Max - Planck - Gymnasium Kollegstufenjahrgang 2001/2003

FACHARBEIT

aus dem Fach

Physik

Thema: Bau eines Roboters, der autonom durch ein Labyrinth findet

Verfasser: Martin Geier
Leistungskurs: Physik (LK Ph1)
Kursleiter: StR R. Rothhardt
Abgabetermin: 03.02.2003

Erzielte Note: _____ in Worten: _____
Erzielte Punkte: _____ in Worten: _____
(einfache Wertung)

Abgabe beim Kollegstufenbetreuer am _____ *(Vom Kursleiter auszufüllen)*

.....

(Unterschrift des Kursleiters)

Inhaltsverzeichnis

1 Einleitung

1.1 Entstehung der Idee

Der Wunsch, Roboter zu bauen, entstand schon viele Jahre vor der Kollegstufe, nämlich 1997. Damals entschied ich mich dazu, einen stationären Roboter zu bauen und diesen mittels einer PC-Interfacekarte anzusteuern. Dieser war vom Aufbau – ein drehbarer Arm mit zwei „Gelenken“ und einer dreh- und schwenkbaren „Hand“ – an einen damaligen Industrieroboter angelehnt und bot viele Möglichkeiten. Meines Erachtens war sein einziger Nachteil die Tatsache, daß er sich nicht von der Stelle bewegen konnte.

Mein Kursleiter bot als Facharbeit den Bau eines Roboters an, dessen Aufgabe es ist, aus einem Labyrinth herauszufinden. Da uns das Thema als zu umfangreich für eine Person erschien, beschlossen wir eine Aufteilung in einen Hard- und einen Softwareteil. Letzteren übernahm Steffen Hausmann. Damit stand einem zweitem Roboter nichts mehr im Weg.

1.2 Übersicht

Zuerst werden die zu lösende Aufgabe und die dafür benötigten Funktionen beschrieben, dann die nötigen Komponenten und deren Eigenschaften vorgestellt. Es wird auch kurz auf die Schnittstelle zwischen Hard- und Software und auf bei der Herstellung des Roboters aufgetretenen Probleme eingegangen.

2 Aufgabe des Roboters

2.1 Beschreibung der Aufgabe

Die Aufgabe des Roboters ist es, von einem beliebigem Punkt am Rande eines ihm unbekannten Labyrinths einen Ausgang aus diesem zu finden.
Dieses Labyrinth muss folgende *Kriterien* erfüllen:

- Es dürfen nur rechtwinkelige „Kurven“ vorkommen
- Die Seitenwände müssen senkrecht zur Grundplatte stehen
- Die Seitenwände dürfen keine glänzende Oberfläche haben
- Die Breite der Gänge muss ca. 20 cm betragen
- Das Labyrinth darf während der Fahrt nicht verändert werden

Dem Roboter steht zur Lösung dieser Aufgabe beliebig Zeit zur Verfügung.

2.2 Benötigte Funktionen

Der Roboter muss den Abstand zu den Labyrinthwänden erfassen, wofür ein Sensor notwendig ist *(Eingabe)*. Die folgende *Verarbeitung* der Eingangswerte und die *Ausgabe* von Steuersignalen an den *Antrieb* und an die *Lenkung* muss von einer *Steuerung* übernommen werden. Um die *Messwerte* (wie Akkuzustand) und die bereits *gefahrene Route* ansprechend darzustellen, ist eine *grafikfähige Anzeige* nützlich.

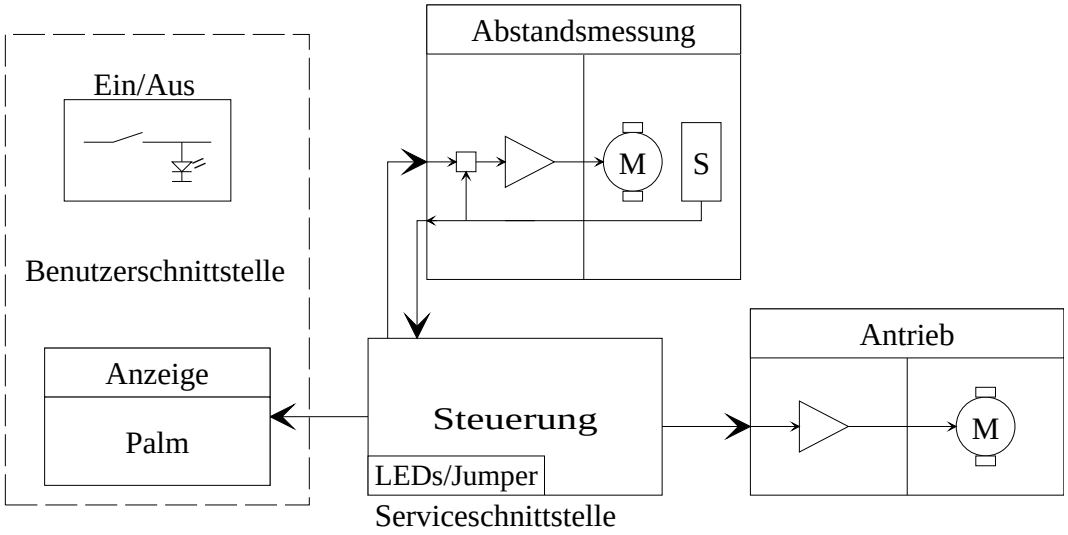


Abbildung 1: Blockschaltbild

3 Anforderungen und Beschreibung der Komponenten

3.1 Chassis

Ich habe mich für ein *rundes Chassis* entschieden, da dies den großen Vorteil hat, daß der Roboter keine Ecken hat, an denen er hängenbleiben könnte. Desweiteren ist es damit möglich, daß der Roboter sich auf engem Raum gefahrlos um die *eigene Achse drehen* kann (einen passenden Antrieb vorausgesetzt).

Das Chassis wurde in mehrere Ebenen aufgeteilt:

Die *obere Ebene* teilen sich Spannungsregelung, Microcontroller und die Schrittmotorsteuerung, weil diese Komponenten zu Servicezwecken gut erreichbar sein müssen. Der Antriebsmotor für den Messarm befindet sich ebenfalls hier.

Die *mittlere Ebene* enthält nur den drehbar gelagerten Abstandssensor, damit dieser „freie Sicht“ auf die Umgebung hat.

Die *untere Ebene* ist mit den Schrittmotoren und den Rädern bestückt. Um einen möglichst

tiefen Schwerpunkt zu erhalten, wurden die Akkus auf dieser Ebene montiert.
Diese drei Ebenen werden über einen Schacht verbunden.

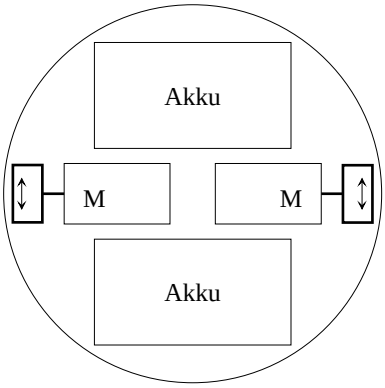


Abbildung 2: Untere Ebene

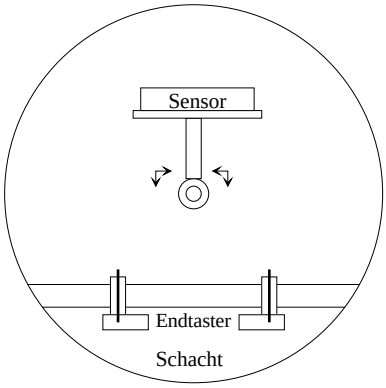


Abbildung 3: Mittlere Ebene

3.2 Antrieb und Lenkung

Beim Antrieb standen zwei Grundideen zur Auswahl:

- Idee 1: Zwei von einem einzigen Motor angetriebene Räder mit zwei lenkbaren (nicht angetriebenen) Rädern (vgl. einem Auto mit Heckantrieb) (siehe Abbildung 4)
- Idee 2: Zwei gegenüber angeordnete, unabhängige Räder mit zwei Motoren in der Mitte des Fahrzeugs und zwei Gleitern (vgl. einem Rollstuhl) (siehe Abbildung 5)

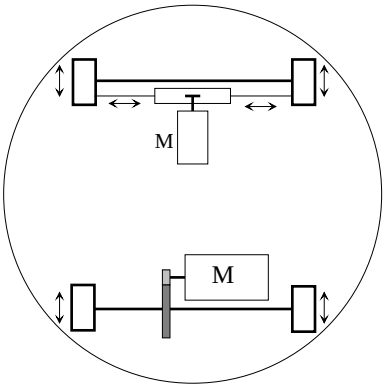


Abbildung 4: Idee 1

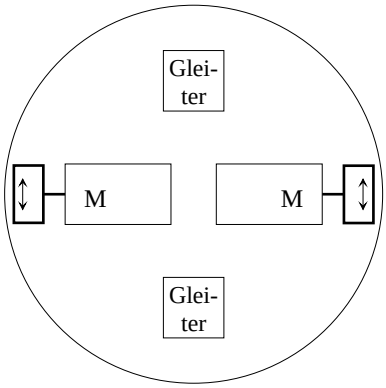


Abbildung 5: Idee 2

Idee 1 hätte den Vorteil, daß ein *einzelner Getriebemotor für den Antrieb* zuständig wäre und man sich somit nicht über unterschiedlich schnell laufende Motoren kümmern müsste. (Dennoch müsste ein Taktgeber o.ä. benutzt werden, damit der Microcontroller „weiß“, wie weit der Roboter gefahren ist.) Es wäre ein *zweiter Motor für die Bewegung der Lenkstange* nötig. Die Schwierigkeit dabei wäre, daß man eine *Positionsrückmeldung der Lenkstange*

bräuchte, um zu erfassen, wie weit die Lenkung momentan eingeschlagen ist. Weitere Nachteile dieser Lösung wären, daß sich der Roboter nicht um die eigene Achse drehen könnte. Außerdem wäre Mechanik nötig, die sehr viel Platz beanspruchen würde. Das hat uns dazu veranlasst, diese Idee zu verwerfen.

Idee 2 umgeht diese Probleme und ermöglicht die Drehung um die eigene Achse, was ein engeres Labyrinth zulässt: Hier werden *zwei Räder direkt auf die Motoren* geflanscht und somit der mechanische Aufwand minimiert. Die Räder müssen sich aus Stabilitätsgründen so weit außen wie möglich befinden.

Um vor- bzw. rückwärts zu fahren, lässt man die Motoren in die gleiche Richtung, um den Roboter auf der Stelle drehen zu lassen, lässt man die Motoren in unterschiedliche Richtungen laufen. Bei beiden Möglichkeiten müssen die Motoren *absolut gleich schnell* sein. Ist dies nicht der Fall, so fährt der Roboter, anstatt geradeaus, einen Bogen, bzw. bleibt beim Drehen nicht auf der Stelle. Da dies bei Getriebemotoren nur mittels eines Regelkreises (pro Motor) möglich wäre, werden *Schrittmotoren* verwendet, weil sich diese Motoren (in ihrem Nutzlastbereich) synchron bewegen:

Diese Motoren bestehen aus zwei oder mehr Spulen, die im Motor fest verankert sind, und einem drehbar gelagertem (magnetischem) Anker, der bei passender Ansteuerung der Spulen durch das entstehende Magnetfeld in Rotation versetzt wird. Großer Vorteil dieser Motoren ist, daß der Winkel, um den sich die Motorachse pro Schritt dreht, bekannt ist. Die Spulen werden mit einem konstanten Strom (hier etwa 200 mA pro Spule) betrieben, wofür eine spezielle Steuerung erforderlich ist.

Diese Steuerung wird pro Motor mit je zwei ICs der Firma ST realisiert, die die Ansteuerung aus Sicht der Steuerung stark vereinfachen: Pro Motor müssen nur zwei Digitalsignale an die Steuerplatine übergeben werden: Der Schrittakt und die Richtung. Bei jeder steigenden Flanke bewegt sich der Motor einen Schritt weiter. So lassen sich durch die Erzeugung einer Rechteckspannung beliebige Fahrtgeschwindigkeiten (abhängig von der Frequenz des Signals) erzeugen.
Die Funktionsweise der Steuerung lässt sich in zwei Teile aufteilen: Die *Spulensteuerung* und die *Leistungsregelung*.

Die *Spulensteuerung* wird von einem Schrittmotorcontroller L297 übernommen. Dieser Controller generiert aus zwei Signalen (Takt und Richtung) die vier Signale, die von der *Leistungsregelung* benötigt werden. In Kombination mit dem dazu verwendetem L298 bietet der L297 eine einstellbare *Strombegrenzung*, die zur effizienten Ansteuerung der Motoren nötig ist. Der L298 verstärkt die übernommenen Signale und kann einzelne Spulen mit bis zu 2 A ansteuern. Der Spulenstrom wird gemessen und für die Strombegrenzung an den L297 übertragen. Diese Kombination wird vom Hersteller empfohlen und auch in vielen kommerziellen Produkten verwendet.

Nachteil dieser Lösung: Da der Motor bei inaktiven Spulen kaum Widerstand leistet und der Roboter somit wegrollen könnte, muss der Spulenstrom auch beim Stehen oder kurzen Fahrtunterbrechungen aufrecht erhalten werden. Außerdem würde die Synchronität der beiden Motoren verloren gehen, weil sich die Motorachsen bei inaktiven Spulen an dem Magnetfeld der Permanentmagneten neu ausrichten würden.

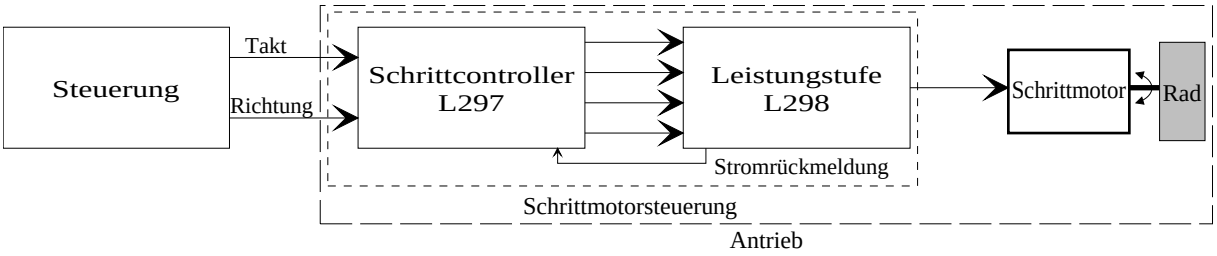


Abbildung 6: Blockschaltbild Antrieb

3.3 Abstandsmessung

Zur Abstandsmessung kann man entweder einen Ultraschall- oder einen Infrarotsensor benutzen. Wir haben uns für einen *Infrarotsensor* entschieden, da es bei Ultraschall zu zuvielen störenden Reflexionen an den Labyrinthwänden kommen würde: Der GP2D120 von Sharp ist ein Sensor, der *unabhängig vom Reflexionsgrad des angepeilten Materials* den Abstand ermittelt. Zur Verdeutlichung der Funktionsweise ein Auszug aus dem Datenblatt:

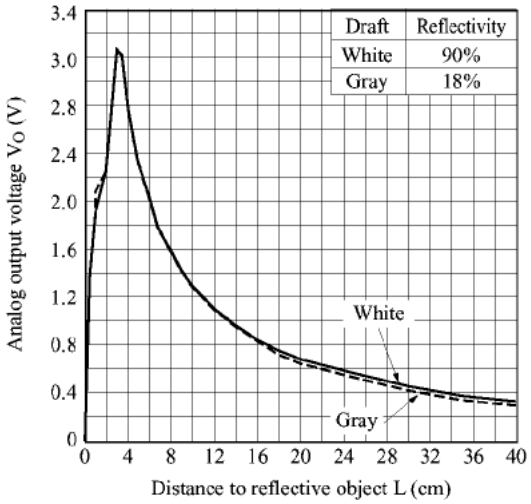


Abbildung 7: L-V_O-Diagramm

Der Sensor misst mittels Infrarotstrahlen nach dem *Triangulationsprinzip* [1] den Abstand L (im Bereich von 4-30 cm) zur Wand und gibt dann abhängig davon eine *Spannung* V_O zwischen 0,4 V und 3,0 V aus. Dieser Zusammenhang wird im obigen Diagramm dargestellt.

Um die benötigten Messwerte – Abstand nach vorne und nach links – zu erfassen, wären bei fester Montage mindestens zwei dieser Sensoren nötig. Da ein Sensor mit 20€ recht teuer ist, wird er auf einem *drehbaren Arm* montiert. Dieser von einem kleinen Getriebemotor angetriebene Arm kann drei Positionen anfahren: Links, Vorne und Rechts[1]. Um die aktuelle Armposition zu erkennen wurden *drei Microtaster* eingebaut, die mit dem Microcontroller

[1] Diese Messposition kann zur Erkennung einer Sackgasse verwendet werden.

verbunden sind. In die zur Ansteuerung des Arm-Motors nötige Schaltung wurde zusätzlich noch eine Schutzfunktion integriert, die den Motor selbständig am Endanschlag abschaltet, um bei eventuellen Programmfehlern oder Störungen eine Beschädigung des Arms zu verhindern. Da der Messbereich erst bei 4 cm beginnt, kommt es zu unsinnigen Ausgabewerten bei kleineren Abständen. Weil sich diese nicht erkennen lassen, wurde die Länge des Arms so gewählt, daß der Sensor immer mindestens 4 cm von der Außenkante des Roboters entfernt ist.

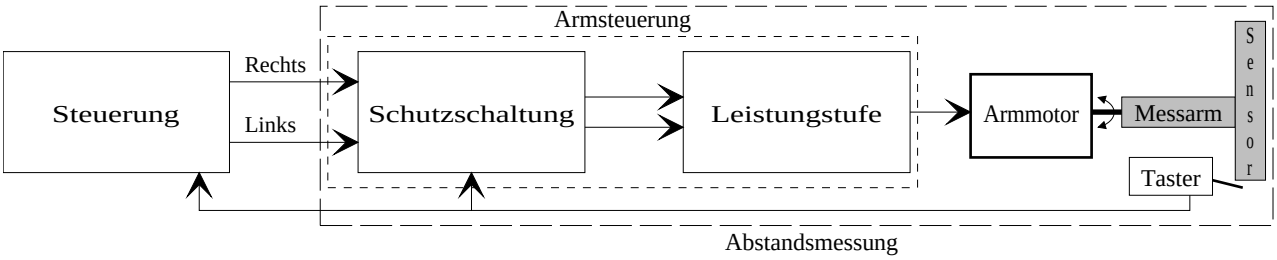


Abbildung 8: Blockschaltbild Abstandsmessung

3.4 Versorgung und Steuerung

Zur *Energieversorgung* kommen acht *NiCd-Akkus* mit Lötfahnen zum Einsatz, die in Serie geschaltet eine Spannung von $1{,}2\text{V} \cdot 8 = 9{,}6\text{V}$ liefern. Sie ermöglichen einen *völlig autonomen Betrieb* des Roboters. Bei einer Kapazität von 2,2 Ah und einem Stromverbrauch von durchschnittlich 1,1 A kann man folglich eine maximale Laufzeit von 2 Stunden erwarten. Um das Wiederaufladen zu vereinfachen, wurde eine Ladebuchse integriert. Die Schrittmotoren werden direkt mit den 9,6 V versorgt; für die ICs wird mittels eines Festspannungsreglers 7805 der Firma ST eine konstante Spannung von 5 V erzeugt.

Da das *Auswerten der Messergebnisse* und die *Ansteuerung der Motoren* mit einfachen Mitteln aus der CMOS-Technik (wie Logikgattern, Timern oder Komparatoren) nur schwer zu lösen ist, bietet sich dafür der Einsatz eines kleinen Microcontrollers an. Der Einsatz eines solchen Bausteins anstelle von CMOS-Technik bringt viele Vorteile mit sich, z.B. daß die gesamte Logik in Software realisiert wird und somit die Hardware bei Veränderung der Logik nicht umgebaut werden muss. Durch den Einsatz des Microcontrollers entstehen allerdings auch Nachteile: Er benötigt eine präzisere Spannungsversorgung als die CMOS-Technik und zusätzliche Peripherie.

Es kommt ein *ST6225-Microcontroller* zum Einsatz. Dieser bietet/fordert u.a. folgendes:

- 8 MHz Prozessortakt[2]
- 20 frei programmierbare E/A-Leitungen
- einen Analog-Digital-Wandler

[2] schnell genug für unseren Anwendungszweck

Der Prozessortakt muss extern erzeugt werden. Hier wird ein fertiger Oszillator verwendet, der ohne Zusatzbeschaltung die 8 MHz ausgibt.

Der Microcontroller ist mit der Schrittmotorsteuerung, dem Abstandssensor und der Armsteuerung verbunden. Desweiteren erhält er über einen Spannungsteiler die Akkuspannung, um den *Ladezustand des Akkus* überprüfen zu können. An einem weiteren Eingang kann man mittels mehrerer Jumper eine variable Spannung anlegen. Dies kann dazu benutzt werden, verschiedene *Betriebsmodi* auszuwählen.

3.5 Serielle Schnittstelle und grafikfähige Anzeige

Als Schnittstelle zwischen Steuerung und Palm bietet sich die bei Computern (noch) gebräuchliche serielle Schnittstelle (RS232) an. Über sie lassen sich in beide Richtungen Daten verschicken. Da die serielle Schnittstelle, laut Standard, Spannungen um $\pm$12 V benötigt, war eine direkte Verbindung zwischen Microcontroller und Palm nicht möglich. Deshalb wird der Treiberbaustein MAX232 von Maxim verwendet. Dieser Treiber generiert aus einer 5 V-Versorgung mit wenigen externen Komponenten Spannungen von $\pm$10V, die zur Benutzung der seriellen Schnittstelle ausreichen. Der IC ist über vier Leitungen mit dem Microcontroller verbunden, um die Übertragung in beide Richtungen zu unterstützen. Zum Anschluss des Palms wird ein neunpoliger SubD-Steckverbinder verwendet.

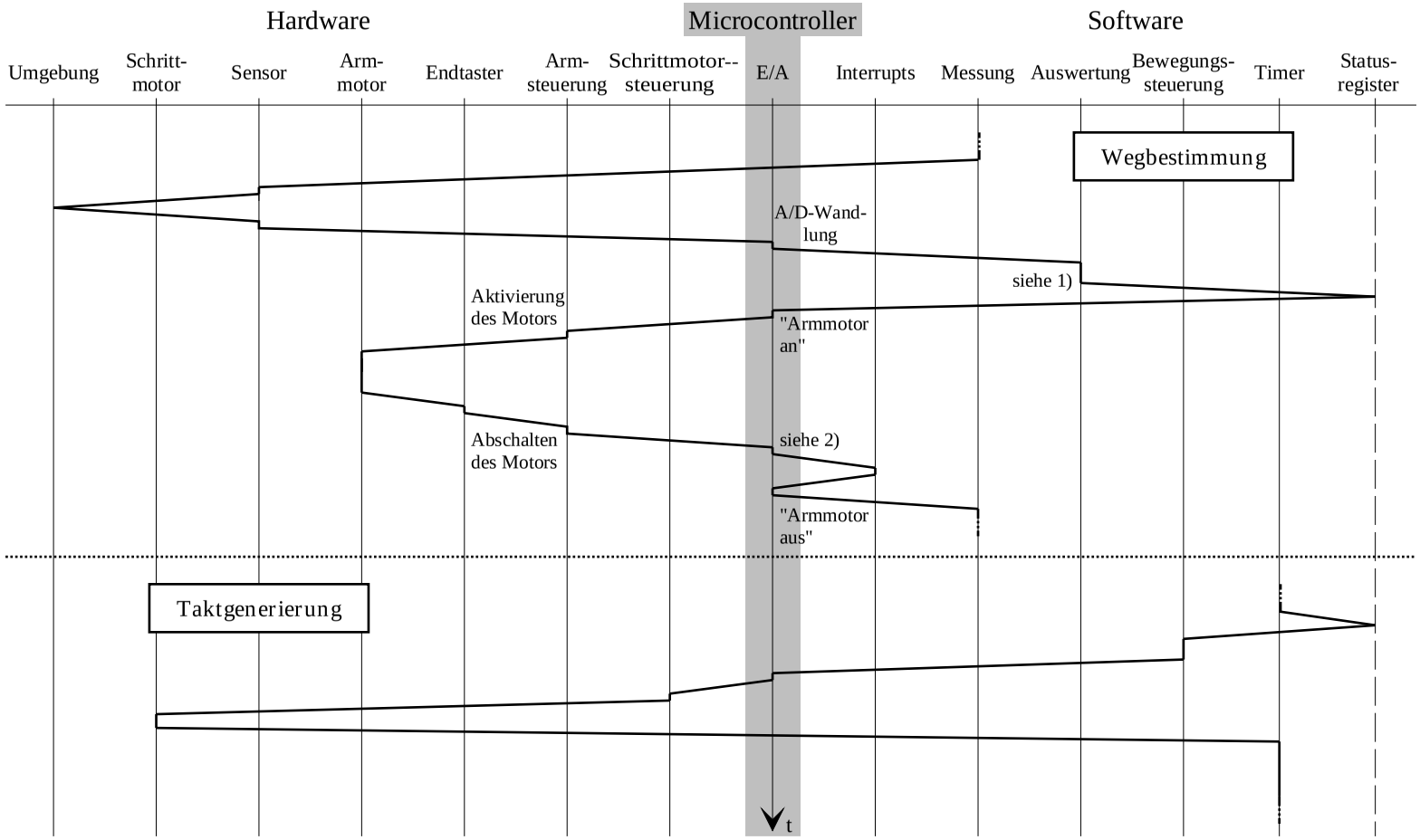
4 Schnittstellenbeschreibung

4.1 Steuerung

Port	Funktion	Port	Funktion
PA0	Arm rechts	PB2	Endtaster 1
PA1	Arm links	PB3	Endtaster 2
PA2	LED 1	PB4	Endtaster 3
PA3	LED 0	PB5	Akkuspannung (analog)
PA4	Schrittmotor 1 (Takt)	PB6	-
PA5	Schrittmotor 2 (Takt)	PB7	-
PA6	Schrittmotor 1 (Richtung)	PC4	RS232-Treiber Leitung 1
PA7	Schrittmotor 2 (Richtung)	PC5	RS232-Treiber Leitung 2
PB0	Jumper zur Programmauswahl (analog)	PC6	RS232-Treiber Leitung 3
PB1	Abstandssensor (analog)	PC7	RS232-Treiber Leitung 4

4.2 Ablaufdiagramm

In folgendem Diagramm lässt sich die zeitliche Interaktion der verschiedenen Bestandteile erkennen. Links sind die verschiedenen Hardwarekomponenten dargestellt, in der Mitte die Schnittstelle zwischen Hard- und Software, der E/A-Bereich des Microcontrollers, und rechts finden sich die einzelnen Softwareroutinen wieder.



Zu 1: Abhängig vom Messwert wird das Statusregister geändert, das die Bewegungssteuerung beeinflusst. Desweiteren wird der Arm an die nächste Messposition gefahren.

Zu 2: Die Schutzfunktion der Armsteuerung reagiert schneller als der Microcontroller auf das Signal des Endtasters.

5 Probleme bei der Realisierung

5.1 Platinenherstellung

Aus Platz- und Schaltungsgründen musste ich für die Schrittmotorsteuerung eine doppellagige Platine fertigen lassen. Es fand sich schnell ein Hersteller, bei dem es möglich war, kleinere Platinen für Beträge unter 20€ in Auftrag zu geben. Das Problem war, daß dieser Hersteller nicht in der Lage war, die Platine chemisch durchzukontaktieren[3]. Er verwendete sogenannten Hohlnieten, was jedoch eine unbefriedigende Lösung darstellt: An diesen eingepressten Nieten entstehen gerne Wackelkontakte und manche Nieten stellten wegen ihrer Größe nicht erwünschte Verbindungen zu Nachbarbahnen her. Die sehr zeitaufwändige Lösung war, die Hohlnieten erst zu zuschneiden und dann zu verlöten.

5.2 Übertragungsfehler zwischen RS232-Treiber und Palm

Sehr lange hat uns das Phänomen beschäftigt, daß Übertragungsfehler auf der seriellen Leitung auftreten, wenn der Palm direkt am RS232-Treiber angeschlossen wurde. Sobald man einen PC dazwischenschaltete, kamen die Daten fehlerlos an. Die Übertragungsqualität ließ sich durch Verwendung eines kleinen Kondensators und eines Pull-Up-Widerstandes an der Datenleitung lösen, anscheinend kommt die RS232-Schnittstelle des Palms nicht mit der durch Schwingkreise erzeugten Spannung des MAX232 zurecht.

5.3 Sonstiges

Als störend bei der Entwicklung erwies sich die Tatsache, daß die Herstellerfirma ST auf dem Experimentierboard des ST6-Microcontrollers die Anschlussleiste für die E/A-Leitungen nicht bestückt hatte. Als Provisorium steckten wir die Drähte in die vorgesehenen Bohrungen auf der Leiterplatte, was allerdings zu Wackelkontakten führte. Ich habe dann eine Buchsenleiste auf das Experimentierboard gelötet, was mit den passenden Steckern zu einer guten Verbindung führte.

Desweiteren wurden die Auslösehebel der verwendeten Endtaster abgeschliffen, was zu unzuverlässiger Funktion führte. Sie wurden durch andere, besser geeignete mit Auslösehebel aus Metall ersetzt.

[3] durchkontaktieren: die obere und die untere leitende Lage der Platine miteinander verbinden

6 Ergebnis und Ausblicke

Der Roboter ist nun in der Lage, die gestellte Aufgabe fehlerfrei zu lösen.

Am Microcontroller sind noch zwei I/O-Leitungen unbenutzt, die in Zukunft mit weiteren Funktionen belegt werden können (z.B. Funkverbindung zu einem PC). Auch ist das Potential der seriellen Schnittstelle bei weitem noch nicht ausgeschöpft: Man könnte eine Rückverbindung vom Palm zum Roboter realisieren, über die der Palm eventuelle Kurskorrekturen veranlassen könnte, wenn der Roboter im Kreis fährt.

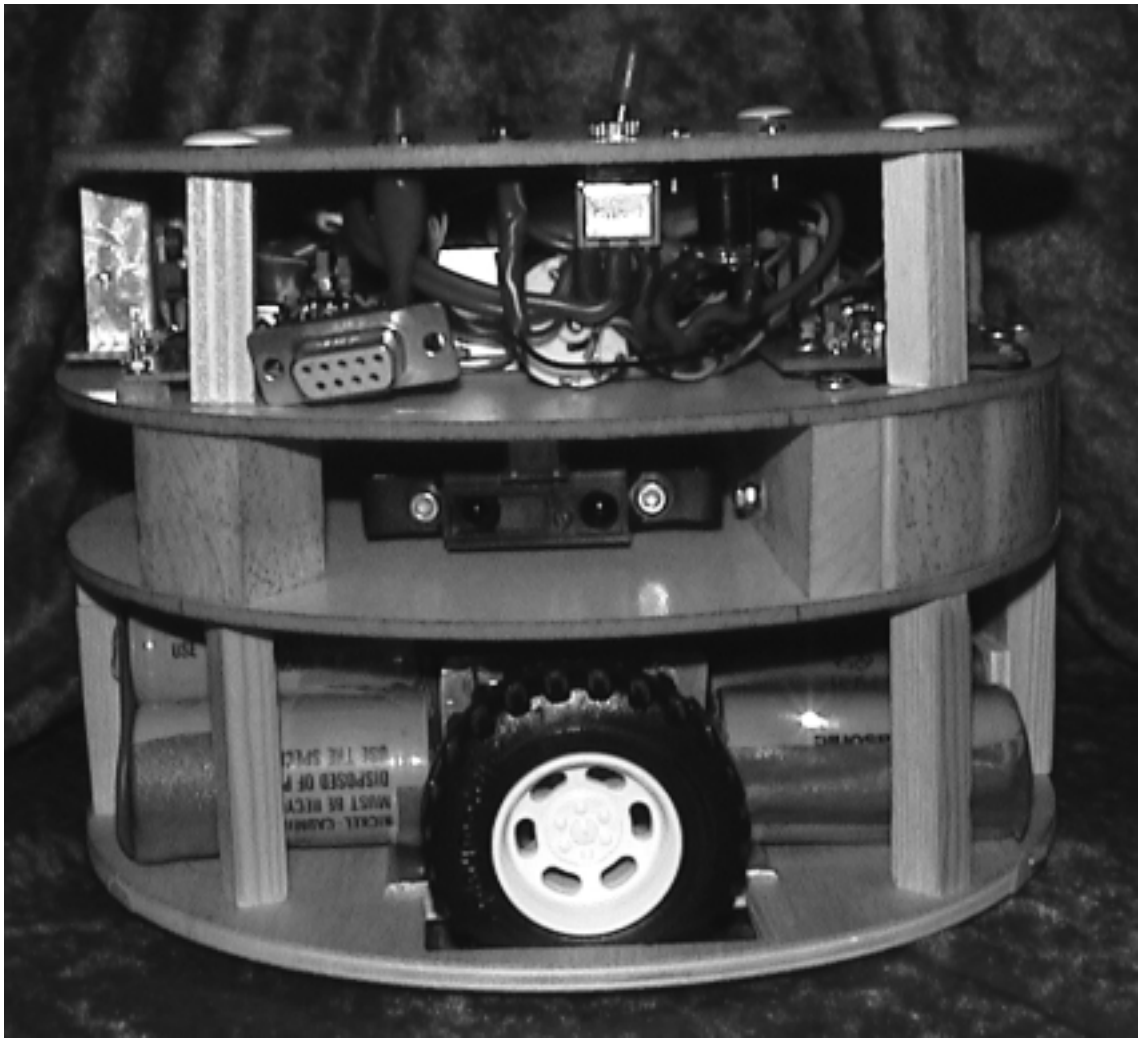


Abbildung 9: Seitenansicht des Roboters (ohne Verkleidung)

7 Literatur- und Hilfsmittelverzeichnis

Alle URLs wurden zuletzt am 30. Jan. 2003 aufgerufen; verantwortlich sind die jeweiligen Hersteller oder die angegebenen Personen.

Datenblätter:
L297: http://www.st.com/stonline/books/pdf/docs/1334.pdf
L298: http://www.st.com/stonline/books/pdf/docs/1773.pdf
GP2D120: http://www.sharp.co.jp/products/device/ctlg/jsite22_10/table/pdf/osd/optical_sd/gp2d120_j.pdf
7805: http://us.st.com/stonline/books/pdf/docs/2143.pdf
ST6225: http://us.st.com/stonline/books/pdf/docs/5266.pdf
MAX232: http://pdfserv.maxim-ic.com/arpdf/MAX202E-MAX241E.pdf

Sonstiges:
[1] Funktionsprinzip Triangulation bei Sharp-Sensoren:
http://www.hopa-robotics.de/content/psd_sensoren.html (Dirk Moser)
FAQ von de.sci.electronics: http://dse-faq.e-online.de/dse-faq.htm
LaTeX Kurzbeschreibung: ftp://dante.ctan.org/tex-archive/info/lshort/german/l2kurz2.pdf (W. Schmidt, J. Knappen, H. Partl, I. Hyna)

8 Erklärung des Kollegiaten

Ich erkläre hiermit, daß ich die Facharbeit ohne fremde Hilfe angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel benutzt habe.

München, den 03.02.2003 .....

(Martin Geier)