

# Debuggen des c't-Bot über USB

## An der Leine

**Praxis & Tipps | Praxis .. Uhr**  
Benjamin Benz, Thorsten Thiele

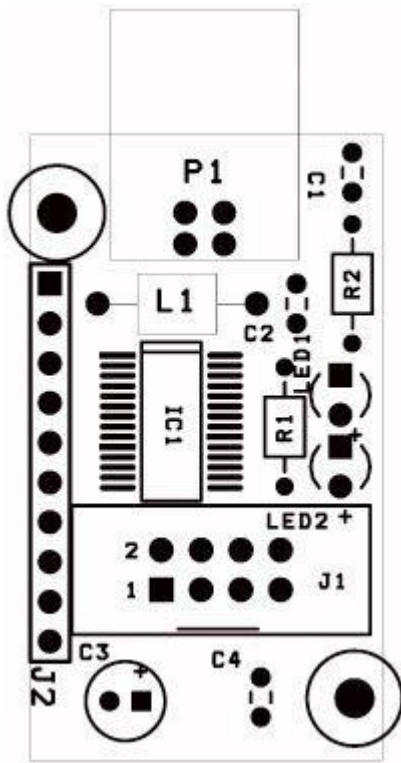
Ein Fussel auf dem Spielfeld mag für seltsame Sensorwerte sorgen, Streulicht oder unterschiedlich stark reflektierende Oberflächen verwechseln die Messdaten. Außerhalb der Simulation ist es schwer, Fehler im eigenen Verhaltenscode zu finden: Sensordaten kann der Roboter bisher nur auf dem eigenen Display 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.



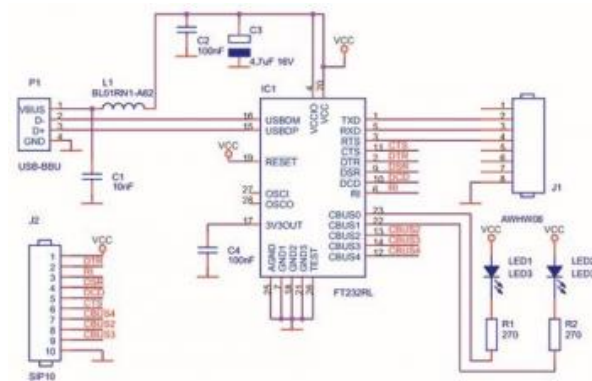
USB-2-Bot-Adapter

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[1]** und **Segor Electronics[2]** 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.



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



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.

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 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[3]**)

## URL dieses Artikels:

<http://www.heise.de/-290394>

Unsere Beispiel-Firmware sendet alle Sensor- und

**Links in diesem Artikel:**

[1] <http://www.edia.de>

[2] <http://www.segor.de>

[3] <mailto:bbe@ct.de>

*Copyright © 2006 Heise Medien*