Benjamin Benz

c't-Bot: Roboter selbst bauen

c't-Bots drehen auf der Stelle, spielen mit Kollegen Räuber und Gendarm, sammeln Golfbälle ein und bieten viel Raum für eigene Experimente. Wer nicht löten möchte, der erschafft mit c't-Sim virtuelle Roboter, lässt sie gegeneinander antreten und transplantiert ihre Intelligenz vielleicht später bei Freunden in reale c't-Bots.

Kurzinfo

Zeitaufwand:
c't-Bot macht süchtig! Simulator: ab 10 min., Hardware: mehrere Tage

Kosten:
Simulator kostenlos, Hardware ab 250 €

Löten:
auch für weniger geübte, aber geduldige Löter

Programmieren:
C-Grundkenntnisse empfehlenswert, Interesse erforderlich

Elektronik-Kenntnisse:
Grundkenntnisse reichen

Mobile Roboter faszinieren ebenso wie Modellautos, nur bieten sie viel mehr Möglichkeiten. Mit wenigen Code-Zeilen bringt man ihnen bei, Hindernissen auszuweichen, Wollmäuse in dunklen Ecken zu jagen oder einem neugierig hinterherzufahren. Beherrscht man erst einmal die Ansteuerung von Motoren und anderen Aktuatoren sowie die Auswertung der Sensoren, steht auch der Lösung komplexer Aufgaben nichts entgegen. Einen Golfball in einem Spielfeld zu suchen, ihn einzufangen und schließlich an einer bestimmten Stelle einzulochen, erfordert zwar etwas Tüftelei, muss aber nicht den gut ausgestatteten Forschungslabors vorbehalten bleiben – wie der c't-Bot zeigt.

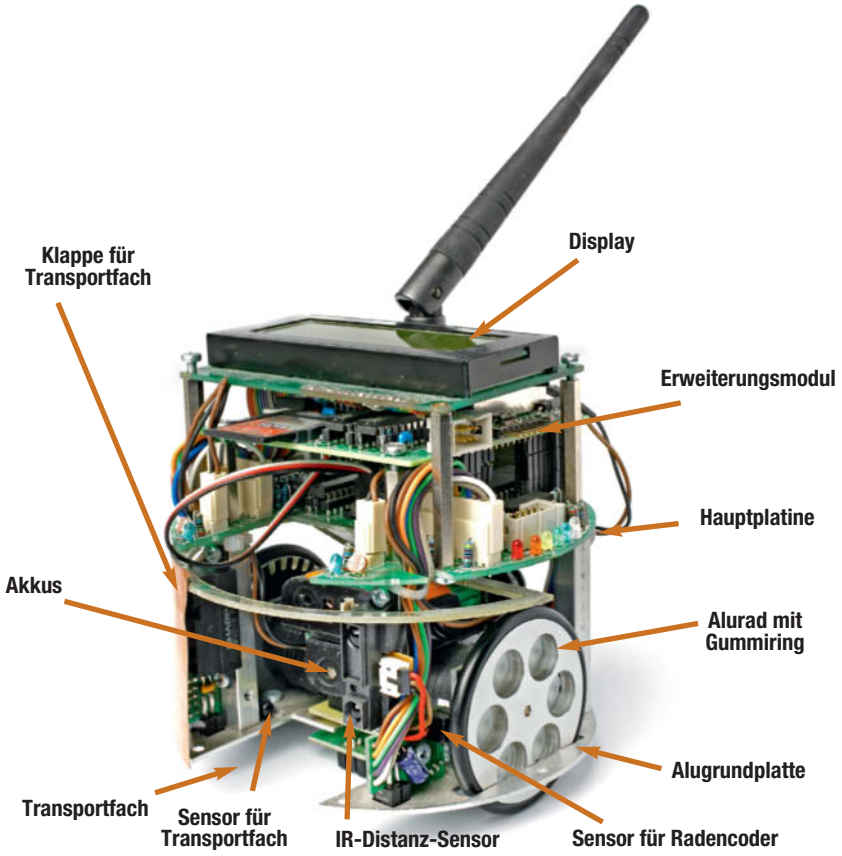
Den c't-Bot haben wir vor über fünf Jahren als Mitmach-Projekt aus der Taufe gehoben. Seitdem erfreuen mehr als 1600 der runden Mini-Roboter nicht nur Bastler und Hobbyrobotiker, sondern haben sich auch in den Ausbildungs-Laboren einiger Unis etabliert. Die ursprüngliche Idee gilt nach wie vor:

– Alle benötigten Komponenten von den Platinen über die Elektronik bis zu Rädern, Aluminiumteilen und Motoren sind – abgesehen von gelegentlichen Gebrauchtverkäufen – nur als Bausatz zu haben. Der Vertrieb erfolgt durch die Berliner Firma segor electronic. Die Tabelle rechts unten listet die verschiedenen Bausätze auf.
– Ein umfangreiches Software-Framework übernimmt nicht nur die Low-Level-Ansteuerung von Sensoren und Aktuatoren, sondern enthält auch Demoroutinen.
– Der dritte im Bunde, der Simulator c't-Sim, hilft nicht nur bei der Fehlersuche, sondern ermöglicht auch einen Einstieg in das Projekt ganz ohne Anschaffung irgendwelcher Hardware.

Der Selbstbau spart Montage- und Bestückungskosten, erfordert allerdings etwas Zeit und Lötarbeit. Der Aufbau des c't-Bots sollte niemandem Probleme bereiten, der halbwegs mit einem Lötkolben umgehen kann, weil wir ganz bewusst auf SMD-Bauteile verzichtet haben. Wir raten dennoch dringend dazu, die bebilderte Aufbauanleitung im Projekt-Wiki (siehe Kasten) genau zu beachten. Einige Komponenten verzeihen keinerlei Verpolung, andere lassen sich nur in der richtigen Reihenfolge einfach montieren. Im Wiki finden Sie auch eine Reihe von optionalen Modifikationen, die kleinere Probleme des ursprünglichen Entwurfs beheben oder den Roboter an bestimmte Gegebenheiten anpassen.

c't-Bot

Die robuste und solide Konstruktion des c't-Bot bildet eine relativ preiswerte und flexible Plattform für eigene Entwicklungen: Dank der runden Grundfläche sowie den axial angeordneten Rädern dreht der c't-Bot auf der Stelle – Festfahren ausgeschlossen. Die beiden kraftvollen Motoren bringen den kleinen Kerl auf beachtliche Geschwindigkeit. Gegenstände sammelt und transportiert er in der vorderen Aussparung, für die es einen nachrüstbaren Verschluss gibt.



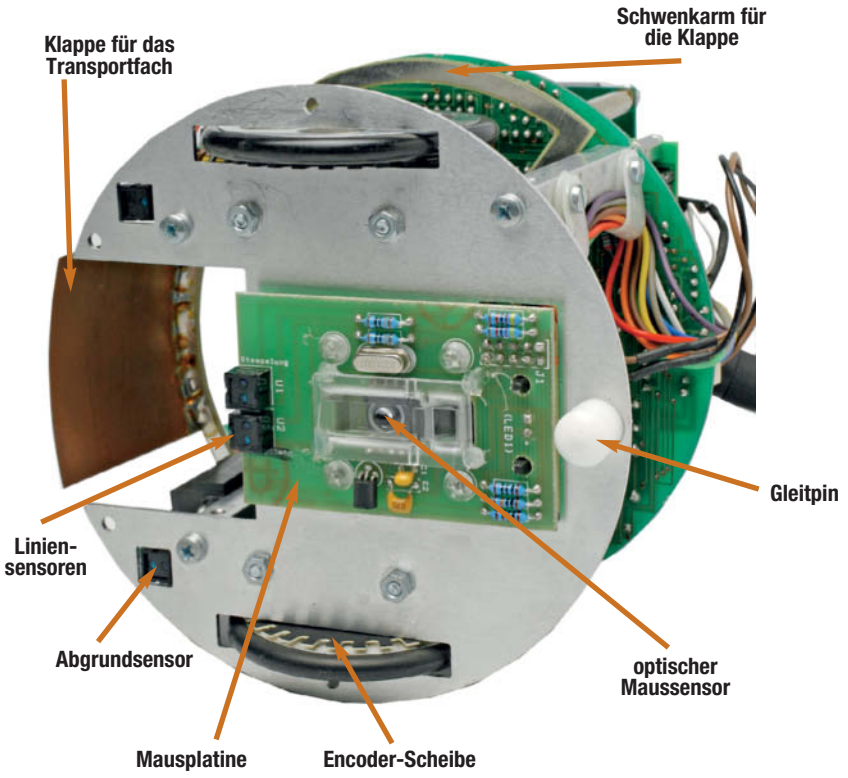
In der Aussparung kann der c't-Bot Golfbälle, Filmdosen oder Ähnliches transportieren.

Seine Umgebung nimmt er über je zwei nach vorn gerichtete Abstands- und Lichtsensoren wahr. Zudem überwachen vier Sensoren den Boden. Sie warnen vor Abgründen und erlauben das Verfolgen von Linien. Das Innenleben einer optischen Maus hilft bei der Positionsbestimmung, indem es Bewegungen über den Boden registriert. Zu Beginn des Projektes bildete der 8-Bit-Mikrocontroller ATmega32 mit 32 KByte Flash-Speicher sowie 2 KByte RAM das Gehirn des kleinen Roboters. Mittlerweile gibt es nicht nur pinkompatible Nachfolger mit deutlich mehr Speicherplatz (ATmega644P oder ATmega1284P) sowie eine Speichererweiterung in Form eines SD-Kartenlesers, sondern auch ein Leserprojekt, das den c't-Bot um einen potenten 1-GHz-ARM-Prozessor erweitert (siehe Kasten). Alternativ dazu kann man auch Rechenaufgaben an einen PC auslagern, die Kommu-

Bauteile für den c't-Bot

Paket	Inhalt	Hinweise	Preis
Basispaket	Platinen, Mechanik, Elektronik (ohne Akkus, ohne Display)	unbedingt erforderlich	249,00 €
Displaysatz	Display, Kabel, Montagematerial	empfehlenswert	21,50 €
Fernbedienung	Fernbedienung (ohne Batterien)	empfehlenswert	7,90 €
Klappensatz	Klappe, Mechanik, Servo	optional	28,60 €
Erweiterungspaket	Erweiterungsplatine, Kartenleser, Kleinteile (ohne WLAN)	optional	36,40 €
WLAN-Teilesatz	Funkmodul und Antenne (ohne Kleinteile)	optional, erfordert Erweiterungspaket	130,00 €
USB-Seriell-Kabel	Diagnose per USB	optional	23,90 €
AVRISP mkII	Programmieradapter	erforderlich (Alternative: mySmartUSB)	38,00 €

Außerdem fehlen noch: Akkus (5x Mignon 1,2 V) und Batterien für die Fernbedienung. Alle Teile erhältlich bei www.segor.de



nikation mit dem Bot erfolgt per WLAN. Per WLAN können übrigens auch mehrere Bots über die Lösung komplexer Aufgaben konferieren.

c't-Sim

Eine Besonderheit dieses Projektes ist, dass es sich nicht nur an hartgesottene Selbst-Löter und Assembler-Programmierer wendet, sondern auch reine Soft-Werker zum Mitspielen einlädt: Der Simulator c't-Sim kommuniziert mit mehreren virtuellen Robotern, gaukelt ihnen eine Umgebung vor und liefert realistische Sensorwerte zurück. Dazu simuliert er eine dreidimensionale Welt mit Objekten und Lichtquellen. Ausgehend von der Position eines Bots verfolgt der c't-Sim beispielsweise die Blickstrahlen der Abstandssensoren und prüft, in welcher Distanz diese ein Objekt treffen. Die exakten Werte verzerrt er entsprechend der Kennlinie der Sensoren des realen Bots und verrauscht sie noch ein wenig. Somit haben reale und simulierte Bots mit annähernd den gleichen Messungenauigkeiten zu kämpfen. Der c't-Sim nimmt auch Steuerbefehle für Aktuatoren – etwa die beiden Antriebsmotoren – entgegen und bewegt den Bot dementsprechend in der virtuellen Welt. Dabei erkennt der Simulator auch Kollisionen mit Objekten sowie Stürze in virtuelle Abgründe..

Die vier nach unten gerichteten CNY-70-Sensoren erkennen Abgründe und Linien, während der optische Maussensor bei der Positionsbestimmung hilft.

Alle Links im Artikel auch unter
www.ct.de/cs1120108

Der Clou dabei: Im Wesentlichen läuft auf simulierten und realen c't-Bots derselbe Code. So lassen sich etwa komplexe Verhaltensmuster bequem am PC testen. Aber auch wer gar nicht plant, in Hardware zu investieren, kann mit dem c't-Sim eine Menge anfangen. Wir haben darin beispielsweise einen ganzen Labyrinth-Wettbewerb für Leser rein virtuell ausgetragen.

Ganz nebenbei dient der c't-Sim aber auch als Schalt- und Kommunikationszentrale für reale, per USB oder WLAN angekoppelte Roboter. Er zeigt den Status von Sensoren und Aktuatoren, visualisiert die interne Karte des c't-Bot und übermittelt Fernsteuerungskommandos. Stehen mehrere Roboter in Kontakt zu einem c't-Sim, so leitet dieser Nachrichten zwischen ihnen weiter. Dabei dürfen Sie übrigens reale und simulierte Roboter nach Belieben mischen. Es ist noch nicht einmal nötig, dass sich am selben Ort die realen Exemplare versammelt haben, weil der c't-Sim die gesamte Kommunikation per TCP/IP abwickelt.

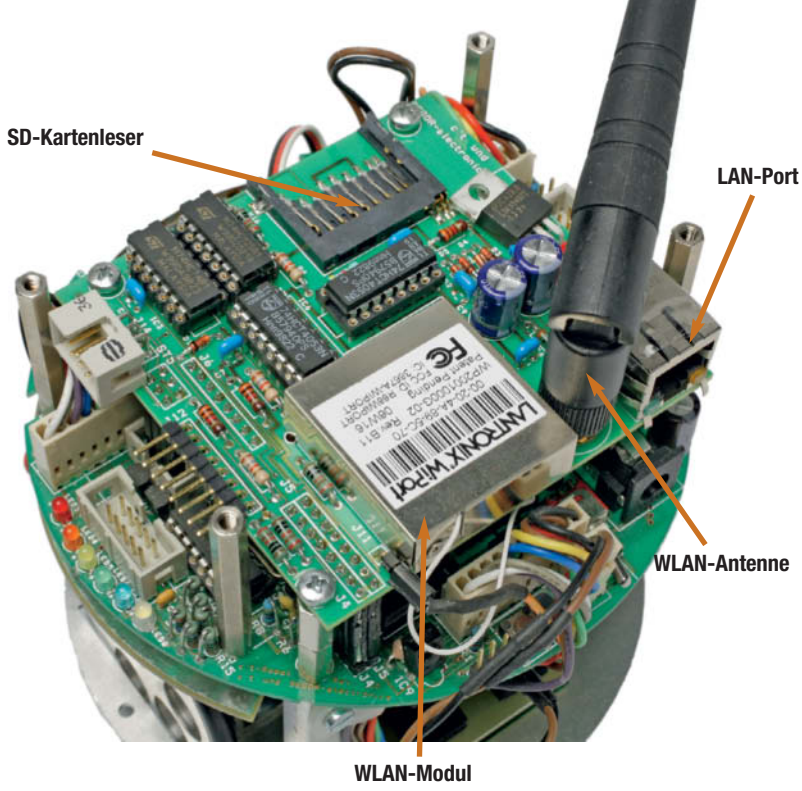
Die grundlegenden Konzepte des c't-Sim beschreibt eine Reihe von Artikeln, die allesamt online stehen. Wen die konkrete Umsetzung interessiert, der sollte allerdings einen Blick in den Java-Code werfen, denn der Simulator wurde kontinuierlich weiterentwickelt. Zwei Schmankerl wollen wir hier explizit erwähnen: Erstens kann man ganz leicht selbst virtuelle Welten gestalten, weil der c't-Sim dazu ganz einfache XML-Dateien einliest. Diese enthalten – neben ein paar Angaben zu

Anlaufstellen

Die zentrale Anlaufstelle für alles rund um c't-Bot und c't-Sim ist die Projekt-Webseite www.ct-bot.de. Ergänzend dazu haben wir eine Trac-Instanz eingerichtet,mit vollem Schreibzugriff für die Community. Diese Entwicklungsplattform besteht aus einem Wiki, das unter anderem die aktuelle Aufbauanleitung, aber auch eine Bildergalerie und Videos von Leserprojekten enthält. Den gesamten Code von c't-Sim und c't-Bot verwaltet ein Subversion-Repository (SVN), zum Stöbern reicht aber auch erst einmal der Webbrowser aus. Wünsche für Erweiterungen, Bug-Reports und Ähnliches gehören ins Ticket-System, über aktuelle Änderungen im Trac informiert die Timeline.

Erste Wahl für Fragen rund um das Projekt von Hard- bis Software ist die Mailingliste, für die es übrigens auch ein Archiv gibt und die auch die Autoren der Artikel abonniert haben. Die Mailingliste informiert auch über größere Code-Updates. Außerdem gibt es auf heise online ein Diskussionsforum. Wenn es mal ganz dringend ist, tummeln sich im IRC-Chatraum fast immer ein paar Entwickler.

Viele Fragen beantworten jedoch bereits die zahlreichen c't-Artikel zum c't-Bot, die alle online stehen und deren Lektüre wir dringend empfehlen. Antworten auf die meist gestellten Fragen gibt die FAQ. Darüber hinaus enthält auch unser Wiki noch zahlreiche Tipps, Tricks und Zusatzinformationen sowie Links zu Datenblättern und externer Dokumentation. Ebenfalls online verfügbar ist die detaillierte Beschreibung aller Funktionen und Variablen von c't-Bot-Framework und c't-Sim.



Farbgebung und Texturen – vor allem den Grundriss der Welt, definiert per ASCII-Zeichen. So symbolisiert ein „X" beispielsweise ein Stück quadratisches Mauerstück, ein Leerzeichen eine frei befahrbare Fußbodenkachel und ein „*" eine Lampe. Zeichen wie „|", „\", oder „-" stehen für Linien auf dem Boden. Anregungen für eigene Kreationen liefern zahlreiche Beispiel-Parcours.

Zweitens ist der c't-Sim für sogenannte Schiedsrichter-Module („Judges") vorbereitet, um Roboter-Wettkämpfe auszutragen. Wir haben das beispielsweise in unserem Simulator-Wettbewerb genutzt, um automatisch den Startschuss zu geben und später zu ermitteln, welcher von zwei Robotern zuerst ein Labyrinth durchquert hat. Neben sportlichen Wettbewerben helfen die Judges aber auch bei der Bewertung der Effizienz von Algorithmen oder der automatischen Suche nach optimalen Parametern.

Software

Wir haben bereits ganz zu Beginn des Projektes entschieden, nur Open-Source-Entwicklungswerkzeuge einzusetzen. Ganz konkret bedeutet das: gcc als Compiler für Standard-C-Code, Eclipse als Programmierwerkzeug, avrdude zum Übertragen der Binaries und zuletzt Java für den c't-Sim. Dieser auf den ersten Blick recht bunte und umfangreiche Tool-Strauß bringt zwei gewichtige Vorteile: Zum einen können so Windows-Fans, Linuxer und Macies mit denselben Tools an derselben Code-Basis arbeiten und zum anderen läuft der C-Code für den Roboter sowohl im Simulator als auch auf dem realen Roboter. Das erleichtert die Fehlersuche ungemein, denn auf dem PC stehen viel mehr Debug-Möglichkeiten zur Verfügung. Dazu ist der Code soweit irgend möglich plattformunabhängig. Lediglich ein paar Spezial-Routinen behandeln reale Hardware und Simulator getrennt. So bedürfen Sensoren und Aktuatoren des realen Bots direkten Zugriff auf die Register des Controllers. Bei simulierten Bots kommen und gehen diese Werte indes über eine TCP/-IP-Schnittstelle direkt vom und zum Simulator.

Das Gute daran: Das c't-Bot-Framework abstrahiert von all diesen Dingen. Als angehender Roboter-Dompteur müssen Sie sich damit überhaupt nicht herumschlagen und können sich dank einfacher Befehle wie bot_turn() oder bot_drive_distance() ganz darauf konzentrieren, Ihrem c't-Bot Kunststückchen beizubringen. Wie das geht, beschreibt der Artikel ab S. 100.

Community

Allerdings sind nicht alle Mitglieder der c't-Bot-Community Anhänger dieser Philosophie und so sind im Laufe der Jahre auch ein paar alternative Ansätze entstanden. Ein Leser schwört etwa auf den Basic-Compiler BASCOM. In der Tat vereinfacht dieser auf Atmel-Prozessoren zugeschnittene Basic-Dialekt gerade die direkten Hardware-Zugriffe. Allerdings übersetzt die kostenfreie Demoversion maximal 4 KByte Code. Andere Leser haben sich bemüht, den offiziellen Code aufzuräumen, umzugestalten oder zu vereinfachen. Wer allerdings solchen Code seinem eigenen Projekt zu Grunde legt, dürfte nur sehr eingeschränkt von Weiterentwicklungen profitieren.

Sehr viele spannende Ideen aus der Community sind jedoch in das zentrale Repository eingeflossen, das ein paar kreative Köpfe nach wie vor hegen und pflegen. So kam etwa erst kürzlich ein Basic-Interpreter dazu, mit dem man den Roboter per Skripte steuern kann. Beispielhaft für eine ganze Reihe spannender Community-Erweiterungen und Modifikationen haben wir den Leser Timo Sandmann gebeten, zwei seiner Lieblinge vorzustellen (siehe Kasten). Leider finden aber längst nicht alle Erweiterungen wieder den Weg zurück in die Community. So können wir anhand der Teilebestellungen nur vermuten, dass eine Uni für den Lehrbetrieb sogar eine eigene Hauptplatine entwickelt hat. Genauere Informationen dazu liegen leider nicht vor.

Das Erweiterungsmodul bringt den c't-Bot ins WLAN und beschert ihm genug Speicher, um eine Karte der Umgebung zu erstellen.

Mit einer handelsüblichen Infrarot-Fernbedienung, die RC5-Codes sendet, lässt sich der c't-Bot steuern.

Um einen jungfräulichen Mikrocontroller erstmalig mit Code zu befüllen, braucht man einen Programmieradapter (links). Danach reicht dank des integrierten Bootloaders auch die WLAN-Schnittstelle oder ein preiswertes USB-seriell-Kabel (oben).

Leserprojekt

CPU-Erweiterung

Für spektakuläre Features wie Gesichtserkennung per Kamera fehlen dem 8-Bit-Mikrocontroller Performance und Speicher. Doch das lässt sich mit einer Huckepack-Platine und einem flotten Prozessor ändern. Einen guten Kompromiss aus Rechenleistung, Ausstattung und Support auf der einen Seite sowie Leistungsaufnahme, Platzbedarf und Anschaffungskosten (125 bis 150 US-Dollar) auf der anderen Seite, bietet das 8,5 cm x 8,6 cm große BeagleBoard von Texas Instruments. Es passt damit oben auf den c't-Bot, ohne über dessen kreisrunde Grundfläche hinauszuragen. Ebenfalls für dieses Board spricht die große Community, die sich rund um die offene Plattform gebildet hat und die auch von Texas Instruments unterstützt wird.

Auf dem BeagleBoard sitzt ein zweifach superskalarer 32-Bit-Prozessor mit ARM-Kern (Cortex-A8) und bis zu 1 GHz Taktfrequenz, wie er auch in vielen aktuellen Smartphones steckt. Dazu kommen bis zu 512 MByte RAM. Das lässt trotz vollwertigem Linux-Betriebssystem (beispielsweise Ubuntu 11.04 für ARM) reichlich Raum für (Robotik-)Anwendungen.

Die USB-2.0-Ports nehmen handelsüblichen WLAN-USB-Sticks für die Kommunikation mit der Außenwelt auf und machen so das relativ teure WLAN-Erweiterungsmodul überflüssig. So bleibt auch gleich noch Raum für noch mehr Peripherie – etwa eine USB-Webcam zur Bildverarbeitung. Audio-Aus- und -Eingänge liefern mit ein wenig zusätzlicher Elektronik nicht nur Sound- und Sprachausgabe für den Roboter, sondern ebnen auch den Weg in Richtung Spracherkennung. Dabei kann man sich Standard-Software wie Flite (Sprachsynthese) bedienen.

Die xM-Version des BeagleBoards führt sogar noch den Kamera-Port des Prozessors auf einem Erweiterungsanschluss heraus. Mit anderen Elektronik-Komponenten spricht das BeagleBoard über eine der seriellen Schnittstellen (UART, SPI, I2C) oder die digitalen I/O-Pins.

Das BeagleBoard löst den bisherigen Mikrocontroller übrigens nicht ab, sondern entlastet ihn. Das klingt nach großen Code-Umbauten – aber das täuscht: Dank des stark modularen Aufbaus des c't-Bot-Frameworks lässt sich der eigentlich für die Simulation auf dem PC entworfene High-Level-Code mit wenigen Anpassungen auch für ARM-Linux übersetzen und direkt auf dem BeagleBoard ausführen. Anstatt vom Simulator bekommt er Sensor- und Aktuatorwerte dann über eine serielle Schnittstelle direkt vom Mikrocontroller. Auf diesem läuft weiterhin der gesamte Low-Level-Code. Dieser Trick bewahrt die Kompatibilität zum Robotercode für die Standardversion des c't-Bot. Wie dies und der mechanische Aufbau im Detail funktionieren, beschreibt das Wiki ausführlich.

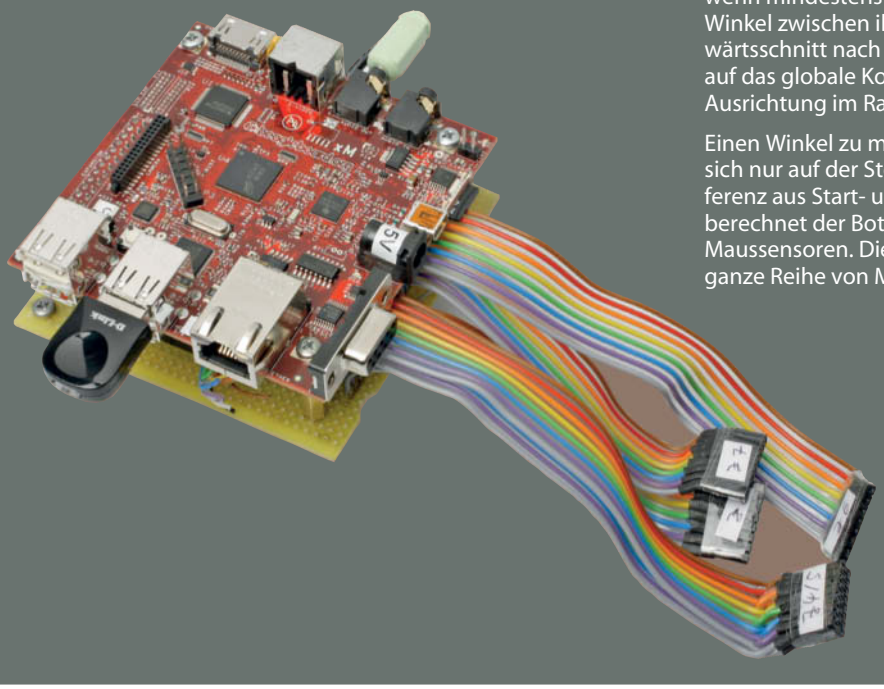
Ortungsdienste

Spätestens wenn zwei oder mehr c't-Bots gemeinsam eine Aufgabe bewältigen sollen, müssen sie ihre Position in einem gemeinsamen (globalen) Koordinatensystem kennen. Aber auch ein einzelner Bot profitiert davon, seine Position ab und an zu verifizieren. Denn bei der Koppelnavigation anhand der eingebauten Sensoren kumulieren sich mit der Zeit systembedingt Ungenauigkeiten. Ein regelmäßiger Abgleich mit einer externen Referenz steigert daher die Genauigkeit der Pfadplanung und der internen Karte.

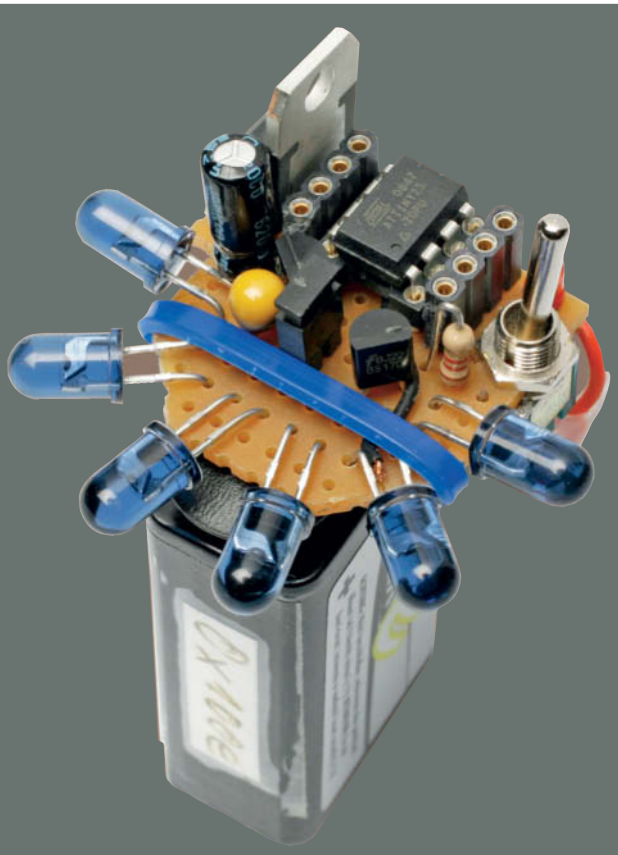
Mit recht einfachen Mitteln lässt sich ein bei Seefahrern schon lange bekanntes Verfahren aus der terrestrischen Navigation auf den c't-Bot übertragen. Dazu muss der Segler mehrere Leuchttürme – deren genaue Position er kennt – anpeilen. Er bestimmt die Himmelsrichtung, in der er den Turm sieht. Zeichnet er nun diese Blickstrahlen ausgehend von den Leuchttürmen in die Seekarte ein, so schneiden sie sich genau am Standort des Schiffes.

Folglich reichen bereits zwei gut positionierte Landmarken, um die Position zu bestimmen. Doch der Seefahrer hat – anders als der c't-Bot – einen präzisen Kompass. Es geht aber auch ohne, wenn mindestens drei Türme zu sehen sind. Dann reichen die Winkel zwischen ihnen und eine Handvoll Mathematik (Rückwärtsschnitt nach Cassini), um nicht nur die Position bezogen auf das globale Koordinatensystem, sondern auch die eigene Ausrichtung im Raum zu bestimmen.

Einen Winkel zu messen, ist für den c't-Bot ein Klacks: Wenn er sich nur auf der Stelle dreht, dann ergibt sich der Winkel als Differenz aus Start- und Endrichtung der Drehung. Die Richtung berechnet der Bot dabei anhand der Daten seiner Rad- und Maussensoren. Die Differenzbildung eliminiert bereits eine ganze Reihe von Messfehlern.



Das BeagleBoard mit schnellem ARM-Prozessor kommt huckepack über die Mikrocontroller-Platine – die sich weiter um Sensoren und Aktuatoren kümmert.



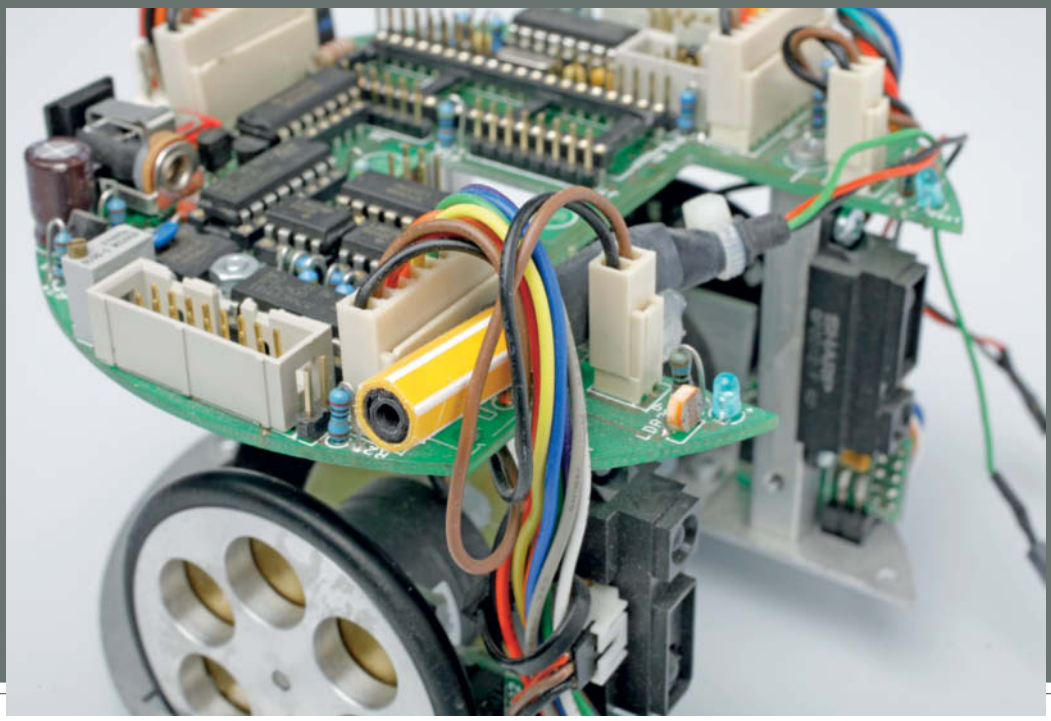
Die Landmarken für den c't-Bot bestehen nur aus einer Handvoll Bauteilen und senden ihre Position als IR-Code.

Die Leuchttürme für den c't-Bot übertragen alle paar Millisekunden ihre ID als Infrarot-Fernbedienungscode. Sie sind billig und leicht zu bauen, weil sie nur aus ein paar IR-LEDs, einem kleinen Mikrocontroller, einer Batterie und einigen Kleinteilen bestehen. Der Controller erzeugt das Modulationssignal und steuert die Leuchtstärke.

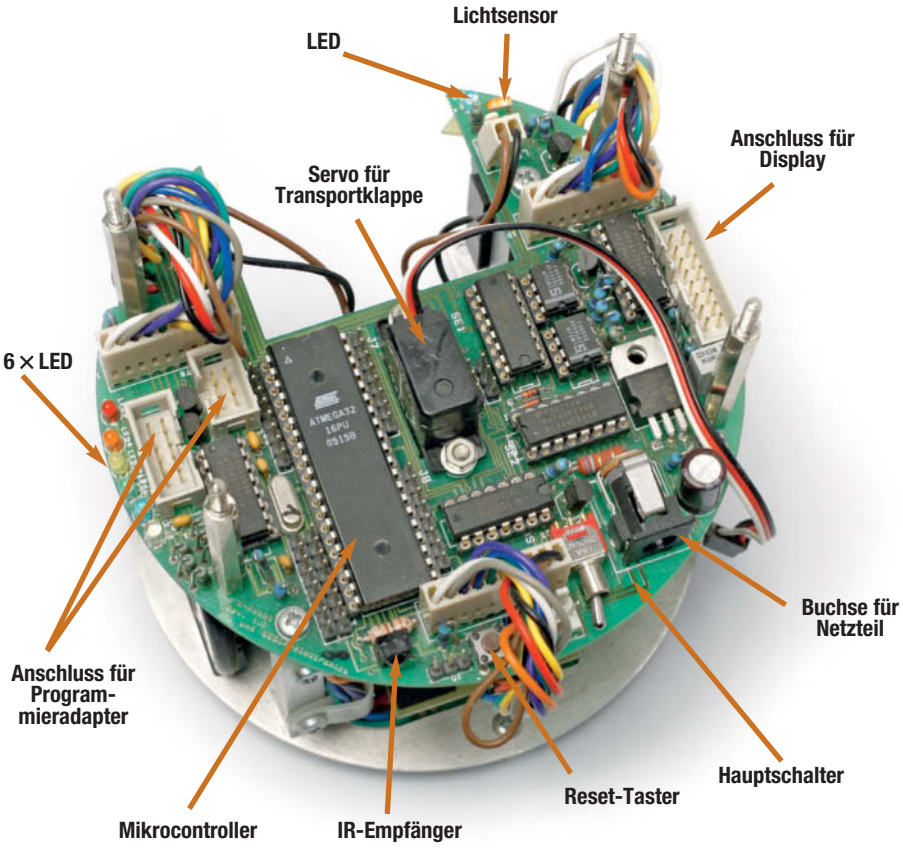
Um zu erkennen, ob er in der aktuellen Blickrichtung einen Leuchtturm sieht, braucht der c't-Bot allerdings noch einen zusätzlichen Sensor: Der besteht im Wesentlichen aus einem ganz gewöhnlichen IR-Empfänger, für den in der Software bereits ein Treiber existiert. Der zentrale Unterschied zum bereits vorhandenen Empfänger (IC9) ist allerdings eine sehr enge Blende, damit er nur auf Infrarotlicht aus einer Richtung reagiert. Für erste Experimente reicht dazu die Hülle eines alten Stifts. Eine ausführliche Bauanleitung sowie Tipps für eine geschickte Montage auf dem Bot finden Sie im Wiki.

Im c't-Bot-Framework existiert bereits Code für ein Roboterverhalten, das nach drei Landmarken sucht und aus den gemessenen Winkeln die aktuelle Position und Ausrichtung ermittelt. Unter guten Bedingungen (hinreichend große Winkeldifferenzen) und in einem 2 m × 1 m - Spielfeld liegt der mittlere Fehler bei bei rund 2 cm. Mit zusätzlichen Landmarken kann man die Genauigkeit noch steigern.

Für weitere (oder erste) Experimente existieren Landmarken und Sensor auch im Simulator c't-Sim. Der bisherige Roboter-Code deckt aber nur einen Teilaspekt der Lokalisierung ab, so dass hier noch viel Platz für eigene Entwicklungen bleibt. Denkbar wäre beispielsweise, auch während der Fahrt kontinuierlich Informationen über Landmarken zu sammeln und damit die Positionsberechnung zu füttern. Letzteres ist mathematisch allerdings deutlich anspruchsvoller und überfordert womöglich den 8-Bit-Mikrocontroller. (Timo Sandmann)



Dem c't-Bot reicht ein fest montierter IR-Empfänger aus, um die Landmarken anzupeilen. Dabei dreht er sich so lange um die eigene Achse, bis er ein Signal empfängt. Die ausgediente Stifthülle reduziert den Blickwinkel und erhöht damit die Winkelauflösung.



Das ist schade, denn wie alle Community-Projekte lebt auch der c't-Bot vom Mitmachen.

Wie kurzweilig das sein kann, belegen die Videos in unserer Wiki-Galerie: So ringen dort mehrere c't-Bots Sumo, verfolgen Linien, treten im Rudel an oder durchqueren Labyrinthe. Wenn Sie also gute Ideen, verbesserten Code oder Hardware-Modifikationen haben, dann lassen Sie doch bitte andere daran teilhaben. Wie das am besten geht, haben wir im Kasten „Anlaufstellen" zusammengefasst. Selbst wenn Sie Code oder Schaltungen nicht veröffentlicht sehen wollen, so freuen sich andere sicher über weitere Bilder und Videos in der Galerie.

Bilanz

Auch nach über fünf Jahren sind weder der c't-Bot noch der c't-Sim fertige Spielzeuge von der Stange.

Die Steuerung des c't-Bot übernimmt ein Mikrocontroller aus der beliebten AVR-Serie von Atmel, der beispielsweise in C programmiert wird.

Mechanik, Elektronik und Platinen für den c't-Bot gibt es als Bausatz. Somit sind Geduld und ein Lötkolben gefragt, um den eigenen Roboter zum Leben zu erwecken.

Sie bieten vielmehr eine Gelegenheit, selbst zu experimentieren und sich mit anderen auszutauschen. Beide sind längst über das hinausgewachsen, was wir uns bei Projektbeginn im stillen c't-Kämmerlein ausgedacht haben. Weder die insgesamt 17 Artikel, mit denen wir das Projekt in c't angeschoben und begleitet haben, noch die 15 Seiten in diesem Sonderheft vermögen es, jede Facette dieses mittlerweile von der Community gepflegten Projektes darzustellen. Dafür haben schlicht zu viele kreative Köpfe ihre pfiffigen, genialen oder auch verrückten Ideen eingebracht. Uns Initiatoren freut insbesondere, dass sowohl auf der Mailingliste als auch im Wiki und dem Code-Repository immer noch Aktivität herrscht. Obwohl die Pflege des Codes seit geraumer Zeit Enthusiasten aus der Community obliegt, verfolgt das ursprüngliche Projektteam weiterhin gespannt Diskussionen auf der Mailingliste und berät bei Problemen. Zudem ist in dem IRC-Chat meist mindestens ein erfahrener c't-Bot-Dompteur anzutreffen, sodass Anfänger schnelle Hilfe bekommen können. Das Studium der in weiten Teilen noch aktuellen c't-Artikel und der ausführlichen Dokumentation im Wiki ersetzt das freilich nicht.

Weil es bei diesem Projekt aber viel mehr um Spaß als um Installationsprobleme und Code-Dokumentation geht, wünschen wir zum Abschluss dieser Einführung spannendes, aber maßvolles Basteln – ganz im Sinne des im Wiki verlinkten Editorials, mit dem wir den c't-Bot im Januar 2006 aus der Taufe gehoben haben. (bbe)

www.ct.de/cs1120108 c't

Einstiegspunkte

Was	Wo
Projekt-Webseite	www.ct-bot.de
c't-Artikel	www.heise.de/ct/projekte/Artikelliste-406396.html
FAQ	http://heise.de/-291940
Wiki, Tickets, Timeline, Code-Browser	www.heise.de/ct/projekte/machmit/ctbot
Code	https://www.heise.de:444/svn/ctbot
Chat-Raum	#ct-bot auf irc.blitzed.org oder http://cgiirc.blitzed.org
Mailingliste	www.heise.de/bin/newsletter/listinfo/ct-bot-entwickler
Archiv	www.heise.de/ct/newsletter/archiv/ct-bot-entwickler
Diskussionsforum	http://heise.de/-1334748