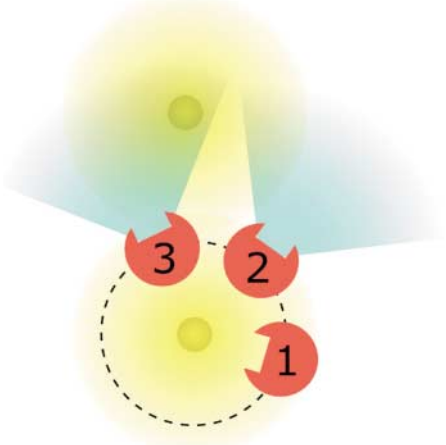


Die beleuchtete Säule nimmt der Bot fest ins Visier (1). Nach Drehung und Bogenfahrt von 20 Zentimetern prüft er, ob sich im grünen Sektor eine weitere Slalomstange befindet (2) – falls nicht, setzt er die Fahrt fort und prüft anschließend erneut (3).

tanz vor sich hat (SWEEP_STATE_CHECK). Ist das der Fall, wählt er diese als neues Slalomziel (SLALOM_STATE_CHECK_PILLAR) und kehrt die Orientierung um. Andernfalls wendet sich der Bot wieder reumütig der alten Säule zu und beginnt die Prozedur von neuem mit dem nächsten Bogenstück von 20 Zentimetern Länge.

Hausaufgaben

Mit der beschriebenen Programmierung kommt der c't-Bot auch mit recht unordentlich gesteckten Kursen zurecht. Trotzdem bleiben noch viele Stellen und Parameter im Programm, an denen sich herumschrauben lässt, um das Verhalten des Bot weiter auszufeilen. Vielleicht erweist es sich als günstiger, wenn er sich nicht so häufig umschaut oder wenn er die Säulen im engeren Bogen nimmt? (Aber Vorsicht, unterhalb von acht Zentimetern Distanz funktionieren die Distanzsensoren nicht mehr zuverlässig!) In den Kommentaren zum Code finden sich an einigen Stellen Hinweise darauf, wo es sich anbietet, Verhaltensweisen zu erweitern und zu verbessern. Codepatches, die den Bot im Slalom auf der Ideallinie halten, veröffentlichen wir wie immer gerne auf der Projektseite [3]. Der vorgestellte Rahmen einer einfachen Subsumptions-Architektur kann aber auch leicht als Grundlage für völlig anderes Verhalten benutzt werden. (pek)

Literatur

[1] W. Grey Walters kybernetische Schildkröten: http://personal.pitnet.net/usr/gasperi/walter.htm

[2] Benjamin Benz, Peter König, Lasse Schwarten, Drängelnde Spielgefährten, Kollisionen und Sensoren für den c't-Sim, neues Verhalten für den c't-Bot, c't 5/06, S. 224

[3] Webseite zum c't-Bot-Projekt: www.heise.de/ct/ftp/projekte/ct-bot

[4] Rodney A. Brooks, A Robust Layered Control System for a Mobile Robot, MIT AI Memo 864, September 1985 (PDF siehe Soft-Link)

Soft-Link 0607218

Benjamin Benz, Thorsten Thiele

An der Leine

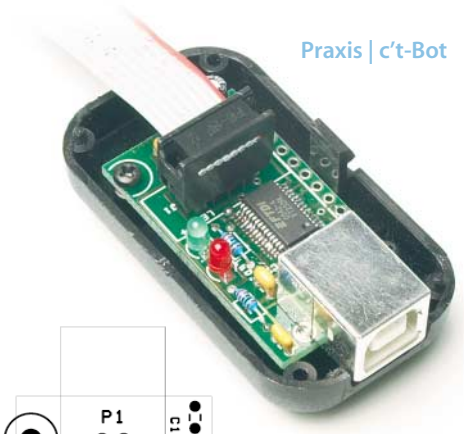
Debuggen des c't-Bot über USB

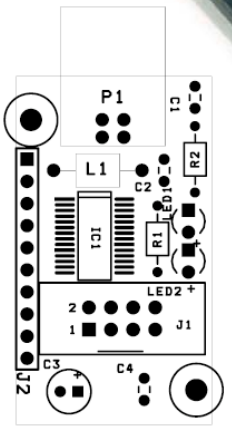
Ein Fussel auf dem Spielfeld mag für seltsame Sensorwerte sorgen, Streulicht oder unterschiedlich stark reflektierende Oberflächen verrauschen die Messdaten. Außerhalb der Simulation ist es schwer, Fehler im eigenen Verhaltenscode zu finden: Sensordaten kann der Roboter bisher nur auf dem eigenen Display darstellen. Für eine genauere Analyse braucht er eine Verbindung zum PC. Für rund 15 Euro erfüllt unser USB-2-Bot-Adapter diesen Zweck. Seine maximal fünf Meter lange USB-Leine gewährt dem Bot genug Freiraum, um Linien zu verfolgen oder Slalom zu fahren.

Herzstück des Adapters ist der USB-Seriell-Wandler FT232RL von FTDI. Die zugehörigen Treiber gaukeln dem Betriebssystem (Windows, Linux, Mac OS X) einen seriellen Port vor – in der Realität fehlt dieser aktuellen Rechnern häufig. Mit dem Roboter kommuniziert der Chip über eine zweiadrige asynchrone serielle Schnittstelle, die Datenrate muss man sowohl dem PC-Treiber als auch dem Mikrocontroller mitteilen. Zum einfachen Protokollieren sollten 9600 Bps reichen, mehr ist auch möglich. Im Unterschied zu den bei PCs üblichen RS-232-Ports liegen die Signalpegel des Mikrocontrollers auf TTL-Niveau (0 und 5 V statt –12 und 12 V).

Da der FT232RL nur in einem SMD-Gehäuse zu haben ist, bieten eMedia (www.emedia.de) und Segor Electronics (www.segor.de) eine Platine mit vorbestücktem Wandler-Chip an, die übrigen Bauteile lötet man selbst ein. Ein Flachbandkabel verbindet dann J1 des Adapters mit J4 des Roboters, die Pin-Belegung beider Stecker stimmt überein.

Unsere Beispiel-Firmware sendet alle Sensor- und Aktuatorwerte mit 9600 Bps (8N1) über die serielle Schnittstelle, sobald man den Schalter `#define BOT_2_PC_AVAILABLE` in ct-Bot.h aktiviert. Dabei kommt dasselbe Protokoll zum Einsatz, das bereits simulierter Bot





Der SMD-Chip auf dem USB-2-Bot-Adapter ist bereits vorbestückt. Das Einlöten der restlichen Bauteile erfordert keine besonderen Kenntnisse.

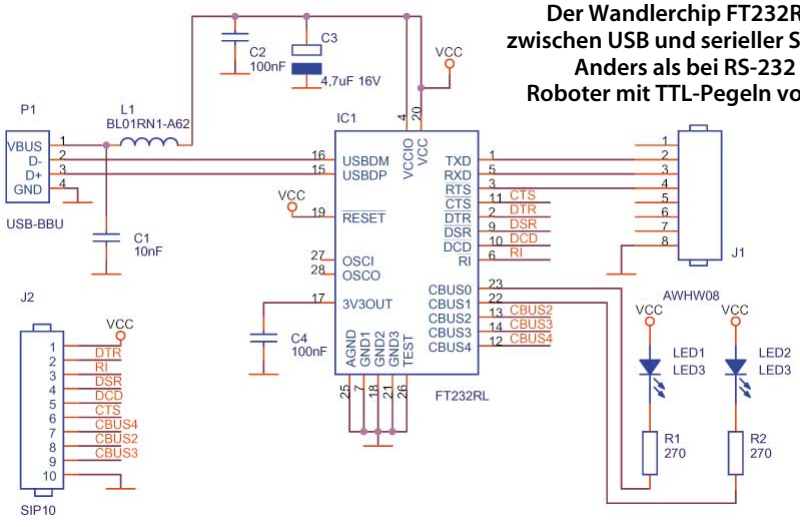
und c't-Sim sprechen. Daher ist der c't-Sim ohne weiteres in der Lage, alle Daten eines realen Bot anzuzeigen.

Unter Windows muss man sich zwischen zwei Treibern entscheiden: Der eine (D2XX) bietet eine Programmierschnittstelle an. Über diese greift der c't-Sim direkt auf den realen Roboter zu und zeigt dessen Sensor- und Aktuatorwerte an. Der andere Treiber (VCP) emuliert einen COM-Port. Wer dessen Binärdatenstrom mitlesen will, benötigt ein HEX-fähiges Terminalprogramm. Die Installation der Windows-Treiber und des API beschreibt die Projektseite.

Ein aktuelles Linux bringt die Treiber für den Chip bereits mit und richtet automatisch dafür das serielle Device /dev/ttyUSB0 ein. Das Hilfsprogramm socat – Link auf der Projektseite – leitet die seriellen Daten per TCP/IP an den c't-Sim weiter:

```
socat TCP4:localhost:10001 /dev/ttyUSB0,raw,b9200
```

Theoretisch könnte auch unter Linux der c't-Sim direkt mit dem c't-Bot kommunizieren, allerdings ist dort die Installation des D2XX-Treibers und der API problematisch. (bbe)



Der Wandlerchip FT232RL vermittelt zwischen USB und serieller Schnittstelle. Anders als bei RS-232 arbeitet der Roboter mit TTL-Pegeln von 0 und 5 V.