Linux im Zementwerk
zur vollrobotischen Kransteuerung von Schüttgutkranen

von Jürgen Sauer

Unter zunehmendem Kostendruck werden immer weiter gehendere Rationalisierungsmaßnahmen notwendig. Beim Schüttgutumschlag stellen sich hierbei besondere Aufgaben. Das Gesamtkonzept einer robotorisierten Schüttgutumschlaganlage muß folgenden Anforderungen genügen:

- möglichst einfache Technik (Standard-Komponenten)
- Nachrüstbarkeit auf bestehenden Krananlagen
- redundanter Betrieb manuell - vollautomatisch
- einfache Wartbarkeit vor Ort
- 24 Stunden/ 365 Tage Betrieb
- flexible Anpassung an Produktionsgegebenheiten

Mit dieser Aufgabenstellung wurde ab 1995 ein universelles Softwarepaket für Kransteuerungen entwickelt, das Schüttgut so effektiv wie möglich umschlagen kann. Die Aufgabe lautete, die manuelle Kranbedienung weitestgehend zu ersetzen und nicht die Aufgabe auf einen Kranführer in einer Leitwarte zu verschieben. Des weiteren sollten sämtliche erfaßten Daten über eine EDV-Standard-Schnittstelle für das Controlling und die Materialwirtschaft des Kunden zur Verfügung stehen. Auch musste sichergestellt sein, dass Betriebsdaten und erfolgte Manipulationen des Lagergutes sowohl im manuellen wie auch im vollautomatischen Betrieb auf aktuellem Stand bleiben. Die Problemanalyse zeigte, dass die Aufgabe im Rahmen eines einzelnen Automatisierungsprojektes nicht erreicht werden konnte. Eine allgemein gültige Lösung war gefragt, und das Softwarepaket musste mit geringem Aufwand auf andere Anlagen übertragbar sein. Damit stand das Entwicklungsziel, eine universelle Steuerung zu konstruieren. Das aus diesem Projekt entstandene Softwarepaket „OffenesLager“ vereint diese Merkmale in einem universellen System.

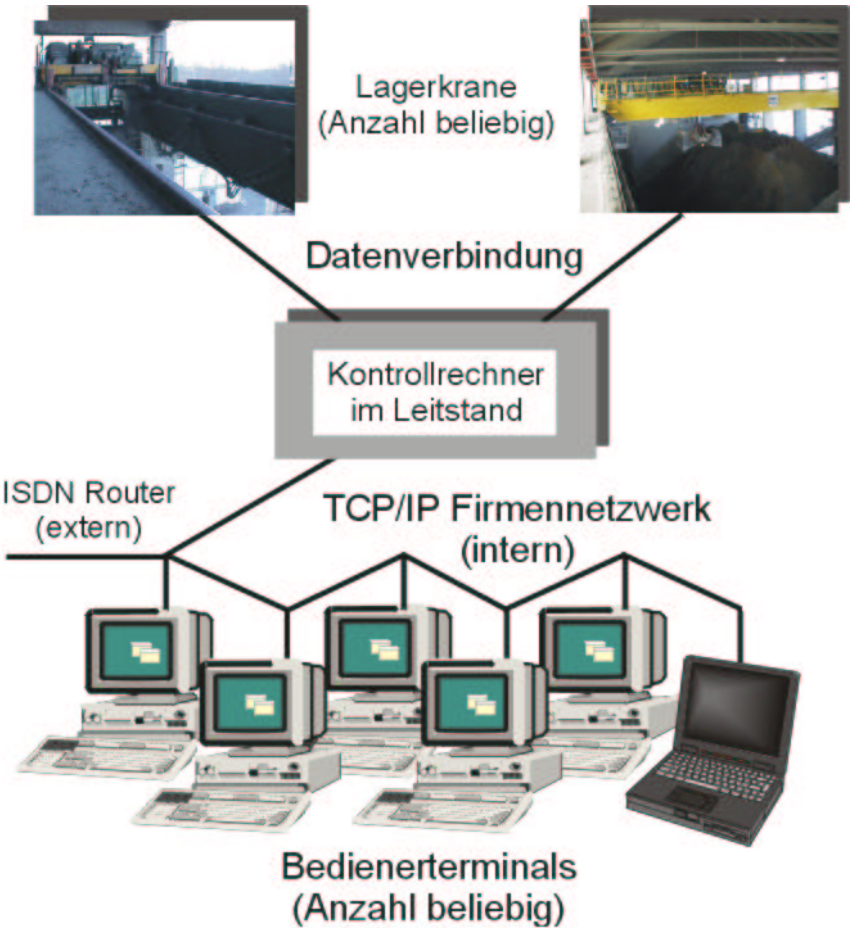
„*Offenes Lager*“ - System

Das Programm „*Offenes Lager*“ setzt auf einem „automatisierten“ Kransystem herkömmlicher Bauart auf (Positionierung, SPS-Automatik) und erweitert solches um die Fähigkeiten, selbst entscheiden zu können, **was**, **wann** und **wie** erledigt werden muß.

Das Vorbild für diesen Entscheidungsprozeß ist der reale Betriebsablauf in der Produktion. Dieser Entscheidungsprozeß ist selbstständig ausgelegt um Produktionshallen vollständig robotisch zu betreiben.

Das Programm ist ein modular aufgebautes, frei skalierbares und erweiterbares Lagerkransteuerungs- und Lagerverwaltungssystem.

Um die Produktionssicherheit zu gewährleisten, basiert das Programm auf dem Betriebssystem LINUX und kann vollständig in bestehende EDV-Firmennetzwerke eingebunden werden. Die Hauptmodule beinhalten die Steuerung des oder der Lagerkrane und die Lagerverwaltung, deren Geometrie vom Kunden **während der Laufzeit** beliebig strukturiert werden kann.
Sowohl Steuerungssystem, als auch das Lagerverwaltungssystem sind für Mehrplatzbedienung entworfen.



Die Anzahl der gesteuerten Lagerkrane und der angeschlossenen Terminals ist nach oben hin nahe zu offen, der Bedienkomfort richtet sich nach den jeweiligen Fähigkeiten des entsprechenden Terminals.

Das Lagerverwaltungssystem ist für vollautomatische, oder auch wenn nötig, für manuelle Ausführung entworfen. Manuelle Manipulationen des Lagerbestandes durch Berechtigte werden

mittels Verfolgung der Kransteuerung registriert. Zugriffsrechte, vollautomatisch oder manuell, können über eine umfangreiche Benutzerrechteverwaltung erteilt oder entzogen werden.

Die permanente Überwachung der Ein/Auslagerfunktionen ermöglichen eine variable Lagergeometrie. Die so anfallenden Daten werden parallel auf mehreren an unterschiedlichen Standorten befindlichen Systemrechnern gehalten, um Verluste durch mögliche Systemausfälle auszuschließen. Die Funktionsdaten der Subsysteme werden seriell an das Steuerprogramm weitergegeben.

Alle Informationen über Material, Lagerort und Materialmengen können von jedem dazu berechtigten Nutzer im angeschlossenem Firmennetzwerk über eine ODBC Standardschnittstelle in kürzester Zeit aus der SQL Datenbank (SAP DB vgl. http://www.sapdb.org) abgerufen werden.

Im Vollautomatikbetrieb werden die Fahraufträge für den Kran vom *„Offenen Lager“ - System* dynamisch selbsttätig generiert, mit einer Priorität versehen und in eine abzuarbeitende Auftragsliste eingetragen.
Die Generierung der Fahraufträge wird durch Füllstandsmeldungen der Sensoren (Ultraschall-/bzw. Lasersonden) in den Bunkern oder durch Schließen der Schranken in den Abladezonen ausgelöst. Prinzipiell kann jeder Aktor/Sensor verwendet werden, der über eine serielle Schnittstelle (RS232C, RS422, RS485, TTY-20mA) ansprechbar ist.

Bisher werden unterstützt :

- SPS Steuerungen diverser Hersteller
- Lenord und Bauer Positioniersystem
- diverse Waagen
- Endress und Hauser Prosonic Serie
- Sick LMS/LMI Laser Meßsystem
- div. Barcode Scanner

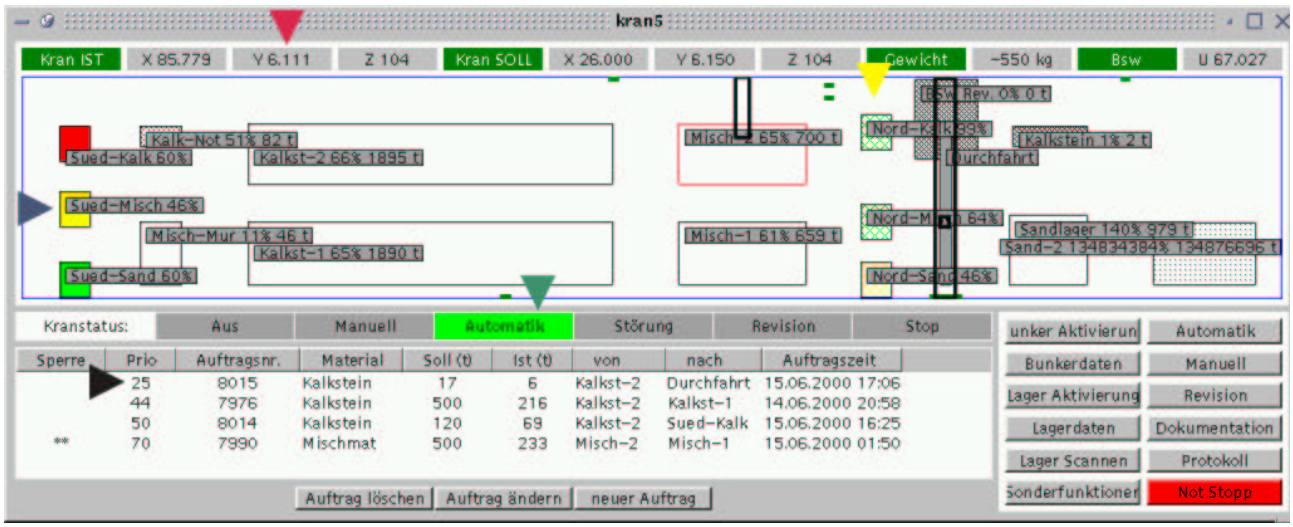
Das „*Offene Lager*“ System kann durch seinen modularen Aufbau dynamisch erweitert werden. Dies wird durch eine standardisierte Treiberumgebung erreicht , d.h. das bisher nicht unterstützte Dateneinlesegeräte durch Installation des neuen Gerätetreibers eingebunden werden können.
Für Sonderanwendungen und Kundenspezifische Erweiterungen steht eine standardisierte Programmierer -Schnittstelle zur Verfügung (Kran API / Library).

Diese selbstständig erstellte Auftragsliste paßt sich den anstehenden Anforderungen an, d.h., die Prioritäten wechseln je nach Dringlichkeit, werden entweder höher oder niedriger gesetzt. Sobald der letzte Fahrbefehl des Fahrauftrags mit jetzt niedrigerer Priorität erfolgt ist, wechselt die Automatik zu dem Fahrauftrag mit höherer Priorität. Aufträge gleicher Priorität werden im Wechsel ausgeführt.

An dieser Stelle zeige ich ein Video, daß ich hier kommentiere, Länge ca. 5 Minuten.

Anschließend zeige ich live einen Kran und erläutere die Aktionen am Bildschirm, Länge ca. 10 Minuten, dazu wird eine Handy GSM PPP Verbindung mit dem Kran live aufgebaut.

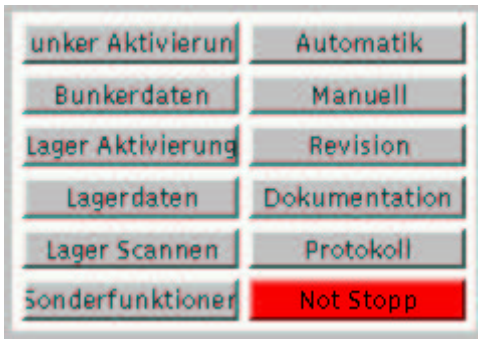
Eine Kontrolle des Betriebszustandes ist über die grafische Bedienoberfläche jederzeit gegeben und ermöglicht einen schnellen Überblick über:



- den Betriebszustand (grüner Pfeil)
- den aktuellen Fahrauftrag und die Folgeaufträge (schwarzer Pfeil)
- die Bewegungen des Krans in x, y und z -Richtung (roter Pfeil)
- welche Bunker im Vollautomatikbetrieb beschickt werden (blauer Pfeil) und welche nicht (gelber Pfeil)
- die aktuellen Füllstände der Bunker (Minimum = rot, Mittelwert = gelb, Maximum = grün)
- die Grenzen der Lager, welche Lager im Vollautomatikbetrieb aktiv oder nicht aktiv sind (Einlagern/Auslagern), den Füllstand der Lager und welche Lager für eine Überfahrt des Krans gesperrt sind
- welche Sicherheitstüren, Sperren oder Schranken geöffnet oder geschlossen sind

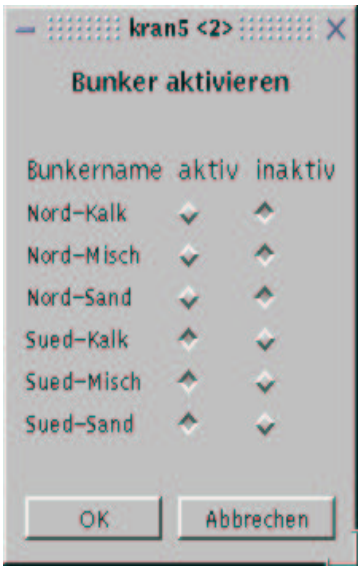
Die anstehenden Fahraufträge werden in Bewegungsbefehle für den Kran in x, y und z Richtung umgerechnet und vom Kran umgesetzt. Dabei findet eine Fahrtstreckenoptimierung unter Berücksichtigung dauernder oder zeitweise gesperrter Flächen statt.
Permanente Hintergrundaufträge, z.B. Einlagerung bestimmter Materialien in den diversen Lagerbereichen, werden abgearbeitet, wenn die Anforderungen der Produktion erfüllt wurden.
Sollten auch diese Hintergrundaufträge abgearbeitet sein, so können zur zusätzlichen Ergänzung der Datenbank die Füllstände der Lagerbereiche gescannt werden. Dies ist jedoch nur zu Anfang und in Zweifelsfällen nötig, da der Kran während aller Fahrten die Bereiche unter sich mit Sensoren scannt. Ansonsten geht der Kran in Parkposition und wartet auf Abruf.

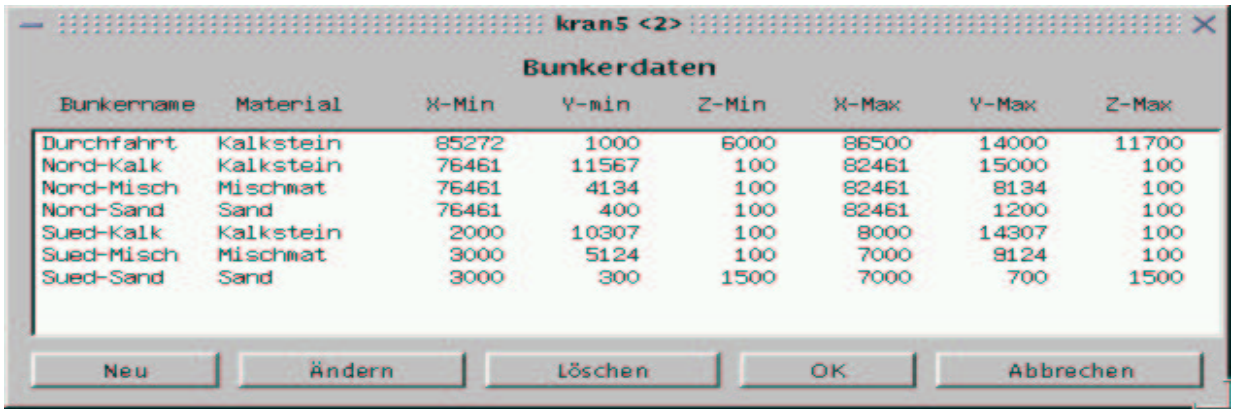
Innerhalb der grafischen Oberfläche befindet sich das zweispaltige Menü, dass sich in ***Betriebsparameter*** und in ***Betriebsart*** unterteilt.



Mit Hilfe der ***Betriebsparameter*** können die Bedingungen für den Vollautomatikbetrieb variabel vorbestimmt werden.

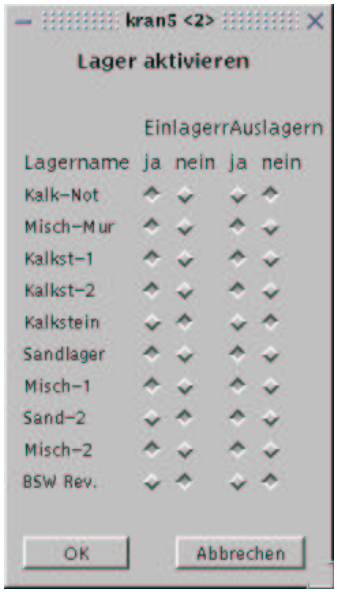
Unter ***Bunker Aktivierung*** können Bunker aktiviert oder inaktiviert werden, d.h. dass inaktivierte Bunker auch bei Minimummeldung im Vollautomatikbetrieb nicht beschickt werden. Inaktivierte Bunker werden grau unterlegt, Bunker die mit Material versorgt werden, hell.



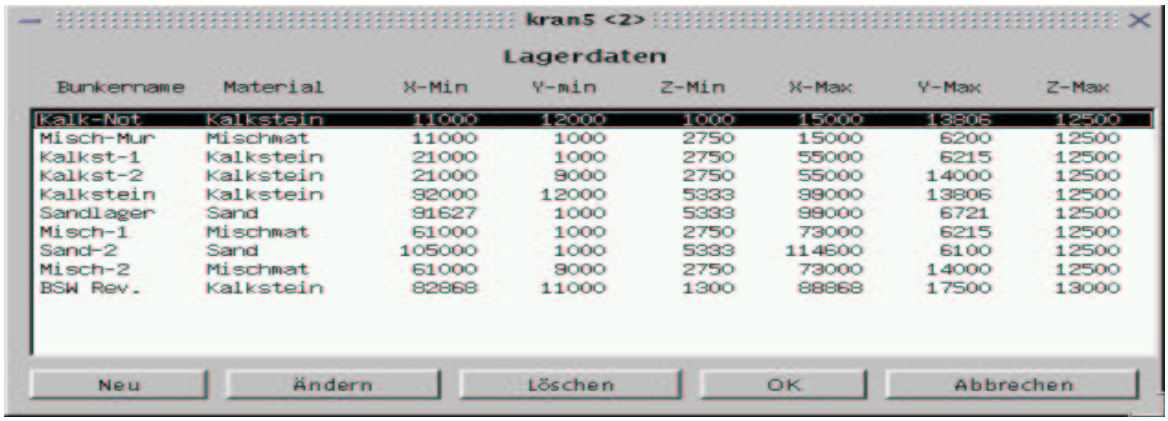


Innerhalb der ***Bunkerdaten*** können neue Bunker eingegeben, bestehende Angaben zu den Bunkern geändert oder komplette Bunker gelöscht werden.

Unter ***Lager Aktivierung*** kann der Status eines Lagers bestimmt werden. Mögliche Einstellungen sind, dass sowohl eingelagert als auch ausgelagert, nur eingelagert oder nur ausgelagert werden darf, oder aber auch dass ein Lager komplett gesperrt wird.



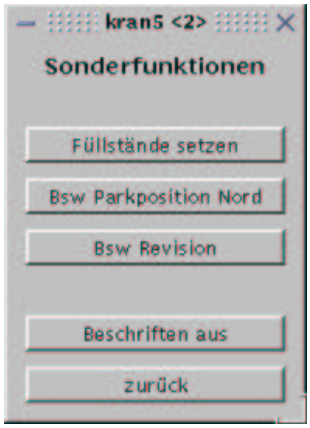
Die Lagergrenzen bestimmen sich variabel nach vorgegebenen *Xmin*, *Xmax*, *Ymin*, *Ymax*, *Zmin* und *Zmax* Werten, die in mm unter ***Lagerdaten*** eingegeben werden können.



Der Menüpunkt ***Lager Scannen*** ist für die erstmalige Aufnahme des Automatikbetriebs unerlässlich, da es sonst bei Hubarbeiten dazu führt, dass der Greifer mit zu hoher Geschwindigkeit auf das Füllmaterial trifft und beschädigt werden kann.

Dies ist auch nötig, wenn durch einen kompletten Systemausfall bedingt die manuellen Veränderungen der Lagerflächen nicht verfolgt werden konnten.

Unter dem Menüpunkt ***Sonderfunktionen*** finden sich einige spezifische Einstellungen.



Nach Vorgabe dieser Parameter ist es weder Aufgabe des Leitstandpersonals, die Fahraufträge zu erstellen, noch die Abarbeitungsreihenfolge festzulegen. Im Automatikbetrieb werden die Fahraufträge für den Kran dynamisch selbsttätig generiert, mit einer Priorität versehen und in eine abzuarbeitende Auftragsliste eingetragen. Vor Ausführung des Fahrauftrags wird die vorgesehene Fahrtstrecke auf mögliche gesperrte Flächen überprüft, seien es gesperrte Produktions- oder Lagerflächen oder eine Sperrung durch einen anderen Kran. Je nach Direktive kann das Hindernis umfahren (wenn möglich), der Auftrag zurückgestellt oder komplett storniert werden.

Die Lagerkrane werden indirekt durch das Lagerverwaltungssystem gesteuert, das heißt, die Lagersoftware erteilt Fahranweisungen, Lastaufnahmeanweisungen und Lastablageanweisungen an die ausführenden Subsysteme**.** Dabei wird das Material grundsätzlich an der höchsten Stelle des Lagers geholt und an die tiefste Stelle des Lagers gebracht.

Die Aufgabe des Leitstandpersonals beschränkt sich auf eine aus welchen Gründen auch immer nötige Übersteuerung der Vollautomatik und die Gewährleistung des geregelten Übergangs in die verschiedenen Betriebsarten (Vollautomatik, Manueller Betrieb, Wartungsbetrieb, NotAus).
Es hat auf Störmeldungen während des Automatikbetriebs zu reagieren und die erforderlichen Schritte für die Beseitigung einzuleiten. Dabei wird es durch eine Auskunft über Art, Ort und mögliche Behebung der Störung unterstützt.

Der Einsatz dieses Lagerverwaltungsprogramms beschränkt sich nicht nur auf die Kombination mit einem Kran, sondern kann, da die Steuerung auch anderen Transportmöglichkeiten angepasst werden kann, auch in anderen Bereichen eingesetzt werden.

Technische Konzepte

Die Datenbank als zentrales Element

Hier wurde ein Modell in ANSI C unter Linux mit Hilfe einer kommerziellen relationalen Datenbank (Adabas D von der Software AG, später wurde die SAP-DB verwendet) gewählt. Die Datenbank ist zentraler Dreh- und Angelpunkt der Anlage. Sämtliche Daten eines Kranes sind hier abgelegt. Damit ist es auch möglich, mehrere Krane im Gesamtverbund an einer Aufgabe einzusetzen. Die Visualisierungs- und Bedienungsoberfläche greift alle darzustellenden Daten aus der Datenbank des jeweiligen Kranes ab. Die Lagergeometrie ist so ohne Eingriffe in die Hardware, Schaltung oder SPS Programmierung der Anlage änderbar. Falls der Betreiber es wünscht, können eigene Materialwirtschaftssysteme oder Produktionscontroller in Echtzeit auf aktuelle Materialbestände im Lagersystem zugreifen. Im Betrieb der Anlagen werden sämtliche Entscheidungen der Arbeitslogik, Aktionen und Störungen exakt mitprotokolliert und von Tag zu Tag archiviert. Ebenso werden mittels einer Videoüberwachung Aktionen des Kranroboters visuell mitprotokolliert. Somit ist es für Techniker möglich, im Fehlerfalle die Ursachen zu rekonstruieren und gegebenenfalls Abhilfe zu schaffen.

Im Anfang war der Auftrag

Wie sieht nun der Ablauf des Programms in etwa aus? Zuerst wird dem System ein Auftrag erteilt. Dieser Auftrag kann von der automatischen Auftragsgenerierung, die den Arbeitsprozeß überwacht, oder gleichwertig von der Maschinenwarte durch den Bediener erteilt werden. Der Bediener-Eingriff ist die Möglichkeit, besondere Aufgaben ausführen zu können, die selten auftreten, oder bei der automatischen Auftragsgenerierung nicht erfaßt werden können. Jeder Auftrag ist mit einer Priorität und einer Auftragszeit versehen. Über die Auftragszeit können Aufgaben für bestimmte Zeiten voraus geplant werden

Aufbau des Softwarekonzeptes:

- Treiber für Geräte, ein Daemon, der Subprozesse steuert, die die Geräteschnittstellen überwachen, ebenso der Mastercontrollprozess, der alle abhängigen Prozesse kontrolliert.
- Auftragserstellendes Modul, ein Daemonprozess, der externe Sensoren kontrolliert und gemäß der Aufgabenprofile einen Auftragsdatensatz erstellt.
- Auftragsbearbeitendes Modul, der Hintergrundprozess, der die Auftragsdatensätze überwacht, und anhand der Prioritätsprofile die wichtigste Aufgabe auswählt und die Bewegungsvektoren hierfür erstellt.
- Bewegungsumsetzendes Modul, der verantwortliche Prozeß, der die physikalischen Bewegungen ausführt, und die Sicherheit sowie den Systemstatus überwacht.
- Schnittstellenmodule (SAP/R3 eDoc, u.a.)
- ODBC-Schnittstellen integriert in der Trägerdatenbank
- Visualisierungssoftwarepakete, auf Java und auf Basis KDE, oder als PHP Webinterface

Ein Auftrag könnte sein: „Hole 50 Tonnen Material XY aus Halde 'Viel Platz' und bringe diese nach ZYX mit der Priorität „Eilaufgabe/sofort“.

Ab in die Warteschlage...

Wenn ein Auftrag zur Bearbeitung ansteht - dies wird an der Priorität erkannt - wird der Auftrag im zweiten Schritt analysiert nach Ausführbarkeit (Material-Stimmigkeit, Flächensperrung, darf dies ausgeführt werden?). Ebenso wird überprüft, ob dieser Auftrag abgeschlossen wurde. Ist der Auftrag ausführbar, wird die beste Position zum Greifen des gewünschten Materials in der Halde bestimmt. Dies ist in der Regel der höchste Materialstand, wenn keine anderen Direktiven vorliegen. Alle bestimmten Ausführungsanweisungen werden in einer Ausführungswarteschlage gespeichert.

...und irgendwann geht's los

Im dritten Schritt schließlich lesen die Aktorprogramme die Ausführungswarteschlange aus und übermitteln die Ausführungsanweisungen an die Gerätetreiber. Während der Ausführung einer Bewegung achten die Aktorprogramme auf Fehler und Ergebnisse. Die einzelnen Programmteile arbeiten im Multitaskingbetrieb unabhängig voneinander und können über mehrere Computersysteme im Netzwerkcluster verteilt werden. Zu diesen einzelnen Modulen existiert eine vollständig dokumentierte Funktionsbibliothek. Es ist eine „Shared Object“ Bibliothek vorhanden, mit der die Entwicklung von Third Party Software wie auch von Gerätetreiber-Modulen möglich wird.
In Listing 1 ist ein kurzes Programmierbeispiel aufgeführt, das nichts wirklich Sinnvolles tut, sondern zur Demonstration ein Quadrat abfährt und einmal mit dem Greifer klappert.

```
/*********************************************************************************
*
*               AA                                        X   X
*              A  A       t                          t      X X
*              A  A u  u ttt  oo  mm m   aa   ttt i    X
*              AAAA u  u  t  o  o m m m a  a   t  i   X X
*              A  A  uuu  tt  oo  m m m  aa a  tt ii X   X    GmbH
*
*********************************************************************************
*
*                       Copyright (C) 1999 by AutomatiX GmbH
*
*  This file contains copyrighted material. No part of it may be reproduced
*  or used except as authorised by contract or other written permission.
*
*  See the written contract with AutomatiX GmbH for details on licensing terms
*  and warranty.
*
*********************************************************************************
*
* File    : $RCSfile: kran-demo.c,v $
* Author  : $Author: jojo $
* Date    : $Date: 2000/11/01 10:51:37 $
* Version : $Revision: 1.1.1.1 $
*
*********************************************************************************
*
* Description : actiond
* Note        :
*/

/****
 **** Demo Programm für Kranautomatisierung
 **** Einfach mal im Viereck Fahren
 ****/

#include "kran-demo.h"
#include "rc.h"

int main(int argc, char **argv)
{
  int i,hand, automatik, sicher;
  char meldungen[5][120];
  char **m;

  *m = (char *) speicher(4096);

  if (rc_lesen (SERCONF) != 0)
    {
      printf("%s: %s\n", program_name, rc_fehler ());
      return 1;
    }

  /* anhaengen an seriald */
  if (libser_sd_plugin ("actiond", SD_OPT_EXCLUSIVE, pcs, m) != 0) {
    fprintf (stderr, "%s\n", m);
    return 1;
  }
```

```
  /* wir benutzen Libser-SPS- und Achs-Funktionen */
  libser_sps_init (pcs->sd, pcs->shm, pcs->src, pcs->dst);
  libser_achs_init (pcs->sd, pcs->shm, pcs->src, pcs->dst);


  i=libser_sps_getbit("sicherheitskette-geschlossen",&sicher,m);

  if(sicher)
    printf("\n*** Sicherheitskette ist geschlossen\n");
  else
    printf("\n*** Sicherheitskette ist *** OFFEN ***\n");


  i=libser_sps_getbit("kran-hand",&hand,m);
  i=libser_sps_getbit("kran-automatik",&automatik,m);

  printf("\nKran ist: ");

  if(hand)
    printf("in Handbetrieb\n");

  if(automatik)
    printf("in Automatik - Betrieb\n");

  kran_ein(pcs, m);
  fahre(40000,  2000, 100, m);
  fahre(80000,  2000, 100, m);
  fahre(80000, 14000, 100, m);
  fahre(40000, 14000, 100, m);
  fahre(40000,  2000, 100, m);

  greifer_auf(m);
  greifer_zu(m);
  greifer_auf(m);

  kran_aus(m);

  return(0);
}
```

Bei der Entwiclung des Automatisierungs- und Steuerungssystems hatten wir oft die größten Schwierigkeiten von den Herstellern einzelner Komponenten vernünftige Herstellerunterstützung und Protokollinformationen zum Ansteuern der externen Hardware zu bekommen. Großteils haben sich die Hersteller als geradezu Linux-feindlich erwiesen. Verzögerte oder gar nicht gelieferte „Hardware Specs“ waren an der Tagesordnung - entgegen anders lautenden Vereinbarungen.

DER AUTOR

Jürgen Sauer arbeitet seit 1986 mit verschiedenen Unix-Betriebssystemen. Über den elterlichen Kranelektrik-Betrieb kam er auf die Idee, einen Kranführer zu emulieren. Im November 1997 gründete er die Firma AutomatiX GmbH. Zum Linux-Enthusiast wurde er mit den ersten verfügbaren Kerneln ab 0.91. Die AutomatiX GmbH fördert freie Software.

SAP.DB ist eingetragenes Warenzeichen von SAP, Darmstadt, Deutschland
Windows(tm) ist eingetragenes Warenzeichen der Microsoft GmbH