Diplomarbeit

Konzeption und Aufbau eines teilautonomen Fußballroboters.

Thomas Klute

Diplomarbeit
am Fachbereich Informatik
der Universität Dortmund

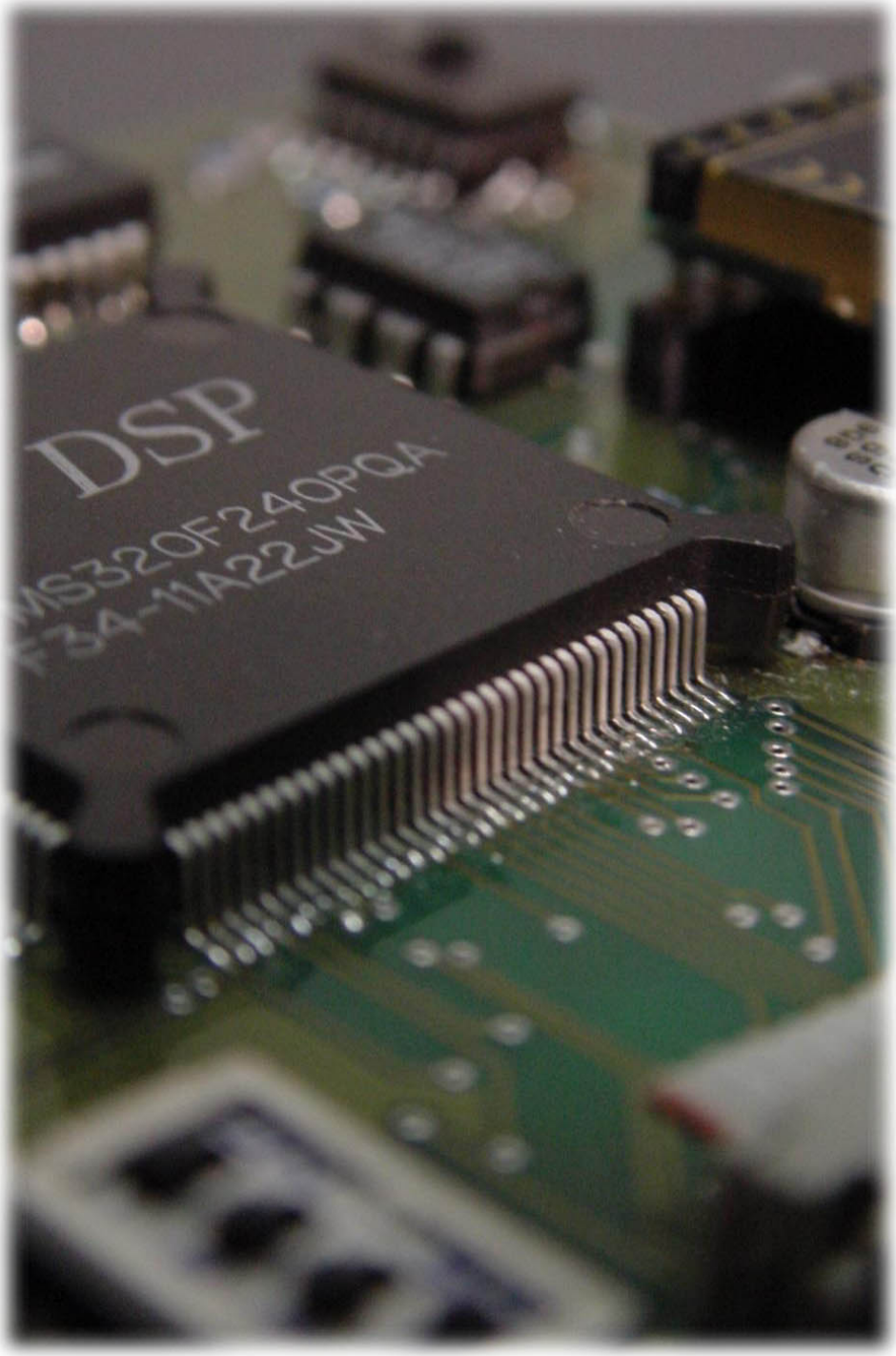
Oktober 2001

Autor:

Thomas Klute

Betreuer:

Prof. Dr. B. Reusch
Dipl.-Inform. K. Liebermann


DSP
MS320F240PQA
34-11A22JW

Zusammenfassung

Roboterfußball hat sich in den letzten Jahren zu einer Standardproblemstellung von Forschungsdisziplinen der künstlichen Intelligenz und der Robotik entwickelt. Das Ergebnis dieser Diplomarbeit besteht in der Konzeption und im Aufbau eines teilautonomen Fußballroboters, der eine Plattform zur Forschung an kooperativen Multiagentensystemen (MAS) und mobilen Robotern bietet und zum Test der Forschungsergebnisse in der FIRA-MiroSot Roboterfußball-Klasse eingesetzt werden kann.
Die Entwicklung des Roboters orientierte sich an Anforderungen, die an mobile Roboter und kooperative autonome Multiagentensysteme gestellt werden. Unter anderem wurden bestehende Roboterfußballsysteme untersucht und die gewonnen Erfahrungen bei der Neuentwicklung berücksichtigt.
Der Hardwareaufbau wird von einer prototypischen Implementierung begleitet und ermöglicht einen grundlegenden Softwarezugriff auf die Hardwarekomponenten des Systems. Der entwickelte Roboter verfügt über einen digitalen Signalprozessor und ausreichenden Hauptspeicher, um die Implementierung von komplexen Softwarekomponenten für die Steuerung und Regelung sowie für die Verarbeitung von Sensordaten zu ermöglichen.
Der Roboter wurde unter Beachtung der Größenanforderungen der FIRA-MiroSot Liga auf eine weitestgehende Unabhängigkeit vom globalen Hostsystem ausgelegt, um die Autonomie des Roboters zu verbessern.
Es wurde eine Funkschnittstelle entwickelt, die eine bidirektionale Roboter-Roboter- und Roboter-Hostsystem-Kommunikation ermöglicht. Ein entsprechendes Protokoll wurde entworfen und implementiert.
Ein zusätzliches lokales Kamerasystem, das dem Roboter zwei Farbkamera-Sensoren zur Objekterkennung zur Verfügung stellt, wurde konzipiert. Die Kamerahardware verfügt über einen weiteren digitalen Signalprozessor und wurde in einer Prototyp-Version realisiert.


DSP
TMS320F24
F34-1NA22

Abstract

Robotsoccer has become a standard problem in artificial intelligence and robotics. The result of this master thesis is the conception and the development of a partly autonomous soccer robot, that offers a platform for research on cooperative multi-agent systems (MAS) and that can be used to test results of research in the FIRA-MiroSot robotsoccer-class.
The robot's development was based on requirements concerning mobile robots and cooperative multi-agent systems. Furthermore, research on existing robotsoccer environments was taken into account for the development of the new robot.
The hardware design was accompanied by a prototypic implementation and offers a basic interface to the hardware components of the system.
The developed robot posesses a digital signal processor and has sufficient memory for the execution of complex software components for robot control and processing of sensor data.
The development aimed at maximum independence on the global hostsystem to achieve a better autonomy of the robot, taking the size restraints of the FIRA-MiroSot league into account.
A radio-frequency interface was developed, enabling bidirectional robot-robot- and robot-host-communication. A communication-protocol was discussed and implemented.
Additionally, a concept for a local camera system was deveolped, providing two color-camera-sensors mounted on the robot. The camera hardware owns a second digital signal processor and was realized as a prototype version.

Zusicherung

Ich versichere, dass ich die vorliegende Arbeit selbständig erstellt und verfasst habe. Die verwendeten Quellen sind vollständig im Literaturverzeichnis aufgeführt und die Zitate als solche gekennzeichnet und mit Quellennachweisen versehen.
Die Diplomarbeit darf in der Bereichsbibliothek Informatik ausgestellt werden und zu Studienzwecken eingesehen und fotokopiert werden, sofern eine Verwendung mit entsprechender Quellenangabe versehen wird. Die weiteren Urheberrechte, insbesondere eine weitergehende Verwendung oder Veröffentlichung, werden hiervon nicht berührt.

Dortmund, im Oktober 2001,

(Thomas Klute)

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

> By the year 2050, develop a team of fully autonomous humanoid robots that can win against the human world soccer champions [Rob01]!

Anforderungen an kooperative Multiagentensysteme unterscheiden sich wesentlich von den Anforderungen an herkömmliche informationsverarbeitende Systeme und sind erst in den letzten Jahren verstärkt Teil wissenschaftlicher Untersuchungen und Entwicklungen geworden. Die Disziplin Roboterfußball bietet eine hervorragende Plattform, um Forschung an kooperativen autonomen Multiagentensystemen und mobilen Robotern durchzuführen und durch den weltweiten spielerischen Wettkampf einen schnelleren Forschungsfortschritt zu erzielen. Es bildeten sich Vereinigungen wie RoboCup und FIRA, die es sich zur Aufgabe machen, diese Entwicklungen zu fördern, eine einheitliche Umgebung für Wettkämpfe festzulegen und diese durchzuführen. Der RoboCup setzt sich dieses Ziel, bis zum Jahr 2050 humanoide Roboter zu entwickeln, die den Fußballweltmeister schlagen können.
Für die resultierenden Roboter sind viele Einsatzgebiete denkbar. Einer der drei Bereiche des RoboCup beschäftigt sich zum Beispiel mit der Anwendung der Forschungsergebnisse für Rettungsroboter.
Der Lehrstuhl Informatik 1 der Universität Dortmund beschäftigt sich seit dem Jahr 1999 mit Roboterfußball, speziell mit der FIRA-MiroSot-Klasse. Die Projektgruppe 340, deren Mitglied der Autor war, erstellte ein solches Roboterfußballsystem unter Verwendung von koreanischen Fußballrobotern vom Typ RobotCavalry; diese erscheinen in vielen Punkten verbesserungswürdig.

1.2 Aufgabenstellung

An diesem Punkt setzt die vorliegende Arbeit an. Es soll ein teilautonomer Fußballroboter für die Verwendung in der MiroSot-Klasse entworfen und die Funktionsweise seiner Module durch eine prototypische Implementierung demonstriert werden. Die Realisierung eines teilautonomen Roboters erlaubt im Gegensatz zur Vollautonomie noch eine Unterstützung des Roboters durch externe Datenquellen wie zum Beispiel eine globale Bildverarbeitung. Die Arbeit umfasst die Diskussion geeigneter Hard- und Softwarekomponenten, den konkreten Hardwareaufbau sowie die Diskussion und prototypische Realisierung einzelner Fähigkeiten.
Nicht zum Thema der Arbeit gehört der Entwurf und Bau eines Gehäuses, Antriebs und sonstiger Arbeiten, die nicht die Elektronik und Software-Implementierung betreffen. Die notwendigen Arbeiten wurden durch eine Studienarbeit im Fachbereich Maschinenbau geleistet.

1.3 Aufbau der Arbeit

In den ersten Kapiteln werden Anforderungen an den zu erstellenden Fußballroboter evaluiert und diskutiert. Anforderungen aus den Bereichen autonome mobile Roboter, eingebettete Systeme und kooperative Multiagentensysteme werden einbezogen.
Es folgt die Realisierung der einzelnen Komponenten des Roboters sowie deren prototypische Implementierung.
In einem weiteren Teil der Arbeit werden beispielhafte Implementierungen für einige wesentliche Fähigkeiten des Roboterfußballsystems diskutiert und realisiert. Insbesondere werden die Bereiche der Kommunikation und der Navigation untersucht.

1.4 Einordnung der Arbeit

Entwurf und Aufbau von autonomen Robotern sind ein interdisziplinäres Thema unter anderem aus den Bereichen Elektronik und Robotik, sowie eingebettete Systeme, autonome Multiagentensysteme und künstliche Intelligenz. Dies wird von dementsprechend vielen Publikationen begleitet. Beispielsweise beschreiben Mondada et al. schon 1993 den Entwurf des Khepera-Roboters [MFI93]. Das Hauptziel bestand in der Erstellung einer Forschungsplattform für autonome mobile Roboter.
Speziell durch die Roboterfußballdisziplin ist die Zahl der Neuentwicklungen von (teil)autonomen mobilen Robotern stark gestiegen. Der Wettkampf stellt hohe Anforderungen an Hard- und Software und in diesen Bereichen

wurde in den letzten Jahren engagiert geforscht und entwickelt. Im Umfeld des RoboCup existieren einige Veröffentlichungen und Dokumentationen bezüglich der Anforderungen, des Entwurfs und der Realisierung von Fußballrobotern. Exemplarisch werden [BGG+99], [Gra99] und [Lie01] aufgeführt. Es existiert eine Entwicklungstendenz in Richtung von vollautonomen Robotern, nicht zuletzt aufgrund der zitierten Zielsetzung des RoboCup.
Diese Arbeit zielt auf die Neuentwicklung eines teilautonomen Robertyps für die FIRA-MiroSot-Klasse. Trotz der erlaubten Teilautonomie soll versucht werden, den Roboter so autonom wie möglich zu gestalten, um weitere Arbeiten in diesem Bereich vorzubereiten.
Der Schwerpunkt und die Innovation der Arbeit liegen in der verstärkten Integration von Sensorik, die eine wesentlichen Voraussetzung für eine weitergehende Autonomie des Roboters ist. Zusätzlich spielt die Miniaturisierung des Roboters im Gegensatz zum RoboCup eine größere Rolle. Der MiroSot-Roboter ist mit einer Kantenlänge von $7,5$ cm kleiner als vergleichbare RoboCup-Roboter. Aus diesem Grund sind in MiroSot-Robotern bisher nur wenige Sensorkomponenten eingebracht worden, und die existierenden Robotermodelle können entsprechend wenige Aufgaben autonom lösen. Diese Arbeit versucht eine Abwägung und Auswahl adäquater Sensoren, um einen autonomeren Robotertyp zu entwerfen. Dabei sollen die Größenvorgaben der MiroSot-Klasse nicht verletzt werden.

Kapitel 2

Grundlagen

Im Folgenden werden wichtige Grundlagen für die verschiedenen Bereiche dieser Arbeit beschrieben. Das Kapitel soll dem Leser einen Überblick über die einbezogenen Technologiebereiche vermitteln und ihn in die Lage versetzen, die spätere Diskussion von geeigneten Hard- und Softwarekomponenten nachzuvollziehen.
Aufgrund der hohen Komplexität eines mobilen Roboterentwurfs können die beschriebenen Grundlagen nur bis zu einem gewissen Detailliertheitsgrad vorgestellt werden. Für eine tiefergehende Beschäftigung wird auf weiterführende Literatur verwiesen.
Nach einer einführenden Beschreibung von Multiagentensystemen und mobilen Robotern wird eine kurze Einführung in die Disziplin des Roboterfußballs und der damit verbundenen Problemstellungen gegeben.

2.1 Kooperative autonome Multiagentensysteme

Unter kooperativen autonomen Multiagentensystemen versteht man mehrere von einander unabhängige Systeme, die mit ihrer Umwelt interagieren und die Ziele mit Hilfe von Zusammenarbeit erreichen sollen. Um dem Leser ein Verständnis zu vermitteln, werden diese Begriffe im Folgenden einzeln beschrieben. Der Begriff des Agenten geht auf Minskis *The society of mind* [Min85] zurück, in dem er die Bedeutung des *mental agent* prägt:

> I'll call „Society of Mind" this scheme in which each mind is made of many smaller processes. These we'll call agents. Each mental agent by itself can only do some simple thing that needs no mind or thought at all. Yet when we join these agents in societies - in certain very special ways - this leads to intelligence.

Franklin und Graesser [FG97] extrahieren 1997 aus vielen inzwischen existierenden Ideen eine Definition für den Begriff eines *autonomen Agenten*:

> An **autonomous agent** is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.

Ein *autonomes Multiagentensystem (MAS)* wird von Franklin und Graesser als eine Gemeinschaft oder Gruppe einzelner solcher autonomer Agenten beschrieben.

2.2 Roboter

Ein Roboter[1] wird vom *Robot Institute of America* 1979 wie folgt definiert:

> A reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through various programmed motions for the performance of a variety of tasks [Dow96].

Die definierten *kooperativen autonomen Multiagentensysteme* (vgl. Abschnitt 2.1) müssen nicht zwangsläufig Roboter sein: im Roboterfußball handelt es sich aber zumeist entweder um simulierte oder reale Roboter.
Die Unterscheidung in *teilautonome* und *(voll)autonome* Roboter betrifft den Grad an Autonomie des Roboters. Ein teilautonomer Roboter kann nicht alle seine Aufgaben gänzlich unabhängig lösen, sondern ist auf externe Unterstützung angewiesen. So können seine Entscheidungen von Daten abhängig sein, die sich nicht über seine lokale Sensorik ermitteln lassen. Dies sind im Roboterfußball beispielsweise Positionsdaten, die durch eine globale Bildverarbeitung ermittelt und von einem Hostrechner zur Verfügung gestellt werden.

2.3 Roboterfußball

Roboterfußball hat sich in den letzten Jahren zu einer Standardproblemstellung der Wissenschaft entwickelt. Es sind unter anderem Disziplinen wie künstliche Intelligenz, Robotik, autonome Multiagentensysteme, Bildverarbeitung sowie Steuerungs- und Regelungstechnik vertreten. Zumeist spielen dabei zwei Gruppen mobiler Roboter gegeneinander nach fußballähnlichen Regeln, die an die Erfordernisse der jeweiligen Robotik angepasst sind. Es

[1] Das Wort *Roboter* stammt vom tschechischen Wort für Leibeigener bzw. Arbeiter ab, benutzt wurde der Begriff erstmals in diesem Kontext in Karel Capeks 1921 erschienenen Theaterstück *Rossum's Universal Robots* [Dow96].

existieren mehrere Vereinigungen für den Roboterfußball, die für verschiedene Klassen Reglements festlegen und die Durchführung von Turnieren organisieren. Erwähnenswert ist zum einen die RoboCup-Initiative, an der sich weltweit ca. 150 Universitäten und Forschungszentren beteiligen und die damit sicherlich die größte Vereinigung dieser Art ist, zum anderen die FIRA, für deren MiroSot-Liga der zu entwickelnde Roboter bestimmt ist.

2.3.1 RoboCup

Die *Robot World Cup Initiative* (kurz: RoboCup) ist eine internationale Initiative für Forschung und Lehre. Diese wurde 1993 zuerst unter dem Namen *Robot J-League* von der Forschergruppe Minoru Asada, Yasuo Kuniyoshi und Hiroaki Kitano gegründet.

> It is an attempt to foster AI and intelligent robotics research by providing a standard problem where wide range of technologies can be integrated and examined, as well as being used for integrated project-oriented education [Rob01].

Der von Robotern gespielte Fußball wird als Problemstellung standardisiert und zur Beurteilung von Forschungsergebnissen der künstlichen Intelligenz und mobiler Roboter herangezogen. Neben der Organisation und Durchführung von Wettkämpfen richtet RoboCup jährlich Konferenzen aus und bietet *educational programs* für die Lehre an.
Die Initiative gliedert sich in drei Bereiche. Der größte Bereich *RoboCupSoccer* befasst sich mit der bereits beschriebenen Roboterfußballdisziplin. Es existieren vier verschiedene Roboterfußballklassen, darunter eine reine Simulatorklasse.
Der *RoboCupRescue* Bereich befasst sich mit der Entwicklung von Rettungsrobotern, die in für Menschen gefährlicher oder schwer zugänglicher Umgebung Rettungsoperationen unterstützen sollen. Auch in diesem Fall existiert neben der Liga für reale Roboter eine reine Simulatorliga. Der *RoboCupJunior* Bereich unterstützt lokale, regionale und internationale Veranstaltungen, um die Roboterfußballdisziplin bei Schülern bekannt zu machen. Der Schwerpunkt liegt im Bereich der Ausbildung und Lehre. Der RoboCup führt in diesem Bereich ebenfalls Wettbewerbe in drei Kategorien durch, neben den bekannten Fußball- und Rettungsroboter können auch tanzende Roboter entworfen werden.

2.3.2 FIRA

Die FIRA (Federation of International Robotsoccer Association) wurde im September 1995 von Jong-Hwan Kim am KAIST (Korea Advanced Institute

of Science and Technology) in Korea gegründet und ist im asiatischen Raum weit verbreitet. FIRA gründete unter anderem die MiroSot-Klasse (MiroSot = Micro-Robot World Cup Soccer Tournament), in welcher der Lehrstuhl Informatik I seit 1999 aktiv ist. Es existieren fünf weitere Kategorien, die sich mehr oder weniger stark in ihren Eigenschaften unterscheiden.

2.3.3 Roboterfußballklassen

Die Zahl und Größe der Roboter sowie der Aufbau des Spielfeldes sind die häufigsten Unterscheidungsmerkmale der Roboterfußballklassen. So existieren verschiedenste Ausprägungen von Miniaturrobotern in der FIRA-NaroSot Liga (4 cm x 4 cm x 5,5 cm) bis zu 45 cm im Durchmesser und 80 cm hohen Robotern in der RoboCup Middle-Size Liga, deren Spielfeld mit einer Größe von 100 m x 64 m eine kleinere Halle belegt. Eine Liga mit humanoiden Robotern bis zu 70 cm Höhe, deren Körperbau dem eines Menschen angenähert werden soll, ist sowohl in der FIRA als auch im RoboCup in Planung. Wettbewerbe in der *RoboCup Humanoid League* sollen erstmals im Jahr 2002 stattfinden.
Es ist eingänglich, dass sich auch die Anschaffungs- und Betriebskosten dieser verschiedenen Klassen stark unterscheiden. Der Lehrstuhl Informatik I entschied sich mit der MiroSot-Klasse für einen Platzbedarf des Spielfeldes von 150 cm x 130 cm, der durch einen üblichen Büroraum gedeckt werden kann und einen erträglichen Kostenfaktor von ca. 2 TDM pro Roboter bei der Anschaffung des Systems. Roboter der RoboCup-Middle-Size Kategorie sind im Anschaffungspreis wesentlich teurer und ihr Platzbedarf ist weitaus größer.
Große Roboter bieten viel Flexibilität in Bezug auf den Einbau von Sensorik und können im Prinzip beliebig komplexe Sensoren sowie ausreichend Rechenkapazität beherbergen. Diese Flexibilität bezüglich der Sensorik ermöglicht die Realisierung von vollautonomen Robotern, zumal die entsprechenden Roboterfußball-Ligen zumeist keine globale Bildverarbeitung oder eine sonstige externe Unterstützung erlauben. Kleinere Roboter sind in der Auswahl der Sensorik stark eingeschränkt und müssen dementsprechend Kompromisse bezüglich ihrer Autonomie hinnehmen; die Unterstützung durch Daten einer globalen Bildverarbeitung ist üblicherweise erlaubt.

MiroSot-Roboterfußball

Roboterfußball wird in der MiroSot-Klasse auf einem 150 cm x 130 cm großen Feld (Abbildung 2.2) wird mit zwei Mannschaften, bestehend aus jeweils drei Robotern gespielt. Die Dauer eines Spiels beträgt zwei mal fünf Minuten.

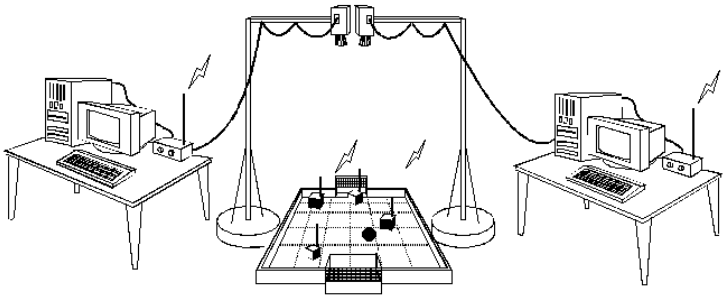


Abbildung 2.1: MiroSot Spielaufbau (Quelle: FIRA)

Die Roboter werden durch ein Hostsystem unterstützt, welches hauptsächlich für die globale Bildverarbeitung zuständig sein soll. Über dem Spielfeld ist zu diesem Zweck eine Kamera montiert, die das gesamte Spielfeld überblickt (Abbildung 2.1). Die Roboter sind anhand von Farbmarkierungen zu unterscheiden; es existieren die Teamfarben (blau bzw. gelb) sowie bis auf wenige Ausnahmen frei wählbare Farben, die jedes Team zur Unterscheidung ihrer Roboter einsetzen darf.
Die Kommunikation zwischen dem Hostsystem und den Robotern wird üblicherweise durch ein Funksystem realisiert, bei dem sich die Mannschaft im Voraus auf die zu verwendenden Funkfrequenzen einigen.
Während des Spiels darf nicht in den Spielverlauf eingegriffen werden, die Systeme bestehend aus den Robotern plus dem Hostcomputer müssen das Spiel autonom durchführen. Das Spiel wird allerdings durch einen Schiedsrichter überwacht, der bei Regelverstößen eingreifen kann und somit für einen fairen Spielverlauf sorgen soll. Im diesem Fall dürfen die Teambetreuer die Systeme manuell stoppen und die resultierenden Feldpositionen der Roboter setzen.
Das genaue Reglement der MiroSot-Klasse ist der FIRA Webseite [FIR01b] zu entnehmen.

2.3.4 Simulatoren

Sowohl RoboCup als auch FIRA bieten neben den Kategorien für reale Roboter auch Simulationsligen an. Roboter werden durch Softwareagenten repräsentiert, die mit einem Simulatorprogramm verbunden werden und einen Spieler innerhalb des Simulators steuern. Der Simulator vermittelt den Softwareagenten ein funktionierendes reales Roboterfußballsystem. Aufgrund dieser Idealisierung ist es wesentlich einfacher möglich, Algorithmen zu testen und komplexe Lernstrategien zu erproben, die zum Beispiel während der Entwicklungs- und Trainingsphase eine hohe Zahl von Lernzyklen benötigen. Reale Roboterfußballumgebungen benötigen zum Betrieb momentan

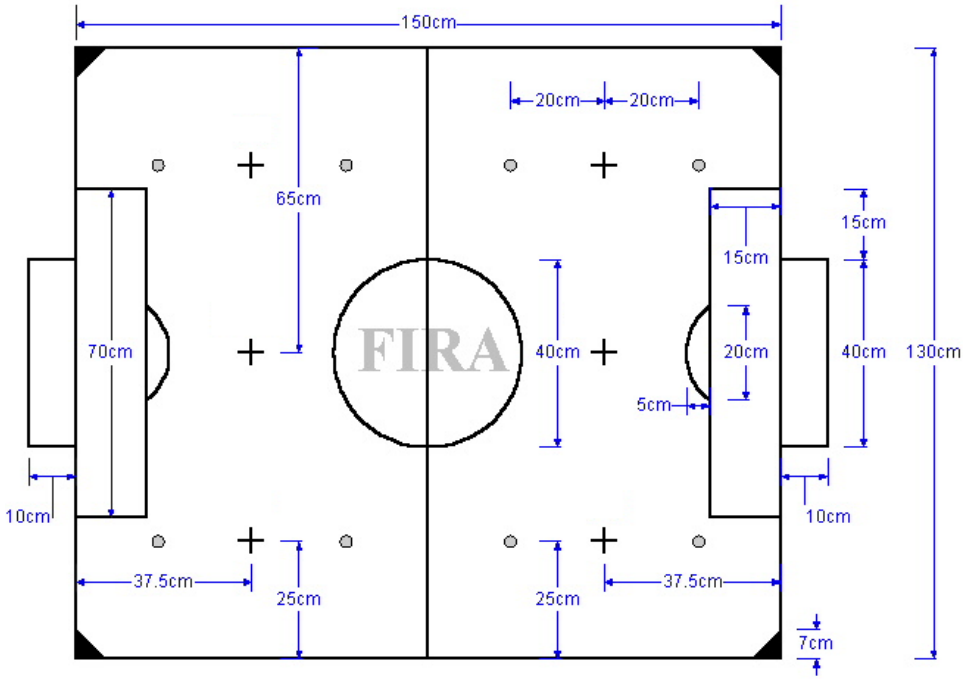


Abbildung 2.2: MiroSot Spielfeld (Quelle: FIRA)

noch einen erhöhten Aufwand und sind für Lern- und Trainingsphasen nur bedingt einsetzbar.

2.4 Navigation

Die Navigation eines autonomen mobilen Roboters erfordert Verfahren zur Messung und Bestimmung der Roboterposition, die im Idealfall vom Roboter selbst ausgeführt werden können.
An dieser Stelle soll ein kurzer Überblick über verschiedene Verfahren und die dafür notwendige Sensorik gegeben werden.
Die möglichen Verfahren zur Positionsbestimmung lassen sich vereinfacht in absolute und relative Verfahren unterteilen. Absolute Verfahren werten Sensordaten aus, die auf die absolute Position des Roboters schließen lassen. Relative Verfahren verwenden Sensordaten, welche die Veränderung der Roboterposition innerhalb diskreter Zeitabschnitte beschreiben. Durch Summierung dieser Veränderungen kann bei den relativen Verfahren auf die absolute Roboterposition geschlossen werden; dementsprechend erfordern die verschiedenen Verfahren absolute beziehungsweise relative Sensorik.

2.4.1 Absolute Verfahren

Trilateration und Triangulation

Ein gutes Beispiel für ein absolutes Verfahren zur Positionsbestimmung und für den dazugehörigen Sensor stellt sicherlich das GPS (global positioning system) dar. Ein GP-System ermöglicht durch den Empfang von GPS-Satellitendaten unter anderem eine Positionsbestimmung auf der Erdoberfläche mit einer üblichen Genauigkeit von 10 bis 20 Metern[2].
Es wird der Zeitversatz beim Empfang der Daten von mindestens drei Satelliten gemessen, so dass auf den Abstand des Empfängers von den Sendepunkten und somit auf die Position des Empfängers geschlossen werden kann (Trilateration). Zur Trilateration werden üblicherweise Funksysteme eingesetzt. Die notwendigen Sensoren besitzen einen sehr komplexen Aufbau und erfordern einen hohen Grad an Genauigkeit. Inzwischen existieren allerdings komplette GPS-Empfänger als ein-Chip Lösung, wie zum Beispiel der TRF5001 von Texas Instruments [Tex00].
Falls nicht die Entfernung sondern die Richtung der Sender bestimmt wird, spricht man von Triangulation. Für dieses Verfahren werden zumeist Infrarot-Sender benutzt. Ein rotierender Sensorempfänger bestimmt die Winkel zu den einzelnen Sendern und kann aus diesen Daten die absolute Position des Sensors ermitteln.

Landmarken

Ein ähnliche Positionsbestimmung kann durch optische Erkennung (zum Beispiel durch einen Kamera-Sensor) von mindestens drei markanten (oder markierten) Punkten (Landmarks) in der Umgebung des Sensors erfolgen. Es wird ebenfalls die Ausrichtung des Sensors zu den jeweiligen Punkten gemessen und die Position anhand dieser Daten bestimmt. Ein solches Verfahren wird im RoboCup durch die Einfärbung der Tore sowie in der Sony 4-legged-Liga (ebenfalls RoboCup) durch weitere Spielfeldmarkierungen unterstützt.

2.4.2 Relative Verfahren

Deduktive Berechnung

Das Verfahren der deduktiven Berechnung (*Dead reckoning* oder *deduction reckoning*) geht auf die Seefahrt zurück. Mit Hilfe von einfachen, relativen Messinstrumenten kann durch wiederholte Messung relativer Bewegungen die absolute Position des Schiffes annähernd berechnet werden. Gemessen

[2] Das verbesserte DGPS (Differential GPS) erreicht Genauigkeiten von 1 bis 3 Metern.

werden die Geschwindigkeit des Schiffes (Knoten), die Ausrichtung (Kompass) und die Zeit (Chronograph). Aus Geschwindigkeit und Zeit lässt sich die zurückgelegte Strecke berechnen, die mit Hilfe der Fahrtrichtung auf die Bewegung (genauer auf den Bewegungsvektor) des Schiffes im gemessenen Zeitraum schließen lässt. Wird der auf diese Weiser bestimmte relative Bewegungsvektor zur bisher berechneten Position addiert, lässt sich die aktuelle Position des Schiffs annähern. Es gelingt die schrittweise Nachführung der absoluten Position des Schiffes wobei sich die Fehler jeder Messung addieren.
Für dieses Verfahren eignen sich alle Sensoren, die Daten aufgrund einer Positionsänderung des Objekts liefern, zum Beispiel Bewegung eines Antriebsrades oder die Drehung des Objekts. Ein häufig verwendeter Sensortyp sind sogenannte Encoder (siehe Abschnitt 2.7), die an Rädern oder Motoren angebracht sind.

Inertiale Navigation

Ein Verfahren ähnlich der deduktiven Berechnung bieten die inertialen Navigationssysteme. Bestimmte Sensoren liefern Daten über den Ruhezustand des Systems. Bei anschließender Bewegung werden die Veränderungen der Sensordaten verwendet, um auf den aktuellen Zustand des Systems, auf seine Position und Ausrichtung zu schließen. Beispiele für entsprechende Sensoren sind Kreisel (Gyroskope) und Beschleunigungs-Sensoren, die in Abschnitt 2.7 vorgestellt werden.

2.5 Antrieb

Für den Antrieb von mobilen Robotern werden üblicherweise Gleichstrommotoren verwendet, wie sie auch im Modellbau eingesetzt werden. Für ihre Ansteuerung werden üblicherweise Elektronikkomponenten eingesetzt, die eine Motorsteuerung mit Hilfe der Pulsweitenmodulation ermöglichen.

2.5.1 Pulsweitenmodulation

Die Ansteuerung von Motorelektronikschaltkreisen wird üblicherweise durch ein Pulsweitenmodulationssignal durchgeführt (PWM, pulse width modulation). Ein PWM-Signal besteht aus einem Wechsel von logisch 1 und logisch 0 mit der Möglichkeit, die Breite des logisch 1 Pulses zu verändern (Abbildung 2.3). Mit diesem Verfahren kann die gewünschte Motorspannung digital kodiert und von der Leistungselektronik der Motorsteuerung ausgewertet werden.

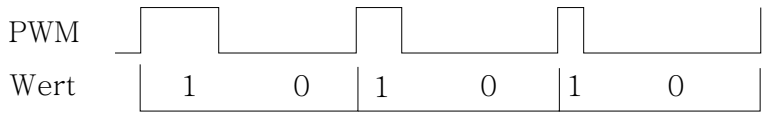


Abbildung 2.3: Pulsweitenmodulation (PWM)

2.5.2 Steuerung und Regelung

Steuerung und Regelung Mögliche Konzepte für die Regelung des Antriebs und die Steuerung des Roboters werden in Kapitel 9 vorgestellt; es werden daher an dieser Stelle einige grundlegende Begriffe der Steuerungs- und Regelungstechnik erläutert:
Der zu regelnde Wert muss messbar oder aus Messwerten berechenbar sein; er stellt den *Ist-Wert* dar, dessen Differenz zum vorgegebenen *Soll-Wert* zum Zeitpunkt t die *Regelabweichung* $e(t)$ ergibt. Die Anwendung des *Reglerfunktionals* $u = F\{e(t)\}$ ergibt die *Stellgröße* u, welche auf die *Regelstrecke* einwirkt. Abbildung 2.4 zeigt die übliche Struktur eines Regelkreises.

lineare Regler

Das Reglerfunktional $F(e(t))$ ist grundsätzlich frei wählbar, es lassen sich diesbezüglich lineare und nicht lineare Regler unterscheiden. Für lineare Regler gilt das von Kiendl [Kie97]$_{S.20f}$ beschriebene *Superpositionsprinzip*:
Seien die Stellgrößenverläufe $u_1(t)$ und $u_2(t)$ den Eingangsgrößenverläufen $e_1(t)$ und $e_2(t)$ zugehörig. Jede Linearkombination der Eingangsgrößenverläufe

$$e(t) \quad = \quad k_1 e_1(t) + k_2 e_2(t) \tag{2.1}$$

erwirkt eine entsprechende Linearkombination der Ausgangsgrößenverläufe

$$u(t) \quad = \quad k_1 u_1(t) + k_2 u_2(t), \tag{2.2}$$

wobei k_1 und k_2 beliebig wählbar sind.
Auf diese Weise kann die Kenntnis über das Regelverhalten in einigen Spezialfällen auf das Verhalten des Reglers im Allgemeinen schließen lassen.

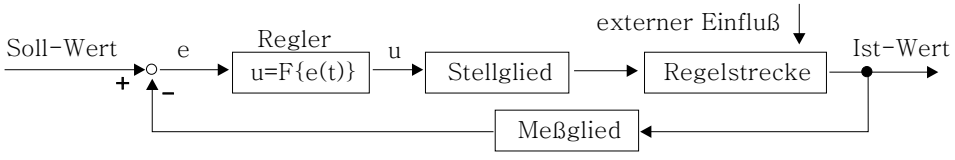


Abbildung 2.4: Strukturbild eines Regelkreises [Kie97]

nichtlineare Regler

Nichtlineare Regler erfüllen das Superpositionsprinzip nicht und sind daher schwieriger in der Einstellung und teilweise komplexer im Aufbau.

2.6 Kommunikation

Die Kooperation zwischen mobilen Robotern erfordert Möglichkeiten der Kommunikation, die in Abschnitt 4.2 diskutiert werden. Bei der Datenkommunikation ist es grundsätzlich sinnvoll, die Umsetzung der Kommunikationsanforderungen in Aufgabenbereiche zu unterteilen und diese hierarchisch zu gliedern. Tanenbaum [Tan96]$_{S.28ff}$ beschreibt mit dem OSI-Schichtmodell eine mögliche Gliederung, die als Grundlage der späteren Realisierung vorgestellt werden soll.

2.6.1 OSI-Schichtmodell

Die verschiedenen Schichten werden im Folgenden erläutert, für eine detaillierte Beschreibung des Designs und der Schnittstellen zwischen den Schichten wird auf [Tan96]$_{Kapitel2-7}$ verwiesen.

2.6.2 Physikalische Schicht

Die physikalische Schicht (physical layer) sorgt für die reale Übermittlung der Daten vom Sender zum Empfänger. Die Umsetzung der Daten in für das Übertragungsmedium adäquate Signale sowie die Rückwandlung der beim Empfänger eintreffenden Signale in Datenbytes wird in dieser Schicht realisiert. Die physikalische Schicht ist auch für die Aufbereitung und Anpassung der Daten an die Erfordernisse der physikalischen Übertragung zuständig, so z.B. das Hinzufügen einer Präambel bei der Funkübertragung, um den Empfänger auf das eintreffende Datenpaket vorzubereiten. Die physikalische Schicht abstrahiert vom tatsächlichen physikalischen Übertragungsverfahren und bietet den übergeordneten Schichten Methoden zum einfachen Senden und Empfangen von Daten.

2.6.3 Datenverbindungsschicht

Die Datenverbindungsschicht (data link layer) benutzt die reinen Datenübertragungsfunktionen der physikalischen Schicht und wandelt diese in eine Datenübertragung, die frei von Übertragungsfehlern ist. Sie nimmt Datenpakete von der Netzwerkschicht entgegen, ergänzt diese um eine Start- und Stopp-Kennzeichnung, fügt eine Prüfsumme hinzu und versendet diese Frames mit Hilfe der physikalischen Schicht. Auf Empfängerseite erhält

die Datenverbindungsschicht einen Datenstrom vom Empfänger und extrahiert anhand der Start- und Stopp-Kennzeichnung einzelne Datenframes, überprüft die Integrität der Daten anhand der Prüfsumme und gibt korrekt empfangene Datenpakete an die Netzwerkschicht weiter. Eine Datenverbindungsschicht kann verschiedene Dienste anbieten, welche die Art der Datenübermittlung betreffen. Grundsätzlich können Daten bei der Übertragung verfälscht werden oder verloren gehen, da die physikalische Schicht nicht garantiert, dass Datenframes korrekt übertragen werden. Die Datenverbindungsschicht kann Dienste implementieren, die einen korrekten Empfang durch Wiederholung von fehlerhaft empfangenen Frames sowie die Bestätigung der Empfängerseite über den korrekten Empfang (Acknowledge) garantieren und somit gegenüber den höheren Schichten eine verlustfreie Datenleitung simulieren. Üblicherweise wird diese Funktionalität aber in einer höheren Schicht implementiert und die Datenverbindungsschicht bietet ausschließlich eine unsichere Datenverbindung an.

2.6.4 Mediumzugriffsschicht

Die Mediumzugriffsschicht (medium access layer) ist eine Unterschicht der Datenverbindungsschicht und koordiniert den Zugriff auf das Übertragungsmedium. Falls auf einem physikalischen Übertragungskanal von mehreren Sendern gleichzeitig gesendet wird, kann es zu Datenverfälschungen, den sogenannten Kollisionen kommen. Es muss versucht werden, die Anzahl von Kollisionen zu minimieren oder diese gänzlich auszuschließen, da Kollisionen immer einen Datenverlust bewirken und die Bandbreite des Übertragungskanals nachteilig beeinflussen. Tanenbaum [Tan96]$_{S.246ff}$ diskutiert verschiedene Möglichkeiten. An dieser Stelle soll vor allem zwischen CSMA-Verfahren (carrier sense multiple access) und TDMA-Verfahren (time division multiple access) unterschieden werden, die Mock und Nett [MN99] genauer auf ihre Eignung für die Echtzeitkommunikation in autonomen Robotersystemen untersuchen. CSMA-Verfahren können einen unbenutzten Kanal erkennen und belegen je nach Algorithmus, abhängig von verschiedenen Faktoren, diesen Kanal, um ein Datenpaket zu versenden. CSMA-Verfahren eignen sich für stark ereignisbasierte Kommunikation, da der Übertragungskanal mit einer gewissen Wahrscheinlichkeit von einem Sender zu jeder Zeit belegt werden kann. Garantiert werden kann die Verfügbarkeit des Kanals allerdings nicht.
TDMA-Verfahren zerlegen den Übertragungskanal in feste Zeitabschnitte und weisen den Kommunikationsteilnehmern bestimmte Zeitabschnitte zu, die diese zum Datenversand nutzen können. TDMA-Verfahren können eine bestimmte Bandbreite garantieren und harten Echtzeitbedingungen genügen. Defizite können eintreten, wenn ein Sender früher als zu seinen festgelegten

Zeitabschnitten senden möchte. TDMA-Verfahren eignen sich daher weniger für ereignisbasierte Kommunikation.

2.6.5 Netzwerkschicht

Die Netzwerkschicht (network layer) kontrolliert die Arbeit des jeweiligen Subnetzes und implementiert Routing-Funktionen auf dem Weg vom Sender zum Empfänger. Falls mehrere Netze an einem Punkt zusammenlaufen, bestimmt die Netzwerkschicht, auf welchen Weg ein Paket weiterversandt wird, um den Empfänger zu erreichen.

2.6.6 Transportschicht

Die Transportschicht (transport layer) nimmt einen Datenstrom einer höher gelegenen Schicht entgegen und unterteilt diesen, falls nötig, in kleinere Pakete. Diese werden zusammen mit Informationen über den Empfänger an die Netzwerkschicht weitergeleitet. Falls die Nachricht - aufgrund ihrer Größe - in mehrere Pakete unterteilt wird, muss von der Transportschicht sichergestellt werden, dass alle Pakete den Empfänger erreichen und vom Empfänger in der richtigen Reihenfolge an die höhergelegenen Schichten weitergegeben werden. Diese Implementierung eines sicheren und verlustfreien Übertragungskanals muss von der Transportschicht geleistet werden, wenn von den bisherigen Schichten kein sicherer und verlustfreier Dienst angeboten wird. Die Implementierung eines solchen Dienstes erfordert die Zwichenspeicherung einer Teilmenge an Datenpaketen, damit diese im Fehlerfall erneut gesendet werden können. Der Empfänger muss den korrekten Empfang eines Datenpakets gegenüber dem Sender bestätigen, damit dieser das zwischengespeicherte Paket verwerfen kann.

2.6.7 Sitzungsschicht

Die Sitzungsschicht (session layer) soll unter anderem für den Auf- und Abbau von Verbindungen und die daraus resultierende Verwaltungsarbeit sorgen. Zusätzlich kann die Sitzungsschicht für token-basierte Protokolle die benötigte Token-Verwaltung übernehmen.

2.6.8 Präsentationsschicht

Die Präsentationsschicht (presentation layer) wird, so Tanenbaum [Tan96]$_{S.33}$, zum Beispiel für die Kodierung (Encoding) von Daten (ASCII, Unicode) verwendet und stellt diese Möglichkeit den Applikationen zur Verfügung; sie ist für die Anpassung von Systemunterschieden in der Datenrepräsentation zuständig und wandelt die übertragenen Datenstrukturen in die im Protokoll festgelegte Form.

2.6.9 Applikationsschicht

Die Applikationsschicht (applikation layer) implementiert Protokolle, die für die Kommunikation häufig benötigt werden; dazu können beispielsweise Protokolle für Terminalapplikationen oder das File Transfer Protocol (FTP) zählen.

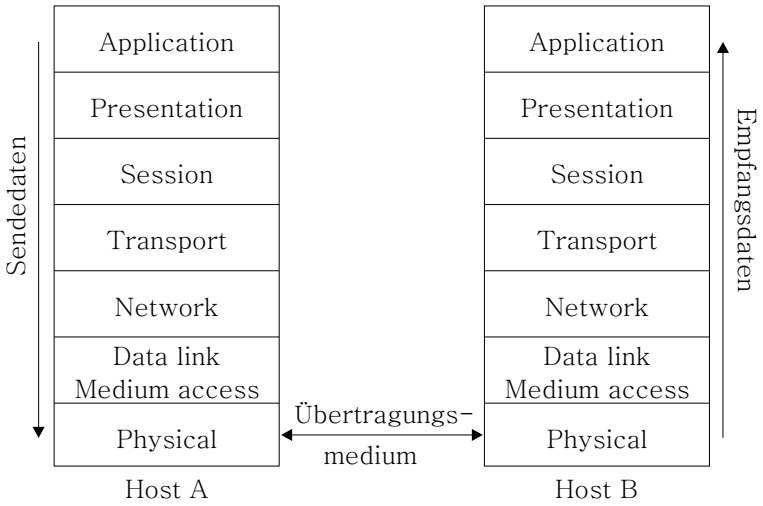


Abbildung 2.5: OSI Schichtmodell

2.7 Sensorik

2.7.1 Radencoder-Sensor

Die Bewegung von (Antriebs-)Rädern kann mit Hilfe von quadratur-Encoder-Sensoren nachvollzogen werden. Zwei versetzt angeordnete Impulsgeber detektieren die Bewegung des Encoders und liefern um 90° versetzte Rechtecksignale - das Quadratursignal (QEP1, QEP2) - aus dem die Drehung des Bezugssystems (Drehwinkel in Impulsen und die Drehrichtung) durch einfache logische Verknüpfung extrahiert werden kann (Abbildung 2.6). Bei ebener Bodenbeschaffenheit und solange das messende Rad den Kontakt zum Boden behält und nicht seitwärts bewegt wird, ist diese Messung sehr exakt und eignet sich als Basisverfahren der Navigation für radgetriebene mobile Roboter.

2.7.2 Infrarot-Sensor

Infrarotnäherungs-Sensoren können in ihrer unmittelbaren Umgebung lichtreflektierende Objekte detektieren; sie besitzen meist eine Sendediode, die Licht im infraroten Spektrum aussendet, und eine Empfängerdiode, welche

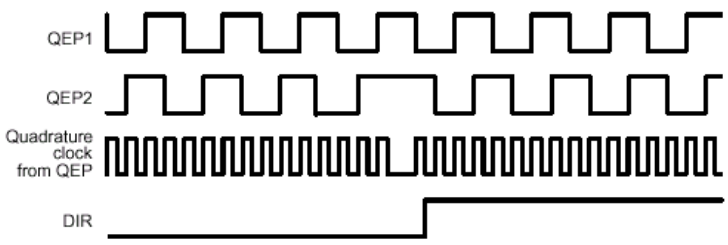


Abbildung 2.6: Quadraturencoding (QEP) (Quelle: Texas Instruments)

die Intensität des reflektierten Lichts misst. Durch Infrarot-Sensoren lassen sich Objekte, die sich in einigen Millimetern Entfernung des Sensors befinden, detektieren. Eine Klassifizierung der Objekte ist allerdings schlecht möglich, da der Sensor nur die Nähe eines Objekts detektiert und keine Informationen über dessen Beschaffenheit liefert.

2.7.3 Laser-Sensoren

Sensoren auf Laserbasis existieren in diversen Ausführungen [JB96]$_{S.101ff}$. Grundsätzlich bestehen diese aus mindestens einer Laserdiode, die entweder gepulstes oder stetiges Laserlicht aussendet. Empfängerdioden messen die Reflexion des ausgesendeten Lichts auf entfernten Objekten und liefern auf diese Art Informationen über die Umgebung des Sensors. Durch eine Messung der Signallaufzeit kann bei gepulsten Lasern zudem die Entfernung zum Objekt betimmt werden. Durch die große Reichweite der Laserdioden können Laser-Sensoren eine Reichweite von mehreren hundert Metern besitzen. Üblicherweise sind diese Sensoren aufgrund ihrer Komplexität größer und damit für den Einsatz in Miniaturrobotern ungeeignet.

2.7.4 Kamera-Sensor

Kamera-Sensoren liefern ein Bild ihrer Umgebung, das mit Hilfe von bildverarbeitenden Systemen ausgewertet werden kann. Kamera-Sensoren werden üblicherweise zur Objekterkennung eingesetzt. Für die Verwendung in einem Roboter bieten sich Miniaturkameras an, die in den letzten Jahren in verschiedensten Versionen erhältlich sind. Übliche Miniaturkameras besitzen zumeist einen analogen Videoausgang, der dann zwecks Auswertung durch einen Framegrabber[3] in ein digitale Repräsentation gewandelt werden muss. Kamera-Sensoren mit digitalem Ausgang sind seltener und im Allgemeinen teurer. Allerdings sind durch die Entwicklung von Webcams (kleinen digitalen Kameras mit geringer Auflösung, die zur Verwendung im Internet

[3]Framegrabber sind Geräte oder Zusatzkarten, die analoge Video-Signale in Digitalbilder wandeln.

entwickelt wurden) digitale Sensoren mit geringerer Auflosung zu günstigen Preisen verfügbar. Zwei grundsätzliche Sensortechnologien werden für Kamera-Sensoren eingesetzt: Einerseits werden die üblichen CCD-Kameras (charge coupeled device) verwendet, die aufgrund ihrer Beschaffenheit eine Versorgungsspannung von ca. 15 Volt benötigen, und andererseits existieren neuere Kameras in CMOS-Technologie, die mit einer Versorgungsspannung von 5 Volt betrieben werden. CMOS-Kameras besitzen im Vergleich zu CCD-Kameras üblicherweise ein etwas stärkeres Bildrauschen.

2.7.5 Kompass-Sensor

Mit Hilfe eines Kompass-Sensors, kann die absolute Ausrichtung des Roboters bestimmt werden, sofern man von einem beschränkten Arbeitsbereich auf der Erdoberfläche, außerhalb der Umgebung der beiden magnetischen Pole, ausgeht. Es existieren sowohl mechanische als auch elektronische Sensoren. Mechanische Sensoren sind grundsätzlich träge und wegen der Abmessungen zu groß für den Einsatz in kleinen mobilen Robotern - wie dem zu entwickelnden MiroSot-Roboter. Elektronische Sensoren sind kostengünstig in Form von integrierten Schaltkreisen verfügbar, die das Erdmagnetfeld durch integrierte Spulen detektieren. Ein solcher Sensor ist beispielsweise der von Philips produzierte KMZ52, mit dem eine Genauigkeit von unter einem Grad erreicht werden kann [Phi00]. Allerdings sind diese Sensoren anfällig gegen elektromagnetische Störungen, die im Bereich der Roboterelektronik bedingt durch die Digitalschaltungen zwangsläufig vorkommen.

2.7.6 Kreisel-Sensor (Gyroskop)

Ein Gyroskop liefert Daten über die Änderung seiner Ausrichtung. Die bekannteste Anwendung von Gyroskop-Sensoren ist die Nachhaltung der Ausrichtung von Flugzeugen während des Flugs. Die verwendeten Sensorsysteme basieren meist auf einer mit hoher Geschwindigkeit rotierenden Masse, die gegenüber ihrem Bezugssystem flexibel aufgehängt ist. Bei Drehung des Bezugssystems bewegt sich die rotierende Masse aufgrund der Massenträgheit nur unwesentlich und ermöglicht die Messung der Drehung. Diese Systeme haben eine große Genauigkeit, sind allerdings entweder groß, schwer und überaus teuer oder haben in kleineren Versionen aufgrund ihrer geringeren Masse eine zu große Drift. Es existieren auf Piezokomponenten basierende kleinere und kostengünstigere Sensoren, die folglich größere Einschränkungen bezüglich ihrer Langzeitgenauigkeit besitzen.

2.7.7 Beschleunigungs-Sensor

Ein Beschleunigungs-Sensor liefert Daten über die Beschleunigung seines Bezugsobjekts. Es existieren kleine piezoelektronische Sensoren, die entlang einer Achse Schwingungsänderungen eines Piezokristalls messen und damit die Beschleunigungswerte in Achsenrichtung bestimmen. Mit Hilfe dieser Sensoren lassen sich sowohl langsame niederfrequente Beschleunigungen als auch hochfrequente Beschleunigungen messen, die zum Beispiel bei Kollisionen mit anderen Objekten auftreten. Langsame Bewegungen sind allerdings aufgrund des Rauschens der Signalwerte häufig nur schlecht messbar. Ein für den Einsatz im MiroSot-Roboter möglicher Sensor ist der von Analog Devices hergestellte biachsiale ADXL202 Sensor [Ana99], der die Beschleunigung auf zwei Achsen in seiner Ebene misst; das Verfahren ist für eine zweidimensionale Bewegung auf einem Roboterfußballfeld ausreichend.

2.8 Echtzeitbedingungen

Ein mobiler Roboter muss während des Betriebs verschiedenste (Teil-) Aufgaben erfüllen. Dazu gehört vor allem die Wahrnehmung seiner Umwelt durch Sensoren, die Auswertung der resultierenden Sensordaten, die Steuerung und Regelung seiner Bewegungen und die Kommunikation mit anderen Robotern. Je nach Art der Aufgabe kann ihr Ergebnis von der Einhaltung gewisser Zeitbedingungen abhängig sein.

2.8.1 Harte Echtzeitbedingungen

Harte Echtzeitbedingungen erfordern die garantierte Einhaltung vorgegebener Zeitbedingungen, da bei Verletzung der Bedingung eine Erfüllung der Aufgabe nicht möglich ist.

2.8.2 Weiche Echtzeitbedingungen

Weiche Echtzeitbedingungen erfordern lediglich die statistische Einhaltung vorgegebener Zeitbedingungen. Eine zeitweilige Verletzung der Bedingung führt nicht direkt zu Fehlverhalten, sondern nur zu einer verminderten Qualität bei der Lösung der Aufgabe.

Kapitel 3

Existierende Robotersysteme

In diesem Abschnitt soll vor allem ein Überblick über verschiedene Robotertypen, deren Fähigkeiten und Funktionsweise, gegeben werden, um dem Leser einen Überblick über die vielfältigen Eigenschaften zu geben. Dies kann und soll nur bis zu einem gewissen Detailgrad geschehen, um die vom Autor getroffenen Design-Entscheidungen deutlich und nachvollziehbar zu machen. Für eine tiefergehende Beschäftigung mit der Robotik wird auf das Buch „Mobile Roboter“ [JF96] verwiesen.

3.1 Cavalry-Roboter

Die Cavalry-Roboter werden von der koreanische Firma *MicroAdventure* [Mic01] hergestellt und waren der Robotertyp, der während der PG340 [PG 00] und teilweise auch während der PG362 [PG 01] eingesetzt wurde. Der Roboter ist für Spiele in der FIRA-MiroSot-Klasse vorgesehen und hat die Größe eines Würfels mit einer Kantenlänge von $7,5$ cm. Dieser Robotertyp war Ausgangspunkt und - bedingt durch seine Nachteile - Motivation die vorliegende Arbeit. Es handelt sich um einen zweiräderigen Roboter, dessen Räder an den Außenseiten angebracht sind (Abbildung 3.1). Jedes Rad ist über ein Getriebe mit einem Mikromotor und einem Radencoder-Sensor verbunden, der zur Kontrolle der Radbewegungen dient. Die Impulsgeber liefern Quadratursignale (siehe Abschnitt 2.7.1), die zur Steuerung des Roboters verwendet werden können. Steuerung und Regelung der Bewegung des Roboters werden von zwei als Hardwarechip realisierte Motion-Controler-ICs LM629 durchgeführt. Diese ICs erhalten eine Soll-Radposition bzw. eine Soll-Geschwindigkeit und regeln den jeweiligen Motor entsprechend. Der Steuerungsalgorithmus innerhalb der ICs entspricht einem industriellen PID-Regler mit der Möglichkeit, Geschwindigkeitsrampen für die Beschleunigung und das Abbremsen der Räder anzugeben.

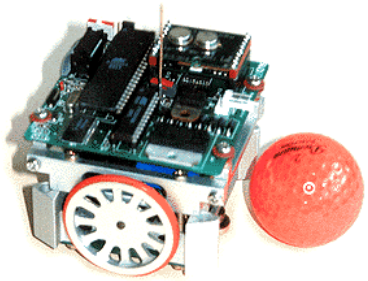


Abbildung 3.1: Cavalry-Roboter der MiroSot-Klasse

Die Motion-Controler-ICs verfügen über je einen Impulsgebereingang und liefern einen PWM-Wert sowie die gewünschte Drehrichtung zur Motorsteuerung. Der Motortreiber L298 sorgt für die Wandlung des PWM-Signals in Spannungspulse für den Motor und schaltet - für den Zeitraum des logisch 1-Wertes des PWM-Signals - die volle Akkuspannung an den Motor. Durch die varriierende Impulsweite und durch die Massenträgheit des Rotors ergibt sich eine unterschiedlich schnelle Drehung des Motors. Bei größtem PWM Wert (durchgängig logisch 1) liegt die Akkuspannung ständig am Motor an, bei kleinstem Wert (durchgängig 0) liegt keine Spannung an.
Eine Drehung des Roboters wird durch unterschiedliche oder entgegengesetzte Geschwindigkeit bzw. Drehrichtung der Räder ermöglicht. Durch die mittige Anordnung der Räder bezogen auf die Längsachse kann der Roboter seine Vorwärts- und Rückwärtsfahrt beliebig vertauschen und - auch wegen der Ballführungseinrichtungen vorne und hinten - Drehungen größer 90° vermeiden, sofern eine Umkehrung der Fahrtrichtung energetisch günstiger ist. Der Roboter kann eine Geschwindigkeit von 106 cm/s erreichen, wobei ohne Absturz der Motion-Controler-ICs maximal die Hälfte der Geschwindigkeit erreichbar ist.
Der Roboter verfügt über einen Funkempfänger BiM der Marke Radiometrix. Mit ihm können serielle Daten empfangen werden, die das Modul des Hostrechners aussendet. Das Funkprotokoll besteht aus einem neun Byte langen Datensatz, der für jeden der drei Roboter im MiroSot den Befehl (STOP, GO) sowie die Geschwindigkeit des linken und des rechten Rades, jeweils in einem Byte kodiert, enthält. Der Roboter kann keine Daten senden.
Kontrolliert wird der Roboter von einem 8-bit Atmel 89C52 Prozessor (5051-Derivat) mit 11MHz Taktfrequenz, 32kByte RAM und 32kByte ROM-Speicher. Der Prozessor kümmert sich ausschließlich um den Funkempfang und die Ansteuerung der Motion-Controler-ICs. Der Roboter verfügt über keine weitere Sensorik und hat daher auch keine Möglichkeit, Sensordaten oder sonstige Informationen per Funk mitzuteilen.

Die Autonomie des Roboters beschränkt sich auf die Einhaltung der mitgeteilten Radgeschwindigkeiten und ist daher als verhältnismäßig gering einzustufen.
Die Stromversorgung des Roboters erfolgt durch einen Akkublock, der im geladenen Zustand 8-10 Volt liefert; er besteht aus einzelnen 1,5 Volt Akkuzellen, die in Reihe geschaltet sind. Ein Betrieb des Roboters ist im Fahrbetrieb 15-20 Minuten möglich, bevor der Akku gewechselt werden muss. Der Roboter verfügt über keine Kontrolle der Akkuspannung und bleibt daher einfach stehen oder verhält sich nicht deterministisch, falls die Akkuspannung zu stark absinkt. Die Spannungsversorgung der Elektronik wird von einem 5 Volt Spannungsregler durchgeführt, der die zu hohe und schwankende Akkuspannung glättet und eine gleichbleibende Spannungsversorgung sicherstellt. Die Motoren dagegen werden über den Motortreiber direkt mit der Akkuspannung versorgt. Eine Versorgung über den Spannungsregler wäre nicht sinnvoll, da die Motoren keine feste Betriebsspannung brauchen und dann nur mit 5 Volt statt maximal mit 10 Volt betrieben würden. Die restliche Energie würde im Spannungsregler in Wärme umgesetzt.
Der Preis eines Cavalry-Roboters beträgt ca. 1000 Euro.

Abbildung 3.2: Motor und Getriebe eines Cavalry-Roboters

3.1.1 Diskussion der Fähigkeiten

Die verwendeten Motoren und Impulsgeber funktionierten leider aufgrund von mechanischen Problemen nicht zufriedenstellend. Durch Verschmutzung des nicht gekapselten Getriebes (Abbildung 3.2) und die einfache Befestigung des Impulsgebers (Heißkleber) wurden diese mechanischen Defizite mit längerer Benutzung erheblich, zumal der Roboter zur Reparatur komplett auseinander gebaut werden muss. Sinnvoll und für die Lebenserwartung nachhaltig ist ein staubgeschütztes Getriebegehäuse und ein in den Motor integrierter Impulsgeber.
Die Motion-Controler-ICs sind sicherlich eine einfach zu handhabende Möglichkeit, mit geringem Aufwand eine gute Steuerung und Regelung zu realisieren; sie sind allerdings in Bezug auf ihre Algorithmen festgelegt. Durch die Anforderung, andere Steuerungskonzepte wie Fuzzy-Steuerungen oder

neuronale Netze zu ermöglichen, ist eine Steuerung und Regelung durch den Prozessor und somit eine reine Softwareregelung die flexiblere wenn auch aufwendigere Variante.
Eine Spannungskontrolle des Akkumulators wäre wünschenswert, da ein leerer Akku sonst erst durch den Ausfall des Roboters bemerkt werden kann oder immer präventiv nach einer bestimmten Zeit ausgewechselt werden muss.
Die Software des Atmel-Prozessors lässt sich ohne Entwicklungsumgebung und ohne Kenntnis des internen Aufbaus nicht ändern: damit sind Funkprotokoll und Fähigkeiten des Roboters festgelegt und die PG340 war nur in der Implementierung der Hostsoftware frei. Folglich musste auch die Ansteuerung von Positionen auf dem Spielfeld mit Hilfe des Hostcomputers und der globalen Bildverarbeitung durchgeführt werden. Dieses Verfahren erwies sich in PG340 als erstens zu langsam und zweitens zu ungenau. Bedingt wurde dieser Nachteil durch Verzögerungen und Ungenauigkeiten in der Bildverarbeitung sowie durch Verzögerungen in der Funkstrecke. Die Unsicherheit des korrekten Empfangs der Befehle durch den Roboter ist ein zusätzlicher Unsicherheitsfaktor. Der beschriebene Regelkreis ist somit weitestgehend ungeeignet, um eine schnelle Bewegung mit geringer Abweichung vom Sollkurs auszuführen; es besteht deshalb ein hoher Bedarf an Verbesserung.
Zusätzlich ist es dem Motion-Controler auf dem Roboter nicht möglich, Abweichungen in der gewünschten Bewegung aufgrund des teilweisen Verlusts an Bodenhaftung (Durchdrehen der Räder) zu erkennen und zu korrigieren. Da die Griffigkeit der Oberfläche je nach Spielfeld und nach dessen Verschmutzung variiert, ist dies eine nicht zu unterschätzende Fähigkeit und damit durchaus wünschenswert.
Die Cavalry-Roboter erreichten beim Einsatz während der PG340 maximal 50% ihrer Höchstgeschwindigkeit; oberhalb dieser kritischen Geschwindigkeit geriet der Roboter mit nahezu einhundertprozentiger Wahrscheinlichkeit in einen nicht kontrollierbaren Systemstatus, aus dem er nur durch einen manuellen Reset „gerettet“ werden konnte. Dem Hersteller soll ein Programmierfehler bei der Ansteuerung der Motion-Controler-ICs unterlaufen sein - so die inoffizielle Aussage des polnischen Teams, die mit dem gleichen Problem zu kämpfen hatten und mit viel Mühe den Prozessorcode analysierten. Aber auch unterhalb dieser kritischen Geschwindigkeit bestand das Risiko dieses Systemabsturzes und machte damit den Umgang mit den Robotern in der Entwicklung und vor allem in Spielen zu einer zeitaufwendigen und frustrierenden Beschäftigung.

3.2 Khepera-Roboter

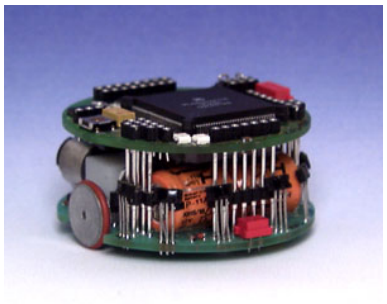


Abbildung 3.3: Khepera-Roboter (Quelle: K-Team)

Die Khepera-Roboter werden seit Jahren erfolgreich im Forschungsbereich autonomer mobiler Roboter eingesetzt. Es existiert neben den Robotern eine Simulatorsoftware, welche die Umgebung des Roboters annähernd gut simuliert. Die Roboter dienen daher zumeist dem Test von Algorithmen, die in Simulatoren erfolgreich entwickelt wurden und mit Hilfe der Khepera-Roboter in die reale Welt übertragen werden. Die Khepera-Website [K-T01] gibt folgende Hintergrundinformationen:

> The Khepera robot was initially developed as part of a research project on artificial intelligence and robotics funded by the Swiss National Science Foundation, and developed in the LAMI lab at EPFL in Lausanne. A hit with other researchers from the start, Khepera was subsequently modified and improved to meet the needs of several research groups within EPFL. Demand for Khepera from external researchers mounted until finally being commercialised by K-Team, a company created by researchers to respond to the needs of fellow researchers in the field of mobile robotics.

Der Khepera-Roboter ist modular als Stecksystem aufgebaut, und besteht aus einem Basismodul (Abbildung 3.3) mit einem Antriebsmodul, das zwei Motoren und Räder, eine Spannungsversorgung sowie die CPU Platine enthält. Zusätzliche Module lassen sich turmartig von oben auf den Roboter stapeln. Es existieren einige Module, die meist mit Sensoren oder Aktoren bestückt sind und je nach Bedarf am Roboter angebracht werden können. Diese Module wurden in Zusammenhang mit Bedürfnissen an Sensorik bei autonomen Robotern entworfen und geben daher gute Hinweise auf eventuell im Roboterfußball verwendbare Sensoren und Bauteile.
Die CPU des Khepera-Roboters ist ein Motorola MC68331 16MHz 32-bit Mikroprozessor mit 512kByte RAM und somit gut geeignet, um komplexe

Programme schnell auszuführen und genügend Speicher zur Verfügung zu stellen.
Die Basisversion des Khepera-Roboters kostet momentan ca. 1700 Euro, das Funkmodul ca. 1000 Euro und das Kamera-Erweiterungs-Board ca. 3800 Euro. Die Khepera-Preise liegen weit über den Cavalry-Roboter-Preisen, allerdings erhält man dafür ein Robotersystem, das eine geeignete Plattform zur Forschung im Bereich autonomer mobiler Roboter bietet. Das Funkmodul

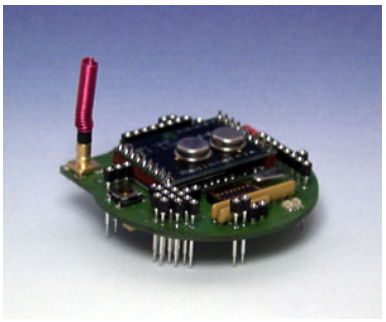


Abbildung 3.4: Khepera Funkmodul (Quelle: K-Team)

des Khepera-Roboters (Abbildung 3.4) enthält, wie die RobotCavalry, ein Radiometrix-BiM-Transceiver[1]. Das Modul ist zwischen Sende- und Empfangsmodus umschaltbar und kann somit zur Kommunikation mit anderen Robotern genutzt werden.
Das Khepera-Visionmodul (Abbildung 3.5) beinhaltet eine Grauwertzeilen-

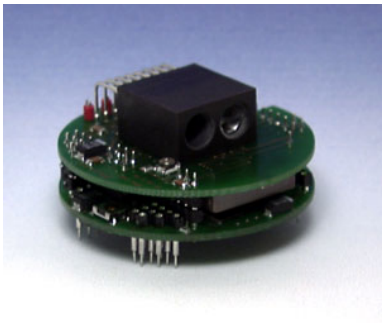


Abbildung 3.5: Khepera Zeilenkamera (Quelle: K-Team)

kamera, die eine horizontale Bildzeile mit 64 Bildpunkten liefert.
Das Khepera Kameramodul (Abbildung 3.6) besteht aus einer Grau- oder Farbkamera, die inzwischen in vielen Elektronikläden als Modul erhältlich ist. Die Kamera besitzt eine übliche Videoauflösung mit 330 Bildzeilen,

[1]Transceiver = Transmitter und Receiver, d.h. ein Modul mit sowohl Sende- als auch Empfangsbetriebsart.

Abbildung 3.6: Khepera Kameramodul (Quelle: K-Team)

benötigt aber einen externen Framegrabber und somit eine Kabelverbindung zu einem externen Bildverarbeitungssytem, da nur über ein analoger Videoausgang verfügbar ist.

3.2.1 Diskussion der Fähigkeiten

Der Khepera-Roboter ist für Roboterfußball zwar grundsätzlich geeignet, aufgrund der schwachen Antriebsauslegung für die MiroSot-Klasse allerdings ungeeignet. Die Geschwindigkeit des Roboters von maximal 60 cm/s ist für den Einsatz in der MiroSot-Klasse mit Geschwindigkeiten von bis zu 160 cm/s wesentlich zu langsam. Eine Ausstattung des Roboters mit einem Funkmodul und weiteren gewünschten Sensoren würde die Größenbeschränkung der MiroSot-Liga verletzen, da der Roboter eine Höhe von 7,5cm nicht überschreiten darf. In der FIRA existiert eine eigene Roboterfußballklasse für Khepera-Roboter, so dass auch diese Roboter für Wettkämpfe eingesetzt werden können.
Der Khepera-Roboter ist allerdings aufgrund seines modularen Aufbaus, wegen seiner Sensorik, Rechenkapazität und seiner Fähigkeit zur Realisierung komplexer Softwaremodule dem RobotCavalry-System weit überlegen. Er ist zudem nicht auf den Bereich des Roboterfußballs fokussiert, sondern durch die Modularität entsprechend den Anforderungen mit unterschiedlichen Sensoren ausstattbar. Der Khepera-Roboter ist ein sehr positives Beispiel für einen Miniaturroboter, dessen Einsatzgebiet die wissenschaftliche Forschung an autonomen mobilen Robotersystemen ist.

3.3 Frauenhofer-AiS-Roboter

Die Fußballroboter des Instituts *Autonome intelligente Systeme* der Frauenhofer-Gesellschaft[2] sind für einen Einsatz in der RoboCup Middle-Size Liga konzipiert und damit wesentlich größer als die bisher vorgestellten Roboter (Abbildung 3.7). Die Roboter sind als vollautonome Systeme konzipiert und besitzen eine Reihe von Sensoren, die ein vollautonomes Verhalten ermöglichen: Eine um 360° schwenkbare Kamera kann die Umgebung des Roboters nach Objekten durchsuchen und auf diese Weise den Ball sowie eigene und generische Roboter lokalisieren. Die Navigation des Roboters wird zusätzlich durch einen Kreisel-Sensor (Gyroskop) unterstützt, das Daten über die Drehung des Roboters liefert.
Die Vermeidung von Zusammenstößen mit anderen Robotern wird durch Entfernungs-Sensoren auf infrarot-Basis und durch auf Druck reagierende Stoßstangen realisiert. Der Roboter verfügt zudem über einen eigenen pneumatischen Schussmechanismus.

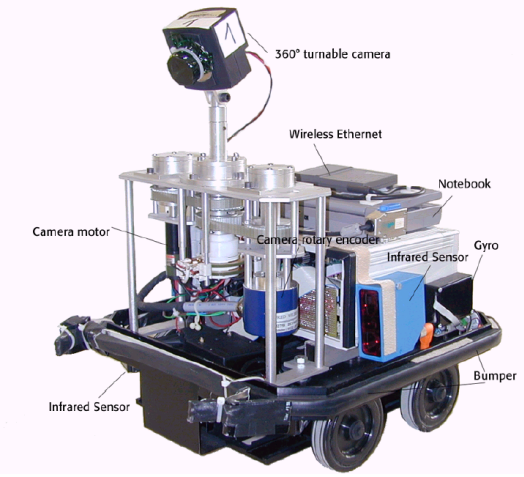


Abbildung 3.7: Fußballroboter der Frauenhofer Gesellschaft

Die Datenverarbeitung übernimmt ein Notebook, das auf den Roboter montiert wird und die Kommunikation zwischen den Robotern wird mit Hilfe von wireless LAN-Schnittstellen ermöglicht.
Der Frauenhofer Roboter ist wesentlich zu groß, als dass seine Komponenten in einem MiroSot-Roboter verwendet werden und als Vorlage der durchzuführenden Entwicklung dienen könnten. Die Konzeption des Roboters als vollautonomes System ist hingegen interessant für die Neuentwicklung. Die

[2]Die Roboter wurden von der GMD - Gesellschaft für Mathematik und Datenverarbeitung - entwickelt, die in diesem Jahr mit der Frauenhofer Gesellschaft funsionierte.

Kombination der Sensorik ermöglicht dem Roboter eine eigenständige Navigation mit Hilfe der Bildverarbeitung, dem Gyroskop und den Radantrieben. Die Erkennung des Balles und anderer Roboter wird ebenfalls mit Hilfe der Kamera durchgeführt. Die notwendige Kommunikation ist bidirektional mit Hilfe eines drahtlosen Netzwerkes möglich. Der Frauenhofer Roboter stellt ein gelungenes Beispiel für einen vollautonomen Fußballroboter dar.

3.4 Ergebnis

Es wurden drei wesentliche, sich in ihrer Autonomie stark unterscheidende Roboterkonzepte vorgestellt: Der ferngesteuerte Cavalry-Roboter stellt die geringste Stufe möglicher Autonomie dar und ist aufgrund seiner Verwendung im bisherigen Roboterfußballsystem der Ausgangspunkt für mögliche Verbesserungen in der Neuentwicklung.
Die Khepera-Roboter sind in der Lage, bestimmte Aufgabenstellungen aufgrund ihrer komplexen Sensorik autonom zu bearbeiten. Sie sind aber aufgrund ihrer kleinen Bauweise in der Integration von Sensorik eingeschränkt.
Die Frauenhofer Fußballroboter stellen als vollautonome Systeme die höchste Stufe möglicher Autonomie dar.
Alle vorgestellten Roboter geben einen Einblick in die verschiedenen Konzepte und ermöglichen im weiteren Verlauf der Arbeit die Diskussion geeigneter Module und Verfahren. Einige der vorgestellten Bauteile könnten in der dargestellten Form weiterverwendet werden, andere hingegen sind beispielsweise aufgrund ihrer Größe in der durchzuführenden Neuentwicklung nicht verwendbar.

Kapitel 4

Ziele der Neuentwicklung

Bei der Neuentwicklung stehen Ziele für die Entwicklung kooperativer autonomer mobiler Roboter und Multiagentensysteme sowie spezifische Ziele, welche den Roboter für die Roboterfußballdisziplin tauglich machen, im Vordergrund. Eine derartige Trennung wird für jeden Bereich vorgenommen und erscheint sinnvoll, da Untersuchungen im Roboterfußball generalisierbar sein sollten und damit auch in anderen Anwendungen der zugrundeliegenden Wissenschaftsbereiche verwendbar sind[1]. Die Entwicklung zielt grundsätzlich auf den Entwurf einer Forschungsplattform für autonome mobile Roboter und kooperative Multiagentensysteme am Beispiel des Roboterfußballs und nicht primär auf die Erstellung eines möglichst guten Fußballroboters. Im Folgenden werden Ziele für die einzelnen Bereiche entwickelt, indem Anforderungen gesammelt und diese anhand bisheriger Erfahrungen mit dem vorliegenden Robotersystem diskutiert werden. Die Anforderungen stellen für ein funktionierendes Roboterfußballsystem notwendige Fähigkeiten dar. Da die Realisierung eines kompletten Roboterfußballsystems aus zeitlichen Gründen weit über die Aufgabenstellung dieser Arbeit hinausgeht, können und sollen diese Anforderungen nur teilweise als Ziele dieser Arbeit gelten. Vielmehr werden aus der Sammlung von Anforderungen Ziele für die Neuentwicklung extrahiert, die eine spätere Realisierung der Anforderungen ermöglichen.

4.1 Navigation

Ein mobiler Roboter, der bezogen auf seine Navigation autonom ist, muss in der Lage sein, innerhalb seiner Arbeitsumgebung eigenständig und ohne fremde Hilfe zu navigieren. Üblicherweise besitzt ein solcher Roboter ein Weltmodell seiner Umgebung und kann seine Position und Ausrichtung

[1] Einer der drei Bereiche der RoboCup Initiative beschäftigt sich mit Rescue-Robots, in der die Forschungsergebnisse eine sinnvolle Anwendung finden.

innerhalb dieses Modells mit einer gewissen Genauigkeit bestimmen. Dies ermöglicht ihm eine kontrollierte Bewegung, so dass ein Anfahren bestimmter Zielpunkte möglich ist.

4.1.1 Anforderungen

Positionsbestimmung

Das Steuerungssystem des Roboters muss seine Position[2] bestimmen können; der absolute Fehler darf dabei einen gewissen Wert nicht überschreiten. Die Messung von Position und Ausrichtung muss ausreichend schnell erfolgen, um bei der gewünschten Maximalgeschwindigkeit eine kontrollierte Bewegung zu ermöglichen.
Teilautonome Roboter erfüllen diese Anforderung in gewisser Weise nicht und sind bei Steuerung oder Sensorik eingeschränkt und daher auf Unterstützung von außen angewiesen. Dies kann z.B. bei der Positionsbestimmung der Fall sein, falls der Roboter über keine absoluten positionsgebenden Sensoren verfügt.
Für den Fußballroboter müssen spezifische Anforderungen bei der Entwicklung berücksichtigt werden.

Bewegung

Der Roboter sollte sich mit ausreichend hoher Geschwindigkeit über das Spielfeld bewegen können. Eine Erhöhung der Geschwindigkeit führt zu einer schlechteren Kontrolle über die Bewegung des Roboters. In Grenzbereichen setzen - bedingt durch die begrenzte Haftreibung der Räder - Schlupfeffekte ein, die je nach Boden- und Radbeschaffenheit die maximale Beschleunigung sowie den minimalen Kurvenradius bei gegebener Geschwindigkeit begrenzen. Das Gewicht des Roboters bedingt die Haftreibung der Räder: bei steigendem Gewicht ergibt sich eine größere Haftreibung, die größere Beschleunigungen ohne Haftungsverlust ermöglicht. Ein schwererer Roboter ist allerdings nur mit größerer Energie zu beschleunigen.
Da die Auswahl von Batterien und Motoren in der MiroSot-Klasse durch die geringen Abmessungen stark eingeschränkt ist, besteht die Optimierung des Verhaltens größtenteils in der Wahl eines guten Reifenbelags und in der Implementierung von guten Steuerungs- und Regelungsalgorithmen.

Drehung

Der Roboter soll möglichst schnell in der Lage sein, seine Fahrtrichtung zu ändern, damit auf veränderte Spielsituationen sofort reagiert werden kann.

[2] Im Folgenden ist die Ausrichtung des Roboters immer impliziert, wenn von der Roboterposition gesprochen wird.

Eine schnelle und genaue Lenkbarkeit ist eine essentielle Voraussetzung. Roboter, die bezogen auf ihre Lenkachse unsymmetrisch aufgebaut sind, und somit in Vorwärtsfahrt und Rückwärtsfahrt unterscheiden, müssen Drehungen bis 180° bewerkstelligen. Roboter, die in dieser Hinsicht symmetrisch ausgelegt sind, können diese Drehungen auf maximal 90° beschränken, indem sie zusätzlich ihre Fahrtrichtung umkehren.

Ballführung

Der Roboter muss in der Lage sein, den Ball auf seinem Weg zu beeinflussen. Die Möglichkeit, aus einem bestimmten Winkel auf den Ball zu fahren und ihn durch die Berührung in die gewünschte Richtung zu lenken, ist allerdings meist nicht ausreichend. Einerseits ist es unter Umständen nicht möglich, aus allen Richtungen den Ball anzufahren, so dass die gewünschte Zielrichtung nicht erreicht werden kann; andererseits erfordern veränderte Spielsituationen eine längerfristige Kontrolle des Balles. Um die Fähigkeit des Ballführens zu ermöglichen, verfügen viele Fußballroboterchassis über Vorrichtungen wie Schaufeln oder Einkerbungen.

Schuss

Der Roboter muss neben der Ballführung auch in der Lage sein, den Ball stärker zu beschleunigen und somit einen Schuss auszuführen. Dies kann entweder durch schnelles Auftreffen des Roboters auf den Ball oder durch eine separate Schussvorrichtung erreicht werden. Schussvorrichtungen des Roboters können den Anlauf ersetzen, der normalerweise verwendet wird, um Geschwindigkeit aufzunehmen. Der Roboter kann somit direkt aus der Ballführung schießen. Schußvorrichtungen bestehen zumeist aus rotierenden Flächen oder Walzen, die im Kontaktfall einen Teil ihrer kinetischen Energie dem Ball übertragen. Der Nachteil derartiger Vorrichtungen ist einzig der zusätzliche Platz- und Energiebedarf.

4.1.2 Diskussion der bisherigen Umsetzung

Der Cavalry-Roboter ist ein differentieller zweiräderiger Roboter (differential wheel robot). Die Bezeichnung deutet auf die Möglichkeit hin, den Roboter durch die Geschwindigkeitsdifferenz seiner Räder lenken zu können. Die Cavalry-Roboter sind in der Lage, die Geschwindigkeiten des linken und rechten Rades per Funk zu empfangen und die Radgeschwindigkeiten durch eine auf dem Roboter vorhandene PID-Regelung anzupassen. Diese begrenzte Steuerungsmöglichkeit der Roboter erfordert die hauptsächliche Implementierung der Steuerung und Regelung auf Hostcomputerseite.

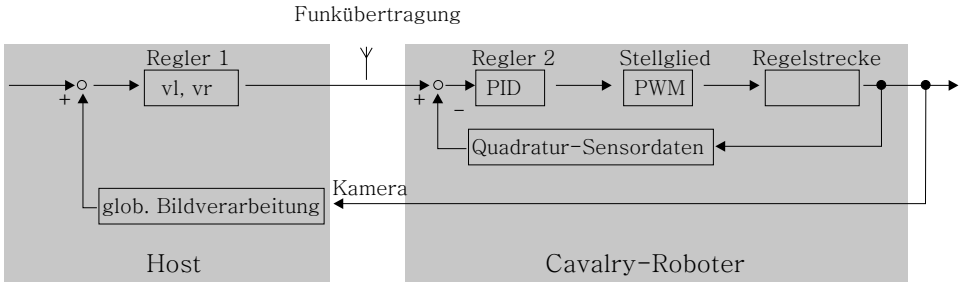


Abbildung 4.1: Regelkreis der Cavalry-Roboter

Die globale Bildverarbeitung ermittelt die aktuelle Position und Ausrichtung des Roboters (Ist-Werte) und der Hostrechner vergleicht diese mit der gewünschten Position und Ausrichtung (Soll-Werte). Abweichungen müssen durch Anpassung der Radgeschwindigkeiten des Roboters korrigiert werden, die per Funk zum Roboter übertragen werden (Abbildung 4.1).

Nachteile

Dieser Regelkreis besitzt einfache, erkennbare Nachteile wie sich im Umgang mit dem Cavalry-Robot System herausstellte.

- Die Erkennung der Roboterposition durch die globale Bildverarbeitung ist ungenau und fehlerbehaftet. Der absolute Fehler lag bei PG340 im Bereich von einigen Millimetern, PG362 konnte dies bis in den Millimeterbereich verbessern. Die Auflösung der Radencoder von ca. $0,26$ Millimetern ist somit mindestens um den Faktor 4-5 genauer.

- Durch schlechte Lichtverhältnisse und unzureichende Kalibrierung der Bildverarbeitung kann es zur fehlerhaften Erkennung der Roboter kommen. Falls ein Roboter in einem Bildframe nicht gefunden wird, existieren bis zum nächsten Bildframe keine Ist-Werte, die zur Steuerung und Regelung herangezogen werden können. Zusätzlich ist eine Vertauschung der Roboter oder eine fehlerhafte Erkennung ihrer ID-Nummer anhand der Farbmarkierungen möglich und wurde von den Bildverarbeitungssystemen der PG340 bzw. der PG361 bis zum heutigen Zeitpunkt beobachtet.[3] Es resultieren offensichtlich falsche Ist-Werte, die verglichen mit den Soll-Werten zu großen Abweichungen und zu entsprechend großen Änderungen der Stellgrößen führen.

- Die Erkennung der Ausrichtung des Roboters durch die globale Bildverarbeitung ist ebenfalls ungenau und fehlerbehaftet. Die Ausrichtung

[3] Diese Nachteile könnten durch eine Plausibilitätsprüfung ausgeschlossen werden, die allerdings bisher nicht implementiert wurde.

des Roboters wird anhand seiner Farbmarkierungen aus den Bilddaten der globalen Kamera berechnet. Zu diesem Zweck wurden verschiedene Farbmarkierungen von PG340 und PG362 verwendet, die unterschiedliche Ergebnisse bezüglich der Erkennungsgenauigkeit lieferten. PG362 konnte eine Ausrichtungserkennung mit bis zu 2° Genauigkeit erreichen.

- Die globale Bildverarbeitung erhält ca. alle 33 Millisekunden ein neues Bild. In dieser Zeit fährt ein Roboter mit Höchstgeschwindigkeit von $160 cm/s$ eine Strecke von $52,8$ Millimetern. Während dieser Zeit findet keinerlei Korrektur statt. Das Zeitverhalten des Regelkreises ist somit nur für Fahrten bei geringer Geschwindigkeit ausreichend; für eine schnellere Bewegung der Roboter ist es zu träge und durch Latenzzeit behaftet.

- Aufgrund der Funkübertragung können Datenpakete verloren gehen oder verfälscht werden. Im Fall eines Verlusts kann der Roboter nicht auf die neuen Daten reagieren und fährt mit den bisherigen Geschwindigkeitseinstellungen weiter. Im schlimmsten Fall kann es zu einer dauerhaft verminderten Qualität des Funkkanals und zu einer weiteren Erhöhung der Latenzzeit der Steuerung kommen.
  Das Risiko der Verfälschung eines Funkpakets ist aufgrund der fehlenden Prüfsumme der übertragenen Daten beim Cavalry-System durchaus als beträchtlich einzustufen. Ein fehlerhaft übertragenes Bit kann je nach Position innerhalb des Datenpaketes die Radgeschwindigkeit und auch die Drehrichtung wesentlich beeinflussen.

- Der vorliegende Regelkreis ist zudem Beispiel für eine zu geringe Autonomie des Roboters. Die im Roboter vorhandenen Radencoder-Sensoren sind für eine Steuerung und Regelung des Roboters auf höherer Ebene geeignet. Die im Cavalry-System durch die PID-Geschwindigkeitsregelung der einzelnen Räder realisierte Lösung erzwingt eine ungeschickte Trennung der Kompetenzen des Roboters und des Hostrechners. Der Roboter ist nicht nur auf externe Daten, sondern auch auf eine ständige externe Regelung angewiesen.

4.1.3 Ziele der Neuentwicklung

Es ergeben sich aus der bisherigen Diskussion folgende Ziele für die Neuentwicklung:

- Der Regelkreis muss soweit wie möglich auf den Roboter verlagert werden, um eine besser Kontrolle über die Bewegungen zu erreichen.

- Der Roboter muss mit Sensorik ausgestattet werden, so dass eine möglichst autonome Navigation ermöglicht wird.
- Der Roboter kann bezüglich seiner Positionsbestimmung vom Hostsystem unterstützt werden und sollte allerdings über einen kurzen Zeitraum ohne Hostsystem navigieren können, um von der Unsicherheit der Funkstrecke nicht negativ beeinflusst zu werden.
- Die Steuerungs- und Regelungsalgorithmen sollten durch Software realisierbar sein, um eine flexibele Wahl geeigneter Steuerungen zu ermöglichen.
- Die im Vorfeld aufgeführten Anforderungen an die Bewegung von Fußballrobotern sollten sich idealerweise mit dem entworfenen System realisieren lassen[4].
- Eine Detektionsmöglichkeit der im Umfeld des Roboters auftretenden Ereignisse scheint ein zusätzliches sinnvolles - wenn auch optionales - Ziel. So könnte die Erkennung von Kollisionen wichtige Informationen über einen eventuellen Genauigkeitsverlust der internen Positionsbestimmung liefern. Außerhalb der Navigation ergeben sich weitere interessante Möglichkeiten; so kann die Erkennung einer Ballberührung Information über die Ausführung eines Schusses liefern und eventuell auf die neue Laufrichtung des Balles schließen lassen.

4.2 Kommunikation

Es soll untersucht werden, welche Kommunikationsanforderungen sich bei der Arbeit mit autonomen Multiagentensystemen und im speziellen bei Fußballrobotern der MiroSot-Klasse ergeben. In einem ersten Schritt werden Anforderungen an die Kommunikation zwischen autonomen Multiagentensystemen gesammelt.

4.2.1 Allgemeine Anforderungen

- Kommunikation ist für eine kooperative Arbeit notwendig, um Absichten mitzuteilen und zu koordinieren (Synchronisation), Rollen- und Aufgabenverteilungen abzusprechen (Verhandlung), Handlungsergebnisse oder sonstige Erkenntnisse (Datenaustausch) mitzuteilen[5]. Im Roboterfußball kann Kommunikation zum Beispiel den Austausch der

[4] Die Realisierung des Ballführens beispielsweise ist äußerst komplex und wird von heutigen Systemen nur teilweise beherrscht: daher würde schon eine Annäherung an die genannten Fähigkeiten eine Verbesserung bedeuten.

[5] Es ist vorerst unerheblich, welches Medium zur Kommunikation verwendet wird.

Ball- und Roboterpositionen oder die Abstimmung der Rollenverteilung auf einzelne Roboter (Stürmer, Passempfänger, Torwart) bedeuten.

- Jedes Agentensystem muss ihm zugeteilte Aufgaben bearbeiten und im Fehlerfall durch jedes der anderen gleichartigen Systeme ersetzbar sein. Deshalb muss jedes System in der Lage sein, bidirektional mit den anderen Systemen zu kommunizieren (falls es mindestens eine Aufgabe gibt, die dies erfordert). Ansonsten wäre eine Austauschbarkeit der Systeme unmöglich und nicht jedes System könnte beliebige Aufgaben übernehmen.

Die Notwendigkeit der Kommunikation zwischen intelligenten Roboteragenten wird von D. Storey [Sto98] unterstrichen:

> A fundamental issue in the ability of a robotic team to cooperate effectively is the level of appropriate inter-agent communication.

Bei teilautonomen Systemen und speziell bei Roboterfußballsystemen in der MiroSot-Klasse lassen sich diese Anforderungen teilweise einschränken, was sich deutlich an der bisherigen Realisierung des Kommunikationsystems erkennen lässt.

4.2.2 Diskussion der bisherigen Umsetzung

Aus Gründen der Einfachheit bietet sich für Roboterfußballsysteme eine Datenübertragung per Funk an[6]. Das Robot-Cavalry-System ermöglicht dies mit Hilfe von Radiometrix-BiM-Transceivern, die an die seriellen Schnittstellen des Hostsystems bzw. der Roboter angebunden sind. Die Datenübertragung wurde als eine festgelegte 1:n-Funkkommunikation realisiert, wobei das Modul des Hostrechners ständig im Transmittermodus und die Module der Roboter ständig im Empfangsmodus betrieben werden. Die eingesetzten Funkmodule sind zwar Transceiver, wodurch eine bidirektionale Kommunikation möglich wäre, allerdings wurde die jeweils benötigte Betriebsart der Module hardwareseitig festgelegt, was zur beschriebenen unidirektionalen 1:n-Kommunikation führt. Da Zustandsmeldungen der einzelnen Roboter somit nicht möglich sind, entstehen zwangsläufig Probleme bei der Klassifizierung von evtl. auftretenden Fehlfunktionen[7]. Das Robot-Cavalry-System geht zudem davon aus, dass der Hostrechner für Strategieplanung und Robotersteuerung zuständig ist und die Roboter nur Geschwindigkeitsbefehle

[6] Eine kabelgebundene Kommunikation ist aus technischer Sicht zweifelsfrei einfacher zu realisieren, allerdings wird die freie Beweglichkeit der Roboter inakzeptabel eingeschränkt.

[7] Es kann z.B. „Akku-leer“ nicht von „kein Funkempfang“ oder „Roboter nicht eingeschaltet“ unterschieden werden, da das Symptom (Roboter bewegt sich nicht) identisch ist.

ausführen. Die Kommunikation zwischen den Robotern ist deshalb nicht notwendig, da alle autonomen Systeme vom Hostrechner simuliert werden. Die Radiometrix-Transceiver-Module des Cavalry-Systems ersetzen eine kabelgebundene serielle Kommunikation durch eine kabellose Funkübertragung. Die Geschwindigkeit der Übertragung ist auf 9,6kBit/s festgelegt.

Die vorliegende vereinfachte Realisierung begründet sich wie folgt:

- Da die Roboter im MiroSot-System nur teilautonom agieren müssen, ist es möglich, die Aufgaben so zu wählen, dass keine Daten via Funk gesendet werden müssen. In diesem Fall beziehen die Roboter alle benötigten Daten, um ihre Handlungen auszuführen, von einem externen Hostsystem. Zusätzlich existiert eine Art der visuellen Kontrolle der Roboter durch das Hostsystem, was Schlüsse auf Erfolg oder Misserfolg der Aufgaben ermöglicht. Die visuelle Kontrolle ersetzt den Funk-Rückkanal für (und ausschließlich für) Aufgaben, die eine direkte Bewegung des Roboters zur Folge haben. Diese Einschränkung der Aufgaben, über deren Ergebnis der Hostrechner Rückmeldung erhält, ist eine der Schwächen des vorliegenden RobotCavalry-Systems.

- Im MiroSot-System ist eine unidirektionale Funkkommunikation mit indirekter visueller Kontrolle durch die globale Bildverarbeitung bei den meisten Teams zu beobachten.

Diese unidirektionale Funkkommunikation ist durch einen einfacheren Hardwareaufbau und einen unkomplizierten Aufbau des Funkprotokolls geprägt und im Grundsatz zu befürworten. Allerdings „verführt“ diese Lösung zu unsauberen Implementierungen der Multiagentensysteme. Vielfach wird auf dem Hostrechner, der über alle notwendigen Daten aufgrund der visuellen Auswertung verfügt, eine globale Strategieplanung implementiert und deren Ergebnis an die Roboter in Form von Bewegungsbefehlen gefunkt. Es existieren dann kaum jene Eigenschaften, die bei autonomen Multiagentensystemen gefordert werden, nämlich die Unabhängigkeit der einzelnen Systeme. Auf diese Weise lässt sich sicherlich ein gutes Roboterfußballsystem erstellen, der wissenschaftliche Aspekt dieser Disziplin tritt allerdings in den Hintergrund[8].
Die RoboCup-Simulatorliga gibt - um dies zu vermeiden - eine bestimmte Systemarchitektur vor. Die Klienten (Spieler) werden an diese Architektur angebunden, die bestimmte Möglichkeiten der Kommunikation zwischen den

[8] Diese Bewertung soll keine Kritik an derartigen MiroSot-Teams sein, sondern lediglich der Hinweis, dass das MiroSot-System zu einer vereinfachten Art der Implementierung einlädt und somit ein wissenschaftlicher Wettkampf von Multiagentensystemen nur bei Kenntnis der jeweiligen Implementierungen zu beurteilen ist.

Klienten zur Verfügung stellt. Es existiert ein „Gentlemen's Agreement", wie von J. Kummrneje [Kum99] beschrieben, das die Zusicherung enthält, nur mit Hilfe der vom RoboCup-Server vorgegebenen Möglichkeiten zu kommunizieren und insbesondere keine eigene inter-Prozess-Kommunikation zwischen den Klientensystemen zu tolerieren. Auf den teilautonomen Aufbau des MiroSot-Systems übertragen bedeutet dies, dass ein Hostsystem ausschließlich Aufgaben übernimmt, die in den Bereich Bildverarbeitung und Positionserkennung (eigene Roboter, gegnerische Roboter sowie Ball) fallen. Eine undeutliche Formulierung findet sich in den MiroSot Regeln [FIR01a]:

> One host computer per team, mainly dedicated to vision processing and for location identifying, can be used.

Damit wird eine zentrale Strategieplanung nicht explizit ausgeschlossen.
Zwei weitere Nachteile einer unidirektionalen Funkkommunikation seien kurz genannt:

- Die Roboter können keine Fehlfunktionen und Systemzustände melden (z.B. Absturz und Reset des Systems, Akkustatus) und sind daher schlecht handhabbar. Diese Einschränkung ließe sich durch optische Signalgeber (z.B. LEDs) auf dem Roboter zumindest für die spielbedienenden menschlichen Wesen ausgleichen, allerdings nicht für den Hostrechner, der z.B. mit Hilfe dieser Informationen den Roboter mit geringstem Akku-Ladezustand als Torwart verwenden könnte.
- Es können keine Sensordaten und daraus resultierende Informationen übertragen werden, die für die anderen Systeme oder den Hostrechner interessant wären (zum Beispiel Ballkontakt, oder Beschleunigungs-Sensordaten zu Trainings- und Entwicklungszwecken oder das Kamerabild der lokalen Kameras).

4.2.3 Ziele der Neuentwicklung

Es ergeben sich aus der bisherigen Diskussion für die zu entwerfende Kommunikationsschnittstelle die im Folgenden beschriebenen Ziele:

Empfangsmöglichkeit: jedes System kann Daten der anderen Systeme empfangen.

Sendemöglichkeit: jedes System kann Daten an eines oder mehrere der anderen Systeme senden.

Bandbreite: die Bandbreite der Kommunikationslösung muss ausreichen, um die notwendige Kommunikation innerhalb des Systems im Spiel- und Entwicklungsbetrieb zu ermöglichen.

Zeitverhalten: jedes System kann unter Normalumständen innerhalb eines Zeitraums Δt mit einer Wahrscheinlichkeit p das Senderecht und die Aufmerksamkeit der Empfängersysteme erlangen.

Fehlertoleranz: Übertragungsfehler durch Funkstörungen oder gleichzeitiges Senden mehrerer Roboter sollten erkannt und korrigiert werden können.

4.3 Rechenleistung

Autonome mobile Roboter müssen viele verschiedene, für die Erfüllung ihrer Aufgaben notwendige Algorithmen ausführen und diese Ausführung koordinieren. Für gewisse Aufgaben können Prioritäts- und Zeitvorgaben bezüglich ihrer Ausführungsdauer und Häufigkeit existieren. Eine Fehlerbehandlung innerhalb des Roboters muss die Funktionsfähigkeit des Roboters auch im Fehlerfall einzelner Module gewährleisten. Ziel ist es, eine Bearbeitung verschiedenster bei autonomen mobilen Robotern notwendigen und sinnvollen Aufgaben zu gewährleisten und Mechanismen für die koordinierte Bearbeitung dieser Aufgaben zu finden. Dieses Profil korreliert mit Anforderungen, die üblicherweise an eingebettete Systeme gestellt werden. Dies wird im Kapitel 8 diskutiert.

4.4 Systemspezifische Ziele

Im Folgenden werden weitere detaillierte Ziele gesammelt, die für den zu entwickelnden Fußballroboter spezifisch sind und nicht den bisher angeführten Kategorien zugehören:

4.4.1 Regelkonformität

Wesentliche Eigenschaften des Roboters werden durch das Reglement der MiroSot-Klasse vorgegeben [FIR01a]: vor allem die Größe und die Form, welche die Größe eines Würfels mit $7,5$ cm Kantenlänge nicht überschreiten darf.
Zusätzlich darf der Ball nur bis zu 30 % vom Roboter verdeckt werden, da der Ball ansonsten nicht mehr von der globalen Bildverarbeitung zu sehen wäre. Diese Einschränkungen beziehen sich größtenteils auf den Entwurf des Chassis des Roboters und deshalb werden in der vorliegenden Arbeit nicht diskutiert, obwohl sie an anderer Stelle natürlich starken Einfluss auf die Auswahl der Sensorik und die Größe der einsetzbaren Elekronikkomponenten haben.

Kapitel 5

Hardwareentwurf

Die Neuentwicklung eines Fußballroboters und die damit verbundene Auswahl der zu verwendenden Bauteile - zum Beispiel Prozessor, Speicher, Sensoren und Aktoren - ist ein komplexer Prozess. Dies liegt unter anderem in den vielfältigen Anforderungen an das Gesamtsystem, in den verschiedenen Fähigkeiten und Anforderungen der einzelnen Bauteile, sowie in den Platz- und Stromversorgungsvorgaben der Elektronik begründet: die Designentscheidungen für oder gegen ein bestimmtes Bauteil können nicht losgelöst vom Rest des Designs betrachtet werden. Im folgenden Kapitel wird die Methodik des Vorgehens während der Entwurfsphase erläutert. Anschließend werden Designentscheidungen, die daraus resultierenden Bauteile und ihre Einbettung in das Gesamtsystem im Detail diskutiert.

5.1 Methodik

Die Entwicklung des Fußballroboters lief grundsätzlich in folgenden Schritten ab:

Evaluierungsphase: Anforderungen an die zu realisierenden Komponenten werden unter Berücksichtigung der gewünschten Ziele und bisheriger Erfahrungen erarbeitet. Durch die Rahmenbedingungen begrenzte Möglichkeiten werden evaluiert, insbesondere wird eine Recherche nach möglichen Bauteilen, Sensor- und Aktorkomponenten durchgeführt.

Planungsphase: Die Verfügbarkeit der gewünschten Bauteile - in der gewünschten Gehäuseart - wird bei Elektronikdistributoren geprüft. Falls keine Bezugsmöglichkeiten existieren, muss in einer wiederholten Evaluierungsphase nach Alternativen gesucht werden.

Schaltungsentwurf: Die Schaltpläne werden mit Hilfe des Layout- und Schaltplaneditors EAGLE entworfen.

Platinenentwurf: Die Platinen werden mit Hilfe des Layout- und Schaltplaneditors EAGLE anhand des erstellten Schaltplanes entworfen. Die Detaileigenschaften der Bauteile - wie die Pinbelegung der ICs und die Gehäuseart - fließen in die Entwicklung ein. Zusätzlich muss die Anordnung der Bauteile unter Berücksichtigung der spezifischen Bauteilbedürfnisse (wie zum Beispiel die Anordnung der Sensorbauteile) realisiert werden. Platzprobleme können nachträglich zu geringfügigen Änderungen am Schaltplan und an den Bauteilen führen.

Prototyp: Anhand der erstellten Platinenlayouts werden Prototypplatinen durch einen externen Dienstleister produziert.

Bestückung und Test: Die Platinen werden in der Reihenfolge der Relevanz der Bauteile für einen Testbetrieb bestückt. Sobald eine testfähige Bestückung vorliegt, kann die Funktionsfähigkeit der Komponenten getestet werden. Aufgrund dieses Vorgehens wird eine frühe Fehlererkennung sichergestellt und ein schrittweiser kontrollierter Aufbau der Schaltung ermöglicht.

Prototypische Implementierung: Die prototypische Implementierung ermöglicht den Test der einzelnen Komponenten des Roboters hinsichtlich ihrer grundlegenden Funktionsfähigkeit. Dies gilt besonders für die Hardwareansteuerung der einzelnen Sensoren und Aktoren. Erste Rückschlüsse auf die Realisierbarkeit der gewünschten Fähigkeiten können gezogen werden. Die Ursachen für auftretende Funktionsfehler müssen geklärt und erforderliche Änderungen am Schaltplan und an den Platinenlayouts vorgenommen werden. Die erneute Produktion einer Prototypplatine kann notwendig werden.

Platinenherstellung: Sobald alle Komponenten erfolgreich auf ihre Funktionsfähigkeit getestet sind, erfolgt die Produktion der Platinen anhand der korrekten Platinenlayouts.

Implementierung der Softwaremodule: Die abschließende Implementierung der Softwaremodule ermöglicht die Demonstration von ausgesuchten Fähigkeiten im Detail. Eine spätere Integration in den Spielbetrieb ist möglich.

Re-Design: Trotz guter Planungen können innerhalb der einzelnen Phasen Probleme auftreten, die in früheren Phasen nur schlecht und gar nicht zu erkennen waren. In diesem Fall kann ein Re-Design notwendig werden, um bestehende Probleme zu beheben. Zusätzlich kann es - aufgrund von Erfahrungen während der Entwicklung - sinnvoll sein, Verbesserungsvorschläge zu verarbeiten und den Roboteraufbau auf diese Weise über mehrere Versionen zu optimieren.

5.2 Designentscheidungen

Um die in Kapitel 4 erarbeiteten Ziele erreichen zu können, sind eine Reihe von Designentscheidungen bezüglich der Hardware-Realisierung zu treffen. Unter einer Designentscheidung ist beispielsweise die Auswahl eines Prozessors aus den momentan auf dem Markt befindlichen Modellen zu verstehen. Diese Entscheidungen sind von Bedeutung, da sie Einfluss auf die späteren Fähigkeiten des Systems und insbesondere auf die Erfüllbarkeit der in Kapitel 4 gesammelten Anforderungen an ein Roboterfußballsystem haben. Entscheidungen für die eine oder andere Komponente begründen sich unter anderem durch die Abwägung des Kosten/Nutzenverhältnisses und beinhalten zum Beispiel den Platzbedarf, den Stromverbrauch, die Kosten sowie die Verfügbarkeit der gewünschten Komponenten. Im folgenden Abschnitt wird die Auswahl aktuell verfügbarer und geeigneter Komponenten wie Prozessoren, Sensoren und Aktoren diskutiert. Die Beschreibung kann aufgrund der Komplexität dieses Bereiches nicht alle Möglichkeiten und Details vermitteln: vielmehr soll der Leser durch eine ausreichende Betrachtung der Möglichkeiten und eine kurze Begründung der Auswahl ein nachvollziehbares Bild des Hardware-Entwurfs vermittelt werden.
Für eine weitergehende Beschäftigung mit der Robotikmaterie wird auf das Buch *Mobile Robots* [JF96] verwiesen, das bei Addison Wesley erschienen ist. Zusätzlich wird auf das Buch *Where am I? Sensors and Methods for Mobile Robot Positioning* von J.Borenstein [JB96] hingewiesen. Beide Werke sind im Detail nicht mehr ganz aktuell; beschreiben aber vor allem grundlegende Technologien und Verfahren der Robotik und Sensorik, die keinen wesentlichen Änderungen unterliegen.

5.3 Hauptsystem

Das Hauptsystem des Roboters stellt sicherlich, bedingt durch die Auswahl eines geeigneten Prozessors, die schwierigste Designentscheidung dar. Der Prozessor muss in der Lage sein, mit allen ausgewählten Peripheriebauteilen effizient zusammenzuarbeiten; dazu gehören die Anbindung der Sensoren und somit das Einlesen von Sensordaten, die Anbindung der Aktoren und somit die Ansteuerung der Motoren sowie die Ansteuerung der Kommunikationsschnittstelle und somit des Funkmoduls.

5.3.1 Prozessor

Die Anforderungen an den Prozessor sind im Detail aufgeführt (Tabelle 5.1). Die Priorität der Anforderungen nimmt nach unten ab, da im unteren Bereich eher optionale Designziele aufgeführt sind. Die Rechenleistung

Anforderungen an den Prozessor			
Bauteil	*Eingänge*	*Ausgänge*	*Funktion*
RAM-Speicher			Datenspeicher
Flash-Speicher			Programmspeicher (nicht flüchtig)
Hostrechner	JTAG[2]	JTAG	Entwicklungs-umgebung
Radencoder L	1 QEP		Radbewegung L
Radencoder R	1 QEP		Radbewegung R
Motortreiber		2 PWM 2 digital	Geschwindigkeit L/R Drehrichtung L/R
Funkmodul	1 seriell 1 digital	1 seriell 1 digital	Kommunikation Carrierdetect Betriebsmodus
DIP-Schalter	3 digital		3-bit Roboter-ID
Beschleunigungs-sensor	2 analog 2 PWM (alt.)		Beschl. X/Y (analog) Beschl. X/Y (PWM)
Akkuspannungs-sensor	1 analog		
Kamera-DSP	1 seriell	1 seriell	
Temperatur-Sensor	1 analog		

Tabelle 5.1: Anforderungen an den Prozessor

des Prozessors muss ausreichen, um interne Aufgaben - wie das Einlesen der Sensordaten oder Ausgabe der Aktordaten - zu lösen und die zusätzliche Ausführung von mehreren, komplexen Algorithmen - zum Beispiel für die Steuerung und Regelung oder für die Kommunikation - durch ausreichende Rechenleistung zu ermöglichen. Die gleiche Anforderung wird an den Hauptspeicher des Systems gestellt, der Messdaten aufnehmen und daher eine ausreichende Größe besitzen muss. Weiterhin ist ein Flash-Speicherbereich[1] notwendig, um Programmcode nicht flüchtig im Robotersystem abzulegen, so dass der Programmcode beim Einschalten ausgeführt wird.

Eine der ersten Entscheidungen während der Evaluierung möglicher Prozessoren betrifft die Busbreite[3] der Prozessoren. 8-, 16- oder 32-bit Prozessoren sind verfügbar, die sich in ihren Eigenschaften grundsätzlich unterscheiden. Die Menge an adressierbarem Speicher und die Rechenleistung steigt mit größerer Busbreite; allerdings nimmt auch die Komplexität der Bauteile

[1] Flash-Speicher ist ein nicht-flüchtiger Speicher, der auch beim Ausschalten des Systems nicht verloren ist.

[3] Die Busbreite gibt die Anzahl an Adress- und Datenleitungen an, die für externe Speicherzugriffe genutzt werden

zu und vor allem die Zahl der Anschlüsse am Gehäuse. 8-bit Prozessoren erfüllen die Anforderungen nach ausreichend großem Hauptspeicher nicht, während 16- und 32-bit Architekturen diesen genügen.
Der zur Verfügung stehende Platz auf der Hauptplatine - die neben dem Prozessor noch eventuelle externe Speicherbausteine, die Kommunikationsschnittstelle und eventuelle Sensorik beherbergen muss - begrenzt die Suche nach geeigneten Prozessoren auf 16-bit Varianten. 32-bit Prozessoren besitzen bei Erfüllung der Anforderungen aus Abschnitt 5.1 eine große Anschlusszahl, die spezielle - dem Autor nicht zur Verfügung stehende - Bestückungstechniken erfordern. Zudem ist die Größe nur schlecht in das vorgesehene Layout integrierbar.
Nach Einschränkung auf eine kleine Anzahl von 16-bit Prozessoren der Hersteller Motorola und Texas Instruments erfolgten Anfragen bei den Herstellern bezüglich einer Entwicklungsunterstützung, die von Texas Instruments positiv beantwortet wurde. DSPs[4] des Typs TMS320F240 [Tex96a] sowie zur Entwicklung notwendige Hard- und Software wurden von Texas Instruments zur Verfügung gestellt. Die angebotene Unterstützung erleichterte die Entscheidung für einen Prozessortyp, zumal der TMS320F240 der „Wunschkandidat“ für die Entwicklung war. Er erfüllt die bereits genannten Anforderungen bis auf den zweiten QEP-Dekodiereingang für den zweiten Radencoder-Sensor. Dieser Nachteil lässt sich mit Hilfe des Up-/Down-Counters des DSPs beheben: das Sensorsignal wird mit Hilfe eines kleinen externen Dekodierschaltkreises in - für den DSP verwertbare - Eingangssignale umgewandelt.
Der DSP besitzt 132 Anschlüsse und ist in einem PQFP-Gehäuse (plastic quad flatpack) verfügbar, das noch mit Hilfe konventioneller Bestückungstechnik verarbeitet werden kann.

5.3.2 Speichertyp und -größe

Der TMS320F240-DSP verfügt nur über einen internen Daten-RAM-Speicher von 544 Worten (1 Wort = 16 Bits), was die gestellten Anforderungen an das Hauptsystem bei weitem nicht erfüllt: bereits die Speicherung von Messdaten kann schnell mehrere kWorte[5] Speicherplatz benötigen. Der DSP verfügt über einen internen Flash-Programm-Speicher von 16 kWorten, der mit Hilfe spezieller Software beschrieben werden kann. Die Größe des nicht-flüchtigen Flash-Speichers reicht aus, um komplexe Softwaremodule aufzunehmen.
Der Hauptspeicher des DSPs kann mit Hilfe des externen Speicherbusses auf bis zu 224 kWorte an externem RAM-Speicher erweitert werden. Es handelt sich dabei um 64 kWorte Daten-, 64 kWorte Programmspeicher, 64 kWor-

[4] DSP = digital signal processor
[5] 1 kWort = 1024 Worte

te IO-Speicher und 32 kWorte globalen Speicher. Der DSP muss zu diesem Zweck durch einen externen Speicherbaustein ergänzt werden.
Die Ergänzung des Datenspeichers ist zwingend notwendig, um die gestellten Anforderungen zu erfüllen. Durch zusätzlichen Programmspeicher kann das System - während des Betriebs - um Softwaremodule ergänzt werden, was aufgrund der verschiedenen Fähigkeiten und Steuerungsmöglichkeiten eines mobilen Roboters eine durchaus interessante Erweiterung darstellt. Das Betriebssystem des Roboters mit den grundlegenden Softwaremodulen muss natürlich weiterhin in den Bereich des Flash-Speichers passen.
Der Speicherbus des DSPs kann mit schnellem statischem RAM-Speicher (engl: *fast SRAM*) zusammenarbeiten. Die Dekodierung der Zugriffe auf Programm-/Daten- und IO-Bereiche muss mit Hilfe eines externen Logikbausteins durchgeführt werden; ein programmierbarer GAL-Baustein (gate-array-logic) kann zu diesem Zweck eingesetzt werden. Grundlage der Entwicklung stellt das vorliegende Evaluierungsboard [Tex99a] des DSPs dar, das mit zwei SRAMs mit jeweils 64 kWorten Daten- und Programmspeicher sowie einem GAL vom Typ $16V8$ bestückt ist. Ab der zweiten Platinenversion wird ein 256 kWorte-Speicherbaustein verwendet [Sam00], der nur die Hälfte des Platzes der beiden 64 kWorte großen Speicherbausteine einnimmt und zusätzlich noch den 32 kWorte großen globalen Speicherbereich verwalten kann. Als Dekodierlogik wird weiterhin ein $16V8$ GAL-Baustein [Lat01] verwendet.

5.4 Aktoren

5.4.1 Motoren

Bewegungen im Roboterfußball erfolgen meist nicht geradlinig und bei konstanter Geschwindigkeit, sondern sind durch häufiges Abbremsen, Beschleunigen und Ändern der Richtung geprägt; daher müssen die Motoren des Roboters genügend leistungsstark sein, um den Roboter möglichst schnell beschleunigen und abbremsen zu können. Die Anbringung bzw. Integration eines Radencoder-Sensors für die Detektion der Motor/Radbewegungen muss zusätzlich möglich sein. Da die Auswahl der Motoren - bis auf den Bereich der Sensorik - nicht Teil dieser Arbeit ist, wird an dieser Stelle nur das Ergebnis der Recherche vorgestellt: verwendet wird (wie bei den meisten MiroSot-Teams auch) ein Gleichstrom-Kleinstmotor des Herstellers *Dr. Fritz Faulhaber GmbH & Co.KG* vom Typ 2224 mit integriertem $IE2-16$ Radencoder-Sensor und einer Nennspannung von 6 V bei einer Leistung von ca. 5 Watt [Dr.00a]. Die Versorgungsspannung des Motors kann im vorliegenden Entwurf bis zu 9 Volt (je nach Energiezustand des Akkus) betragen, was aber laut Hersteller zulässig ist und auf die Lebensdauer des Motors kei-

ne signifikante Auswirkung hat. Durch diese Maßnahme wird die Leistung des Antriebs erheblich gesteigert.

5.4.2 Motortreiber

Um eine Steuerung der Antriebsmotoren zu ermöglichen, muss eine Leistungsregelung der Motoren realisiert werden; um dies zu erreichen existieren zwei unterschiedliche Verfahren, die dem Modellbau entstammen und als *mechanische* und *elektronische Fahrregler* bekannt sind. Mechanische Fahrregler enthalten einen niederohmigen Spannungsteiler und unterteilen die Versorgungsspannung vor dem Motor. Je höher der Anteil des Motors an der Versorgungsspannung ist, desto mehr Leistung liefert der Motor. Der Nachteil des Verfahrens liegt im Verlust der restlichen Versorgungsspannung, die im Fahrregler in Wärme umgesetzt wird.
Elektronische Fahrregler pulsen die Versorgungsspannung mit einer festen Frequenz. Dem Motor wird in schnellem Takt kurzzeitig die volle Versorgungsspannung zur Verfügung gestellt. Durch die Trägheit des Motors wird die Wirkung der Spannungspulse über die Zeit gemittelt. Die Veränderung des Verhältnisses zwischen den beiden Zuständen Spannung und keine Spannung wirkt für den Motor wie eine Gleichspannungsversorgung zwischen 0 Volt und der maximalen Betriebsspannung. Die elektronische Regelung benutzt als Eingangssignal das in Abschnitt 2.5.1 vorgestellte PWM-Signal und schaltet mit Hilfe dieses Signals die Versorgungsspannung der Motoren. Das Verfahren ist - bis auf einen Spannungsabfall von ca. $0,7$ Volt an den Schalttransistoren der Regelung - verlustfrei und daher gut für die Regelung von mobilen Robotern geeignet. Eine solche Regelung kann in den meisten mit Gleichstrommotoren betriebenen, mobilen Robotern beobachtet werden [JF96]$_{S.159ff}$.
Die Elektronik des Motortreibers wird auf einer kleinen, separaten Platine in unmittelbarer Nähe der Motoren aufgebaut; dafür gibt es mehrere Gründe. Erstens sollen unnötig lange Kabelwege vermieden werden; da die Akkus ebenfalls in der Nähe der Motoren angeordnet sind, ist es sinnvoll, die Leistungselektronik auf kürzestem Weg zwischen den Akkus und Motoren anzuordnen. Zweitens entstehen durch die PWM-Pulsung der Motoren innerhalb der Motorspulen Störimpulse aufgrund des schnellen Auf- und Abbaus der Spannung. Diese Impulse sind entgegengesetzt der PWM-Spannung polarisiert und können mit Hilfe von Dioden größtenteils eliminiert werden [ST 00a]. Auch um diese Störungen möglichst weit von der restlichen Elektronik und von empfindlichen Sensoren fern zu halten, wurde die beschriebene Anordnung der Platinen gewählt.

5.5 Sensoren

5.5.1 Radencoder-Sensor

Die Radencoder-Impulsgeber IE2-16 [Dr.00b] sind fest in die Motoren integriert, damit vor jeglichen mechanischen Schäden geschützt und liefern 16 Impulse pro Motorumdrehung, was bei einer QEP-Dekodierung 64-Schritten entspricht[6]. Bei einer externen Getriebeuntersetzung von 1:8 ergeben sich $4 * 16 * 8 = 512$ Impulse pro Radumrehung.
Das Ausgangssignal eines Sensors kann direkt an einen DSP-internen QEP-Dekoder geschaltet werden; die DSP-Hardware führt automatisch einen 16-bit Zähler mit, so dass zu jeder Zeit die aktuelle Radposition gelesen und zur Steuerung und Regelung verwendet werden kann. Für den zweiten Sensor ist ein externer Dekodierbaustein erforderlich, da der DSP zwar über einen zweiten 16-bit Zähler verfügt, die erforderliche Dekodierlogik allerdings nur einmal vorhanden ist. Ein zweiter existierender Zähler besitzt Eingänge, die allerdings die Zählrichtung und die Zählimpulse und nicht die beiden Rechtecksignale des Encoder-Sensors benötigen. Der QEP-Dekoder *LS7084* leistet exakt die geforderte Umwandlung [LSI99].

5.5.2 Beschleunigungs-Sensor

Mit Hilfe eines Beschleunigungs-Sensors kann ein inertiales Navigationssystem für den Roboter realisiert werden, das - falls es präzise genug arbeitet - die Positionsbestimmung des Roboters unterstützen kann. Zusätzlich könnte die Detektion von Erschütterungen und somit von Kollisionsereignissen mit Ball, Robotern und der Spielfeldbegrenzung möglich sein; auch eine Detektierung von Schlupf dürfte erreichbar sein und eine Möglichkeit zur Bestimmung der Untergrundbeschaffenheit bieten, was eine einfache Traktionskontrolle ermöglichen würde.
Beschleunigungs-Sensoren existieren in diversen Varianten. Für die vorliegende Entwicklung eignen sich aufgrund der eingeschränkten Größe des Roboters nur kleine in Chipform verfügbare Beschleunigungs-Sensoren. Analog Devices bietet mit dem ADXL202 [Ana99] einen in dieser Hinsicht gut geeigneten Sensor an, der unter anderem auch in inertialen Navigationssystemen von Raketen eingesetzt wird, um Latenzzeiten der GPS-Positionsbestimmung zu überbrücken.
Der Sensor verfügt über zwei horizontal um 90° Grad gedrehte Messachsen, die Beschleunigungen im zweidimensionalen Raum messen und sich daher gut für den Roboterfußball eignen. Messbar sind Beschleunigungen bis zu $2g$ ($1g = 9,81 m/s^2$ Erdbeschleunigung), die als analoges Ausgangssignal oder

[6] Es werden die steigenden und fallenden Flanken der beiden Rechtecksignale gezählt.

digital als PWM-Signal ausgegeben werden. Der digitale Ausgang erfordert auf Mikroprozessorseite eine adäquate Dekodierung inklusive Zeitmessung für das Verhältnis der PWM-Werte. Dies kann vom DSP aufgrund seiner Verwendung für die Encoder-Sensoren und PWM-Ausgabe nicht mehr geleistet werden; daher werden zwei Analogeingänge des DSPs zur Messung der Beschleunigungen verwendet.
Die Sensordaten unterliegen einer gewissen Drift, die unter anderem abhängig von der Temperatur ist. Um Drift zu kompensieren, wird der in Abschnitt 5.5.6 beschriebene Temperatursensor verwendet. Der Sensor wurde ab der zweiten Platinenversion in den Roboter integriert, um seine Eignung für die genannten Aufgaben zu untersuchen.

5.5.3 Kreisel-Sensor

Die Suche nach einem angemessen kleinen und verfügbaren Kreisel (Gyroskop), welcher sich zur Unterstützung des inertialen Navigationssystems (siehe Abschnitt 10.3) geeignet hätte, war leider erfolglos. Mechanische Gyroskope enthalten meist eine größere bewegte Masse; selbst mechanische Miniaturversionen - die im Modellbau häufig zur Stabilisierung von Hubschraubermodellen verwendet werden - sind zu groß für einen Roboter der MiroSot-Klasse. Integrierte Gyroskopschaltkreise, die ihre Daten aufgrund von Mikromechaniken oder piezoelektronischen Komponenten liefern, waren zur Entwicklungszeit nicht oder nur im Prototypstadium verfügbar. Der beschriebene Beschleunigungs-Sensor liefert durch die senkrecht zur Fahrtrichtung angeordnete zweite Messachse Informationen über eine Drehung des Roboters, deren Verwendbarkeit und Genauigkeit in einem ersten Schritt evaluiert werden soll. Aus diesen Gründen wurde von der Integration richtungsgebender Sensoren vorerst abgesehen. Falls sich der Beschleunigungs-Sensor in dieser Hinsicht als nicht ausreichend herausstellt, kann das System durch die Integration eines Gyroskops oder eines Kompass-Sensors ergänzt werden.

5.5.4 Kompass-Sensor

Die Integration eines Kompass-Senors stellt eine äußerst interessante Möglichkeit dar, die Richtungsbestimmung und damit die Odometrie des Roboters zu verbessern. Ein Kompass besitzt im Gegensatz zu einem Gyroskop keine Drift und erfordert daher keine zwischenzeitliche Neukalibrierung des Systems. Ein Roboter mit Kompass-Sensor wäre in der Lage seine Ausrichtung autonom, ohne globale Datenquellen, zu bestimmen. Ein geeigneter Sensor *KMZ52* des Herstellers *Philips* [Phi00] ermöglicht - in Verbindung mit einem Mikroprozessor - eine Ausrichtungsbestimmung mit einer Genauigkeit von unter einem Grad. Es handelt sich dabei um einen analogen Sensor,

dessen Integration aufgrund seiner Störanfälligkeit als aufwendig angesehen werden muss. Störungen können sowohl von der Digital- oder Leistungselektronik des Roboters sowie von den schnell wechselnden magnetischen Feldern der Motoren ausgehen. Die Ausgangssignale des Sensors müssen zusätzlich aufbereitet werden, damit diese sich für die Verarbeitung durch einen AD-Wandler[7] eignen; die internen Reset- und Korrekturspulen des Sensors müssen zudem adäquat angesteuert werden.
Die Entwicklung und Einbindung dieses Sensors ist aus zeitlichen Gründen nicht im Rahmen dieser Arbeit möglich und kann in späteren Arbeiten erfolgen, falls die Ausrichtungsbestimmung des Roboters als verbesserungswürdig betrachtet wird.

5.5.5 Akkuspannungs-Sensor

Der Akkuspannungs-Sensor wird durch einen einfachen Spannungsteiler, bestehend aus zwei Widerständen, realisiert. Er wird mit einem AD-Wandlerkanal verbunden und ermöglicht die einfache Messung der Akkuspannung und damit die Bestimmung des Energiezustands des Akkus.

5.5.6 Temperatur-Sensor

Der Temperatur-Sensor wird aufgrund der Temperaturdrift des Beschleunigungs-Sensors integriert und besteht aus einem temperaturabhängigen Widerstand, der in einen Spannungsteiler integriert ist. Die resultierende Spannung wird - wie beim Akkuspannungs-Sensor - mit einem AD-Wandlerkanal verbunden; es wird ein Miniatursensor des Typs *KT210* des Herstellers Infineon [Inf00] verwendet.

5.5.7 DIP-Schalter

Die Zuweisung der Roboter-ID, die zur Identifikation des Roboters dient, wird durch 3 DIP-Schalter (Schalter Nr. 2-4) ermöglicht: es können auf diese Weise 8 verschiedene IDs zugewiesen werden. Der erste DIP-Schalter dient der Umschaltung der DSP-Speichermodi zwischen nicht-flüchtigem Flash-Speicher für den Betrieb und flüchtigem RAM-Speicher für die Entwicklungsumgebung des Roboters.

5.5.8 Infrarot-Sensoren

Auf eine Integration von Infrarot-Sensoren wurde zugunsten der Verwendung von Kameras verzichtet, da diese eine wesentlich bessere Erkennung

[7] Ein Analog-Digital-Wandler (AD-Wandler) wandelt die Spannung eines analogen Eingangssignals in einen entsprechenden digitalen Zahlenwert um.

und Klassifizierung von Objekten ermöglichen. Auch besitzen Kamera-Sensoren eine größere Sensorreichweite, die bei Infratrot-Sensoren auf einige Millimeter beschränkt ist.

5.5.9 Kamera-Sensoren

Die Integration von Kamera-Sensoren und die damit verbundene Möglichkeit einer lokalen Bildverarbeitung kann den Roboter zu einem vollautonomen System erweitern. Eine derartige Sensorik ist mit einer komplexen Entwicklung verbunden, deren Durchführung nicht Fokus dieser Arbeit ist. Da einer lokalen bildverarbeitenden Sensorik - nach Auffassung des Autors - eine hohes Potential beizumessen ist, wird in Kapitel 13 in Ergänzung zur Aufgabenstellung die Entwicklung eines miniaturisierten bildverarbeitenden Systems konzeptionell erarbeitet und prototypisch durchgeführt. In Vorbereitung der im Verlaufe dieser Arbeit durchgeführten Konzeption werden im Folgenden Anforderungen an mögliche Kamera-Sensoren diskutiert:
Miniaturkameras - die sich für die Integration in einen MiroSot-Fußballroboter eignen - setzen Eigenschaften voraus, welche die Auswahl an geeigneten Sensoren stark einschränken:

- Die Kamera benötigt einen digitalen Ausgang, damit ein Mikroprozessor direkt auf Bilddaten des Sensors zugreifen kann. Es eignen sich Kamera-ICs, die üblicherweise für Webcams eingesetzt werden; in diesem Fall ist ebenfalls eine ausschließlich digitale Übertragung des Bildes gefordert.

- Die Kamera darf eine gewisse Größe nicht überschreiten, damit eine Integrierung in das Gehäuse des Roboters möglich ist.

- Die Versorgungsspannung der Kamera darf nicht größer sein als die Betriebsspannung der Roboterelektronik (5 Volt). Es eignen sich die seit einiger Zeit verfügbaren Kameras in CMOS-Technologie.

Der gewählte Sensor VV6301 [ST 00b] wird von *STMicroelectronics* hergestellt und im Normalfall für Webcams eingesetzt. Er ist preislich äußerst günstig (ca. 30 DM) und verfügt über eine Auflösung von $164 * 124$ Bildpunkten in Farbe, was für die Detektion und die Klassifizierung von farbigen Objekten - wie zum Beispiel dem orangenen Spielball - ausreichen sollte. Der Sensor besitzt einen Datenbus, der das aufgenommene Bild byteweise zur Verfügung stellt. Für einen Roboter werden zwei solcher Kamera-Sensoren vorgesehen, die in beiden Fahrtrichtungen in das Robotergehäuse integriert werden und eine Sicht nach vorne und hinten ermöglichen.
Das Einlesen der Bilder und deren Verarbeitung ist aufgrund der harten Zeitanforderungen bezüglich der Datenbussignale der Kamera nicht mehr

durch den DSP des Hauptsystems lösbar, zumal der Prozessor in der Ausführung seiner Aufgaben nicht von zusätzlichen, optionalen Sensoren eingeschränkt werden sollte. Ein separates DSP-System wird daher zur Bildverarbeitung entworfen. Es beinhaltet aus Konsistenzgründen und wegen der Verfügbarkeit der Entwicklungswerkzeuge einen TMS320F206 [Tex96b] von Texas Instruments, der aus der gleichen Produktfamilie wie der im Hauptsystem verwendete TMS320F240 stammt; er ist aber ohne Motioncontroling- und Analogschaltkreise wesentlich kleiner und erst diese Platzersparnis machte eine Integration in das Robotergehäuse möglich. Er wird mit dem gleichen Hauptspeicher ausgestattet wie DSP des Hauptsystems, so dass sich beide Systeme von der Programmierung und von den Speichereigenschaften identisch verhalten. Der Bildverarbeitungs-DSP ist in der Lage, mehrere Kamerabilder im Speicher abzulegen und dort zu bearbeiten. Er wird mit Hilfe einer seriellen Hochgeschwindigkeits-Schnittstelle mit dem DSP der Hauptplatine verbunden, die zur Inter-Prozessor-Kommunikation vorgesehen ist.

5.5.10 Zeilenkamera

Eine Alternative zur Integration von Farbkameras besteht in der Verwendung einfacher Zeilenkameras, die für den Khepera-Roboter verwendet werden (siehe Abbildung 3.5). Allerdings erfordert der Einsatz dieser Sensoren ebenfalls einen eigenen - wenn auch möglicherweise kleineren - Prozessor, um die Sensorzeile adäquat anzusteuern. Die von Texas Instruments hergestellten Sensoren werden allerdings nicht mehr produziert und sind daher nicht für Neuentwicklungen zu empfehlen. Die Zeilenkamera der Khepera-Roboter verfügt über 64 Grauwert-Bildpunkte.

5.6 Kommunikation

Das Funksystem ist sicherlich der meist diskutierte Teil der Roboterhardware. Diverse Probleme - bedingt durch Funkstörungen oder eine unzureichende Beachtung der besonderen Erfordernisse der BiM-Transceiver sowie durch leicht fehlerhafte Implementierungen des Funkprotokolls - führten sowohl in PG340[8] als auch in PG362[9] zu größeren Schwierigkeiten mit den Funksystemen. Ziel der Neuentwicklung ist daher nicht nur eine Erweiterung und Verbesserung der Kommunikationsmöglichkeiten sondern auch eine vereinfachte Problemerkennung und -beseitigung. In Kapitel 12 wird eine genauere Diskussion eines geeigneten Protokolls sowie eine Implementierung

[8] Der PG340 standen nur die RobotCavalry-Roboter mit fest implementiertem Funkprotokoll - ohne Fehlerprüfung - zur Verfügung.

[9] Die PG362 implementierte bereits ein eigenes Funkprotokoll, allerdings ohne detaillierte Beachtung der Vorgaben seitens des Herstellers.

durchgeführt, die insbesondere die Vorgaben des Herstellers Radiometrix einhält und somit Übertragungsfehler reduziert.

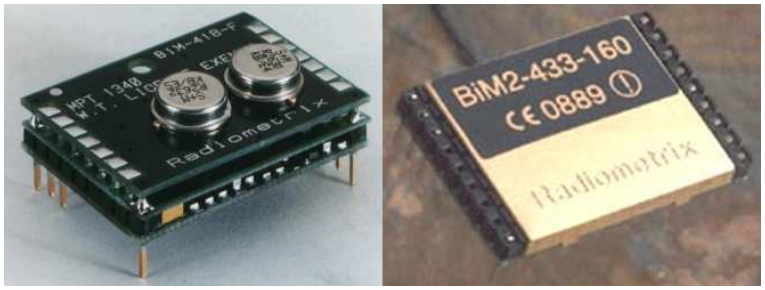


Abbildung 5.1: BiM / BiM2 - Transceivermodule (Quelle: Radiometrix)

Die Radiometrix-BiM-Transceiver (siehe Abbildung 5.1), die auch für die Kommunikationslösung des Robot-Cavalry-Systems verwendet werden, sind mit ihrem geringen Platzbedarf und geringem Stromverbrauch zum Zeitpunkt dieser Arbeit relativ einmalig. Diese Feststellung ist an der Tatsache ersichtlich, dass beinahe alle (Fußball)-Roboter mit MiroSot-ähnlichen Abmessungen (u.a. auch die Khepera-Roboter [K-T95]) die Radiometrix-Module verwenden.
Die Module liegen neuerdings in der Version BiM2 vor und verfügen neben einer verbesserten Abschirmung gegen Störsignale über eine erhöhte Datentransferrate von maximal 64kBit/s. Alternative Technologien, wie beispielsweise DECT oder Bluetooth, die weitere Vorteile - wie z.B. größere Bandbreite oder einen integrierten *Baseband-Controler* - einbringen könnten, sind zum momentanen Zeitpunkt entweder zu groß, zu teuer oder nicht als fertiges Produkt verfügbar.
Vor allem zusätzliche Bandbreite wäre eine wünschenswerte Verbesserung, um z.B. Kamerabilder und Sensordaten zu Entwicklungszwecken zum Hostrechner zu übertragen. Eine durch die Zusammenarbeit mit Texas Instruments initiierte Idee, einen TI-Transceiver Chip für das 833MHz ISM-Band zu benutzen, könnte alle gewünschten Anforderungen erfüllen, ist aber nicht als komplettes Funkmodul verfügbar. Eine entsprechende Entwicklung hätte den zeitlichen Rahmen dieser Arbeit überstiegen und wird im Anschluss vom Autor durchgeführt.
Es lassen sich durch den Einsatz der BiM-Transceiver eine Reihe von Verbesserungen gegenüber dem Cavalry-Roboter erzielen:

Sende-/Empfangsumschaltung: Eine automatische Umschaltung des Betriebsmodus auf Roboter- und Hostseite bietet sich an, um die bisherige statische Festlegung des Hosts als Sender und der Roboter als Empfänger aufzuheben.

Datenrate: Eine Ausnutzung der vollen Datenrate der BiM-Transceiver vervielfacht die zur Verfügung stehende Bandbreite. Die Transceiver sind für 40kBit/s bzw. 64kBit/s (BiM Version 2) ausgelegt.

Fehlerrate: Durch Beachtung bisher nicht implementierter Vorgaben und Einschränkungen aus [Rad01b] kann die Fehlerrate erheblich gesenkt werden.

Die angegebenen Möglichkeiten sind ein wesentliches Verbesserungspotential bezüglich der Kommunikationsfähigkeiten sowie der Datenrate. Die Ausschöpfung dieses Potentials wird im Kapitel 12 detailliert beschrieben.
Das Modul wird mit Hilfe der seriellen Schnittstelle des DSPs und durch zwei digitale Ein/Ausgänge angesteuert.

5.7 Energieversorgung

Die derart entworfene Elektronik der Hauptplatine des Roboters benötigt eine Betriebsspannung von 5 V und verbraucht im Betrieb ca. $0,2$ Ampere. Die vorgesehene Integration der Kamera-Sensoren benötigt weitere $0,2$ Ampere. Dieser Stromverbrauch ist aber im Vergleich zum Stromverbrauch der Antriebsmotoren während der Beschleunigung des Roboters marginal, kurzzeitig kann die Stromstärke bis zu einem Ampere pro Motor betragen. Die Spannung zur Ansteuerung der Motoren darf maximal 9 Volt betragen.

5.7.1 Akkus

Die Akkus müssen den Energiebedarf des Roboters für eine Spielzeit von mindestens fünf Minuten decken und einen Spitzenstrom von bis zu 2 Ampere für die Beschleunigung des Roboters leisten können, ohne einen Spannungsabfall zu verursachen, der einen Reset der Elektronik zur Folge hätte. Die Größenvorgaben des Roboters schränken die mögliche Akkuauswahl stark ein und lassen die Wahl zwischen *Micro-* oder *Mignonzellen*, die jeweils eine Spannung von $1,2$ Volt aufweisen: sieben dieser Akkus können maximal zu einem Akkupack verschweißt werden, so dass sich eine Gesamtspannung von ca. $8,4$ Volt ergibt. Die Mignonzellen verfügen über eine Kapazität von bis zu 1500 mAh, Microzellen besitzen üblicherweise 600 mAh und sind den Mignonzellen bezogen auf ihre Hochstromfähigkeit - der Fähigkeit kurzzeitig große Ströme abzugeben - unterlegen. Da ein höheres Gewicht die physikalischen Eigenschaften des Roboters positiv beeinflusst, bietet sich der Einsatz von Mignonzellen an. Ein Betrieb der Roboterelektronik ist mit dieser Lösung bis zu $3,5$ Stunden möglich, ein Fahrbetrieb bis zu ca. 1 Stunde bei üblichem Spielbetrieb; dabei wird von einem durchschnittlichen Verbrauch von einem Ampere ausgegangen.

5.7.2 Versorgung der Elektronik

Die Versorgung der Elektronik wird durch einen üblichen Festspannungsregler ermöglicht, der die Akkuspannung auf die für die Elektronik notwendigen 5 Volt regelt und glättet. Es wird eine sogenannte Low-Drop-Variante eingesetzt, die eine minimale Akkuspannung von 6 Volt benötigt, um eine geregelte 5 Volt Spannungsversorgung zu gewährleisten.

5.7.3 Versorgung der Motoren

Die Spannungsversorgung der Motoren wird im Gegensatz zur Elektronik direkt vom Akkupack übernommen. Eine Regelung und damit Verringerung der Spannung würde einen Leistungsverlust bedeuten und ist daher nicht sinnvoll.
Der in Abschnitt 5.4.2 besprochene Motortreiber trennt notwendigerweise die Versorgungsspannung von der Steuerspannung der Elektronik, die 5 Volt nicht überschreiten darf.

5.8 Ergebnis

Die Diskussion möglicher Bauteile und ihrer Integration bewirkte die konkrete Auswahl einzelner Teile, deren Funktionsfähigkeit im Rahmen der gestellten Anforderungen in den folgenden Kapiteln untersucht werden muss. Die konkrete Bauteilauswahl ermöglicht den Aufbau und Test erster Protoyp-Platinen und die Weiterentwicklung zu einem funktionsfähigen Fußballroboter.

Kapitel 6

Aufbau und prototypische Implementierung

In diesem Kapitel wird der detaillierte Aufbau des Robotersystems beschrieben. Die Verschaltung der Baugruppen wird erläutert, auf die verwendeten Bauteile und ihre Bezeichnung im Schaltplan, der als Anhang beiliegt, wird verwiesen.
Notwendige Vorbereitungen für die prototypische Implementierung werden durchgeführt und die Ansteuerung der Hardware beschrieben. Die Quelltexte der Implementierung liegen auf CD-Rom bei; es wird im Verlauf dieses Kapitels auf die jeweiligen Quelldateien verwiesen.

6.1 Überblick

Die Elektronik des Roboters besteht aus zwei Baugruppen: zum einen enthält die Leistungselektronik einige Bauteile zur Spannungsversorgung und zur Motoransteuerung. Sie wird auf einer separaten Platine aufgebaut und in das Gehäuse des Roboters - oberhalb der Motoren - integriert (Abbildung 6.1). Zum anderen besteht das DSP-System aus einer quadratischen Platine, die den DSP und den externen Hauptspeicher, das Funkmodul sowie den größten Teil der Sensorik enthält (Abbildung 6.2). Sie wird zu oberst auf das Robotergehäuse montiert. Ein Flachbandkabel verbindet das DSP-System mit der Leistungselektronik.
Die Energieversorgung des Roboters - das Akkupack - wird zwischen die Platine des DSP-Systems und das Gehäuse des Roboters geschoben und mit Hilfe eines zweipoligen Steckers mit der Leistungselektronik verbunden. Abbildung 6.1 zeigt den strukturellen Aufbau des Roboters.

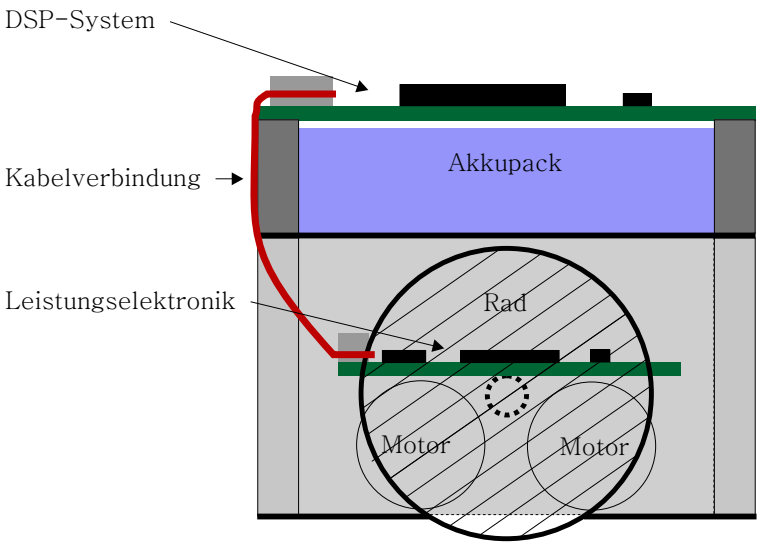


Abbildung 6.1: Struktur des Roboters (seitlich betrachtet)

6.2 DSP

Der DSP (U1[1]) wird seitlich auf der DSP-Systemplatine angeordnet und von dort aus mit dem externen Hauptspeicher, dem Funkmodul, der Sensorik und der Leistungselektronik-Platine verbunden. Für den Betrieb des DSP sind kleinere aktive und passive Bauteile notwendig, die in unmittelbarer Nähe angeordnet werden: ein Spannungswächter-IC (U7) vom Typ TL7757 schaltet den DSP in den Reset-Status, solange (bzw. sobald) die Versorgungs-Spannung einen kritischen Wert unterschreitet und sorgt damit beispielsweise für einen stabilen Zustand nach dem Einschalten des Systems. Die Kondensatoren C12 bis C21 stabilisieren die Versorgungsspannung an den Stromversorgungs-Pins des DSP.
Der Prozessor wird mit Hilfe eines 10 MHz-Quarzes getaktet, DSP-intern wird der Takt auf 20 MHz erhöht. Der Quarz (Q1) wird mit den XTAL1- und XTAL2-Pins des DSP verbunden.
Der Betriebsmodus des DSP kann mit Hilfe des DIP-Schalters Nr. 1 zwischen dem Microcomputer-Modus (Schalterstellung: aus) und dem Microcontroler-Modus (Schalterstellung: an) umgeschaltet werden. Im Microcomputer-Modus wird der DSP-interne 16 kWorte große Flash-Speicher anstelle der ersten 16 kWorte des externen Programmspeichers verwendet, so dass ein im Flash-Speicher abgelegtes Programm beim Einschalten des Roboters ausgeführt wird. Der Microcontroler-Modus wird im allgemeinen während der Ent-

[1]Hinter den jeweiligen Bauteilen angegebene Bezeichnungen beziehen sich auf die in Anhang B beigefügten Schaltpläne.

wicklung verwendet: das Programm kann mit Hilfe des J-TAG-Interfaces in den RAM-Speicher des Roboters geschrieben und anschließend ausgeführt werden. Durch dieses Vorgehen wird ein häufiges Beschreiben des Flash-Speichers vermieden.
Der DSP verfügt über ein J-TAG-Interface, das mit Hilfe einer Stiftleiste (DEBUG) mit dem Entwicklungssystem verbunden werden kann.
Die Initialisierung des DSP wird durch den Assembler-Code der Quelldatei *boot.asm* durchgeführt.

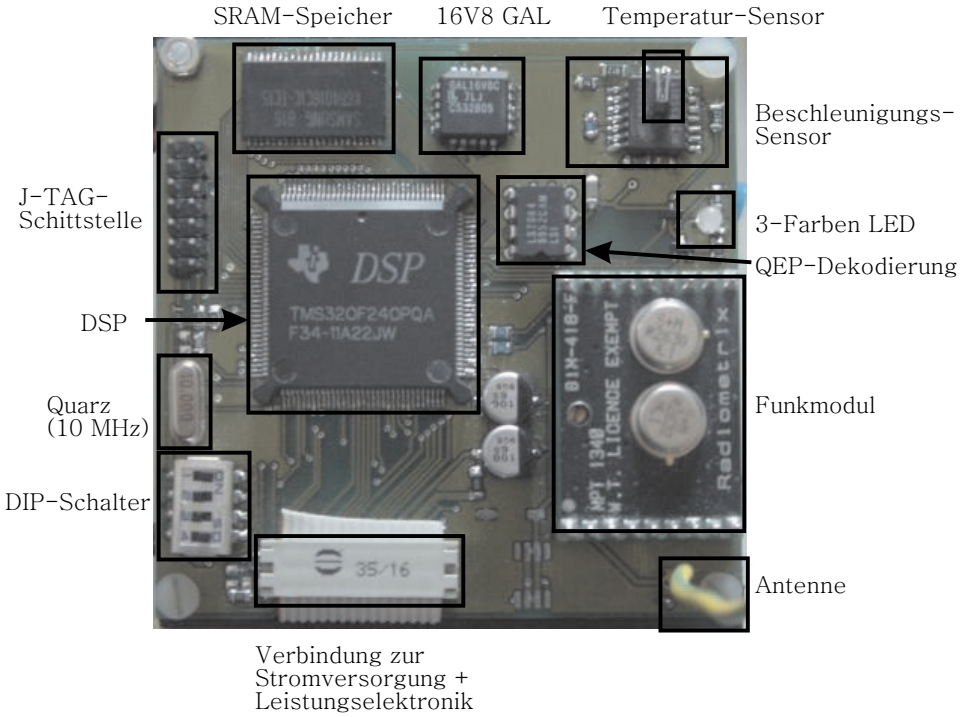


Abbildung 6.2: Bauteilübersicht der DSP-Platine

6.3 Externer Hauptspeicher

Der externe RAM-Speicher besteht aus einem 256 kWorte großen Speicher-IC (U2) und einem 16V8-GAL-IC für die Zugriffsdekodierung (U3). Daten- und Adressbus des DSP (A0-A15 und D0-D15) werden direkt mit dem Speicher-IC verbunden. D16 und D17 der Speichers werden mit den DS- (data-select) und BR- (bus-request) Pins des DSP verbunden: DS unterscheidet zwischen Zugriffen auf Daten- und Programmspeicher und BR zeigt einen Zugriff auf den 32 kWorte großen globalen Speicherbereich an. Der Hauptspeicher wird durch diese Maßnahme geviertelt: je 64 kWorte stehen für Programm- und Datenspeicher zur Verfügung, 64 kWorte sind für den

globalen Speicherbereich verfügbar (von dem nur 32 kWorte genutzt werden können) und 64 kWorte verbleiben generell ungenutzt. Mit Hilfe dieser Schaltung wird dem DSP der maximale extern adressierbare Speicherbereich zur Verfügung gestellt.
Der GAL-IC wird mit der Speicherzugriffslogik (DS, PS, IS, R/W, STRB, WE und BR)[2] verbunden. Der GAL-IC dekodiert die Speicherzugriffe des DSP und gibt nur im Fall eines externen Speicherzugriffs die Signale der Zugriffslogik an den Speicher-IC weiter. Die zur Programmierung der Logik des GAL-ICs benötigten Gleichungen können dem Handbuch des Evaluation-Boards [Tex99a]$_{S.95ff}$ entnommen werden, da das dort eingesetzte GAL eine identische Speicherdekodierung vornimmt.

6.4 Funkmodul

Das DSP-System verwendet weiterhin den BiM-Transceiver [Rad01a] der Firma Radiometrix. Die serielle Schnittstelle des DSP (SCITXD/SCIRXD) wird mit der seriellen Schnittstelle des BiM-Transceivers verbunden. Die Umschaltung des Transceivers zwischen Sende- und Empfangsmodus wird mit Hilfe eines DSP-digital-Ausgangs (IOPC0) und den Invertern U9 und U12 ermöglicht, deren Ausgänge mit den RX-Select- und TX-Select-Eingängen des Transceivers verbunden sind.
Der Transceiver liefert ein *carrier-detect*-Signal, das - so der Transceiver im Empfangsmodus betrieben wird - andere sendende Stationen detektiert. Der carrier-detect-Ausgang des Transceivers wird mit einem digital-Eingang (XINT3/IO) des DSP verbunden.
Die prototypische Implementierung der seriellen Schnittstelle des DSP, ihre Initialisierung und die notwendigen Funktionen zum Datentransfer finden sich in der Datei *rfHalDsp.c*.

6.5 Sensoren

Die Sensorik des Roboters wird - bis auf die Radencoder-Sensoren - auf der DSP-System-Platine angeordnet. In den folgenden Abschnitten wird die Anbindung der einzelnen Sensoren beschrieben und - falls notwendig - ihre Verwendung mit Hilfe von Messungen vorbereitet.

6.5.1 Radencoder-Sensor

Die Encoder-Sensoren IE2-16 [Dr.00b] sind in die Motoren integriert und werden mit Hilfe der Motoranschlusskabel mit der Leistungselektronik-Pla-

[2] Die Bedeutungen der einzelnen Pins sind im CPU-Manual [Tex96a]$_{S.6ff}$ erläutert.

tine verbunden. Das Signal jedes Sensors bestehen aus zwei versetzten Rechtecksignalen (siehe Abbildung 2.6), die durch eine Dekodierlogik in Eingangssignale für einen digitalen Zähler gewandelt werden müssen. Der DSP verfügt über eine solche Dekodierlogik, mit deren Hilfe der DSP-interne Zähler Nr.3[3] verwendet werden kann. Die Sensor-Signale des linken Rades werden - über das Flachbandkabel zwischen der Leistungselektronik-Platine und der DSP-System-Platine - mit den Quadratur-Dekodierungseingängen (QEP1 und QEP2) des DSP verbunden.
Die Signale des zweiten Sensors werden ebenfalls über das Flachbandkabel auf die DSP-System-Platine geführt; dort werden sie allerdings von dem externen Dekodierlogik-IC LS7084 [LSI99] in Zählsignale für den Zähler Nr.2 des DSP aufbereitet und mit den Zählereingängen TMRDIR und TMRCLK verbunden.
Durch diese Schaltung stehen die Radpositionen beider Räder jederzeit in den genannten Zählern des DSP zur Verfügung.
Die Initialisierung der Eingänge und der Zähler sowie die Funktionen zum Lesen der Radpositionen wurden in der Datei *quadinp.c* implementiert.

6.5.2 Beschleunigungs-Sensor

Eingesetzt wird ein Sensor des Typs ADXL202 von Analog Devices [Ana99], der in der Lage ist, Beschleunigungen bis zu $2g$ ($1g = 9,81 \frac{m}{s^2}$ Erdbeschleunigung) in x und y Richtung zu messen und der diese Daten durch ein digitales und ein analoges Ausgangssignal zur Verfügung stellt. Da sich aufgrund der Verwendung der Radencoder-Sensoren und der PWM-Motorsteuerung kein weiterer digitaler Sensor-Eingang realisieren lässt[4]. Aus diesem Grund werden die analogen Ausgänge des Sensors verwendet und mit den AD-Wandlereingängen ADCIN2 und ADCIN3 verbunden.

Kalibrierung

Der Sensor liefert an seinem analogen Ausgang eine Spannung zwischen 0 und 5 V, diese Spannung beträgt in Ruhelage des Sensors (Nullwert) ca. die Hälfte der Versorgungsspannung. Dieser Wert ist abhängig von der Umgebungstemperatur und kann in geringem Maße driften[Ana99]. Eine Ermittlung dieses Wertes ist daher regelmäßig notwendig, um Abweichungen vom gemessenen Nullwert auszugleichen. Die Auswertung der Sensordaten kann

[3] Für eine detaillierte Beschreibung des internen Aufbaus des DSP wird auf das Handbuch der CPU [Tex96a] sowie auf die Beschreibung der DSP-internen Peripherie [Tex99b] verwiesen.

[4] Der DSP verfügt über drei interne Zähler, einer wird für die Taktung des PWM-Ausgangs verwendet, die übrigen zwei werden für die Encoder-Sensoren benötigt. Die Realisierung eines digitalen PWM-Eingangs benötigt einen weiteren Zähler und kann daher in dieser Form nicht realisiert werden.

daher erst nach Abzug des - während der Kalibrierung ermittelten - Nullwertes geschehen.

Rauschverhalten

Das Grundrauschen des Sensorsignals bewirkt, dass langsam beschleunigte Bewegungen nicht messbar sind, da der Signalabstand des Rauschens zum eigentlichen Signal zu klein wird. Die Ermittlung dieser Grenze entscheidet beispielsweise, ab welcher Geschwindigkeit der Beschleunigungs-Sensor einen sinnvollen Beitrag zur Positionsberechnung des Roboters beitragen könnte. Das Rauschverhalten des Beschleunigungs-Sensors im entworfenen System wird im folgenden Abschnitt untersucht und eventuelle Verbesserungsmöglichkeiten evaluiert.

Messung des Rauschens

Für jede Achse werden bei ruhendem Roboter mit einer Frequenz von einem Kilohertz Messwerte über eine Zeitraum von 5 Sekunden gesammelt und anschließend statistisch ausgewertet (siehe Tabelle 6.1): *Max* gibt das Maximun der positiven gemessenen Beschleunigungswerte, *Min* das Minimum der negativen gemessenen Beschleunigungswerte an. *Peak-Peak* gibt das Rauschen von Spitze zu Spitze des Signals an, was der Differenz von *Max* und *Min* entspricht. Das *RMS*-Rauschen (root-mean-square) ist ein häufig gewähltes Kennzeichen für Signalrauschen und wird wie folgt definiert:

$$x_{RMS} = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1} x^2[i]} \tag{6.1}$$

$$\begin{aligned} x_{RMS} &= \text{RMS Rauschen} \\ N &= \text{Anzahl der Messwerte} \\ x_{[}i] &= \text{i-ter Messwert} \end{aligned}$$

Die Messwerte zeigen ein recht starkes RMS-Rauschen, dass um den Faktor $2-3$ höher liegt als der für den Sensor typische Wert von $8,7$ mg. Eine Untersuchung des Sensorausgangs mit Hilfe eines Oszilloskops ergab, dass bei gestopptem Prozessor das Rauschen wesentlich geringer wird als bei laufendem Prozessor; auch beeinflusst der Betrieb des Analog-Digital-Wandlers - der das analoge Sensorsignal umwandelt - das zu messende Signal. Als eine mögliche Ursache wurde die unpassende Impendanz von Sensor-Ausgang und AD-Wandler-Eingang identifiziert. Die Messung des analogen Signals erfordert eine gewisse Stromstärke, die vom AD-Wandler benötigt wird. Ist der Sensor nicht in der Lage diese Stromstärke am Ausgang zur Verfügung zu

Rauschverhalten ohne Impendanzwandler (in mg)					
Messung	*Achse*	*Max*	*Min*	*Peak-Peak*	*RMS*
1	x	124,099	-95,751	219,849	20,120
2	x	92,883	-79,856	172,739	19,766
3	x	108,998	-95,148	204,146	20,611
4	y	125,741	-94,108	219,849	23,473
5	y	109,318	-94,827	204,146	23,321
6	y	109,614	-110,236	219,849	23,752

Tabelle 6.1: Rauschverhalten ohne Impendanzwandler

Rauschverhalten mit Impendanzwandler (in mg)					
Messung	*Achse*	*Max*	*Min*	*Peak-Peak*	*RMS*
7	x	45,929	-48,292	94,221	16,265
8	x	46,837	-63,087	109,925	16,437
9	x	46,903	-47,318	94,221	16,255
10	y	59,771	-50,154	109,925	18,229
11	y	59,171	-50,754	109,925	18,994
12	y	59,714	-50,210	109,925	18,776

Tabelle 6.2: Rauschverhalten mit Impendanzwandler

stellen, verfälscht die Messung das zu messende Signal. Dies trifft im vorliegenden Platinenentwurf zu, da die Inkompatiblität von Sensor und Wandler in bisherigen Versionen nicht bemerkt wurde. Die Impendanz des Sensorausgangs beträgt $32k\Omega$, der AD-Wandler des DSP benötigt aber mindestens eine Impendanz von $9k\Omega$, um das Signal in der geforderten Zeit korrekt messen zu können.
Die Schaltung wurde testweise um einen Impendanzwandler pro Sensorachse erweitert und die Messung wiederholt (Tabelle 6.2).
Durch diese Maßnahme wurden die Signalspitzen halbiert und der Wert des RMS-Rauschens deutlich gesenkt, was als gute Verbesserung anzusehen ist. Der Wert des RMS-Rauschens ist allerdings immer noch um den Faktor 2 höher, als der im Datenblatt [Ana99] angegebene typische RMS-Rauschwert von $8,7$ mg bei einer Bandbreite von $200Hz$.
Eine weitere Ursache könnte in den naheliegenden Digitalschaltungen liegen und eine Entstörung der Stromversorgung des Bauteils durch entsprechende passive Entstörbauteile könnte eine weitere Verbesserung bewirken. Dies wird in der vorliegenden Arbeit nicht untersucht, da die Entstörung von analogen Schaltungsteilen nicht Kernpunkt dieser Arbeit ist und aus elektronischer Sicht nicht trivial ist. Eine weitere Entstörung der Bauteile kann Teil zukünftiger Optimierungen sein. Die Genauigkeit des Sensors reicht für die Evaluierung der gewünschten Fähigkeiten aus.

6.5.3 Akkuspannungs-Sensor

Der Akkuspannungs-Sensor wird mit Hilfe eines einfachen Spannungsteiler aufgebaut und besteht aus den beiden Widerständen R16 und R17. Der Spannungsteiler veringert die Akkuspannung um den Faktor zwei, so dass die resultierende Spannung - bei einer maximalen Akkuspannung von 10 Volt - zwischen 0 und 5 Volt liegt und sich als Eingangsspannung eines AD-Wandlers[5] eignet. Der Ausgang des Spannungsteilers wird mit dem Wandlereingang ADCIN5 verbunden.
Für die Bestimmung der Akkuspannung V_{supply} gilt:

$$V_{supply} = \frac{M_{supply} * 10V}{1024} \tag{6.2}$$

$$M_{supply} = \text{Messergebnis AD-Wandlung}$$

Die Implementierung dieses Verfahrens findet sich in der Quelldatei *adc.c*.

6.5.4 Temperatur-Sensor

Der Temperatur-Sensor wird ebenfalls mit Hilfe eines Spannungsteilers aufgebaut, der statt zweier Widerstände nur einen Widerstand und einen Temperatur-Sensor enthält (R7 und TR1). Der Temperatur-Sensor vom Typ KT 110 [Inf00] verändert seinen Widerstand mit steigender Temperatur. Der Spannungsteiler aus Widerstand und Sensor wird zwischen Masse und 5 Volt geschaltet, so dass eine veränderte Temperatur eine veränderte Spannung am Ausgang des Teiler verursacht. Der Ausgang des Spannungsteilers wird mit dem Wandlereingang ADCIN4 verbunden.
Messungen ergaben, dass der Sensor bei einer Temperatur von $25°C$ einen Widerstand von 1970Ω besitzt, der sich für $10°C > T > 50°C$ um ca. 14,6 Ohm pro Grad erhöht. Für R7 wurde ein Wert von $4,7k\Omega$ gewählt. Für die Bestimmung der Temperatur T in $°C$ gilt:

$$T = 25°C + \frac{R_{TR} - 1970\Omega}{14,6\frac{\Omega}{°C}} \tag{6.3}$$

$$R_{TR} = \frac{1024 - M_{TR}}{M_{TR}} 4,7k\Omega \tag{6.4}$$

$$R_{TR} = \text{Sensor-Widerstand}$$
$$M_{TR} = \text{Messergebnis AD-Wandlung}$$

Die Implementierung dieses Verfahrens findet sich in der Quelldatei *adc.c*.

[5] Die AD-Wandler des DSP wandeln eine Eingangsspannung zwischen 0 und 5 Volt in einen Digitalwert zwischen 0 und 1024.

6.5.5 DIP-Schalter

Die DIP-Schalter sind als Block von vier Schaltern auf der DSP-System-Platine angebracht (Abbildung 6.2). Der erste DIP-Schalter dient der Umschaltung des DSP-Betriebsmodus (siehe Abschnitt 6.2) und wird mit dem MP/MC-Pin des DSP verbunden.
Die Schalter 2-4 werden als dreistellige Binärzahl interpretiert, dabei gibt Schalter 4 das LSB (least significant bit) an. Die Binärzahl gibt die Roboter-ID zwischen 0 und 7 an.
Schalter 2 wird mit dem digitalen Eingang XINT2/IO verbunden, Schalter 3 mit IOPC3 und Schalter 4 mit IOPC2. Die Initialisierung der Eingänge und die Auswertung der Roboter-ID wurde in der Quelldatei *dip.c* realisiert.

6.6 Aktoren

Die Aktoren des Roboters sind die beiden 2224R-Faulhaber Motoren [Dr.00a], die durch einen Motortreiber vom Typ L298 [ST 00a] mit Energie versorgt werden. Die Motoren werden durch ein sechs-poliges Flachbandkabel mit der Leistungselektronik-Platine verbunden, auf der auch der Motortreiber (U2) angeordnet ist. Der Treiber benötigt für jeden Motor einen PWM-Wert - der durch die beiden PWM-Ausgänge PWM7 (linker Motor) und PWM6 (rechter Motor) des DSP geliefert wird - und je einen digital-Wert, der die Drehrichtung der Motoren angibt. Diese werden vom DSP mit Hilfe der Ausgänge IOPA0 und IOPA1 zur Verfügung gestellt. Die vier Signale werden durch das Flachbandkabel von der DSP-System-Platine zur Leistungselektronik-Platine geführt.
Die PWM-Werte für die Motoren werden vom DSP mit Hilfe eines internen Zählers erzeugt, der mit einer Frequenz von 5 MHz (CPU-Clock / 4) zyklisch von 0 bis 255 zählt. Dabei führt der DSP automatisch Vergleiche mit den beiden vorgegebenen PWM-Werten für die Motoren durch. Wird einer der Werte überschritten, wird der entsprechende PWM-Ausgang auf logisch-1 geschaltet. Mit Erreichen des Zählerwerts 255 wird der PWM-Ausgang auf logisch-0 geschaltet und der Zähler auf 0 gesetzt. Es resultiert eine PWM-Frequenz von $5MHz/255 \approx 19,6kHz$, mit der die Motoren gepulst werden.
Die Initialisierung der PWM-Ausgänge und die Funktionen zum Einstellen der PWM-Werte für die Motoren werden in der Quelldatei *pwmmot.c* implementiert.

6.7 Hostrechner-Funksystem

Das Hostrechner-Funksystem ermöglicht dem Hostrechner die Kommunikation mit den Robotern. Es wird mit Hilfe der seriellen RS232-Schnittstelle

an das Hostsystem angeschlossen und ist - im Gegensatz zu den bisherigen Funksystemen - auf Sende- und Empfangsbetrieb ausgelegt. Abbildung 6.3 zeigt den Aufbau, Tabelle 6.3 die Pinbelegung des seriellen Ports. Der Schaltplan sowie das Platinenlayout sind im Anhang beigefügt.
Das Funksystem kann mit einer maximalen Geschwindigkeiten von 40kBit/s (BiM Version 1) bzw. 64kBit/s (BiM Version 2) betrieben werden, höhere Bitraten liegen außerhalb der Spezifikation der Radiometrix-Module [Rad01a]. Angeschlossen wird das Hostrechnermodul einerseits an eine Gleichspannungsquelle, die den Eigenschaften der Roboterspannungsversorgung entspricht, andererseits wird es durch ein serielles Kabel mit dem Hostrechner verbunden. Im allgemeinen existieren zwei verschiedene serielle Kabelarten, nämlich eine direkte Verbindung aller seriellen Pins oder und eine gekreuzte Verkabelung (Nullmodemkabel), die eigentlich für die Verbindung zweier serieller Schnittstellen gedacht ist. Das Hostrechnermodul kann mit Hilfe von Steckbrücken (Jumpern) an die beiden Kabelarten angepasst werden (siehe Abbildung 6.3).

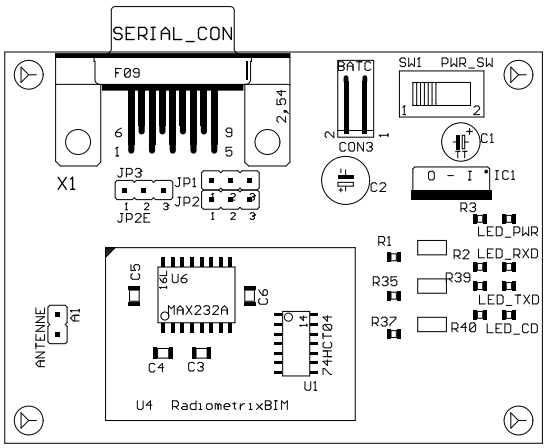


Abbildung 6.3: Bauteillayout des Host-Transceivers

6.7.1 Aufbau der Schaltung

Das Hostrechner-Funksystem wird über den zweipoligen Batterieanschluss mit einer Eingangsgleichspannung zwischen 7 und 12 Volt versorgt, die durch den Spannungsregler U1 (LM2940 Low-Drop 5V Spannungsregler), C1 und C2 auf die für die Elektronik benötigten 5 Volt abgesenkt wird. Das Modul ist mit dem Schalter SW1 ein- und ausschaltbar. Der Anschluss des seriellen Kabels erfolgt über eine übliche 9-polige RS-232 Buchse. Die Art des seriellen Kabels kann durch die Jumper J1, J2, J3 eingestellt werden (Abbildung 6.3 und Tabelle 6.3). Die Pegelanpassung zwischen seriellem Kabel (+12V)

Serielle Schnittstelle des Host-Transceivers				
Pin	*RS232-Funktion*	*Transceiver Funktion*	*Typ*	*Jumper*
#1	Data Carrier Detect (CD)	RF Carrier Detect	out	
#2	Receive Data (RXD)	Receive Data Transmit Data	in out	JP2 1-2 JP1 2-3
#3	Transmit Data (TXD)	Transmit Data Receive Data	out in	JP1 1-2 JP2 2-3
#4	Data Terminal Ready (DTR)	nicht belegt		
#5	Signal Ground (GND)	Ground		
#6	Data Set Ready (DSR)	nicht belegt		
#7	Request To Send (RTS)	TR-Modus High: Transmit-Mode Low: Receive-Mode	in	JP3 1-2
#8	Clear To Send (CTS)	TR-Modus High: Transmit-Mode Low: Receive-Mode	in	JP3 2-3
#9	Ring Indicator (RI)	nicht belegt		

Tabelle 6.3: Serielle Schnittstelle des Host-Transceivers

und dem Funkmodul (CMOS/TTL) wird durch U2, einen seriellen MAX232 Leitungstreiber, vorgenommen. Die Inverter in U3 sorgen für einen fehlerfreien Empfang in eventuellen Sendepausen: der Ausgang des Funkmoduls fällt während einer Sendepause auf *low* (logisch 0) - logisch 0 wird bei der RS232-Schnittstelle allerdings durch ein high-Signal dargestellt. Aus diesem Grund müssen die Pegel der Datenleitungen vor dem Senden und nach dem Empfangen invertiert werden; ansonsten resultieren ständige Datenfehler auf Seiten der Empfänger.
Die Betriebszustände werden über vier grüne und rote LEDs angezeigt, die den Status des Funkkanals angeben. Durch dieses Verfahren wird eine visuelle Kontrolle der folgenden Betriebszustände des Funksystems ermöglicht:

LED1: Betriebsspannung vorhanden (grün)

LED2: Sendedaten TXD (rot)

LED3: Empfangsdaten RXD (grün)

LED4: kein Carrier Detect CD (rot)

Die Platinenlayouts sowie der Schaltplan finden sich im Anhang.

6.7.2 Prototypische Implementierung

Der zu benutzende serielle Port wird über Aufrufe des Betriebssystem-APIs angesprochen. Da diese APIs je nach Betriebssystem unterschiedlich sind, wird eine Library [Bür01] verwendet, die diese Kommunikation für Win32- und Linux-Systeme in einer C++-Klasse kapselt. Diese Library wurde für die besonderen Anforderungen der Radiometrix-Module leicht verändert: insbesondere muss die Funktion der Datenausgabe an den seriellen Port solange in einem Wartezustand verbleiben, bis alle Daten die serielle Schnittstelle verlassen haben, da anschließend das Modul wieder auf Empfangsbetrieb umgeschaltet wird. Geschieht dies, während noch Daten gesendet werden, bricht die Übertragung innerhalb des Datenframes ab.
Die Dateien *rfHalHost.cpp* bzw. *rfHal.h*[6] geben Einblick in die Anbindung der seriellen Schnittstelle und dienen zur Kapselung der betriebssystemspezifischen Implementierung. Die Dateien *rtest.cpp* und *stest.cpp* empfangen bzw. versenden exemplarisch Daten mit Hilfe des in Kapitel 12 entwickelten Protokolls. Details können dem Quelltext auf der beiliegenden CD-Rom entnommen werden.

6.8 Ergebnis

Das entworfene Robotersystem (Abbildung 7.1) wurde erfolgreich aufgebaut und getestet. Alle Vorgaben des Hardware-Entwurfs konnten umgesetzt werden.
Die prototypische Implementierung wurde durchgeführt und eine Kapselung der Details des Hardware-Entwurfs erreicht: die Module des Roboters sind mit Hilfe einfacher Funktionen zu benutzen, ohne beispielsweise die verwendeten Ein- und Ausgänge des DSP kennen zu müssen. Die Quelltexte der Implementierung können der beiliegenden CD-Rom entnommen werden.

[6] Hal = Hardware Abstraction Layer, Vereinheitlichung der verschiedenen Hardwareplattformen.

Kapitel 7

Entwicklungsdokumentation

Der Fußballroboter liegt - zum Abschluss dieser Arbeit - in der vierten Version vor (Abbildung 7.1). In diesem Kapitel wird der Entwicklungsverlauf der einzelnen Komponenten beschrieben. Die Entwicklungs- und Optimierungsprozesse der einzelnen Versionen werden dokumentiert, indem die Entwicklungsstadien sowie die durchgeführten Veränderungen erläutert werden.

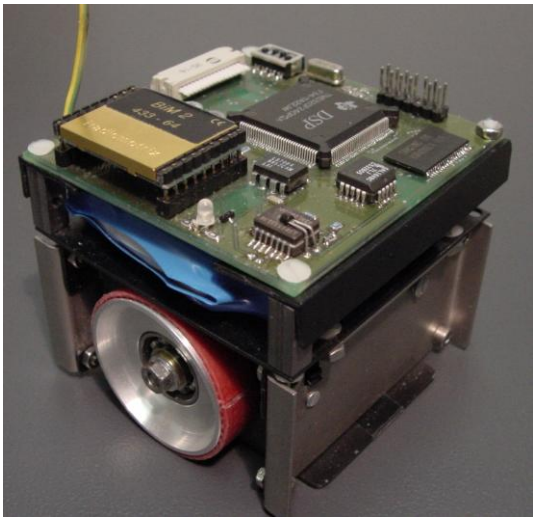


Abbildung 7.1: Entwicklungsergebnis: Der MiroSot-Fußballroboter

7.1 Entwicklung des DSP-Systems

7.1.1 Version 1

Die erste Version des DSP-Systems bestand aus einer Testplatine, auf der ein DSP-Sockel, Hauptspeicher, ein Sockel für das Funkmodul sowie das JTAG-

Interface für die Entwicklungsumgebung vorhanden waren. Diese Platine (siehe Abbildung 7.2) erwies sich bis auf kleinere Layoutfehler als funktionsfähig und wurde zusammen mit einem Prototyp-Aufbau der Leistungselektronik (Abbildung 7.6) erfolgreich getestet. Sie enthielt in diesem Stadium keine Speicherdekodierlogik und konnte aufgrund des Layouts nur im *Mikroprozessormodus* des DSP (in dem nur das externe RAM aktiviert ist) betrieben werden.
Bei den Speicherbausteinen handelte es sich in dieser Version noch um zwei

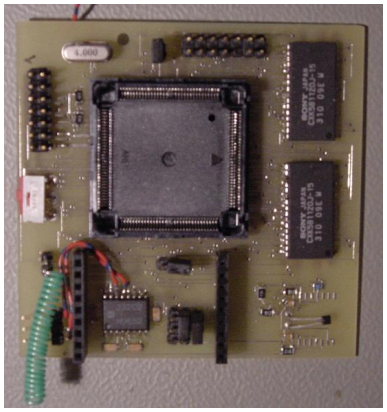


Abbildung 7.2: Erste Version der DSP-Platine

64 kWorte Speicherbausteine (Abbildung 7.2, rechts oben). In der Vorbereitung der zweiten Platinenversion wurden unter anderem folgende Veränderungen vorgenommen:

- Um eine Nutzung des nicht flüchtigen Flash-Speichers zu ermöglichen, wurde eine Umschaltung des DSP-Betriebsmodus vorgesehen.
- Der Sockel des DSP entfällt aus Platzgründen. Der Prozessor wird direkt auf der Platine befestigt.

7.1.2 Version 2

Die zweite Version des DSP-Systems erwies sich - bezogen auf den neu intgrierten *Mikrocontrollermodus* des Prozessors (Nutzung des DSP internen Flash-Speichers) - als fehlerhaft: es konnte keine Software im Flash-Speicher des DSP abgelegt werden. Dies war in der ersten Platinenversion nicht ersichtlich, da die Umschaltung des Betriebsmodus erst in der zweiten Version verfügbar war. Um diesen Fehler zu beheben wurde an Stelle des Beschleunigungs-Sensors eine Speicherdekodierlogik hinzugefügt, die Zugriffe auf den externen Speicher von Zugriffen auf den internen Speicher trennen konnte (vgl. Abbildung 7.3, links oben). Aus diesem Grund war

eine umfassende Layoutänderung notwendig, für die eine dritte Platinenversion angefertigt wurde. Die zweite Platinenversion fand in der abgebildeten, veränderten Form bereits Anwendung auf einer Prototypversion der Robotermechanik.

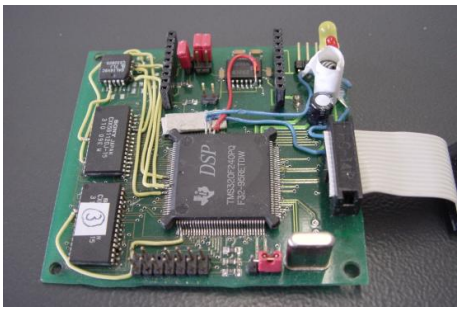


Abbildung 7.3: Zweite Version des DSP-Systems

7.1.3 Version 3

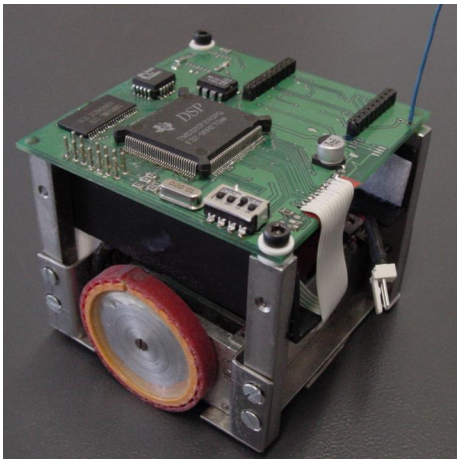


Abbildung 7.4: Version 3 des DSP-Systems mit Roboterprototyp

Für die dritte Version der Platine wurden folgende, wichtige Veränderungen vorgenommen:

- Integration eines GAL16V8 Bausteins zur Dekodierung der Speicherzugriffe.
- Ersetzung der beiden Speicherbausteine durch einen halb so großen Speicherbaustein mit doppelter Kapazität.
- Hinzufügen von DIP-Schaltern zur Einstellung der Roboter-ID.

Die dritte Version der Platine konnte erfolgreich getestet werden und war Grundlage für eine optimierte vierte Version.

7.1.4 Version 4

Die vierte und vorerst endgültige DSP-System-Platine zeigt das Ergebnis der bisherigen Arbeit (Abbildung 7.5). Sie beinhaltet nur kleinere Änderungen und Ergänzungen im Vergleich zur dritten Version, wurde dafür aber in einer Kleinserie von 14 Stück hergestellt. Zum Abschluß dieser Arbeit sind bereits sieben Platinen erfolgreich bestückt und auf Robotermechaniken montiert.

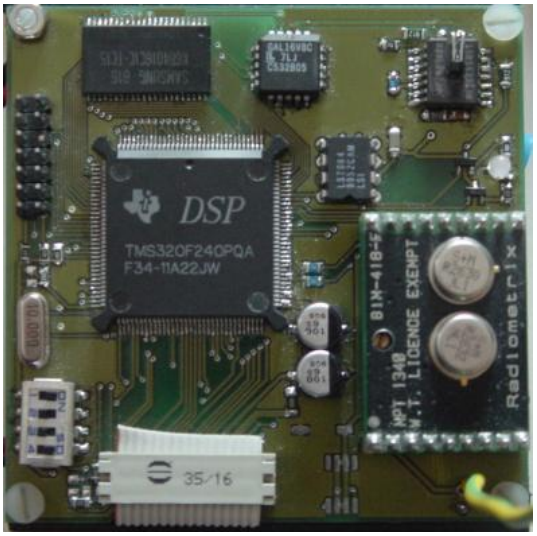


Abbildung 7.5: Vierte Version des DSP-Systems

7.2 Entwicklung der Leistungselektronik und Spannungsversorgung

Die erste Version der Leistungselektronik wurde auf einer Lochrasterplatine entwickelt und getestet (Abbildung 7.6). Anschließend wurde eine erste Version zusammen mit dem ersten Platinenentwurf des DSP-Systems gefertigt und getestet. Es folgten zwei weitere Versionen, deren Funktionalität sich nicht wesentlich von den vorherigen unterschied: ihr Layout wurde an die wechselnden Anforderungen des ebenfalls in der Entwicklung befindlichen Robotergehäuses angepasst, da die Platine im Inneren des Roboters, oberhalb der Motoren angebracht werden muss (Abbildung 7.7).

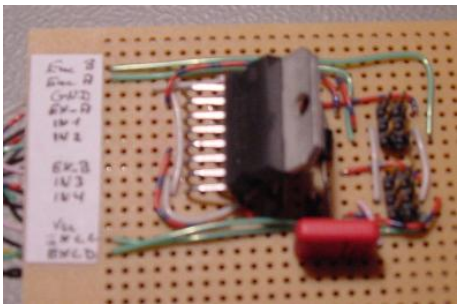


Abbildung 7.6: Testaufbau der Leistungselektronik

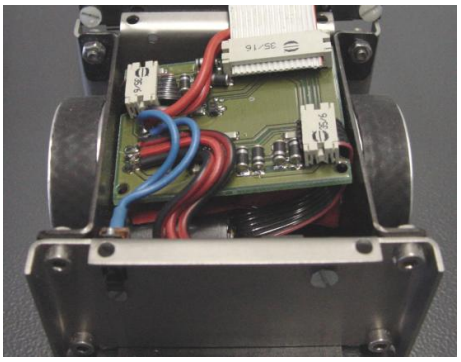


Abbildung 7.7: Aktuelle Version der Leistungselektronik

7.3 Entwicklung des Host-Transceivers

Die Entwicklung des Host-Transceivers benötigte nur eine Platinenentwicklung. Ausgehend von den - als Sendestation ausgelegten - Modulen des Cavalry-Systems wurde die Möglichkeit einer Sende- und Empfangsumschaltung durch die serielle Schnittstelle sowie LEDs zur Statuskontrolle hinzugefügt (Abbildung 7.8).

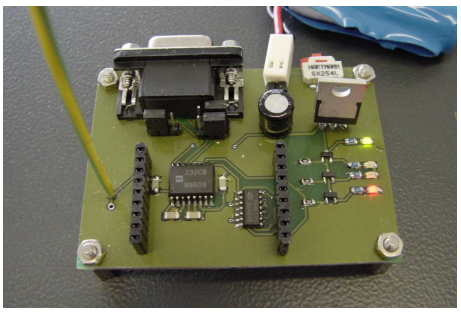


Abbildung 7.8: Host-Transceiver (ohne Funkmodul)

Kapitel 8

Eingebettete Systeme

In diesem Kapitel wird die Anwendbarkeit von Verfahren aus dem Bereich der eingebetteten Systeme bezüglich des vorliegenden Robotersystems untersucht: einige Anforderungen, die an das Robotersystem gestellt werden, korrelieren mit Anforderungen an eingebettete Systeme[1]. Vor allem bezüglich der Echtzeitbedingungen der vom Roboter zu bearbeitenden Aufgaben können Verfahren aus dem Bereich der eingebetteten Systeme angewendet werden. Die Echtzeitanforderungen des Robotersystems werden untersucht und mögliche Verfahren diskutiert, die im weiteren Verlauf dieser Arbeit verwendet werden[2].

8.1 Definition von eingebetteten Systemen

Marwedel [Mar00]$_{S.6}$ gibt vor allem folgende Definitionsmöglichkeiten für ein ES (eingebettetes System) an:

> Eingebettete Systeme sind informationsverarbeitende Systeme, die Größen der Umgebung aufnehmen, verarbeiten und beeinflussen.
>
> Ein eingebettetes System (abgekürzt: ES) ist eine Software/Hardware-Einheit, die über Sensoren und Aktuatoren mit einem Gesamtsystem verbunden ist und darin Überwachungs-, Steuerungs- beziehungsweise Regelungsaufgaben übernimmt. In der Regel handelt es sich bei eingebetteten Systemen um reaktive, häufig auch um hybride verteilte Systeme mit Echtzeitanforderungen. Typischerweise sind solche Systeme dem menschlichen Benutzer

[1] In dieser Arbeit wird mit dem Begriff *eingebettetes System* immer ein *eingebettetes, informationsverarbeitendes System* bezeichnet (vgl. Marwedel [Mar00]$_{S.6}$).

[2] Aufgrund der Echtzeitanforderungen der in späteren Kapiteln vorgestellten Softwaremodule konnten Querverweise nicht vermieden werden.

nicht direkt sichtbar, er interagiert unbewusst mit dem eingebetteten System.

8.1.1 Anwendbarkeit auf Fußballroboter

Ob ein Fußballroboter ein eingebettetes System darstellt, ist bezogen auf die erste Definition einfach zu beantworten: Fußballroboter verfügen zumeist über einen eigenen Prozessor und verarbeiten damit Sensorinformationen ihrer Umwelt - zum Beispiel Funkdaten, Beschleunigung und Radbewegung; die Umgebung wird durch ihre Bewegungen auf dem Spielfeld beeinflusst.
Würde man Fußballroboter ferngesteuert betreiben - wie beispielsweise als Spielwaren erhältliche ferngesteuerte Autos - könnte man nicht von einem ES sprechen, da durch die zumeist direkte Verbindung des Funkempfängers mit den Motoren der informationsverarbeitende Teil (zum Beispiel ein Prozessor) fehlt.
Fußballroboter sind hingegen hochgradig informationsverarbeitend und reaktiv, da sie auf Ereignisse ihrer Umwelt meist unter Einhaltung von Zeitbedingungen reagieren müssen. Die Definitionen der ES sind auf Fußballroboter anwendbar[3]; un gelten etwas weiter gefasst generell für reaktive Roboter und autonome Agentensysteme: jeder Agent nimmt Größen seiner Umwelt auf, verarbeitet diese und beeinflusst mit den Ergebnissen dieser Verarbeitung seine Umwelt.

8.2 Echtzeitbedingungen

Ein wichtiger Bereich der eingebetteten Systeme beschäftigt sich mit der Einhaltung von Zeitvorgaben. Der Prozessor des entworfenen Fußballroboters muss im Normalbetrieb viele (Teil-)Aufgaben „gleichzeitig“ bearbeiten. Je nach Aufgabentyp kann für eine erfolgreiche Ausführung die Einhaltung von Echtzeitbedingungen notwendig sein. Im folgenden Abschnitt wird diese Notwendigkeit für die exemplarisch implementierten Fähigkeiten untersucht und eine Reihe von Maßnahmen vorgestellt, die eine Einhaltung derartiger Echtzeitbedingungen für den Fußballroboter ermöglichen.

8.2.1 Kommunikation

Das Kommunikationsmodul des Roboters besteht aus den im Kapitel 12 vorgestellten Schichten des OSI-Modells: zu versendende Daten durchlaufen das Modell von oben nach unten und werden von der physikalischen Schicht versendet. Empfangene Daten hingegen werden von der physikalischen Schicht

[3] Dies gilt, mit Ausnahme der (Un)sichtbarkeit der zweiten Definition, die sicherlich wünschenswert wäre, um den Gegner zu verwirren.

entgegengenommen und durchlaufen das OSI-Modell in umgekehrter Richtung.
Echtzeitbedingungen müssen dabei vor allem an das Versenden und Empfangen von Daten durch die serielle Schnittstelle gestellt werden: die Baudrate[4] wird durch die Geschwindigkeit der Schnittstelle vorgegeben. Aus diesem Grund muss sichergestellt sein, dass - beim Versenden eines Datenpakets - Bytes innerhalb der vorgegebenen Zeit geliefert und beim Datenempfang innerhalb der geforderten Zeit gelesen werden.
Die serielle Schnittstelle des DSP besitzt Sende- und Empfangspuffer von je einem Byte [Tex99b]$_{S.290ff}$.

Sendedaten

Die Sendedaten werden im OSI-Schichtmodell unterhalb der Applikationsschicht als Datenpakete gehandhabt. Verarbeitungen unterer Schichten behandeln grundsätzlich immer ganze Frames, die neben einem Datenpaket zusätzliche Informationen enthalten und beinahe der physikalisch versendeten Bytefolge entsprechen. Diese Handhabung erlaubt eine gleichzeitige - und damit schnellstmögliche - Übergabe der zu versendenden Bytefolge an die serielle Schnittstelle.
Aufgrund des beschränkten Sendepuffers von nur einem Byte muss die Bytefolge - die eine Länge von bis zu mehreren hundert Bytes erreichen kann - zwischengespeichert werden. Die Übergabe der einzelnen Bytes aus der Zwischenspeicherung an den Sendepuffer des DSP muss innerhalb des Zeitraums geschehen, in dem das zuletzt übergebene Byte von der Schnittstelle versendet wird. Bei einer maximalen Baudrate von 56kBit/s entspricht dies einem Zeitraum von ca. 150 Nanosekunden (für 8 Datenbits + 1 Stoppbit), was bei einem DSP-Takt von 20 MHz ca. 3000 CPU-Zyklen entspricht.

Empfangsdaten

Die Empfangsdaten werden im OSI-Schichtmodell unterhalb der Datenverbindungsschicht in Form von einzelnen Bytes verarbeitet. In der Datenverbindungsschicht werden aus den Daten einzelne Frames extrahiert, auf Korrektheit geprüft und in vollständiger Form an die höherliegenden Schichten weitergereicht. Diese Verarbeitung kann - abhängig von der Bedeutung des empfangenen Bytes für den Frame - längere Zeit in Anspruch nehmen: beispielsweise wird bei der Ankunft eines Stoppbytes die bisher empfangenen Daten mit Hilfe der Prüfsumme auf Korrektheit geprüft und der Frame an die höheren Schichten weitergeleitet, deren Bearbeitungszeit aber unter Umständen unbekannt ist.

[4] Die Baudrate entspricht der Datenübertragungsrate: 1 baud = 1 bit/s.

Für die Empfangsdaten gilt eine Echtzeitbedingung, die der Bedingung der Sendedaten ähnlich ist: der Inhalt des Empfangspuffers muss gelesen werden, bevor die serielle Schnittstelle das nächste Byte vollständig empfangen hat, da der Inhalt des Puffers ansonsten überschrieben wird. Es muss an dieser Stelle ein Lesezugriff auf den Empfangspuffer innerhalb von 150 Nanosekunden garantiert werden können.

Mediumzugriff

Das in der Mediumzugriffsschicht eingesetzte CSMA-Verfahren erfordert eine periodische Zugriffsprüfung des Übertragungskanals innerhalb der Timeslots von einer Millisekunde Dauer. Es resultiert eine weiche Echtzeitbedingung: bei Verletzung werden die Daten in einem späteren Timeslot versendet.

Transportschicht-Uhr

In der Transportschicht wird eine Uhr benötigt, die unter anderem zum Ermitteln von Timeouts der Acknowledge-Pakete verwendet wird. Die Uhr muss periodisch aktualisiert werden; die Frequenz wird in der vorliegenden Implementierung exemplarisch auf 10 Hz festgelegt.

8.2.2 Radencoder-Sensoren

Die Radencoder-Sensoren liefern zwei um 90° versetzte Rechtecksignale, die mit der Radbewegung korrelieren und durch den DSP ausgewertet werden müssen. Eine Signalveränderung tritt bei der Drehung des Rades alle 0,2577 Millimeter auf (siehe Abschnitt 10.2) und muss vom DSP verarbeitet werden, bevor die nächste Signalveränderung eintritt.
Eine Geschwindigkeit von $2,3$ m/s kann als obere Schranke für die Robotergeschwindigkeit angesehen werden, da die Motoren im Leerlauf mit maximaler Akkuspannung keine höhere Geschwindigkeit erreichen. Der Zeitabschnitt zwischen zwei Veränderungen des Quadratursignals beträgt bei dieser Geschwindigkeit ca. 110 Nanosekunden. Da der Roboter zwei dieser Sensoren auswertet, muss mit zwei Signalveränderungen innerhalb dieses Zeitraums gerechnet werden.
Für eine zukünftige Version des Roboters ist die Erhöhung der Sensorauflösung geplant, die eine Verdopplung der Quadratursignale und somit eine Halbierung des Zeitfensters verursacht.

8.2.3 Odometrie

Die Odometrie (siehe Abschnitt 10.2) benötigt eine periodische Aktualisierung der Roboterposition und Ausrichtung unter Verwendung der Rad-

encoder-Sensordaten. Odometrie ist als Näherungsverfahren auf eine häufige Aktualisierung der Daten angewiesen, damit die Bewegung des Roboters möglichst exakt angenähert werden kann. Eine verspätete Aktualisierung der Positionsdaten verschlechtert das Ergebnis der Odometrie, wirkt sich aber ansonsten nicht negativ aus. Die Odometrie unterliegt daher einer weichen Echtzeitbedingung: eine Aktualisierung der Positionsdaten im Rahmen einer Millisekunde ist für die gewünschten Geschwindigkeiten vollkommen ausreichend.

8.2.4 Inertiale Navigation

Das inertiale Navigationssystem (siehe Abschnitt 10.3) benötigt - wie die Odometrie - eine periodische Aktualisierung der Beschleunigungsdaten. Die Beschleunigungswerte sollten mit der Frequenz in Berechnungen des inertialen Navigationssystems einfließen, mit der sich die Bewegungsgeschwindigkeit des Roboters verändern kann. Diese Forderung korreliert mit einer Anforderung der Odometrie, da diese ebenfalls davon ausgeht, dass die Bewegung zwischen den Aktualisierungen gleichförmig verläuft.
Eine verspätete Messung und Berechnung verschlechtert bei diesem Verfahren ebenfalls das Ergebnis, hat aber keine Fehlfunktion zur Folge. Die inertiale Navigation unterliegt somit ebenfalls einer weichen Echtzeitbedingung.

8.2.5 Steuerung und Regelung

Die in Kapitel 9 diskutierte Steuerung der Antriebsmotoren unterliegt einer weichen Echtzeitbedingung. Der Regler muss in möglichst periodischen Abständen aufgerufen werden, um aus den Daten der Radencoder-Sensoren und der seit dem letzten Aufruf vergangenen Zeit die Geschwindigkeit der Räder zu berechnen. Die PWM-Ansteuerung der Motoren wird dem Regelungsergebnis entsprechend korrigiert. Die Frequenz der Regelung hängt von der Zeit ab, die der Motor benötigt, um auf eine Veränderung des PWM-Werts zu reagieren. Eine zu hohe Regelungsfrequenz verschwendet Rechenzeit, da der Motor erst nach mehreren Regelzyklen auf die Änderungen reagiert; bei einer zu niedrigen Regelungsfrequenz hingegen kann durch eine Erhöhung der Frequenz unter Umständen ein verbessertes Regelungsverhalten erzielt werden. Um eine optimale Regelungsfrequenz zu ermitteln, wird die Sprungantwort des Motors untersucht: die Akkuspannung wird an den ruhenden Motor geschaltet und die Reaktionszeit des Motors gemessen. Abbildung 8.1 lässt eine Verzögerung von ca. $4 - 5$ ms erkennen. Die Regelungsfrequenz sollte daher unterhalb von 200 Hz liegen.

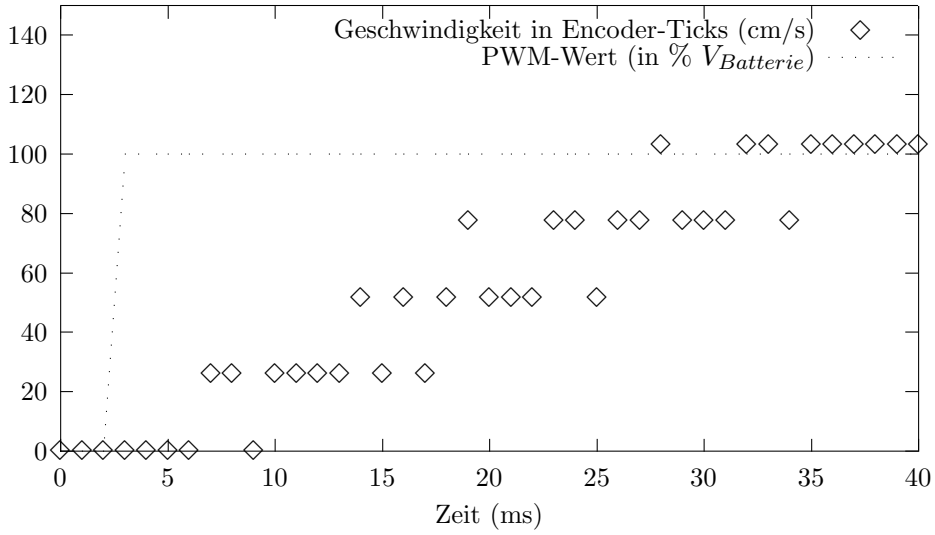


Abbildung 8.1: Sprungantwort des Motors

8.2.6 Ereignisdetektion

Die in Kapitel 11 vorgestellte Ereignisdetektion benutzt den Beschleunigungs-Sensor, um hochfrequente Schwingungen zu detektieren, die zum Beispiel bei einer Kollision mit anderen Objekten auftreten. Für die Detektion sind Beschleunigungsmessungen notwendig, deren Frequenz doppelt so hoch sein sollte, wie die maximal für die Detektion oder Klassifizierung benötigte Frequenz (Abtasttheorem von Whittaker und Shannon [Whi15], [Sha49]). Die in dieser Arbeit verwendete Frequenz von 1 kHz reicht zur Detektion von Ereignissen aus; ob dieser Wert für eine Klassifizierung ausreichend ist, kann in zukünftigen Arbeiten untersucht werden. In dieser Arbeit wird von einer harten Echtzeitbedingung für die periodische Messung der Beschleunigungen mit einer Frequenz von 1 kHz ausgegangen.

8.2.7 Watch-Dog

Der DSP verfügt über einen sogenannten Watch-Dog-Timer, der das DSP-System überwacht. Der Watch-Dog muss mindestens einmal pro Sekunde zurückgesetzt werden; sonst geht er davon aus, dass sich das DSP-System in einem fehlerhaften Zustand befindet und veranlasst den Reset des Systems. Mit diesem Verfahren wird verhindert, dass ein ausser Kontrolle geratener Roboter längere Zeit in einem fehlerhaften Zustand verbleibt. Es resultiert eine harte Echtzeitbedingung für den Reset des Watch-Dog-Timers von einer Sekunde.

Echtzeitbedingungen			
Modul	*Aufgabe*	*Typ*	*Zeitvorgabe*
Sensorik	Radencoder	reaktiv, hart	55 ns
Kommunikation	Datenempfang	reaktiv, hart	150 ns
Kommunikation	Datenversand	reaktiv, hart	150 ns
Sensorik	Ereignisdetektion	periodisch, hart	1 ms
Sensorik	Odometrie	periodisch, weich	1 ms
Sensorik	inertiale Navigation	periodisch, weich	1 ms
Kommunikation	CSMA	periodisch, weich	1 ms
Steuerung	PID-Regelung	periodisch, weich	5 ms
Kommunikation	Transportschicht-Uhr	periodisch, weich	100 ms
Prozessor	Watch-Dog-Timer	periodisch, hart	¡ 1 s

Tabelle 8.1: Echtzeitbedingungen (nach Priorität geordnet)

8.3 Umsetzung

Die erarbeiteten Echtzeitbedingungen (siehe Tabelle 8.1) werden im folgenden Abschnitt mit Hilfe verschiedener Verfahren realisiert.

8.3.1 Hardware-Realisierung

Der TMS320F240-DSP bietet für die besonders zeitkritische Dekodierung der Radencoder-Signale eine Hardwareunterstützung an, da er - im Gegensatz zu den meisten anderen Mikroprozessoren - speziell für Motion-Control-Aufgaben konzipiert wurde. Eingänge, die eine automatische Dekodierung des Quadratursignals vornehmen und das Ergebnis in einem 16-bit Zähler verfügbar machen, sind vorhanden. Durch dieses Verfahren wird die korrekte Messung der Raddrehung garantiert und die Echtzeitbedingung erfüllt; gleichzeitig wird der DSP nicht durch eine ansonsten erforderliche Software-Realisierung mit Hilfe von Interrupts belastet, so dass mehr Rechenkapazität für andere Aufgaben zur Verfügung steht.

8.3.2 Interrupts

Interrupts sind vom Prozessor ausgelöste Unterbrechungen des Programmablaufs. An Stelle des weiteren Programms wird eine Interrupt-Routine ausgeführt, die auf die Unterbrechung reagieren muss. Die Ausführungsdauer dieser Routine sollte möglichst kurz sein, da innerhalb eines Interrupts keine weiteren Unterbrechungen ausgeführt werden können. Eine zu lange Interrupt-Behandlung würde daher die weitere Signalisierung von Systemereignissen verzögern und die Einhaltung der Echtzeitbedingungen gefährden.

Die Echtzeitbedingung des Datenempfangs wird durch eine Zwischenschicht unterhalb des OSI-Schichtmodells erfüllt: da die Verarbeitung innerhalb der OSI-Schichten den kurzen Zeitrahmen der Echtzeitbedingung nicht einhalten kann, werden empfangene Bytes vorerst in einem Ringpuffer[5] zwischengespeichert. Der Empfang eines Bytes wird vom DSP durch einen Interrupt signalisiert; die vorgesehene Interrupt-Routine liest das Byte ein und speichert es im Ringpuffer ab.
Die Verarbeitung innerhalb des OSI-Schichtmodells kann mit Hilfe dieses Verfahrens gelegentlich erfolgen - beispielsweise in Abhängigkeit vom Füllstand des Puffers. Durch dieses Verfahren wird der Zeitrahmen der harten Echtzeitbedingung - entsprechend der Größe des Ringpuffers - erweitert. Die Größe des Puffers kann so gewählt werden, dass der resultierende Zeitrahmen von den Schichten des OSI-Modells eingehalten werden kann.
Der Versand der Daten erfolgt analog: die physikalische Schicht speichert die Bytes des zu versendenden Pakets in einem Ringpuffer. Bei Auftreten des Interrupts, der die Verfügbarkeit des DSP-Sendepuffers signalisiert, wird ein Byte aus dem Ringpuffer in den Sendepuffer des DSPs geschrieben.

8.3.3 Real-Time Interrupt

Die Echtzeit-Unterbrechung (Real-Time Interrupt) ermöglicht eine periodische Ausführung von Funktionen, um wiederholte, zeitbedingte Echtzeitanforderungen zu erfüllen. Die Frequenz des Real-Time Interrupts orientiert sich am kleinsten Zeitrahmen der verbleibenden unerfüllten Echtzeitbedingungen: 1 ms, was einer Frequenz von 1 kHz entspricht[6]. In der Interrupt-Routine wird eine Art interrupt-basiertes Scheduling ausgeführt. Zu festgelegten Zeiten (jeweils nach n-Real-Time Interrupts) werden Funktionen zur Bearbeitung der restlichen, periodischen Aufgaben aufgerufen. Es muss allerdings sichergestellt sein, dass die Ausführungsdauer dieser Funktionen in der Summe kürzer als eine Millisekunde ist, da ansonsten der nachfolgende Real-Time Interrupt blockiert wird. Dies kann bei allen verbleibenden Aufgaben sichergestellt werden, da es sich zumeist nur um kurze Berechnungen handelt.

8.3.4 Übergeordnete Aufgaben

Das Hauptprogramm des Softwaresystems kann folglich übergeordnete Aufgaben bearbeiten, ohne sich um die Einhaltung der Echtzeitbedingungen

[5] Ein Ringpuffer erlaubt die Implementierung einer Warteschlage mit Hilfe eines Arrays. Sequenzielles Schreiben und Lesen von Bytes ist mit Zugriffszeiten in $O(1)$ [CLR94]$_{S.201ff}$ möglich.

[6] Durch die DSP-interne Organisation des Interrupts ist nur eine Frequenz von 976 Hz konfigurierbar.

kümmern zu müssen. Falls mehrere übergeordnete Aufgaben existieren, kann die Verteilung der Rechenzeit mit Hilfe von Scheduling-Verfahren durchgeführt werden - was für die prototypische Implementierung in dieser Arbeit nicht benötigt wird.

8.3.5 Ergebnis und zukünftige Arbeiten

Es ist mit Hilfe des DSPs gelungen, alle erarbeiteten Echtzeitanfoderungen der exemplarischen Implementierung zu erfüllen. Die Hardware-Realisierung der Radencoder-Dekodierung leistet einen wesentlichen Beitrag zur Entlastung des Prozessors und erhöht die verfügbare Rechenzeit. Die restlichen Echtzeitanfoderungen sind mit Hilfe von Interrupts erfüllbar.
Zukünftige Arbeiten könnten sich mit dem Einsatz von Echtzeit-Betriebssystemen (RTOS) befassen: in Zukunft ist eher eine komplexere Aufgabenstruktur zu erwarten, die mit Hilfe von vorgefertigten Softwarekomponenten besser handhabbar sein dürfte.
Die Handhabung verschiedener übergeordneter Aufgaben - beispielsweise eine Strategieplanung oder Sensordatenverarbeitung - könnte alternativ auch durch die separate Implementierung eines Scheduling-Verfahrens ermöglicht werden, das in der vorliegenden Implementierung bisher nur interrupt-basiert realisiert wurde.

Kapitel 9

Steuerung und Regelung

Ein Ziel der Konzeption besteht in der Möglichkeit, hardwareunabhängige Steuerungs- und Regelungsalgorithmen mit Hilfe des Robotersystems zu realisieren (siehe Abschnitt 4.1.3). Im folgenden Kapitel wird ein Überblick über die Steuerungs- und Regelungsproblematik des vorliegenden Robotertyps gegeben und anhand der Implementierung einer einfachen Regelung gezeigt, dass der vorliegende Roboter besagtes Konzeptionsziel erfüllt.

9.1 Problematik

Der entwickelte Roboter kann sich grundsätzlich nur im zweidimensionalen Raum innerhalb seines Spielfelds bewegen. Durch seine Konstruktion als differentieller zweiräderiger Roboter (differential wheel robot) ist er in der Bewegungsmöglichkeit - aufgrund seines Antriebs - eingeschränkt (underactuated) und kann sich nicht seitlich bewegen. Er muss sich vielmehr vorwärts oder rückwärts bewegen und kann durch unterschiedliche Geschwindigkeiten der beiden Räder Drehungen ausführen. Er entspricht damit einem von Braitenberg [Bra84] bereits 1984 beschriebenen Vehikel[1].
Derartige Systeme werden - falls die Freiheitsgrade der Geschwindigkeiten des Objekts eingeschränkt sind - auch als *non-holonomic* (von *holo nomos*: gänzlich beschreibbar) bezeichnet. Das resultierende kinematische Modell, das unter anderem vom Indiveri [Ind01] untersucht wird, unterliegt dem Theorem von Brockett [Bro83]. Brockett zeigt unter anderem, dass sich lineare Steuerungs- und Regelungskonzepte nicht für die übergeordnete Steuerung des vorliegenden Robotermodells eignen, wie von Yang und Kim [YK99] beschrieben wird:

[1] Braitenberg beschreibt in seinem lesenswerten Buch *Vehicles: Experiments in Synthetic Psychology* bereits 1984 einfache Sensor-Aktor Kopplungen, die erstaunlich komplexes Verhalten erzeugen können.

> A well-known work of Brockett ... identifies non-holonomic systems as a class of systems that cannot be stabilized via smooth state feedback. It implies that problems of controlling nonholonomic systems cannot be applied to methods of linear control theory, and they are not transformable into linear control problems.

Vorschläge zur Steuerung des in dieser Arbeit verwendeten Robotertyps existieren und wurden erfolgreich implementiert; exemplarisch werden Indiveri [Ind01] und Kim [YK99] genannt.

9.2 Entwicklung eines einfachen Reglers

Die prinzipielle Möglichkeit der Umsetzung von Regelungskonzepten durch Software-Implementierung wird mit Hilfe des Roboter-DSPs gezeigt; dabei wird ein einfacher Regler $[JF96]_{S.216ff}$ adaptiert.
In dieser Arbeit können weder die Vor- und Nachteile einzelner Regelungskonzepte diskutiert werden, noch kann eine vollständige Implementierung eines solchen geleistet werden; dies würde den Umfang der vorliegenden Arbeit überschreiten.

9.2.1 Anforderungen an den Regler

Folgende einfache Anforderungen werden an den zu entwerfenden Regler gestellt:

- Der Roboter soll einer Strecke geradlinig folgen können, so dass sich während der Fahrt keine signifikanten Änderungen in der Ausrichtung ergeben und der Roboter seitlich nicht signifikant von der Strecke abweicht.
- Der Roboter soll seine Ausrichtung ändern können, um Drehungen zu ermöglichen.
- Der Roboter soll zu große Beschleunigungswerte begrenzen und eine Beschleunigungs- und Bremsphase durchführen - zu starke Beschleunigung kann zum Durchdrehen der Räder führen, was sich negativ auf das Anfahr- und Bremsverhalten sowie auf die interne Positionsberechnung auswirkt und daher vermieden werden sollte.

9.2.2 Aufbau des Reglers

Der Regler besteht aus zwei Teilen: zum einen aus jeweils einem PID-Regler für die Geschwindigkeitsregelung der beiden Räder und zum anderen aus

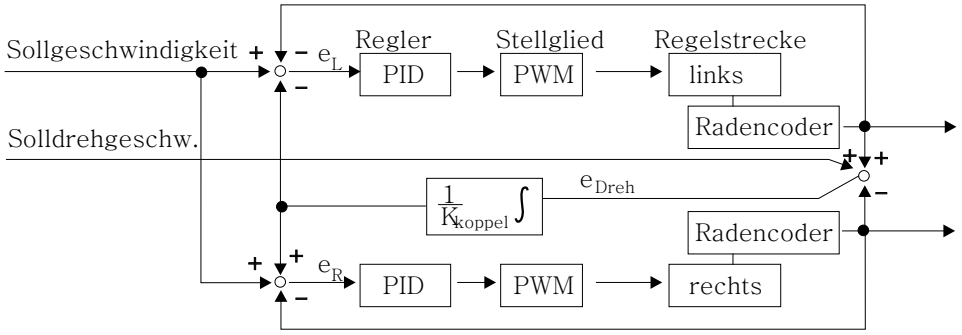


Abbildung 9.1: Aufbau des Reglers

einer Radkopplung, die für eine Synchronisation der Räder sorgt und zusätzlich die Möglichkeit bietet, Drehbewegungen in die Regelung zu integrieren. Der Regler (Abbildung 9.1) bezieht seine Sensorinformationen ausschließlich aus den Radencoder-Sensoren (dies wird im weiteren Verlauf dieser Arbeit in Abschnitt 10.2 verwendet, um die Genauigkeit des Odometrieverfahrens zu messen).
Eine PID-Regelung der einzelnen Räder ist durchaus sinnvoll, auch wenn es sich bei klassischen PID-Reglern um lineare Regler handelt. Üblicherweise werden von übergeordneten Steuerungen Sollwerte für die Geschwindigkeiten der einzelnen Räder vorgegeben. Da die untergeordnete Geschwindigkeitsregelung eines einzelnen Rades von einem linearen Regler zufriedenstellend gelöst werden kann, ist die PID-Regelung der Radgeschwindigkeiten eine mögliche Grundlage für übergeordnete nicht-lineare Steuerungs- und Regelungskonzepte.

Radregelung

Die Regelung der Räder erfolgt über je einen PID-Regler (Abbildung 9.2), wie er von Kiendl [Kie97]$_{S.16f}$ beschrieben wird:

$$u = K_R(e + \frac{1}{T_n}\int e(t')dt + T_v\frac{de}{dt}) \tag{9.1}$$

$$\begin{aligned} e &= \text{Fehler} \\ u &= \text{Stellgröße} \\ K_R &= \text{Reglerverstärkung} \\ T_n &= \text{Vorhaltzeit} \\ T_v &= \text{Nachstellzeit.} \end{aligned}$$

Die Implementierung des Reglers ist in zeitdiskreter Form aufgebaut. Der Fehler e wird zu diesem Zweck nach einer bestimmten Zeit gemessen, und

die Stellgröße u - den Parametern entsprechend - berechnet.

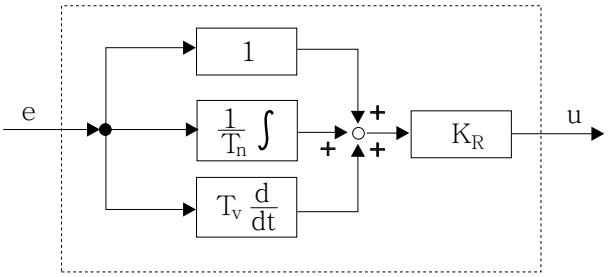


Abbildung 9.2: Struktur des PID-Reglers

Der Proportionalteil des Reglers in Verbindung mit der Reglerverstärkung bzw. dem Proportionalitätsfaktor K_R gibt an, wie sich die Stellgröße bezogen auf den absoluten Fehler e verändern soll.
Der Integralteil summiert den Fehler über die Zeit und gibt an, wie sich das bisherige Fehlerintegral auf die Stellgröße auswirken soll.
Der Differentialterm beschreibt schließlich die Auswirkung der Fehleränderung auf die Stellgröße.
Der bisher beschriebene Regler entspricht dem klassischen PID-Reglerkonzept und wird um einige Eigenschaften industrieller PID-Regler erweitert [Kie97]$_{S.43ff}$:

- Ein Begrenzungsglied schränkt die Stellgröße auf den maximalen Stellbereich der PWM-Motorsteuerung von -255 bis 255 ein.
- Ein Anti-Windup-Hold hält den Integralteil des Reglers konstant, falls das Begrenzungsglied anspricht; damit wird das unerwünschte Hochlaufen des Fehlers im Integralteil verhindert.

Um die geforderte maximale Beschleunigung nicht zu überschreiten, wird dem Regler ein Filter vorgeschaltet, das die Geschwindigkeit schrittweise anpasst, falls die Änderung der Sollgeschwindigkeit die maximal erlaubte Beschleunigung überschreitet. Die Eigenschaften des Filters entsprechen denen eines Hochpassfilters.
Auf diese Weise wird eine geregelte Beschleunigungs- und Bremsphase erzwungen, was auch als Regelung unter Berücksichtigung von Geschwindigkeitsrampen bezeichnet wird.
Die in der Bremsphase zurückzulegende Strecke muss allerdings im Voraus eingerechnet werden, da sich der Roboter bei abruptem Stoppbefehl (dem Setzen der Sollgeschwindigkeit auf 0) aufgrund der Bremsphase noch eine bestimmte Strecke weiterbewegt; diese lässt sich einfach berechnen. Für ein Stoppen ausgehend von der Anfangsgeschwindigkeit v gilt die Beziehung:

$$v = at \tag{9.2}$$

eingesetzt in

$$s = \frac{1}{2}at^2 \tag{9.3}$$

ergibt

$$s = \frac{1}{2}a\left(\frac{v}{a}\right)^2 = \frac{v^2}{2a} \tag{9.4}$$

s = Bremsstrecke
a = maximale Beschleunigung
v = Anfangsgeschwindigkeit.

Radkopplung

Der zweite übergeordnete Teil des Reglers besteht aus einem einfachen Integralregler, der die Bewegungsabweichungen des linken und rechten Rades summiert und die Sollgeschwindigkeit der Räder durch einen Faktor K_{koppel} beeinflusst (siehe Abbildung 9.1). Auf diesem Weg werden die beiden PID-Regler gekoppelt, und es wird beispielsweise sichergestellt, dass sich das rechte Rad nicht weiterdreht, falls das linke Rad blockiert ist.
Der Drehungssollwert - eine externe Vorgabe - ermöglicht eine Drehung des Roboters, indem künstlich ein Ausrichtungsfehler erzeugt wird, der vom Regler korrigiert wird. Auf diese Weise wird eine kontrollierte Drehung und folglich das Fahren von Kreisbahnen möglich.

9.3 Implementierung

In der durchgeführten Implementierung wird die Regelung mit einer Frequenz von ca. 122 Hz betrieben, die für eine exemplarische Implementierung ausreicht. Es sollen geringe Geschwindigkeiten von bis zu 50 mm/s gefahren werden: die maximale - innerhalb eines Zeitabschnitts von ca. 8 ms gefahrene - Strecke liegt unter 0.5 cm. Der DSP des Roboters ist in der Lage, diese Berechnung mit einer Frequenz bis zu 4096 Hz durchzuführen, was für die zukünftig gewünschten Maximalgeschwindigkeiten von ca. 2 m/s mehr als ausreicht.
Der Quellcode ist der Datei *pid.c* der Roboterquellen auf der beilegenden CD-Rom zu entnehmen und wird nur in Auszügen dargestellt (Abbildung 9.3).

```
#define MAXU (255)

int pid(pidInfo *p, int ticks) {
  static float e,u;
  e = p->speed - (float)ticks; /* Fehler */
  p->ie += e;                  /* Integral */
  /* Berechnung der Stellgroesse */
  u = p->KR*(e+(1/p->TN)*p->ie+p->TV*(e - p->eAlt));
  p->eAlt = e;
  if (u>MAXU) {                /* Begrenzungsglied */
    u=MAXU;
    p->ie -= e;                /* Anti-Wind-Up Hold*/
  } else if (u<-MAXU) {
    u=-MAXU;
    p->ie -= e;                /* Anti-Wind-Up Hold*/
  }
  return((int)u);
}
```

Abbildung 9.3: PID-Regler des Roboters

9.3.1 Einstellung des Reglers

Zur Einstellung des Reglers wurden die bekannten Einstellregeln von Ziegler und Nichols [ZN42] verwendet. Die minimale Reglerverstärkung, bevor der Regler zu schwingen beginnt ($K_{R,k}$), kann ermittelt und die dazugehörige Schwingungsdauer T_k gemessen werden. $K_{R,k}$ wird experimentell bestimmt, indem die Reglerverstärkung (K_R) ausgehend von 0 schrittweise erhöht wird, bis der Regler zu schwingen beginnt. Für eine erste Einstellung des PID-Reglers können dann folgende Werte verwendet werden:

$$K_R = 0,6 * K_{R,k} \tag{9.5}$$

$$T_n = 0,5 * T_k \tag{9.6}$$

$$T_v = 0,12 * T_k \tag{9.7}$$

Die resultierenden Reglereinstellungen der beiden PID-Regler können sicherlich weiter optimiert werden; die ermittelten Werte führten allerdings im praktischen Test bereits zum gewünschten Ergebnis. Bestimmt wurde für $K_{R,k}$ ein Wert von 16, T_k liegt bei ca. 30 Hz beziehungsweise bei 4 Zeit Einheiten für eine Regelfrequenz von 122 Hz: es ergibt sich $T_n = 2$ und $T_v = 0,48$.
Die Einstellung des Integralreglers für die Radkopplung ist experimentell zu ermitteln und kann nur für eine vorgegebene maximale Geschwindigkeit eingestellt werden. Dies ist ein Nachteil des vorgestellten Reglers; er neigt bei höheren Geschwindigkeiten zu Schwingungen bezüglich der Ausrichtung des Roboters. Es wurde für Geschwindigkeiten von bis zu 50 mm/s ein Wert

für K_{koppel} von 300 ermittelt.

9.4 Ergebnis

Die vorliegende Implementierung demonstriert die Möglichkeit einer flexiblen Software-Implementierung von Regelungskonzepten mit Hilfe des Roboter-DSPs.
Der erstellte Regler ist in der Lage, die gewünschten Anforderungen bei vorgegebener Maximalgeschwindigkeit zu erfüllen. Messungen zeigen eine maximale seitliche Abweichung von ±3 mm bei der Fahrt einer geradlinigen Trajektorie von 1,6 Metern Länge[2] (Abbildung 9.4).

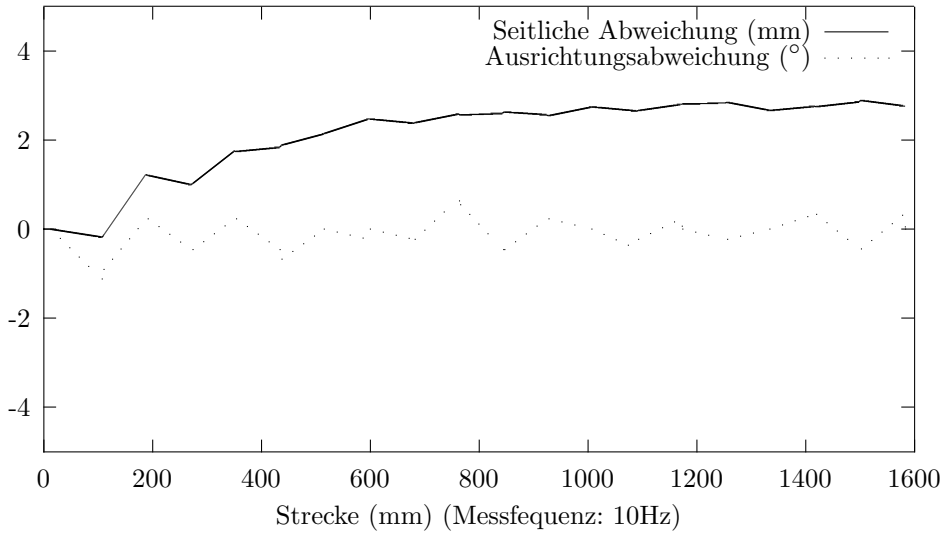


Abbildung 9.4: Geregelte geradlinige Fahrt

9.5 Ausblick und zukünftige Arbeiten

Für den Spielbetrieb ist die Implementierung eines komplexeren Steuerungskonzepts wünschenswert, um höhere Geschwindigkeiten und eine exaktere Steuerung zu erzielen.
Das von Indiveri [Ind01] vorgestellte Konzept erlaubt das Anfahren von Punkten auf dem Spielfeld mit exponentieller Konvergenz des Abstands- und des Ausrichtungsfehlers gegen Null.
Das von Kim [YK99] entwickelte Konzept ermöglicht das Verfolgen einer

[2] 1,6 Meter entspricht der Länge der MiroSot-Spielfeldes.

vorgegebenen Trajektorie, wobei sich die Roboterbewegung der Vorgabe asymptotisch annähert. Beide Konzepte können im Umfeld des Roboterfußballs benutzt werden, um beispielsweise einen Schuss durchzuführen und die Ballposition anzusteuern. Um eine neue Position auf dem Spielfeld anzufahren und Kollisionen mit anderen Robotern zu vermeiden, kann eine mögliche Trajektorie ermittelt und verfolgt werden.
Eine Implementierung und Evaluierung der vorgeschlagenen Konzepte sind eine Anregung für weitere Arbeiten.

Kapitel 10

Navigation

Im diesem Kapitel werden mögliche Navigationsverfahren - aufgrund der in Abschnitt 4.1 erarbeiteten Anforderungen an die Navigationseigenschaften des zu entwickelnden Roboters - vorgestellt und die zu diesem Zweck in Kapitel 5 integrierten Sensoren evaluiert.

10.1 Möglichkeiten des entworfenen Systems

Ziel der Neuentwicklung ist die Ermöglichung eines verbesserten Steuerungs- und Regelungsverhaltens sowie ein höherer Grad an Autonomie bezüglich der Navigation (siehe Abschnitt 4.1.3).
Durch die Ansteuerung der Motoren mit Hilfe des DSPs wird eine Softwarerealisierung der Steuerung und Regelung möglich. Zu diesem Zweck kann der Regelkreis auf den DSP des Roboters verlagert werden, was einen schnellen Zugriff auf die Daten der Radencoder- und Beschleunigungswerte des Roboters garantiert. Latenzzeiten der Bildverarbeitung und der Funkschnittstelle werden durch diese Maßnahme vermieden und fehlerhafte Erkennungen der globalen Bildverarbeitung - die zu gänzlich falschen Ist-Werten führen - ausgeschlossen. Vom Hostrechner gesendete Positionsdaten können zusätzlich zur Überprüfung und Korrektur der lokal ermittelten Daten dienen. Um solche komplexen und umfassenden Steuerungen und Regelungen zu ermöglichen, ist die Berechnung der Roboterposition anhand der lokalen Sensordaten von großem Vorteil. Im Folgenden wird die Realisierung von zwei möglichen Verfahren - Odometrie und inertiale Navigation - mit Hilfe der integrierten Sensorik entwickelt und implementiert, und die resultierenden Ergebnisse werden evaluiert.

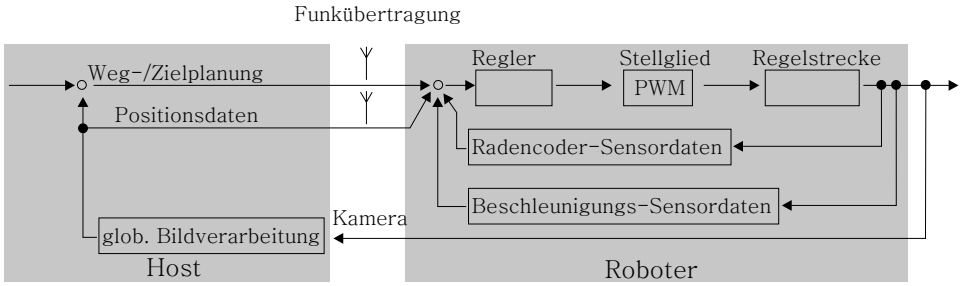


Abbildung 10.1: Möglicher Regelkreis der Roboterneuentwicklung

10.2 Odometrie

Odometrie ist für radgetriebene mobile Roboter das übliche Verfahren zur Berechnung von Position und Ausrichtung. Es basiert auf der bereits vorgestellten deduktiven Berechnung und benutzt die in die Motoren integrierten Radencoder-Sensoren.

10.2.1 Beschreibung und Herleitung

Während der Bewegung des Roboters liefern die integrierten Encoder-Sensoren exakte Daten über die Drehung der Motoren und somit über die Drehung der Räder. Die Encoderdaten werden vom DSP automatisch dekodiert und die Ergebnisse innerhalb des DSPs durch zwei 16-bit Zähler verfügbar gemacht, die mit c_L (linkes Rad) und c_R (rechtes Rad) bezeichnet werden. Die Dekodierung der Quadratursignale wird durch DSP-interne Hardware[1] realisiert, da sich die hohe Zahl von Impulsen (ca. 6200 Impulse pro Sekunde bei Höchstgeschwindigkeit) nicht durch Software-Interrupts bearbeiten lässt, und die Gefahr besteht, Impulse aufgrund von kollidierenden Interrupts nicht zu zählen. Die Hardwarerealisierung **garantiert** die Exaktheit der Sensordaten.
Die Veränderung $\Delta c = c_{t_2} - c_{t_1}$ eines der beiden Zähler innerhalb eines Zeitintervalls $\Delta t = t_2 - t_1$ liefert sowohl die Bewegungsrichtung ($sgn(\Delta c)$) als auch die Länge des zurückgelegten Weges ($|\Delta c|$) des Rades, gemessen in Encoderticks:

Weglänge

Für die Anzahl der Encoderticks e_{Rad} pro Radumdrehung gilt:

$$e_{Rad} = e_{Motor} * 4 * g \tag{10.1}$$

[1] Genauer gilt, dass einer der Sensoren durch den DSP, der zweite durch einen externen Dekodierschaltkreis realisiert wird.

$$
\begin{aligned}
&= 16 * 4 * 8 : 1 \\
e_{Rad} &= 512
\end{aligned}
$$

e_{Motor} = Encoderticks pro Motorumdrehung

g = Getriebeuntersetzung vom Motor zum Rad.

Für einen Durchmesser der Räder d_{Rad} von 42 Millimetern ergibt sich die Strecke pro Tick s_e wie folgt:

$$
\begin{aligned}
s_e &= \frac{d_{Rad} * \pi}{e_{Rad}} && (10.2) \\
&= \frac{42mm * \pi}{512} \\
&\approx 0,2577mm. && (10.3)
\end{aligned}
$$

Die nach Δt zurückgelegte Strecke Δs entspricht dem Mittelwert aus den Strecken des linken und rechten Rades (Δs_l und Δs_r), also

$$
\begin{aligned}
\Delta s &= (\Delta s_l + \Delta s_r)/2 \\
&= (\Delta c_L * s_e + \Delta c_R * s_e)/2. && (10.4)
\end{aligned}
$$

Bei geradliniger Bewegung des Roboters gilt beispielsweise $\Delta s = \Delta s_l = \Delta s_r$. Bei einer entgegengesetzten Bewegung der Räder ($\Delta s_l = -\Delta s_r$), was einer Drehung um den Mittelpunkt des Roboters entspricht, ist $\Delta s = 0$.

Drehung

Um die Ausrichtungsänderung (Drehung) des Roboters nach Δt zu ermitteln, muss die Auswirkung der zurückgelegten Weglängen der beiden Räder auf die Drehung des Roboters untersucht werden. Eine Drehung des Roboters wird durch unterschiedlich schnelle Bewegung der beiden Räder bewirkt: läuft das rechte Rad schneller, dreht sich der Roboter nach links; läuft das linke Rad schneller erfolgt die Drehung nach rechts. Die Geschwindigkeitsdifferenz bewirkt eine Differenz zwischen der vom linken und rechten Rad zurückgelegten Strecke; diese ist proportional zur Drehung des Roboters:
Bei einer Drehung um 360° gilt für die Streckendifferenz Δs_d:

$$
\begin{aligned}
\Delta s_d &= \Delta s_l - \Delta s_r \\
&= 2\pi r \\
&= 2\pi d_{Radabst} && (10.5) \\
&= 2\pi 6,5cm \\
&\approx 40,84cm && (10.6)
\end{aligned}
$$

$d_{Radabst}$ = Abstand der beiden Räder.

Es gilt folglich für den Drehwinkel $\Delta\varphi$ im Gradmaß

$$\Delta\varphi = \frac{360° \Delta s}{2\pi d_{Radabst}} \tag{10.7}$$

beziehungsweise im Bogenmaß

$$\Delta\varphi = \frac{\Delta s}{d_{Radabst}}. \tag{10.8}$$

Position

Die aktuelle Roboterposition berechnet sich aus der letzten bekannten Roboterposition und der relativen Bewegung des Roboters:

$$x_{t_i} = x_{t_{i-1}} + \Delta s \cos \Delta\varphi \tag{10.9}$$

$$y_{t_i} = y_{t_{i-1}} + \Delta s \sin \Delta\varphi. \tag{10.10}$$

10.2.2 Genauigkeit der Odometrie

Das Odometrie-Verfahren ist ein Näherungsverfahren: je kleiner der Zeitabstand Δt der Berechnungen, desto genauer wird die Positionsbestimmung; das Verfahren sieht die Bewegung des Roboters innerhalb des zurückgelegten Zeitabschnitts als geradlinig an. Die Bewegung wird durch infinitesimal kleine Zeitabschnitte - eine entsprechende Auflösung der Sensoren vorausgesetzt - ideal angenähert. Der Fehler der Näherung ist bei Bewegungen von wenigen Millimetern zwischen den Berechnungen klein und daher akzeptabel.
Der entworfene Roboter verfügt über eine Maximalgeschwindigkeit von ca. $200 \frac{mm}{s}$; der DSP ist in der Lage die Position einige tausendmal pro Sekunde zu aktualisieren, was für diese Maximalgeschwindigkeit mehr als ausreicht.
Die Ungenauigkeit der Ausrichtungserkennung liegt - bedingt durch die Auflösung des Encoders - bei $\pm 0,5$ Encoderticks; die Ungenauigkeit φ_ε von ca. $\pm 0,2°$ erscheint akzeptabel, allerdings kann sich der Fehler bei jeder Positionsberechnung summieren.
Angenommen der Roboter würde ausgehend vom Punkt $P1$ entlang der Strecke s den Punkt $P2$ anfahren wollen, so bewirkt ein initialer Fehler φ_ε bei geradliniger, fehlerfreier Fahrt ein Erreichen des Punktes $P2'$ mit einer Abweichung d von $P2$. d entspricht bei kleinem Fehler ($\varphi_\varepsilon << 90°$) näherungsweise der Länge eines Kreisbogens mit dem Winkel φ_ε und dem Radius s (siehe Abbildung 10.2).
Für die Abweichung d gilt:

$$d = \frac{2\pi s \varphi_\varepsilon}{360}. \tag{10.11}$$

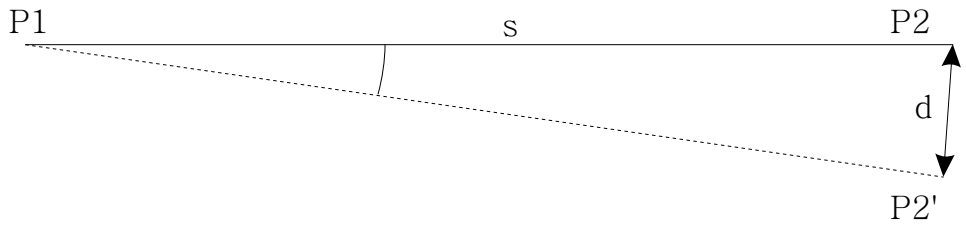


Abbildung 10.2: Auswirkung eines Ausrichtungsmessfehlers

Bei einer Fahrt über das MiroSot-Spielfeld (160 cm) beträgt d bei einem initialen Fehler von 0.2° beispielsweise ca. $55mm$. Es lässt sich an dieser Stelle erkennen, dass sich die Odometrie für die Navigation im lokalen Bereich des Roboters innerhalb einer gewissen Zeit mit akzeptabel kleinem Fehler eignet. Für eine dauerhafte exakte Positionsbestimmung ist die Odometrie mit Hilfe der Radencoder-Sensoren aufgrund ihrer approximierenden Natur ungeeignet. Selbst eine Vervielfachung der Encoderauflösung bringt über die Dauer eines Spiels gesehen keine ausreichende Verbesserung, da unter anderem weitere Fehlerquellen teilweise zu wesentlich größeren Abweichungen führen.
Die Fehler können sowohl (system)interne als auch externe Ursachen haben. Interne Fehler lassen sich korrigieren, externe dagegen nur bedingt.

Systeminterne Fehlerquellen

Systeminterne Fehlerquellen sind auf Maßabweichungen der Mechanik des Roboters zurückzuführen und ergeben, während der Bewegung systematische Abweichungen von der errechneten Position. Falls diese Abweichungen bekannt oder messbar sind, können sie in zukünftige Odometrieberechnungen einfließen und damit korrigiert werden.

- Der Radabstand hat wesentlichen Einfluss auf den Wendekreis des Roboters und damit auch auf die Berechnung der Ausrichtung. Eine Abweichung zwischen dem in der Rechnung verwendeten Maß und der Realität führt zu fehlerhaften Ergebnissen bezüglich der Ausrichtung des Roboters. Falls die Räder mit breiten Reifen bestückt sind - was beim vorliegenden Roboter aufgrund der besseren Bodenhaftung der Fall ist - kann der Auflagepunkt der Räder je nach Rad- sowie Belagbeschaffenheit und der Art der Drehung variieren und damit den Radabstand beeinflussen.

- Eine Abweichung des Radumfangs bei einem Rad bewirkt sowohl Fehler bei der Berechnung der Ausrichtung als auch bei der Berechnung der zurückgelegten Strecke.

- Ein unrundes Rad bewirkt eine stetige Drehung des Roboters, obwohl die Sensordaten eine korrekte geradlinige Bewegung suggerieren.

Systemexterne Fehlerquellen

Neben roboterbedingten, systeminternen Fehlern existieren noch eine Reihe weiterer Fehler, die während der Bewegung des Roboters auftreten können:

- Wenn Bodenunebenheiten von nur einem Rad erfasst werden, bewirken diese einen Fehler in der Ausrichtung des Roboters, da ein Rad einen durch die Unebenheit bedingten längeren Weg zurücklegt.
- Ein Verlust der Bodenhaftung durch glatten Untergrund, Staub oder zu starke Beschleunigung bzw. zu starkes Abbremsen bewirkt das Durchdrehen der Räder, so dass eine starke Abweichung der Radencoder-Sensordaten von der tatsächlichen Bewegung des Roboters erfolgt.
- Kollisionen mit der Bande oder gegnerischen Robotern können ebenfalls den teilweisen Verlust der Bodenhaftung verursachen und die berechnete Position verfälschen.

10.2.3 Messung der Genauigkeit

Die Genauigkeit der Odometrie bezüglich der mechanischen Eigenschaften des Roboters soll im Folgenden durch ein von Borenstein und Fend [JB96]$_{S.134}$ vorgestelltes Verfahren bestimmt werden: Das *bidirectional square-path Experiment*, das auch als *UMBmark* (University of Michigan Benchmark) bekannt ist, besteht aus einem quadratische Pfad, den der Roboter in beide Richtungen abfahren muss (siehe Abbildung 10.3). Die Abweichungen von der Zielposition werden gemessen und für mehrere Fahrten in beide Richtungen separat notiert. Für den UMBmark-Test ist ein Quadrat mit einer Seitenlänge von 4 Metern vorgesehen. Aufgrund der geringen Robotergröße und des kleinen Spielfelds der MiroSot-Liga wird die Seitenlänge für die vorzunehmende Messung auf einen Meter reduziert.
Der UMBmark-Test eignet sich zur Festellung der Odometrie-Genauigkeit in Verbindung mit der verwendeten Mechanik und zur Aufdeckung systeminterner Odometrie-Fehler. Würde der Roboter den Pfad nur in einer Richtungen abfahren, könnten sich verschiedene systeminterne Fehler gegenseitig eliminieren; durch eine Fahrt in beide Richtungen wird eine gegenseitige Auslöschung der Fehler verhindert. Abbildung 10.4 zeigt die Ergebnisse des UMBmark-Tests für einen der gebauten Fußballroboter. Die Abweichung der Fahrt gegen den Uhrzeigersinn sind mit ca. 4 cm relativ gering, bei der Fahrt im Uhrzeigersinn betragen sie hingegen bis zu 14 cm. Die Gruppierung

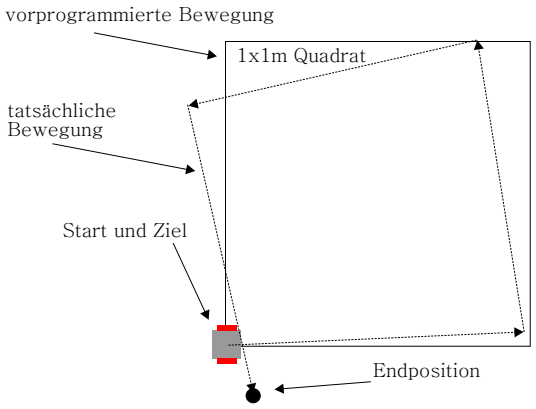


Abbildung 10.3: UMBmark-Test (gegen den Uhrzeigersinn)

der Messwerte deutet auf die Existenz systeminterner Odometrie-Fehler hin, die ihre Ursache in Abweichungen der zugrundegelegten mechanischen Maße haben. Die Genauigkeit der Odometrie könnte durch eine weitergehende Messung und eine Korrektur der systeminternen Fehler wesentlich verbessert werden.

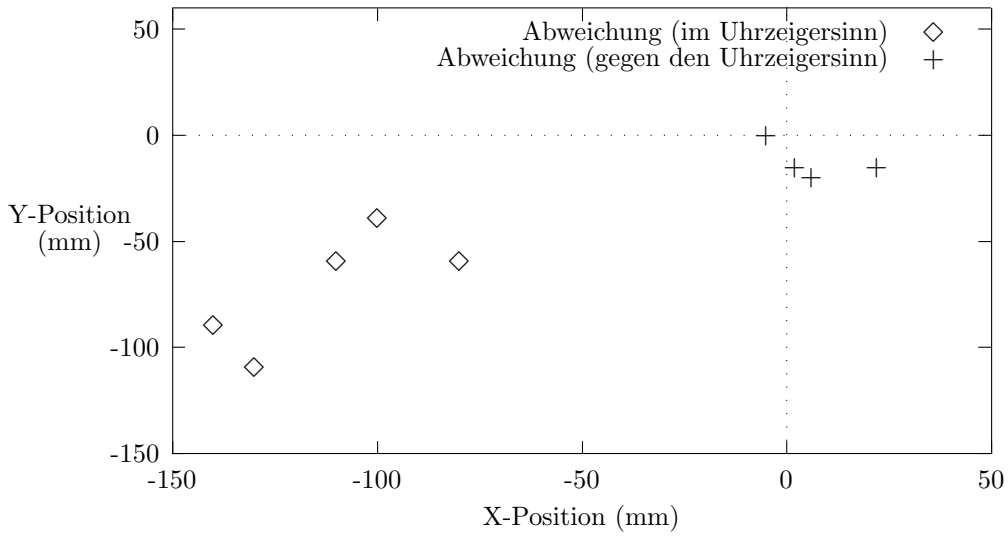


Abbildung 10.4: UMBmark des Fußballroboters

10.2.4 Verbesserungmöglichkeiten

Systeminterne Odometriefehler lassen sich durch verschiedene Tests messen und bis zu einem gewissen Grad zufriedenstellend korrigieren (Borenstein

[JB96]$_{S.137}$), systemexterne Fehler lassen sich durch zusätzlichen mechanischen und sensorischen Aufwand verringern; zum Beispiel wirkt sich die Trennung des Antriebs von der Sensorik positiv aus, da ein Traktionsverlust der Antriebsräder keine Fehler in der Odometrie verursacht. Eine derartige Trennung kann durch das Hinzufügen eines dedizierten Encoderrades erreicht werden. Derartige Verfahren sind allerdings bei den vorgegebenen Roboterabmessungen nicht realisierbar. Eine Verbesserung kann beim entworfenen Roboter allerdings durch die Einbeziehung weiterer Sensordaten erreicht werden. Die Art der Sensordaten sollte sich zu diesem Zweck von den bisherigen - für die Odometrie herangezogenen - Daten unterscheiden: da die Daten der Radencoder-Sensoren relativ zur jeweiligen Radbewegung sind, summieren sich vorhandene Fehler. Um diesen Fehler zu beseitigen, ist eine gelegentliche Korrektur durch absolute Positionsdaten sinnvoll.

Positionsdaten des Hostrechners

Die durch die globale Bildverarbeitung ermittelten Positionsdaten eignen sich nicht für die direkte Robotersteuerung (siehe Abschnitt 4.1.2), stellen allerdings eine wichtige und notwendige Möglichkeit zur Korrektur von Odometriefehlern bei teilautonomen Robotern dar. Die globale Bildverarbeitung liefert aufgrund der externen Positionierung der Kamera absolute Daten bzgl. der Roboterpositionen und deren Ausrichtung. Der dabei existierende Fehler ist begrenzt und summiert sich nicht in späteren Berechnungen. Eine Korrektur des Odometriefehlers ist nur in gewissen Zeitabständen notwendig und nicht, wie der Regelkreis der Cavalry-Roboter, an feste Zeitvorgaben gebunden.
Um eine sinnvolle Korrektur der Odometriedaten durch den Hostrechner zu ermöglichen, ist eine Zeitsynchronisierung des Roboters mit dem Hostrechner notwendig. Die Positionsdaten des Hostrechners sind aufgrund der Bildverarbeitung und der Funkübertragung mit einer Latenz behaftet, die vom Roboter kompensiert werden muss, bevor die externen Daten in die Korrektur der eigenen Odometriedaten einbezogen werden können. Falls der Hostrechner die Zeit der Datenerhebung zusammen mit den Positions- und Ausrichtungsdaten sendet, kann der Roboter aufgrund seiner Bewegungshistorie eine Fehlerkorrektur der aktuellen Odometriedaten durchführen.
Die vorgeschlagene Korrektur durch externe Positionsdaten bewirkt eine Einschränkung der Autonomie des Roboters. Diese ist durch die Konzeptionierung eines teilautonomen Roboters ausdrücklich erlaubt - trotzdem soll das Ziel der Roboterentwicklung verfolgt werden, eine möglichst weitgehende Autonomie des Roboters von externen Datenquellen zu erzielen. Es wird aus diesem Grund geprüft, ob alternative Sensorsysteme die Positionsberechnung unterstützen bzw. verbessern können.

10.3 Inertiale Navigation

In der zweiten Platinenversion wurde der in Abschnitt 6.5.2 beschriebene Beschleunigungs-Sensor hinzugefügt, um eine zweite unabhängige Methode zur Positionsbestimmung zu evaluieren. Mit Hilfe des Sensors kann ein inertiales Navigationssystem für die zweidimensionale Bewegung des Roboters erstellt werden.

10.3.1 Aufbau

Der Sensor wird diagonal versetzt zum Drehmittelpunkt M des Roboters auf der Hauptplatine angeordnet (siehe Abbildung 6.2) und ist in der Lage, alle vom Roboter ausführbaren Bewegungen nachzuvollziehen. Eine geradlinige Bewegung nach vorne oder hinten erzeugt Beschleunigungs-Messwerte in x-Richtung, bei Drehung des Roboters wirkt die Zentripetalkraft in y-Richtung. Eine Integration der Sensordaten der x-Achse (angefangen aus der Ruhelage) ergibt die aktuelle Fahrgeschwindigkeit, eine Integration der Fahrgeschwindigkeit liefert die zurückgelegte Strecke. Es gilt die Beziehung:

$$v_{Roboter} = \int_0^t a_x(t) - a_{Null} \tag{10.12}$$

$v_{Roboter}$ = Robotergeschwindigkeit
a_x = Beschleunigungs-Messwert in x-Richtung
a_{Null} = Null-Wert der Beschleunigung (durch Kalibrierung).

Für die Zentripetalbeschleunigung a_{zp} gilt:

$$a_{zp} = \frac{v^2_{Roboter}}{r} \tag{10.13}$$

r = Radius der Kreisbahn.

Da die Zentripetalkraft der in y-Richtung gemessenen Kraft entspricht, gilt für den Radius der Kreisbahn die Beziehung:

$$r = \frac{v^2_{Roboter}}{a_y(t)} \tag{10.14}$$

a_y = Beschleunigungs-Messwert in y-Richtung.

10.3.2 Genauigkeit und Fehlerquellen

Die Genauigkeit eines inertialen Navigationssystems muss leider als relativ gering angesehen werden. Borenstein [JB96]$_{S.145}$ gibt den typischen Drift-

wert von qualitativ hochwertigen inertialen Navigationssystemen, die in kommerziellen Passagierflugzeugen eingesetzt werden, mit 1850 Metern pro Stunde an. Im Fall des entworfenen Roboters fällt das Ergebnis eher schlechter aus, da der eingesetzte Beschleunigungs-Sensor ungenauer arbeitet als qualitativ hochwertige inertiale Navigationssysteme.
Die Ungenauigkeiten begründen sich durch die Drift der berechneten Geschwindigkeit, der Position sowie des Kreisbahnradiuses. Solche Abweichungen entstehen durch geringste Fehler bei der Integration der Messwerte und können durch eine kleine Ungenauigkeit des kalibrierten Null-Wertes verursacht werden. Eine zusätzliche Fehlerquelle existiert bei der vorgestellten Anordnung des Beschleunigungs-Sensors in den Messwerten der y-Achse des Sensors: die Berechnung geht von einer ausschließlichen Wirkung von Zentripetalkräften in y-Richtung aus. Ausnahmen in dieser Hinsicht treten auf, wenn der Roboter die Bodenhaftung verliert. In diesem Fall gibt der y-Wert des Sensors nicht die Zentripetalkraft an, sondern unter Umständen die tatsächliche Beschleunigung des Roboters in y-Richtung. Ursache für diesen Fehler kann zum Beispiel eine seitliche Kollision mit einem anderen Roboter sein.
Zusätzlich wirkt die Zentripetalkraft - bedingt durch die diagonale Anordnung des Sensors - bei Drehungen mit kleinem Radius auch komponentenweise auf die x-Achse, da die y-Achse des Sensors nur annähernd durch den Drehmittelpunkt verläuft. Eine seitlich zum Mittelpunkt des Roboters versetzte Anordnung kann diesen Fehler beseitigen; dort ist allerdings in der momentanen Platinenversion - bedingt durch die Anordnung des Funksystems - kein Platz verfügbar.

10.3.3 Driftmessung

Durch die gesammelte Erfahrung deutete sich an, dass die inertiale Navigation ohne Korrektur der Drift für die Positionsbestimmung der Roboters ungeeignet ist. Aus diesem Grund wurde eine Messung der Drift von Position und Geschwindigkeit des stillstehenden Roboters ausschließlich unter Verwendung der x-Achse des Sensors durchgeführt (Abbildung 10.5). Es ist eine Drift der Roboterposition von ca. 30 cm innerhalb von 12 Sekunden zu beobachten. Dieser Wert wächst mit zunehmender Dauer - bedingt durch die Drift der Geschwindigkeit - schneller und ist bereits nach einigen Sekunden wesentlich zu hoch, um eine akzeptable Positionsberechnung ausschließlich mit Hilfe der inertialen Navigation durchzuführen. Bei fahrendem Roboter schwanken die Messwerte auf Grund von Vibrationen wesentlich stärker (vgl. Abschnitt 11.4), so dass ein weiter verschlechtertes Driftverhalten resultiert.

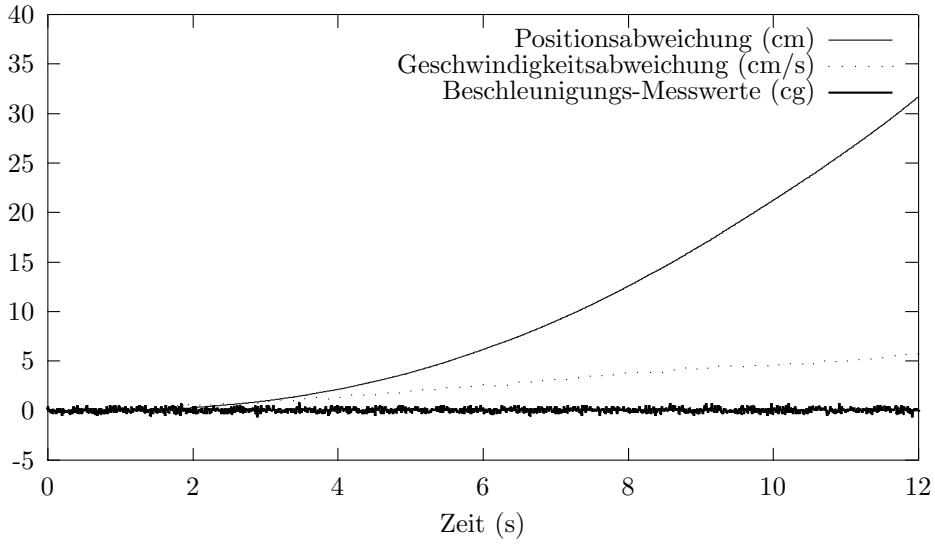


Abbildung 10.5: Driftmessung der inertialen Navigation

10.4 Diskussion der Ergebnisse

Ein Einsatz des inertialen Navigationssystems als unabhängiges Verfahren zur Positionsbestimmung ist mit der momentanen Sensoranordnung nicht möglich. Die mehrfache Integration der Messwerte vergrößert anfänglich vorhandene Fehler schnell. Der Beschleunigungs-Sensor kann allerdings zur anderweitigen sinnvollen Ergänzung des Robotersystems genutzt werden, was im weiteren Verlauf der Arbeit in Kapitel 11 beschrieben wird.
Die schrittweise Berechnung der Roboterposition ausschließlich mit Hilfe der Odometrie ist im Rahmen einer Bewegung von 1-2 Metern ohne große Abweichungen möglich: das Fahren von vorgegebenen Trajektorien oder das Anfahren eines Punktes ist mit diesem Verfahren bei gelegentlicher äußerer Korrektur realisierbar. Der zugehörige Regelkreis kann somit vollständig auf den Roboter verlagert werden, so dass eine weitgehende Autonomie - ein Ziel der Roboterneuentwicklung - erreicht wird.
Die Korrektur von systeminternen Fehlern kann mit Hilfe des UMBmark-Tests für jeden Roboter durchgeführt werden. Eine Korrektur von systemexternen Fehlern kann durch Einbeziehung absoluter Positionsdaten der globalen Bildverarbeitung durchgeführt werden. Trotzdem kann der Roboter innerhalb eines Zeitraumes von mehreren Sekunden ohne externe Korrekturen autonom navigieren.

10.5 Zukünftige Arbeiten

Das Ziel, den Roboter bezogen auf seine Positionsbestimmung so autonom wie möglich zu gestalten, kann durch die Integration weiterer Sensoren verfolgt werden. Absolute Sensoren sind in disem Fall vorzuziehen, da sich Messfehler nicht mit der Zeit summieren.

Kreisel-Sensor

Eine Möglichkeit, Ausrichtungsänderungen exakter zu messen, besteht im Einsatz eines Kreisel-Sensors (Gyroskop) (siehe Abschnitt 5.5.3), der die Ausrichtung des Roboters bezogen auf sein Kreiselsystem misst. Der Sensor wäre allerdings idealerweise im Drehmittelpunkt des Roboters anzubringen: an dieser Position ist aufgrund der Größenbeschränkungen der MiroSot-Liga und der Anordnung des größten Bauteils, des DSPs, kein ausreichender Platz verfügbar. Grundsätzlich sind Gyroskope von einer Drift betroffen [JB96]$_{S.146}$, so dass sich auch bei diesem Sensor (Drift-)Fehler addieren. Eine sinnvolle Alternative zu einem Kreisel-Sensor stellt ein Kompass-Sensor dar.

Kompass-Sensor

Der in Abschnitt 5.5.4 vorgestellte Kompass-Sensor kann die Ausrichtungserkennung des Roboters durch absolute Daten unterstützen. Der Roboter ist somit in der Lage, seine Ausrichtung ohne externe Korrekturen über einen beliebig langen Zeitraum mit einem absoluten Fehler von unter 1° zu messen: dies ist ein wesentlicher Vorteil gegenüber Kreisel-Sensoren. Mögliche Kompass-Sensoren sind allerdings störanfällig, so dass exakte Messergebnisse mit Fehlern von unter einem Grad nur mit einer entsprechenden Entstörung der Bauteile zu erwarten ist. Erfahrungen mit dem Beschleunigungs-Sensor bestätigten diese Vermutung.
Eine Entstörung des Beschleunigungs-Sensors könnte ebenfalls zu einer signifikanten Verminderung des Rauschens führen und den Sensor zukünftig eventuell für die Unterstützung der Positionsberechnung einsetzbar machen.

Kamera-Sensoren

Die in Kapitel 13 konzeptionell beschriebene Integration von digitalen Farbkamera-Sensoren ermöglicht eine Landmarken- und Objekterkennung. In Verbindung mit inter-Roboter-Kommunikation können absolute Roboterpositionen und Ausrichtungen ermittelt werden. Der Roboter kann mit Hilfe dieses Verfahrens vollkommen autonom vom externen Hostsystem navigieren.

Kapitel 11

Ereignisdetektion

Die Detektionsmöglichkeit der im Umfeld des Roboters auftretenden Ereignisse wurde in Abschnitt 4.1.3 als optionales Ziel der Roboterentwicklung erarbeitet. Es muss für die grundlegenden Fähigkeiten des Roboters nicht notwendigerweise realisiert werden, kann allerdings sinnvolle zusätzliche Informationen über Ereignisse im direkten Umfeld des Roboters liefern.
Im folgenden Kapitel wird die grundlegende Eignung des Beschleunigungs-Sensors zur Detektion bestimmter Ereignisse gezeigt, bei denen Erschütterungen des Roboters auftreten. Ein Verfahren zur Detektion wird erarbeitet, entsprechende Ereignisse werden simuliert und die Messung der Sensordaten durchgeführt. Anhand der Messergebnisse lässt sich eine Eignung des Sensors für den genannten Zweck erkennen.

11.1 Mögliche Ereignistypen

Im Folgenden werden mögliche Ereignistypen im Umfeld des Roboters beschrieben, die mit Hilfe des Beschleunigungs-Sensors detektiert werden könnten.

11.1.1 Objektkollision

Die Kollision des Roboters mit einem anderen Objekt erzeugt zwangsläufig Erschütterungen und somit Beschleunigungen des Sensors. Verschiedene Kollisionen mit dem Ball, der Spielfeldbegrenzung und gegnerischen oder eigenen Robotern sind möglich.
In einem ersten Schritt ist zu untersuchen, ob diese Ereignisse detektierbar sind. Weitere Informationen könnten durch eine Klassifizierung des Objekttyps und durch die Bestimmung der Stoßrichtung gewonnen werden.

11.1.2 Traktionsverlust

Überschreiten die an den Rädern wirkenden Antriebs- bzw. Bremskräfte die Haftung zwischen den Reifen und dem Spielfeldbelag, entsteht Schlupf (Durchdrehen der Räder). Schlupf bewirkt den Verlust der Traktion (Zugkraft der Räder). Die resultierende ruckartige Bewegung der Räder dürfte sich von den üblichen Beschleunigungen des Roboters unterscheiden und müsste aufgrund der entstehenden Vibrationen messbar sein.

11.2 Vorteile der Detektion

Falls es mit Hilfe des Beschleunigungs-Sensors möglich ist, derartige Ereignisse zu detektieren und eventuell zu klassifizieren, ist dies von übergeordnetem Nutzen für die Roboterfußballmannschaft:

- Eine Kollision mit dem Spielball kann die Ausführung eines Schussversuchs bestätigen und eventuell weitere Informationen über die resultierende Geschwindigkeit des Balles liefern. Diese Informationen können an andere Roboter übermittelt werden, die ihr Verhalten kooperativ an die veränderte Spielsituation anpassen. Unabhängig davon ist die Detektion dieses Ereignisses eine wichtige Informationsquelle für jeden einzelnen Roboter, der den Ball führen soll.

- Eine Kollision mit einem Roboter - unabhängig davon, ob es ein Roboter des gegnerischen oder des eigenen Teams ist - stellt die interne Positions- und Ausrichtungsberechnung durch die Odometrie in Frage: durch den Zusammenprall kann die Position des Roboters nachdrücklich beeinflußt worden sein und ein Verlust der Haftreibung der Räder ist anzunehmen. Die aktuelle Bewegung des Roboters kann erst nach einer Korrektur von Position und Ausrichtung durch Daten anderer Sensoren fortgesetzt werden.

- Eine Kollision mit der Spielfeldbegrenzung sollte bei korrekter Navigation nicht auftreten. Sie liefert aber, falls sie dennoch auftritt, zumindest Informationen über eine Komponente der aktuellen Roboterposition, je nach dem, welche Begrenzung vom Roboter getroffen wurde. Eine Korrektur der Positionsdaten ist auch in diesem Fall erforderlich, um weitere Bewegungen auszuführen.

- Ein Traktionsverlust tritt bei zu geringer Haftreibung zwischen Radbelag und Spielfeld auf. Da die Beschaffenheit der Spielfelder und der Reifen durchaus variieren kann, ist eine statische Festlegung der maximalen Beschleunigung des Roboters nicht sinnvoll. Eine Detektion

des Traktionsverlusts kann zur Maximierung der kontrollierten Beschleunigung des Roboters bei unbekannten Umgebungsbedingungen verwendet werden.

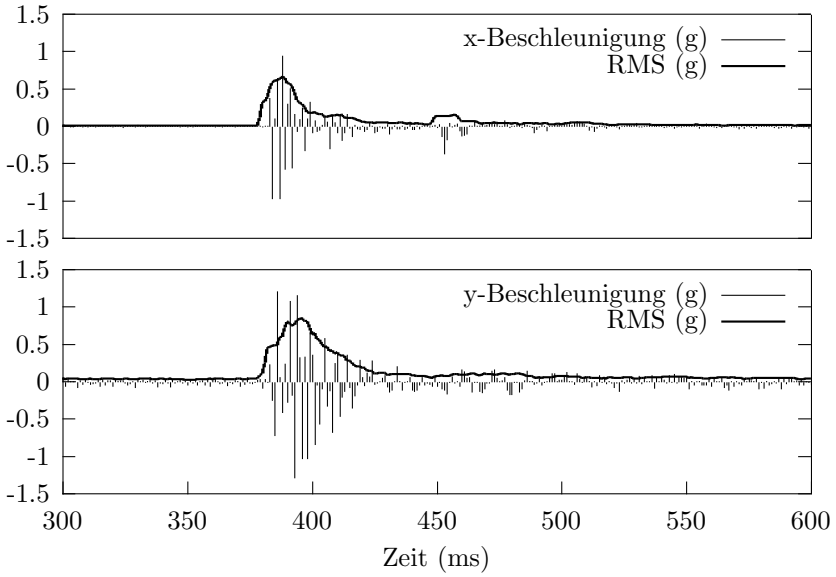


Abbildung 11.1: Kollision mit dem Spielball

11.3 Entwicklung eines Detektionsverfahrens

Der Beschleunigungs-Sensor ist fest auf der Platine des DSP-Systems angebracht; die Platine ist wiederum fest mit dem Gehäuse des Roboters verbunden. Erschütterungen des Roboters wirken ungedämpft auf den Sensor, der daher ähnlich einem Seismographen reagiert. Es entstehen hochfrequente[1] Schwingungen mit Amplituden, die bereits bei einem schwachen Stoß bis zu $\pm 1,5$ g betragen können. Abbildung 11.1 zeigt die Auswirkung der Kollision eines Spielballs mit dem Roboter. Die resultierende Schwingung dauert ca. 50 ms und erreicht Beschleunigungswerte von $\pm 1,3$ g (siehe Abbildung 11.1).
Grundlage eines Detektionsverfahrens kann die Unterscheidung von systemintern bedingten und systemextern bedingten Beschleunigungen sein. Die gewünschten Beschleunigungen während des Fahrbetriebs werden dabei als systemintern und die aller beschriebenen Ereignisse als systemextern bezeichnet. Die im Normalfall auf den Roboter wirkenden Beschleunigungswerte lassen sich anhand der in Kapitel 10 beschriebenen Kausalität zwischen

[1] Hochfrequent wird an dieser Stellt in Bezug zu den normalen Beschleunigungsschwankungen des Roboters verwendet und bezeichnet Frequenzen oberhalb von ca. 200 Hz.

der Roboterbewegung und den Sensormesswerten einfach herleiten. Werden die berechneten Beschleunigungswerte von den Messwerten des Sensors subtrahiert, sollte - die korrekte Ausführung der vom Roboter geplanten Bewegung vorausgesetzt - nur das Sensorrauschen um den Nullwert der Kalibrierung sichtbar sein. Ereignisse, die Erschütterungen verursachen, bewirken einen plötzlichen starken Anstieg der Amplitude. Zusätzlich kann es zu dauerhaften Abweichungen der Messwerte vom Nullwert kommen, falls der Roboter anders als geplant beschleunigt und damit von seiner Sollposition abweicht.

11.4 Exemplarische Messungen

Die Detektierbarkeit der oben genannten Ereignisse wird im Folgenden untersucht, indem Messdaten während absichtlich herbeigeführter Ereignisse aufgezeichnet werden. Zu diesem Zweck werden die Beschleunigungen in x- und y-Richtung über einen Zeitraum von 2,5 Sekunden mit einer Frequenz von einem kHz gemessen. Die Abbildungen 11.1 bis 11.4 zeigen exemplarisch Ausschnitte der resultierenden Daten.

Die Messergebnisse weisen einige spezifische Eigenschaften auf, die für

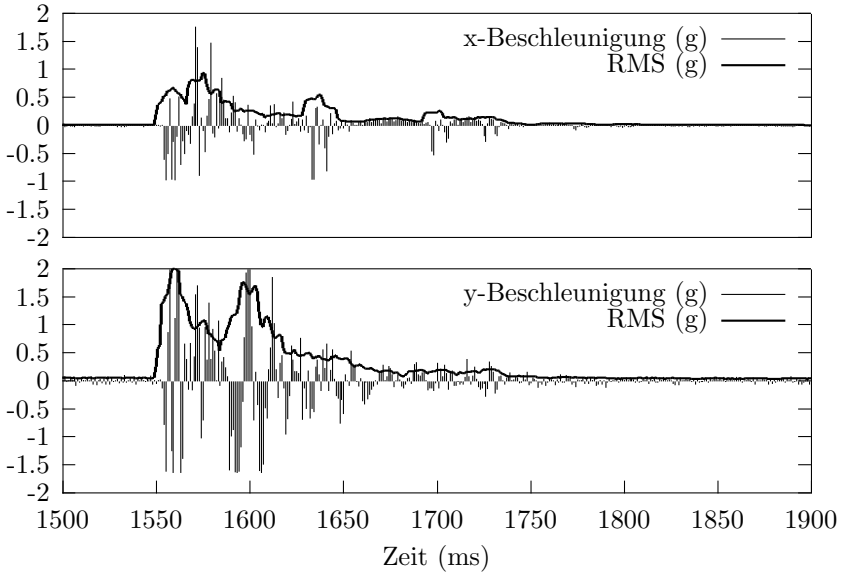


Abbildung 11.2: Kollision mit einem Roboter

eine Detektion und eine eventuelle anschließende Klassifizierung verwendet werden können:

- Die Erschütterungen des Sensors sind bei stehendem Roboter sehr gering (Rauschen), werden aber mit zunehmender Geschwindigkeit

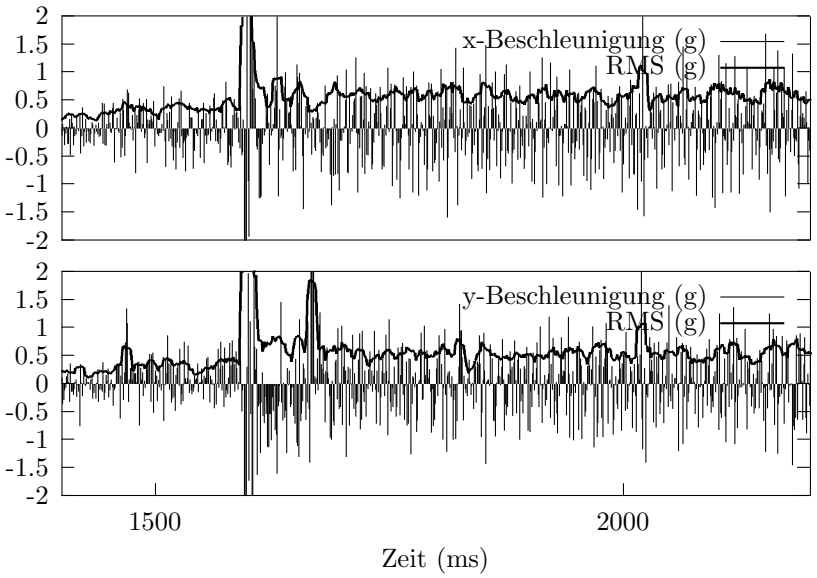


Abbildung 11.3: Kollision mit der Bande

stärker. Dies bewirkt gute Detektionsmöglichkeiten von Ereignissen im Stand oder bei langsamer Geschwindigkeit; schnellere Bewegungen hingegen erschweren die Detektion schwächerer Kollisionen.

- Die Dauer der verschiedenen Ereignissarten ist unterschiedlich: eine Kollision mit dem Spielball ist für ca. $40 - 50$ ms zu detektieren, die Kollision mit einem Roboter ca. 150 ms.
  Bei der Kollision mit der Bande treten kurzzeitig starke Vibrationen auf, deren Maxima bei den durchgeführten Messungen oberhalb des Messbereichs des Sensors von $2g$ lagen.
  Ein Traktionsverlust erzeugt (Abbildung 11.4) zu Beginn der übermäßigen Radbeschleunigung ebenfalls sehr starke Vibrationen, die auch im weiteren Verlauf nicht signifikant schwächer werden.

- Der *root mean square*-Wert (vgl. Abschnitt 26) der Messwerte (vgl. Abbildungen 11.1 bis 11.4) steigt bei Eintreffen eines Ereignisses signifikant an. Je nach Dauer des Ereignisses verbleibt der Fehler einige Zeit auf höherem Niveau und sinkt nach Beendigung des Ereignisses ab. Ein sich stark positiv verändernder RMS-Wert zeigt - aufgrund Korrektur der Messwerte um die erwartete Roboterbeschleunigung - eine starke Zunahme der Vibrationen an. Der RMS-Werte ist aus diesen Gründen eine geeignete Kenngröße zur Detektion von Ereignissen.

Während der durchgeführten Messungen stellte sich heraus, dass die Mechanik des Roboters in bestimmten Fällen systeminterne Vibrationen erzeugt,

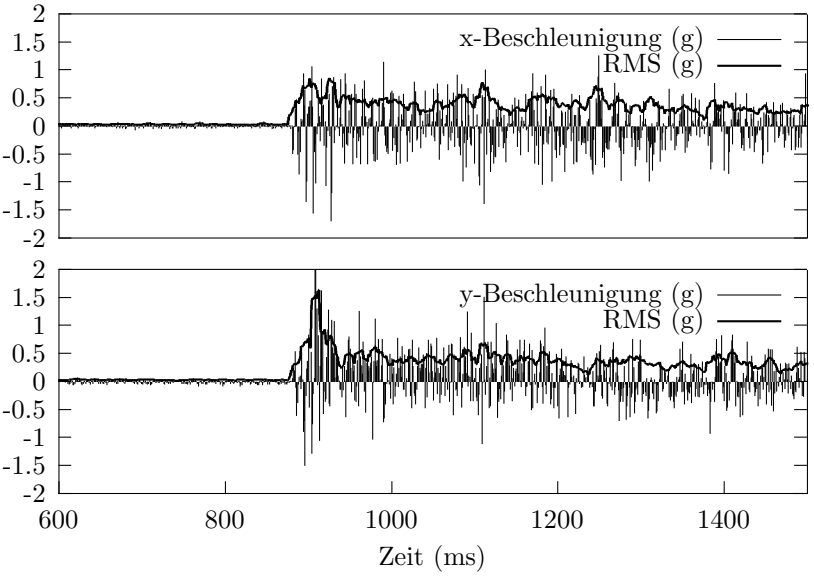


Abbildung 11.4: Traktionsverkust beim Anfahren

die bezüglich ihrer Amplitude mit den Vibrationen externer Ereignisse vergleichbar sind: durch die zweirädrige Bauweise des Roboters existieren vorne und hinten am Gehäuse kleine Schleifer, die einen direkten Bodenkontakt des Gehäuses verhindern. Bei einem Wechsel der Fahrtrichtung kippt der Roboter aufgrund der Beschleunigung von einem Schleifer auf den anderen. Die entstehenden Vibrationen sind stark genug, um als Ereignis detektiert zu werden. Je höher der Abstand der Schleifer vom Boden ist, desto stärker fällt die Vibration beim Kippen aus. Zukünftige Untersuchungen müssen zeigen, ob sich die Mechanik in dieser Hinsicht optimieren lässt.
Die Implementierung des vorgestellten Detektionsverfahrens findet sich in der Quelldatei *detect.c*.

11.5 Ergebnis und zukünftige Arbeiten

Die Detektion von Ereignissen, die Vibrationen des Roboters verursachen, kann mit Hilfe des Beschleunigungs-Sensors unter Verwendung des vorgestellten Verfahren erreicht werden.
Weitere Informationen könnten aus einer Klassifizierung der Ereignisse gewonnen werden. Eine weitere Untersuchung der Messergebnisse auf verwendbare Merkmale und die Implementierung einer Klassifizierung können Anregungen für zukünftige Arbeiten bieten.

Kapitel 12

Kommunikation

In diesem Kapitel wird mit Hilfe der erarbeiteten Ziele (Abschnitt 4.2, 5.6 und 6.4) und der resultierenden Hardware-Realisierung eine Kommunikationslösung entwickelt, implementiert und das Ergebnis evaluiert.

12.1 Voraussetzungen

Die eingesetzten BiM-Transceiver der Firma Radiometrix besitzen einige spezifische Eigenschaften, die für den erfolgreichen Entwurf einer Kommunikationslösung beachtet werden müssen:

- Die Module sind zwischen Sende- und Empfangsbetrieb durch den Roboter/Hostrechner umschaltbar. Eine Umschaltung vom Sende- in den Empfangsmodus benötigt eine Zeitspanne von typischerweise 3 ms, bevor die Datenausgabe fehlerfrei funktioniert.

- Zu Beginn des Sendebetriebsmodus muss laut Herstellerangabe [Rad01a] über einen Zeitraum von mindestens 3 ms eine Präambel - bestehend aus mehreren bit-symmetrischen Bytes (z.B. $0x55 = 0b01010101$) - gesendet werden, um die Datenausgabe der Empfangsmodule zu initialisieren. Der Grund dieser Notwendigkeit begründet sich in der vorangegangenen Sendepause oder Sende-/Empfangsumschaltung, die eine längere unsymmetrische-Bitfolge verursacht. Die Datennachführung (Data-Slicer) arbeitet erst nach einer Präambel fehlerfrei[1]. Anschließend müssen ein oder zwei $0xFF$ Bytes gesendet werden, um das Timing der RS232-Empfänger zu initialisieren (Abbildung 12.1).

- Die Module liegen als 433 MHz und als 418 MHz[2] Version vor. Sie verhalten sich beinahe identisch - die Anzahl der fehlerhaft übertragenen

[1] Messungen zeigten allerdings, daß eine Präambeldauer von 2 ms ausreichend ist.
[2] Je nach Einsatzort variieren die national freigegebenen Funkfrequenzen.

Präambelbytes ist allerdings unterschiedlich. Dies ist bei korrekter Implementierung nicht relevant, allerdings führte es zu scheinbarer Fehlfunktion der 418 MHz Module in der PG362, da die Präambel in die Checksumme zur Prüfung der Datenintegrität einbezogen wurden und die Entwicklung ausschließlich mit den 433 MHz Modulen durchgeführt wurde. So führte eine differierende Zahl von fehlerfrei erkannten Bytes der Präambel bei den 418 MHz bzw. den neueren 433MHz Modulen zu dauerhaft auftretenden Datenfehlern. Dies kann einfach vermieden werden, in dem nur solche Bytes in die Prüfsumme einbezogen werden, die der Präambel und den beiden $0xFF$ Bytes folgen.

- Während der Datenübertragung muss die Anzahl der übertragenen 1- und 0-Bits über einen Zeitraum von 4 ms identisch sein, da sonst die Datennachführung (Data-Slicer) am digitalen Ausgang des Empfängers nicht mehr fehlerfrei arbeitet.

- Die Module liefern ein Carrier-Detect Signal, falls das Modul im Empfangsmodus betrieben wird und ein Trägersignal auf der Empfangsfrequenz erkannt wird. Dieses Trägersignal wird von einem Modul im Sendemodus ausgesandt, kann allerdings auch von Sendern auf gleicher Frequenz in der näheren Umgebung verursacht werden, was dann zumeist häufige Datenfehler zur Folge hat beziehungsweise im schlimmsten Fall eine Kommunikation auf dieser Frequenz gänzlich unmöglich macht. Da das 433MHz Band in Deutschland frei benutzbar ist, kommunizieren diverse kabellose Geräte aus dem Bereich der Haustechnik auf dieser Frequenz[3].

- Sowohl auf Hostseite als auch auf Roboterseite werden stabförmige Antennen verwendet, da diese die gute Sende-/Empfangsleistungen bieten. Die von Radiometrix in [Rad01b] angegeben Alternativen haben sich in Versuchen als unzureichend bzgl. der Empfangsqualität herausgestellt.

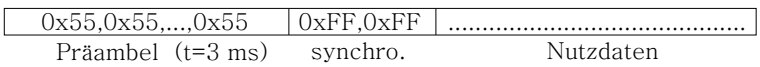


Abbildung 12.1: Aufbau der Funkdaten

[3]Beispiele für derartige Gerärte sind elektronische Thermometer mit Außentemperaturfühler oder Zeitgeber für automatische Rolladenantriebe

12.2 Entwurf eines geeigneten Protokolls

Die neu entwickelten Roboter sind hardwareseitig mit der Möglichkeit bidirektionaler Funkkommunikation ausgestattet. Die verwendeten Radiometrix-Transceiver ermöglichen 64kBit/s Bandbreite[4] auf einem Sende-/Empfangskanal. Daraus resultiert eine halbduplex-Kommunikation zwischen den Robotern bzw. zwischen Robotern und Host, für die ein adäquates Protokoll mit folgenden Anforderungen gefunden werden muss:

- ein sendendes System und ein oder mehrere empfangende Systeme.
- variable Länge der Daten, deren Inhalt beliebig sein sollte.
- Möglichkeit der Emfangsbestätigung von Daten (Acknowledge).
- Erkennung von Übertragungsfehlern und Kollisionen.
- Möglichkeit der Wiederholung von Datensatzteilen, die durch fehlerhafte Übertragung den/die Empfänger nicht erreicht haben.

Um größere Datenmengen zu übertragen, ist es sinnvoll, diese in mehrere kürzere Datensätze zu unterteilen. Die Wahrscheinlichkeit von Übertragungsfehlern sinkt bei kürzeren Datensätzen bezogen auf den einzelnen Datensatz[5]. Bezogen auf die gesamten zu übertragenden Daten erhöht sich allerdings die Wahrscheinlichkeit eines Fehlers, da pro Datenpaket mehrere zusätzliche Bytes für die Präambel, zur Adressierung, Nummerierung und Fehlerprüfung benötigt werden: die Gesamtzahl der zu übertragenden Bytes steigt und damit auch die Wahrscheinlichkeit eines Fehlers. Der Vorteil der Unterteilung liegt darin, dass im Fehlerfall nicht die komplette Datenmenge erneut übertragen werden muss, sondern nur der als fehlerhaft erkannte Datensatz.
Diese Unterteilung ist bei kleinen Datenmengen nicht notwendig und nicht alle Datensätze müssen unbedingt im Fehlerfall wiederholt werden. Es bietet sich eine Kategorisierung in mehrere Dienste (Services) an.
Tanenbaum [Tan96]$_{S.177}$ gibt drei denkbare Möglichkeiten an:

1. unacknowledged connectionless service,
2. acknowledged connectionless service und
3. acknowledged connection-oriented service.

[4] 64kBit/s Bandbreite sind nur mit Einschränkungen zu erreichen, oberhalb von 30kBit/s werden bitsymmetrische Daten vorausgesetzt.

[5] Es wird voraussetzt, dass die Wahrscheinlichkeit eines Übertragungsfehlers pro Bit konstant ist.

Der *unacknowledged connectionless* Dienst wird für Daten eingesetzt, deren erneute Übermittlung im Fehlerfall nicht sinnvoll ist. Dies kann bei Daten mit aktuellem Zeitbezug der Fall sein, zum Beispiel bei der aktuellen Ballposition. Ein Verlust der Daten durch fehlerhafte Übertragung ist natürlich unerwünscht, eine Wiederholung zu einem späteren Zeitpunkt allerdings ebenfalls, da sich die Ballposition inzwischen verändert haben kann und eventuell bereits aktuellere Daten gesendet wurden. Die Fehlerwahrscheinlichkeit steigt bei diesem Dienst mit der Länge des Datensatzes - das Verfahren eignet sich aus diesem Grund nur für kurze Datensätze.
Der *acknowledged connectionless* Dienst bietet sich für kurze Datensätze an, die mit Sicherheit beim Empfänger fehlerfrei ankommen müssen und deren Wiederholung im Fehlerfall gewünscht ist. Dies könnten zum Beispiel Nachrichten über globale Ereignisse wie den Spielstart oder einen Torerfolg sein. Es fallen im Gegensatz zum *connection-oriented* Dienst keine zusätzlichen Daten zum Auf- und Abbau einer Verbindung an. Die Fehlerwahrscheinlichkeit steigt bei diesem Dienst mit der Länge des Datensatzes - er eignet sich daher ebenfalls nur für kurze Datensätze.
Der *acknowledged connection-oriented* Dienst bietet sich für größere Datenmengen an, die mit Sicherheit fehlerfrei beim Empfänger ankommen müssen. Beispiele sind Sensordaten oder nachladbare Softwaremodule. Dieser Dienst wird sicherlich weniger während eines Spiels, als vielmehr während des Entwicklungs- und Testzeitraums oder zur Initialisierung des Roboters benötigt. Eine Unterteilung größerer Datenmengen in kleinere Datensätze, eine Fehlerprüfung sowie Bestätigungen des korrekten Empfangs von Datensätzen werden von diesem Dienst implementiert. Er sorgt zusätzlich beim Empfänger für die anschließende Zusammensetzung der Datensätze in korrekter Reihenfolge.
Um die einzelnen Aufgaben auf Seiten des Senders und der Empfänger übersichtlicher und die Implementierung strukturiert zu gestalten, werden die Aufgaben entsprechend dem in Abschnitt 2.6.1 vorgestellten OSI-Schichtenmodell verteilt. Im Folgenden wird die Anwendbarkeit und Relevanz der einzelnen Schichten für die vorliegende Aufgabenstellung diskutiert und die Implementierung der Kommunikationslösung beschrieben.

12.2.1 Physikalische Schicht

Die Funkschnittstelle nutzt zum Datenaustausch die serielle Schnittstelle, deren Signale vom Funkmodul in hochfrequente Funksignale im 433 bzw. 418 MHz-Band gewandelt werden. Die seriellen RS232-Schnittstellen sind sowohl auf Roboter- als auch auf Hostseite vorhanden; daher besteht die Implementierung dieser Schicht unter anderem in der Ansteuerung dieser Schnittstellen. Die Datenframes werden um Präambel, Start- und Stopp-

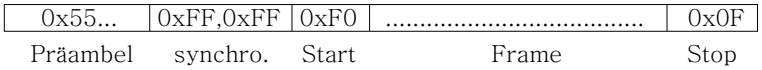


Abbildung 12.2: Zusammensetzung der seriellen Daten

byte (Abbildung 12.2) ergänzt und mit Hilfe der seriellen Schnittstelle transferiert. Die Umschaltung der Funkmodul-Betriebsmodi wird ebenfalls durch die physikalische Schicht ermöglicht. Der Algorithmus zum Senden bzw. Empfangen von Daten sorgt für die Einhaltung der Erfordernisse des Mediums (Funk) und der Hardware (BiM-Transceiver) - die in Abschnitt 12.1 aufgeführten Vorgaben werden wie folgt realisiert:

Betriebsart: Die Transceiver werden für die Versendung eines Frames in den Sendemodus geschaltet. Direkt nach dem Senden des letzten Bytes wird wieder der Empfangsmodus aktiviert. Es darf sich immer nur ein Roboter im Sendemodus befinden, da es sonst zu einer Kollision und somit zu fehlerhaften Daten kommt. Die Vermeidung von Kollisionen ist Aufgabe der Mediumzugriffsschicht (siehe Abschnitt 12.2.3).

Präambel: Eine Präambel ($0x55$) von 2 ms geht jedem versendeten Frame voraus. Anschließend werden zwei $0xFF$ Bytes gesendet, um die RS232-Empfänger zu synchronisieren.

Bitsymmetrie: Die über einen Zeitraum von 4 ms geforderte, gleiche Zahl von 0-Bits und 1-Bits begründet sich im Ladezustand eines Kondensators innerhalb des BiM-Transceivers, welcher - je nach Frequenzabweichung des eintreffenden Signals - entladen oder geladen wird. Sein Ladezustand entscheidet über die Ausgabe von 0 oder 1. Bei zu vielen aufeinanderfolgenden 1-Werten wird ein folgender 0-Wert weiterhin als 1 interpretiert und führt somit zu fehlerhaften Ausgangsdaten. Für 2 ms kann prinzipiell ein maximal unsymmetrisches Signal gesendet werden, da es in den folgenden 2 ms durch das invertierte Signal ausgeglichen werden könnte. Bildet man eine Summe über alle gesendeten Bits (wobei ein 0-Bit als -1 in die Summe eingeht), so gibt das Ergebnis den Grad der Unsymmetrie der bisherigen Daten an.
Die Forderung kann so weit abgeschwächt werden, dass der Betrag der gebildeten Summe eine obere Schranke nicht überschreiten darf. Die in 2 ms gesendete Anzahl von Bits ist von der Baudrate abhängig und gibt diese obere Schranke und - in negierter Form - die untere Schranke an, welche die gebildete Summe nicht über- bzw. unterschreiten darf. Falls dies eintritt, kann die gebildete Summe durch ein $0x00$ bzw. $0xFF$-Byte zurück in den erlaubten Bereich bewegt werden. Die physikalische Schicht realisiert dies durch einen einfachen Greedy-Algorithmus, der

beim Versenden die Summe mitführt und an den notwendigen Stellen jeweils ein $0x00$ oder $0xFF$-Byte einfügt. Auf der Empfängerseite wird dies ebenfalls durchgeführt und - einen korrekten Empfang vorausgesetzt - an den entsprechenden Stellen das eingefügte Byte wieder entfernt.
Die Länge des Frames wird im worst case beinahe verdoppelt, im best case bewegt sich die gebildete Summe während der Übertragung innerhalb des erlaubten Bereichs. Eine genaue Analyse ist nur bei Kenntnis des Aufbaus der zu versendenden Daten sinnvoll, da diese im Wertebereich üblicherweise nicht gleichverteilt sind. Die vorgestellte Lösung ist aber im average case sicherlich besser als die von Radiometrix [Rad01b] diskutierten Möglichkeiten, die immer mindestens eine Halbierung der Datenrate zur Folge haben: die dort vorgestellte Methode 1 verdoppelt jedes Bit, indem nach jedem Bit ein inverses Bit gesendet wird. Methode 2 schränkt die Werte eines Bytes auf 70 von 256 möglichen - mit gleich vielen 1- und 0-Bits - ein und reduziert die Information in einem gesendeten Byte und damit die Datenrate um ca. 73%. Methode 3 sendet nach jedem Byte das selbe Byte in invertierter Form, do dass die Datenrate um 50% reduziert wird.

12.2.2 Datenverbindungsschicht

Die Datenverbindungsschicht sorgt im Sendebetrieb für eine Zusammenstellung der Frames und für das Hinzufügen der Prüfsumme. Sie implementiert zusätzlich das notwendige Byte-Stuffing in Bezug auf das Start- und Stopp-Byte $0xF0$ bzw. $0x0F$: falls innerhalb des Datenpakets Start- bzw. Stoppbytes vorkommen, so würden diese auf Empfängerseite den Beginn oder das Ende eines Frames signalisieren und zu einem fehlerhaften Datenpaket führen. Dies kann durch einfache Verdopplung jedes Start- bzw. Stoppbytes innerhalb des Datenpakets umgangen werden, was als Byte-Stuffing bezeichnet wird.
Das Hinzufügen der Prüfsumme wird mit Hilfe des Adler-Algorithmus mit einer 32-bit Prüfsummme (Adler-32) [Deu96] durchgeführt: der Adler-32-Algorithmus bildet eine Prüfsumme über eine Menge von Bytes, indem jedes Byte für die Bildung zweier Summen $s1, s2$ verwendet wird. $s1$ ist die Summe aller Bytes, $s2$ ist die Summe aller $s1$-Werte. Beide Summen werden modulo 65521 gerechnet, der größten Primzahl kleiner als 65536. Listing 12.3 zeigt den wesentlichen Algorithmus in C-Quellcode. Der Adler-32-Algorithmus wurde aufgrund seiner schnellen Berechenbarkeit und seiner einfachen Implementierung ausgewählt und genügt den Ansprüchen der durchzuführenden prototypischen Implementierung:

> The Adler-32 algorithm is much faster than the CRC32 algo-

rithm yet still provides an extremely low probability of undetected errors [Deu96].

```
#define BASE 65521 /* largest prime smaller than 65536 */
unsigned long adler = 1L;

unsigned long update_adler32(unsigned long adler,
    unsigned char *buf, int len) {
  unsigned long s1 = adler & 0xffff;
  unsigned long s2 = (adler >> 16) & 0xffff;
  int n;
  for (n = 0; n < len; n++) {
    s1 = (s1 + buf[n]) % BASE;
    s2 = (s2 + s1) % BASE;
  }
  return (s2 << 16) + s1;
}
```

Abbildung 12.3: Adler-32-Algorithmus als C-Quellcode, Quelle: RFC1950

Es sei allerdings erwähnt, dass andere Algorithmen im Vergleich zum Adler-32-Algorithmus bei kurzen Datenpaketen mit weniger als 128 Bytes eine größere Sicherheit bezogen auf die Wahrscheinlichkeit nichterkannter Fehler von 1 in 2^{32} (CRC-Algorithmus) bieten; dies kann detailliert in [Ada01] und [SGHP98] nachgelesen werden.

12.2.3 Mediumzugriffsschicht

Die Mediumzugriffsschicht speichert Frames, die von der Datenverbindungsschicht weitergereicht werden, in einer Warteschlange (Ringpuffer). Sie wartet mit dem Versenden jedes Frames bis eine Reihe von Bedingungen zutreffen, die vom verwendeten Verfahren (beispielsweise CSMA oder TDMA) vorgegeben werden. In der vorliegenden Implementierung wurde ein CSMA-Verfahren verwendet, das sich für ereignisbasierte Kommunikation (deren Vorkommen im Roboterfußball überwiegt) besser eignet als TDMA oder Token-basierte Verfahren [MN99].
In der prototypischen Implementierung wird der *p-persistent CSMA* Algorithmus eingesetzt, der wie folgt abläuft: die Zeit wird in Intervalle (Timeslots) unterteilt. Falls eine Station senden möchte, prüft sie mit Hilfe des Carrier-Detect Signals, ob das Medium verfügbar ist. Falls ja, beginnt die Station mit der Wahrscheinlichkeit p zu senden und wartet mit der Wahrscheinlichkeit $q = 1 - p$ bis zum nächsten Zeitintervall. Das Vorgehen wird solange wiederholt, bis entweder die Nachricht versendet wurde oder eine andere Station das Medium belegt.
Die Wahrscheinlichkeit p in Verbindung mit der Anzahl von potentiellen

Sendern entscheidet über die Wahrscheinlichkeit einer Kollision und über die Auslastung des Mediums. Tanenbaum [Tan96]$_{S.246ff}$ führt eine Analyse verschiedener Verfahren durch und vergleicht das Verhalten für verschiedene Wahrscheinlichkeiten p. Zu Zwecken unserer Demonstration wird eine Wahrscheinlichkeit $p = 0.02$ gewählt, welche die Wahrscheinlichkeit von Kollisionen auch für eine größere Zahl von Sendeversuchen gering hält [Tan96]$_{S.252}$. Durch Anpassung von q kann in weiteren Untersuchungen der Durchsatz des Datenkanals optimiert werden, diese Verfahren geht aber über die Intention der Demonstration der grundsätzlichen Kommunikationsfähigkeit in dieser Arbeit hinaus.

12.2.4 Netzwerkschicht

In der vorliegenden Aufgabenstellung des Roboterfußballs existiert nur ein Subnetz und alle Roboter befinden sich innerhalb der Sendereichweite jedes anderen Kommunikationspartners. Aus diesem Grund kann diese Schicht bei der vorliegenden Problemstellung entfallen. In einer erweiterten Aufgabenstellung, bei der nicht mehr alle Systeme innerhalb der Sendereichweite liegen, und Systeme Daten auf dem Weg vom Sender zum Empfänger weiterleiten müssen, kann diese Schicht zusätzlich implementiert werden.

12.2.5 Transportschicht

Die Transportschicht implementiert die in Abschnitt 12.2 vorgestellten Dienste:

Dienst 1

Dieser Dienst implementiert den *acknowledged connection-oriented service*, indem der Auf- und Abbau von Verbindungen zwischen je zwei Kommunikationspartnern ermöglicht und die Übertragung der Daten sicherstellt wird. Dabei bleibt zusätzlich die korrekte Reihenfolge der Daten erhalten. Das verwendete Protokoll nutzt die sogenannte *sliding window* Technik, bei der mehrere Frames zwischengespeichert werden. Erst nach korrekter Übertragung und Empfangsbestätigung durch den Kommunikationspartner wird ein Frame aus der Zwischenspeicherung des Senders gelöscht. Jeder Frame erhält eine eindeutige Sequenz-Nummer, damit der Empfänger die korrekte Reihenfolge der einzelnen Frames rekonstruieren kann. Beide Kommunikationspartner verwalten ein Fenster der gültigen Sequenznummern, die sie jeweils vom Partner erwarten. Wenn der Frame mit der kleinsten erwarteten Sequenznummer empfangen wird, kann das Fenster um einen Schritt weiterbewegt werden. Der genaue Ablauf der Kommunikation kann in [Tan96]$_{S.206ff}$ nachgelesen werden.

Dienst 2

Dieser Dienst implementiert den *acknowledged connectionless service*: das zu sendende Paket wird zwischenspeichert und auf eine Empfangsbestätigung des Empfängers gewartet. Falls diese innerhalb einer gewissen Zeit nicht eintrifft, wird das Paket erneut gesendet. Nach einer maximalen Zahl von Sendewiederholungen wird der Kommunikationsversuch abgebrochen, da der Empfänger scheinbar nicht erreicht werden kann.

Dienst 3

Dienst 3 implementiert den *unacknowledged connectionless service*, indem ein Paket ohne Zwischenspeicherung direkt an die Datenverbindungsschicht weitergegeben wird. Der Eingang des Pakets beim Empfänger ist ungewiss, da das Paket während der Übertragung im Gegensatz zu Dienst 1 und Dienst 2 verloren gehen kann.

12.2.6 Präsentationsschicht

Die Kodierung der Nutzdaten ist in der vorliegenden Implementierung nicht notwendig, da es sich in der prototypischen Implementierung um Befehls- und Datenprotokolle auf Byteebene handelt. Aus diesem Grund wird keine Präsentationsschicht implementiert.

12.2.7 Applikationsschicht

In dieser Schicht können die von den Diensten zur Verfügung gestellten Funktionen zur Implementierung der einzelnen Kommunikationsaufgaben verwendet werden.

12.3 Implementierung

Die Implementierung der diskutierten Kommunikationslösung erfolgt sowohl für die Host- als auch die Roboterplattform. Zu diesem Zweck werden die gleichen C-Quelldateien verwendet, die sich nur in der Implementierung einer Quelldatei *DspHal.c, HostHal.c* für die jeweils definierte Zielplattform unterscheiden (DSP, Win32/Linux). Auf diese Weise wird eine Inkonsistenz der Quelldateien und damit der Algorithmen und Datenstrukturen zwischen den einzelnen Plattformen weitgehend verhindert. Im Folgenden sollen die wichtigsten der entwickelten Algorithmen und Datenstrukturen erläutert werden - Details der einzelnen Schichten können dem Quellcode auf der beiliegenden CD-Rom entnommen werden.

Datenstruktur *packet*		
Variable	*Typ*	*Bedeutung*
length	unsigned short	Nutzdatenlänge in Bytes
data	unsigned char[MAX_PKT]	Nutzdaten

Tabelle 12.1: Datenstruktur *packet*

Datenstruktur *frame*		
Variable	*Typ*	*Bedeutung*
sender	unsigned char	ID des Senders
receiver	unsigned char	Bitmaske der Empfänger-IDs
kind	unsigned char	Art des Frames (Dienst)
con_id	unsigned char	ID der Verbindung
seq_nr	unsigned char	Nummer des Frames
ack	unsigned char	Acknowledge (piggybacked)
info	packet	Nutzdatenpaket
chksum	unsigned long	Adler32 Checksumme

Tabelle 12.2: Datenstruktur *frame*

12.3.1 Datenstrukturen und Algorithmen

Die verwendeten Datenstrukturen werden in der Datei *rfCom.h* definiert.

Datenpaket (Datenstruktur)

Ein Datenpaket *packet* (Tabelle 12.1) besteht aus einem Array von Nutzdaten *data* sowie der Länge der Nutzdaten in Byte *length*. Das Nutzdatenarray hat eine maximale Länge von $MAX\_PKT = 512$ Bytes; dies ist gleichzeitig die Obergrenze der Nutzdatenbytes in jedem gesendeten Frame. Ein Datenpaket wird je nach Dienst entweder von der Sitzungsschicht oder von der Applikation direkt erzeugt und an die darunter liegende Schicht übergeben.

Frame (Datenstruktur)

Ein Datenframe *Frame* (Tabelle 12.2) enthält neben einem Datenpaket *info* mit den Nutzdaten weitere Daten, die von den jeweiligen Schichten gesetzt und ausgelesen werden können. Die Bytes in dieser Datenstruktur entsprechen bis auf Präambel und den Synchro-, Start- und Stoppbytes den an die physikalische Schicht übergebenen und somit näherungsweise den gesendeten bzw. empfangenen Bytes (Abbildung 12.2).

12.4 Ergebnis

Die gewünschten Dienste wurden erfolgreich implementiert und getestet. Anwendungsentwickler können die in den Headerdateien zur Verfügung gestellten Schnittstellen nutzen, um unter Auswahl der verschiedenen Dienste zu kommunizieren.
Einige vom Autor durchgeführte Tests zeigten folgende Ergebnisse:

- Die Funkverbindung funktioniert mit allen vorhandenen BiM und BiM2 Varianten zuverlässig. Die angegebenen maximalen Datenraten sind ohne einen signifikanten Anstieg der Übertragungsfehler erreichbar. Dies bedeutet eine Verbesserung um den Faktor sechs im Vergleich zum RobotCavalry System. Der Nutzdatendurchsatz ist durch den Overhead für Präambel, Start- und Stoppbytes und den Framedaten wie Sender, Empfänger und Prüfsumme entsprechend geringer. Pro Frame werden mindestens 14 Bytes für administrative Daten verwendet, die Länge der Präambel hängt von der Baudrate der Verbindung ab und beträgt bei 56kBit/s 15 Byte.

- Der Abstand zwischen Sender zum Empfänger wirkt sich - im Bereich von bis zu 10 Metern - nicht meßbar negativ auf die Fehlerrate der Übertragung aus. Abstände über 10 Meter kommen in der MiroSot-Klasse nicht vor.

- Die Störung des Funkempfangs zum Beispiel durch Einschalten eines weiteren Senders unterbricht zwar den Datenverkehr, nach Abschalten der Störung werden Daten der acknowledged-Dienste durch das implementierte sliding window Protokoll nachträglich gesendet und die Kommunikation wird fortgesetzt.

12.5 Zukünftige Arbeiten

Die Mediumzugriffsschicht entscheidet darüber, wann ein Roboter einen Frame absenden darf. Die Konzipierung dieser Schicht entscheidet somit über eine Einhaltung von Echtzeitbedingungen. Der vorgestellte CSMA-Algorithmus eignet sich aufgrund der Möglichkeit von Kollisionen nicht für harte Echtzeitbedingungen. Er bietet allerdings den Vorteil, den Zugriff auf das Medium nicht an ein festes Zeitfenster zu binden und somit bei Eintritt eines Ereignisses mit einer gewissen Wahrscheinlichkeit sofort senden zu können. Mock und Nett [MN99] diskutieren Echtzeitkommunikation für autonome Robotersysteme. Harte Echtzeitanforderungen können mit Hilfe von TDMA oder Token-basierten Ansätzen gelöst werden, eignen sich aber aus genannten Gründen weniger für ereignisbasierte Kommunikation:

> For achieving predictable hard real-time communication on multiple-access busses, TDMA based [FH 76, ML 81] and token based [MZ 94] approaches are well established. However, they address only the timely delivery of synchronous messages that occur in time-triggered environments, such as described in [KG 94].

Für ereignisbasierte Nachrichten, die in autonomen Roboterfußballsystemen häufig vorkommen, eignen sich CSMA-Verfahren, allerdings ohne harte Echtzeitbedingungen zu erfüllen:

> For the delivery of event-triggered, asynchronous messages, a number of collision based soft real-time protocols have been proposed [KY 88, ZR 87, Bos 91], which however lack of integration with hard real-time communication.

Mock und Nett stellen eine Lösung durch eine Erweiterung des TDMA-Ansatzes vor: innerhalb von manchen TDMA-Slots ermöglichen sie CSMA und mischen somit beide Verfahren. Durch die weiterhin bestehenden TDMA-Slots können Garantien bzgl. der harten Echtzeitbedingungen eingehalten werden und durch die CSMA-Slots ist eine ereignisbasierte Kommunikation möglich. Dies entspricht den Verfahren, die für wireless LAN Protokolle verwendet werden. Eine Mischung von TDMA und CSMA erfordert allerdings ein exaktes Timing; diese Voraussetzung ist auf Hostseite ohne eigenen Mikroprozessor für das Funksystem (Basebandcontroller) schlecht realisierbar und ist in der Beschaffenheit der seriellen Schnittstelle begründet, die nur unzureichend über Betriebssystem-APIs kontrolliert werden kann. Ein *Basebandcontroller*-DSP könnte zum Hostrechner-Funkmodul hinzugefügt werden und weitere Untersuchungen und Implementierungen ermöglichen.
In Zukunft werden Technologien wie zum Beispiel Bluetooth kleiner und kostengünstiger werden, um sie in kleine mobile Roboter zu integrieren. Auch wireless-LAN und ein embedded-Linux auf den Robotern wären denkbar. Diese Erweiterungen würden zusätzliche Bandbreite und weitere Flexibilität bei der Kommunikation bedeuten. Allerdings benötigen diese Technologien aufgrund ihrer Komplexität zumeist einen größeren Aufwand an Rechenleistung, Speicherplatz und spezieller Hardware und werden daher erst mit künftigen Hardwareplattformen in der geforderten Größe zu realisieren sein.

Kapitel 13

Lokale Bildverarbeitung

In diesem Kapitel wird die konzeptionelle Integration einer lokalen Bildverarbeitung für den entwickelten Fußballroboter beschrieben. In einem ersten Schritt wird die Verwendung einer globalen Bildverarbeitung bei teilautonomen Fußballrobotern diskutiert, um anschließend Potentiale einer lokalen Bildverarbeitung aufzuzeigen.

13.1 Möglichkeiten der globalen Bildverarbeitung

Eine globale Bildverarbeitung besteht bei teilautonomen Roboterfußballsystemen aus einem bildverarbeitenden Hostsystem und einer Kamera, die mittig über dem Spielfeld angeordnet ist (vgl. Abbildung 2.1) und im allgemeinen für die des Balles und der Roboter verwendet wird.
Auf diese Weise erkannte Positionen werden üblicherweise auf dem Hostsystem weiterverarbeitet und den Robotern die resultierenden Bewegungsbefehle mit Hilfe des Funksystems übermittelt. Dieses Verfahren weist verschiedene Nachteile auf:

- Die Latenzzeit und die Unsicherheit des Funksystems können zu verspäteten oder verlorenen Datenpaketen führen. In jedem Fall basiert die Bewegung des Roboters auf nicht mehr aktuellen Daten bzw. wird im schlimmsten Fall nicht ausgeführt.

- Falls der Roboter - wie in dieser Arbeit angeregt - Bewegungen planen und über einen gewissen Zeitraum autonom durchführen soll, ist eine stetige Übertragung der Ball-Position notwendig, falls der Roboter den Ball anfahren oder führen soll. Diese stetige Übertragung unterliegt ebenfalls den bereits angeführten Nachteilen: eine schnelle Ballführung erfordert einen schnell reagierenden Regelkreis und erscheint - anhand der bisherigen Erfahrungen - mit Hilfe einer globalen Bildverarbeitung nur unzureichend realisierbar.

- Die bisher von PG340 und PG362 implementierten globalen Bildverarbeitungs-Systeme sind in hohem Maße fehleranfällig und reagieren mehr oder weniger stark auf veränderte Lichtverhältnisse. Dies kann zu fehlerhafter Erkennung der Roboter-IDs und im schlimmsten Fall zur Verwechselung von Robotern führen.

- Erkennungsprobleme der globalen Bildverarbeitung (beispielsweise aufgrund schlechter Lichtverhältnisse) wirken sich auf die Funktionsfähigkeit des gesamten Systems aus. Die Autonomie der Roboter ist durch den Einsatz einer globalen Bildverarbeitung derart eingeschränkt, dass sie bei Ausfall des externen Systems nicht mehr spielfähig sind.

Vor allem derartige Abhängigkeiten sollten bei autonomen Multiagentensysteme nicht vorkommen. Der Einsatz einer lokalen Bildverarbeitung, für die jeder Roboter ein eigenes Bildverarbeitungs-System erhält, könnte von Vorteil sein und bietet folgende Verbesserungsmöglichkeiten:

- Ein Roboter nimmt Objekte und deren Bewegung in seinem lokalen Umfeld wahr und ist in der Lage, zum einen schnell auf diese Aktivitäten zu reagieren und zum anderen diese Informationen den anderen Systemen zur Verfügung zu stellen. So kann aus den gesammelten lokalen Informationen aller Systeme auf die globale Spielsituation geschlossen werden.

- Das Anfahren, Führen und Schießen des Balles kann durch sehr effizient arbeitende Regelungskonzepte erreicht werden, da alle zu dieser Fähigkeit benötigten Informationen durch die Sensorik des Roboters geliefert werden können. Die Erkennung eines Balles in direktem Kontakt zum Roboter kann von einer lokalen Bildverarbeitung - im Gegensatz zu einer globalen Bildverarbeitung - mit kleinerem absoluten Fehler geleistet werden, da für diese Fähigkeiten die relative Position des Balles bezüglich des Roboters wichtiger ist, als seine absolute Position auf dem Spielfeld.

- Die Fehlertoleranz des Systems wird erhöht, da jeder Roboter durch die Erkennung von Ojekten einen Beitrag zu globalen Informationen über die Position und Bewegung des Balles, der eigenen Roboter und der gegenerischen Roboter beitragen kann.

- Die Autonomie jedes Roboters wird erhöht. Eine lokale Bildverarbeitung bietet das Potential, alle vom Hostrechner durchgeführten Aufgaben durch dezentrale Verfahren mit Hilfe der einzelnen autonomen Robotersysteme zu bearbeiten.

Die Nachteile einer solchen Lösung liegen in der erhöhten Komplexität und der notwendigen Miniaturisierung des bildverarbeitenden Systems.
Die geführte Diskussion zeigt wesentliche Potentiale auf, die Gegenstand der wissenschaftlichen Untersuchungen anderer Roboterfußball-Ligen sind, bei denen globale Bildverarbeitungs-Systeme nicht zugelassen sind. In diesen Ligen sind die Größenvorgaben für die Roboter so gewählt, dass bildverarbeitende Systeme in üblicher Größe und Bauweise in die Roboter integriert werden können (vgl. Abschnitt 3.3).

13.2 Konzeption einer lokalen Bildverarbeitung

Im folgenden Abschnitt wird die Konzeption eines bildverarbeitenden Systems für den erstellten MiroSot-Fußballroboter beschrieben.

13.2.1 Hardware

Die mögliche Kamera-Sensorik wurde bereits im Verlauf der Arbeit beschrieben (vgl. Abschnitt 5.5.9). Im folgenden Abschnitt wird die Verwendung und Anbindung der Sensoren an das lokale Bildverarbeitungs-System erläutert: Das lokale Bildverarbeitungs-System liegt zur Zeit der Arbeit in einer Prototypversion vor. Es besteht aus einem TMS320F206-DSP, der mit dem gleichen externen Hauptspeicher ausgestattet ist, wie der TMS320F240 des DSP-Systems. Die Platine des Bildverarbeitungs-Systems liegt zum Abschluss dieser Arbeit in einer funktionierenden Prototypversion vor. Sie besitzt eine Größe von $38 * 42$ mm und lässt sich im Inneren des Roboters, oberhalb der Leistungselektronik montieren (Abbildung 13.1).

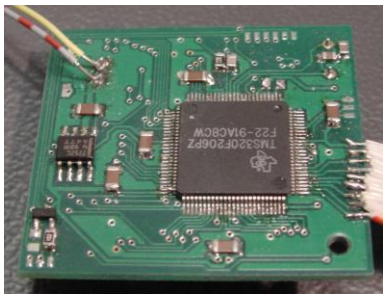


Abbildung 13.1: Prototyp des Bildverarbeitungs-DSPs

Der Bildverarbeitungs-DSP wird mit zwei Kamera-Sensoren vom Typ VV6301 [ST 00b] mit Hilfe des I/O-Adressbereichs des DSP verbunden. Die notwendige Dekodierlogik für Speicherzugriffe und für Zugriffe auf die Kamera-Daten stellt wiederum ein 16V8-GAL-IC zur Verfügung.

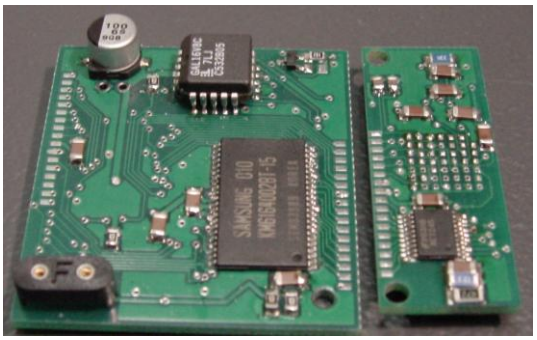


Abbildung 13.2: Bildverarbeitungs-DSP (Unterseite) mit Kamera-Sensor

Der Bildverarbeitungs-DSP wird mit Hilfe einer seriellen interprozessor-Schnittstelle mit dem DSP des Haupt-Systems verbunden. Mit diesem Verfahren kann der Bildverarbeitungs-DSP dem DSP des Haupt-Systems beispielsweise Resultate der Objekterkennung mitteilen. Auch die Übertragung ganzer Bilder ist in Echtzeit mit Hilfe dieser Schnittstelle möglich.
Details der Schaltung des Bildverarbeitungs-Systems können den im Anhang beigefügten Schaltplänen entnommen werden.

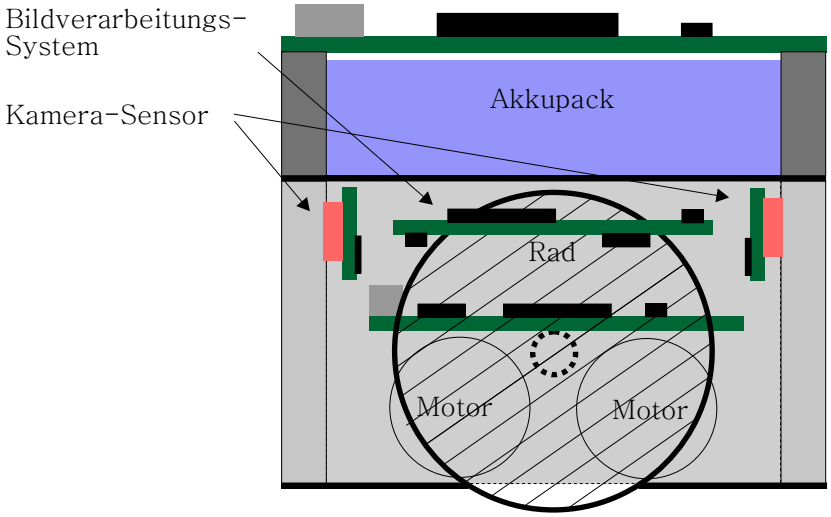


Abbildung 13.3: Struktur des Bildverarbeitungs-Systems

13.2.2 Kamera-Sensor

Die Kamera-Sensor-Platine enthält einen Kamera-Sensor-IC und einen 8-bit Bus-Treiber-IC vom Typ *ACT 245* [Fai99], der den Datenbus des Kamera-IC und des DSP trennt und nur bei Zugriffen des DSP auf bestimmte I/O-Adressen die beiden Datenbusse verbindet (Abbildung 13.4).

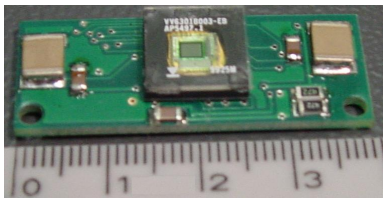


Abbildung 13.4: Prototyp des Kamera-Sensors

13.2.3 Integration der Kamera-Sensorik

Die Platine des Bildverarbeitungs-Systems wird oberhalb der Leistungselektronik-Platine im Robotergehäuse angebracht. Auf beiden Seiten der Platine kann senkrecht je eine Kamera-Sensor-Platine angebracht werden, die mit Hilfe einer Linse im Gehäuse des Roboters ein Bild in bzw. gegen die Fahrtrichtung aufnehmen kann (Abbildung 13.3). Die Ausrichtung der Kamera-Sensoren in bzw. gegen die Fahrtrichtung ermöglicht dem Roboter den direkten „Blick“ auf vor ihm liegende Objekte. Diese Fähigkeit unterstützt im besonderen Aufgaben, bei denen eine zielgenaue visuelle Anfahrt eines - sich eventuell bewegenden - Objekts durchgeführt werden muss. Vor allem ist dies für den Schuss und das Führen des Balles von Bedeutung (siehe Abbildung 13.5).

Abbildung 13.5: Mögliches Bild des Kamera-Sensors

13.2.4 Software

Die entworfene Bildverarbeitungs-Hardware ermöglicht die Verarbeitung einzelner Bild-Frames der Kamera-Sensoren. Die Daten eines Frames mit ca. 160*120 Bildpunkten können in $9,6$ kWorten des Hauptspeichers abgelegt werden. Dem DSP stehen 64 kWorte Datenspeicher sowie 32 kWorte globaler Speicher zur Verfügung, so dass Speicherplatz für ca. $8-10$ Frames vorhanden ist. Der realistische Speicherbedarf liegt bei 4 Frames: die aktuellen Frames der beiden Kamera-Sensoren werden in zwei Speicherbereichen bearbeitet, während die neuen Frames von den Sensoren geliefert werden

und in zwei andere Bereiche des Speichers geschrieben werden. Dieses Verfahren wird häufig als *double-buffering* bezeichnet.
Der DSP kann - sobald Frames in digitaler Form in seinem Hauptspeicher vorliegen - beliebige bildverarbeitende Algorithmen anwenden. Er verfügt über eine Rechenleistung von bis zu 20 MIPS (Millionen Instruktionen pro Sekunde), die für die Verarbeitung zweier Frames mit maximaler Framerate der Kameras[1] ausreichen müsste.

13.2.5 Objekterkennung

Die Erkennung von Objekten wäre primäre Aufgabe eines lokalen bildverarbeitenden Systems. Die zu detektierenden Objekte sind zum einen der Ball und zum anderen eigene und gegnerische Roboter.

Ball

Die Erkennung des Balles kann ähnlich der Ballerkennung der globalen Bildverarbeitung durchgeführt werden (vgl. [PG 00]$_{S.24ff}$ und [PG 01]$_{S.43ff}$). Der Ball ist durch seine - in den Spielregeln festgelegte - orange Färbung ein farblich sehr gut zu erkennendes Objekt. Kein anderes Objekt dürfte aus Sicht der lokalen Kamera-Sensoren eine solche Färbung aufweisen[2].

Roboter

Roboter sind für die globale Bildverarbeitung anhand von Farbmarkierungen erkennbar, die auf ihrer Abdeckung angebracht sind. Für eine lokale Bildverarbeitung könnte eine ähnliche Reglung verwendet werden, bei der die Teamfarbe seitlich oder in Fartrichtung aufgebracht wird. Eine Anbringung in Fartrichtung kann in Verbindung mit der Anordnung der Kamera-Sensoren eine Abstimmung der Ausrichtung zweier Roboter ermöglichen (vgl. Abschnitt 13.2.6).

Ausrichtung und Entfernung der Objekte

Der Roboter kann aus seiner lokalen Sicht Aussagen über die Richtung des Objekts und über die Entfernung des Objekts machen. Informationen über die Richtung können aus der lokalen Ausrichtungsinformation des Roboters in Verbindung mit der horizontalen Position des Objekts im Kamerabild gewonnen werden. Die Entfernung eines Objekts kann ebenfalls aus dem

[1] Die Framerate der Kamera-Sensoren ist bei diesem Typ abhängig von der Helligkeit des Bildes, sollte aber bei normaler Helligkeit bei 25-30 Frames pro Sekunde liegen.

[2] Eine solche Klausel ist für die globale Bildverarbeitung in den Regeln festgelegt, sie dürfte bei Verwendung von lokalen Bildverarbeitung auch für die Seitenflächen der Roboter gelten

Kamerabild bestimmt werden. Zu diesem Zweck kann die Kamera einen leicht geneigten Blick auf das Spielfeld erhalten, so dass aus der vertikalen Position eines Objekts im Bild auf die Entfernung geschlossen werden kann (vgl. Abbildung 13.6). Zusätzliche Informationen können aus der Größe des Objekts gewonnen werden: je weiter beispielsweise der Ball entfernt ist, desto kleiner wird er aus Sicht der Kamera-Sensoren.

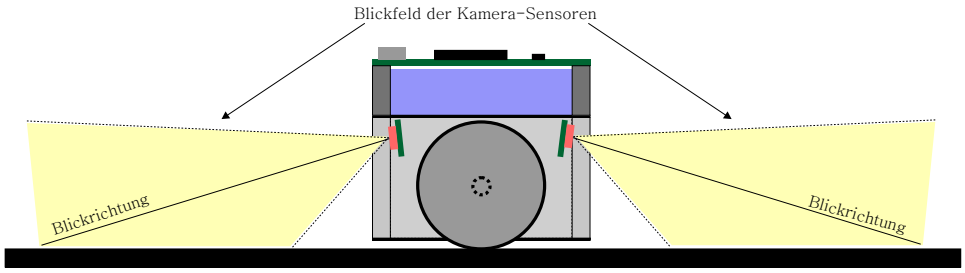


Abbildung 13.6: Blickfeld der Kamera-Sensoren

13.2.6 Navigation

Die Verwendung einer lokalen Bildverarbeitung kann die Roboter auch in Hinsicht ihrer Navigation unterstützen. Die bisher vorgestellten Navigationsverfahren sind auf eine Korrektur der Positionsdaten durch eine globale Bildverarbeitung angewiesen.
Für die Navigation mit Hilfe lokaler bildgebender Sensorik existieren verschiedene Verfahren - beispielsweise Navigation mit Hilfe von künstlichen oder natürlichen Landmarken (vgl. Borenstein [JB96]$_{S.173ff}$).

Auch die beschriebene Objekterkennung kann - in einem ersten Schritt - zur Korrektur der lokalen Ausrichtung und der relativen Positionen der Roboter zueinander verwendet werden: zwei sich gegenseitig „anschauende“ Roboter können sich - eine Markierung der Roboter in Blickrichtung der Kamera vorausgesetzt - einander zuwenden und ihre - mit Hilfe der Odometrie berechnete - Ausrichtung vergleichen und korrigieren. Durch die Bestimmung des Abstands beider Roboter kann die relative Position der beiden Objekte zueinander berechnet werden. Auf diese Weise ist auch eine Positionskorrektur möglich.
Ein weitere Maßnahme stellt die Anbringung von künstlichen Markierungen am Spielfeld dar. Um lokale bildverarbeitende Systeme zu unterstützen, wird in den RoboCup-Ligen eine Farbmarkierung der Tore durchgeführt. Dieses Verfahren ist in den Regeln der MiroSot-Liga nicht vorgesehen, da bisher keine lokale Bildverarbeitung in MiroSot-Roboter integriert wurde. Die Sony-4-legged-League des RoboCup verwendet zusätzlich farbige Mar-

kierungen in den vier Ecken des Spielfelds um den Robotern eine bessere Navigation zu ermöglichen. Alle bisher in diesen Ligen vorgenommenen Markierungen sind künstlicher Natur und werden mit Hilfe von Landmarken-Erkennungsalgorithmen verarbeitet.

13.3 Ergebnis und zukünftige Arbeiten

Anhand der Überlegungen lässt sich erkennen, dass die Integration von Kamera-Sensoren einen wesentlicher Schritt bezüglich einer Verbesserung der Autonomie der Roboter darstellt. Dies wird von Graf [BG99] unterstrichen:

> We believe that that the most essential mobile robot research and education can be performed using small mobile robots with vision, as opposed to large, heavy, expensive and potentially harmful robot systems.

Lokale Bildverarbeitung unterstützt den Roboter in vielen Aufgaben und kann unter Umständen das globale bildverarbeitende System ersetzen.
Zukünftige Arbeiten können die vorgestellten Verfahren implementieren und untersuchen, ob das in dieser Arbeit konzipierte und prototypisch realisierte System die geforderten Aufgaben erfüllen kann.

Kapitel 14

Abschluss

In diesem Kapitel wird abschließend eine Zusammenfassung der Arbeitsergebnisse präsentiert und ein Ausblick auf mögliche zukünftige Arbeiten gegeben.

14.1 Erzieltes Resultat

Die Aufgabe dieser Diplomarbeit bestand in der Konzeption und im Aufbau eines teilautonomen Fußballroboters für die FIRA-MiroSot-Liga.
Eine prototypische Implementierung des Softwaresystems sollte - verbunden mit der Demonstration exemplarischer Fähigkeiten - die Verwendbarkeit des Roboters zeigen und Schlüsse auf das Potential des Systems zulassen.

14.1.1 Konzeption

Die Konzeption des Roboters orientierte sich am Entwurf einer Forschungsplattform für die beteiligten Wissenschaftsbereiche und an der optimierten Verwendbarkeit des Systems für den Roboterfußball: die Ziele der Konzeption wurden erreicht. Das entwickelte System unterstützt zum einen - im Gegensatz zur bisher verwendeten Roboterplattform - die wissenschaftliche Arbeit und ermöglicht zum anderen die Anwendung der Forschungsergebnisse ausserhalb des Roboterfußballs.
Ein möglichst autonomes Verhalten des Roboters bezüglich seiner verschiedenen Aufgabenbereiche konnte durch die Integration von zusätzlicher Sensorik erreicht werden.
Anhand der erzielten Ergebnisse lässt sich erkennen, dass die Größenbeschränkung der FIRA MiroSot-Klasse nicht zwangsläufig zu Einschränkungen der verwendeten Sensorik und den resultierenden Fähigkeiten führen muss: ein MiroSot-Roboter kann - durch die in dieser Arbeit vorgestellten Maßnahmen - einen hohen Grad an Autonomie erreichen.

14.1.2 Aufbau und prototypische Implementierung

Der Aufbau des Roboters wurde über insgesamt vier Versionen optimiert: nach jeder Version wurde eine Überprüfung der bereits realisierten und der potentiellen Fähigkeiten durchgeführt und das Design bezogen auf die Auswahl der Bauteile, ihre Anordnung und Schaltung sowie in Bezug auf die mechanischen Vorgaben[1] optimiert.
Die prototypische Implementierung bezog sich vor allem auf die Ansteuerung der einzelnen Hardwarekomponenten und ermöglicht in der vorliegenden Version die Abstraktion von den Details des Hardware-Entwurfs. Die Implementierung wurde erfolgreich abgeschlossen und bildet eine Grundlage für die effiziente Erstellung zukünftiger Softwarekomponenten.

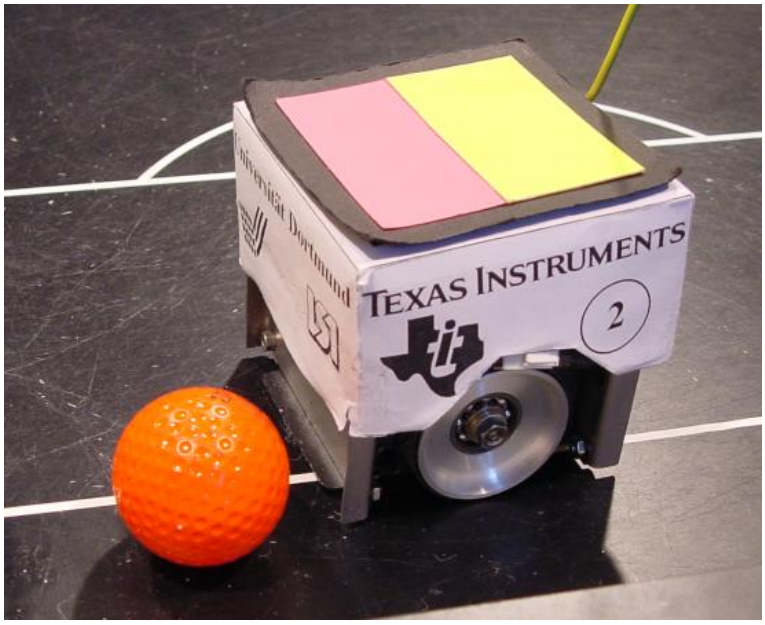


Abbildung 14.1: Ein Ergebnis der Arbeit: der Fußballroboter

14.1.3 Exemplarische Fähigkeiten

Um das Potential des Roboters zu demonstrieren, wurden verschiedene ausgewählte Fähigkeiten aus angrenzenden Forschungsgebieten diskutiert und implementiert. Die Auswahl der Bereiche diente zur Evaluierung des Potentials der entworfenen Hardware und orientierte sich an der Notwendigkeit grundlegender Fähigkeiten. Die Diskussion dieser Bereiche zeigte potentielle Lösungen für spezifische Problemstellungen auf; durch eine Implementie-

[1] An dieser Stelle sei Markus Mohr für den Entwurf der Mechanik und die nachhaltige Begleitung des Entwicklungsprozesses gedankt.

rung konnten die Fähigkeiten im Vergleich zum bestehenden System deutlich verbessert werden. Die erzielten Ergebnisse werden in den folgenden Abschnitten zusammengefasst.

Positionsbestimmung

Das bisherige Roboterfußballsystem verwendete ein Steuerungs- und Regelungskonzept, das sich in verschiedenen Punkten als nachteilig herausstellte. Mögliche Verbesserungen wurden diskutiert und eine Regelungsstruktur wurde entworfen, die Verbesserungen durch eine erhöhte Flexibilität und eine erweiterte Autonomie des Roboters ermöglicht. Grundlage dieser Stuktur bildet die Bestimmung der Position und der Ausrichtung des Roboters mit Hilfe der lokalen Sensorik des Roboters.
Odometrie und inertiale Navigation wurden als zwei mögliche Verfahren zur Positionsbestimmung mit Hilfe lokaler Sensorik vorgestellt, und ihre Verwendbarkeit wurde evaluiert.
Anhand der erzielten Ergebnisse lässt sich erkennen, dass die Odometrie sich für eine Navigation im lokalen Umfeld des Roboters - mit Bewegungen von zwei bis drei Metern Länge - eignet. Die Implementierung dieses Verfahrens bestätigte seine Verwendbarkeit.
Die inertiale Navigation ist aufgrund ungenauer Sensordaten in der vorliegenden Version des Roboters für die Navigation unzureichend. Ob grundsätzlich ein inertiales Navigationssystem mit der erforderlichen Genauigkeit für einen Fußballroboter entworfen werden kann, ist fraglich. Kommerzielle und hoch qualitative inertiale Navigationssysteme weisen bereits Driftfehler auf, die eine Anwendung im Roboterfußball in Frage stellen.
Der verwendete Beschleunigungs-Sensor wurde allerdings nicht mit der Zielsetzung eines inertialen Navigationssystems verwendet, sondern um Ereignisse zu detekieren, die für andere Sensoren des Roboters nicht messbar sind. Solche Ereignisse können Kollisionen mit anderen Objekten oder das zu starke Beschleunigen des Roboters und einem daraus resultierenden Haftungsverlust der Räder sein. Mit Hilfe dieser Detektion lassen sich Ereignisse wahrnehmen, die unter Umständen großen Einfluß auf die Bewegung des Roboters haben.

Kommunikation

Die entwickelte Kommunikationslösung bietet wesentliche Verbesserungen verglichen mit der bisherigen, unidirektionalen Funkkommunikation. Die Notwendigkeit einer bidirektionalen Kommunikation wurde gezeigt, und die zu diesem Zweck entworfene Kommunikationshardware erfolgreich eingesetzt. Eine Software-Architektur wurde - auf dem OSI-Schichtmodell basierend - entworfen und erfolgreich implementiert. Mit diesen Maßnahmen

konnten folgende Verbesserungen erreicht werden:

- Die Bandbreite des Kommunikationskanals kann - unter den gegebenen Umständen - optimal ausgenutzt werden, so dass eine Verbesserung um den Faktor sechs im Vergleich zu den bisherigen Fußballrobotern erzielt wird.

- Eine bidirektionale und gleichzeitige Kommunikation mehrerer Partner wurde durch die Implementierung eines CSMA-Verfahrens (carrier sense multiple access) ermöglicht.

- Die Kommunikation kann auf drei verschiedene Dienste zurückgreifen, die verschiedene Kommunikationsarten unterstützen. Die Dienste unterscheiden sich in der Verlässlichkeit der Kommunikation und in der Länge der möglichen Datenübertragung.

- Übertragungsfehler werden detektiert und ermöglichen somit eine Integritätsprüfung der Daten.

- Durch ein neu entworfenes Funkmodul für das Hostsystem wurde die Erkennung und Analyse von Funkproblemen wesentlich verbessert. Dies wurde durch optische Signale erreicht, die den Zustand des Kommunikationskanals sichtbar machen.

- Verschiedene Versionen der verwendeten Funkmodule funktionieren erstmals zuverlässig mit einer einheitlich konzipierten Kommunikationslösung.

Die resultierende Kommunikationslösung erfüllt erstmals notwendige Voraussetzungen für eine effektive Kommunikation zwischen autonomen Multiagentensystemen im Roboterfußball.

Steuerung und Regelung

Der entwickelte Roboter ermöglicht eine Bewegung mit einer Geschwindigkeit bis zu 2 Metern pro Sekunde, was für die Verwendung in der MiroSot-Liga mehr als ausreicht. Aufgrund der Konzeption ist der Roboter in der Wahl von Steuerungs- und Regelungsverfahren flexibel. Derartige Verfahren können als Softwaremodul implementiert werden und sind folglich einfach austauschbar. Diese Lösung ist ein wesentlicher Vorteil zum bisher verwendeten Fußballroboter, dessen Steuerung und Regelung durch die Hardware des Roboters festgelegt war.
Durch die Integration von Sensorik wurde die Verlagerung des Regelkreises

vom Hostsystem zum Roboter ermöglicht und eine Grundlage für wesentlich kompaktere und schnellere Regelungen geschaffen. Das externe Hostsystem ermöglicht die Unterstützung des Roboters durch absolute Positionsdaten, die zur gelegentlichen Korrektur der lokalen fehlerbehafteten Positionsberechnung herangezogen werden können.

Lokale Bildverarbeitung

Die Integration einer lokalen Bildverarbeitung in das entworfene System kann unter anderem eine vollautonome Navigation ermöglichen. Um dieses Potential zu erschließen wurde - als Ergänzung der bereits integrierten Sensorik - die Konzeptionierung und Entwicklung einer lokalen Bildverarbeitung durchgeführt. Zu diesem Zweck wurde eine Ausstattung des Roboters mit Farbkamera-Sensoren vorbereitet. Die Elektronik eines lokalen Bildverarbeitungssystems wurde entworfen und prototypisch aufgebaut.
Die vorgestellten Verfahren zur Objekterkennung bieten Anregungen für zukünftige Arbeiten.

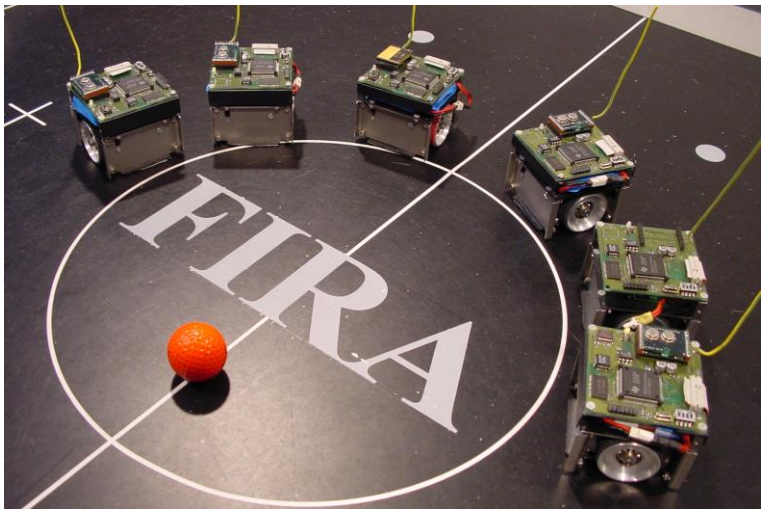


Abbildung 14.2: Zwei Teams der MiroSot-Fußballroboter

14.2 Zukünftige Arbeiten

Die Entwicklung eines Fußballroboters ist - bedingt durch die Verknüpfung mehrerer Wissenschaftsbereiche - ein transdisziplinäres Feld. Viele Bereiche konnten daher in dieser Arbeit nur grundlegend behandelt werden. Anregungen für zukünftige Arbeiten dürften in weiteren Verbesserungen der Fähigkeiten des Roboters und vor allem in der Erforschung und der Untersuchung

von Aufgabenstellungen angrenzender Wissenschaftsdisziplinen liegen.
In den folgenden Abschnitten werden Anregungen für zukünftige Arbeiten in den einzelnen Bereiche gegeben.

14.2.1 Navigation

Eine unterstützende Verwendung der beiden eingesetzten Sensortypen könnte zu einer guten approximierten Berechnung der Roboterposition in Verbindung mit der Detektion externer Ereignisse und somit potentieller Fehler für die lokale Positionsberechnung führen.
Die Integration weiterer Sensorik könnte eine autonomere Navigation des Roboters ermöglichen. In dieser Arbeit wurde die Integration eines Kompass-Sensors angeregt. Mit Hilfe dieser Sensorik wird eine exakte Bestimmung der absoluten Ausrichtung des Roboters realisierbar, die mit Hilfe der bisher vorgestellten relativen Methoden fehlerbehaftet ist.

14.2.2 Steuerung und Regelung

Die Implementierung eines verbesserten Steuerungskonzepts ist für den Spielbetrieb wünschenswert, um die gewünschte Maximalgeschwindigkeit zu erreichen und eine exaktere Steuerung zu erzielen. Es eignen sich unter anderem zwei vorgestellte Steuerungskonzepte, deren Implementierung und Verwendung als Softwaremodul möglich ist. Mit ihrer Hilfe ist sowohl das Folgen vorgegebener Trajektorien als auch das optimierte Anfahren bestimmter - sich unter Umständen bewegender - Punkte auf dem Spielfeld möglich.

14.2.3 Kommunikation

Eine Erweiterung des vorgestellten CSMA-Verfahrens um Teile des TDMA-Verfahrens (time division multiple access) kann die Einhaltung harter Echtzeitbedingungen garantieren. Zukünftige Arbeiten könnten untersuchen, ob derartige Echtzeitbedingungen für die Kommunikation von kooperativen Multiagentensystemen von Bedeutung sind und die Implementierung einer Mischform beider Verfahren - die in wireless-LAN-Systemen verwendet werden - durchführen.

14.2.4 Sensorik

Die Integration der Farbkamera-Sensoren bietet Ansatzpunkte für zukünftige Arbeiten: die vorliegende prototypische Schaltung sollte zu einer funktionsfähigen Version entwickelt werden. Anschließend können der Entwurf und die Implementierung eines bildverarbeitenden Systems und die Anbindung der Bildverarbeitung an das Hauptsystem des Roboters durchgeführt werden.

14.3 Zusammenfassung

Der entworfene Roboter erfüllt alle Konzeptionsziele. Er eignet sich für wissenschaftliche Arbeiten bezüglich kooperativer Multiagentensysteme und mobiler Roboter sowie für den Einsatz in der FIRA MiroSot-Roboterfußball-Liga. Gegenüber dem bisher verwendeten Fußballroboter weist er wesentliche Verbesserungen auf. Zum Abschluss dieser Arbeit stehen dem Lehrstuhl Informatik I der Universität Dortmund zwei Mannschaften der entwickelten Roboter zur Verfügung.
Der Roboter bietet viele Ansatzpunkte für zukünftige Arbeiten und kann durch die Realisierung der vorgeschlagenen Erweiterungen zum wahrscheinlich kleinsten vollautonomen Fußballroboter erweitert werden.

Abkürzungsverzeichnis

CCD
: Charge Coupeled Device

CD
: Carrier Detect

CMOS
: Complementary Metal-Oxide-Silicon

CSMA
: Carrier Sense Multiple Access

DSP
: Digital Signal Processor

ES
: eingebettetes System

FIRA
: Federation of International Robotsoccer Association

GPS
: Global Positioning System

LED
: Light Emitting Diode

MAS
: Multi Agent System

RTOS
: Real Time Operating System

PWM
: Pulse Width Modulation

QEP
: Quadrature Encoder Pulse

RoboCup
: Robot World Cup Initiative

TDMA
: Time Division Multiple Access

TTL
: Transistor-Transistor Logic

Literaturverzeichnis

[Ada01] R.A. Adams. On referencing RFCs in ITU-T Recommendations, checksums and CRCs for SCTP. Lucent Technologies, 2001.

[Ana99] Analog Devices. Low Cost $\pm 2g/\pm 10g$ Dual Axis Accelerometers with Digital Output. Analog Devices, Norwood, MA, USA, 1999.

[BG99] Thomas Bräunl and Birgit Graf. Autonomous Mobile Robots with On-Board Vision and Local Intelligence. The University of Western Australia, Perth, WA, Australia, 1999.

[BGG+99] Ansgar Bredenfeld, Wolf Gohring, Horst Gunther, Herbert Jaeger, Hans-Ulrich Kobialka, Paul-Gerhard Ploger, Peter Scholl, Andrea Siegberg, Arend Streit, Christian Verbeek, and Jorg Wilberg. Description of the GMD robocup 2000 team. In RoboCup, pages 711–714, 1999.

[Bra84] Valentino Braitenberg. Vehicles: Experiments in Synthetic Psychology. MIT Press, Cambridge, MA, USA, 1984.

[Bro83] R. Brockett. Asymtotic stability and feedback stabilization. In Differential Geometric Control Theory (R. Brockett, R. Millman, and H. Sussmann, eds.), vol. 27., pages 181–191, 1983.

[Bür01] Joachim Bürmann. Serial communication toolbox for cross plattform development, 2001.
http://www.iftools.com/.

[CLR94] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. Introduction to Algorithms. MIT Press, Cambridge, MA, USA, 11 edition, 1994.

[Deu96] P. Deutsch. RFC 1950, ZLIB Compresses Data Format Specification version 3.3, 1996.
http://www.alternic.org/rfcs/rfc1900/rfc1950.pdf.

[Dow96] Kevin Dowling. Robotics: comp.robotics frequently asked questions, 1996. Available as a hypertext document at
http://www.frc.ri.cmu.edu/robotics-faq.

[Dr.00a] Dr. Fritz Faulhaber GmbH & Co. KG. DC Kleinstmotoren, Serie 2224..SR. Schönaich, Germany, 2000. http://www.faulhaber.de/zip_pdf/produkte/kat_2000/2_kleinstmotoren/dM2224SR.pdf.

[Dr.00b] Dr. Fritz Faulhaber GmbH & Co. KG. Magnetische Impulsgeber, Serie IE2-16. Schönaich, Germany, 2000. http://www.faulhaber.de/zip_pdf/produkte/kat_2000/8_impulsgeber/dE16.pdf.

[Fai99] Fairchild Semiconductor Corporation. Octal Bidirectional Transceiver with 3-STATE Inputs/Outputs. South Portland, ME, USA, 1999. http://www.fairchildsemi.com/ds/74/74ACT245.pdf.

[FG97] Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In Jörg P. Müller, Michael J. Wooldridge, and Nicholas R. Jennings, editors, Proceedings of the ECAI'96 Workshop on Agent Theories, Architectures, and Languages: Intelligent Agents III, volume 1193, pages 21–36. Springer-Verlag: Heidelberg, Germany, 12–13 1997.

[FIR01a] FIRA. Fira small league mirosot rules, 2001. http://www.fira.net/fira/f2001/docs/games/mirosotSmall.html.

[FIR01b] FIRA. Fira website, 2001. http://www.fira.net/fira/f2001/index.html.

[Gra99] Birgit Graf. Robot soccer. Master's thesis, Universität Stuttgart, Fakultät Informatik, Stuttgart, Germany, 1999.

[Ind01] Giovanni Indiveri. On the motion control of a nonholonomic soccer playing robot. GMD - German Research Center for Information Technology, St. Augustin, Germany, 2001.

[Inf00] Infineon Technologies. Silicon Temperature Sensors. Munich, Germany, 2000. http://www.infineon.com/cmc_upload/0/000/012/047/kt_10_.pdf.

[JB96] L. Feng J. Borenstein, H.R. Everett. Where am I? Sensors and Methods for Mobile Robot Positioning. The University of Michigan, Ann Arbor, MI, USA, 1996.

[JF96] Joseph L. Jones and Anita M. Flynn. Mobile Roboter. Addison-Wesley, Bonn, Germany, 1996.

[K-T95] K-Team S.A. Khepera User Manual. Preverenges, Switzerland, 1995.

[K-T01] K-Team S.A. Khepera website, 2001.
http://www.k-team.com/robots/khepera/.

[Kie97] H. Kiendl. Fuzzy Control methodenorientiert. Oldenbourg Verlag, München, Germany, 1997.

[Kum99] J. Kummrneje. Robocup simulation league soccerserver manual v5.1, 1999.
http://www.dsv.su.se/ johank/RoboCup/manual/.

[Lat01] Lattice Semiconductor Corporation. GAL16V8 Data Sheet. Hillsboro, OR, USA, 2001.
http://www.latticesemi.com/.

[Lie01] Achim Liers. Roboter-elektronik 2001 - fu-fighters berlin, 2001.
http://www.inf.fu-berlin.de/r̃obocup/doc/elektronik2001.ppt.

[LSI99] LSI Computer Systems, Inc. Quadrature Clock Converter. Melville, NY, USA, 1999.
http://www.lsicsi.com/pdfs/ls708384.pdf.

[Mar00] Peter Marwedel. Begleitmaterial zur vorlesung prozessrechnertechnik / eingebettete systeme, wintersemester 2000/2001, 2000.

[MFI93] Francesco Mondada, Edoardo Franzi, and Paolo Ienne. Mobile robot miniaturisation: A tool for investigation in control algorithms. In Proc. Third Int. Symposium on Experimental Robotics, Kyoto, 1993.

[Mic01] Micro Adventure. Microadventure website, 2001.
http://www.robotkorea.co.kr/.

[Min85] Marvin Minski. The society of mind. Simon and Schuster, Ney York, NY, USA, 1985.

[MN99] Michael Mock and Edgar Nett. Real-Time Communication in Autonomous Robot Systems. GMD - German Research Center for Information Technology, St. Augustin, Germany, 1999.

[PG 00] PG 340, Lehrstuhl Informatik 1. Roboterfußball - better than the bavarians - Fußball spielende kooperative Multiagentensysteme. Endbericht der Prjektgruppe. Universität Dortmund, Germany, 2000.

[PG 01] PG 362, Lehrstuhl Informatik 1. Endbericht der Projektgruppe 362, Roboterfußball. Universiät Dortmund, Germany, 2001.

[Phi00] Philips Semiconductors. Electronic Compass Design using KMZ51 and KMZ52, Application Note AN00022. Hamburg, Germany, 2000.
http://www.semiconductors.philips.com/acrobat/applicationnotes/AN00022_compass.pdf.

[Rad01a] Radiometrix Ltd. 433MHz high speed radio transceiver module. Watford, England, 2001.
http://www.radiometrix.co.uk/dsheets/bim2.pdf.

[Rad01b] Radiometrix Ltd. Low Power UHF Data Transceiver Module. Watford, England, 2001.
http://www.radiometrix.co.uk/dsheets/bim.pdf.

[Rob01] Robot World Cup Initiative (RoboCup). Robocup website, 2001.
http://www.robocup.org/.

[Sam00] Samsung Electronic Co. Ltd. 256Kx16 Bit High Speed Static RAM(5V Operating). Seoul, Korea, 2000.
http://samsungelectronics.com/semiconductors/SRAM/High_Speed/Asynch_Fast/4M_bit/KM6164002C/KM6164002C.PDF.

[SGHP98] J. Stone, M. Greenwald, J. Hughes, and C. Partridge. Performance of checksums and CRCs over real data. IEEE Trans. on Networks, 1998.

[Sha49] C. E. Shannon. Communication in the presence of noise. Proceedings of the IRE, 37:10–21, 1949.

[ST 00a] ST Microelectronics. Dual Full-Bridge Driver. Saint Genis Pouilly, France, 2000.
http://www.st.com/stonline/books/pdf/docs/1773.pdf.

[ST 00b] ST Microelectronics, VLSI Vision Ltd. Mono and Colour QSIF Digital Video CMOS Image Sensors. Edinburgh, UK, 2000.
http://www.vvl.co.uk/products/image_sensors/301/cd5301_6301f-3-0.pdf.

[Sto98] Daniel Storey. Wireless Communication for Intelligent Robotic Agents. Department of Electrical and Electronic Engineering, The University of Western Australia, Australia, 1998.

[Tan96] Andrew S. Tanenbaum. Computer Networks. Prentice-Hall, Upper Saddle River, NJ, USA, third edition, 1996.

[Tex96a] Texas Instruments. TMS320C240 TMS320F240 DSP Controllers. Dallas, USA, 1996.
http://www-s.ti.com/sc/psheets/sprs042d/sprs042d.pdf.

[Tex96b] Texas Instruments. TMS320F206 Digital Signal Processor. Dallas, USA, 1996.
http://www-s.ti.com/sc/psheets/sprs050a/sprs050a.pdf.

[Tex99a] Texas Instruments. TMS320F240 DSP Controllers, Evaluation Module, Technical Reference. Dallas, USA, 1999.
http://www-s.ti.com/sc/psheets/spru248b/spru248b.pdf.

[Tex99b] Texas Instruments. TMS320F/C240 DSP Controllers Reference Guide. Dallas, USA, 1999.
http://www-s.ti.com/sc/psheets/spru161c/spru161c.pdf.

[Tex00] Texas Instruments. TRF5001 GPS RF Receiver. Dallas, USA, 2000.
http://www-s.ti.com/sc/psheets/slws084/slws084.pdf.

[Whi15] E. T. Whittaker. On the functions which are represented by the expansion of interpolating theory. Proceedings of the Royal Society of Edinburgh, 35, 1915.

[YK99] Jung-Min Yang and Jong-Hwan Kim. Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots. In IEEE Transactions on Robotics and Automation, pages 578–587, 1999.

[ZN42] J. G. Ziegler and N. B. Nichols. Optimum setting for automatic controllers. In Trans. ASME,Vol. 64, pages 759–768, 1942.

Anhang A

Schaltpläne

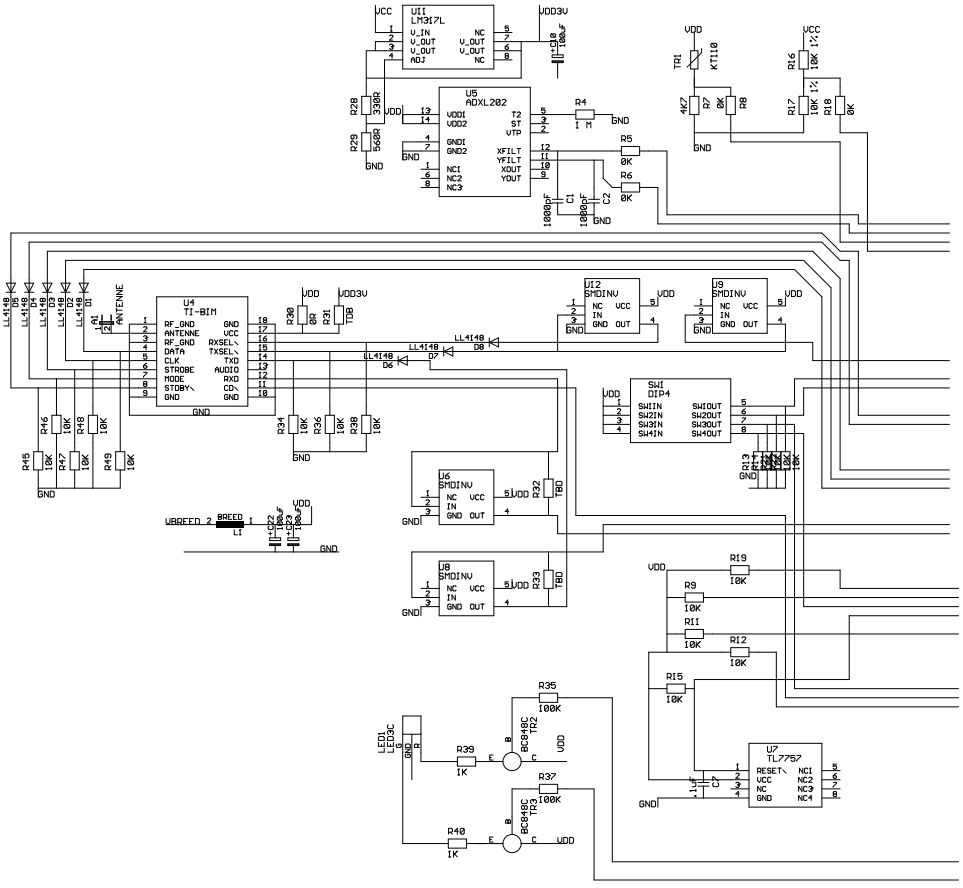


Abbildung A.1: Schaltplan des DSP-Systems (Teil 1)

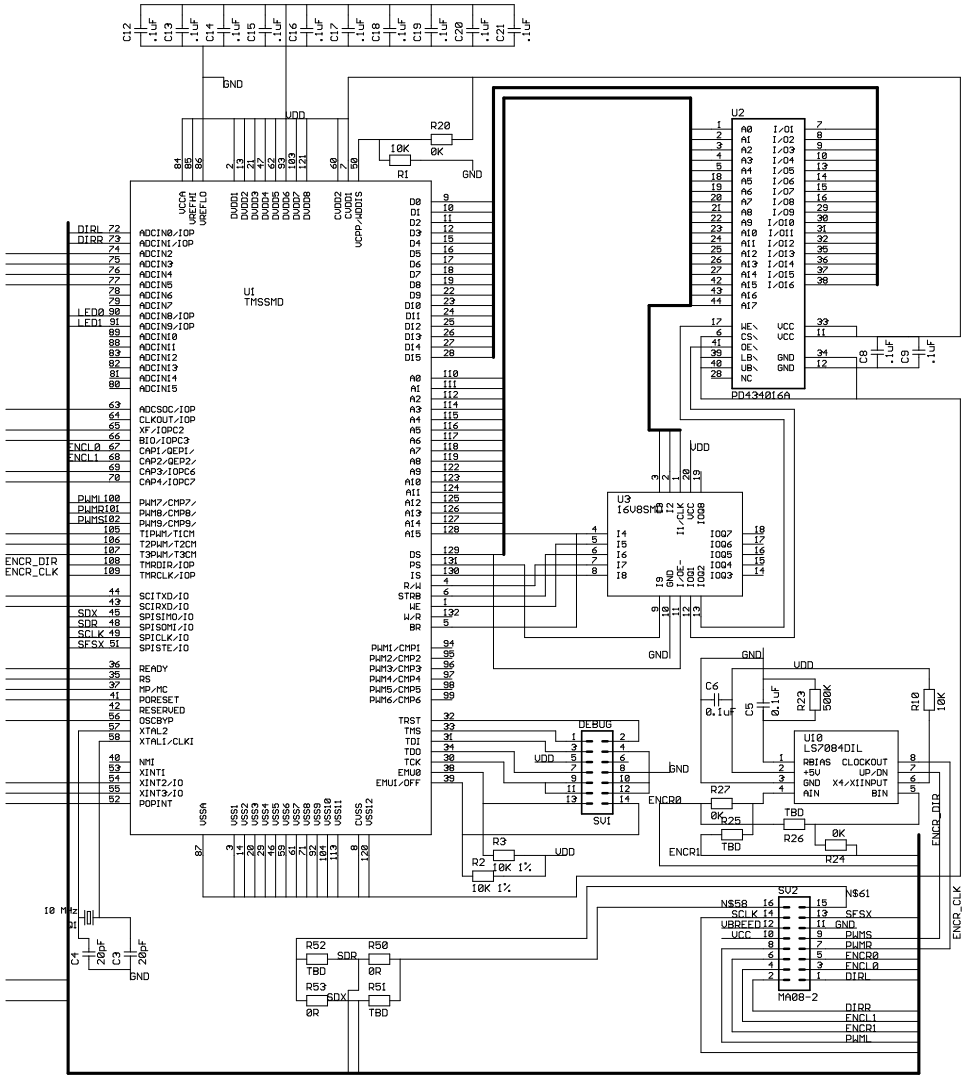


Abbildung A.2: Schaltplan des DSP-Systems (Teil 2)

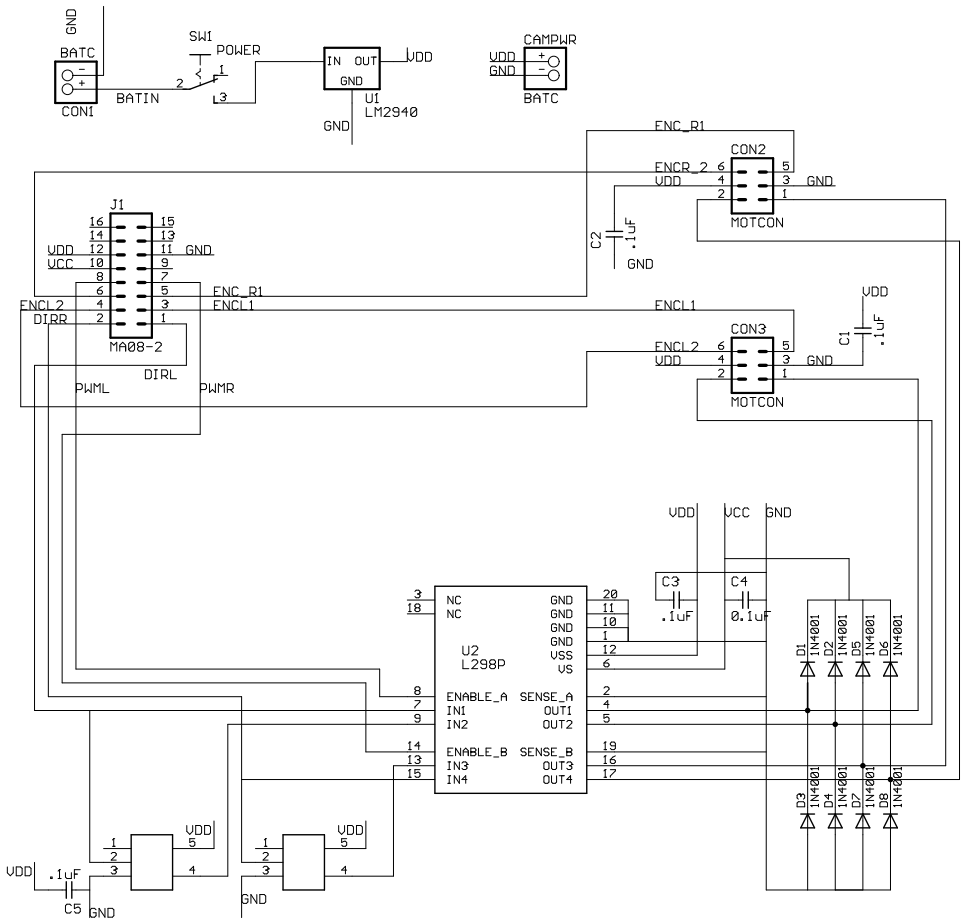


Abbildung A.3: Schaltplan der Leistungselektronik

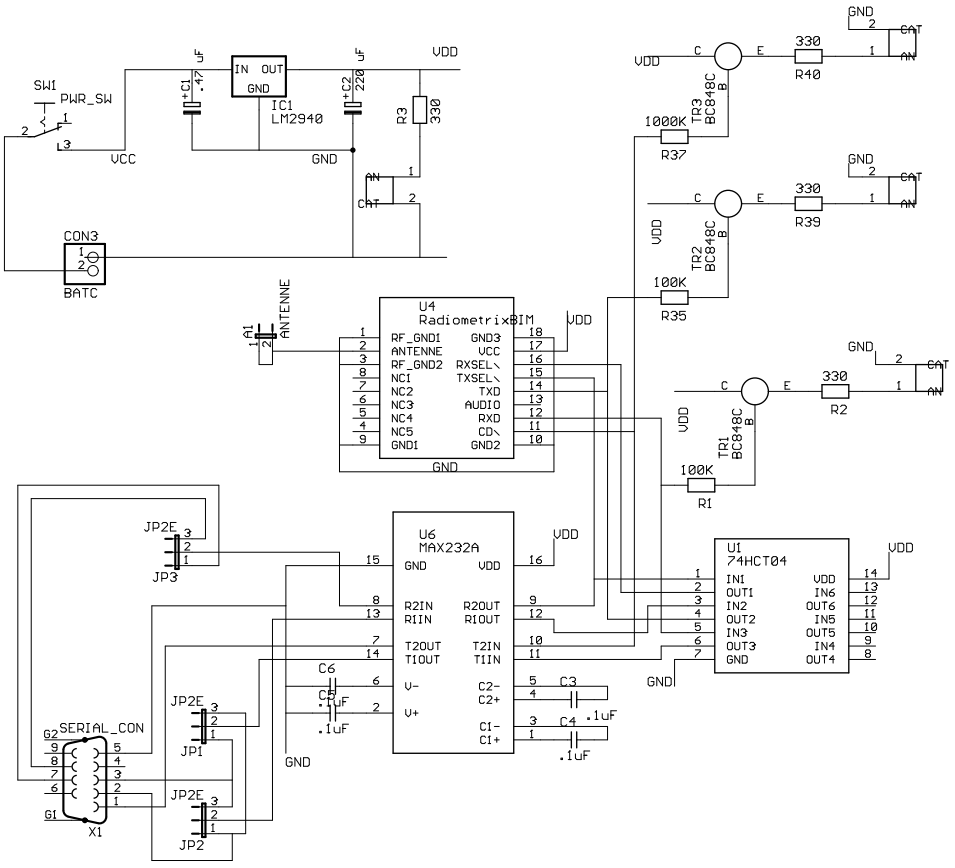


Abbildung A.4: Schaltplan des Host-Transceivers

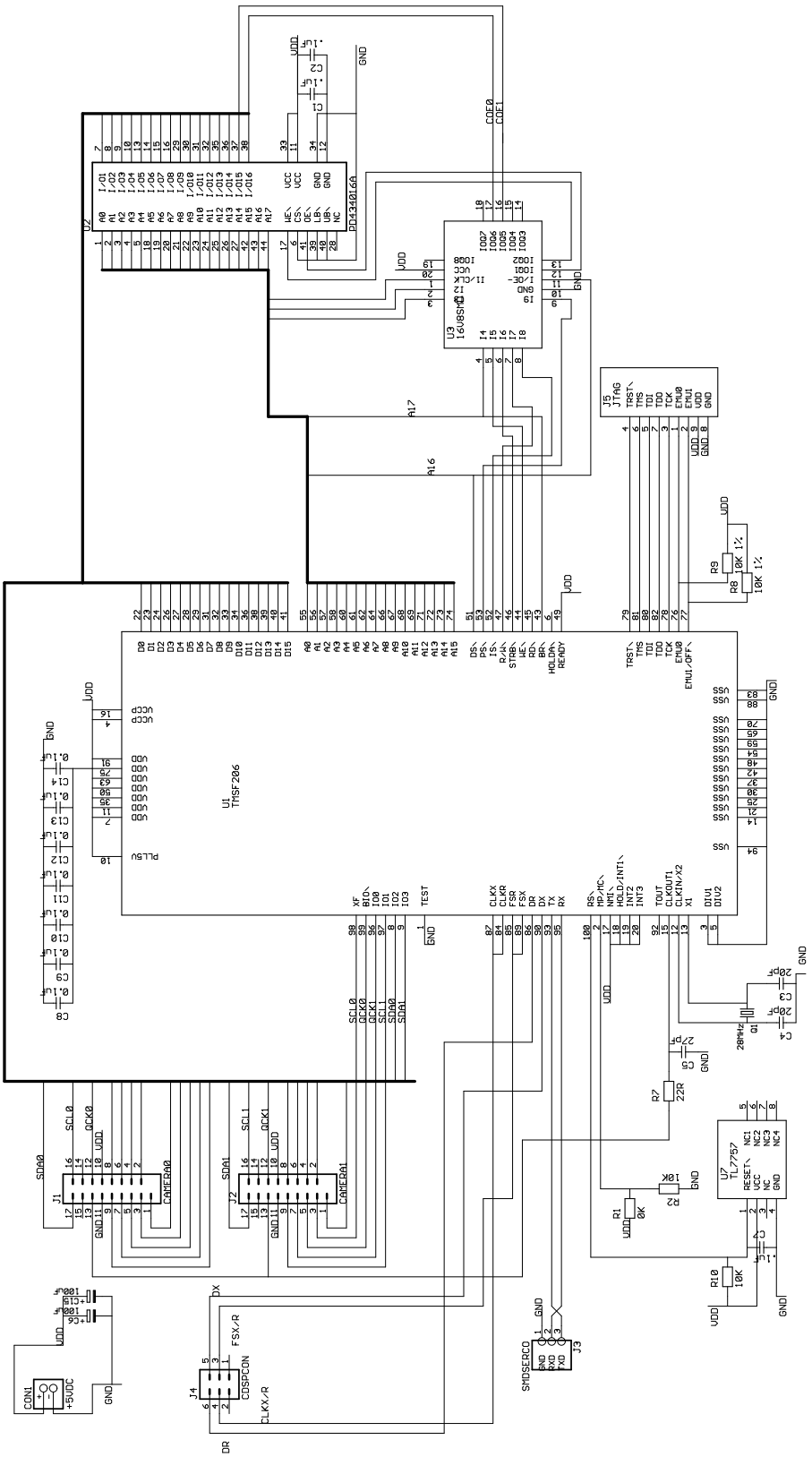


Abbildung A.5: Schaltplan des BV-Systems

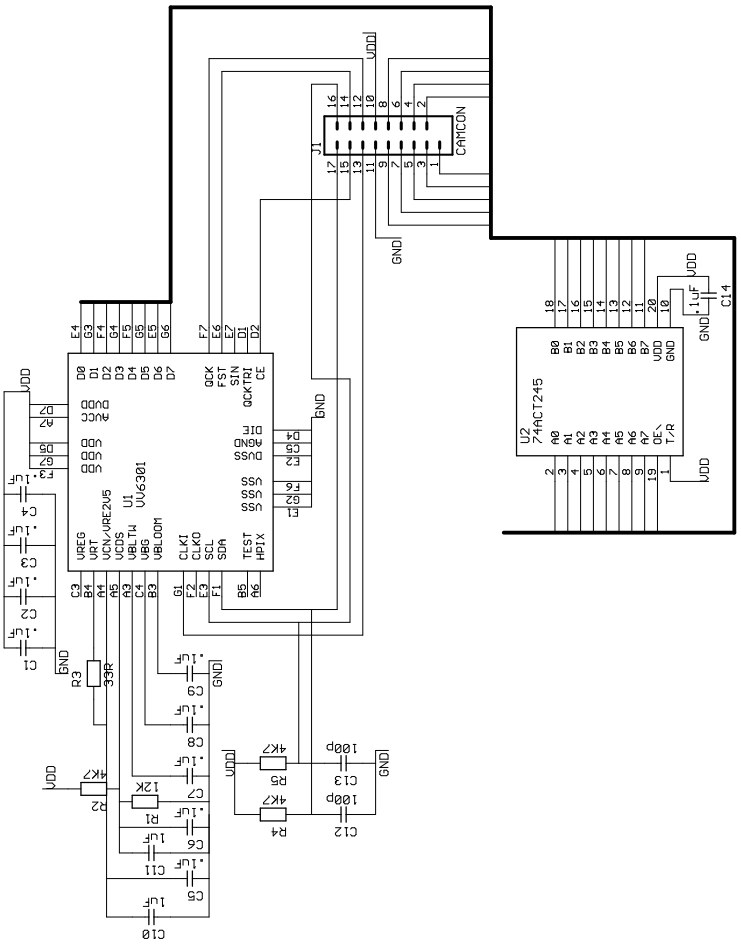


Abbildung A.6: Schaltplan des Kamera-Sensors

Anhang B

Platinenlayouts

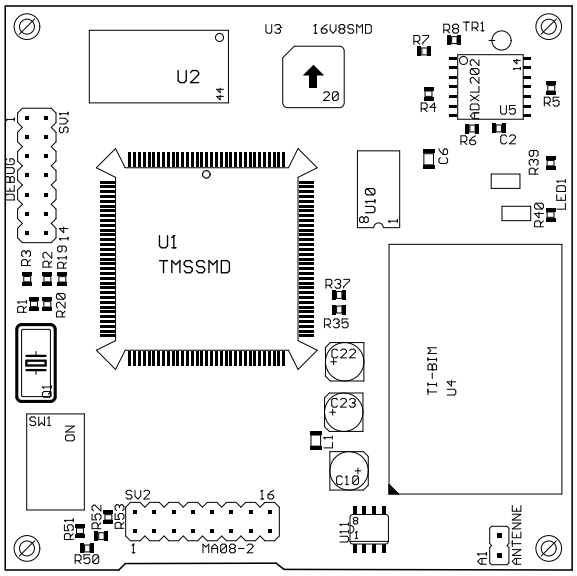


Abbildung B.1: Bauteillayout des DSP-Systems (Oberseite)

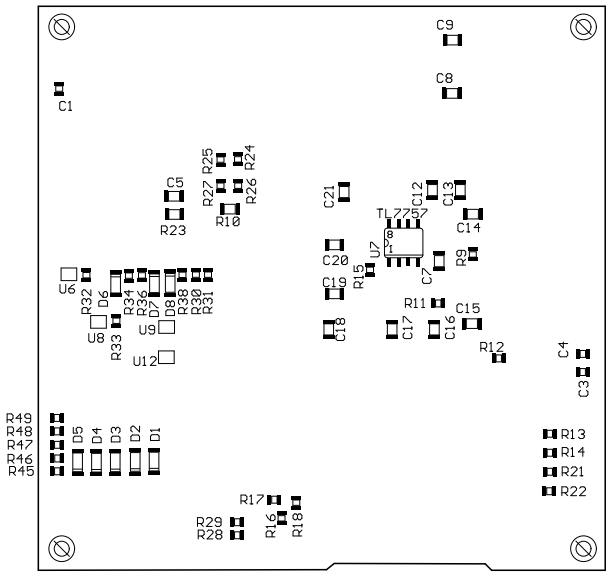


Abbildung B.2: Bauteillayout des DSP-Systems (Unterseite)

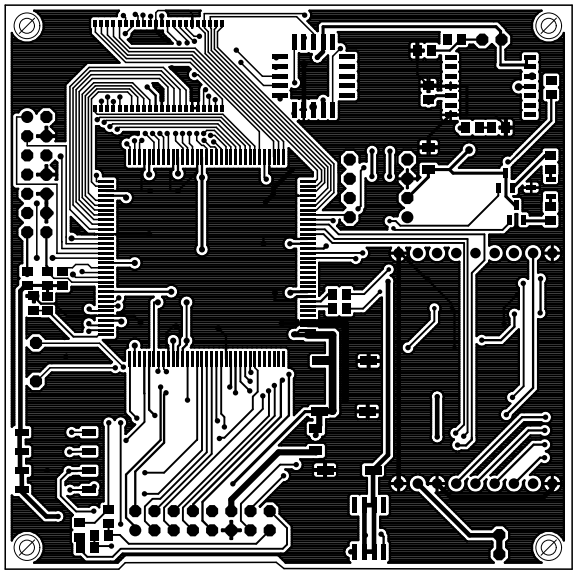


Abbildung B.3: Top-Layer des DSP-Systems

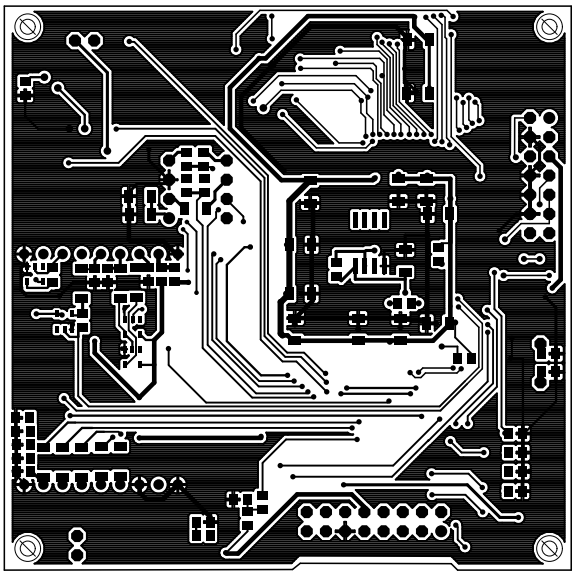


Abbildung B.4: Bottom-Layer des DSP-Systems

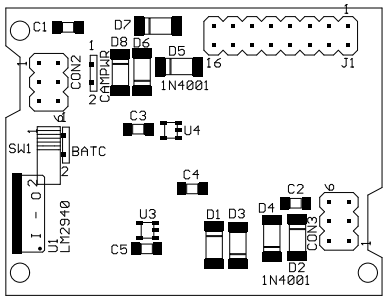


Abbildung B.5: Bauteillayout der Leistungselektronik (Oberseite)

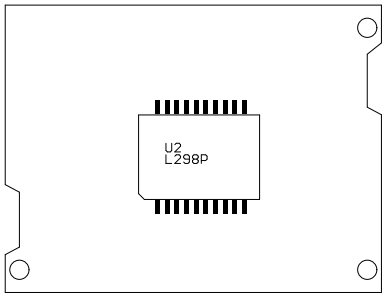


Abbildung B.6: Bauteillayout der Leistungselektronik (Unterseite)

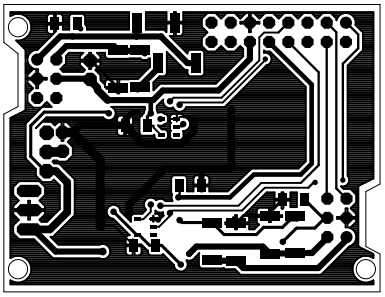


Abbildung B.7: Top-Layer der Leistungselektronik

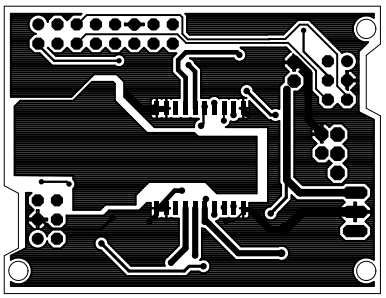


Abbildung B.8: Bottom-Layer der Leistungselektronik

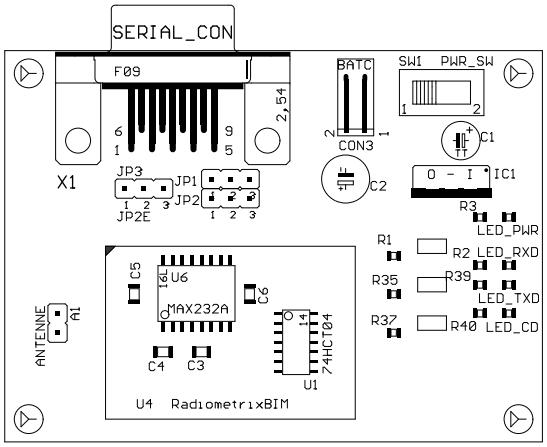


Abbildung B.9: Bauteillayout des Host-Transceivers

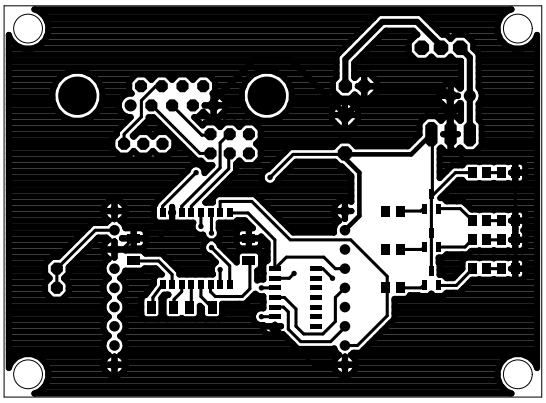


Abbildung B.10: Top-Layer des Host-Transceivers

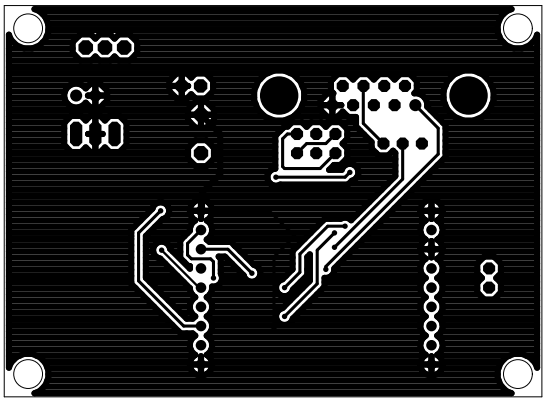


Abbildung B.11: Bottom-Layer des Host-Transceivers

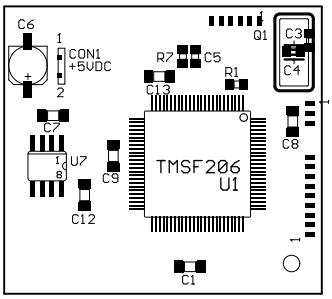


Abbildung B.12: Bauteillayout des BV-Systems (Oberseite)

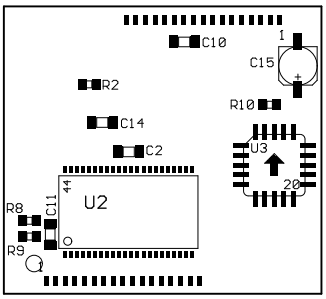


Abbildung B.13: Bauteillayout des BV-Systems (Unterseite)

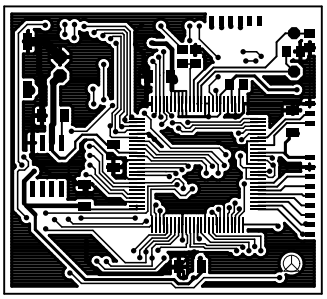


Abbildung B.14: Top-Layer des BV-Systems

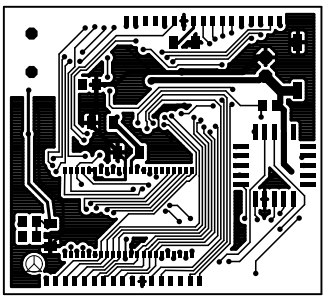


Abbildung B.15: Bottom-Layer des BV-Systems

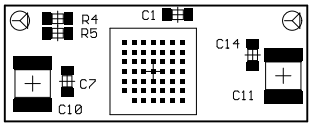


Abbildung B.16: Bauteillayout des Kamera-Sensors (Oberseite)

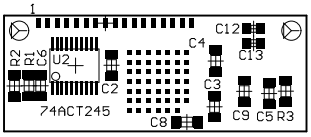


Abbildung B.17: Bauteillayout des Kamera-Sensors (Unterseite)

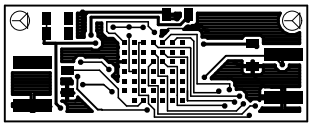


Abbildung B.18: Top-Layer des Kamera-Sensors

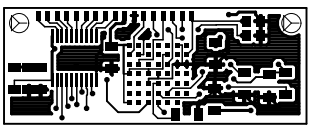


Abbildung B.19: Bottom-Layer des Kamera-Sensors

Anhang C

Beiliegende CD-Rom

Inhalt

Der Inhalt der beiliegenden CD-Rom ist in die folgenden Unterverzeichnisse gegliedert:

- src
  In diesem Ordner findet sich der Quellcode der in Arbeit vorgestellten Implementierungen.
  - host
    C-Quellcode der Kommunikationsschnittstelle des Hosts.
  - roboter
    C- und Assembler Quellcode für den Roboter. Enthalten sind die Quellen der prototypischen Implementierung in Verbindung mit den demonstrierten Fähigkeiten.
- Literatur
  In diesem Ordner finden sich Kopien der im Literaturverzeichnis angegebenen URLs vom Zeitpunkt der Erstellung dieser Arbeit.

Kontakt

Thomas Klute
Krähenbruch 12
44227 Dortmund

Telefon: +49 (0)231 / 752826
Email: `thomas@klute.com`

Index