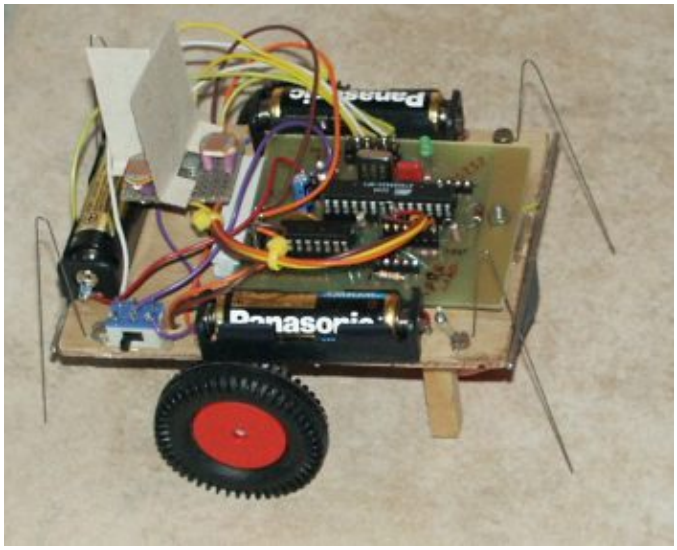
by Katja and Guido Socher
<katja/at/linuxfocus.org
guido/at/linuxfocus.org>

About the authors:

Katja ist die deutsche Redakteurin von LinuxFocus. Sie mag Tux, Computergraphik, Film, Fotografie und das Meer. Ihre Homepage befindet sich hier.

Guido ist ein langjähriger Linuxfan und er mag Linux, weil es einem Wahlmöglichkeiten und Freiheit gibt. Man kann Lösungen entsprechend seinen eigenen Bedürfnissen wählen und entwickeln.

Bau eines autonomen Lichtfinder–Robots



Abstract:

In diesem Artikel beschreiben wir, wie man einen autonom handelnden Roboter mit Microcontroller baut, der immer versucht, zum hellsten Punkt zu laufen.

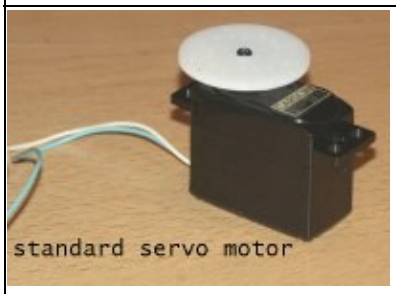
Einführung

Vor zwei Jahren haben wir in LinuxFocus einen "Linux gesteuerten gehenden Roboter" vorgestellt. Er war besonders in seinem Design, da er auf Beinen lief und keine konventionellen Motoren besaß. Dies war ein sehr interessanter Aspekt dieses Roboters, er war jedoch sehr langsam, verbrauchte viel Strom und erforderte viele spezielle Teile und Fähigkeiten, um ihn zu bauen.

Das Design unseres neuen Roboters ist ganz anders. Er ist billig und man kann ihn mit Teilen bauen, die es fast überall auf der Welt geben sollte. Es ist ein autonomer Roboter, der von einem AVR Microcontroller gesteuert wird. Als ein autonomer Roboter (der nicht von einer Person gesteuert wird), haben wir ihn so programmiert, daß er zur hellsten Stelle in einem Raum rennt.

Die Mechanik

Der kleine Getriebemotor von conrad

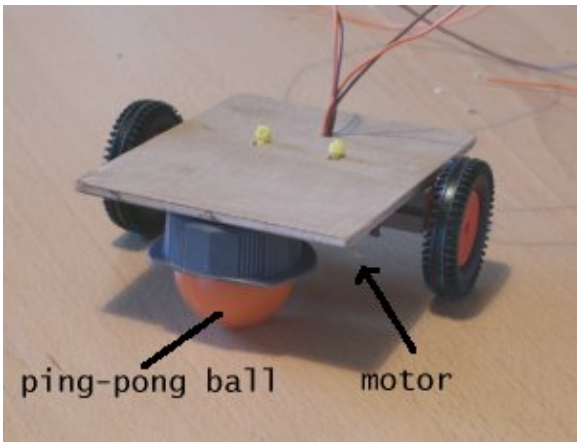


Ein modifizierter Standardservomotor, um als Motor arbeiten zu können. Dies ist wahrscheinlich die beste Lösung, aber wir hatten die Idee erst, als der Roboter schon fertig gebaut war.

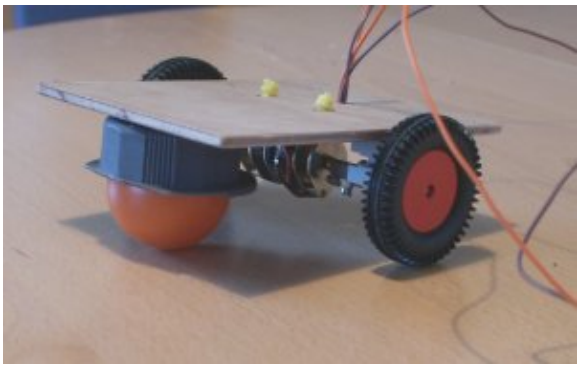
Der Roboter hat nur zwei Räder, die von zwei unabhängigen Motoren angetrieben werden. Das dritte Rad ist ein Tischtennisball. Dies ermöglicht es dem Roboter, sich auf der Stelle zu drehen. Wir haben Gummiräder von altem Spielzeug benutzt, aber du mußt nicht dein altes Spielzeug auseinander nehmen. Der Deckel eines Marmeladenglases mit einem Gummi drum, gibt ebenfalls ein sehr nettes Rad.

Für einen autonomen Roboter ist es natürlich wichtig, daß er mit Batterien arbeiten kann. Da der Microcontroller mit 4.5V läuft, müssen die Motoren auch mit 3−4.5V arbeiten. Sie dürfen auch nicht zu viel Strom ziehen, da sonst die Batterien und die Elektronik zu groß und zu schwer werden. Für das Design haben wir einen integrierten Motortreiberchip namens l293d benutzt. Der l293d Motortreiberchip kann Spitzenströme bis zu 0.5A verkraften. Die Motoren sollten deshalb im schlimmsten Fall weniger als 0.5A verbrauchen.

Wir haben zwei kleine Getriebemotoren von Conrad (www.conrad.de, Teilenummer: 242802) benutzt, aber du kannst auch jeden anderen kleinen Motor mit Getriebe benutzen. Tatsächlich denken wir jetzt, daß die beste Lösung ein Standard−Servomotor gewesen wäre, so wie sie für Fernsteuerungen von kleinen Booten, Autos oder Flugzeugen benutzt werden. Normalerweise können diese Servomotoren sich nur in einem bestimmten Winkel drehen, aber man kann die Getriebebox des Servos öffnen, den Stopper herausnehmen und das Potentiometer und die Elektronik entfernen. Es ist ein perfekter kleiner starker Motor und Servomotoren sind leicht zu bekommen.



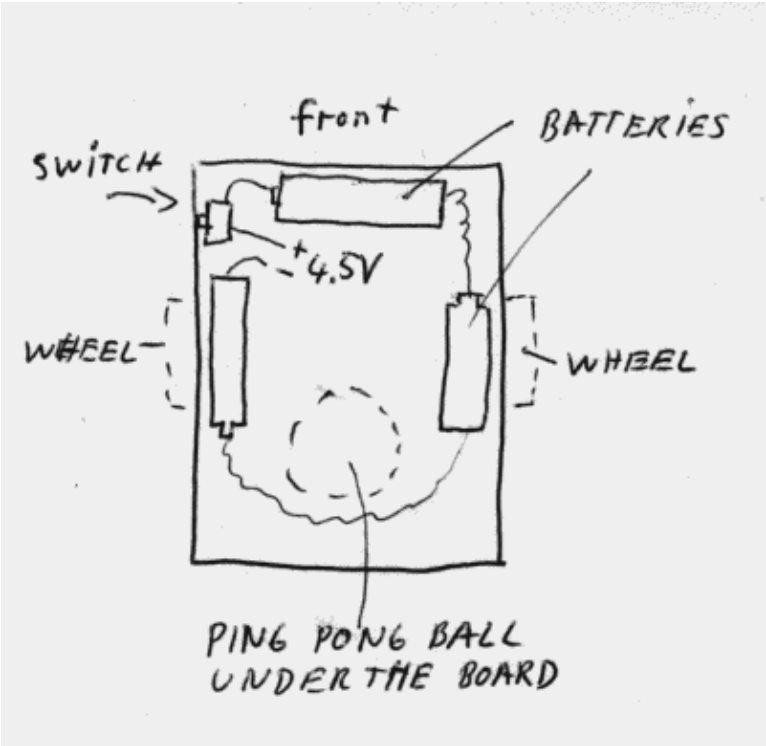
Um den Roboter zu bauen, montiere die Motoren unter ein kleines Holzbrett (12cm x 9cm) und bringe sie in der Mitte an, so daß die Hauptlast auf den beiden Achsen liegt. Das dritte Rad, der Tischtennisball, darf nur einen kleinen Teil des Gewichts des Roboters übernehmen, damit er leicht in seiner "Halterung" gleiten kann (siehe Abbildungen).



Die Halterung des Tischtennisballs ist der Deckel einer kleinen Plastikflasche, die gerade exakt die richtige Größe hatte.

Damit der Roboter läuft, haben wir drei AAA Batterien verwendet. Bringe die Batteriehalterungen wie unten gezeigt an. Die Batterien sind recht schwer, paß auf, daß die Hauptlast auf den Rädern liegt und nur ein

kleiner Teil auf dem Tischtennisball. Man kann einen An–/Ausschalter für den Roboter irgendwo an der Seite anbringen.

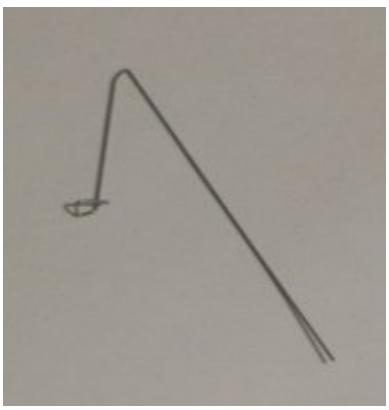


Sensoren

Wir geben unserem Roboter zwei Typen von Sensoren:

- Berührungssensoren: auf diese Weise weiß der Roboter, wenn er an ein Objekt gestoßen ist
- Lichtsensoren: damit der Roboter die hellste Stelle finden kann

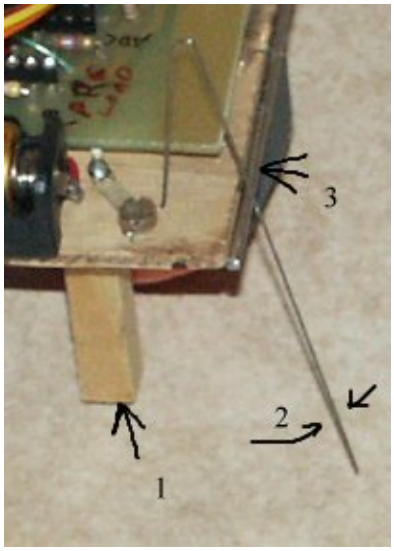
Die Berührungssensoren sind einfache Schalter aus Stahldraht. Wir haben sie so gebogen, wie im Bild unten zu sehen ist:



Die vier Berührungssensoren werden mit einer Schraube an den Ecken des Holzbretts befestigt.

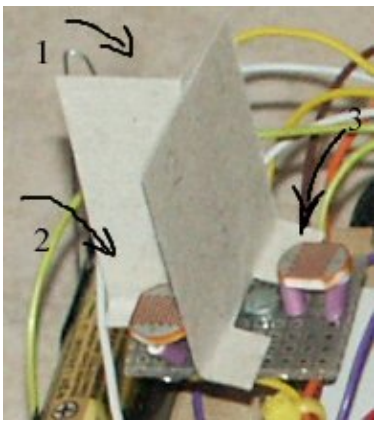
Wenn der Roboter an ein Objekt stößt, dann berührt der Stahldraht (2, siehe die Abbildung unten) einen zweiten Draht an dem Brett (3) und dies schließt eine elektrische Verbindung zwischen Stahldraht und Draht auf dem Holzbrett.

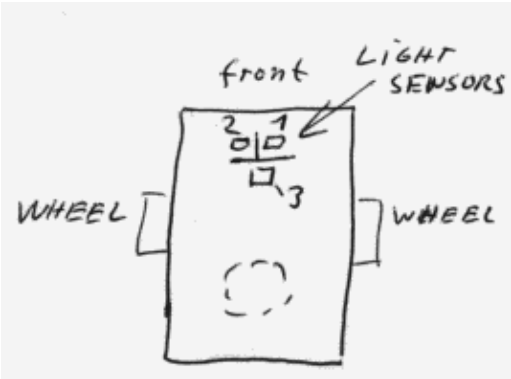
Um zu verhindern, daß der Stahldraht sich verbiegt, wenn der Tischtennisball nicht in seiner Halterung ist, haben wir ein kleines Holzstück (1) unter dem Brett hinzugefügt. Dieses Holzstück muß 5mm über den Boden sein, wenn der Tischtennisball in seiner Halterung sitzt.
Der Stahldraht sollte 5−7mm über dem Boden enden.



Die Lichtsensoren sind drei Fotowiderstände. Wir haben Pappe zwischen die Fotowiderstände gestellt, so wie im Bild unten gezeigt. Die Pappe erzeugt Schatten auf den Widerständen, wenn das Licht von der Seite kommt. Nur wenn das Licht genau von oben kommt, liefert es einen gleichen Betrag an Licht auf alle drei Sensoren. Durch Vergleichen der Werte der drei Sensoren kann der Roboter entscheiden, in welche Richtung er gehen soll.

Du kannst die drei Fotowiderstände auf ein kleines Experimentierbrett (diese Bretter mit vielen Löchern) löten und das ganze mit einer einzigen Schraube auf dem Roboter befestigen.

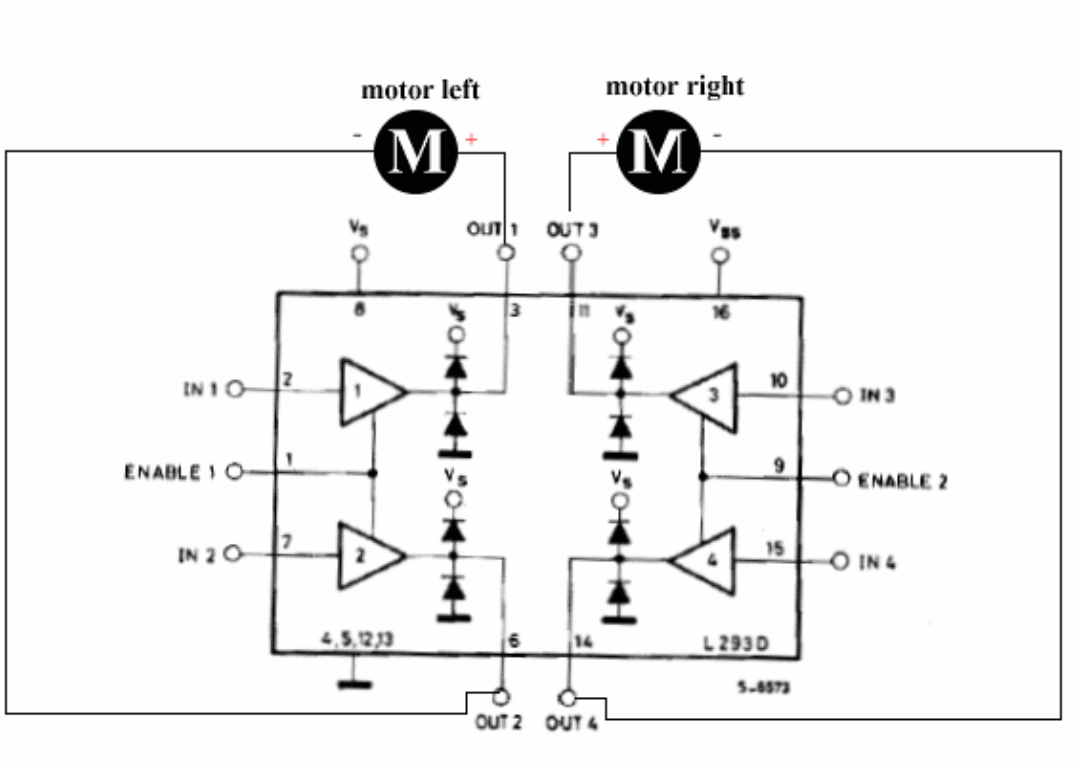




Wie man die Sensoren und die zwei Motoren mit der Schaltung und dem Microcontroller verbindet, wird weiter unten erklärt. Jetzt, wo die mechanischen Teile fertig sind, laßt uns das "Gehirn" des Roboters anschauen.

Die Schaltung

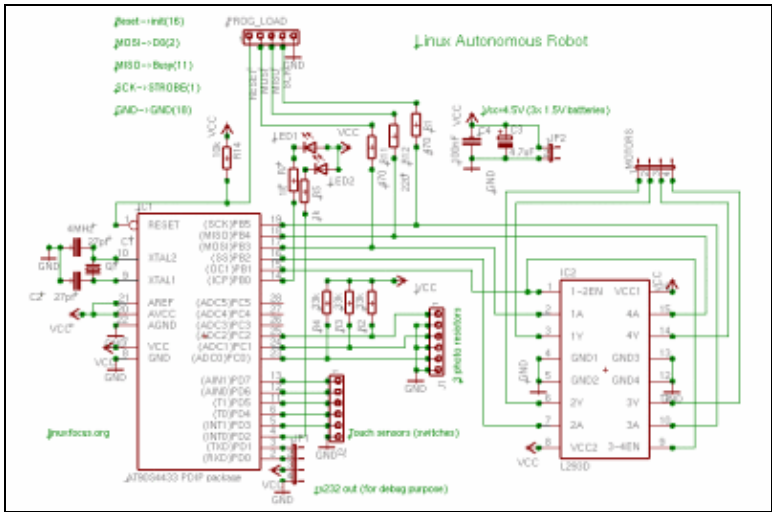
Wir benutzen einen AT90S4433 Microcontroller als "Gehirn" unseres Roboters, aber das "Gehirn" kann nicht direkt genug Kraft liefern, um die Motoren anzutreiben. Das ist der Punkt, wo der L293D Motortreiberchip ins Spiel kommt. Der Chip enthält vier digitale Ausgangsverstärker mit Extra−Schutzdioden, um gegen hohe, durch Spulen eines Motors induzierten Spannungen zu schützen. Jeweils zwei dieser Ausgangsverstärker können benutzt werden, um einen Motor anzutreiben. Auf diese Weise ist es möglich, den Motor links und rechtsrum zu drehen.



Wir packen einen Motor zwischen output 1 und output 2 und den anderen zwischen output 3 and output 4. Die enable pins des Chips können zur Steuerung der Geschwindigkeit der Motoren verwendet werden, wenn wir Pulse von variabler Länge an die enable pins senden.

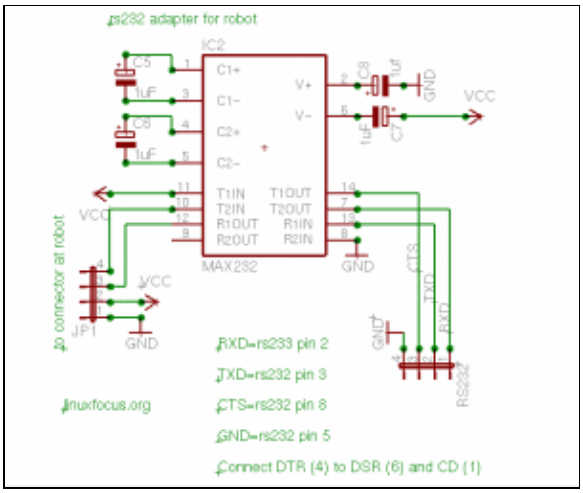
Der Rest der Schaltung ist sehr einfach: Wir benutzen wieder den Atmel AT90S4433 Microcontroller. Du kennst den Microcontroller bereits aus früheren LinuxFocus−Artikeln. Seine analogen Eingänge können zum Messen des Lichts der Fotowiderstände benutzt werden und die Berührungssensoren können direkt mit den digitalen Eingangsleitungen verbunden werden, wie unten gezeigt.

Mehr Informationen über die Details des Microcontrollers finden sich in Guidos März 2002 Artikel: Programmierung des AVR Microcontrollers mit GCC.



Die Schaltung arbeitet mit 4.5V. Drei AAA Batterien sind deshalb ausreichend, um den Roboter zum Laufen zu bringen.

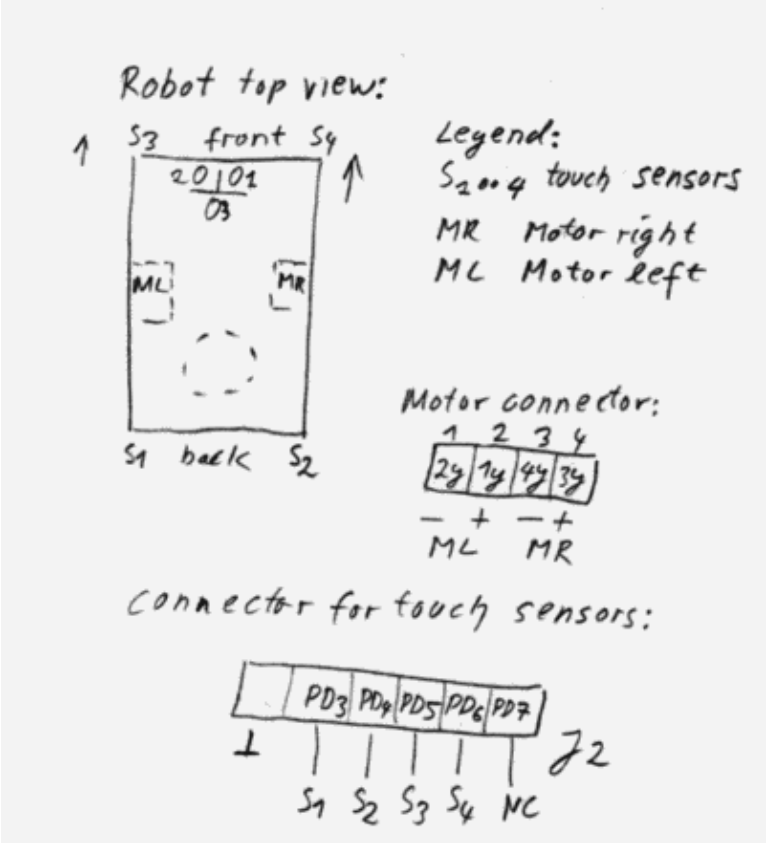
Jetzt wäre die Schaltung unseres autonomen Roboters fertig. Was machst du jedoch, wenn er nicht so läuft, wie erwartet, weil etwas mit der Software nicht stimmt? Du kannst nichts sehen. Du kennst die Werte der Lichtsensoren nicht, du weißt nicht, warum die Robotersoftware die eine oder andere Entscheidung getroffen hat. Was wir brauchen. ist eine Art Ausgabebildschirm oder –anzeige, um zu verstehen, was unser Roboter macht. Die RS232 Schnittstelle ist für diesen Zweck gut geeignet. Wir können die Werte von Variablen ausdrucken und könnten sogar mit dem Roboter kommunizieren. Wir wollen es nicht die ganze Zeit anschließen, aber wir brauchen es zum Debuggen. Es macht deshalb Sinn, den max232 und andere Teile, die für die RS232 Verbindung gebraucht werden, auf ein separates Board zu packen und nur dann anzuschließen, wenn es gebraucht wird:



Die kompletten Eagle Schaltungsdiagramme und Boardlayouts können am Ende des Artikels zusammen mit der Software für den Roboter heruntergeladen werden. Wir beschreiben das Boardlayout hier nicht. Du kannst es in Eagle sehen. Die Schaltung ist klein genug, so daß sie zwischen die Batterien paßt.

Unten ist eine Zeichnung, wo du sehen kannst, welcher Berührungssensor auf welcher Seite des Roboters mit welchem Pin in der Schaltung verbunden ist. Die Zeichnung zeigt auch, wie man die Motoren verbindet. Die Polarität der Motoren ist so gewählt, daß der Roboter vorwärts laufen würde (in die Richtung des Pfeils),

wenn +3V am "+"−pin angeschlossen wären und Erde (GND) am "−"−pin. 1y bis 4y sind die Namen der Pins auf dem l293d.



Die Software

Wir wollen hier nicht groß ins Detail gehen. Das Hauptprogramm kann in der Datei linuxrobot.c (Download der Software am Ende des Artikels) gefunden werden. Das Programm enthält eine Menge Kommentare und sollte für einen C−Programmierer leicht zu lesen sein. Die Hauptschleife überprüft zuerst die analogen Werte der Fotowiderstände durch dreimaliges Laufenlassen des internen analog zu digital Umwandlers des Microcontrollers in single shot Conversionsmode. Danach werden die Berührungssensoren überprüft. Wenn einer der Berührungssensoren angestoßen worden ist, bekommen er Priorität über die Lichtsensoren, weil der Roboter wahrscheinlich auf ein Hindernis gestoßen ist. Der Roboter dreht den Motor für einige Millisekunden in die entgegengesetzte Richtung zu dem Berührungssensor, der angestoßen ist. Wenn kein Berührungssensor angstoßen ist, dann werden die Fotowiderstände miteinander verglichen. Dieser Vergleich wird in der Funktion compare_with_tol() durchgeführt, wo wir einen Wert gegen einen Durchschnittswert von zweien vergleichen. Um zu vermeiden, daß wir von zu viel "Rauschen" beeinflußt werden, sagen wir, daß zwei Werte gleich sind, wenn ihr Unterschied weniger als 5% beträgt.

Basierend auf dem Vergleich der Fotosensoren können wir dann entscheiden, welcher Motor gedreht werden muß. Da wir nur zwei Räder haben, können wir den Roboter auf der Stelle drehen durch das schnellere Drehen eines Motors oder durch das Drehen in die entgegengesetzte Richtung. Da der Microcontroller die Messung sehr schnell einige Male pro Sekunde wiederholt, erscheint die Bewegung des Roboters als kontinuierlich, auch wenn wir einen Motor für einen Bruchteil einer Sekunde anhalten, um ein bißchen nach

links ode rechts zu drehen.

Alles zusammensetzen

Wenn du die Elektronik zusammenbaust, ist es immer eine gute Idee, sie schrittweise zu testen. Auf diese Weise kann man leicht mögliche Fehler eingrenzen.

Es sind drei verschiedene Testpogramme im linuxrobot Softwarepaket (Download am Ende des Artikels) dabei. Das Programm ledtest läßt die zwei LEDs blinken. Du lädst es mit dem Befehl "make ledtestload". Dies kompiliert das Programm und lädt es in den Microcontroller. Die zwei LEDs sollten sofort anfangen zu blinken, sobald das Programm geladen ist. Wenn dieser Test erfolgreich ist, kannst du sicher sein, daß der Microcontroller mit seinem Oscillator und die Verbindung zum ladene der Software funktioniert.

Als nächstes kommt das Motortestprogramm. Dieses Testprogramm implementiert einen "elektronischen Gummiball". Du lädst es mit dem Befehl "make motortestload". Das Motortestprogramm überprüft die ganze Zeit die Berührungssensoren und wenn einer von ihnen angestoßen wurde, bewegt sich der Roboter etwas von dem angestoßenen Sensor weg. Wenn du den Roboter mit deiner Hand an einer Seite anstößt, springt er zurück. Lege deine Hand hinter den Roboter und er springt zwischen deinen zwei Händen vor und zurück, wie ein Gummiball. Wenn der Roboter diesen Test besteht, dann ist alles außer den Lichtsensoren und der RS232 Verbindung getestet.

Das letzte Testprogramm heißt adctest (kompiliere und lade es mit make adctestload). Das Programm testet die RS232 Verbindung, die da ist, um den Roboter zu debuggen und es testet den ADC (analog zu digital Wandler) mit den drei Fotowiderständen. Lade das Programm in den Microcontroller und verbinde dann den Adapter für die RS232 Schnittstelle zu deinen PC. Danach laß die folgenden drei Befehle in einer Shell laufen:

```
make ttydevinit
./ttydevinit /dev/ttyS0
cat /dev/ttyS0
```

Der Roboter sollte periodisch die Werte der Lichtintensität, die er an den Fotowiderständen gemessen hat, ausdrucken.

Wenn alle Tests bestanden wurden, kannst du das eigentliche Programm mit "make load" in den Roboter laden. Der beste Testplatz für die ersten Tests ist ein Raum mit nur einer einzigen Lampe in der Mitte. Der Roboter sollte einfach gerade in die Richtung der Lampe laufen und dort anhalten.

Es ist wirklich lustig zu sehen, wie der Roboter sich dreht, wenn du ihn mit dem Rücken zur Lichtquelle gewandt auf den Boden stellst und wie er Schatten vermeidet.

Probleme und Verbesserungsmöglichkeiten

Wir haben diesen Roboter als kleines Experiment gestartet. Es hat Spaß gemacht, einen autonomen Roboter zu bauen, der selber Entscheidungen treffen kann und keine Datenverbindung zu einem PC braucht. Das Programm, daß im Linux−Roboterpaket enthalten ist und du weiter unten im Artikel herunterladen kannst, ist immer noch klein und einfach, aber es macht, was wir wollten: Der Roboter läuft zur hellsten Stelle.

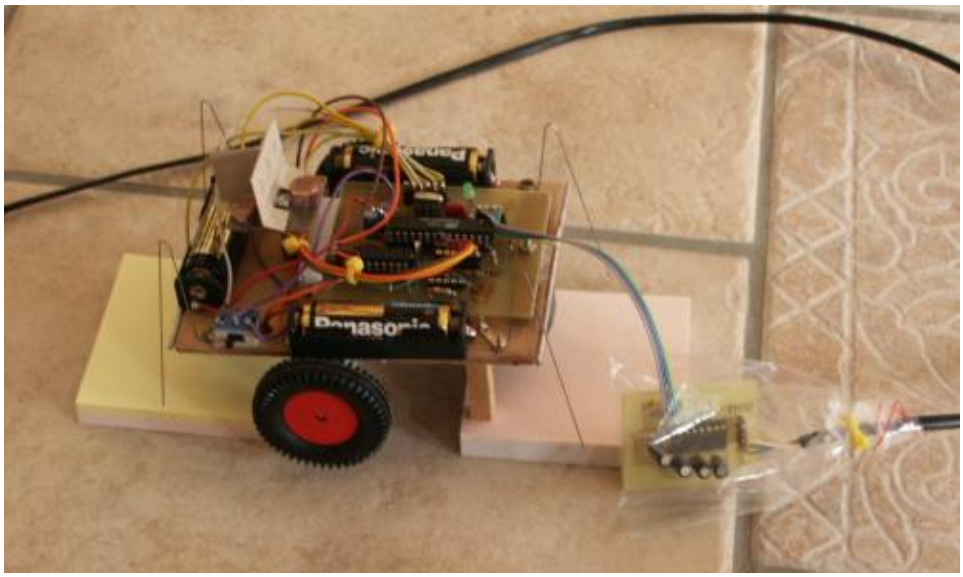
Wir würden gerne ein paar Dinge erwähnen, die als Ausgangspunkt für die weitere Entwicklung genommen werden könnten:

- Die Berührungssensoren werden nur in eher großen Intervallen (ein paar ms) überprüft, was die Antwortsmöglichkeiten des Roboters begrenzt. Sie sollten öfter überprüft werden.
- Wenn einer der Berührungssensoren gedrückt ist, dann erhält dies Priorität über alle anderen Dinge und der Roboter bewegt sich für ein paar 100 Millisekunden in die entgegengesetzte Richtung. Wenn ein anderer Sensor während dieser Zeit gedrückt wird, wird dies momentan ignoriert.
- Die Sensitivität der Fotowiderstände nimmt in schlechten Lichtverhältnissen ab. Dies kann zu dem Effekt führen, daß die gemessene Differenz zwischen den Sensoren unter der im Programm hartcodierten Schwelle von 5% liegt und der Roboter denkt, daß alle Sensoren die gleiche Menge Licht bekommen. Die Lichtwerte, die aus dem ADC kommen, könnten durch eine nicht−lineare Filterkurve korrigiert werden, um diesen Effekt zu kompensieren.

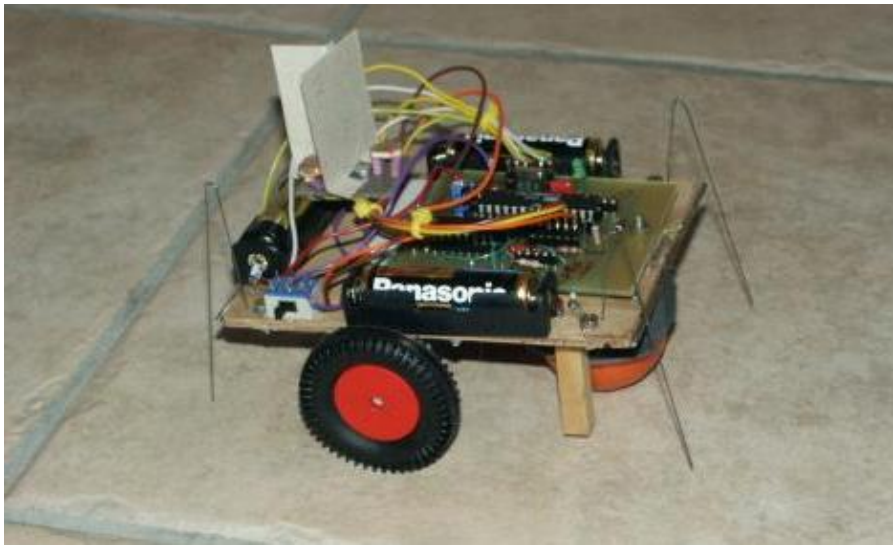
Momentan ist das linuxrobot Programm klein und einfach, du solltest daher in der Lage sein, es zu verstehen und vielleicht weiterzuentwickeln. Es braucht nur 50% des Speichers des 4433 Microcontrollers, d.h. du kannst noch eine Menge Dinge hinzufügen.

Das Gute an diesem Roboter ist, daß die Hardware recht generisch ist: Es sind einfach zwei Motoren und einige Sensoren, befestigt an einem Microcontroller. Alle Logik ist in der Software implementiert. Das bedeutet, daß du durch Verändern der Software fast alles, was du willst, verändern kannst.

Hier ist ein Bild des Roboters in Testposition. Wir haben einige post−it Blöcke untergelegt, damit er nicht wegläuft. Die rs232 Verbindung ist zu Debuggingzwecken verbunden:



... und der fertige Roboter, wie er nach Licht sucht....:



Referenzen

- LinuxFocus März 2002 Artikel 231: Programmierung des AVR Microcontrollers mit GCC
- linuxrobot−0.1.tar.gz: Die Software und Schaltbilder im Eagleformat
- Downloadseite für diesen Artikel : Das Datenblatt für den l293d und mögliche zukünftige Updates der linuxrobot Software finden sich hier.

2005−01−11, generated by lfparser_pdf version 2.51