Ein lose gekoppeltes Rechnersystem für Echtzeitverarbeitung in einem autonomen mobilen Roboter*

Ralf Hinkel, Thomas Knierlemen, Ewald von Puttkamer

Fachbereich Informatik, Universität Kaiserslautern
Erwin Schroedinger Straße ,D-6750 Kaiserslautern, West Germany
Telefon: 0631/205 -2656 oder -2624

Schlüsselwörter:

autonome mobile Roboter, Echtzeitverarbeitung, Multiprozessor-Architektur, verteiltes Rechnersystem, autonomes Kontrollsystem

Zusammenfassung:

In einer allgemeinen Einführung wird das Gebiet der autonomen mobilen Roboter (AMR) beschrieben und eine Abgrenzung gegenüber anderen mobilen Systemen vorgenommen. Die praktischen Einsatzmöglichkeiten, speziell im Industrie- und Dienstleistungsbereich, mit den jeweiligen Problemen und Fragestellungen werden diskutiert und mit existierenden fahrerlosen Transportsystemen (FTS) verglichen. Für das Projekt Mobot-III werden die Zielsetzung, die Einsatzumgebung und die Systemkonzeption vorgestellt. Dabei werden speziell die Entscheidungskriterien für die gewählte Rechnerarchitektur und die Kommunikationsstruktur erläutert. Autonome mobile Roboter stellen wie andere komplexe Echtzeitsysteme hohe Anforderungen an das Rechnersystem. Dabei müssen Randbedingungen wie Echtzeitfähigkeit, geringe Leistungsaufnahme, gute Testmöglichkeiten, Modularität und Erweiterbarkeit berücksichtigt werden. Für das MOBOT-III- Projekt wurde ein zweistufiges verteiltes System entworfen und gebaut.

* Die hier vorgestellten Ergebnisse wurden teilweise durch das Ministerium für Wirtschaft und Verkehr, Rheinland-Pfalz gefördert.

1. Einführung

Autonome mobile Roboter sind in den letzten Jahren ein zentrales Forschungsgebiet der Robotik geworden. Gründe dafür sind nicht nur wissenschaftliche Fragestellungen und die Erprobung von Methoden der künstlichen Intelligenz sondern auch vielseitige praktische Einsatzmöglichkeiten.

Unter einem autonomen mobilen Roboter (AMR) versteht man eine Maschine, die sich selbständig und aus eigener Kraft innerhalb einer von ihr erkannten Umgebung frei bewegen kann, um gestellte Aufgaben auszuführen /1/. D.h. der Roboter muß je nach Einsatzumgebung über genügend Sensorik und Rechenleistung verfügen, damit er den erhaltenen Auftrag mit eigener Entscheidungsfreiheit durchführen kann. Dazu gehört insbesondere die Wahl des Weges und das selbständige Erkennen und Umfahren von Hindernissen.

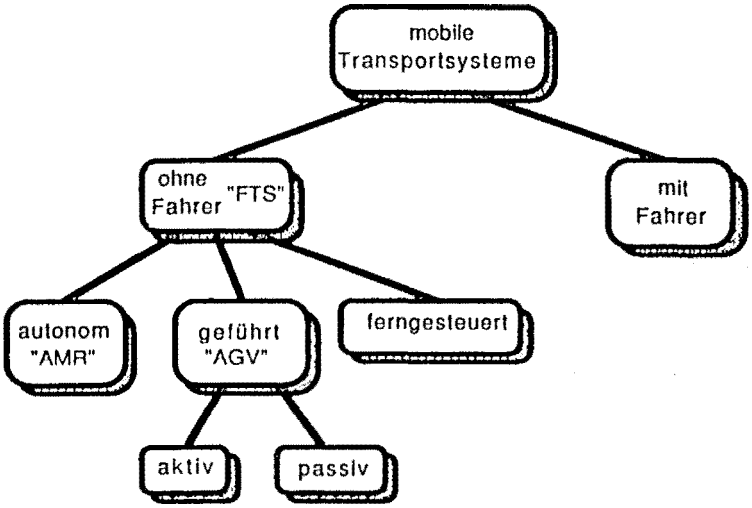

Bild 1: Kontrollmöglichkeiten für mobile Transportsysteme

Die wesentlichen Bestandteile eines AMR's sind die Antriebs- und Steuerungskomponente, das Sensorsytem, das Kontrollsystem und die Kommunikationseinheit. Entsprechend der Anwendung ist der AMR mit applikationsspezifischen Komponenten wie Montagerahmen, Hub- oder Drehtisch, Reinigungsaggregaten oder Manipulatoren ausgestattet, die jedoch den prinzipiellen Aufbau des AMR's nicht beeinflussen. Das charakteristische Merkmal eines AMR's ist die selbständige Planung und Ausführung der Aufgabenstellung. Dieser hohe Grad an Autonomie wird durch die Klassifkation der Kontrollmöglichkeiten für mobile Transportsysteme in Bild 1 belegt. Dabei ist insbesondere die Abgrenzung gegenüber den zur Zeit in der Industrie eingesetzten AGV's (automated guided vehicle) hervorzuheben. Der Unterschied liegt darin, daß der AMR für die Navigation keinen

Leitdraht benötigt, sondern sich in seiner Umgebung orientiert, indem er charakteristische Punkte als Landmarken speichert und wiedererkennt.

Bei der Realisierung eines AMR's ist die geplante Einsatzumgebung (Bild 2) ein entscheidender Faktor, der die Auswahl und den Aufbau der einzelnen Systemkomponenten bestimmt. Beispielsweise wird ein Fahrzeug, das außerhalb von Gebäuden ("outdoor") arbeitet, mit Kameras und einem aufwendigen System zur Bildauswertung ausgestattet sein, auf die beim Einsatz innerhalb von Gebäuden ("indoor") unter Umständen verzichtet werden kann. Der Indoor-Bereich läßt sich nach der Umgebungsstruktur weiter unterteilen in Büro-, Grossraum- und Fertigungsumgebung. Nimmt man die Korridore und Verbindungshallen als primäres Einsatzgebiet der Büroumgebung so bewegt man sich in relativ klaren Strukturen, die durch Wände und Türen begrenzt sind. Ähnliches gilt für Grossraumumgebungen wie Turnhallen oder Metrostationen, nur daß hier die zusammenhängende Fläche wesentlich größer ist. Die Fertigungsumgebung mit all ihren Einrichtungen stellt hingegen eine äußerst unstrukturierte Umgebung dar.

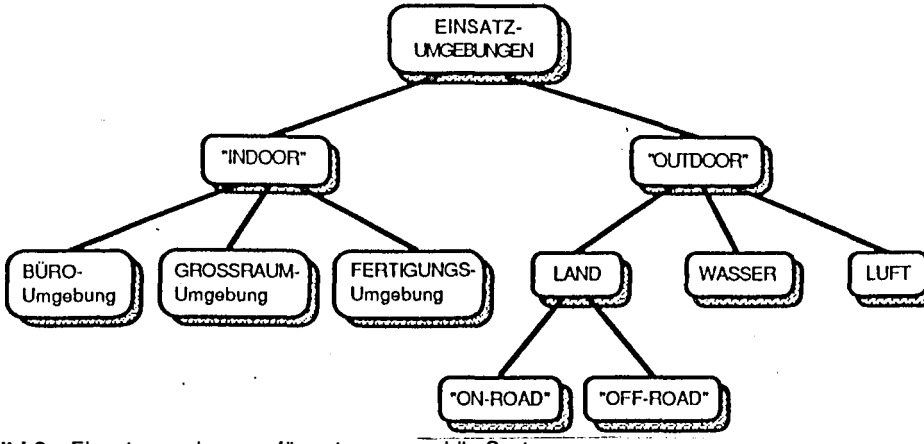

Bild 2: Einsatzumgebungen für autonome mobile Systeme

Die Komplexität der Aufgabenstellung wird weiterhin dadurch beeinflußt, ob sich der AMR in einer statischen oder dynamischen Umwelt zurechtfinden muß und wieweit mobile Hindernisse auftreten können. Wird eine statische Umgebung gefordert, so führen kleinste Veränderungen, wie ein umgestellter Stuhl oder ein versetzter Papierkorb zum Stillstand bzw. Ausfall des Systems.

Eine Frage, die zu Beginn gelöst werden muß, ist die Art und Weise wie der AMR seine Arbeitsumgebung erlernt. Dazu gibt es prinzipiell zwei Möglichkeiten, erstens die offline Grundrißgenerierung mittels entsprechendem CAD-System, d.h. die Umgebung muß explizit einprogrammiert werden, und zweitens die selbständige sensorgestützte Konstruktion des internen

Weltmodells. Letztere kann durch ein interaktives "Teach-In"-Verfahren vereinfacht werden, bei dem ein Operator das System zu den markanten Umgebungspunkten dirigiert

2. Einsatzmöglichkeiten

Grundsätzlich kann man zwei Einsatzgebiete für AMR im "indoor"-Bereich unterscheiden: den Einsatz im Dienstleistungsbereich und den Einsatz in der Fertigungsumgebung. Beide sollen im folgenden näher betrachtet werden.

2.1. Fertigungsumgebung

Das langfristige Ziel der CIM-Bestrebungen, die Fertigung nach Kundenauftrag, kann nur durch sehr kurze Entwicklungs- und Fertigungsdurchlaufzeiten erreicht werden. Daß dafür höchste Flexibilität in allen Teilbereichen der Produktion gefordert ist, steht außer Frage. Die flexible Automatisierung der Transportaufgaben zwischen Lager und Fertigung, sowie innerhalb des Fertigungsprozeß ist dabei ein wichtiges Teilziel. In Kombination mit automatisierten Lagersystemen kann die innerbetriebliche Logistik verbessert und somit die Gesamtdurchlaufzeit wesentlich reduziert werden.

Die flexible Automatisierung des Materialtransports unter möglichst geringen Lagerzeiten ist ein angestrebtes Ziel im Fertigungsbereich. Stand der Technik sind fahrerlose Transportsysteme, die entweder mit Hilfe eines passiven Metallbandes /2/ oder induktiv mit einem stromdurchflossenen Leiter /3/ geführt werden und somit einen eingeschränkten Grad an Flexibilität zulassen. Der nächste Schritt ist das spurfreie Fahren ohne Kontakt mit einem Leitdraht und entspricht somit der sensorgeführten Steuerung von AMR. Damit kann die gesamte Ablaufsteuerung wesentlich flexibler gestaltet werden und die Probleme wie Gegenverkehr oder blockierte Fahrbahnen können gelöst werden.

Für das spurfreie Fahren gibt es zwei Möglichkeiten, einmal die Navigation mit künstlichen Markierungen und zum anderen die Orientierung mittels der Umgebungsstruktur. Die Navigation mit Hilfe eines Echtzeit-Bildverarbeitungssystems, das sich an vorhandenen oder zusätzlich aufgebrachten hellen Farbmarkierungen orientiert /4/, ist ein Beispiel für den ersten Ansatz und wurde in diesem Jahr vorgestellt.

Aufgrund der komplexen Umgebungsstruktur einer Fertigungshalle stellt die Navigation nach der Umgebungsstruktur ein großes Problem dar. Im allgemeinen sind nur kleine Wandsegmente sichtbar, sodaß zur Orientierung zusätzliche Referenzflächen benutzt werden müssen. Eine denkbare Möglichkeit wäre die Verwendung von stationären Werkzeugmaschinen oder größeren

Einrichtungsgegenständen, falls diese eine relativ homogene Seitenstruktur besitzen /5/. Die Anzahl der verwertbaren Referenzflächen ist jedoch erfahrungsgemäß äußerst niedrig und das explizite Einprogrammieren dieser Punkte ist mit einigem Aufwand verbunden.

Eine andere Möglichkeit ist die Verwendung von flexibel positionierbaren Stellwänden zur Markierung von Kreuzungen und Abzweigungen, mit denen die Knoten der Fahrbahn festgelegt werden. Der AMR benutzt diese Knoten zur Positionskorrektur und Neuorientierung, um zwischen zwei Knoten eine Geradeausfahrt mit Hilfe der internen Sensorik durchführen zu können. Durch diese Struktur ist es dem AMR möglich, ähnlich wie in der Büroumgebung, seine Umwelt selbst aufzunehmen und Umgebungsveränderungen automatisch in seine interne Darstellung einzutragen. Ein weiterer Vorteil dieses Ansatzes liegt darin, daß innerhalb der Fahrbahn die Fahrspur frei gewählt werden kann und somit die Probleme wie Gegenverkehr und lokale Hindernisse im Rahmen der Fahrbahn gelöst werden können. Bild 3 demonstriert den Einsatz dieser Orientierungshilfen, deren Höhe relativ variabel gehalten werden kann (20 bis 100 cm) und vergleicht sie mit der Leitdrahtführung.

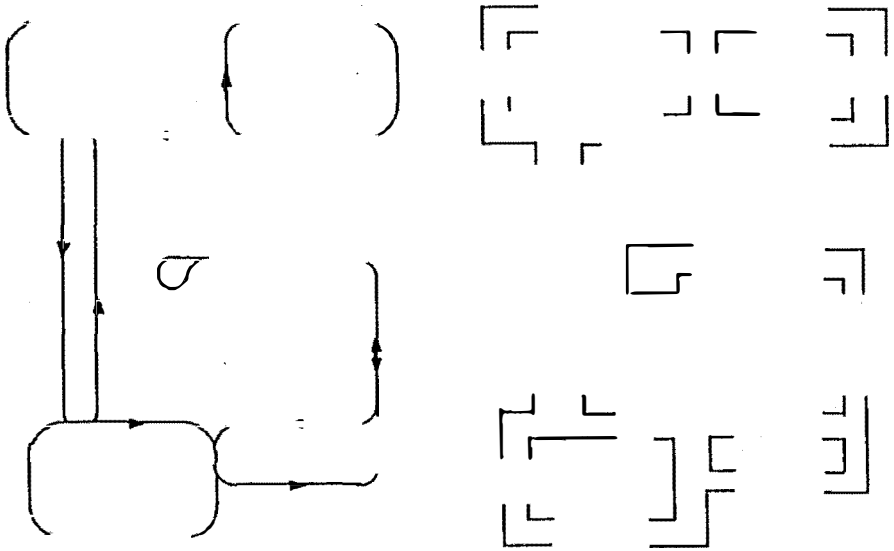

Bild 3: Leitdrahtführung versus Umgebungsnavigation mittels Stellwänden

Dieser Ansatz ist mit minimalem Aufwand zu realisieren und beeinträchtigt weder die Produktionsumgebung noch die Flexibilität des AMR's. Im Gegensatz zur Führung mit Leitdrähten oder Markierungen ist eine Änderung des Fahrkurses unproblematisch und mit geringem Aufwand durchführbar.

Sicherlich spielt auch der Sicherheits- und Kostenaspekt bei den Transportfahrzeugen eine große

Rolle. Ein Fahrzeug, das durch eine passive Leitspur geführt wird, ist sicherheitstechnisch einfacher zu handhaben, als ein im Korridor freifahrendes Fahrzeug. Auch ist es durch die einfachere Sensorik sicher kostengünstiger als ein AMR. Deshalb wird man zumindest in naher Zukunft die Systeme mit Spurbindung beibehalten, aber die Steuerung solcher Fahrzeuge "autonom" machen. D. h. die Fahrzeuge werden u. a. in der Lage sein das Spurnetz selbst zu erlernen, bei Blockaden selbständig eine neue Strecke suchen und Informationen mit anderen Fahrzeugen austauschen. Am Zielpunkt angekommen werden sie, wenn nötig, ihre Spur kurzfristig verlassen um flexible Montageaufgaben ausführen zu können. Somit wäre eine Kombination zwischen Bahnbindung im Transportbereich und lokaler Bahnfreiheit im Montagebereich eine denkbare Lösung für den Einsatz von AMR in der Fertigungsumgebung.

Neben der Automatisierung des Materialtransports kommt dem FTS auch im Montagebereich eine wachsende Bedeutung zu. Speziell die Abkehr von der Fließbandproduktion und damit vom starren Fördersystem hin zur flexiblen Montage erfordert ein neues Fördersystem als Verkettungselement zwischen den einzelnen Montageschritten. Prinzipiell können drei Gestaltungsformen unterschieden werden /6/: der Taxi-Betrieb, die mobile Werkband und das Mitfahrsystem.

Beim Taxi-Betrieb dient das FTS als reines Überbringersystem zwischen den einzelnen Montageplätzen /7/, es liefert das Montagegut an und holt es nach der Bearbeitung wieder ab. Diese Verkettung wird vorwiegend dann eingesetzt, wenn an den einzelnen Arbeitsplätzen längere Bearbeitungszeiten entstehen. Bei geringeren Bearbeitungszeiten ist eine zeitweise Trennung zwischen Montageobjekt und Transportfahrzeug nicht mehr sinnvoll und führt zum System der mobilen Werkbank. Das Mitfahrsystem ist dadurch gekennzeichnet, das der Werker seine Montageaufgabe von einer am FTS angebrachten Standplattform aus durchführt, während sich das Fahrzeug mit langsamer Geschwindigkeit fortbewegt.

Die Integration dieser Montagesysteme in komplexe Produktionsprozesse stellt neben dem flexiblen Transport ein wichtiges Einsatzgebiet für fahrerlose Transportsysteme dar. CIM-Konzeptionen der Zukunft werden entscheident durch die Möglichkeiten der FTS beeinflußt.

2.2. Dienstleistungsbereich

Die typischen Aufgaben für einen AMR im Dienstleistungsbereich sind Transportaufträge jeder Art und Reinigungsarbeiten für Fußböden /8/. Als Einsatzumgebungen kommen beispielsweise große Verwaltungs- und Bürogebäude, Krankenhäuser, Schulen- und Sporthallen in Betracht. Allen gemeinsam ist die relativ einfache Struktur der internen Verbindungswege, d.h. breite Korridore mit wenigen Einrichtungselementen, großräumige Eingangshallen, meist automatische Verbindungstüren und Fahrstühle.

Diese Umgebung bildet eine gute Grundlage für den Einsatz von AMR mit Radantrieb. Lediglich die Aufzüge müssen um eine automatische Kommunikationseinheit ergänzt werden, mit der sie die Steuerbefehle des AMR's aufnehmen können.

Unter diesen Voraussetzungen können sowohl routinemäßige Transportaufgaben der Ver- und Entsorgung, für die es bestimmte Fahrpläne gibt, als auch Spezialaufträge mit Zielvorgabe erledigt werden. Die Problematik liegt dabei weniger in der Umgebungsstruktur, die im allgemeinen nicht sehr komplex ist, sondern in der Behandlung von beweglichen Hindernissen.

Im Gegensatz zu Transportaufträgen, die normalerweise während der Dienstzeit bearbeitet werden müssen, können Reinigungsaufgaben außerhalb der Arbeitszeit durchgeführt werden. Dies bringt den Vorteil, daß die Anzahl der Personen, die sich in unmittelbarer Nähe des AMR's aufhalten, stark reduziert ist. Die Schwierigkeit bei Aufträgen im Reinigungssektor, wie Polieren von Hartböden oder Kehrsaugen von Teppichböden liegt bei dem flächendeckenden Abfahren der Umgebung. Die dabei zu fahrenden parallelen Bahnen erfordern ein präziseres Sensorsystem und genauere Navigationsalgorithmen als bei Transportaufgaben.

3. MOBOT-III - Ein autonomer mobiler Roboter

3.1. Einsatzumgebung und Systembeschreibung

Mobot-III ist ein autonomer mobiler Roboter, der für Aufgaben aus dem Dienstleistungsbereich (innerhalb von Gebäuden) konzipiert ist. Eine unbekannte Umgebung, die gewissen Randbedingungen genügt, soll automatisch mit der Navigationssensorik (Bild 4) aufgenommen und in mehrschichtiger Darstellungsform gespeichert werden. Damit ist es dem Roboter möglich sich in dieser Umwelt zurechtzufinden und zielgerichtete Aktionen durchzuführen.

Als erste Einsatzumgebungen sind großräumige Büros, kleine Hallen und Korridore geplant. Dabei wird keine statische Umwelt gefordert, sondern neben zeitlichen Veränderungen, wie verstellte Papierkörbe oder Stühle und geschlossenen bzw. geöffneten Türen sind auch ständig bewegte Objekte (Personen) erlaubt. Weiterhin soll die Aufnahme und Verarbeitung der Umgebungsdaten in Realzeit durchgeführt werden, d.h. daß der AMR ohne Stillstandphasen seine Aufgaben ausführt und dabei Kollisionen mit Hindernissen vermeidet.

MOBOT-III ist ein Dreirad, bei dem das vordere Rad angetrieben und gelenkt wird (Bild 4). Somit können die Hinterräder frei mitlaufen und der bei der Navigation störende Schlupf der Antriebsräder entfällt. Die Hinterräder sind mit inkrementellen Positionsgebern ausgerüstet und besitzen eine Auflösung von 2000 Schritten pro Umdrehung. Die Abmessungen des Fahrzeugs liegen

bei 70 * 52 * 65 Zentimetern (L * B * H), wobei das reine Fahrzeuggewicht etwa 55 Kg beträgt. Die Stromversorgung des Fahrzeugs besteht aus zwei 12 Volt Bleibatterien mit zusammen etwa 2 kWh Kapazität. Das System ist für eine Maximalgeschwindigkeit von 2 m/sec und eine durchschnittliche Arbeitsgeschwindigkeit von 1 m/sec ausgelegt.

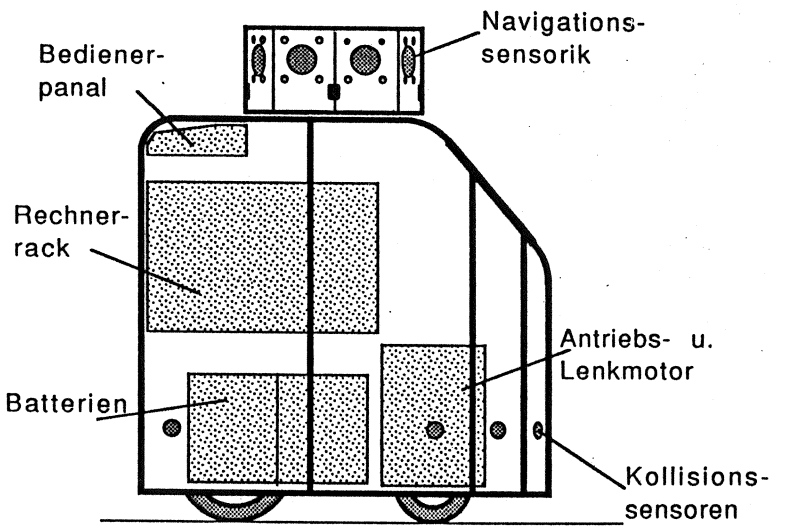

Bild 4: MOBOT-III - Aufbau

Ein 4-stufiges Sensorsystem übernimmt die Aufgaben der externen Datenaufnahme. Dabei wird ein umlaufender Sensorkopf mit 8 Ultraschall- und 4 Laserabstandssensoren für die Navigation und detaillierte Hinderniserkennung eingesetzt. Mit 13 ringförmig um das Fahrzeug angeordneten Ultraschallsensoren wird die grobe Hinderniserkennung durchgeführt und zur Kollisionsvermeidung im Nahbreich bis ca. 40 cm werden Infrarotabstandssensoren eingesetzt. Sollte diese Schutzzone trotzdem durchbrochen werden, so stoppt ein taktiler Sicherheitsring die Fahrzeugbewegung.

3.2 Entwurfsentscheidungen bezüglich des Rechnersystems

Betrachtet man sich ein AMR mit seinen komplexen Hardware-Funktionen wie Antrieb, Lenkung, sowie den unterschiedlichen Sensoren und Kommunikationseinrichtungen, so erkennt man schnell, daß ein zentraler Rechner eine komplexe Software erfordert. Betrachtet man nur die Bewegungsfunktionen, so werden ausgehend von der Zielpunktvorgabe über die Wege- und Bahnplanung bis zur Motorkoordination und Reglung komplexe Entscheidungsvorgänge durchlaufen. Gleiches gilt auch für die Sensordatenverarbeitung, die von der Steuerung der Sensorik, der Vorverarbeitung und Datenreduktion über die Fusion der Sensordaten und Generierung der

Umgebungslandkarte bis hin zur Erkennung von Objekten einen großen Rechenaufwand erfordert.

Bei der Konstruktion eines AMRs kommt daher, neben dem Sensorsystem, dem Kontrollsystem in seinem hardware- und softwaremäßigen Aufbau große Bedeutung zu. Für die Konfiguration des Rechnersystems, das die Hardwarerealisierung des Kontrollsystems darstellt, sind Randbedingungen wie OnBoard-Rechnersystem, Echtzeitfähigkeit, geringe Stromaufnahme, kleine Baugröße und Sicherheit zu beachten. Soll das System nach oben hin offen sein, spielt sicher auch die Erweiterbarkeit und Modularität der Hardware eine große Rolle.

Weiterhin müssen effiziente Fehler- und Selbstdiagnosefunktionen implementiert werden, die ein leistungsfähiges Testsystem unterstützen. Dazu müssen einfache Hardware-Funktionen, wie Ladung der Akkus, Spannungswandlung oder Temperaturüberwachung durch kleinere Rechner überwacht und gesteuert werden, damit Fehlfunktionen, wie sie durch das Absinken der Versorgungsspannung oder zu hoher Stromentnahme verursacht werden, frühzeitig durch das Rechnersystem selbst erkannt werden. Basis für ein gutes Testsystem ist sicherlich auch eine komfortable Kommunikationseinrichtung. Hierzu gehören neben den Standardschnittstellen, wie RS232, eine telemetrische Datenübertragungseinrichtung mittels Infrarot- oder Funkstrecken, und direkte Ein- und Ausgabemedien, wie graphikfähigem LCD-Schirm und Sprache. Bei der Programmentwicklung ist es sicherlich von Vorteil, wenn neue Programme direkt, unter Umgehung des EPROM-Weges, in jedes beliebige Modul geladen und dort in einem laufenden System getestet werden können.

Bezüglich der Konstruktion des Rechnersystems eines AMRs kann man dieses grob in zwei Ebenen einteilen (Bild 5). Die untere Ebene (Lowlevel) umfasst dabei die gesamten Hardware-Schnittstellenfunktionen mit der Motorsteuerung, der Systemüberwachung und den Kommunikationseinrichtungen. Auch die einfache Sensorik, wie Schalter, Infrarottaster, Ultraschallsystem und Positionssensorik ist Bestandteil dieser Ebene. Im Gegensatz dazu ist die Highlevel-Ebene praktisch unabhängig von der gewählten Hardwarestruktur und beherbergt mit ihren Rechnern und Datenspeichern das Kontrollsystem, d.h. sie stellt die eigentliche "Intelligenz" des Fahrzeugs dar.

Der Aufbau dieser oberen Ebene ist zur Zeit noch Gegenstand der Forschung /9/, sodaß der Entwurf des Rechnersystems dieser Ebene sehr flexibel gehalten werden muß. Die Lowlevel-Ebene dagegen ist mit ihren einzelnen Modulen überschaubar, sodaß hier keine Änderungen innerhalb der Module zu erwarten sind. Diese Ebene muß lediglich so flexibel sein, daß neue Module einfach ohne Hardware-Änderungen hinzugefügt werden können.

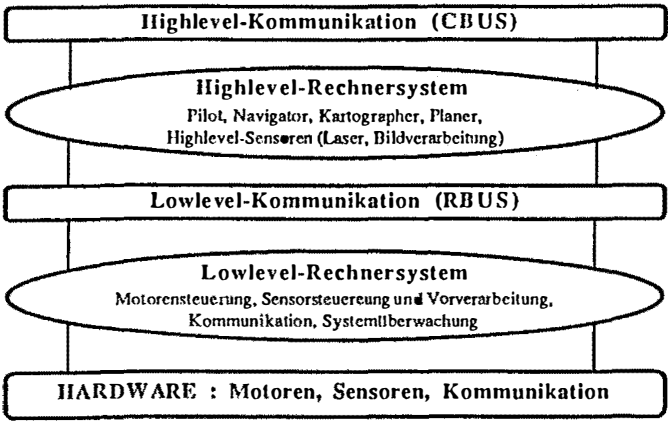

Bild 5: : Zwei-Ebenen-Struktur des Rechnersystems

Nicht zuletzt spielt auch der Sicherheitsaspekt eine große Rolle. Ein mobiles System, das bis zu 2 m/sec Geschwindigkeit erreichen soll, ist sicherlich für die in der Nähe befindlichen Personen und Gegenständen eine Gefahrenquelle. Deshalb muß bei der kleinsten Unregelmäßigkeit das System die Motoren stoppen. Wie in anderen Sicherheitssystemen müssen auch hier mehrere unabhängige Instanzen das System überwachen und das Fahrzeug anhalten können.

Aufgrund all dieser Überlegungen fiel die Entscheidung zugunsten eines zweistufigen dezentralen, mittels LANs gekoppelten Rechnersystems (Bild 6). Im folgenden werden nun das verwendete Bussystem und die Rechnermodule vorgestellt

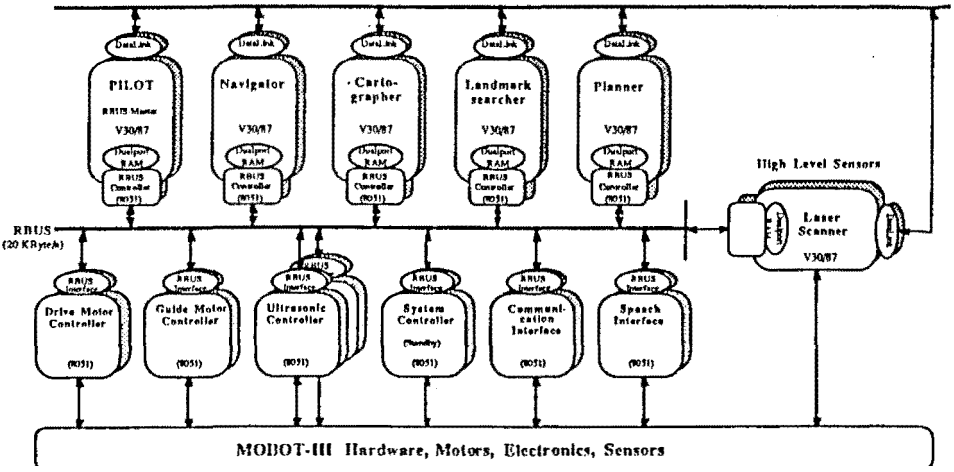

Bild 6: MOBOT-III Rechnersystem

3.3 Der Lowlevel-Bus (RBUS)

Nachdem die Entscheidung nun zugunsten eines dezentralen Systems gefallen war, stellte sich die Frage nach dem Aufbau der gemeinsammen Modulschnittstelle. Sicherlich kann man sich theoretisch sehr komfortable und funktionale Schnittstellen vorstellen. Jedoch erzwingen die Randbedingungen des AMRs und die Verfügbarkeit der benötigten Hardware bestimmte Kompromisse.

Eine Vorgabe an die Lowlevel-Ebene war die Bedingung, auch sehr einfache Module, die nur aus einem einfachen Mikrokontroller bestehen, einsetzen zu können. Deshalb war es nicht möglich ein LAN wie Ethernet einzusetzen, da der benötigte Hardwareaufwand zur Implementierung der Schnittstelle aufwendiger als die Steuerungshardware selbst geworden wäre. Solche Systeme erfordern neben dem eigentlichen Kontroller meist noch die zusätzliche Möglichkeit des DMA, damit die Daten von der Schnittstelle schnell in den Speicher transportiert werden können.

Die Entscheidung fiel letztlich zugunsten eines 187.5 KBaud schnellen seriellen Bussystems, wie es von der INTEL 8051 und 8096 Familie hardwaremäßig unterstützt wird. Dies bedeutet zwar die Festlegung auf eine bestimmte Rechnerfamilie, bereitet aber keine Probleme, da es neben mehreren Zweitherstellern auch eine große Anzahl aufwärtskompatibler Prozessoren gibt. Auf Basis dieser Mikrokontroller sind schon unterschiedliche Master-Slave-Bussysteme entwickelt und vorgestellt worden. Ein auf den ersten Blick Interessantes System, wie der INTEL 8044 Kontroller mit der BITBUS-Schnittstelle /10/, hätte zwar die Schnittstellen-Software etwas vereinfacht, wurde aber wegen der fehlenden CMOS-Version nicht eingesetzt. Außerdem ist er der einzige Kontroller mit der BITBUS-Schnittstelle, sodaß keine Auswahl bezüglich des Prozessors mehr möglich gewesen wäre. Auch sind heute serielle Kontrollerbaustelne, wie der INTEL 82510, erhältlich, die ebenfalls die seriellen Möglichkeiten der 8051 Familie besitzen und einen Anschluß an jeden Prozessortyp ermöglichen. Der Name "RBUS" steht als Abkürzung für Remote-BUS, was die Aufgabe dieses Busses treffend beschreibt : Hier wird Hardware ferngesteuert.

Die asynchrone Übertragung erfolgt bitserieil mittels Start- und Stopp-, aber ohne Paritybit. Die wichtigste Eigenschaft dieser Schnittstelle ist die Möglichkeit ein neuntes Datenbit zu übertragen und bei Empfang eines solchen Bits im Slave einen Interrupt zu erzeugen (Wake-Up). Mit diesem Mechanismus lassen sich jetzt einfach mehrere Kontroller über die Schnittstelle koppeln. Wird vor jeden Datenblock eine Zieladresse mit aktivem 9-ten Datenbit gesendet, erhalten alle angeschlossenen Kontroller einen Interrupt Ihrer seriellen Schnittstelle. In der entsprechenden Interruptprozedur müssen sie nun nachsehen, ob das empfangene Adressbyte mit der ihnen zugeordneten Adresse identisch ist. Sind sie angesprochen, dann steuern sie gemeinsam mit dem Sender den weiteren Busablauf. Im anderen Fall löschen sie das Empfangsregister und führen das zuvor unter-

brochene Programm weiter. Dieser grundlegende Mechanismus bestimmt weitgehend den Aufbau des RBUS-Systems und seiner Protokolle.

Bei der Busvergabe wurde zugunsten eines Master-Slave-Systems mit Round-Robin entschieden, bei dem die jeweiligen Highlevel-Module nach einem festgelegten Schema die Masterfunktion erhalten. Damit konnte die relativ geringe Busrate von 20 KByte/sec optimal ausgenutzt werden. Da die Lowlevel-Einheiten (die meist nur aus einem Mikrokontroller bestehen) keine Masterfunktion übernehmen, werden die von ihnen zur Verfügung gestellten Informationen vom jeweiligen Busmaster abgerufen und nach dem Broadcasting-Prinzip von allen anderen Highlevel-Moduln ebenfalls aufgenommen.

Damit der Bus durch die Verarbeitungszeiten der Slaves nicht zulange belastet wird, wurde der Datenverkehr strikt in drei Transferrichtungen eingeteilt. Die ersten beiden Modi realisieren dabei eine feste Punkt zu Punkt Verbindung, wobei beim Master-Slave-Transfer (MST) der jeweilige Master Daten oder Kommandos an die Applikation des Slaves sendet und im Slave-Master-Transfer (SMT) Daten vom Slave abholt (Bild 7). Der dritte Bustransfer, der wie ein SM-Transfer abläuft, holt ein Datenpaket von einer Einheit ab und stellt es dabei allen angeschlossenen Highlevel-Einheiten zur Verfügung. Ein Beispiel dafür ist die Positionsmeldung des Bahnrechners, die in allen anderen Moduln benötigt wird.

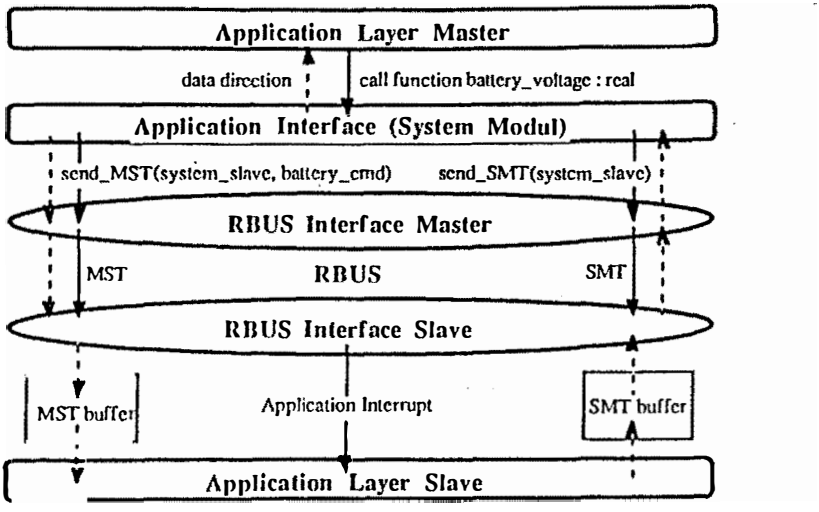

Bild 7 : Transferverlauf zwischen Master und Slave (-----> data, ⟶ command)

An einem Beispiel wird dieses Vorgehen verständlich. Betrachtet man die Ultraschallsysteme, so vergeht zwischen dem Kommando eine Messung auszuführen und dem Zeitpunkt, ab dem das

Ergebnis vorliegt, eine Zeitspanne von einigen Zehnmillisekunden. Würde der Master hier auf das Ergebnis warten, wäre während dieser Zeit der Bus blockiert. Auch ist es hier sinnvoller einmal ein MST- Kommando zur Festlegung der Betriebsparameter zu übergeben, um dann nur noch über folgende SM- Transfers das Ergebniss abzuholen. Die Trennung in MST und SMT ist auch deshalb nötig, da der im Slave befindliche RBUS-Treiber bei einem eingehenden MST-Befehl diesen nicht interpretieren und auf Daten der Anwendungsebene zugreifen kann.

Da es also keinen direkt funktionalen Zugriff des Masters zum Slave (außer bei Test- und Diagnosebefehlen) gibt, muß ein funktionaler Zugriff auf der Anwendungsebene in zwei Teile aufgespalten werden. Zuerst wird das Kommando zum Slave mit einem MS-Transfer übertragen und dann nach einer vorgegebenen Zeit, bzw. asynchron durch den Slave angefordert, das Ergebnis mit einem SM-Transfer abgeholt.

Auf der Basis der beiden Transfer-Modi wurde ein Paketformat implementiert, das im Gegensatz zu anderen Formaten keinen geschlossenen Aufbau besitzt. Es enthält zwar auch Adress-, Kontrol-, Status-, Datenlängen-, CRC- Feld und natürlich das Datenfeld selbst, wird aber nicht in einem zusammenhängenden Block übertragen. Stattdessen wird an bestimmten Stellen auf die Quittierung des Slaves gewartet. Damit wird neben der Synchronisation von Master und Slave erreicht, daß der Bus nur dann belastet wird, falls der Slave auch bereit ist das Paket zu übernehemen.

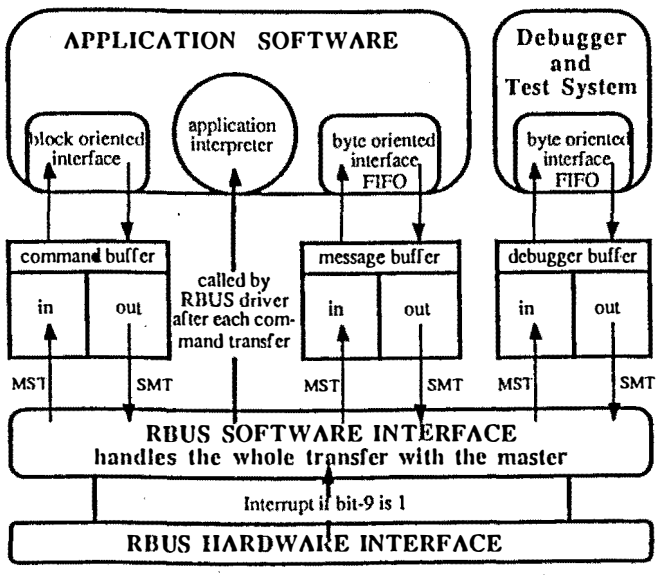

Bild 8 : RBUS Softwarestruktur auf den Interfacerechnern

Das RBUS-System übernimmt auf den 8051-Einheiten außer der reinen Transportfunktion der Nachrichten auch Betriebssystemfunktionen (Bild 8). Zur Aufnahme der Nachrichten von der

Anwendungsebene des Masters, bzw. zur Übergabe von Nachrichten an diese, steht der Command-Puffer zur Verfügung. Er kann für jede Transferrichtung jeweils genau ein Paket aufnehmen. In diesem Puffer wird über die Transportbefehle "Application-Master-Slave" (AMS) und "Application-Slave-Master" (ASM) eine Nachricht eingeschrieben bez. abgeholt. Nachdem ein Transfer erfolgreich abgeschlossen wurde, wird eine zuvor mit der Anwendungsebene vereinbarte Prozedur, der sogenannte Application-Interpreter, gestartet. Dies ermöglicht die rasche Behandlung eines solchen Ereignisses durch die Anwendungsebene.

Neben dem Command-Puffer wurden auf der Netzwerkebene noch zwei weitere Puffer mit jeweils einer MS- und SM-Datenrichtung implementiert. Während im Kommandopuffer nur jeweils eine Nachricht der kommunizierenden Anwendungsebenen eingetragen werden kann, fungieren die beiden anderen Puffer als zeichenorientierte FIFOs. Dieser zeichenorientierte Transfer dient hauptsächlich der Diagnose und dem Test der Module. Dazu werden die von einer Anwendugsebene auf einem Modul erzeugten Statusmeldungen vom Master zu einer I/O-einheit transportiert.

Aufgebaut wird der Übertragungskanal durch entsprechende Kommandos der Kommunikations-moduln. Der sogenannte Debugger-Puffer, der wie der Message-Puffer funktioniert, dient der Kommunikation mit dem Testsystem des jeweiligen Rechnermoduls. Über ihn können Speicherplätze angezeigt, modifiziert und auch Programme geladen und gestartet werden. Damit ist es möglich das gesamte Rechnersystem während des Betriebs zu testen und von außen mit neuer Software zu versorgen. Im Fehlerfall können alle Einheiten so direkt angesprochen und auf ihren Zustand hin untersucht werden. Auch ist es möglich zu bestimmten Zeitpunkten Speicherdumps durchzuführen, sodaß der Modulstatus festgehalten werden kann.

3.5 Rechnermodule

Bei der Konzeption der Rechnermodule spielte die Frage nach dem eingesetzten Prozessortyp, zumindest in der Highlevel-Ebene, eine untergeordnete Rolle, da die RBUS-Schnittstelle davon unabhängig ist. Die wichtigsten Vorgaben für das Rechnersystem waren :

- Die ausschließliche Verwendung von CMOS-Bauteilen um den Stromverbrauch so gering wie möglich zu halten. Damit ist die Stromversorgung einfacher, es werden keine Lüfter benötigt und die Rechner können vor Umwelteinflüssen sicher geschützt werden.
- Die Möglichkeit eine leistungsfähige Hochsprache einsetzen zu können, damit die Teamarbeit unterstützt und das Softwaresystem überschaubar bleibt.
- Der Rechner sollte universell einsetzbar sein und die Möglichkeit einer leistungsfähigen Realarithmetik besitzen, damit die komplexen Bahn- und Koordinatenberechnungen schnell und bei hoher Genauigkeit durchgeführt werden können.
- Der Sicherheit wegen soll jedes Rechnermodul mit einem Watchdog-Timer ausgerüstet sein,

damit Programmfehler nicht zum totalen Absturz des Systems führen.
Der Rechner soll über einen kleinen nichtflüchtigen Speicher verfügen, damit Systemparameter, auch über ein Ausschalten hinaus, abgelegt werden können.

Entschieden wurde in der oberen Ebene vorerst zugunsten eines INTEL 8086/87 Systems, allerdings wurde als Rechner, der etwas schnellere V30 von NEC eingesetzt. Zwei Gründe haben dabei hauptsächlich die Wahl entschieden. Erstens gibt es ein großes Angebot an Entwicklungssoftware, die kostengünstig auf IBM PC/XT/AT Systemen lauffähig ist. Zweitens läßt sich die einmal entwickelte Software sehr einfach auf die leistungsfähigeren Prozessoren iAPX286 und iAPX 386 portieren, sodaß bei Bedarf die Rechenleistung erhöht werden kann.

Eine offene Frage bezüglich der Rechnermodule betraf den Anschluß der Module an den Lowlevel-Bus. Dazu stehen grundsätzlich die folgenden beiden Möglichkeiten offen. Erstens die direkte Ankopplung mit einem seriellen Kontrollerbausteln und zweitens die indirekte Kopplung über einen 8051 Mikrokontroller. Um den Durchsatz des Hauptprozessors durch den Datentransfer nicht unnötig zu verringern, wurde zugunsten des Konzepts mit dem Mikrokontroller entschieden. Dies ermöglicht auch eine größere Flexibilität im Nachrichtenformat, da zum Handling der Schnittstelle ein eigener Rechner zur Verfügung steht. Offen bleibt nun noch, wie der Hauptprozessor mit dem Interfaceprozessor kommuniziert. Eine Portverbindung wurde wegen der hohen Rechnerbelastung und der Zugriff über DMA wegen der unterschiedlichen Busstruktur ausgeschlossen.

Die Entscheidung fiel aus zwei Gründen zugunsten einer Dualport-Struktur (Bild 9). Erstens kann der Interfaceprozessor, ohne daß der Master dazu Rechenzeit zur Verfügung stellen muß, die empfangenen Daten im gemeinsamen Speicher zur Verfügung stellen, bez. von dort zum Senden abholen. Zweitens kann der Interfacerechner die Daten aufbereiten und sich um die Probleme der Datensicherung und des Datentransports kümmern.

Softwaremäßig sieht es für den Hauptrechner so aus, als ob die im Dualport befindlichen Daten direkt von der jeweiligen Hardware dort eingetragen worden wären. Führt der Interfaceprozessor in regelmäßigen Zeiten die Datentransfers zu den einzelnen Modulen durch, so braucht der Hauptrechner die aktuellen Daten nur bei Bedarf aus dem Dualport-Ram zu lesen. Wann die Datentransfers zu den Slaves durchgeführt werden müssen, wird dem Interfacekontroller vom Hauptrechner beim Systemstart mitgeteilt. Beispielswelse wird zur Abfrage der Systemparameter, wie Akku-Kapazität,-Spannung und Temperatur, ein Auftrag definiert, der alle 500 msec diese Daten vom Systemkontroller in den Dualport-Speicher des Masters transferiert. Benötigt der Master nun die aktuellen Werte, so sieht er einfach im Dualport-Speicher nach.

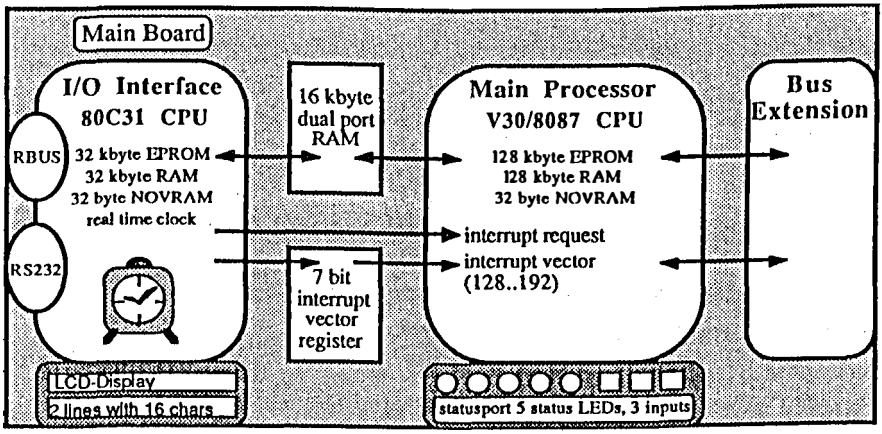

Bild 9 : MOBOT-III-Rechnermodul

Damit der Master nicht selbst dauernd diese Parameter überwachen muß, kann der Interfacerechner im Gefahrenfall beim Hauptrechner einen Interrupt erzeugen kann. Da der Interfacerechner aber die Semantik der Daten nicht kennt, muß der die Nachricht sendende Slave diese mit einer speziellen Markierung versehen, damit dann beim Master ein Interrupt erzeugt wird. Durch dieses Verfahren wird der Master entlastet und der Slave, der seine Hardware ja zu genüge kennt, trifft die Entscheidung, ob eine besondere Situation das Eingreifen des Masters erfordert.

4. Schlussbetrachtung

Das vorgestellte 2-stufige, verteilte Rechnersystem bildet die Hardwarebasis für das Kontrollsystem des autonomen mobilen Roboters MOBOT-III. Seine Installierung ist im wesentlichen abgeschlossen, sodaß sich der Schwerpunkt der Arbeiten auf den Entwurf und die Implementierung der Applikationssoftware konzentriert. Diese kann aufgrund des gewählten Kommunikationssystems sowie der Dual-Port-Struktur einfach und über funktionale Schnittstellen mit den dezentralen Sensor- und Aktuator-Moduln kommunizieren. Somit konnte zur Implementierung die Programmiersprache MODULA-2 gewählt werden, die u. a. den Vorteil bietet, daß sie durch Lehrveranstaltungen eingeführt wird und die meisten Studenten mit ihr vertraut sind.

5. Literaturverzeichnis

/1/ Jerkov, G., Knieriemen, T., "Autonome Mobile Roboter - Einführung und Überblick" Interner Bericht 171/87 im Fachbereich Informatik der Universität Kaiserslautern

/2/ ..., Transcar Produktbeschreibung der Firma Telelift GmbH & Co, 1987

/3/ Braun, P., "Fahrerlos in die Zukunft: automatische Transportsysleme", TÜ Bd. 26 Nr.7/8, 1985

/4/ ..., "Identifikationssysteme in der Produktionslogistik", Logistik im Unternehmen, VDI-Verlag, September4/87

/5/ Drunk, G., "Mobiler autonomer Roboter fährt ohne Draht", VDI-Nachrichten Nr. 13, März 1987

/6/ Schmidt, M., "Montageverkettung mit FTS", ZwF 82 Carl Hanser Verlag, Sept. 1987

/7/ ..., "Neues hochautomatislertes Montagesystem", Sonderdruck aus Daimler-Benz intern, März 1986

/8/ ..., Produktbeschreibung MIDI-ROBOTS, Ramonville-Saint-Agne (F), 1986

/9/ Harmon, S.Y., "Practical Implementation of Autonomous Systems: Problems and Solutions", Preprints of the International Conf. of Intelligent Autonomous Systems, Amsterdam, Dez. 1986

/10/ ..., INTEL "Distibuted Control Modules Databook (BITBUS Specification)", Santa Clara, 84