Roboter Kunst

Ein Beitrag

Die Entwicklung einer Steuerlogik für Roboter der Künstlergruppe f18

Ein interdisziplinäres Entwicklungsprojekt

Studienarbeit

Arne Wischmann

1 EINLEITUNG 3

2 ROBOTER 3

2.1 STATIONÄRE ROBOTER 4
2.2 AUTONOME MOBILE ROBOTER 4
2.2.1 Kartenbasierte Systeme 4
2.2.2 Verhaltensbasierte Systeme 4
2.3 HUMANOIDE ROBOTER 5
2.4 ROBOT ART 5
2.5 DIE AUFGABE 6

3 SPEZIFIKATIONSPHASE 7

3.1 VORGEHENSMODELLE 7
3.2 PROJEKTBESCHREIBUNG 8
3.3 ANFANGSSPEZIFIKATION 9
3.3.1 Ultraschall Sensor 10
3.3.2 GPS – Empfänger 11
3.3.3 Der Kompaß 11

4 DAS DESIGN 12

4.1 VERHALTENSBASIERTES DESIGN 12
4.2 SUBSUMPTION ARCHITEKTUR 12
4.3 BESCHREIBUNG DES SYSTEMS 13

5 DIE REALISIERUNG 14

5.1 DIE SERIELLE KOMMUNIKATION 15
5.1.1 Das Serielle Interface des 6.270 Boards 15
5.1.2 Das Protokoll 16
5.2 DER ERSTE PLATTFORMPROTOTYP 17
5.3 DIE VERHALTEN 18

6 ZUSAMMENFASSUNG, KRITIK UND AUSBLICK 20

7 LITERATURLISTE 22

1 Einleitung

Diese Studienarbeit entstand im Rahmen eines Projektes der Künstlergruppe f18 in Zusammenarbeit mit dem Robotlab der Fachhochschule Hamburg und beschreibt die experimentelle Zusammenarbeit von Künstler und Informatikern, in dessen Mittelpunkt die Entwicklung dreier mobiler Roboter und ihrer Choreographie steht. Sie beschreibt die unterschiedliche Arbeitsweise der partizipierenden Parteien und deren Disziplin in Hinblick auf die Fertigstellung der Roboter und befaßt sich mit der Frage welchen Beitrag die Informatik in dem Bereich der Robotart leisten kann.

Das Institut für Information und Technologie f18 ist eine aus 5 Künstlern bestehende Gruppe, die sich in der Maschinenkunst verwirklichen will. Das Team des Robotlabs bestand aus 3 Studenten der Technischen Informatik, Rainer Balzerowski, Niels Dröge und mir. Wir hatten den Künstlern Unterstützung bei ihrem aktuellen Projekt, der Entwicklung und Konstruktion dreier mobiler autonomer Roboter, zugesagt.

Diese Studienarbeit beschreibt die Entstehung der drei Roboter für eine Promotion der Künstler von f18 in der Schweiz. Sie befaßt sich mit der Anbindung verschiedenster Sensorik und Aktorik an ein Microcontrollerboard unter Verwendung einer Seriellen Kommunikation sowie mit der Programmierung einfacher verhaltensbasierter Strukturen zur Realisation kollisionsfreier Fortbewegung.

Im Einzelnen skizziert Kapitel zwei eine Übersicht über verschiedene Robotertypen und deren Klassifikation, sowie eine grobe Beschreibung der sich stellenden Aufgabe. Kapitel 3 befaßt sich mit der Spezifikationsphase des Projektes in bezug auf die möglichen Vorgehensmodelle und beschreibt die zu Einsatz kommende Sensorik und Aktorik. Die Möglichkeiten des High Level Desigs, wie zum Beispiel die der Subsumption Architektur werden in Kapitel 4 umrissen sowie die abstrakte Beschreibung des Systems und seine Aufgabenverteilung. Die exakte Realisierung wird in Kapitel 5 mit Schwerpunkt auf die serielle Kommunikation und die globalen Verhalten beschrieben. Kapitel 6 schafft noch einmal einen Überblick über dieses Projekt, befaßt sich kritisch mit dem geleisteten und gibt einen Ausblick auf zukünftige Projekte in diesem Bereich.

Danken möchte ich an dieser Stelle Herrn Prof. Dr. Kai von Luck welcher den Kontakt zu der Künstlergruppe ermöglichte und natürlich den anderen beiden Teammitgliedern, insbesondere Rainer Balzerowski, dessen Engagement kaum zu übertreffen war.

2 Roboter

Im Allgemeinen sind Roboter[1] Maschinen, deren Handeln eine gewisse Selbständigkeit aufweist. Sie sind in der Lage ihre Aufgaben zu lösen, ohne vom Menschen überwacht oder gesteuert zu werden und dienen meist dem Zweck, dem Menschen komplexe oder gefährliche Aufgaben abzunehmen. Laut der Robotics Industry Association (RIA) „a robot is a re-programmable, multi-functional, manipulator designed to move material, parts, tools or specialised devices through variable programmed motions for the performance of a variety of tasks."

Nach [Knoll 00] lassen sich die Roboter in folgende Klassen einteilen:

[1] Der Begriff Roboter geht auf den tschechischen Schriftsteller Karel Capek zurück, der ihn 1921 von dem tschechischen Wort für Zwangsarbeit ableitete.

2.1 Stationäre Roboter

Hierbei handelt es sich um fest montierte Roboter (Arme) zur Erledigung stationärer Aufgaben wie zum Beispiel Schweissroboter in der PKW Fertigung. Hauptproblematik ist, neben der Planung kollisionsfreier Bewegungen des Armes, die Erkennung und Verfolgung der zu bearbeitenden Objekte sowie die exakte Bestimmung ihrer Lage. Die Interaktion des Roboters beschränkt sich allerdings auf das zu bearbeitende Objekt, da seine Umgebung aufgrund seines stationären Charakters vollständig deterministisch ist.

2.2 Autonome mobile Roboter

Als autonome mobile Roboter gelten alle Maschinen, die in der Lage sind sich frei in bekannten bzw. unbekannten Umgebungen zu bewegen. Das Spektrum der Mobilität erstreckt sich über alle bekannten Fortbewegungsarten. Bei allen stellt sich dasselbe Problem: Kollisionsfreie zielgerichtete Fortbewegung – Navigation.

> „An intelligent robot is a machine able to extract information from ist environment and use knowlegde about its world to move safely in a meaningful and purposive manner." [Arkin 98]

2.2.1 Kartenbasierte Systeme

Sie lassen sich in 2 unterschiedliche Gruppen einteilen: Solche, die eine vorgefertigte Karte ihrer Umgebung haben und solche, die ihre Umgebung erkunden und sich so eine Karte selbst anfertigen. Beide Systeme legen einen deliberativen Ansatz nahe, d.h. es sollte versucht werden das Wissen über die Umgebung logisch umzusetzen und aus ihr zu lernen, um so zum Beispiel nicht nur irgendeinen, sondern den kürzesten Weg von A nach B zu finden. Das größte Problem der kartenbasierten Roboter besteht darin die eigene Position innerhalb der Umgebung, also auch im Bereich der Karte zu bestimmen, denn nur wenn der Roboter weiß, wo er sich befindet, ist er in der Lage nach einer Karte zu navigieren. Häufig bewegt er sich zudem noch in einer dynamischen Umwelt, in der er auch plötzlich auftretende Hindernisse, die nicht in der Karte verzeichnet sind, wie z.B. Menschen, erkennen und umfahren können muß. Zu diesem Zweck verwendet man verschiedene Sensoren, die meist auf unterschiedlichen physikalischen Prinzipien wie Ultraschall, Laserscanner und Video basieren. Der Einsatz mehrerer verschiedener Sensoren zur Abbildung komplexer Umgebungen bedarf jedoch entsprechender Algorithmen zur Sensordatenfusion. Die Implementierung dieser Algorithmen auf den mitgeführten, häufig nicht so leistungsstarken Rechnern, unter Berücksichtigung einer zeitnahen Sensor Aktor Kopplung, erweist sich meist als schwierig.

2.2.2 Verhaltensbasierte Systeme

Verhaltensbasierte Roboter benötigen kein kartographisches Wissen über ihre Umwelt, sie verhalten sich rein reaktiv. Tritt ein bestimmtes Ereignis ein, so reagieren sie nach einem vorher für dieses Ereignis definierten Muster, unabhängig von ihrem Standort. Hieraus folgt, daß unvorhergesehene Situationen durchaus in eine Sackgasse führen können. Deshalb müssen alle möglichen Ereignisse dem System schon im Vorfeld bekannt sein, um möglichst sinnvolles Verhalten und geringe Fehleranfälligkeit zu modellieren. Somit sind diese Systeme ausschließlich für Umgebungen geeignet, die

den, in dem Verhalten des Roboters festgelegten Bedingungen genügen, also vollständig deterministisch sind. Verhaltensbasierte Roboter sind nicht in der Lage sinnvoll auf unvorhergesehene bzw. auf nicht im Verhaltensmuster festgelegte Ereignisse zu reagieren.

> „Simply put, reactive control is a technique for tightly coupling perception and action, typically in the context of motor behaviors, to produce timely robotic response in dynamic an unstructured worlds“ [Arkin 98]

2.3 Humanoide Roboter

In dieser Kategorie stellt sich nicht nur das Problem der Manipulation und Fortbewegung, sondern auch eine umfassende Implementation kognitiver Fähigkeiten, sowie die Integration in einen „Körper“. Diese Anforderungen machen humanoide Roboter zur „Königsdisziplin“ der Robotik, denn grade die Fähigkeiten, die einem Menschen grundsätzlich zur Verfügung stehen, wie das gehen auf zwei Beinen, sehen, hören und fühlen, stellen einen der komplexesten Bereiche der Robotik dar. Seine Fähigkeiten und sein Aussehen sollten dem Menschen möglichst ähnlich sein, um eine bruchlose Einbindung in die vorhandene Gesellschaft zu ermöglichen, doch davon ist man heute noch weit entfernt.

2.4 Robot Art

In dieses Bereich der Robotik gehören alle Maschinen, deren Bestimmung darin liegt Kunstwerke zu sein oder Kunstwerke zu erstellen. Es stehen nicht mehr Logische und sinnvolle Verhaltensweisen im Vordergrund, sondern Verhalten die dem Künstlerischen Anspruch seines Erschaffers, dem Künstler entsprechen. Die Bauformen dieser Roboter kennen keinerlei Regeln, da das Aussehen wie auch die Programmierung der Maschinen als Kunstwerk zu verstehen ist.
Ein bekannter Künstler auf diesem Gebiet ist Stellarc dessen Ansatz die sogenannte ‚Mensch Maschine Schnittstelle‘ also eine Symbiotische Verbindung zwischen Mensch und Maschine darstellt und auf die Ersetzung des menschlichen Körpers durch eine Maschine hinausläuft. Er selber ist daher immer Teil seiner Installationen.

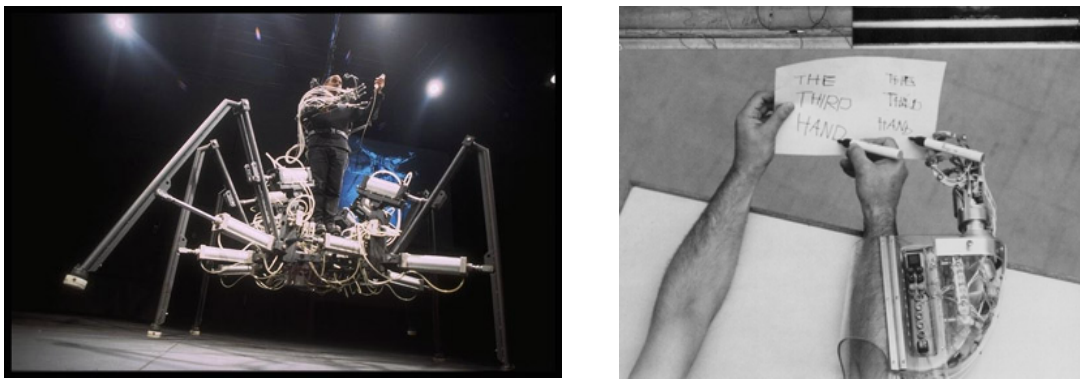


Bild 2-1 : Links: Stelarc auf dem Exoskeleton, Rechts: Stellarcs 'Third Hand'

Chico McMurtrie ist ebenfalls ein sehr bekannter Künstler auf dem Gebiet der Robot Art. Seine Künstlergruppe Armorphic Robot Works hat seit 1992 über hundert computer gesteuerte humanoide und abstrakte Maschinen gebaut und auf vielen performances ausgestellt. Sein Ziel ist die Schaffung einer Symbiose zwischen Kunst und Technologie. Er selber beschreibt seine Arbeit so: "The work is an ongoing endeavor to uncover the primacy of movement and sound. Each machine is inspired or influenced, both, by modern society, and what I physically experience and sense. The whole of this input informs my ideas and work." [McMurtrie00]

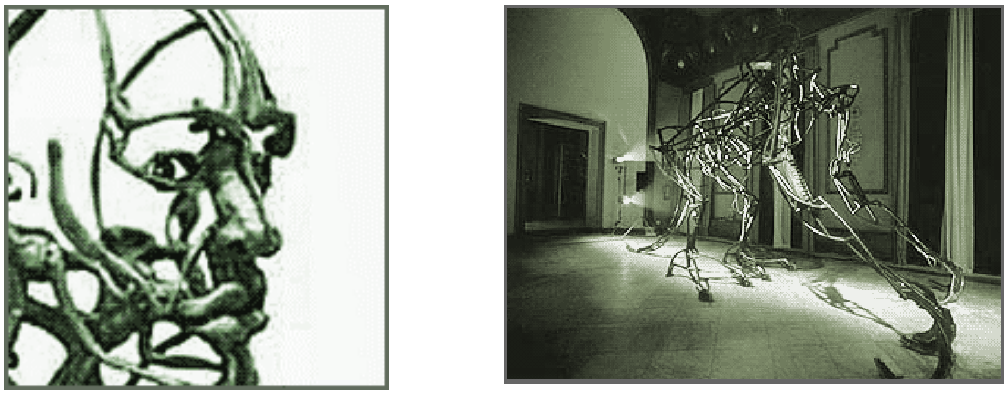


Bild 2-3 Skulpturen der Armorphic Robots Works

2.5 Die Aufgabe

Für eine Ausstellung in Zürich sollten drei unterschiedliche autonome mobile Roboter entstehen, die zusammen mit einem grobauflösenden Display und einer Serverstation als Gesamtkunstwerk im Park des Gottlieb Duttweiler Institutes installiert werden sollten. Die Roboter sollten in Form zweier fahrender Plattformen und einer 6 beinigen Laufmaschine realisiert werden, die sich innerhalb eines ‚Spielfeldes' autonom bewegten. Hierbei sollte dem Betrachter ein gewisses Maß an Interaktion geboten werden.

Hieraus ergab sich die Aufgabenstellung: Die Entwicklung einer Steuerung für die drei mobilen Roboter, zugeschnitten auf die Visionen der Künstler. Deshalb sollte die Entwicklung nicht unter rein technischen Gesichtspunkten geschehen, sondern hauptsächlich unter dem Gesichtspunkt Robotart.

3 Spezifikationsphase

Am Anfang eines jeden Software Entwicklungsprozesses steht die Frage , welches Vorgehensmodell sich für das jeweilige Projekt eignet. Vorgehensmodelle sind Handlungsschemata, die verschiedene Arten der Partizipation der beteiligten Parteien, meist Entwickler und Anwender, beschreiben und somit den groben Ablauf des Projektes und seiner Realisierung festlegen.

3.1 Vorgehensmodelle

Phasenmodelle: Sie teilen den Entwicklungsprozeß in benannte und standardisierte Entwicklungsschritte, die sequentiell durchlaufen werden. Der Rückgriff auf eine vorhergegangene Phase ist in den meisten Phasenmodellen erlaubt und in der Regel unausweichlich. Das bekannteste Modell dieser Art ist das lineare Phasen oder Wasserfallmodell. Seine Gültigkeit erstreckt sich jedoch hauptsächlich auf den logischen Ablauf der Softwareentwicklung, während die zeitliche Folge so nicht einzuhalten ist.

Spiralmodell: Dieses Modell trägt der zeitlichen Abfolge der Entwicklungschritte Rechnung. Der Entwicklungsprozeß wird nicht mehr als ein sequentieller, sondern als ein zyklischer Ablauf verstanden. Alle Phasen werden nacheinander mehrmals durchlaufen. Hierbei spielen Softwareprototypen eine entscheidende Rolle bei der Validation von Einsatz und Möglichkeiten. Am Ende dieser Entwicklung steht ein geschlossenes, wartbares Softwareprodukt.

Zyklische Modelle: In diesen Modellen wird Software nicht mehr als ein fertiges wartbares Produkt verstanden, sondern als eine Folge von Versionen. Wartung entfällt ganz, da sie durch den Übergang zu einer neueren Version abgebildet wird. Dieses Modell parallelisiert Einsatz und Entwicklung von Software. Deshalb modelliert es besonders gut die Entwickler – Anwender Interaktion. Im Gegensatz zum Spiralmodell erzeugt diese Vorgehensweise ein offenes Softwareprodukt, dessen Entwicklung theoretisch nie endet.

Evolutionäre Softwareentwicklung: Diese Vorgehensweise orientiert sich an dem evolutionären Charakter von Software und berücksichtigt besonders den gegenseitigen Lernprozeß zwischen Entwickler und Anwender. Durch den ständigen Dialog zwischen den Parteien ist es möglich Veränderungen im technischen wie auch im Einsatzkontext Rechnung zu tragen und gleichzeitig Anforderungen und Verbesserungen an der bereits im Einsatz befindlichen Software zu berücksichtigen. In der Praxis wird dieses Konzept meist durch Prototyping realisiert.

Prototyping: Bei diesem Verfahren werden Prototypen designed, die anschließend vom Anwender wie auch vom Entwickler bewertet werden. Sie schaffen praktische Erfahrung und eine Kommunikationsebene für beide Parteien, auf deren Basis Änderungen, Designalternativen und Erweiterungen diskutiert werden können. Man unterscheidet die verschiedenen Arten des Prototyping nach [Floyd00]:
Exploratives Prototyping soll helfen, die Problemstellung aus Anwendersicht zu klären.

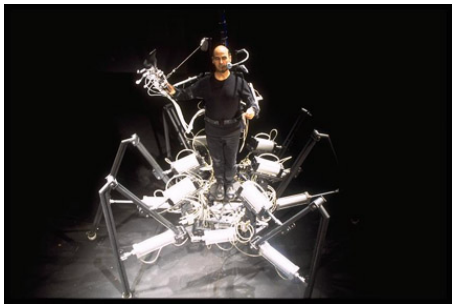
Experimentelles Prototyping unterstützt die konstruktive Umsetzung der Anforderungen an ein System
Evolutionäres Prototyping ist Teil eines kontinuierlichen Verfahrens in dem Anwendersoftware entwickelt und innerhalb einer Organisation an die sich ständig ändernden Randbedingungen angepaßt wird.

> *Prototyp:* Er ist eine lauffähige Version eines Softwareproduktes, das ausgewählte Teilbereiche des Gesamtproduktes realisiert.

Da in diesem Projekt nicht der Aspekt der Robotik, sondern der künstlerische Anspruch im Vordergrund stand kam nur ein solches Modell in Frage, welches die Künstler in einem möglichst hohen Maße an dem Entwicklungsprozeß der Software und den daraus entstehenden Möglichkeiten beteiligte. Aus diesem Grund fiel die Entscheidung auf das Modell der Evolutionären Softwareentwicklung. Dieses Modell fand nicht nur in der Entwicklung der Software Anwendung, sondern dehnte sich auf alle Bereiche, wie Sensorik und Aktordesign, innerhalb des Projektes aus. So entstand ein interdisziplinäres Team aus Künstlern und Informatikern, dessen Designentscheidungen immer im gegenseitigen Dialog stattfanden.

3.2 Projektbeschreibung

Die Künstlergruppe f18, die dieses Projekt ins Leben gerufen hatte, besteht aus 5 Künstlern, die alle unterschiedliche Fähigkeiten in ihre künstlerischen Projekte einbringen. Diverse vorangegangene Projekte im Bereich der Maschinenkunst wie z.B. ein steuerbarer 6 beiniger Walker, pneumatisch betrieben, mit einem Gesamtgewicht von ca. 0,5 Tonnen, zeugten von einem Team, das schon einige Erfahrung im Bereich des Plattformdesigns für mobile Maschinen hatte. Dieses Projekt sollte ihren Maschinen nun selbständiges Handeln ermöglichen, und war für sie eine neue Dimension auf dem Weg ihre Visionen von Robotart zu realisieren.
Das Team der FH-Hamburg, hatte im Rahmen des Robotlabs der Fachhochschule schon einige Erfahrungen in dem Bereich Robotik gesammelt. Diese erstreckten sich überwiegend auf den Bau und die Programmierung verhaltensbasierter kleiner Roboter, zur Erledigung einfach strukturierter Aufgaben innerhalb einer Kunstwelt. Die hier zum Einsatz kommenden Software Architekturen wie z.B. die Subsumption Architektur und Endliche Automaten waren dem Team ebenso vertraut, wie die Entwicklung einfacher Sensorik auf Infrarot- und Ultraschallbasis.
Dieses Projekt ermöglichte den Einsatz von vorhandenem know how, ließ aber auch genug Spielraum neue Techniken zu evaluieren, um sich somit neue Möglichkeiten im Rahmen der verhaltensbasierten Roboter zu erarbeiten.



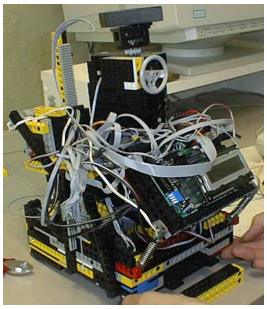


Bild 3-1: Links Exoskeleton – der 6 Beiner der Künstler. Rechts ein Lego Roboter aus dem FH Labor

Das Gesamtkunstwerk im Park des Gottlieb Duttweiler Institutes sollte aus einem Display, einer Serverstation und drei mobilen Robotern bestehen. Die Serverstation sollte als zentraler Steuerungs- und Kommunikationsrechner dienen, um eine globale Steuerung des Verhaltens und den Datenaustausch unter den Robotern zu realisieren. Zusätzlich war sie für die Ansteuerung des Displays gedacht, das die Bilddaten der Videokameras auf den Robotern wiedergeben sollte.
Die ersten Planungen zu diesem Projekt mit den Künstlern begannen Anfang Mai 2000, bis zur Ausstellung in der Schweiz waren es zu diesem Zeitpunkt noch gute 3 Monate. In dieser Zeit sollten alle Komponenten des Kunstwerkes entstehen und programmiert werden. Ein sehr knapp kalkulierter Zeitrahmen. Auch die zur Verfügung stehenden finanziellen Mittel waren sehr begrenzt, was sich besonders bei der sehr kostenintensiven Sensorbestückung der Roboter bemerkbar machte. Hier waren Kompromisse an der Tagesordnung. Ein weiterer Fixpunkt war das Microcontroller Board. Hierbei handelte es sich um das 6.270 Board des MIT (Massachusetts Institute of Technology) basierend auf einem Motorola 68HC11 Prozessor. Dieses Board hatte sich schon länger im Robotlab bewährt, und es stand eine recht komfortable Entwicklungsumgebung in C zur Verfügung.

3.3 Anfangsspezifikation

Begonnen wurde mit der Spezifikation der Plattformen, die selbstverständlich von den Künstlern durchgeführt wurde, da das Aussehen der Roboter ein wichtiger Aspekt des Kunstwerkes sein sollte. Eine Rücksprache erfolgte nur in Bezug auf die technische Realisierung. Hieraus ergab sich vorerst eine grobe Vorstellung von dem Aussehen und dem gewünschten Verhalten der Roboter. Weiterhin wurden die technischen Eckdaten der Maschinen festgelegt. Alle drei sollten bei einer Größe von circa 1m x 1m von zwei Elektromotoren angetrieben werden, und eine Spannungsversorgung von 24V erhalten. Schon jetzt wurde deutlich, daß die Realisierung des Walkers große maschinenbauliche Probleme aufwerfen würde, so daß die Künstler, die für den Bau der Plattformen verantwortlich waren, ihn in der Spezifikation durch eine weitere fahrende Plattform ersetzten.
Die Auswahl der Sensoren war ein ständiger Diskussionspunkt über den gesamten Zeitraum der Projektentwicklung. Jede Veränderung der Sensorbestückung eines Roboters verändert auch die Möglichkeiten in Bezug auf die Realisierung eines gewünschten Verhaltens. Da das Verhalten der Roboter einen entscheidenden Teil der Performance widerspiegelte, ergab sich für die Künstler immer wieder die Frage „ Was wäre wenn...? Allerdings ließ sich aus den beiden grundsätzlichen Forderungen nach Kollisionsfreier Fortbewegung und Einhaltung der Spielfeldgrenzen schon ein Teil der Sensorik ableiten.
Zur Kollisionserkennung sollten sogenannte Bumper zum Einsatz kommen. Hierbei handelt es sich meist um eine Art beweglich Stoßstange, die bei der Kollision mit einem Objekt einen Taster oder ähnliches bedient. Da es bei autonomen Robotern auch zu Fehlverhalten und Fehlinterpretationen der Umgebung kommen kann, müssen Bumper zum Schutz von Personen und Gegenständen vorgesehen werden.
Für die Kollisionsvermeidung und Detektierung den näheren Umgebung kommen generell mehrere Sensortypen wie Videokameras, Laserscanner oder Ultraschall in Frage, die ersten beiden Möglichkeiten schieden aber aufgrund des schmalen Budgets

von vorn herein aus. Die dritte Möglichkeit, der Ultraschall Sensor, war die Einzige die im finanziellen Rahmen lag.
Die Erkennung der Spielfeldgrenzen ließ sich, wenn man von der Möglichkeit einer Bande am Spielfeldrand absieht, nur über einen GPS Empfänger (Global Positioning System) elegant realisieren. Die Roboter wären so jederzeit in der Lage ihre Position zu ermitteln, was auch für koordiniertes Verhalten von Vorteil wäre.
Angeregt von der Idee die Roboter mit Hilfe des GPS innerhalb des Spielfeldes exakt Positionieren zu können, äußerten die Künstler den Wunsch nach einer Art ‚Roboter Ballett'. Genauer gesagt einem Verhalten, bei dem die Roboter sich an verschiedenen Orten aufstellen und synchron unterschiedliche Figuren abfahren. Da solch ein Verhalten eine genaue Bestimmung des zurückgelegten Weges, wie auch der Himmelsrichtung erfordert, wurden noch ein elektronischer Kompaß und Shaftencoder in das Konzept aufgenommen.

3.3.1 Ultraschall Sensor

Ultraschall Sensoren werden zur Entfernungsmessung benutzt. Ihre Arbeitsweise beruht auf der Tatsache, daß der Schall sich in der Luft mit einer Geschwindigkeit von ca. 338.8 m/s fortpflanzt. Mißt man die Zeit, die zwischen dem Senden eines Tones und dem Empfang seines Echos vergeht, so kann man daraus die Entfernung ermitteln.
Mit Hilfe von Ultraschall Sensoren wird genau diese Zeit ermittelt. Bei einem Abstand von 10 cm, zum Beispiel, ist die Entfernung, die der Schall zurücklegen muß bis er durch Reflexion wieder beim Sender ankommt, genau der doppelte Abstand, also 20 cm. Die dafür benötigte Zeit beträgt also 590 µs.
Der einzige Sensor der aufgrund seiner Verfügbarkeit und seines Preises für dieses Projekt in Frage kam war ein Modell der Firma Polaroid bestehend aus einem Transducer der 7000 Serie und einem Ranging Modul der 6500 Serie. Bei dem Transducer handelt es sich um einen elektrostatischen Ultraschall Sender und Empfänger für eine Frequenz von 50kHz. Das Ranging Modul wird zur Erzeugung der 50 kHz benötigt wie auch zum Auswerten des Echos. Die Zeitmessung muß allerdings extern erfolgen. Um auch mit einem dieser Sensoren eine ausreichende Erfassung der Umgebung zu ermöglichen wurde der Transducer auf einem Servomotor montiert, so daß ein 180° Scan der Umgebung durchgeführt werden konnte

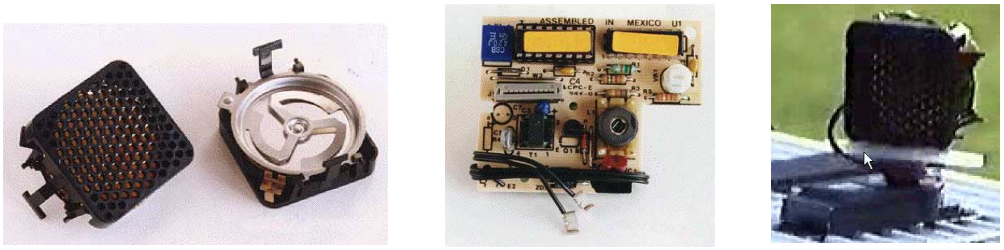


Bild 3-2: Zeigt von links nach rechts den Transducer, das Ranging Modul und den auf dem Servo montierten Transducer

3.3.2 GPS – Empfänger

Der GPS Empfänger arbeitet nach dem Gleichen Wirkungsprinzip wie der Ultraschall Sensor, der Laufzeitmessung eines Signals. Bei GPS handelt es sich jedoch nicht um Schall, sondern um ein Funksignal, ausgesandt von 24 verschiedenen Satelliten, dessen Laufzeit gemessen wird. Die Umlaufbahnen der Satelliten sind so aufeinander abgestimmt, daß von jedem Ort auf der Erde zu 95% der Zeit vier von ihnen zu sehen sind. Jeder Satellit sendet ständig seine eigene Position und Koordinaten sowie die Koordinaten der Anderen (almanac). Weiterhin sendet er eine OK-Meldung für die Satelliten (health bit) und die genaue Atomzeit. Diese Informationen werden zu einem Signal mit strengen Zeitcharakteristika verschlüsselt.

Wenn vier Satelliten sichtbar sind, lassen sich über die Laufzeit alle vier Variablen genau bestimmen: X (Länge), Y (Breite), Z (Höhe) und t (genaue Zeit).
Für das Projekt kamen nur GPS Empfänger in Frage, die eine serielle Schnittstelle haben, da die Positionsdaten weiterverarbeitet werden mußten. Der GPS 12 von Garmin erfüllte diese Voraussetzungen und war darüber hinaus in der Lage Differential GPS Daten zu verarbeiten. Weiterhin verfügt er über ein 12 Kanal Parallelempfänger, der für eine schnelle Akquisition und stabilen Empfang sorgt.

Bild 3-3: Der GPS 12 von Garmin

3.3.3 Der Kompaß

Die Funktion eines Kompasses braucht hier wohl nicht weiter erläutert zu werden. Allerdings weicht der Kompaßsensor V2X stark von der normalen Vorstellung eines Kompasses ab. Es handelt sich hierbei um einen Elektronischen Sensor der aus zwei in einem 90° Winkel zueinander angeordnete Spulen besteht. Während eine Spule die Feldstärke in X-Richtung mißt, ermittelt die andere die Feldstärke in Y-Richtung. Ein Mikroprozessor berechnet aus den beiden Meßgrößen die exakte Gradzahl. Er verfügt über ein serielles Interface, so daß die genaue Gradzahl direkt ausgelesen werden kann. Magnetische Störgrößen können durch Programmierung kompensiert werden.

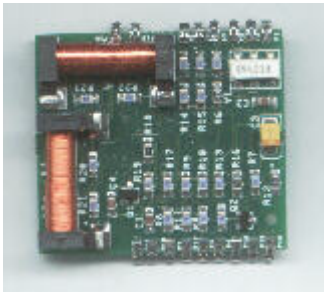


Bild 3-4: Der Kompaßsensor V2X von PNI

4 Das Design

Bei der Wahl des Designs gab es mehrere entscheidende Faktoren. Im Vordergrund standen die Wünsche der Künstler, eingegrenzt durch die Randbedingungen Zeit und Leistungsfähigkeit der eingesetzten Komponenten. So schied ein deliberativer Ansatz von vornherein aufgrund aller drei Faktoren aus. Ein verhaltensbasiertes System kam den Erwartungen der Künstler am nächsten und schien auch unter den gegebenen Umständen realisierbar.

4.1 Verhaltensbasiertes Design

Was ein Verhaltensbasierter Roboter ist wurde in Kapitel 2.2.2 eingehend erläutert, doch was genau ist ein Verhalten? Der Bereich der Robot Programmierung unterscheidet sich elementar von der Programmierung eines normalen Computers. Ein Computer verhält sich vollständig deterministisch. Alle zu verarbeitenden Daten liegen in digitaler Form mit beliebiger Genauigkeit vor. So ist es möglich präzise Anforderungen an die zu verarbeitenden Daten zu stellen, sowie das Ergebnis der Berechnungen mit 100% Sicherheit zu propagieren, und das ohne einen Gedanken daran zu verschwenden, daß das Ergebnis vielleicht Falsch sein könnte.
Bei der Robot Programmierung würde ein entsprechendes Handeln allerdings schnell in eine Sackgasse führen. Ein Roboter ist ein dynamisches System, welches sich in einer dynamischen Umwelt bewegt. Alle zu verarbeitenden Daten werden von Sensoren ermittelt, deren Genauigkeit zwar sehr hoch sein kann, aber dennoch immer einem Messfehler unterliegt. Auch die Ausführung von Aktionen erzeugt, aufgrund der Umsetzung in mechanische Bewegungen durch Aktoren, ebenfalls einen Fehler. Aufgrund der Tatsache, daß diese Fehler sich summieren, kann nie davon ausgegangen werden, daß Aktionen genau zu dem gewünschten Ergebnis führen.
Genau diesem Problem versucht man durch verhaltensbasierte Programmierung zu begegnen. Während der Ausführung einer Aktion werden alle eingehenden Sensordaten dahingehend überprüft, ob die Tätigkeit erfolgreich verläuft. Ist dies nicht der Fall wird versucht durch geeignete Gegenmaßnahmen die Aktion erfolgreich zu Beenden. Dieses „Stimulus – Response“ Verhalten bezeichnet man als Verhaltensbasiert.

4.2 Subsumption Architektur

Hierunter versteht man eine Methodik bei der mehrere Verhalten miteinander gekoppelt werden und somit gemeinsam das Gesamtverhalten bestimmen. Sie werden in aufeinander aufbauenden Schichten angeordnet, um beim hinzufügen einer höheren Ebene die darunterliegenden (fast) nicht verändern zu müssen. Jede Ebene hat direkten zugriff auf Sensoren und Aktoren, wodurch eine quasiparallele Architektur ermöglicht wird. [Knoll 00]. Die Ausgangssignale einer Ebene können von der Darüberliegenden entweder überschrieben oder unterbunden werden, höhere Ebenen können niedrigere Ebenen subsumieren, daher die Bezeichnug dieser Architektur. Ein Beispiel: Ein Roboter soll auf eine Lichtquelle zu fahren und etwaigen Hindernissen auf seinem Weg ausweichen. Dies ließe sich am einfachsten durch zwei Verhalten *Fahre zum Licht* und *Vermeide Kollision* realisieren. Steht nun ein Hindernis im Weg kann beim Ausweichen eventuell nicht mehr auf das Licht zugefahren werden und die beiden Verhalten kämen in Konflikt. Da es höchstwahrscheinlich wichtiger ist dem

Hindernis auszuweichen als weiter auf das Licht zuzufahren, würde man das Verhalten *Vermeide Kollisionen* eine Ebene höher ansiedeln, als das Verhalten *Fahre zum Licht*. Dies hätte zur Folge, daß im Falle eines Hindernisses die Ausgangskanäle des Verhaltens *Fahre zum Licht* von denen des *Vermeide Kollisionen* Verhaltens überschrieben werden. Nach erfolgtem Ausweichmanöver gibt *Vermeide Kollisionen* die Ausgangskanäle wieder frei und *Fahre zum Licht* kann fortgesetzt werden. Der Roboter würde jetzt erst einmal dem Hindernis ausweichen und anschließen wieder auf die Lichtquelle zufahren.

4.3 Beschreibung des Systems

Der strukturelle Aufbau der Robotersteuerung erfolgte prinzipiell nach einer Art Schichtenmodell. Jede Schicht ist auf einer anderen Ebene der Komplexität an der Fortbewegung des Roboters beteiligt.
Die unterste Schicht dieses Modells wird durch die Roboterhardware realisiert. Sie besteht aus der Plattform an sich, bestückt mit Aktoren (Motoren) und einer Vielzahl von Sensoren zur Detektierung der näheren Umgebung, wie Ultraschall, GPS und Kompaß. In dieser Schicht werden Keinerlei Daten verarbeitet, sie ist eine reine Exekutive.
Die darüberliegende Schicht, realisiert durch einen Atmel Microcontroller, hat die Aufgabe der Low Level Datenverarbeitung. Hier werden die Sensordaten gesammelt, um sie der darüber liegenden Schicht in einer einheitlichen Form zur Verfügung zu stellen. Weiterhin ist sie in der Lage Kommandos der darüberliegenden Schicht entgegen zu nehmen und an die entsprechenden Aktoren der nächst tieferen Ebene weiterzuleiten.
Die Oberste Schicht bildet das 6.270 Board. Hier erfolgte die eigentliche Verarbeitung der von der Schicht 2 zur Verfügung gestellten Daten, dem Stimulus. Alle High Level Strukturen zu Erzeugung eine sinnvollen Response, wie Verhalten und Endliche Automaten, wurden hier implementiert. Diese Schicht bildet sozusagen das „Gehirn“ des Roboters.

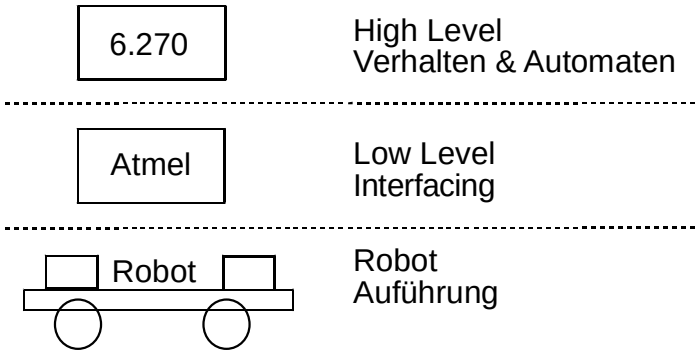


Bild 4-1 Schichtenmodell der Roboter

5 Die Realisierung

Begonnen wurde mit der Entwicklung des Interfaces, um die Sensoren an das 6.270 Board anzuschließen. Das Board hat zwar diverse Anschlußmöglichkeiten für analoge und digitale Sensorik sowie mehrere Motorports zur Ansteuerung von Motoren, leider eignet sich keiner der Eingänge zur Realisation eines seriellen Protokolls, noch sind die Motorports für Motoren dieser Größe geeignet. An diesem Punkt gab es große Unterstützung von Lars Vaupel, Ingenieur der Elektrotechnik und Mitglied der Künstlergruppe. Dieser hatte im Vorfeld schon viele Erfahrung mit einem Atmel Controller der 8515 Serie gemacht und bot an mit diesem ein Interface für die Sensoren zu entwickeln. Der erste Prototyp den er vorstellte (Bild 4-1) war in der Lage den Ultraschallsensor auszuwerten und das Servo auf dem der Sensor montiert war anzusteuern. Auch die GPS – Daten konnten ausgelesen werden. Alle ermittelten Daten wurden serialisiert, über eine RS232 Schnittstelle zu einem Laptop übertragen und dort zur Anzeige gebracht.

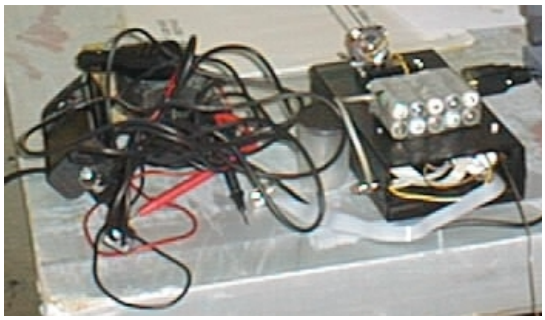


Bild 5-1 : Der erste Prototyp

Aufgrund der Erfahrung die Lars mit der Herstellung des ersten Prototypen gesammelt hatte, sagte er zu ein vollständiges Interface zu entwickeln. Somit übernahm er auch die Entwicklung des Motorinterfaces sowie das Auslesen des Kompasses und der Shaftencoder. Die einzigen Sensoren, die jetzt noch direkt mit dem 6.270 Board verbunden werden sollten, waren die Bumper. Diese ließen sich direkt an die digitalen Eingänge des Boards anschließen. Dies hatte den Vorteil, daß im Falle einer Kollision extrem schnell reagiert werden konnte, da das Ereignis nicht erst seriell übertragen werden mußte.
Über den Zeitraum der Entwicklung ergab sich nach und nach eine vollständige Systembeschreibung, die einen realisierbaren Eindruck machte. Deutlich zeigt sich hier die Unterteilung des Gesamtsystems in zwei Hauptkomponenten, das 6.270 Board (Main Controller) und den Atmel (Peripherie Controller). Der Atmel AT90S8515 Microcontroller bildet ein zentrales Interface für alle Komponenten des Roboters. Er sammelt die Daten der Sensoren, bereitet sie auf und überträgt sie auf Anfrage zum 6.270 Board. Außerdem kann er verschiedene Kommandos interpretieren und damit Sensorik und Aktorik ansteuern. Die Programmierung erfolgte in Assembler, da kein Compiler für eine Hochsprache zur Verfügung stand und die Realisierung einfacher Strukturen auch problemlos in Assembler möglich ist.

Das 6.270 Board übernimmt die gesamte Steuerung des Roboters. Alle Kontrollstrukturen, die für das Verhalten der Maschinen verantwortlich sind, wurden hier implementiert, da es für dieses Board einen C-Interpreter gibt und der Einsatz einer Hochspache ein unbedingtes Muß für die Realisierung komplexer Strukturen ist.

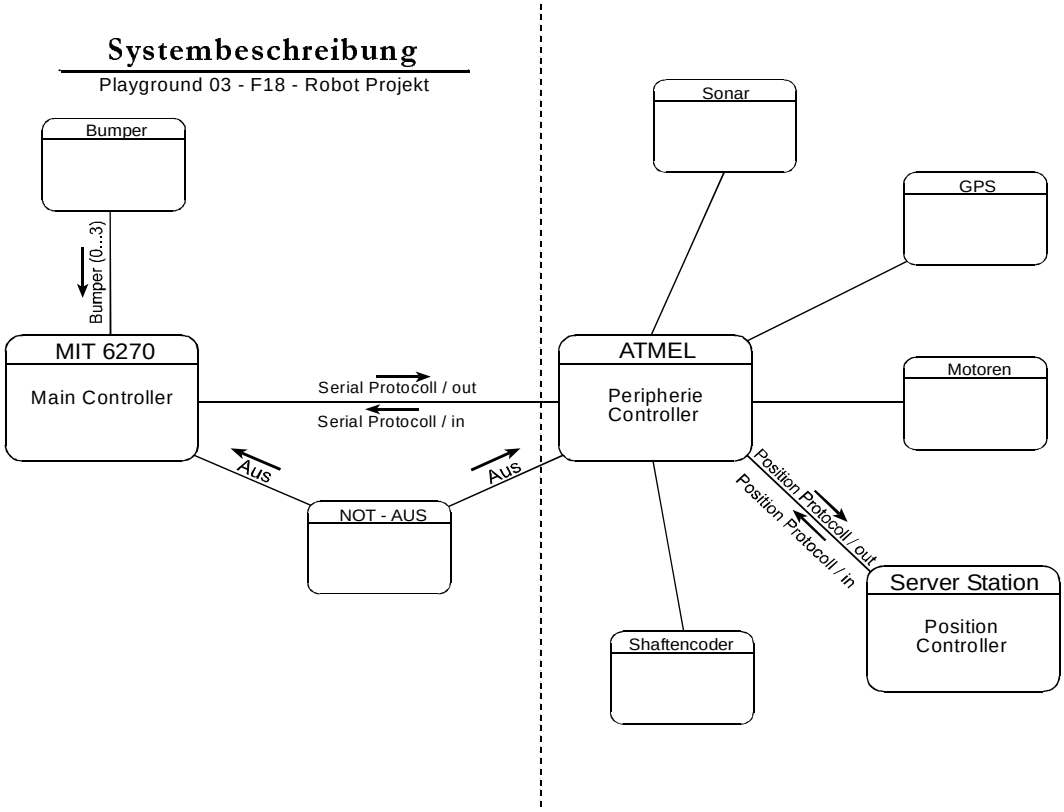


Bild 5-2 : Systembeschreibung des Projektes

5.1 Die Serielle Kommunikation

5.1.1 Das Serielle Interface des 6.270 Boards

Die einzige sinnvolle Möglichkeit das 6.270 Board mit dem Sensorinterface zu verbinden bestand in der Realisierung einer seriellen Kommunikation. Das 6.270 Board stellt nur eine Serielle Schnittstelle zur Verfügung, und diese ist normalerweise der Programmierung des Boards vorbehalten. Es handelt sich hierbei um eine RS 232 kompatible Schnittstelle, mit Rx Data und Tx Data ohne Handshake, und ermöglicht so eine Drei Draht Kopplung. Da diese Schnittstelle ebenfalls dazu gedacht ist interaktive Kommandos aus der Entwicklungsumgebung zu empfangen kann sie nicht einfach zur seriellen Übertragung von User Daten genutzt werden. Nach einigem Suchen aber fanden sich im Internet ein paar kleine Prozeduren, mit deren Hilfe sich ein serieller Datenaustausch realisieren lassen sollte. Die ersten beiden Prozeduren sind für das Umschalten zwischen User Mode und Programm Mode. Realisiert wurden diesen Funktionen durch einen einzigen POKE Befehl, der an die Speicherstelle 0x3c eine ‚0' oder eine ‚1' schreibt. Was im Einzelnen durch diesen Befehl verursacht wird war nicht zu ergründen, da es sich bei dieser Adresse nicht um ein Register handelt.

Die anderen beiden Funktionen zum Senden und Empfangen von Zeichen benutzen direkt die SCI (Serial Communications Interface) Register des 68HC11. Beim Senden wird erst das TDRE Bit (Transmit Data Register Empty) des SCSR — SCI Status Register überprüft, ist dies true so wird das an die Funktion übergebene Zeichen durch einen POKE Befehl in das SCDR — SCI Data Register geschrieben. Die Funktion wirkt blockierend, bis das übergebene Zeichen in den Sendepuffer geschrieben wird oder ein Timeout auftritt, damit die Funktion nicht versucht auf Godot zu warten falls sie ihr Zeichen nicht los wird. Die Empfangsroutine wertete in erster Version nur ein Bit des Statusregisters aus. Das RDRF Bit (Receive Data Register Full), was normalerweise ausreicht um ein neu eingetroffenes Zeichen zu erkennen. Fortwährende Probleme mit der Seriellen Kommunikation machten allerdings die Benutzung aller für den Empfang verantwortlichen Status Bits unumgänglich. So wurde eine Überprüfung des OR-, Noise- und Framing Error - Flags nachimplementiert. Das Overrun (OR) Flag wird gesetzt wenn ein neues Zeichen übertragen wird und ein altes noch nicht abgeholt wurde. Das Noise Flag wird immer dann gesetzt wenn die drei während einer Bitzeit gesampelten Werte nicht übereinstimmen und schließlich das Framing Error (FE) Flag welches eine Verletzung der Framegrenzen anzeigt. Aber auch diese Änderungen führten nicht zu einer signifikanten Verbesserung der Übertragung. Weiterhin zeigte sich während der ersten Tests, daß das 6.270 Board nicht in der Lage ist, während User Tasks ausgeführt werden, einen Datenstrom bei 9600 Baud im reinen Pollingbetrieb zu verarbeiten. Diese Umstände erzwangen eine vollständige Umstrukturierung des Protokolls.

5.1.2 Das Protokoll

Auf der Grundlage der Systembeschreibung und den bis jetzt gemachten Erfahrungen wurde das Serielle Protokoll neu definiert. Zum Auslesen der Sensordaten war eine möglichst einfache Struktur Anzustreben, die gleichzeitig eine hohe Betriebssicherheit garantierte. Da das 6.270 Board nicht in der Lage war 9600 Baud zu verarbeiten, wurde festgelegt, das jedes Zeichen explizit durch das Senden eines ASCII Zeichens angefordert werden muß, so daß ein Overrun ausgeschlossen werden konnte. Alle Sensordaten sollten in einem Frame übertragen werden, daraus ergab sich das Problem die Daten wieder den Sensoren zuzuordnen, insbesondere beim Auftreten von Übertragungsfehlern. Da sich Übertragungsfehler im Bereich der Sonar - und Kompaß Daten kaum durch eine Plausibilitätsüberprüfung der eingehende Werte ermitteln ließen, wurde dazu übergegangen jedes Datum innerhalb des Frames zu Numerieren, so daß eine Verwechslung nicht mehr möglich war und gleichzeitig die Fehlererkennung erleichtert wurde. Jeder Frame sollte mit ‚s' beginnen und mit 255 terminiert sein. Somit ergab sich folgender Sensor Frame:

s	1	SonarData1	2	SonarData2	3	SonarData3	4	SonarData4	5	SonarData5	6	SonarData6	7	SonarData7	8	SonarData8
	9	Kompass	10	nk Breite	11	nk Breite	12	nk Breite	13	nk Breite	14	nk Länge	15	nk Länge	16	nk Länge
	17	Nk Länge	255													
nk – Nachkommastelle; Länge des Frames insgesamt : 37 Byte																

Bild 5-3 : Sensor Data Frame

Die Kommunikation in die entgegengesetzte Richtung, also Kommandos vom Board an das Sensorinterface, gestaltete sich wesentlich einfacher, da der Atmel Microcontroller ohne Probleme in der Lage war die seriellen Daten einzulesen. Aus diesem Grund wurde bei den Kommandos auf eine Numerierung der Bytes verzichtet

und statt dessen eine einfache Prüfsumme bestehend aus 2 Byte eingesetzt, was den Overhead des Frames stark reduzierte.
Es sollten 4 verschiedene Kommandos mit ihren Parametern zum Sensorinterface übertragen werden. Hieraus ergab sich folgende Protokollstruktur: Das erste Byte enthält das Kommando, die 5 folgenden Bytes seine Parameter, das 7 und 8 Byte die Prüfsumme.

Kommando	Code	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Parameter 5	Prüfsumme1	Prüfsumme2
Sonar Parameter	s	Anschlag links	Anschlag Rechts	Geschwindigkeit	0	0	Prüfsumme1	Prüfsumme2
Scan Start/Stop	g	0	0	0	0	0	Prüfsumme1	Prüfsumme2
Reset	r	0	0	0	0	0	Prüfsumme1	Prüfsumme2
Motor	o	links +127/-127	rechts +127/-127	0	0	0	Prüfsumme1	Prüfsumme2
Länge des Frames: 8 Byte								

Bild 5-4 : Command Data Frame

Alle Parameter müssen angegeben werden. Die Parameter 4 und 5 werden (noch) nicht benutzt, sie waren als Reserve gedacht, so daß im Falle einer Erweiterung nicht der gesamte Frame überarbeitet werden muß.

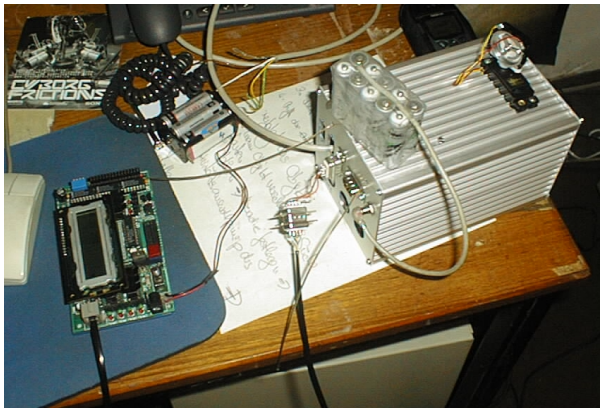


Bild 5-5 : links das 6.270 Board, rechts das Sensorinterface

Das neue Protokoll ermöglichte nun endlich eine sichere Kommunikation zwischen den beiden Microcontrollern, besonders der neue Sensor Data Frame brachte Vorteile, da sich die eingehenden Daten endlich sicher parsen ließen, und aufgrund ihrer Numerierung ein effektives Debugging möglich war.

5.2 Der Erste Plattformprototyp

Circa 4 Wochen vor Abreise in die Schweiz war der erste Plattformprototyp fertig, und es konnten erste Tests gemacht werden, da bis dato noch keiner wußte wie sich die Maschinen verhalten würden und ob die Motorelektronik den Anforderungen gewachsen war. Jetzt konnte man sich ein Bild von der Geschwindigkeit, Wendigkeit und Gewicht der Plattformen machen. Die Fahrtests, realisiert mit einer Testsequenz von Motorkommandos, verliefen erfolgreich. Die Plattform war sehr schnell und aufgrund der guten Motorisierung auch sehr wendig. Allerdings war auch das Gewicht nicht unerheblich, so daß es später unter keinen Umständen zur Kollision kommen durfte. Nun konnte mit der Programmierung der Verhalten begonnen werden.

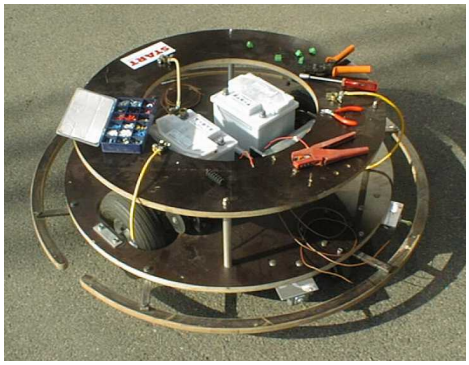



Bild 5-6 : Der erste Prototyp bei den letzten Arbeiten vor den Tests

Zu diesem Zeitpunkt begann es sich abzuzeichnen, daß ein Teil des gewünschten Verhaltens aufgrund von Zeitmangel nicht mehr zu realisieren sein würde.

5.3 Die Verhalten

Die primäre Anforderung nach kollisionsfreier Fortbewegung innerhalb eines abgegrenzten Spielfeldes ist mit rein verhaltensbasierten Design nicht zu erreichen. Deshalb sollte eine Art ‚hybrid-design' realisiert werden, in dem ein Verhalten die kollisionsfreie Fortbewegung ermöglicht und ein weiteres Verhalten mit Hilfe einer ‚Karte' des Spielfeldes die Einhaltung seiner Grenzen garantiert. Die in Verbindung mit dem Kartenbasierten Ansatz entstehenden Problem der Selbslokalisation, sollten mit Hilfe des Kompasses zur Bestimmung der Fahrtrichtung und des GPS zur Erfassung der absoluten Position, gelöst werden. Weiterführend sei hier an [Gutmann00] verwiesen.
Die Programmierung erfolgte in aufeinander aufbauenden Schichten. Die unterste Schicht der Programmstruktur, bestand aus einem Seriellen Treiber und einem Protokollinterface zum Senden der Kommandos und Parsen der eingehenden Sensordaten. Sie wurde so realisiert, das sie der darüberliegenden Verhaltensschicht eine sichere serielle Kommunikation garantierte. Die Implementierung der an dieser Schicht beteiligten Funktionen erfolgte im Sinne des Experimentellen Prototypings (vergl. Kap. 3.1) und wurden im Laufe des voranschreitenden Projektes immer wieder überarbeitet und optimiert.
In der zweiten Schicht sollten die Verhalten realisieren werden. Das erste Verhalten war eine Art Flucht. Der Roboter sollte immer versuchen dorthin zu fahren, wo er nach einem Sonarscan die größte Entfernung ermittelt hatte. Hauptgrund für dieses Verhalten war die einfache Überprüfbarkeit, so daß man durch einfaches Zuschauen feststellen konnte, ob das Zusammenspiel der Microcontroller sowie des Sonars zufriedenstellend funktionierte.
Die erste Erfahrung war, daß es aufgrund der Dauer eines Sonarscans und der Latenzzeiten der seriellen Übertragung unmöglich war während der Fahrt die Umgebung zu scannen. Trotz des Einsatzes eins Hochgeschwindigkeitsservos dauerte ein vollständiger Scan immer noch zwei Sekunden, was bei der normalen Fortbewegungsgeschwindigkeit die aufgenommenen Werte so sehr verzerrte, daß sich

keine sinnvolle Aussage über die Umgebung machen ließ. So wurde dazu übergegangen den Roboter schrittweise zu bewegen. Die Fortbewegung wurde in zwei Schritte aufgeteilt.

- Scannen: Ausführen eines Sonarscans und auswerten der Daten.
- Fahren: Aufgrund der Daten den nächsten Bewegungsschritt[2] ausführen

Ein weiteres Problem war ‚daß bei jedem Scan unerklärliche Verschiebungen innerhalb des Wertearrays auftraten, die wie sich später herausstellte auf das Programm des Atmel Microcontroller zurückzuführen waren. Leider konnte der Fehler nie vollständig behoben werden, allerdings ließ er sich aber mit einem ‚workaround' unschädlich machen.
Nach etlichen, langwierigen Tests mit verschiedenen Software Prototypen, war endlich eine zufriedenstellende Parametrisierung erreicht, so daß der Roboter in der Lage war sich autonom und kollisionsfrei zu bewegen.

Das zweite Verhalten dieser Schicht sollte die Einhaltung der Spielfeldgrenzen garantieren. Der Roboter sollte mit Hilfe des Kompasses und des GPS Empfängers in regelmäßigen Abständen seine eigene Position und Richtung mit den Koordinaten einer gespeicherten Karte des Spielfeldes vergleichen. So würde sich ein Übertreten der Grenzen vollständig vermeiden lassen. Aufgrund der recht einfachen Struktur des Spielfeldes, ein einfaches Rechteck, ließ es sich auch dementsprechend einfach in eine Karte umwandeln. Sie Bestand aus 4 Vektoren, die die Spielfeldgrenzen abbildeten und zusammengesetzt waren aus Startpunkt, Endpunkt und Himmelsrichtung. Dies schaffte die Möglichkeit zu bestimmen unter welchem Winkel der Roboter versucht die Grenze zu überschreiten, und ermöglichte so ein noch differenziertes Verhalten.
Die Auswertung der Kompaßdaten gestaltete sich recht einfach. Das Ausrichten der Roboter in eine bestimmte Himmelsrichtung hingegen, verursachte aufgrund der noch fehlenden Shaftencoder einige Schwierigkeiten. Das zeitgesteuerte Ausrichten der Roboter erforderte eine mühevolle Parametrisierung der Software und sehr viele Tests. Zusätzlich stellte sich während dieser Tests heraus, daß die Kompaßdaten von den Magnetfeldern der Motoren stark verfälscht wurden. So mußten auch hier Stillstandszeiten in Kauf genommen werden, um eine genaue Messung der Himmelsrichtung vorzunehmen. Die Arbeit an diesem Verhalten wurde nach wenigen Tagen aufgrund von Zeitmangel mit nicht zufriedenstellendem Ergebnis zurückgestellt.
Die Auswertung der GPS Daten war ebenfalls einfach, da der GPS Empfänger über sein serielles Interface NMEA Daten zur verfügung stelle in denen Länge und Breite im Klartext vorhanden sind. Allererste Tests mit einem Softwaremodul, welches die GPS Koordinaten nur auf dem Display des 6.270 Boards darstellte zeigten, daß die vom GPS Ermittelten Koordinaten einer Drift unterlagen. Die ermittelten Daten bewegten sich circa in einem Umkreis von 5 Metern um den Roboter, für die Einhaltung einer Borderline viel zu ungenau. Einzige Möglichkeit die Genauigkeit zu erhöhen bestand in der Realisierung eines DGPS- Empfängers.
Bei diesem Verfahren benutzt man einen weiteren Empfänger der in der Lage ist DGPS Daten zu empfangen. Diese Daten werden von einer Bodenstation erzeugt, deren Position exakt vermessen ist, und somit in der Lage ist die Drift des GPS Signals

[2] Fortbewegung um ca. 1,5 meter, evtl. mit Richtungskorrektur.

zu berechnen. Der DGPS Empfänger empfängt diese Daten und überträgt sie direkt über eine Serielle Verbindung an den GPS Empfänger. Dieser ist dann in der Lage seine eigene Drift zu berechnen und kann so seine Koordinaten korrigieren. Leider ist das DGPS Signal im Gegensatz zum GPS Signal nicht weltweit verfügbar und auch nicht in jedem Land auf den selben Funkfrequenzen. Aufgrund der guten Verbindungen des Gottlieb Duttweiler Institutes in der Schweiz stellte das Landestopographieamt der Schweiz drei DGPS Empfänger leihweise zur Verfügung. Die ersten Experimente mit diesen Empfängern verliefen sehr erfolgreich, die Drift reduzierte sich auf unter einen Meter. So wurde das Verhalten zur Einhaltung der Spielfeldgrenzen vollständig implementiert, doch die folgenden Test mit einem der Roboter verliefen katastrophal. Die elektromagnetischen Störungen, erzeugt durch das 6.270 Board waren so massiv, daß der DGPS Empfänger jeglichen Dienst verweigerte. Auch das mehrmalige Umverdrahten der Roboter sowie der Einsatz verschiedenster Antennen brachten keinerlei Verbesserungen mit sich.
Mit diesem Status wurden die Entwicklungsarbeiten aus Zeitgründen eingefroren. Alle bis Dato fertiggestellten Softwaremodule wurden noch einmal revidiert und finalisiert, so daß die Roboter in der Lage waren, sich auch ohne Kompaß und GPS sicher zu bewegen. Allerdings mußte die Überwachung der Spielfeldgrenzen von Hand erfolgen. Alle Höheren Verhalten, wie zum Beispiel das am Anfang geplante Roboter Ballett, konnte aus Zeitgründen nicht mehr realisiert werden.

6 Zusammenfassung, Kritik und Ausblick

Rückblickend auf dieses Projekt stellt man fest, daß das größte Problem der Zeitfaktor war. Dies ergab sich aus der Tatsache, daß der erste Prototyp erst circa 4 Wochen vor der Abreise zur Verfügung stand. Jede Implementierung von Verhalten für einen Roboter erfordert jedoch eingehende Tests, da sich alle Einflüsse unserer hoch komplexen dynamischen Umwelt nicht vorhersagen lassen. Somit konnte auch die Programmierung der Verhalten erst mit der Fertigstellung dieses Prototyps beginnen. Diese Problematik war allerdings zu Beginn der Planungen nicht einzuschätzen, da das Timing des Plattformbaus von den Künstlern festgelegt wurde. Auch die Realisierung der seriellen Kommunikation zwischen 6.270 Board und Sensorinterface stellte sich unerwartet schwierig und langwierig dar (vergl. Kap. 5.1).
Tatsächlich realisiert wurden 3 autonome Roboter versehen mit einem grundlegenden Verhalten konzipiert für eine freie Wiese. Taucht ein Objekt im Sonar des Roboters auf versucht dieser sich diesem Objekt bis auf einen Mindestabstand zu nähern, so daß es für einen Menschen möglich ist, den Roboter immer hinter sich her fahren zu lassen. Taucht kein Objekt im Sonar auf, so wird ‚willkürlich' herumgefahren, bis etwas gefunden wird. Verharrt das Objekt nach erfolgreicher Annäherung an seinem Platz erfolgt eine Drehung auf der Stelle um circa 180°, anschließend wird ein neues Objekt gesucht. Weiterhin wurde ein Sensor- und Leistungsinterface für das 6.270 Board entwickelt, welches über die Serielle Schnittstelle angebunden werden kann, und so die Möglichkeiten dieses Boards massiv erweitert.
Die ‚to do' Liste dieses Projektes ist lang, und könnte sicherlich noch um einiges erweitert werden. Priorität sollte die realisierung der Selbslokalisierung mit Hilfe des GPS sowie des Kompasses haben, da diese Funktionen schon zum großen Teil implementiert sind. Die vollständige Realisierung des Kompasses, sowie des DGPS Sensors würde eine neue Grundlage zur Implementation komplexerer Verhalten

schaffen, denn schon die Möglichkeit der Festlegung von Spielfeldgrenzen wäre ein großer Fortschritt.
Weiterhin sollte die Idee des zentralen Kommunikationsservers weiterverfolgt werden. Auch hier sind entscheidende Grundvoraussetzungen, wie zum Beispiel mehrere Serielle Funkstrecken, vorhanden. Dies würde eine Möglichkeit zur Fernlenkung der einzelnen Roboter schaffen sowie eine zentrale Verhaltenssteuerung ermöglichen. Somit ließe sich aus den drei unabhängigen Agenten eine Multiagentenlösung aus kooperierenden beziehungsweise konkurrierenden Agenten realisieren [Jung00]. Die Roboter könnten als Team miteinander oder gegeneinander agieren und so dem Betrachter sehr viel anschaulicher die Möglichkeiten von interaktiven und korrespondierenden Agenten vermitteln und den Künstlern weitere Dimensionen ihrer Ausdrucksmöglichkeiten erschliessen.

7 Literaturliste

[Arkin 98] :	Ronald C. Arkin, Behavior – Based Robotics, The MIT Press
[Floyd 00] :	Floyd, C.: A Systematic Look at Prototyping. In: Budde, R.; Kuhlenkamp, K.; Mathiassen, L.;
[Knoll 00] :	Robotics. In: Handbuch der künstlichen Intelligenz Kap.23. G.Görz, C.-R. Rollinger, J. Schneeberger
[HC11] :	Motorola 68HC11 Reference Guide Rev. 3.0
[Brech 92] :	Brechmann, Dzieia, Hörnemann, Hübscher, Jagla, Petersen – Elektrotechnik Tabellen – Kommunikationselektronik.
[Gutmann00] :	Jens-Steffen Gutmann, Thilo Weigel, Bernhard Nebel - A Fast, Accurate, and Robust Method for Self-Localization in Polygonal Environments Using Laser-Range-Finders, Albert-Ludwigs-Universit¨ at Freiburg, Institut für Informatik Am Flughafen 17, D-79110 Freiburg, Germany
[Jung00]	Agentenbasiertes Einsammeln verschiedenfarbiger Objekte. M. Jung.
[Garmin99]	www.garmin.de/Produktbeschreibungen/GPS12.html
[GPS00]	www.carrier.co.at/res/mac/tidbits/TidBITS-de-388.html
[Acro00]	www.acroname.com/robotics/parts/R11-6500.html
[V2x]	www.sander-electronic.de
[McMurtrie00]	www.cronos.net/~bk/amorphic/about.html