both algorithms I will apply the model checker SPIN in order to derive properties about upper bounds on execution times in the presence of certain failures. It turns out that central bus guardian devices are essential, as well as the configuration of FlexRay nodes. Furthermore, failure scenarios are identified which prevent a system to synchronize. It is also shown that the host application design affects the ability to tolerate faults drastically.

4 Zeitmessungen unter Linux

4.1 Motivation

Linuxsysteme werden immer häufiger in Echtzeitanwendungen eingesetzt. Dabei ist bekannt, dass Linux aus Prizip nicht echtzeitfähig ist. Allerdings wird heutzutage der Begriff Echtzeit in allen möglichen Zusammenhängen verwendet, bei denen die die Zeitanforderungen recht moderat sind. Eine etwas ältere Untersuchung zur Eignung von Linux für zeitkritische Anwendungen findet sich in [1].

In dieser Arbeit geht es um die Messung der Zeitstabilität von einer einfachen Programmverzögerung. Genauigkeit der Verzögerung, wie auch deren Verläßlichkeit sind notwendige Vorkenntnisse, um eine die Abschätzung über die Einsatzbarkeit des Opensourcebetriebssystems überhaupt machen zu können.

In diesem Beitrag werden erste Messergebnisse zur Zeitstabilität von Linux vorgestellt. Die Möglichkeiten von verfügbaren Erweiterungen des Standardlinuxkernels sind Ziel eines anderen Artikels.

4.2 Zeitmessung

Die Ausstattung von PCs mit Standardschnittstellen wird zwar immer dünner, jedoch findet sich zumindest an Desktopsystemen meist noch ein paralleler Druckerport. Diese Schnittstelle ist von Prozessor aus mit direkten I/O-Befehlen erreichbar und kann auf diese Weise als einfache Digital-E/A angesehen werden. Die Messung der Zeit erfolgt mit einer einfachen quarzgetakten Zählerschaltung (Taktfrequenz $80MHz$) mit 24Bit. Dies liefert eine theoretische Zeitauflösung von $12,5ns$. Die Zählerstufe kann über die Steuerleitungen *gelöscht*, *gestartet* und *gestoppt* werden. Der Zählerwert wird über die Datenleitungen des Parallelports ausgelesen. Damit stellt diese Schaltung einen **T**ime to **D**igital **C**onverter (TDC) dar. Für diese Arbeit wurde der TDC in seiner Auflösung softwaremäßig herabgesetzt, sodass lediglich eine Auflösung von $1\nu s$ benutzt wurde.

4.3 Testprogramm

Als einfachste Anwendung wird hier die Systemfunktion `usleep` herangezogen, die eine Zeitverzögerung in Einheiten von Mikrosekunden zur Verfügung stellen soll. Die Ziele der Messung sind nun:

1. Die Überprüfung der Einhaltung der vorgegebenen Zeit
2. Die Überprüfung der Stabilität der Zeiten, wobei für Echtzeitanwendungen die größten Abweichungen von der Sollzeit von Bedeutung sind.

Das Testprogramm ist sehr einfach (vgl. Abb. 1).

```
for(i=0;i<zahl_der_messungen; i++)
{
   reset_tdc();               /* loeschen des TDCs           */
   start_tdc();               /* start der Zeitmessung       */
   usleep(delay);             /* <<< zu testende Funktion    */
   stop_tdc();                /* stop der Zeitmessung        */
   duration = read_tdc();     /* lese vergangene Zeit        */
   inc_histo(duration);       /* erhoehe Kanal im Histogramm */
}
```

Abbildung 1: Hauptteil des Messprogramms. Der Wert von `delay` ist für jeden Programmlauf konstant.

Seit zumindest Kernelversion 2.2 stehen als weitere Schudulingpolicy für *soft real time scheduling* die Verfahren `SCHED_RR` und `SCHED_PRIO` zur Verfügung. Diese beiden Scheduler arbeiten auf einer Prioritätsebene oberhalb *aller* normalen Linuxprozesse. Über eine Option kann das Meßprogramm als normaler Linuxprozess laufen, oder einem anderen Schedulingverfahren zugeordnet werden. Es ist zu erwarten, dass die Verlässlichkeit einer zeitlichen Verzögerung mit diesen Schedulern dann deutlich besser wird.

4.4 Messergebnisse

Die Messungen wurden an einem älteren PC durchgeführt (vgl. Tab. 2). Dabei wurden jeweils die Messungen im unbelastetem Betrieb gemacht und mit einer Belastung, bei der 3 Prozessen jeweils in einer Endlosschleife CPU-Zeit verbrauchen.

Die Einhaltung der Zeitvorgabe erwiest sich als recht unzuverlässig (vgl. Tabelle 1). Für die Messungen wurden jeweils 10^6 Schleifendurchläufe verfolgt. Für die Berechnung des Mittelwertes wurden nur die die Werte berücksichtigt, die in einem $4ms$ breiten Fenster um den häufigsten Wert lagen. Hier wurde beobachtet, dass bis zu 6% der Verzögerungen außerhalb dieses Intervalls lagen. Wesentliche Unterschiede zwischen dem „belasteten“ und dem unbelastetem System wurden nicht beobachtet. Die Kalibrierung der Messwerte wurde mit einem Oszilloskop überprüft.

Die Verzögerungen sind offensichtlich nur in Vielfachen von $4ms$ möglich. Dies rührt vom internen Zeitsystem des Linuxkernels her. Die Funktion `usleep` liefert keinerlei Informationen an den User, dass die Zeiten nicht korrekt eingehalten wurden.

System-	gewünschte	gemessene Verzögerung			im $\pm 2ms$
zustand	Verzögerung	min	max	Mittelwert	Bereich
idle	$4\mu s$	$69\mu s$	$40.245ms$	$3.968ms$	99.5%
belastet	$4\mu s$	$71\mu s$	$40.033ms$	$3.970ms$	99.5%
idle	$3ms$	$3.061ms$	$64.043ms$	$3.978ms$	98.8%
belastet	$3ms$	$3.047ms$	$40.033ms$	$3.979ms$	98.9%
idle	$4ms$	$4.054ms$	$44.095ms$	$7.971ms$	96.1%
belastet	$4ms$	$4.085ms$	$40.025ms$	$7.978ms$	99.9%
idle	$7.9ms$	$7.926ms$	$44.078ms$	$7.978ms$	93.8%
belastet	$7.9ms$	$7.922ms$	$41.764ms$	$7.978ms$	94.4%
idle	$8.0ms$	$10.005ms$	$51.543ms$	$11,971ms$	99.4%
belastet	$8.0ms$	$8.047ms$	$71.686ms$	$11,972ms$	99.4%
idle	$50.0ms$	$50.487ms$	$141.488ms$	$51.980ms$	99.9%
belastet	$50.0ms$	$50.096ms$	$168.440ms$	$51.981ms$	99.9%

Tabelle 1: Abweichungen der Istverzögerung vom Soll. Die Funktion `usleep` arbeitet in Vielfachen von $4ms$. Die Sollzeiten werden im Mittel stets übertroffen.

Die beobachteten Werte stimmen insofern mit den Zielvorgaben von Linux überein, dass die Sollzeit nie unterschritten wurde. Jedoch sind die Mittelwerte meist in der Nähe des nächsthöheren Vielfachen von $4ms$. Bei kurzen Verzögerungszeiten fällt auf, dass die Zuverlässigkeit der Zeiteinhaltung geringer ist, als bei größeren Verzögerungszeiten. Allerdings treten in allen Messungen starke Abweichungen nach oben auf. Diese sind nicht weiter verwunderlich, da Linux kein Echtzeitbetriebssystem ist. Linux verfolgt die

Strategie, dass jeder Prozess Rechenzeit zur Verfügung gestellt bekommt. Systemprozesse kommen auf diese Weise vereinzelt zum Zuge und verhindern so die korrekte Abarbeitung dieser Messanwendung.

Bei einer genaueren Analyse, der Verteilungen (vgl. Abb. 2 und 3) treten seltsame Häufungen auf, die erst bei einer großen Spreitzung der Darstellung sichtbar werden. Sowohl bei einer Verzögerungvon $8ms$, wie auch bei $50ms$ treten Nebenpeaks im Abstand von $100\mu s$ auf beiden Seiten der häufigsten Verzögerung auf. Die Anhäufung bei der größeren Zeit ist evtl. noch durch die Behandlung von Interrupts erklärbar. Jedoch ist die Anhäufung bei den kürzeren Verzögerungszeiten noch nicht geklärt.

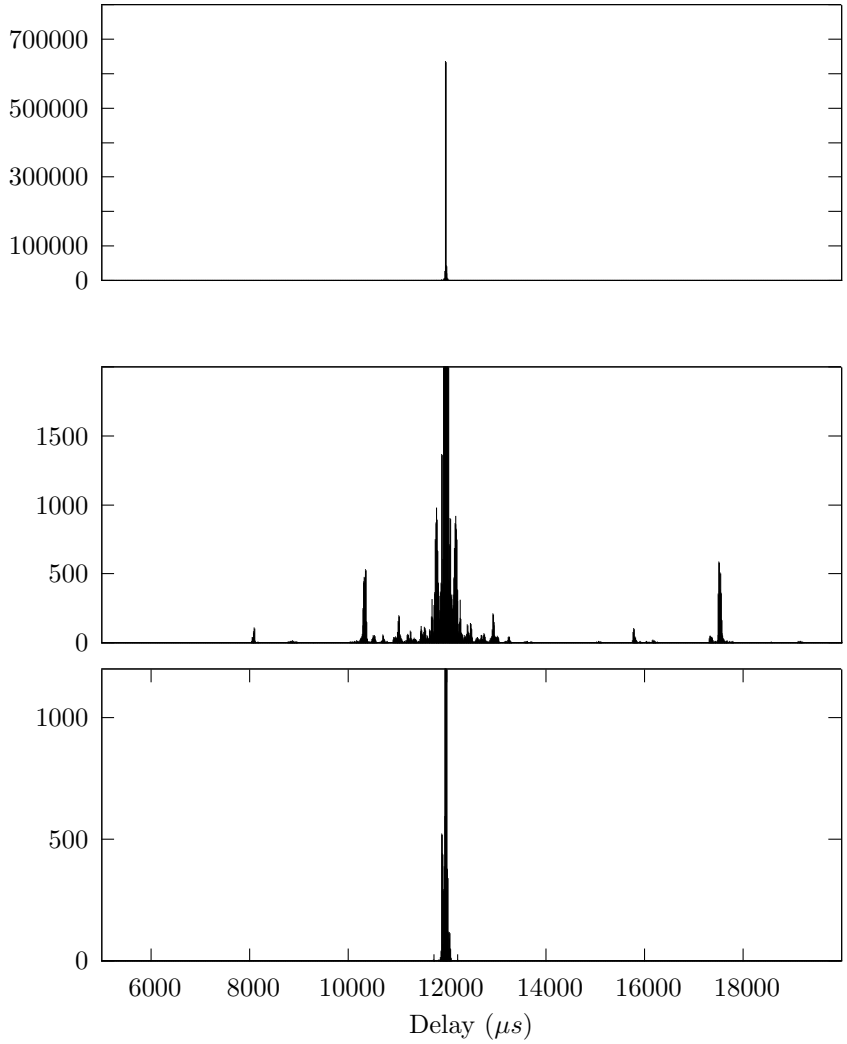

Abbildung 2: Verteilung der Verzögerungszeiten bei `usleep(8000)`. Die y-Skala wurde für die unteren beiden Darstellungen gespreitzt, sodass auch seltene Abweichungen erkennbar sind. Oben und Mitte: Normales Scheduling; unten: Round-Robin-Scheduling

Bei Wechsel des Schedulers auf das Verfahren `SCHED_RR` (vgl. Abb. 2 und 3 jeweils unten) fällt sofort auf, dass die Streuung der Verzögerungen deutlich geringer wird. Die Streubreite der Werte sinkt auf ein Intervall von $2.4ms$ (vgl $118ms$ in Tab. 1 bei normalem Scheduling) bei der Messung mit $50ms$ und sogar auf $0.5ms$ bei der Messung mit $8ms$ Verzögerungszeit. Eine verläßliche Aussage über eine Obergrenzen der Verzögerung ist dies aber sicherlich nicht.

4.5 Ergebnis

Diese Messungen geben nur einen groben Hinweis auf die Verlässichkeit der Zeitangaben in einem Standardlinuxsystem. Aussagen, welche Anforderungen erfüllbar sind, wenn ein einziges Verfehlen schon zum Systemfehler führt, sind über diese Methodik sicherlich nicht erreichbar.

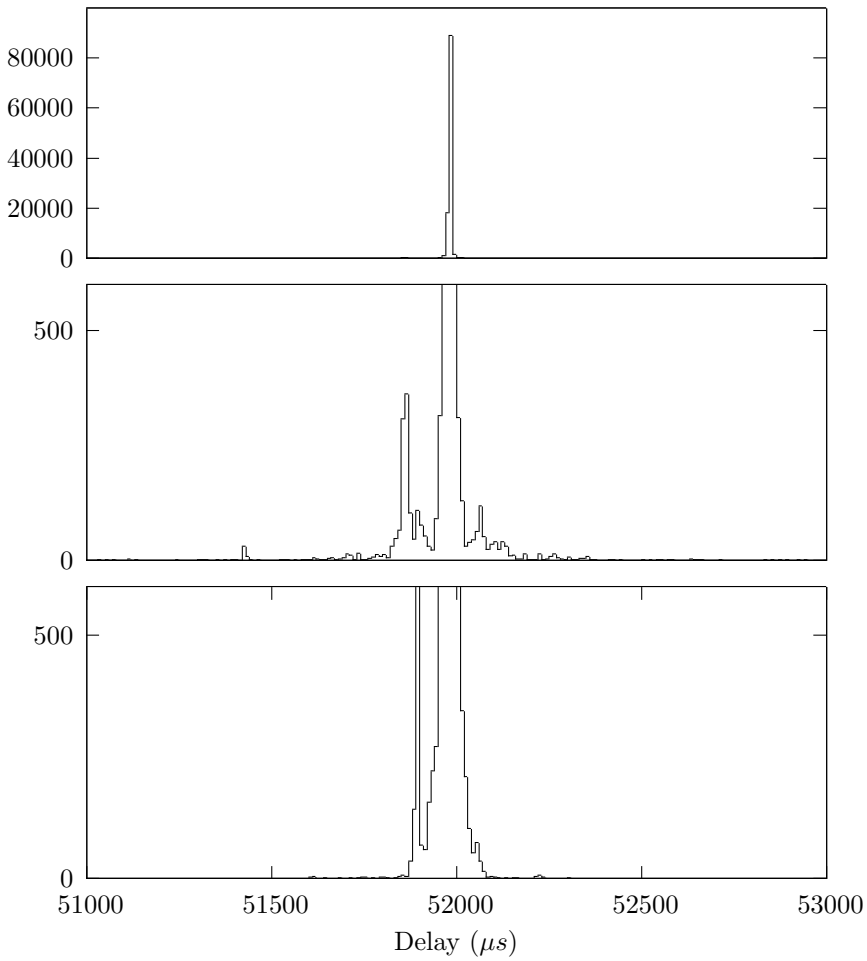

Abbildung 3: Verteilung der Verzögerungszeiten bei `usleep(50000)`. Die y-Skala wurde für die unteren beiden Darstellungen gespreitzt, sodass auch seltene Abweichungen erkennbar sind. Oben und Mitte: Normales Scheduling; unten: Round-Robin-Scheduling

Der Round-Robin-Scheduler ermöglicht ohne weitere Systemarbeiten schon deutlich bessere Reaktionszeiten.

Die Beeinflussung der Messergebisse durch eine hohe Interruptlast ist noch zu überprüfen.

Es gibt noch einige Parameter, mit denen sich das Zeitverhalten steuern läßt. Seit Kernel 2.6 ist die interne Zeitbasis auf $1ms$ umgestellt worden [2]. Sonst wären die oben beschriebene Werte bei normalem Linuxscheduling auch gar nicht erzielbar gewesen. Es wäre interessant, welche Verbesserung sich ergibt, wenn diese Auflösung noch verfeinert wird. Es gibt die Möglichkeiten mit Realtimeerweiterungen wie *Xenomai*, *RTAI* oder *RTLinux* die Ergebnisse zu verbessern.

Diese Punkte sind Ziel der weiteren Untersuchungen.

Literatur

[1] *H. Rzehak, A. Heursch:* Die Eignung von Linux für zeitkritische Anwendungen; in PEARL2000; Springer Verlag (2000)

[2] http://www.pronix.de/pronix-187.html

CPU	INTEL Celeron
Takt	1 GHz
Speicherausbau	1GB (128kB Cache)
Betriebssystem	Linux (Kernel 2.6.25-26-default)
BIOS	Award

Tabelle 2: Eigenschaften des eingesetzten PCs. Es wurden keinerlei Optimierungen im Bereich Cache, usw. durchgeführt.

Rainer Müller
Hochschule Furtwangen
mueller@hs-furtwangen.de

5 Studentische Wettbewerbe im Echtzeit-Umfeld

Abstract

Bekanntlich wirken studentische Wettbewerbe motivierend auf die Studierenden und bilden ein geeignetes Mittel, um während des Studiums Projekt- und Teamerfahrung, Entwurfsmethoden und verschiedene inhaltliche Aspekte zu vermitteln. Wir berichten in diesem Beitrag von studentischen Wettbewerben, an denen im Rahmen des Bachelor- und Masterstudiums Informatik an der HAW Hamburg teilgenommen wird. Dabei kommen die Studierenden insbesondere mit Problemstellungen in Echtzeitsystem in Kontakt.

5.1 Forschungsschwerpunkt FAUST

Die studentischen Wettbewerbe, die in diesem Bericht beschrieben werden, finden statt im Rahmen des Forschungsschwerpunkts FAUST des Departments Informatik der HAW Hamburg [5]. Der Forschungsschwerpunkt fasst mehrere Teilprojekte mit speziellen Aufgabenstellungen zusammen, die unter dem Leitthema Fahrerassistenz- und autonome Systeme stehen. Die erklärten Ziele des Forschungsschwerpunkts FAUST sind die Entwicklung von Industrie- und forschungsrelevanten Technologien und deren Anwendung. Die hochschulinternen Projekte sollen möglichst realitätsnah das spätere Arbeitsumfeld der Studierenden wiedergeben. Entwicklungsbeiträge sollen im Bachelorstudium erarbeitet und im konsekutiven Master fortgesetzt und vertieft werden. Unterschiedliche Spezialgebiete der Professoren sollen in gemeinsamen Projekten zur Anwendung kommen und sich ergänzen.

Die entwickelten Technologien werden auch in Firmenkooperationen z.B. mit STILL und Helms Technologie eingebracht.

5.2 Der Carolo-Cup der TU Braunschweig

Bereits im Februar 2008 hat das Team FAUST der HAW Hamburg am damals ersten Carolo-Cup Wettbewerb der TU Braunschweig teilgenommen. Information und das Reglement zum Carolo-Cup findet man auf der Webseite [1]. Der Wettbewerb soll jährlich fortgeführt werden und findet im Vorfeld des jährlichen AAET Symposiums in Braunschweig im Februar statt. Einen Bericht zum Ausgang des Wettbewerbs 2008 findet man in [3].

Die Aufgabe besteht in der Entwicklung eines autonomen Modellfahrzeugs, welches einer Landstraßenähnlichen Fahrspur folgen muss. Die Fahrten finden einzeln ohne „Verkehr“, jedoch mit stationären Hindernissen statt. Zudem müssen die Fahrzeuge autonom einparken. Neben den dynamischen Disziplinen werden auch Konzepte zur Herstellung, Spurführung, Ausweichen von Hindernissen und Einparken bewertet. Dazu tragen die Studierenden in einem Fachvortrag vor hochwertig besetzter Juri ihre Konzepte vor.

Im Februar 2008 konnten die Studierenden der HAW Hamburg mit ihrem Team FAUST die besten Konzepte für Spurführung mit Ausweichen sowie Einparken vorweisen. In der Gesamtwertung kamen sie auf Platz 4 von 6 teilnehmenden Teams. Das autonome Modellfahrzeug selbst wurde von der TU Braunschweig geliehen. Es war bereits regelkonform mit Sensorik und Prozessor ausgestattet. Für den Wettbewerb im Februar 2009 hat das Team FAUST im vergangenen Jahr ein eigenes Fahrzeug entwickelt. Voraussichtlich wird die HAW Hamburg dann sogar mit zwei Teams antreten: mit dem eigenen Fahrzeug und mit dem geliehenen Fahrzeug. Inzwischen beschäftigen sich gut 16 Studierende in Form von studienbegleitender Tätigkeit und Abschlussarbeiten im Bachelor- und Masterstudiengang mit den Technolgien rund um die beiden Fahrzeuge.

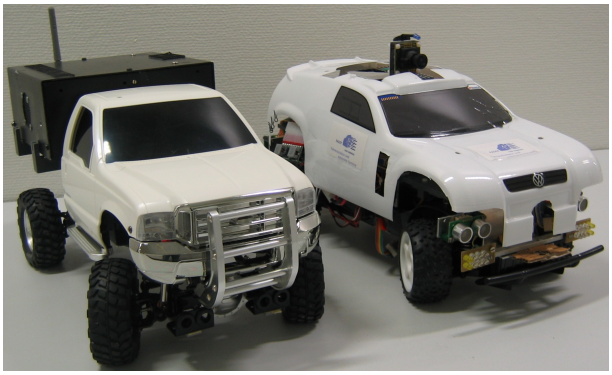

Abbildung 4: Die Fahrzeuge für den Carolo-Cup, links die Eigenentwicklung der HAW Hamburg, jedoch ohne Netbook und noch ohne Kamera.

Das Hardware-Konzept des neuen eigenen Fahrzeugs basiert auf einem Mini-Laptop (Netbook) mit 8,9 Zoll Display und Intel Atom 1,6 GHZ Prozessor. Die Rechner werden mit Linux oder Windows geliefert. Wir haben uns für Linux entschieden, insbesondere weil diese Rechner keine Festplatte haben, sondern mit einer 8GByte SSD ohne bewegliche Teile arbeiten. Die Anbindung an die Sensorik und Aktorik geschieht über USB an einen ARM-Prozessor mit IO-Board. Alle Komponenten sind kurzfristig verfügbar und preiswert. Die Studierenden sind mit der ARM-Architektur bereits durch die Grundausbildung im Studiengang Technische Informatik vertraut. Das Netbook hat gegenüber einem PC104 des Leihfahrzeugs aus Braunschweig den Vorteil, dass Stromversorgung, USB-Anschluss und Bildschirm in einem Gehäuse integriert sind. Durch den Bildschirm und die Tastatur hat man auf dem Netbook auch die Möglichkeit ohne WLAN-Anbindung direkt zu debuggen oder Konfigurationsdaten zu editieren. Bildverarbeitungsalgorithmen können durch die eingebaute Kamera (die jedoch im Einsatz nicht genutzt wird) im Netbook unmittelbar getestet werden.

Inhaltlich arbeiten die Studierenden zwar auch an der Lösung der konkreten Anforderung durch den Wettbewerb. Die Thematiken gehen jedoch über diese Anforderungen hinaus. Es wird an der Erzeugung von Umgebungskarten gearbeitet, die auf Bild- und odometrischen Daten basieren. Hierzu werden aktuelle SLAM-Verfahren (Simultaneous localization and mapping) angewendet. Im Bereich Bildverarbeitung wird an Positionierverfahren anhand von SIFT-Eigenschaften gearbeitet. Basierend auf den Kartendaten werden Ausweichmanöver geplant und Fahrspur-Watchdogs entwickelt. Zudem werden Echtzeit- Lernverfahren basierend auf Reinforcement Learning entwickelt, welche auf optimale Kurvenfahrt abzielen.

5.3 Der Crazy-Car-Contest der FH Westküste in Heide

Im Sommersemester 2009 planen die Autoren ein Projekt als Teil des Curriculums Technische Informatik, in dem die Studierenden als Ziel am Crazy-Car-Contest der FH Westküste in Heide teilnehmen sollen. Die Teilnahme ist für die HAW Hamburg erstmalig, der Wettbewerb wird jedoch schon seit einigen Jahren ausgetragen (siehe [2]).

Parallel zum Projekt (8 Semesterwochenstunden, 10 Credit Points) wird von einem weiteren Kollegen (Reinhard Baran) ein Wahlpflichtkurs angeboten, der passend dazu die speziellen fachlichen Inhalte erarbeitet. Beim Wettbewerb fahren je zwei autonome Fahrzeuge (maximale Grundfläche DIN A3) gegeneinander. Der Kurs wird durch senkrechte Fahrbahnwände begrenzt.

Als Fahrzeugplattform ist bisher geplant ein Standard-1:10-Modellfahrzeug zu verwenden (z.B. Tamiya TT-01), welches mit einem ARM-Prozessor und IO-Board ausgerüstet wird. Die Sensorik und Softwa-

reentwicklung ist von den Studierenden zu ergänzen. Das Fahrzeug aus dem Carolo-Cup-Wettbewerb erfüllt die Anforderungen des Wettbewerbs zwar auch. Die Studierenden sollen jedoch möglichst ein eigenes Fahrzeug entwickeln.

5.4 Formula Student Wettbewerb mit HAWKS Racing Team

Im Jahr 2001 begann die Geschichte des HAWKS Racing Teams der HAW Hamburg[1] (HAWKS) mit einen Besuch an der University of Hertfordshire, wo Professoren unter anderem den aus den Vereinigten Staaten kommenden internationalen Wettbewerb "Formula-Student"[2] kennen lernten. Die nur aus Studierenden bestehenden Teams der teilnehmenden Universitäten müssen hierfür ein virtuelles Unternehmen führen. Das Unternehmensziel ist die Entwicklung und Vermarktung eines Monoposto-Sportfahrzeugs für Privatkunden.

Abbildung 5: Das HAWKS Raching Team der HAW Hamburg

Das HAWKS Racing Team war das vierte deutsche Formula-Student-Team und das erste in Hamburg. Mittlerweile fahren über 50 Teams in der deutschen Formula-Student mit. HAWKS ist ein interdisziplinäres Team aus Fahrzeug- und Maschinebaustudierenden, Informatik- und E-Technikstudierenden, Design- und Marketingstudierenden. HAWKS nimmt seit 2004 an nationalen und internationalen Wettbewerben (England und Italien) der Formula-Student teil. Beim letzten Wettbewerb in Hockenheim (2008) belegte HAWKS den neunten Gesamtrang von 78 teilnehmenden Teams und konnte somit an die Erfolge von 2007 anknüpfen.

Seit Ende 2006 ist die Informatik im HAWKS Team aktiv und hatte als erste Aufgabe ein Telemetriesystem für Motor- und Fahrzeugdaten entwickelt und in den Wagen integriert. Mittlerweile sammeln mehrere Mikrocontroller Daten von ca. 30 Sensoren, die über TTCAN [4] und eine WLAN Brücke auf einen Leitstand übertragen werden. Diese Daten, die sowohl online ausgewertet als auch archiviert werden, unterstützen Design, Analyse und Test des Wagens signifikant. Sie haben unter anderem dazu beigetragen, dass der Wagen des HAWKS Racing Teams den als Härteprüfung geltenden Ausdauerlauf über 24 Runden auf dem Infield des Hockenheimrings problemlos absolvierte. Erfahrungsgemäß halten diese Belastung nur die Hälfte der Wagen durch. Mittlerweile ist die Informatik bei der Entwicklung folgender zentraler Komponenten involviert:

- Eine Motorsteuerung, die das Wechseln von Kennfeldern während der Fahrt und die Unterstützung von ABS durch die Motorsteuerung erlaubt.
- Ein Antriebsschlupfregelung (ABS), die insbesondere die aktuelle Beschleunigung des Fahrzeugs verbessern soll.

[1] http://www.hawksracing.de
[2] http://www.formulastudent.de

- Elektronische Schaltung.
- Zusammen mit dem Interieur Team wird ein Carbonlenkrad mit integriertem TFT-Display entwickelt. Die Informatik steuert neben der Mikrocontroller-, LINUX- und QT Technologie auch Designentscheidungen bei, die auf Untersuchungen im Usability Labor der HAW Hamburg beruhen. Gemäß der Anforderung, dass ein Monoposto-Sportfahrzeug für Privatkunden entwickelt werden soll, unterstützt das Lenkrad unterschiedliche Modi wie Boxenstop, Hobby- und Rennfahrer. So wird zum Beispiel im Modus Rennfahrer auf die aktuelle Geschwindigkeit verzichtet; im Hobbyfahrer Modus hingegen muss sie zentral dargestellt werden.
- Zur Verbesserung der Ausfallsicherheit des TTCAN Bussystems wird ein auf mehreren Timemastern basierendes Redundanzkonzept entwickelt und in den Wagen integriert. Das ist eine zwingende Voraussetzung für ABS und weitere elektronische Unterstützung in Fahrwerksbereich.

Die obige Darstellung zeigt deutlich, dass die Studenten des HAWKS Racing Teams Aufgaben erledigen, die sie optimal auf das spätere Berufsleben vorbereiten. Im Gegensatz zu einem Industriepraktikum treffen die Studenten die Entscheidungen selbst und erhalten in kürzester Zeit ein spürbares Feedback. Die Umsetzung der Informatikaufgaben innerhalb des HAWKS Rennwagen erfordert Fachwissen auf unterschiedlichsten Gebieten wie zum Beispiel Mikrocontroller Architekturen, Echtzeitbetriebssysteme, Usability oder Rechnernetzen. Für den Forschungsschwerpunkt FAUST ist entscheidend, dass aktuelle Forschungsthemen wie Realtime Ethernet, Kalibrierung auf Basis von Hardware in the Loop sowie Formale Verifikationstechniken durch die Anforderungen von HAWKS getrieben werden.

5.5 Fazit

Die Projekte selbst und die Teamarbeit führen zu einer sehr hohen Motivation der Studierenden. Dies strahlt auf andere Lehrveranstaltungen aus. Studierenden beschäftigen sich mit Vorlesungsinhalten, weil sie zentral für die Lösung von Fragestellungen der studentischen Projekte sind.

Viele Studierende schreiben ihre Abschlussarbeit innerhalb der studentischen Projekte. Daraus ergeben sich zwei Aspekte, die beachtet werden müssen. Zum einen müssen die Projekte an gezielten Stellen durch studentische Hilfskräfte unterstützt werden. Diese sichern Voraussetzungen für die Abschlussarbeiten, die ansonsten nur teilweise durch Projektgruppen oder im Kontext der Abschlussarbeiten erledigt werden können.

Ein zweiter Aspekt für den Erfolg der studentischen Projekte ist Kontinuität. Ein Studierender arbeitet im Schnitt nur 10 bis 14 Monate in einem Projekt. Daher ist es entscheidend, dass über Versionsmanagement, Qualitätsicherung, Dokumentation und Unterstützung durch den Mittelbau der Hochschule die Kontinuität gewährleistet wird.

Finanzielle Unterstützung erhalten die studentischen Projekte durch Studiengebühren.

Ohne den intensiven und erfolgreichen Einsatz der Studierenden sowie die Unterstützung der Mitarbeiter und Kollegen der Fakultät wären diese studentischen Projekte nicht möglich. Dafür möchten wir uns an dieser Stelle herzlich bedanken.

Literatur

[1] TU Braunschweig. Carolo-Cup, Studenten entwickeln autonome Modellfahrzeuge. *http://www.carolo-cup.de/*, 2007.

[2] S. Hussmann and D. Jensen. Crazy car race contest: Multicourse design curricula in embedded system design. *IEEE Transactions on Education, Volume 50, Issue 1, Feb. 2007 Page(s):61 - 67*, 2007.

[3] Ulrich Knorra. Beachtliche wissenschaftliche Leistungen beim Carolo-Cup 2008. *ATZ online, http://www.atzonline.de/*, 2008.

[4] G. Leen and D. Heffernan. TTCAN: A new time-triggered controller area network. *Microprocessors and Microsystems, 26(2):77-94*, 2002.

[5] S. Pareigis, F. Korf, and B. Schwarz. FAUST: Entwicklung von Fahrerassistenz- und autonomen Systemen. *Mobilität und Echtzeit- Pearl 2007: Fachtagung der GI-fachgruppe Echtzeitsysteme Boppard, Springer, ISBN 3540748369*, 2007.

Stephan Pareigis, Franz Korf
HAW Hamburg, Department Informatik
stephan.pareigis@haw-hamburg.de, franz.korf@haw-hamburg.de