Ich möchte mich an dieser Stelle ganz herzlich bei allen bedanken, die mich bei meinem Projekt mit ihrem Wissen und Ihrer Erfahrung unterstützt haben.

Ein ganz besonderer Dank gilt der Firma Beta LAYOUT GmbH [BLG04], die mir die Platinen für mein Projekt kostenlos angefertigt hat.

Der wichtigste Dank aber gilt meinen Eltern, Freunden und Verwandten, die mir mit viel Verständnis und Geduld die ganze Zeit zur Seite gestanden haben und mich finanziell bei meinem Projekt unterstützt haben.

Fabian Greif

Januar 2004

Inhaltsverzeichnis

1. Einleitung

Vor einiger Zeit stellte man fest, dass die Fortbewegungsart „Laufen“ technisch viel schwieriger zu realisieren ist als zum Beispiel „Fliegen“ oder „Schwimmen“. Die ersten Modelle von Laufrobotern mussten noch aufwendig von einem Bediener ferngesteuert werden, doch mit der Entwicklung der Chiptechnologie wuchsen auch die Möglichkeiten, die Maschinen selbständig laufen zu lassen.
„Wozu braucht man eigentlich Laufroboter“, mag der eine oder andere fragen, „haben wir bisher nicht immer alles mit Autos bzw. Rädern erreichen können ?“ Doch Räder brauchen immer eine glatte Fläche, auf denen sie rollen können. Stoßen sie an ein starres Hindernis, so muss ein Fahrzeug mit Rädern kapitulieren, ein Laufroboter hingegen könnte einfach darüber hinwegsteigen und seinen Weg fortsetzen. Allerdings sind Laufroboter viel aufwendiger und damit auch störanfälliger und teurer als vergleichbare Fahrzeuge, so dass ihr hauptsächlicher Anwendungsbereich wohl in unwegsamem Gelände, wie zum Beispiel bei der Erkundung neuer Planeten oder der Minensuche usw. zu finden sein wird.

Leider sind die heute gebauten Laufroboter noch viel zu langsam, so dass eine sinnvolle Nutzung bisher nur eingeschränkt möglich war. Es gibt aber schon erste laufende, industriell eingesetzte Maschinen, wie zum Beispiel den Finnischen Harvester, der sich auf sechs Beinen durch den Wald bewegen kann, ohne den Boden so stark wie normale Fahrzeuge zu beanspruchen.
Trotzdem werden die laufenden Maschinen eher eine Randgruppe bleiben, ohne die aber bestimmte Aufgaben nur sehr schwer gelöst werden können.

Leider reicht der Rahmen dieser Arbeit nicht für eine vollständige Beschreibung des Projektes aus. Ich habe daher versucht, mich auf die wichtigsten Punkte zu beschränken und so einen Überblick über das Projekt zu geben.

1.1 Warum gerade ein Laufroboter ?

Ich beschäftige mich schon seit zirka zwei Jahren mit dem Gebiet der Robotik. Alle Roboter, die ich bis jetzt gebaut habe, fuhren auf Rädern, doch immer mehr kam der Wunsch in mir auf, auch einmal einen laufenden Roboter zu bauen, da laufende Roboter viel lebendiger wirken und viel universeller einsetzbar sind. Lange habe ich dieses Projekt als zu aufwendig und zu teuer abgetan.
Doch die Faszination der Laufroboter ließ mich nicht mehr los und so habe ich dann mehr spontan mit den Planungen zu einem Laufroboter angefangen.
Es war von Anfang an klar, dass der Roboter vier oder sechs Beine haben sollte. Dass er am Schluss vier Beine hatte, war zum großen Teil auch eine Kostenfrage. Die Servos für die Beine waren das Teuerste am ganzen Roboter und zwei Beine mehr oder weniger fielen da ganz schön ins Gewicht.
Wie sich jetzt herausgestellt hat, ist das Halten des Gleichgewichts für einen Roboter mit vier Beinen

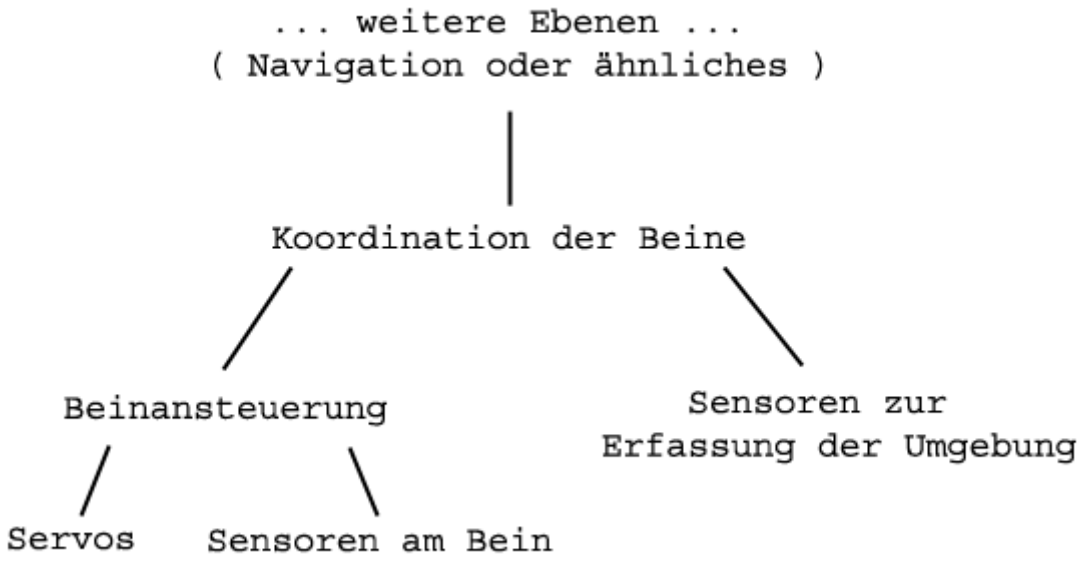
wesentlich schwerer als für einen Roboter mit sechs Beinen. Trotzdem ist auch das möglich wie der von mir entwickelte Roboter eindrucksvoll demonstiert.

2. Konzept

Viele Laufroboter werden von einem zentralen Computer gesteuert. Er liest die Sensordaten ein, verarbeitet sie und steuert die Mechanik an. Dies hat den großen Nachteil, dass der Prozessor sehr leistungsfähig sein muss, da er sehr viele Dinge auf einmal erledigen muss.
Ich habe versucht, mich mit der Konstruktion meines Laufroboter mehr an dem dezentralen Prinzip der Steuerung in der Natur zu orientieren [Sch03]. Dies bedeutet, dass nicht ein zentraler Prozessor die Steuerung des Roboters übernimmt, sondern die Aufgaben auf mehrere Prozessoren verteilt werden.
Ein ganz klares Ziel bei der Konstruktion war, dass der Roboter auf jeden Fall autonom agieren können sollte, also nicht zum Beispiel über Funk an einen externen PC angeschlossen ist und von dort seine Befehle erhält.

2.1 Schichtenmodell

Die Aufteilung der Aufgaben im Laufroboter erfolgt nach einem Schichtenmodell. Dies soll eine größtmögliche Autonomie ermöglichen.



Die unterste Schicht stellt die Bewegung der Beine dar. Pro Bein kommt ein Mikroprozessor zum Einsatz, der die Servos ansteuert und die Sensoren am Bein ausliest.
Die nächste Schicht besteht aus einem Mikroprozessor, der die Bewegung der Beine koordiniert und für die „globalen“ Sensoren des Roboters (Entfernungsmesser, Ausrichtung im Raum, Neigung, evtl. Temperatur usw.) zuständig ist. Diese beiden Schichten sind für die Bewegung des Roboters

zuständig. Weitere Schichten darüber könnten sich dann mit der Bewegungsplanung, Navigation usw. befassen.
Stößt eine Schicht auf ein Problem (zum Beispiel ein Hindernis oder einen Abgrund), so versucht zuerst nur der betroffene Mikroprozessor das Problem zu lösen. Ist dies nicht möglich (Bein findet keinen Boden mehr zum Aufsetzten, da der Roboter an einem Abgrund steht), so wird die nächst höhere Schicht informiert und diese versucht wiederum das Problem zu lösen usw. (der Roboter könnte sich kurz rückwärts bewegen und dann in eine andere Richtung weiterlaufen). Auf diese Weise werden zum Beispiel leichte Unebenheiten im Boden und ähnliches von den Beinprozessoren ausgeglichen, ohne dass die anderen Prozessoren etwas davon „bemerken“ und sich so um andere Probleme kümmern können. Das System kann so sehr flexibel auf einen unebenen bzw. unbekannten Untergrund reagieren.

Ein großer Vorteil dieses Systems ist die Unabhängigkeit der Prozessoren. Will man zum Beispiel die Prozessoren einer Ebene erweitern, so muss man nur dafür sorgen, dass die Software kompatibel zum Vorgänger ist, ansonsten kann man aber fast alles ändern, ohne dass man die anderen Ebenen anpassen muss. Würde man hingegen nur einen zentralen Prozessor benutzen und es würde sich herausstellen, dass die Rechenleistung nicht ausreicht, so müsste man das ganze Programm erst wieder an den neuen Prozessor anpassen.

Allerdings entsteht bei dieser Art der Aufteilung der Aufgaben ein Problem : die Mikrocontroller müssen ständig miteinander kommunizieren, um sich gegenseitig „abzustimmen“. Daher ist ein komplexes Netzwerk notwendig, welche eine Kommunikation aller beteiligten Prozessoren untereinander ermöglicht. Es müssen Befehle definiert werden, welche die einzelnen Prozessoren ausführen usw. .

2.2 Statisches/ dynamisches Gehen

Man kann das Gehen grob in die zwei grundlegenden Formen des „dynamischen Gehens“ sowie des „statischen Gehens“ einteilen. Beim statischen Gehen befindet sich der Körper immer in einem statischen Gleichgewicht, dass heißt, der Schwerpunkt befindet sich über den Füßen. Dadurch kann der Körper nicht umfallen. Um einen Schritt zu machen, wird der Schwerpunkt so verlagert, dass ein Bein frei ist. Dieses Bein tastet sich dann nach vorne, bis es einen festen Halt gefunden hat. Dann wird der Schwerpunkt wiederum verlagert, so dass ein anderes Bein nach vorne gesetzt werden kann. Diese Art der Bewegung hat den Vorteil, dass sie zu jedem Zeitpunkt angehalten werden kann, ohne dass der Körper umfallen würde. Allerdings ist statisches Gehen sehr langsam, da immer erst der Schwerpunkt verlagert werden muss usw. .
Wesentlich schnellere Bewegungen ermöglicht das dynamische Gehen. Im Gegensatz zum statischen Gehen ist hier der Schwerpunkt meist außerhalb des von den auf dem Boden stehenden Füßen aufgespannten Schwerpunktdreiecks. Es kann sogar soweit gehen, dass kein Bein auf dem Boden aufsitzt, sondern sich alle Beine in der Luft befinden (beim schnellen Rennen zum Beispiel). Bei dieser Art der Bewegung wird mit Körperschwingungen gearbeitet. Dabei muss kein statisches Gleichgewicht gegeben sein, um einen Fuß anheben zu können. Vielmehr schwingt der Körper in die

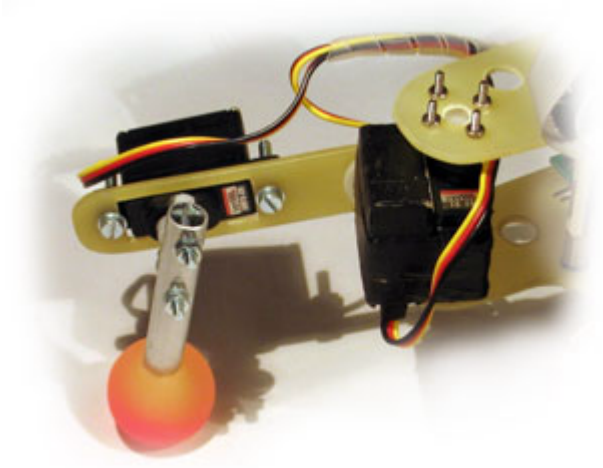
Richtung des aufgesetzten Beins. In der Zeit, die vergeht bis der Körper wieder zurückschwingt, kann jetzt das andere Bein bewegt werden.
Hielte man allerdings diesen Bewegungsablauf zu einem beliebigen Zeitpunkt an, so würde der Körper umfallen. Versuchen Sie zum Beispiel einfach mal beim normalen Laufen kurz vor dem Aufsetzen eines Fußes in der Bewegung inne zu halten. Es ist nicht möglich. Daher spricht man beim dynamischen Gehen auch von einem kontrollierten Fallen von einem Bein auf das andere. Dabei können große Kräfte an den Beinen auftreten. Diese müssen über eine flexible Struktur der Beine abgeleitet werden. Dies ist mit technischen Mitteln nur sehr schwer nachzubilden. Beim statischen Gehen hingegen kann die Struktur wesentlich einfacher sein, da keine plötzlichen Kräfte auftreten können.

Dynamisches Gehen ist zudem ohne Wissen über den Untergrund nur schwer möglich. Man kann dies sehr einfach selbst ausprobieren. Im Normalfall fallen wir beim Gehen von dem einen Bein auf das andere. Wenn man aber versucht, mit geschlossenen Augen über einen unbekannten, unebenen Untergrund zu laufen, so wird man immer mehr dazu übergehen, den Schwerpunkt auf ein Bein zu verlagern und sich mit dem anderen vorzutasten.
Es ist leider sehr schwer, einem Roboter das nötige Wissen über den Untergrund ohne optische Hilfsmittel (wie zum Beispiel Kameras) zu vermitteln.
Daher habe ich mich entschieden, meinen Roboter statisch gehen zu lassen, da ein dynamisches Gehen einen wesentlichen größeren Arbeits- und Zeitaufwand bei der Konstruktion bedeutet hätte.

3. Mechanik

Die Mechanik orientiert sich an dem Aussehen von Echsen mit nach außen abgewinkelten Extremitäten. Es ist diesem Vorbild aber nur sehr frei nachempfunden und ergibt sich hauptsächlich aus dem benötigen Platz für die Servos und die Platinen.

3.1 Konstruktion der Beine

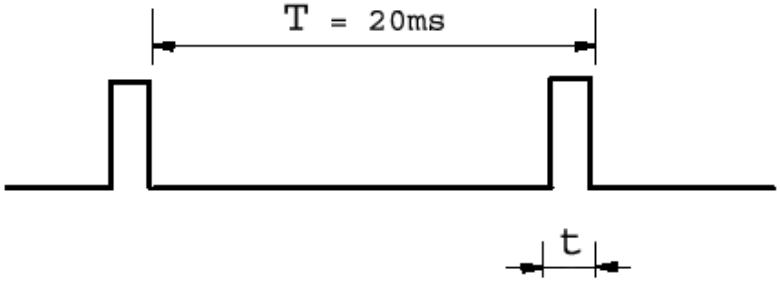


Konstruktion der Beine

Mit der Entscheidung für ein statisches Gehen hatte sich auch die Frage nach der Konstruktion der Beine wesentlich vereinfacht. Da beim statischen Gehen keine plötzlichen Kräfte auftreten können, müssen die Beine nur die Last des Roboters tragen.
Um ein flüssiges Laufen zu ermöglichen, entschied ich mich für eine Konstruktion mit drei Freiheitsgraden pro Bein. Es wäre auch möglich, die Beine nur mit zwei Freiheitsgeraden zu bauen, allerdings „schleifen“ die Beine dann immer über den Boden, da der Bewegungsradius sehr eingeschränkt ist.

3.2 Servos

Die Bewegung der Beine erfolgt über Servos. Servos werden häufig im Modellbau für die Lenkbewegungen von Rudern bzw. Rädern eingesetzt. Sie werden aber auch in der Industrie eingesetzt, wenn es auf genaue Positionierung ankommt (zum Beispiel die motorisierte Blendensteuerung bei Fotoapparaten).



Die Ansteuerung erfolgt über ein pulsweitenmoduliertes Signal. Alle 20 Millisekunden wird ein Impuls von t = 1 bis 2 ms (bei einigen Herstellern auch 0,8 bis 2,2 ms) gesendet. Bei 1,5 ms steht das Servo genau in der Mitte, bei einem Impuls von 1 ms auf der einen Seite, bei einem Impuls von 2 ms auf der anderen Seite. Versucht man von außen die Position zu verändern, so regelt die interne Elektronik dagegen und versucht die Position, die vom Steuersignal vorgegeben ist, wieder einzustellen.
Auf diese Weise wird die Position selbst dann gehalten, wenn von außen eine Kraft auf das Servohorn einwirkt.

Aufgrund ihrer einfachen Ansteuerung, der hohen Kraft und der kompakten Bauweise boten sich Servos geradezu für diese Anwendung an.

3.3 Körper

Der Aufbau des Körpers ist recht einfach gehalten. Er besteht im wesentlichen aus zwei GFK (**G**las**f**aserverstärker **K**unststoff) Platten, zwischen denen die Servos befestigt sind. Auf bzw. zwischen diesen beiden Platten sind die Platinen für die Steuerung des Roboters befestigt.
Das Design ist eher zweckmäßig als schön. Wenn der Roboter dann einmal ganz fertig ist, bekommt er auch noch eine ansprechendere Verkleidung.

4. Elektronik

Die Elektronik teilt sich auf mehrere Platinen auf : zum einen pro Bein jeweils eine Beinsteuerung mit je einem Mikroprozessor, zum anderen eine Erweiterungsplatine, deren Mikroprozessor für die Koordination der Bewegung zuständig ist. Es ist mittlerweile schon die zweite Version der Elektronik, die sich auf dem Roboter befindet, da die erste noch nicht ausgereift genug war und ein paar Fehler enthielt.

Die Layouts für die verschiedenen Platinen wurden mit der Freeware Version des Layoutprogramms Eagle 4.09r2 erstellt. [CCG04]

4.1 Der Mikroprozessor

Bei der Wahl der Mikroprozessoren für mein Projekt habe ich mich für die AVR Reihe von Atmel [Atmel01] entschieden, da diese einige interessante Eigenschaften haben und ich schon mit ihnen gearbeitet hatte und mich deshalb schon etwas damit auskannte:

- 8 – Bit Prozessor
- Echte RISC Architektur, das heißt bis zu 1 Mio. Befehle bei 1 Mhz Quarzfrequenz
- Flashspeicher von 1kB bis zu 128 kB, internes EEprom bis 4 kByte und bis zu 4 kByte RAM
- Komfortable Registerarchitektur mit 32 gleichwertigen Registern
- Optimiert für Hochsprachen Verwendung
- Sehr guter Open Source C – Compiler verfügbar (GCC Portierung)
- Umfangreiche interne Peripherie: 10 Bit Analog/Digital Wandler, 8/16 Bit Timer, SPI und TWI Schnittstelle, UART, Brown Out Dection usw.
- Versorgungsspannung 4,5 bis 5,5 Volt
- Geringe Stromaufnahme
- Bis zu 16 Mhz Taktfrequenz

Ich habe mich dann letztendlich für die ATmega8 und ATmega16 Typen entschieden. Diese haben 8 bzw. 16 kByte Flashspeicher, 10 Bit A/D Wandler und 1 kByte RAM. Zuerst wurden nur Atmega8 Prozessoren eingesetzt, aber es hat sich herausgestellt, dass der Programmspeicher von 8 kByte für die Beinansteuerung nicht ausreichte, so dass ein Umstieg auf das nächst leistungsfähigere Modell nötig war.

4.2 Beinsteuerung

Die Ansteuerung der Beine besteht aus zwei kleinen Platinen:
Der Analogteil dient der Verstärkung und Filterung der Signale der Servostrommessung. Dies ist über einen Tiefpass realisiert, der das Signal glättet, und einen Operationsverstärker, der das Signal so verstärkt, dass es gut messbar ist. [Sch04]

Da die Servos mit einer anderen Spannung als der Mikrocontroller betrieben werden, war es notwendig, für die Signalleitung der Servos eine Pegelanpassung durchzuführen. Dazu wird mit einem zusätzlichen Transistor ein Open Collector Ausgang geschaffen. So wird außerdem verhindert, dass Störungen bei den Servos auf den Mikroprozessor übertragen werden können.

Außerdem befindet sich auf der Platine des Analogteils noch ein Stecker, über den der I2C Bus des Mikrocontroller abgegriffen wird. Daran könnte man zum Beispiel über einen I2C Portexpander später noch zusätzliche Sensoren am Bein anschließen.

Auf der anderen Platine ist der Mikrocontroller mit externer Beschaltung untergebracht. Weiterhin befinden sich noch ein Treiber für das RS485 Netzwerk und einige LEDs zur Statusanzeige mit auf der Platine. Der Stecker für das RS485 Netzwerk ist so ausgelegt, dass man mehrere dieser Platinen zusammenstecken kann, wenn man das möchte.
Die Referenzspannung für den Analog/Digital Wandler kann über ein Potentiometer eingestellt werden.

Die Verbindung der beiden Platinen erfolgt über ein 10-poliges Flachbandkabel. Sie haben außerdem die gleiche Größe, so dass man sie sehr einfach stapeln kann.

4.3 Erweiterungsplatine

Die Erweitungsplatine enthält neben dem Mikroprozessor und dem RS485 Treiber noch Anschlüsse für ein LC – Display, auf dem Informationen über die aktuelle Situation angezeigt werden können, weiterhin für zwei weitere Servos und einen Infrarot Entfernungsmesser.
Die beiden Servos und der Entfernungsmesser bilden den „Kopf“ des Roboters. Der eingesetzte Entfernungsmesser ist ein GP2D120 von Sharp. Dieser liefert an einem analogen Ausgang eine Spannung, abhängig von der gemessenen Entfernung. Der GP2D120 kann Entfernung von 4 bis ca. 40 cm messen. Leider ist die ausgegebene Spannung nicht proportional für Entfernung, so dass dies im Mikrocontroller umgerechnet werden muss.
Um den gesamten Bereich vor dem Roboter erfassen zu können, wurde der GP2D120 auf zwei Servos montiert, welche ein Schwenken in zwei Achsen zulassen, so dass der gesamte Bereich abgetastet werden kann.

Auch auf dieser Platine befindet sich ein Anschluß für den I2C Bus, um weitere Sensoren anschließen zu können. An diesem Bus sind auch zwei I2C EEproms angeschlossen, die zum Speichern von Sensordaten oder ähnlichem gedacht sind, in der aktuellen Version aber noch nicht benutzt werden. Außerdem sind die nicht benutzten Pins des Mikrocontrollers auf Stiftleisten geführt worden, um das System später auch noch erweitern zu können.

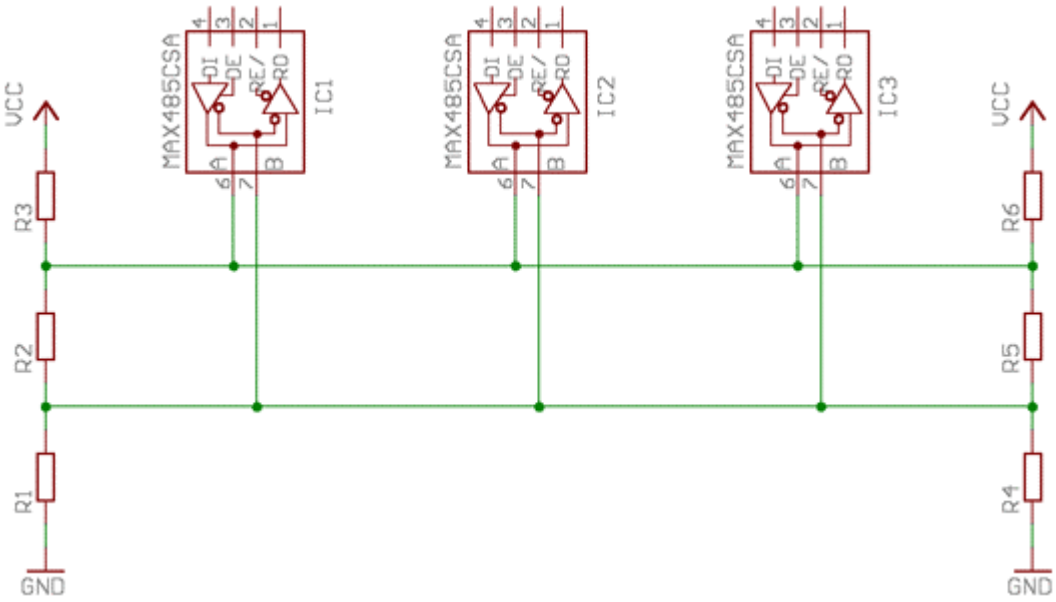
5. Software

Die Software für die einzelnen Module wurde mit AVR-GCC geschrieben, einer Portierung des aus der Unix Welt bekannten GCC C-Compilers. Der Compiler ist als fertiges Programmpaket für Windows erhältlich [WinAVR04], so dass die Entscheidung relativ leicht gemacht wurde.

Es lassen sich damit sehr leistungsfähige Programme erstellen. Man muss allerdings beachten, dass die Programme später auch auf Mikroprozessoren laufen sollen, das heißt, man muss immer versuchen, das Programm so Ressourcen sparend wie möglich zu gestalten, da die Mikroprozessoren nur einen begrenzten Programmspeicher haben. Dies hat sich gezeigt, als für die Routine, die Koordinaten in Winkel umrechnet, einige Gleitkommaoperationen (32-Bit) gebraucht wurden. Da der Mikroprozessor intern nur mit 8-Bit rechnet, müssen alle diese Operationen per Software nachgebildet werden, was sich in einem erhöhten Speicherverbrauch niederschlägt. Außerdem brauchen solche Operationen dann natürlich viel länger als vergleichbare 8-Bit Operationen.
Die einzige Möglichkeit, dies zu beschleunigen, wäre die Verwendung von Tabellen, in denen die entsprechenden Werte einfach nur nachgeschaut werden können. Allerdings ist diese Methode noch sehr viel speicheraufwendiger, da riesige Tabellen gebraucht werden würden, so dass dies nicht praktikabel ist.

5.1 Netzwerk

Das Netzwerk im Roboter besteht aus einem RS485 Netzwerk. Es ist als Bus mit gleichwertigen Teilnehmern realisiert, dass heißt, es gibt keinen definierten Master, der den gesamten Datenverkehr auf dem Bus kontrolliert.



RS 485 Busstruktur
(Die ICs Max485 sind Treiberbausteine [MIP03])

Es ist für RS 485 kein festes Protokoll definiert, es ist lediglich die physikalische Übertragung der Daten vorgegeben. Um alles weitere muss sich der Anwender selbst kümmern.

Um die einzelnen Mikrocontroller miteinander kommunizieren zu lassen, war ein Netzwerkprotokoll notwendig, welches Kollisionen der Daten auf dem Bus verhindert und Fehler in der Übertragung erkennt.
Zudem musste sichergestellt werden, dass nicht zwei Mikroprozessoren gleichzeitig senden, da es sonst zu nicht definierten Zuständen auf dem Bus kommen würde und die Daten verloren gehen würden.
Das Netzwerkprotokoll ist folgendermaßen aufgebaut:

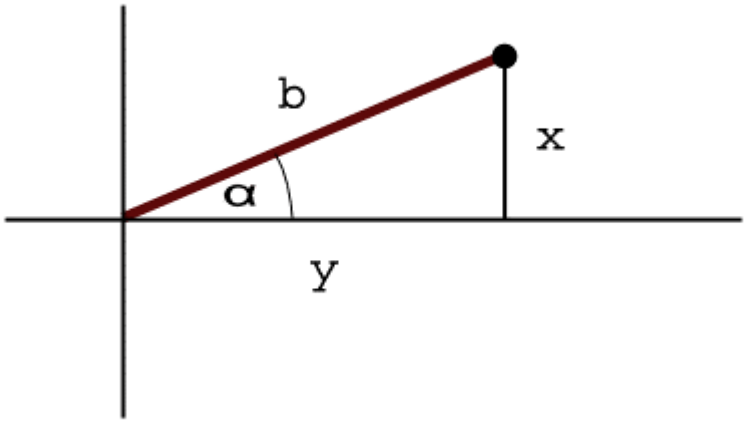
Prebyte – Preämblebyte
Das Prebyte stellt eine Art Vorspann dar. Es dürfen beliebig viele Prebytes gesendet werden.
Syncbyte – Syncronisationsbyte
Mit dem Syncbyte startet die Übertragung. Es darf daher auch nur einmal gesendet werden.
Empfänger
Das nächste Byte enthält den Empfänger der Nachricht. Die Mikrocontroller vergleichen das Byte mit ihrer eigenen Adresse. Nur wenn sie übereinstimmen oder die Nachricht an alle gerichtet ist, wird, nachdem die Nachricht vollständig empfangen ist, ein internes Bit gesetzt und der Mikrocontroller wertet die Nachricht aus.
Sender
Der Absender der Nachricht.
Länge
In diesem Byte steht die Anzahl der folgenden Datenbytes.
Daten .....
Die eigentlichen Datenpackete, die übertragen werden. Hier können bis zu 256 Byte gesendet werden.
CRC-Highbyte
Das CRC-Highbyte ist das höherwertige Byte der 16-Bit CRC Checksumme.
CRC-Lowbyte
Niederwertiges Byte der CRC Checksumme.

Dieses Protokoll stellt allerdings nur die unterste Ebene der Netzwerkkommunikation dar. Darauf aufgesetzt sind noch weitere Schichten, welche sich mit der sinnvollen Auswertung der empfangenen Nachrichten beschäftigen.

Um die einzelnen Beine zu steuern, wurden Befehle definiert, welche die einzelnen Controller dann selbständig ausführen.

5.2 Umrechnung von Koordinaten in Winkel für die Ansteuerung der Servos

Ein großer Teil der Rechenleistung der Controller für die Beinsteuerung wird von dieser Routine beansprucht. Um die Bewegungen des Roboters sinnvoll programmieren zu können, kann man nicht mit Winkeln der Servos arbeiten. Wenn man zum Beispiel eine Bewegung des Beins am Roboter entlang mit gleichbleibendem Abstand zum Körper betrachtet, so wird schnell klar, warum. Es ist sehr schwer, diese Bewegung mit den Winkeln der drei Servos zu beschreiben, da sich alle drei ändern. Betrachtet man die gleiche Bewegung aus der Koordinatenform heraus, so wird sie viel einfacher. Es ändert sich nur der X-Wert, Y- und Z-Wert bleiben gleich.
Die Umrechnung von Koordinaten in Winkel ist allerdings nicht ganz einfach. Sie erfolgt in zwei Schritten: Zuerst wird der Winkel berechnet, um den das Bein von oben gesehen gedreht werden muss.



Bein von oben gesehen

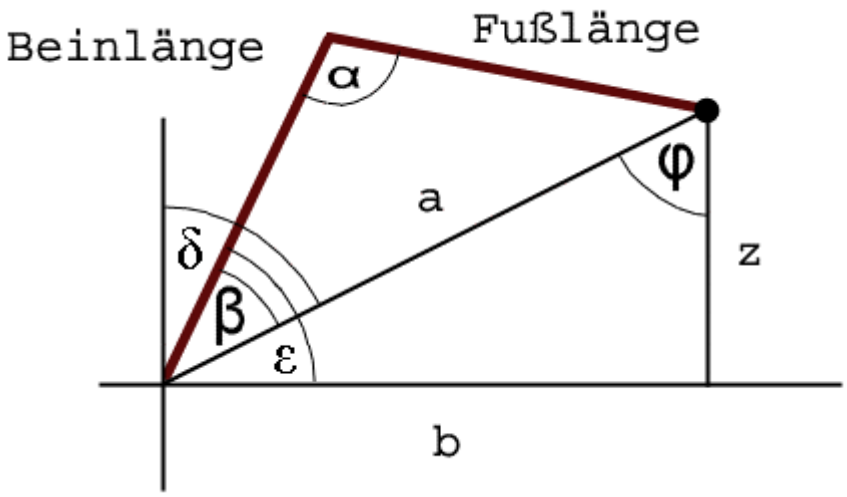
Der Winkel von α lässt sich sehr einfach über den Arcussinus ausrechnen:

$$b = \sqrt{x^2 + y^2}$$

$$\alpha = \arcsin\left(\frac{x}{b}\right)$$

Man hätte α direkt auch aus x und y berechnen können, allerdings braucht diese Routine auf dem Mikroprozessor länger (die Berechnung von b ist dabei egal, da der Wert für b im zweiten Teil sowieso noch einmal gebraucht wird), so dass ich mich für die Variante mit dem Arcussinus entschieden habe.

Dann werden im nächsten Schritt auch die beiden Winkel für die zwei übrig bleibenden Servos berechnet:



Bein von der Seite gesehen.

(Die untere Linie des Koordinatensystems entspricht dem Verlauf der dunkelroten Linie im oberen Bild)

Den benötigten Winkel von **α** kann man über den Kosinussatz ausrechnen:

$$a = \sqrt{b^2 + z^2}$$

$$\alpha = \arccos\left(\frac{Beinlänge^2 + Fußlänge^2 - a^2}{2 * Beinlänge * Fußlänge}\right)$$

Mit diesem Winkel lässt sich wiederum über den Sinussatz der Winkel für **β** ermitteln:

$$\beta = \arcsin\left(\frac{Fußlänge}{a} * \sin\alpha\right)$$

Aus den Strecken **b** und **a** kann man **φ** ausrechnen:

$$\varphi = \arcsin\left(\frac{b}{a}\right)$$

Da **δ** und **φ** Wechselwinkel sind, sind sie gleich groß.

Und damit wiederum lässt sich dann der eigentlich benötigte Winkel **ε** ermitteln:

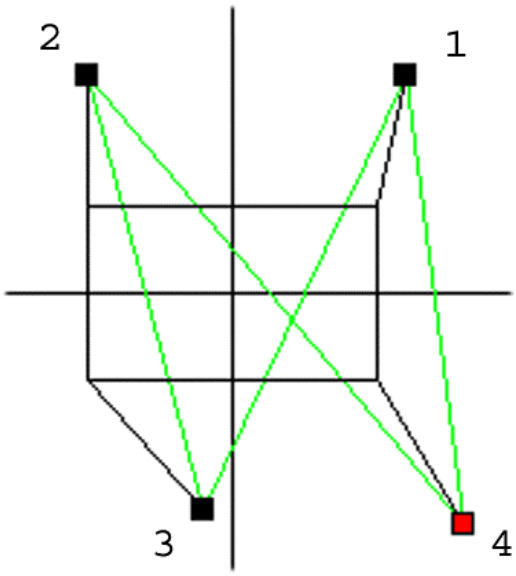
$$\epsilon = 90° - \delta + \beta$$

Es ist eigentlich erstaunlich, dass die 8-Bit Prozessoren der AVR Reihe dies überhaupt ausrechnen können, da sie nur addieren, subtrahieren und multiplizieren mit 8-Bit Zahlen (0 bis 255) beherrschen.

So kommt es dann, dass es bei 16 Mhz Taktfrequenz ca. 2,5 bis 3,5 ms dauert, die benötigten Winkel auszurechnen. (Dies entspricht ca. 40 000 bis 60 000 Befehlen, die der Prozessor ausführen muss)

5.3 Schwerpunktberechnung

Um laufen zu können, muss der Mikroprozessor, der für die Koordination der Bewegung zuständig ist, wissen, wann welches Bein gehoben und nach vorne gesetzt werden kann. Dies lässt sich am einfachsten über Schwerpunktdreiecke bestimmen.
Damit ein Bein gehoben werden darf, muss der Schwerpunkt in einem Dreieck liegen, das von den drei anderen Beinen aufgespannt wird.
Folgende Grafik soll das einmal verdeutlichen:



Schematische Darstellung einer Draufsicht des Roboters

Der Mittelpunkt des Koordinatensystems stellt den Schwerpunkt des Roboters dar. Die schwarzen bzw. roten Rechtecke stellen die Aufsetzpunkte der Beine auf dem Boden dar. Die zur Berechnung benötigten Linien sind grün dargestellt.
Man sieht, dass Bein 1, 2 und 3 ein Dreieck bilden, in dem der Koordinatenursprung liegt. Dies wäre also eine stabile Position, so dass der Roboter das Bein 4 anheben könnte.
Allerdings bilden auch Bein 2, 3 und 4 ein Dreieck, in dem der Koordinatenursprung liegt. Das heißt, es könnte genau so gut auch das erste Bein angehoben werden.

Die Berechnung im Mikrocontroller sieht dann so aus, dass der Abstand des Schnittpunktes der grün eingezeichneten Linien mit der X-Achse berechnet wird. Dann wird für jedes Bein geprüft, ob die anderen Beine ein Schwerpunktdreieck aufspannen. Dies wird für alle vier Beine durchgeführt und dann überprüft, welches Bein letztendlich angehoben wird.

6. Ausblick

Der Ausblick könnte eigentlich der größte Teil dieser Arbeit werden, da mir sehr viele neue Ideen für Weiterentwicklungen und Verbesserungen gekommen sind, die ich leider nicht bis zum Wettbewerb umsetzen konnte.

Ein grundlegendes Problem bei dem jetzigen Roboter ist die sehr spärliche Bestückung mit Sensoren. Es wird lediglich der Strom der Servos und der ungefähre Abstand von Hindernissen gemessen. Der Roboter bräuchte auf jeden Fall noch Sensoren für Berührungen an den Beinen, einen Sensor, der ihm die Lage im Raum mitteilt (den Winkel zur Horizontalen und über Magnetfeldsensoren den Winkel zu Norden).
Weiterhin müssten an den Beinen Kraftmesser installiert werden, über die man den genauen Druck ermitteln kann, mit dem das Bein auf dem Boden steht. Die Messung des Stroms der Servos liefert zwar einen ungefähren Wert, für genaue Messungen ist die Methode aber leider viel zu ungenau.
Als nächstes könnte man die Controller, die die Beine steuern, durch 16-Bit Mikrocontroller ersetzen. Die Berechnung der Winkel der einzelnen Servos aus Koordinaten benötigt einiges an Zeit. Diese Routinen ließen sich durch 16-Bit Mikrocontroller sehr beschleunigen.
Ein weiterer großer Teil der Rechenleistung wird durch das Netzwerk bzw. das Protokoll verschlungen, da jeder Prozessor immer jede Nachricht empfangen muss um zu schauen, ob sie nicht vielleicht für ihn ist. Dieses Netzwerk könnte man vereinfachen, indem man ein anderes Bussystem wie zum Beispiel CAN benutzt. Der CAN Controller würde das Empfangen der Nachrichten selbständig ausführen und den Mikrocontroller nur dann beanspruchen, wenn wirklich eine Nachricht für ihn eintrifft.
Mit der durch diese Maßnahmen gewonnenen Rechenzeit ließen sich mehr Sensoren anschließen und auswerten. Ein weitere Idee wäre, den Beinen eine Art Gedächtnis zu verschaffen, in Form einer kleinen Karte, die die Umgebung um den Roboter darstellt. Findet nun ein Bein ein Loch oder so etwas in der Art, dann trägt es dies in die Karte ein und das nächste/ hintere Bein kann dann entsprechend darauf reagieren.
Ein weiteres Ziel wäre die Erweiterung der „Intelligenz“ des Roboters. Dies könnte durch das Hinzufügen weiterer Schichten erfolgen. Diese Schichten müssten sich dann gar nicht mehr um das Laufen an sich kümmern, sondern könnten Aufgaben wie eine sinnvolle Navigation (gezieltes Erkunden der Umgebung) übernehmen.

Wenn der Roboter einmal vollständig fertig ist, ließe er sich universell für viele verschiedene Aufgaben einsetzen. Auf das Grundgerüst, bestehend aus der Mechanik und den ersten beiden Schichten, ließen sich weitere Schichten aufsetzen, die sich dann zum Beispiel um die Inspektion von schwer zugänglichen Rohrleitungen, um die Erkundung von fremden Planeten oder um Minenräumung im unwegsamen Gelände „kümmern“, um nur ein paar zu nennen.

7. Literatur

[Atmel01] Atmel Corporation, *Datenblatt : ATmega8*, http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf, 2001

[Atmel01] Atmel Corporation, *Datenblatt : Atmega16*, http://www.atmel.com/dyn/resources/prod_documents/doc2466.pdf, 2001

[Bra04] F. Brall, *Roboterforum*, http://www.roboternetz.de, Stand: 11.1.2004

[Bur04] M. Burrer, *Elektronik/Roboter Forum*, http://www.elektronik-projekt.de, Stand: 10.1.2004

[CCG04] CadSoft Computer GmbH, *Homepage des Herstellers der Layoutsoftware Eagle*, http://www.cadsoft.de, Stand : 5.1.2004

[Leo97] R. Leonhardt, *Ansteuerung einer sechsbeinigen Gehmaschine*, Studienarbeit-Nr. 1617, Universität Stuttgart Fakultät Informatik, 1997

[MIP03] Maxim Integrated Products, *Datenblatt : MAX485CSA*, http://pdfserv.maxim-ic.com/en/ds/MAX1487-MAX491.pdf, 2003

[BLG04] Beta LAYOUT GmbH, Homepage des PCB-POOL, http://www.pcb-pool.com/ppde/info.html, Stand : 9.1.2004

[Sch03] J.-U. Schamburek, *Bewegungssteuerung bei Insekten*, Seminar „Informationsverarbeitung in Lebewesen“, Universität Karlsruhe, 2003

[Sch04] A. Schwarz, *Mikrocontroller Forum*, http://www.mikrocontroller.net/forum/, Stand: 9.1.2004

[Sch04] P. Schnabel, *das Elektronik-Kompendium*, http://www.elektronik-kompendium.de/, Stand: 5.1.2004

[Wie04] A. Wiedekind-Klein, *Elektronik/Roboter Forum*, http://www.roboterwelt.de, Stand: 9.1.2004

[WinAVR04] *Homepage des Open Source Projektes WinAVR*, http://winavr.sourceforge.net/, Stand : 9.1.2004