Zeitkritische Regelungsaufgaben mit Windows: Regelung eines inversen Pendels mit ICONNECT

Stefan Hauk, Markus Ramsauer, Bernhard Sick
Lehrstuhl für Rechnerstrukturen (Prof. Dr.-Ing. Werner Grass)
Fakultät für Mathematik und Informatik, Universität Passau, Innstraße 33, 94032 Passau
{hauk,ramsauem,sick}@fmi.uni-passau.de

INHALT

Seit einigen Jahren steigt die Bedeutung graphischer Programmiersysteme zur Erstellung industrieller Anwendungen, z.B. in den Bereichen Messen, Steuern und Regeln. ICONNECT ist ein derartiges Programmiersystem, mit dem Anwendungen mit dem Betriebssystem Windows (95/98/ME/NT/2000) erstellt und ausgeführt werden. Windows ist kein Echtzeitbetriebssystem; d.h., die Einhaltung harter Echtzeitbedingungen kann nicht garantiert werden. Trotzdem können Zeitbedingungen mit geringen Toleranzen eingehalten werden; man spricht in diesem Zusammenhang auch von weichen Echtzeitbedingungen. Der vorliegende Beitrag zeigt, daß mit ICONNECT die Regelung eines inversen Pendels (inklusive Aufschaukeln des Pendels, Abfahren vorgegebener Bahnen, Überschlag des Pendels) realisiert werden kann. Der Regelalgorithmus mit graphischen Anzeige- und Bedienelementen ist als Signalgraph in ICONNECT implementiert. Die Zykluszeiten bei der Ausführung dieses Signalgraphen liegen etwa im Bereich von 5 ms. Dieses Beispiel zeigt, daß bestimmte zeitkritische Regelungsaufgaben mit ICONNECT unter Windows ohne Verwendung teuerer Zusatzhardware gelöst werden können.

1. EINFÜHRUNG

Die Regelung eines inversen Pendels ist ein Standard-Benchmark-Problem aus dem Bereich der Regelungstechnik. Auf einer Schiene ist ein horizontal frei beweglicher Schlitten angebracht (siehe Abbildung 1). Auf diesem Schlitten ist ein Stab montiert, der an einer Achse um 360° drehbar ist. Aufgabe der Regelung ist es, den Stab durch geeignete Ansteuerung des Schlittens in einer aufrechten Position balanciert zu halten. Die Position des Schlittens soll sich außerdem einem vorgegebenen Wert nähern.

Ähnliche Aufgabenstellungen finden sich häufig in der Praxis, wobei hier drei eher spektakuläre erwähnt werden sollen [1]:

1. die Regelung der vertikalen Abweichung eines Space Shuttle beim Start,
2. das Balancieren einer Rakete bei der Fahrt von der Montagehalle zur Startrampe und
3. das Halten eines zweibeinigen Roboters in einer aufrechten Position.

Das inverse Pendel kann als nichtlineares System vierter Ordnung angesehen werden [2]. Der Entwurf eines geeigneten Reglers soll hier nicht näher betrachtet werden; bei der Realisierung des Reglers können verschiedene Reglerparadigmen, wie z.B. PID-Regler, Fuzzy Systeme oder Neuronale Netze zum Einsatz kommen.

Ein zu entwickelnder Regler muß dem Versuchsaufbau entsprechend ausreichend kurze Reaktionszeiten aufweisen; die Ausgaben des Reglers müssen in etwa in äquidistanten Zeitabständen erfolgen. Wenn ein geeigneter Regelalgorithmus unter Windows implementiert und ausgeführt wird, so darf das Balancieren des Pendels nicht durch Maus-Interrupts, Ausführung anderer Programme usw. gestört werden.

Abb. 1: Inverses Pendel (Versuchsaufbau)

2. VERSUCHSAUFBAU

Der hier verwendete Versuchsaufbau (siehe Abbildung 1) besteht aus einem 6.5 cm langen Schlitten auf einer etwa 1 m langen Schiene. Der Schlitten kann mittels eines Seilzugs horizontal bewegt werden. Das Pendel hat eine Länge von etwa 20 cm; am Stabende ist eine Scheibe mit etwa 9 cm Durchmesser angebracht. Die Position des Schlittens auf der Schiene sowie der Pendelwinkel werden mit Hilfe zweier Potentiometer gemessen. Die A/D- bzw. die D/A-

Umsetzung erfolgen mit einer Genauigkeit von 12 Bit. Abbildung 2 zeigt schematisch den Aufbau des Regelkreises. Eingangsgrößen des Reglers sind die Position des Schlittens bzw. der Winkel des Pendels; Ausgangsgröße ist die Geschwindigkeit des Schlittens.

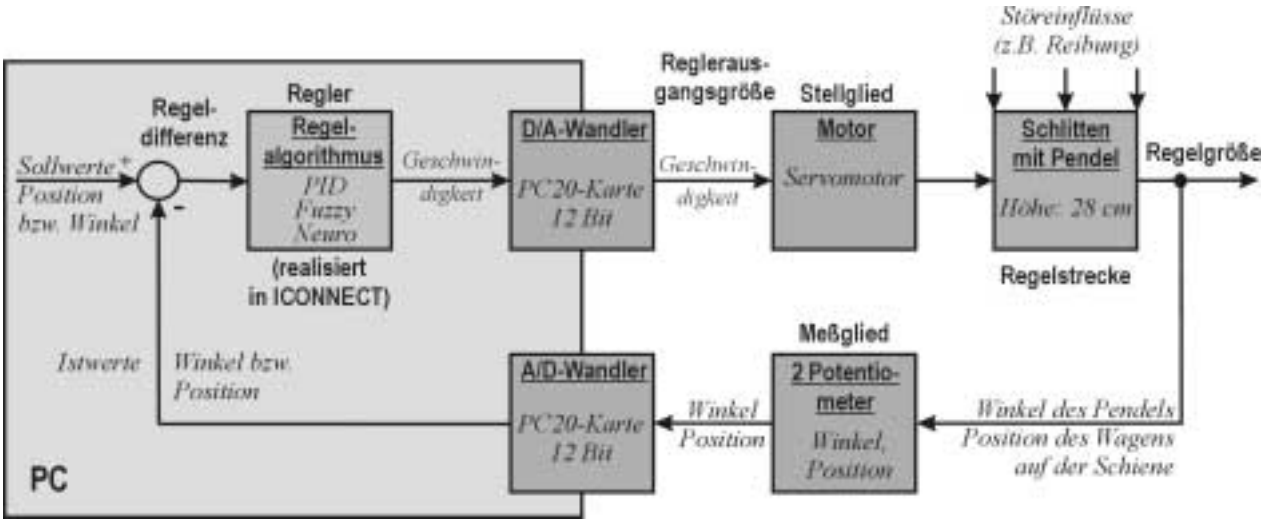

Abb. 2: Schematische Darstellung des Pendelregelkreises

3. ICONNECT

ICONNECT ist ein Werkzeug zur graphischen (visuellen) Programmierung von Software für die Bereiche Messen, Steuern und Regeln mittels einer Bibliothek parametrisierbarer Basisalgorithmen (Module). ICONNECT stellt dazu einen graphischen Editor bereit, mit dessen Hilfe komplexe Programme in Form von gerichteten Graphen (sogenannten Signalgraphen) erstellt werden können. Abbildung 4 zeigt einen derartigen Signalgraphen.

Quellen des Graphen (i.a. Knoten ohne Eingänge) stellen üblicherweise Daten zur Verarbeitung bereit, z.B. Signale von Sensoren, Werte von graphischen Bedienelementen oder Parameter aus Dateien oder Datenbanken. Senken des Graphen (i.a. Knoten ohne Ausgänge) geben Ergebnisse an Aktoren weiter, versenden sie über Rechnernetze, speichern sie in Dateien ab, zeigen sie in Displays an usw. Im Beispiel des Pendels werden insbesondere Meßwerte der beiden Potentiometer (Pendelwinkel und Schlittenposition) mit Hilfe von Quellenmodulen erfaßt und die Sollgeschwindigkeit des Schlittens wird an einem Senkenmodul ausgegeben. Zwischen Quellen und Senken werden Signale über Kanten des Graphen übertragen und in den Knoten (Modulen) verarbeitet. Hierzu steht eine große Palette unterschiedlichster Module in einer umfangreichen Modulbibliothek zur Verfügung, beispielsweise Module zur Signalanalyse im Zeit- und Frequenzbereich, zur Bildverarbeitung, zur Realisierung von Fuzzy-Systemen, zur Realisierung linearer Regler usw. Für die Regelung des inversen Pendels werden insbesondere Module für PID-Regler benötigt. Abbildung 3 zeigt den Parameterdialog des PID-Moduls.

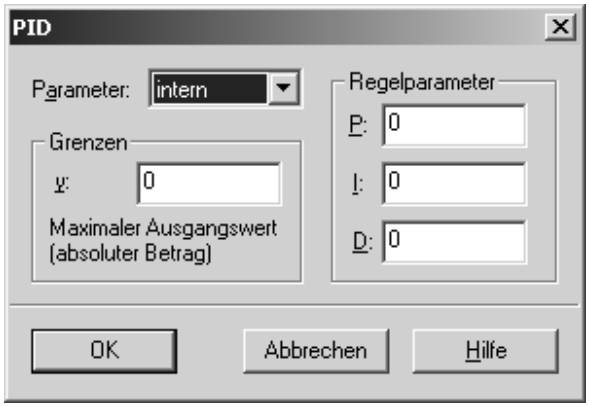

Abb. 3: Parameterdialog des PID-Moduls

Die durch Signalgraphen repräsentierten komplexen Algorithmen werden von einer Ablaufsteuerung ausgeführt, die den Datenfluß überwacht und die Bearbeitungsreihenfolge der Module eines Graphen dynamisch (d.h. zur Laufzeit) nach dem Datenflußprinzip bestimmt.

ICONNECT wird von der Firma Micro-Epsilon Meßtechnik für Windows 95/98/ME/NT/2000 angeboten. Nähere Informationen sind im WWW unter der URL http://www.micro-epsilon.de erhältlich bzw. in [3,4,5,6] zu finden.

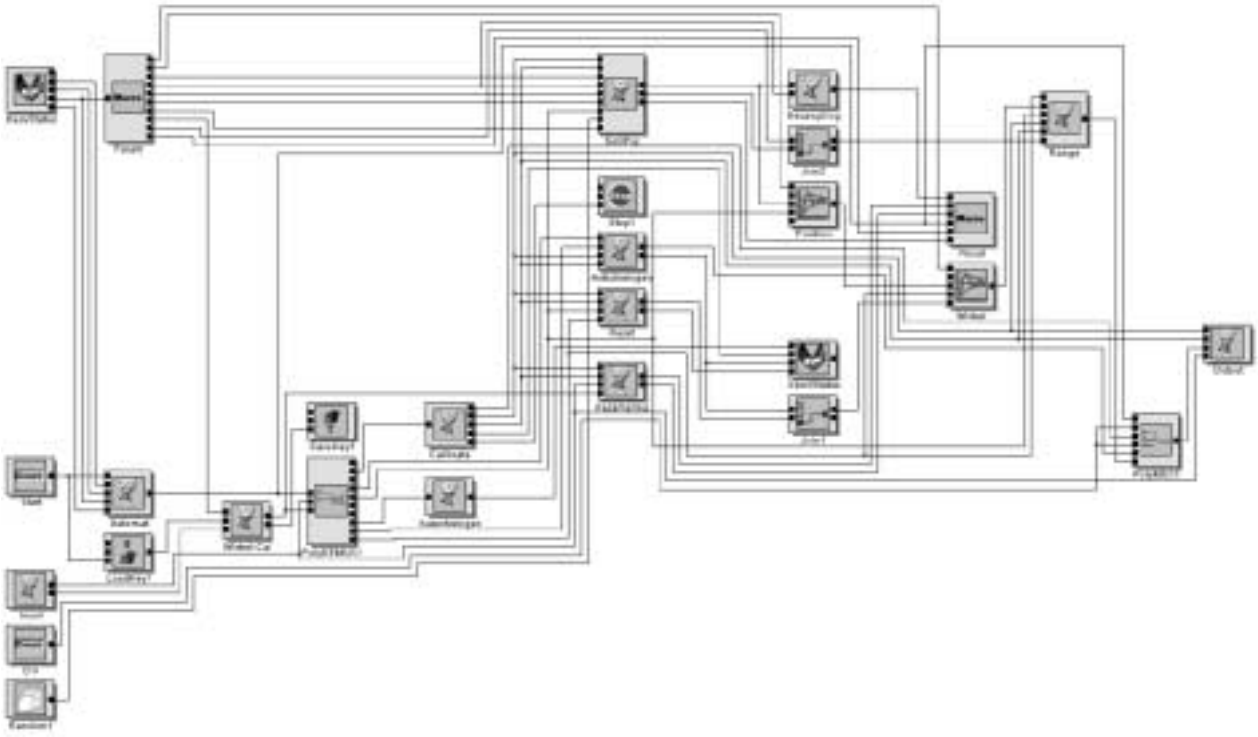

Abb. 4: Algorithmus zur Regelung des inversen Pendels in ICONNECT (Signalgraph)

4. REGELUNG DES PENDELS

Die Regelung erfolgt in mehreren Phasen. Vor dem Aufschwingen wird das Pendel initialisiert und kalibriert (Winkel und Schienenbegrenzungen). Dann wird es in die aufrechte Lage aufgeschwungen und balanciert. Während des Balancierens kann der Schlitten verschiedene Bewegungen durchführen oder das Pendel kann zum Überschlag gebracht werden. Der hierzu entwickelte Signalgraph ist in Abbildung 4 dargestellt.

Um das Pendel aufschwingen zu lassen, wird der Schlitten zunächst an den linken Rand der Schiene bewegt und es wird gewartet, bis das Pendel einigermaßen ruhig nach unten hängt (Ausschwingen). Dann wird der Schlitten zur Mitte der Schiene hin maximal beschleunigt und dort abrupt an einem vorgegebenen Punkt abgebremst, wodurch die Trägheit des Pendels zu einer Aufschwungbewegung führt. Abhängig von der erreichten Winkelgeschwindigkeit muß das Pendel nachgeregelt werden, um die Bewegung zu verstärken bzw. einen Überschlag zu verhindern. Dazu wird in einem Bereich von etwa 40° um die aufrechte Position die Winkelgeschwindigkeit gemessen und durch schnelle Schlittenbewegungen korrigiert. Sobald das Pendel die aufrechte Lage mit akzeptabler Winkelgeschwindigkeit erreicht hat, wird die Haltephase eingeleitet.

Das Halten des Pendels wird durch einen PID-Regler bewerkstelligt, der den aktuellen Winkel des Pendels als Eingabe und die Soll-Geschwindigkeit des Schlittens als Ausgabe hat. Die Regelung ist stabil genug, um auch mit stärkeren, von außen auf das Pendel einwirkenden Stößen zurechtzukommen.

Aufgrund der beschriebenen Regelung mit einem fest vorgegebenen Winkel (im allgemeinen 0°, d.h., das Pendel soll senkrecht stehen) kann es sein, daß der Schlitten allmählich an ein Ende der Schiene driftet und somit das Pendel nicht mehr gehalten werden kann. Um dies zu vermeiden, wurde dem PID-Regler der Eingang Soll-Winkel hinzugefügt. Der entsprechende Wert wird von einem PD-Regler erzeugt. Dieser hat die Aufgabe, den Schlitten in der Mitte der Schiene zu halten. Dies geschieht dadurch, daß der Soll-Winkel leicht in Richtung des Zielpunktes auf der Schiene zeigt.

Zu Demonstrationszwecken kann das Pendel auf Knopfdruck oder in periodischen Abständen fallen gelassen werden. Der Schlitten kann während des Balancierens auch eine sinusförmige Bewegung durchführen oder verschiedene zufällige oder frei wählbare Positionen anfahren. Es ist sogar möglich, die Schiene während des Balancierens an einer Seite anzuheben und die so entstandene Schrägstellung per Mausklick automatisch zu korrigieren.

Beim Fallenlassen des Pendels kommt es meist dazu, daß sich das Pendel überschlägt. In diesem Fall kann der Regler das Pendel beim erneuten Erreichen der aufrechten Position wieder fangen und halten. Sobald das Pendel wieder gehalten wird, fährt der Schlitten an seine Soll-Position zurück.

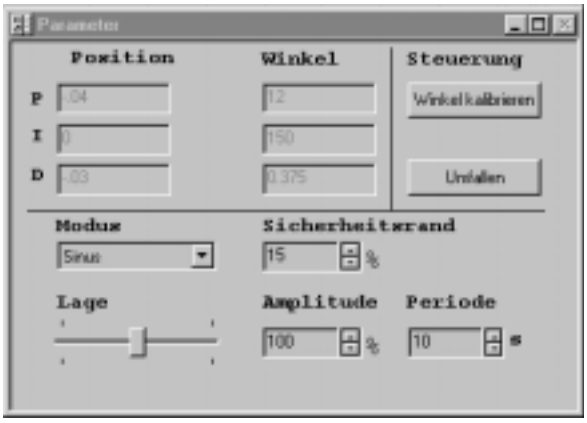

Abb. 5: Input-Manager (Bedienelemente)

In Abbildung 5 ist der Eingabedialog des Regelalgorithmus zu sehen, in dem unter anderem vor und während der Regelung die Parameter der PID-Regler eingestellt werden können. Die Schalter *Winkel kalibrieren* und *Umfallen* bewirken eine Justage des Schienenwinkels bzw. einen Überschlag des Pendels. Im Feld *Modus* kann man zwischen den Möglichkeiten zur Positionsvorgabe wählen (*Sinus*, *Zufall*, *manuell*). Im Feld *Sicherheitsrand* ist festgelegt, wie nahe an den Rändern die Soll-Position liegen darf. Durch den Schieberegler *Lage* kann die Soll-Position im Modus *manuell* eingestellt werden. Der Wert *Periode* gibt die Zeit an, in der das Pendel im Modus *Sinus* einmal hin und her fährt, im Modus *Zufall* die Zeit, nach der eine neue zufällige Soll-Position eingestellt wird. Als letzter einstellbarer Parameter legt die *Amplitude* fest, wie weit der Schlitten im Modus *Sinus* nach links und rechts bewegt wird.

Abbildung 6 zeigt das Ausgabefenster des Regelalgorithmus. In diesem Fenster werden der aktuelle *Winkel* des Pendels sowie *Ist- und Soll-Position* des Schlittens angezeigt werden. Das Feld *Phase* gibt den aktuellen Zustand der Regelung an (z.B. *Kalibrieren*, *Ausschwingen*, *Aufschwingen*, *Halten*) und der Balken *Periode* zeigt die verbleibende Zeit bis zum Beginn der nächsten Periode.

In ICONNECT wird der Entwurf von komplexen Bedien- und Anzeigefeldern sehr gut unterstützt. Zuerst wählt man vorgefertigte graphische Bedien- und Anzeigeelemente, die die gewünschten Eingaben aufnehmen bzw. die Resultate darstellen. Anschließend faßt man diese Fenster mit dem *Input-* bzw. *Display-Manager* zusammen, skaliert ihre Größen und beschriftet sie.

Bei der Entwicklung dieser komplexen und zeitkritischen Anwendung wurden überwiegend vorgefertigte Module eingesetzt, die jedoch mittels der Parameterdialoge an die Anwendung angepaßt wurden. Der Anteil der Module vom Typ *Interpret* (enthält einen einfachen C-Interpreter) liegt hier über dem sonst üblichen Maß, da viele einfache, kleine Aufgaben wie der Automat, der die verschiedenen Phasen der Regelung anstößt, oder die Winkelkalibrierung so leichter realisiert werden konnten.

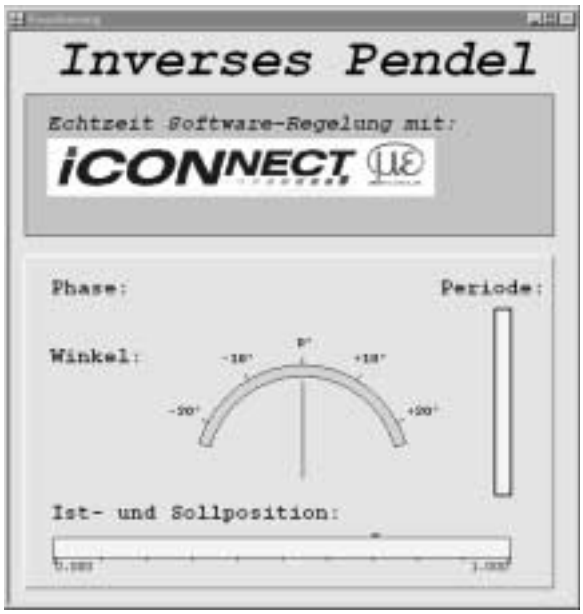

Abb. 6: Display-Manager (Anzeigeelemente)

Neben der auf PID-Reglern basierenden Lösung wurden auch Lösungen mit Neuronalen Netzen bzw. Fuzzy-Reglern untersucht [7]. Entsprechende Module werden in ICONNECT ebenfalls bereitgestellt.

5. ERGEBNISSE

Umfangreiche Experimente mit dem Versuchsaufbau zeigten, daß ein Starten weiterer Programme, Bewegungen der Maus o.ä. keine Störungen des Pendelverhaltens zur Folge haben. Das Pendel kann sogar so gut in die senkrechte Position gebracht werden, daß der Regler sich selbst abschalten kann und erst bei einer Störung (z.B. durch Anstoßen des Pendels) wieder in Aktion tritt und die Regelung fortsetzt. Dieses Verhalten kann allerdings nur mit Windows NT oder 2000 erreicht werden.

Interessant ist auch, wie eine vorgegebene Zykluszeit für die Ausführung des Regelalgorithmus in ICONNECT eingehalten werden kann. Bei einer Vorgabe von 5 ms war die tatsächlich gemessene durchschnittliche Zykluszeit 5.1 ms. Diese Zeit schließt die Ausgaben auf Displays und die Abfrage der Bedienelemente ein. Die mittleren Abweichungen von dieser Zykluszeit lagen bei etwa 0.1 ms.

Das untersuchte Beispiel des inversen Pendels zeigt somit, daß die Erledigung bestimmter zeitkritischer Aufgaben der Signalverarbeitung oder Regelung mit ICONNECT unter Windows möglich ist, auch wenn ein Echtzeitverhalten im eigentlichen Sinne (harte Echtzeitbedingungen) unter Windows prinzipiell nicht realisiert werden kann. In vielen praktischen Anwen-

dungen ist auf diese Weise der Verzicht auf teuere Zusatzhardware (z.B. Einsteckkarten) mit entsprechender Software möglich. In einem Dual-Prozessor-PC können mit ICONNECT sogar minimale Zykluszeiten von etwa 0.2 ms realisiert werden.
Abschließend soll noch erwähnt werden, daß die Regelung des inversen Pendels mit LabVIEW ebenfalls untersucht wurde [7]. Ein geeigneter Signalgraph mit Anzeige- und Bedienelementen war leicht zu erstellen, allerdings mit etwas höherem Zeitaufwand, da dieses Tool weniger intuitiv zu bedienen ist. Unterschiede zeigten sich aber schnell in den Zykluszeiten; hier wurden in LabVIEW Zeiten von 16 ms und mehr gemessen. Die Folge waren trotz angepaßter PID-Parameter ruckartige Pendelbewegungen. In LabVIEW gelang es im Gegensatz zu ICONNECT nicht, das Pendel in der Mitte des Schiene zum Stillstand zu bringen. Außerdem konnte beobachtet werden, daß Aktionen wie z.B. das Verschieben eines Fensters den Regelalgorithmus bereits empfindlich störten. In vielen Fällen konnte das Pendel dann nicht mehr senkrecht gehalten werden.

ANMERKUNGEN

Filme und Standbilder zum Versuchsaufbau des inversen Pendels können unter folgender WWW-Adresse betrachtet werden:
http://lrs.fmi.uni-passau.de/projekte/iconnect/filme
bzw.
http://lrs.fmi.uni-passau.de/projekte/iconnect/bilder
Eine Demoversion von ICONNECT findet sich unter der WWW-Adresse
http://www.micro-epsilon.de/de/produkte/download.htm

LITERATUR

[1] Deasy, S.; Dinneen, M.: *The Inverted Pendulum.* Technischer Bericht, University College Cork, Projektbeschreibung im WWW unter der Adresse http://ipendulum.cjb.net

[2] Messner, W. C.; Tilbury, D. M.: *Control Tutorials for MATLAB and Simulink – A Web-Based Approach.* Addison-Wesley Publishing, Menlo Park, Reading, Harlow, 1999

[3] Micro-Epsilon Messtechnik GmbH & Co. KG: *ICONNECT – Graphisches Entwicklungssystem zum objektorientierten Design von Datenverarbeitungsalgorithmen.* Handbuch zur Version 3.0, Ortenburg, 2000

[4] Sicheneder, A.; Bender, A.; Fuchs, E.; Mandl, R.; Sick, B.: *A Framework for the Graphical Specification and Execution of Complex Signal Processing Applications.* In: Proceedings of the 1998 International Conference on Acoustics, Speech, and Signal Processing (ICASSP'98), Seattle; Bd. 3, S. 1757 – 1760, 1998

[5] Sicheneder, A.; Bender, A.; Fuchs, E.; Mandl, R.; Mendler, M.; Sick, B.: *Tool-supported Software Design and Program Execution for Signal Processing Applications Using Modular Software Components.* In: Proceedings of the International Workshop on Software Tools for Technology Transfer (STTT'98), Aaalborg; (Hrsg. Margaria, T.; Steffen, B.); S. 61 – 70, 1998

[6] Maydl, W.; Ramsauer, M.; Schwarzfischer, T.; Sick, B.: *ICONNECT: Ein datenflußorientiertes, visuelles Programmiersystem für die Online-Signalverarbeitung*. Tagungsband Engineering komplexer Automatisierungssysteme, Braunschweig, 2001 (akzeptiert)

[7] Ehrnböck, K.; Hauk, S.: Jünger, C.; Sick, B.: *LabVIEW und ICONNECT: Zwei Signalverarbeitungstools im Vergleich.* Technischer Bericht (unveröffentlicht), Universität Passau, 2000

AUTOREN

Stefan Hauk studiert Informatik an der Universität Passau. Er war im Rahmen von Industrieprojekten als studentische Hilfskraft am Lehrstuhl für Rechnerstrukturen tätig und beschäftigte sich dabei mit der Anwendung von ICONNECT und der Implementierung von Modulen. Derzeit spezialisiert er sich im Hauptstudium mit dem Schwerpunkt „Datenbanken".

Dipl.-Inform. Markus Ramsauer hat Informatik an der Universität Passau studiert. Er ist derzeit Wissenschaftlicher Mitarbeiter am Lehrstuhl für Rechnerstrukturen und beschäftigt sich in Forschungs- und Industrieprojekten mit den Themen „Programmierwerkzeuge im Bereich Messen, Steuern und Regeln" und „Scheduling in Echtzeitsystemen".

Dr. Bernhard Sick hat Informatik an der Universität Passau studiert. Er ist derzeit Wissenschaftlicher Assistent am Lehrstuhl für Rechnerstrukturen und beschäftigt sich in Forschungs- und Industrieprojekten mit den Themen „Programmierwerkzeuge im Bereich Messen, Steuern und Regeln", „Entwurf von Echtzeitsystemen", „Robotik" sowie „Methoden des Soft-Computing".