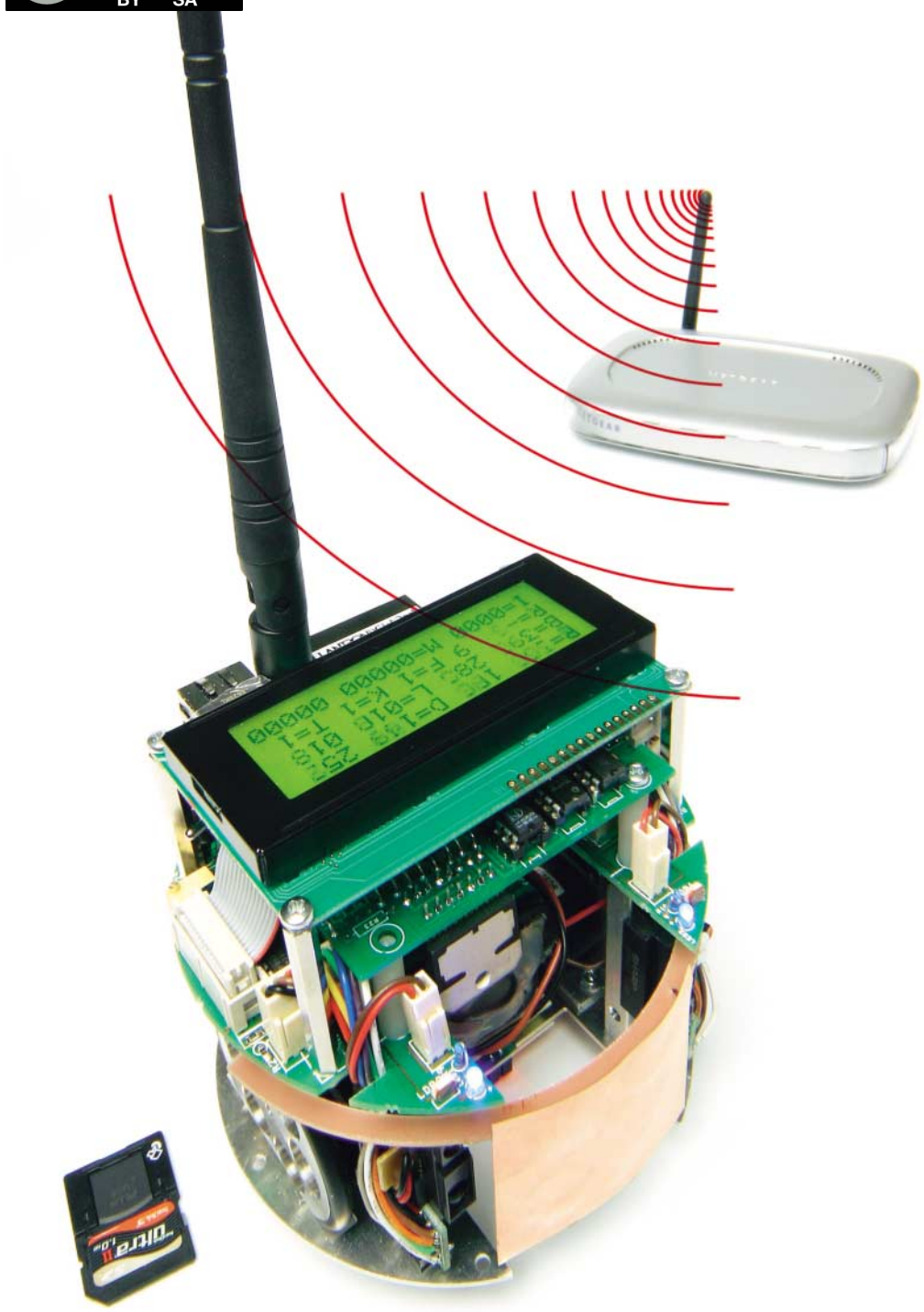


Thorsten Thiele, Benjamin Benz

Außendienstler

Funkmodul, Massenspeicher und Klappe für den c't-Bot

Mit einem Erweiterungsmodul lernt der Selbstbauroboter c't-Bot, sich per WLAN mitzuteilen und Kommandos vom PC zu empfangen. Eine Klappe verriegelt das Transportfach, und eine MMC- oder SD-Karte bietet genug Speicherplatz für die Kartierung der Umgebung.

Es ist faszinierend, einem autonomen Roboter beim Lösen komplexer Aufgaben zuzusehen. Um zu verstehen, was im Roboterhirn gerade vorgeht, und die Algorithmen zu verfeinern, lohnt es, wenn der Roboter seine Sensordaten an einen PC übermittelt. Dort lassen sie sich leichter aufbereiten und visualisieren. Der eine oder andere Steuerbefehl kann den c't-Bot auch mal aus einer verfahrenen Situation befreien.

Mit dem c't-Bot-Erweiterungsmodul lernt der Kleine, per WLAN und LAN zu kommunizieren. Obendrein bekommt er auch gleich einen integrierten Webserver, über den er sich fernbedienen lässt.

Das Erweiterungsmodul bringt aber noch mehr: Der Arbeitsspeicher des ATmega-Mikrocontrollers quillt schnell über, wenn der Bot versucht, eine detaillierte Karte der Umgebung abzulegen. Eine Flash-Speicherkarte hebt diese Einschränkung auf. Mit ein paar Programmiertricks spricht der Bot eine MMC- oder SD-Karte mit vielen hundert MByte Speicher an.

Eine schöne Übungsaufgabe für Roboterelteren ist das Suchen, Auffinden und Einfangen von Gegenständen. Die Einbuchtung vorne am c't-Bot prädestiniert ihn dafür, beispielsweise Tischtennisbälle oder Filmdosen zu transportieren. Die hier vorgestellte Klappe verriegelt das Transportfach und verhindert so, dass einmal eingefangene Gegenstände wieder entkommen.

Wie bereits bei den vorangegangenen Teilen des c't-Bot-Projektes [1] braucht man auch für die Montage des Erweiterungsmoduls einen Lötkolben und ein wenig Zeit. Auf der Projektseite beschreiben wir besonders knifflige Abschnitte der Montage und illustrieren sie mit ein paar Fotos. Die Platine gibt es wie üblich bei eMedia. Komplette Bausätze, den Servo und die Mechanik der Klappe bietet die Firma Segor Electronics an. Die Klappe samt Servo kostet 28,60 Euro, der Bausatz für das Erweiterungsmodul mit WiPort und Antenne 171,40 Euro. Wer kein WLAN oder LAN braucht, aber auf den Karten-Slot nicht verzichten will, zahlt ohne WiPort 36,40 Euro. Da fast jeder Supermarkt mittlerweile SD- und MMC-Karten verramscht, gehört keine Speicherkarte zum Bausatz.

Nach Hause telefonieren

Für die drahtlose Kommunikation zwischen PC und c't-Bot kämen theoretisch verschiedene Funkverfahren in Frage. Eine proprietäre Punkt-zu-Punkt-Lösung (beispielsweise ISM-2-Band bei 864 MHz) ist zwar unter Umständen relativ preiswert zu realisieren, erfordert aber auf Seiten des PC nicht nur spezielle Hardware, sondern auch Treiber.

Bei Bluetooth existieren zwar Standardadapter für den PC, und auch diverse Handys und PDAs beherrschen das Verfahren. Möchte man allerdings mehrere Bots miteinander und mit einem PC kommunizieren lassen, wird dies bei Bluetooth teuer und aufwendig. Die preiswerten Bluetooth-Module beherrschen nämlich nur das serielle Portprofil (SPP) und damit nur zwei Partner pro Verbindung. Die im Bluetooth-Standard spezifizierten Scatter-Netze mit vielen Knoten erfordern in der Praxis erheblichen Software-Aufwand.

WLAN hat sich im PC-Umfeld durchgesetzt. Nahezu alle aktuellen Notebooks und viele PCs haben bereits eine eingebaute WLAN-Schnittstelle, oder im heimischen Netzwerk steht ohnehin ein WLAN-Accesspoint. Wenn nicht, kostet ein USB-WLAN-Stäbchen nur ein paar Euro. Es liegt daher nahe, den c't-Bot mit WLAN auszurüsten. So erreicht jeder beliebige Rechner im Netzwerk mit Standard-Software den Roboter.

Vermittlungsstelle

Die WLAN-Anbindung übernimmt ein WiPort-Modul von Lantronix [2]. Dieser Baustein vermittelt zwischen LAN oder WLAN auf der einen und zwei seriellen Schnittstellen auf der anderen Seite. Er beherrscht die aktuellen WLAN-Standards, unter anderem 802.11g und WPA-PSK-Verschlüsselung (TKIP, AES). Er meldet sich entweder an einem bestehenden Funknetz an, oder kommuniziert im Ad-hoc-Modus mit einem PC. Neben dem WLAN-Interface besitzt er auch einen LAN-Port – es geht also auch per Ethernet-Kabel.

Mit dem c't-Bot-Gehirn spricht der WiPort über eine seiner beiden seriellen Schnittstellen. Daten, die von dort kommen, stellt er im Netzwerk zur Verfügung und umgekehrt. Um den Bot anzusprechen, öffnet man eine TCP-Verbindung auf die IP-Adresse des WiPort (Port 10002). Am Protokoll zwischen Bot und PC hat sich gegenüber dem USB-2-Bot-Adapter [3] nichts geändert.

Der USB-Adapter gehört übrigens damit nicht zum alten Eisen. Er lässt sich auch an das Erweiterungsmodul anschließen. Ein kleiner Kippschalter wählt aus, ob der USB-2-Bot-Adapter – wie bisher – direkt mit dem Mikrocontroller verbunden ist oder ob der WiPort die Kommunikation übernimmt. Ist das der Fall, so hängt der USB-2-Bot-Adapter an der anderen seriellen Schnittstelle des WiPort. So kann man auch einen falsch konfigurierten WiPort mit einem einfachen Terminalprogramm wiederbeleben. Wie das geht, beschreiben das WiPort-Handbuch und ein FAQ-Eintrag auf der Projektseite [1].

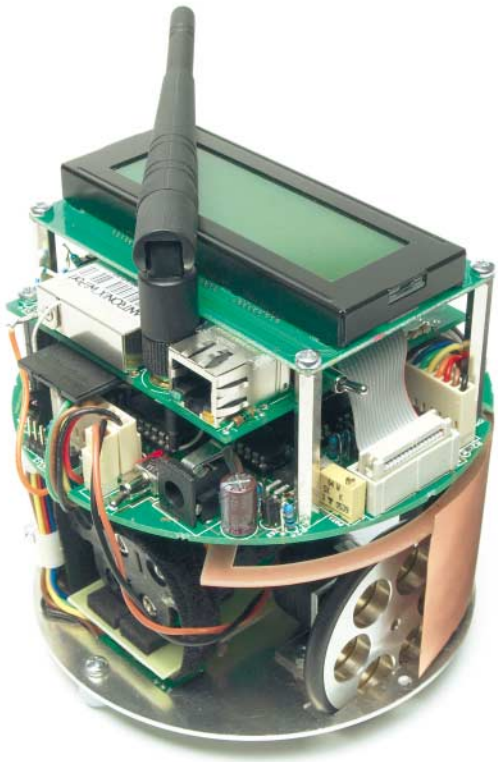
Normalerweise übernimmt der Device-Installer (siehe Soft-Link) von Lantronix die Konfiguration des Bausteins. Im Auslieferungszustand ist das WLAN-Interface deaktiviert, sodass ein PC ihn erst einmal nur über ein Netzwerkkabel erreicht. Neben den Netzwerkparametern (gewünschtes Interface, IP-Adresse, SSID, Verschlüsselung und Co.) im Network- und WLAN-Menü muss man auch die Baudrate der seriellen Schnittstelle im Channel-2-Menü festlegen. Unsere c't-Bot-Firmware braucht hier die Einstellung „57600" und „8N1". Hat alles geklappt und läuft der Mikrocontroller mit der c't-Firmware (entweder das Hex-File von der Projektseite oder ein eigenes Kompilat mit aktiviertem „BOT_2_PC_AVAILABLE"), sollte ein

```
telnet <WIPORT-IP> 10002
```

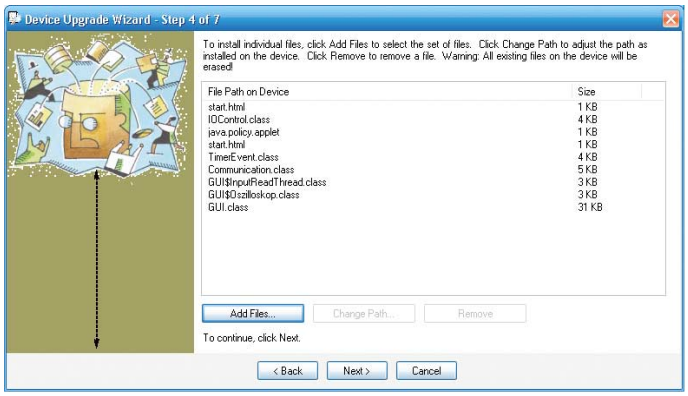
eine Verbindung herstellen. Der Bot liefert dann eine wahre Flut von kryptischen Zeichen ins Terminalfenster. Besser als wir Menschen versteht der c't-Sim dieses Protokoll. Ein Klick auf „Verbinde mit Bot per TCP …" und die Eingabe der korrekten IP-Adresse, und schon zeigt der Sim die Telemetriedaten des Roboters an, wie er das auch für einen per USB-2-Bot-Adapter angebundenen c't-Bot macht. Der Sim überwacht übrigens auch mehre Bots auf einmal.

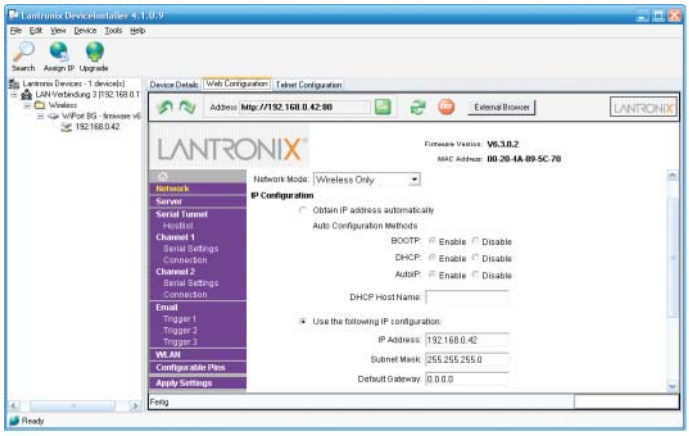
Soll der Roboter sich automatisch am Sim anmelden, muss man nur im Connection-Menü des Channel 2 die IP-Adresse des c't-Sim eintragen und Active Connention wählen. Sobald der Bot dann Daten schickt, baut der WiPort die Verbindung zum c't-Sim auf.

Bei geschlossener Transportklappe parkt der Tragarm genau unter dem Klappensensor. Ein rechteckiges Stück Platinenmaterial verschließt dann das Transportfach.



Das Erweiterungsmodul bringt den Bot ins LAN und WLAN. Aber auch USB bleibt nach wie vor eine Option, um mit dem Bot zu kommunizieren (Schalter nach vorn). Zeigt der kleine Schalter nach hinten, kann man über den USB-2-Bot-Adapter den WiPort konfigurieren. Das Foto zeigt einen Prototyp.





Sobald der Device Installer den WiPort gefunden hat, kann er eigene Webseiten und das Steuerapplet für den c't-Bot auf den integrierten Webserver hochladen.

All inclusive

Jeder beliebige PC mit einem Webbrowser kann den c't-Bot – ohne weitere Software-Installation – kontrollieren, wenn man den im WiPort integrierten Webserver nutzt. Dieser bietet 1,2 MByte Platz für Webseiten und Java-Applets – auf die der Bot selbst aber ohne Weiteres keinen Zugriff hat. Der Device-Installer übernimmt den Transfer der Daten auf den Webserver. Der c't-Sim lässt sich in einer abgespeckten Version als Java-Applet übersetzen. Lädt man dieses zusammen mit der von uns bereitgestellten HTML-Datei „index.html" auf den WiPort, so reicht ein gewöhnlicher Browser mit aktivierter Java-Unterstützung aus, um den Bot zu überwachen und zu steuern.

Damit der volle Funktionsumfang des Bot-Frameworks auch vom Webbrowser aus erreichbar ist, haben wir das Protokoll etwas überarbeitet. Der c't-Sim – egal ob als Java-Applikation oder als Applet – kann nun mit einem speziellen Kommando einzelnes Verhalten auf dem Bot aufrufen. So lassen sich Aufträge wie „Drehe Dich um 80 Grad" oder „Folge einer schwarzen Linie" an den Bot übermitteln. Mit diesen Funktionen sollte es sowohl möglich sein, Bot-Intelligenz auf den PC auszulagern als auch komplexere Verhaltensabläufe zu steuern. Spinnt man diesen Gedanken weiter, so ließen sich auch komplette Befehlssequenzen alias Bot-Programme auf die Flash-Speicherkarte übertragen. Ein paar zusätzliche Elemente wie Schleifen und If-Abfragen müsste man wohl noch programmieren, hätte dann aber eine eigene Bot-Sprache.

Wem der Mikrocontroller zu schwach auf der Brust und der PC zu weit vom Bot weg ist, der kann auch die Rechenleistung des WiPort für eigene Programme nutzen. Auf Anfrage bietet Lantronix ein Software-Entwicklungskit für den WiPort an, mit dem sich die Firmware an eigene Bedürfnisse anpassen lässt.

Speicher satt

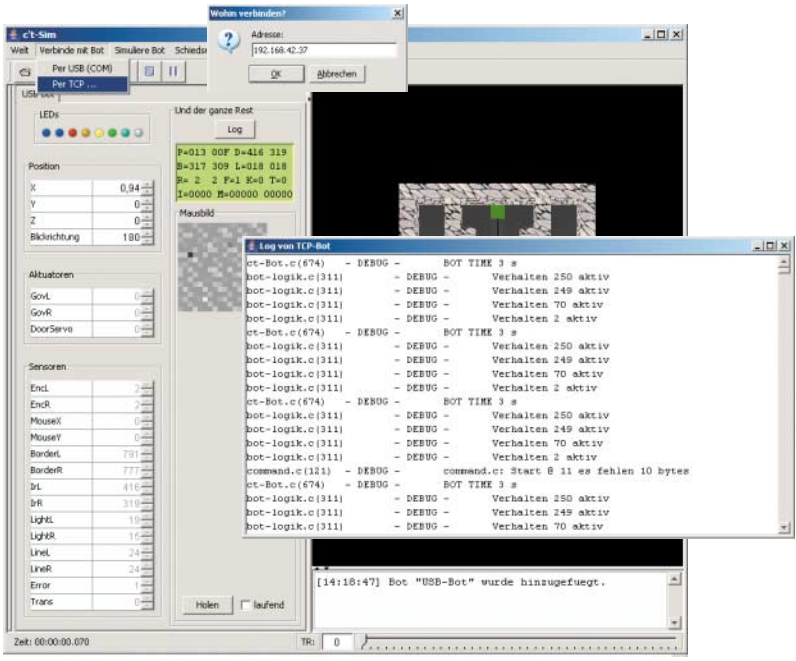
Nur 2048 Byte – jawohl, Byte und nicht KByte – misst der Arbeitsspeicher des c't-Bot. Soll er beispielsweise eine Karte seiner Umgebung erstellen, so reicht der Speicher nicht weit. Bei einer Auflösung von nur einem Punkt (1 Byte) alle 2 cm belegt schon eine Karte von 64 cm × 64 cm den halben RAM.

Der Karten-Slot auf dem Erweiterungsmodul nimmt gewöhnliche SD- und MMC-Karten auf. Da alle Schnittstellen des Mikrocontrollers bereits belegt sind, haben wir in die Trickkiste gegriffen und das Karten-Interface parallel zum Maussensor geschaltet. Damit sich die beiden nicht in die Quere kommen, klemmt ein Tri-State-Puffer dem Maussensor den Takt (SCLK) ab, wenn die Speicherkarte an der Reihe ist. Dazu muss man den Pin 4 des Maussensorsteckers von ST7 auf das Erweiterungsmodul (J11) umstecken. Umgekehrt kappt ein weiterer Puffer den Datenausgang der Karte. Wer den Schaltplan auf Seite 191 genauer studiert, findet zwischen den beiden ENA-Leitungen und Karte respektive Maussensor noch zwei Flipflops. Diese sorgen dafür, dass nach einem Reset keines der Geräte die MISO-/MOSI-Leitungen blockiert, denn diese werden für das Programmieren des Chips benötigt.

Die Demo-Firmware spricht die Karten im sogenannten SPI-Modus an. Daher spielt es auch keine Rolle, ob man zu einer SD- oder MMC-Karte greift – SPI sprechen beide Typen. Flash-Karten gewähren keinen byteweisen Zugriff auf die gespeicherten Daten. Sie lesen und schreiben immer ganze Blöcke (beispielsweise 512 Byte). Diese muss der Mikrocontroller im RAM puffern und nach einer Modifikation wieder komplett in die Karte schreiben. So kommt schnell eine große Menge an Transfers zusammen und das beschäftigt den Bot durchaus eine Weile. Der Leser Timo Sandmann hat die Routinen für den Low-Level-Zugriff auf die Karte als Assembler-Code beigesteuert. So schafft der Controller je nach Karte einen 512-Byte-Seitenwechsel in 3,6 bis 5,5 ms.

Dateisystem

Normalerweise formatieren PCs und Kameras Flash-Karten mit einem Dateisystem wie FAT(16) oder FAT32. Der Code für das Auswerten des Dateisystems ist jedoch recht groß und umständlich. Aufgrund der begrenzten Ressourcen des ATmega32 haben wir keinen FAT-Treiber implementiert, sondern behelfen uns mit einem Kniff: Wenn der Datenträger nicht fragmentiert ist, liegt eine Datei, wie man sie vom PC her kennt, Byte für Byte an aufeinanderfolgenden Adressen der Speicherkarte. Ist bekannt, wo eine Datei beginnt,



Eine abgespeckte Version des c't-Sim startet auch als Java-Applet direkt vom Webserver des Erweiterungsmoduls. Zur Überwachung des Bot reicht ein normaler Webbrowser.

kann man sie lesen und schreiben – natürlich ohne Veränderung der Größe –, ohne sich um das Dateisystem zu kümmern. Unser Mini-FAT-Treiber (mini-fat.c) durchsucht die Karte blockweise, bis er den Anfang einer zuvor auf dem PC markierten Datei findet.

Idealerweise formatiert man zuerst die Karte auf dem PC mit FAT als Dateisystem und kopiert dann eine vorbereitete Datei darauf. Für den Kartographie-Algorithmus findet sich eine solche Datei im Contrib-Verzeichnis des Quelltextes. Aus Platzgründen haben wir sie jedoch komprimiert. Wer eigene Dateien braucht, erstellt sie mit dem PC-Binary des c't-Bot:

```
ct-Bot.exe -c <DATEINAME> <ID>
                          <GRÖSSE>
```

Die ID bilden dabei drei ASCII-Zeichen, die man später der Funktion mini_fat_find_block() übergibt. ct-Bot.exe konvertiert mit dem Schalter „-M <DATEINAME>" übrigens auch die Spezialdateien, in denen der Bot seine Karte ablegt, in ein PGM-Bild, sodass übliche Bildbetrachtungsprogramme das Ergebnis der Kartographie darstellen können.

Wohin mit den Daten ?

Auch wenn dem Bot mit einer Flash-Speicherkarte Hunderte MByte Speicher zur Verfügung stehen, sind beim Programmieren die Eigenschaften der vier verschiedenen Speicherarten zu beachten. Variablen, auf die der Code oft zugreift, gehören in den SRAM, auch wenn dort nur 2048 Byte hineinpassen. Muss man ein paar einzelne Bytes – beispielsweise für Regelparameter – auch über einen Reset hinwegretten, legt man diese am besten im EEPROM (1024 Byte) ab. Der Zugriff ist etwas umständlicher als auf Daten im SRAM, erfolgt jedoch byteweise. Die Daten bleiben auch nach dem Abschalten der Versorgungsspannung erhalten. Die 32 KByte Flash, die der ATmega32 mitbringt, sind dem Programmcode vorbehalten und im laufenden Betrieb nur sehr umständlich beschreibbar. Karten, Logdateien und andere große Daten gehören auf die Flash-Karte. Allerdings erfolgt der Zugriff immer in Blöcken zu je 512 Byte und ist damit relativ langsam.

Wer die Speicherkarte für eigene Programme nutzen will, sollte daher mit so wenig Seitenwechseln wie möglich auskommen. Noch praktischer wird es, wenn man den ATmega32 durch den pinkompatiblen ATmega644(P) ersetzt, für den sich unser Framework auch übersetzen lässt. Dieser hat 4096 Byte SRAM, sodass beispielsweise sechs Seiten zu je 512 Byte im Puffer bleiben können. Im Code-Archiv findet sich bereits ein kleiner Speicherverwalter, der sich darum kümmert, bei Bedarf Seiten nachzuladen oder wieder auf die Karte zu schreiben. Die Datei Documentation/mmc-vm.txt beschreibt, wie man ihn benutzt.

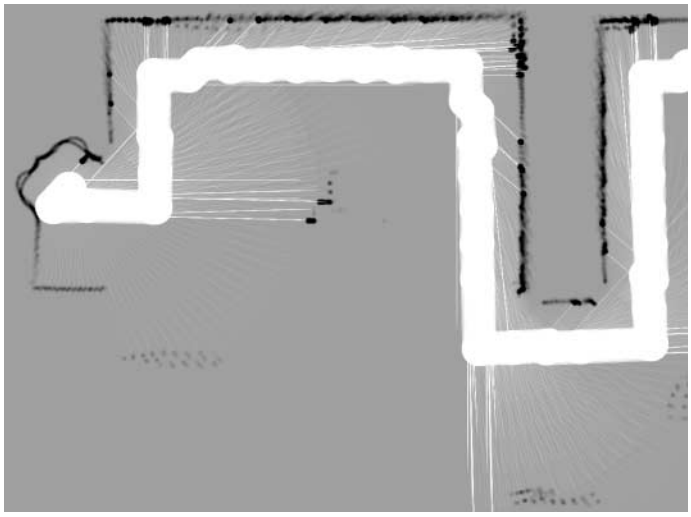
Pacman lebt

Für die Verriegelung des Transportfaches haben wir eine schlichte und preiswerte Lösung mit wenig mechanischem Aufwand gewählt. Ein Servo (Futaba S3107) kommt kopfüber von oben in das noch freie Loch in der Mitte der c't-Bot-Platine. Der D-förmige Tragarm wird direkt mit dem Servo verschraubt und bewegt sich rund 10 mm unterhalb der Platine und knapp oberhalb der Abstandssensoren. Er schiebt ein rechteckiges Stück Platinenmaterial als Klappe vor das Transportfach.

Der Tragarm sollte dabei so auf der Servoachse sitzen, dass er in geöffneter Stellung von innen an die rechte vordere Stütze des Bot anschlägt und bei geschlossener Klappe von außen an die einzelne hintere Stütze. In geschlossener Position steht der Tragarm genau unterhalb des Klappensensors auf der Unterseite der Hauptplatine.

Auf dem c't-Bot geht es mittlerweile recht eng zu, sodass man den Stecker des rechten Abstandssensors aus- und die Kabel direkt auf die Platine einlöten muss, damit die Klappe an dem Stecker nicht hängen bleibt.

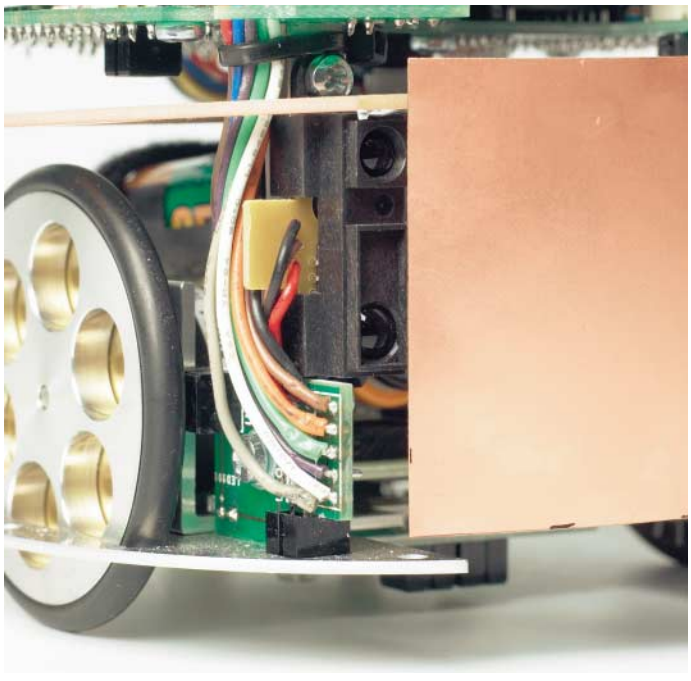
Da sowohl der Tragarm als auch die Klappe mit Kupfer kaschiert und – anders als bei dem Prototyp auf dem Foto – verzinnt sind, kann man sie einfach miteinander verlöten. Dazu biegt man am besten zuerst das rechteckige Platinenstück in die richtige Form. Nun dreht man den Tragarm in die geschlossene Stellung (gegen den Uhrzeigersinn bis auf Anschlag) und fixiert dann die Klappe mit zwei winzigen Lötpunkten. Sie sollte dabei weder einen der beiden Abstandssensoren abdecken noch auf der Grundplatte aufstehen. Sollte die Klappe in geöffnetem Zustand den rechten Sensor blockieren, so muss man sie nochmals verschieben. Steht die optimale Position fest, verbinden zwei dicke Zinnspuren die Klappe oben und unten mit dem Tragarm.



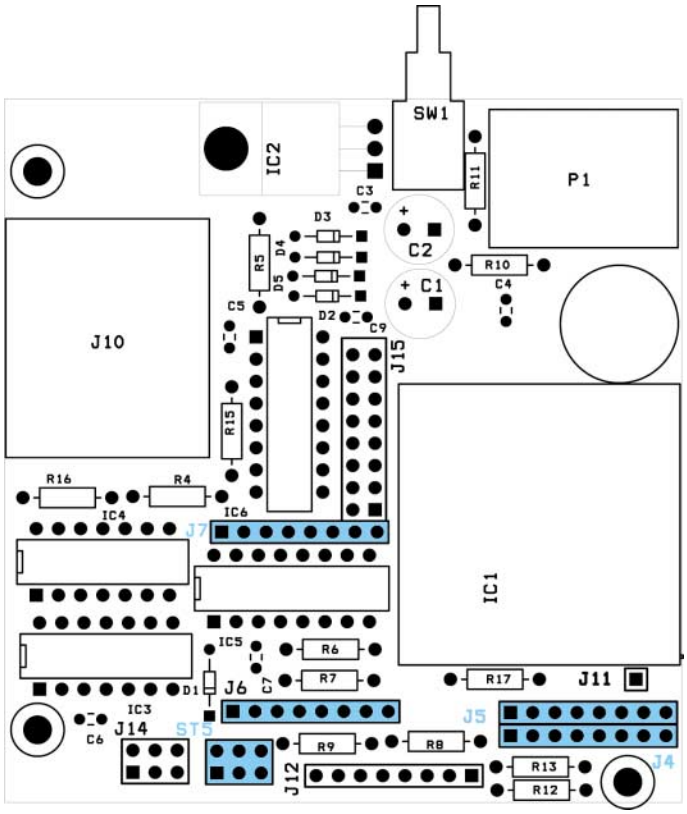
Ein Flash-Speicherkärtchen stellt dem c't-Bot genug Speicherplatz zur Verfügung, sodass er wie sein virtueller Kollege auch hoch aufgelöste Karten seiner Umgebung erstellen kann.

Bevor man zu ersten Tests bei eingeschaltetem Bot schreitet, muss man den Servo an die Hauptplatine (J1) anschließen. Achtung, die Pin-Belegung des Futaba-Steckers entspricht nicht der von J1. Mit einer Stecknadel lassen sich die Plastikhäkchen des Steckers entriegeln, um das rote und das schwarze Kabel zu tauschen: An Pin 1 von J1 kommt das rote Servo-Kabel, an Pin 2 das schwarze und an Pin 3 das weiße.

Unsere Demo-Firmware öffnet die Klappe, sobald sie das Fernbedienungssignal „Channel –" bekommt und schließt sie bei „Channel +". Nach einem kurzen Timeout, der dem Servo ausreicht, um die Klappe zu verfahren, schaltet die Firmware den Servo wieder ab. Das spart Strom.



Damit die Klappe am GP2D12-Sensor vorbeikommt, muss man dessen Stecker aus- und die drei Kabel direkt einlöten.

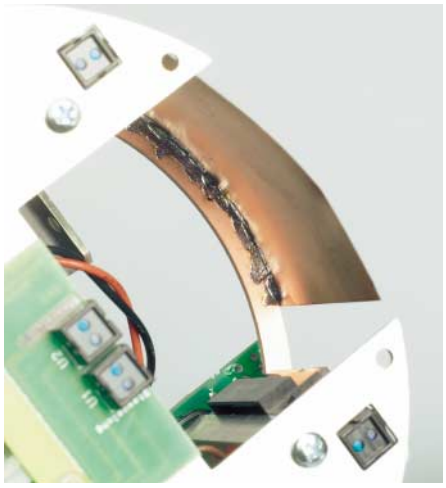


Alle Steckverbinder, die blau unterlegt sind, werden von unten auf die Platine gelötet. Sie verbinden das Erweiterungsmodul mit der Hauptplatine.

Wenn die Klappe geschlossen ist, detektiert der CNY70-Sensor auf der Unterseite der Hauptplatine den Tragarm. Im Code lässt sich das einfach auslesen, indem man prüft:

```
if (sensDoor == DOOR_CLOSED)
```

Meldet der Sensor keinen Vollzug, könnte die Klappe an einem Hindernis hängen geblieben sein. Solange der Servo versucht sie zuzudrücken, nimmt er relativ viel Strom auf. Eine kleine Schaltung auf der Hauptplatine prüft, ob der Servostrom eine bestimmte Schwelle überschreitet. Ist dies der Fall, liefert die Variable sensError den Wert 0. Um die Schwelle anzupassen, muss man die Widerstände R30 und R34 anpassen. Wird R34 im Verhältnis zu R30 kleiner, spricht die Schaltung schon bei geringeren Strömen an. Achtung: Das Ausgangssignal dieser Strommessschaltung wird mit dem der Batterieüberwachung verknüpft. Der Wert 0 von sensError kann also auch bedeuten, dass die Batteriespannung unter einer bestimmten Schwelle – einstellbar über R33 und R29 – liegt. Für eine Fallunterscheidung kann die Software einfach die Variable einmal auslesen, bevor sie den Servo aktiviert.



Je eine dicke Zinnspur in der oberen und unteren Ecke zwischen Tragarm und Klappe schaffen eine stabile Verbindung.

Klappe, die Erste!

Um beispielsweise eine Filmdose im Transportfach einzufangen, muss der Bot eine ganze Reihe von Aufgaben nacheinander ausführen. Seine Sensoren liefern dazu wertvolle Hinweise, wie folgendes Beispiel zeigt. Angenommen, Bot und Filmdose stehen alleine auf einem leeren Tisch. Nun kann sich der Bot so lange gegen den Uhrzeigersinn drehen, bis sein linker Sensor die Dose erblickt. Abstand und aktuelle Blickrichtung merkt sich der Bot. Noch ein Stück weiter gedreht, und die Dose verschwindet wieder aus dem mit 3 Grad recht engen Blickfeld des linken Sensors und taucht dafür beim rechten auf. Damit die Dose genau mittig vor dem Bot steht, muss er sich nun ein kleines Stück zurückdrehen. Nun ist es an der Zeit, die Klappe zu öffnen und so lange auf die Dose zuzufahren, bis die Lichtschranke im Transportfach eine 1 liefert (sensTrans == TRANS_FULL). Klappe zu, Affe tot. Der Code dazu findet sich im Framework als behaviour_catch_pillar() und wird per Fernbedienungstaste „4" aktiviert.

Ausblick

Dem c't-Bot eine Kamera zu spendieren wäre sicher auch noch eine reizvolle Erweiterung. Wir haben uns jedoch entschieden, diese nicht im Rahmen des c't-Projektes anzubieten. Der Grund hierfür ist recht einfach: Eigenbauten sind softwaretechnisch zu komplex für diese Artikelreihe, und darüber hinaus gibt es am Markt bereits einige Kameralösungen speziell für Roboter. Diese bestehen meist aus einer Kamera und einem leistungsfähigen Prozessor, der sich um die Bildverarbeitung kümmert. Die Hersteller liefern auch gleich die nötige Software mit. Als Beispiel kann hier die CMUCAM2 oder die POB-EYE dienen. Beide können über eine serielle Schnittstelle mit dem ATmega kommunizieren. Dieser muss dann nur noch der Kamera mitteilen, dass sie beispielsweise ein viereckiges Symbol verfolgen soll. Das Kameramodul liefert dann zurück, wo es ein solches Symbol entdeckt hat. Leider sind beide Systeme nicht gerade preiswert. Möchte man allerdings eine preiswerte (Web-)Cam einsetzen, muss man sich selbst mit der Auswertung der Bilder herumschlagen und dafür genug Rechenleistung vorhalten. Ist das nicht möglich, kann man immer noch die Videobilder per Funk an den PC übertragen und dort auswerten.

Wer eigene Hardware-Erweiterungen für den c't-Bot konzipiert, kann sie entweder per I2C-Bus [4] anschließen oder die serielle Schnittstelle nutzen, an der derzeit das WLAN-Modul oder der USB-2-Bot-Adapter hängt. Alternativ dazu eignet sich auch der SPI-Port des ATmega, den bereits der Maussensor und die Speicherkarte nutzen. Deshalb müsste man noch eine Chip-Select-Leitung freischaufeln. Wer den einzigen noch nicht benutzten Pin (PWM2) dafür nicht verwenden will, kann ein paar LED-Kanäle opfern.

(bbe)

Literatur

[4] Projektseite: www.ct-bot.de

[2] Benjamin Benz, Thorsten Thiele, Funkbrücke, Umsetzer von WLAN nach RS-232 im Eigenbau, c't 26/04, S. 228, www.heise.de/ct/ftp/projekte/com2lan

[3] Benjamin Benz, Thorsten Thiele, An der Leine, Debuggen des c't-Bot über USB, c't 7/06, S. 223

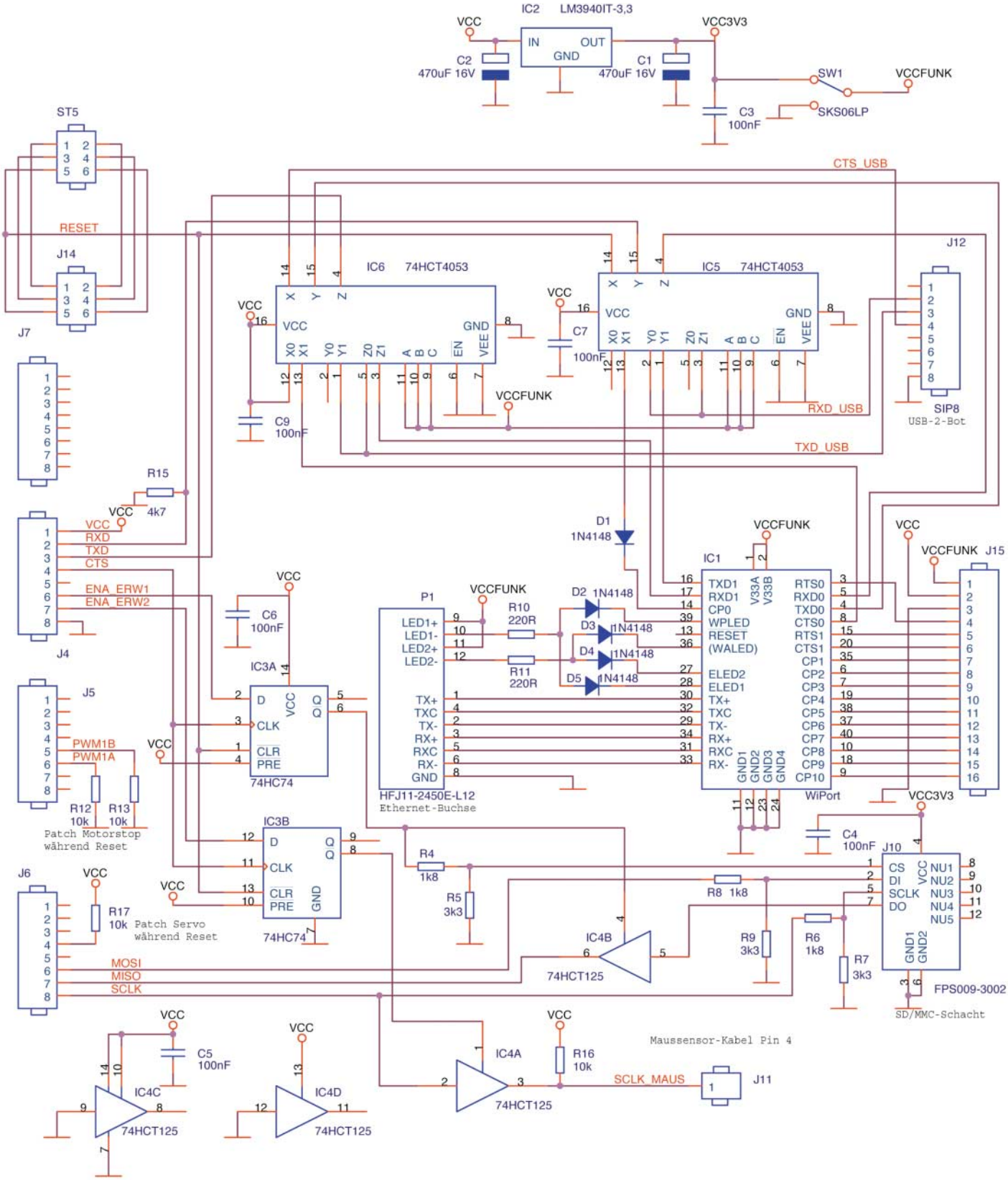
[4] Benjamin Benz, Nervensystem, Programmierung des c't-Bot von der Pike auf, c't 6/06, S. 264

Soft-Link 0702184

Stückliste

Bauteil	Wert
C1,C2	470uF/16V
C3,C4,C5,C6,C7,C9	100nF
D1,D2,D3,D4,D5	1N4148
IC1	WiPort (WP2001000G)
IC2	LM3940IT-3,3
IC3	74HC74
IC4	74HCT125
IC5,IC6	74HCT4053
J4[1], J5[1], J6[1], J7[1]	Buchsenleisten
J12	Stiftleiste gewinkelt
J10	SD-Kartenschacht
J11	Einzelstift, gewinkelt
J14	Stiftwanne, gewinkelt
J15	nicht bestueckt,
P1	RJ45-Buchse mit Ferrit und LED
R4,R6,R8	1,8 kOhm
R5,R7,R9	3,3 kOhm
R10,R11	220 Ohm
R12,R13,R16, R17	10 kOhm
R15	4,7 kOhm
ST5[1]	Buchsenleiste, zweireihig
SW1	Schalter
Servo	Futaba S3107

[1] Bestückung erfolgt von unten



Zwei Flipflops und Tri-State-Puffer sorgen dafür, dass sich Speicherkartenschnittstelle und Maussensor nicht ins Gehege kommen. Das WLAN-Modul lässt sich per Schalter deaktivieren. Dann ist der Bot über den USB-2-Bot-Adapter erreichbar. Sind Bot und WiPort miteinander verbunden, hilft der USB-2-Bot-Adapter beim Konfigurieren des WiPort. c't