Benjamin Benz, Thorsten Thiele

Fernkontrolle

Mikrocontroller zum Messen und Steuern über das LAN, Teil 2

Der c't-Mikrocontroller-im-LAN misst analoge und digitale Signale, steuert über zwanzig digitale Ports und kommuniziert mit seriellen Geräten. Aufgesteckt auf das COM-auf-LAN-Modul lässt er sich über das Netzwerk fernsteuern und mit einem beliebigen Webbrowser überwachen.

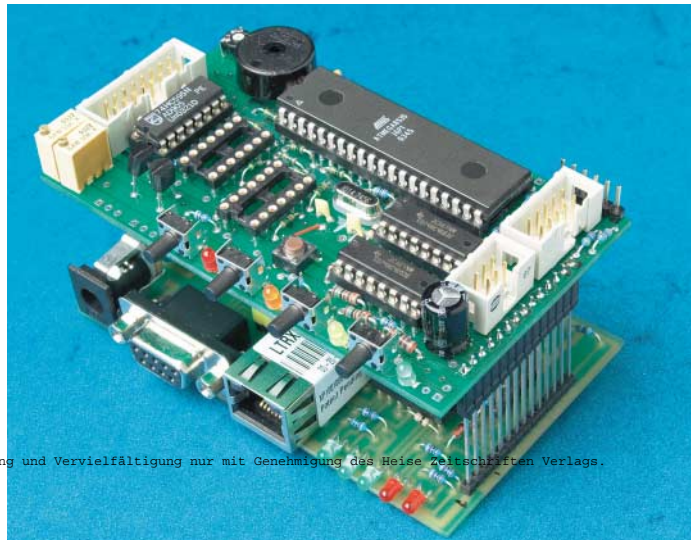
Aufbauend auf dem im letzten Heft [1] vorgestellten COM-auf-LAN-Adapter übernimmt der Mikrocontroller (MCU) im zweiten Teil des c't-Projekts Mess- und Steueraufgaben. So lassen sich beispielsweise Temperaturen messen und auf einem entfernten PC anzeigen. Das System eignet sich auch als Fernschalter und Signalmelder. Außerdem stellt es fast alle seine Ports für eigene Experimente und Erweiterungen zur Verfügung. Zusammen mit dem XPort auf dem COM-auf-LAN-Adapter (A-Modul) erhält man so einen relativ günstigen Mikrocontroller mit Ethernet-Schnittstelle, ohne selbst den ganzen IP-Stack und eine Ethernet-Schaltung implementieren zu müssen.

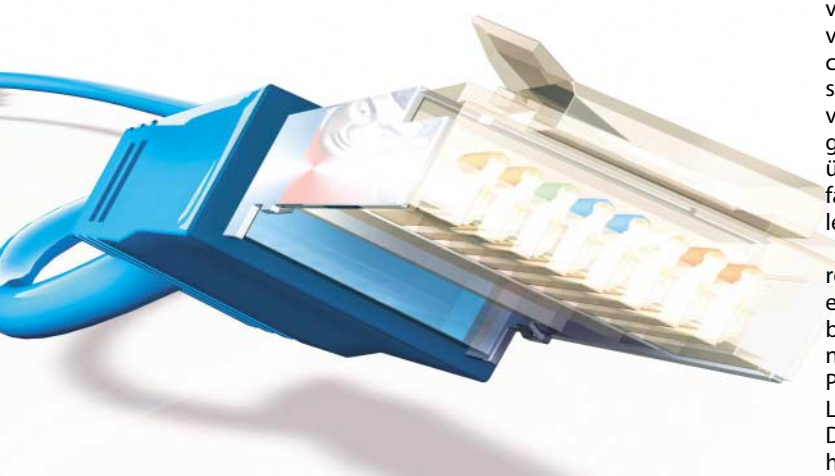
Das Kernstück des B-Moduls bildet ein 8-Bit-Mikrocontroller von Atmel. Der ATMega 8535 besitzt 32 Register, 512 Byte SRAM, 512 Byte EEPROM, 8 KByte Flash-Speicher und einen Befehlssatz mit 130 Instruktionen. Da der Chip fast alle Befehle in einem Taktzyklus ausführt, erreicht er bei einem Takt von 14,7 MHz auch maximal 14,7 MIPS, die für Aufgaben wie das Messen von Temperaturen, die Ausgabe auf einem Text-Display, das Steuern von Relais oder die Kommunikation im LAN mehr als ausreichen.

Das B-Modul sitzt Huckepack auf dem c't-COM-auf-LAN-Adapter. Über den seriellen Tunnel des XPorts und ein kleines Java-Applet kann jeder PC im LAN den Mikrocontroller des B-Moduls fernsteuern.

Kontakt und Kontrolle

An vier Ports mit jeweils acht Pins stellt der Mikrocontroller insgesamt 32 Ein-/Ausgabe-(I/O-) Leitungen zur Verfügung, die die Firmware für unterschiedliche Aufgaben konfiguriert. So bietet zum Beispiel der Port A entweder acht digitale Ein-/Ausgänge oder aber Zugang zu dem integrierten achtkanaligen 10-Bit-Analog-Digital-Umsetzer.





Auf Port B liegt die In-System-Programmierschnittstelle (ISP), die Steuerlogik für die serielle Schnittstelle und deren CTS-Signal. Port C nutzen wir entweder für eigene I/O-Aufgaben oder für die Ansteuerung des Displays und eines Drehgebers. Die Datenleitungen der seriellen Schnittstelle (TTL-Pegel) liegen zusammen mit den Tastern und LEDs auf dem Port D.

Das B-Modul unterbricht die serielle Brücke des COM-auf-LAN-Adapters (alle Jumper von Leiste P5 auf dem A-Modul entfernen) und steuert mit zwei analogen Schalter-ICs (74HC4053), welches Gerät mit welchem anderen kommunizieren darf. Drei Fälle sind möglich: Der XPort spricht mit dem Mikrocontroller, um Messwerte und Steuerbefehle zu übertragen. Soll die Ethernet-Seriell-Brücke des A-Moduls genutzt werden, ist eine Verbindung von XPort und COM-Schnittstelle nötig. Schließlich kann auch der ATMega direkt mit einem Gerät an der COM-Schnittstelle Daten austauschen – beispielsweise, um Messwerte auszudrucken.

Um die nötigen Verbindungen herzustellen, sind fünf analoge elektronische Schalter nötig. Die drei Leitungen PB0 bis PB2 steuern diese (siehe Tabelle Konfiguration seriell). In jeder der drei Konstellationen zeigt die Tabelle ein „egal" für eine der Steuerleitungen. Für den Fall, dass der XPort direkt mit dem seriellen Pegelwandler in Verbindung steht, hat die Leitung PB0 aber doch eine nützliche Bedeutung: Sie schaltet den seriellen Eingang des Mikrocontrollers entweder auf den Ausgang des XPorts oder den des Pegelwandlers. Dadurch kann der AtMega 8535 die Kommunikation zwischen den beiden anderen belauschen und sich, sobald er eine bestimmte Kommandofolge erkennt, wieder in die Verbindung einklinken.

Den meisten modernen seriellen (RS-232-)Geräten reichen die Rx- und Tx-Leitung aus. Für bestimmte Anwendungen, bei denen zum Beispiel der Controller einem Gerät signalisieren will, dass es warten muss, sind aber auch die Leitungen der Flusskontrolle (RTS, CTS) auf dem B-Modul angeschlossen.

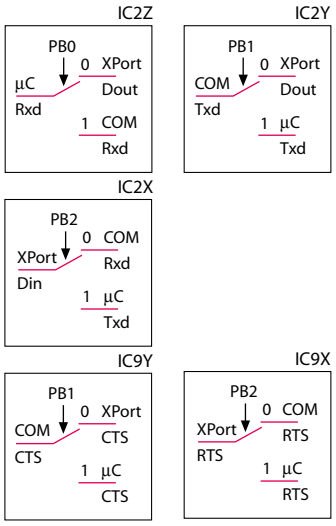
Sensibel und sprachbegabt

Fernthermometer mit LAN-Port kosten im Handel meist über 200 Euro. Unser B-Modul lässt sich deutlich preiswerter mit zwei PT100-Sensoren (an P6, P7) in ein 2-Kanal-Thermometer verwandeln. Dazu nutzen wir zwei der AD-Umsetzer des ATMega 8535 (PA0 und PA1). Der untere Punkt des Temperaturbereichs liegt typischer Weise bei 0 °C, die obere Grenze legen die beiden Potenziometer TR1 und TR2 über den Verstärkungsfaktor der Endstufe fest. Die Auflösung von 10 Bit der Analog-Digital-Umsetzer (ADU) reicht bei einem Messbereich von 0 °C bis 100 °C für eine Stufenbreite von zirka 0,1 °C aus. Wer andere analoge Sensoren anschließen oder den Messbereich verändern will, passt die Widerstandswerte der analogen Verstärker an (R2 bis R8, TR1 für Kanal 1 und R10 bis R16, TR2).

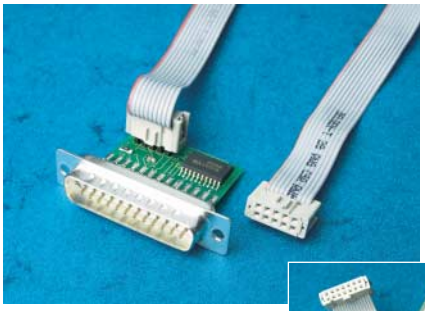
Ein Display, ein Piepser und verschiedene Eingabe-Optionen verwandeln die Black Box Mikrocontroller-im-LAN in ein eigenständiges Gerät. Für die Anzeige von ASCII-Text eignen sich viele gängige Text-LCD-Module, die über ein paralleles 8-Bit-Interface und einen eigenen Controller verfügen.

Um I/O-Leitungen einzusparen sitzt vor dem Display noch ein serielles Schieberegister – so belegt das Display mitsamt seinen Steuerleitungen nur vier Port-Pins. Die Ansteuerung des LCDs ist relativ einfach und im Demo-Quelltext (Soft-Link) anhand eines 4x20-Displays mit HD44780-kompatiblem-Controller beschrieben.

Das Beispielprogramm ist zugunsten der Lesbarkeit in C geschrieben. Für den ATMega existieren gleich mehrere C-Compiler, sodass es nicht unbedingt nötig ist, zur Programmierung auf Assembler zurückzugreifen, obwohl gewisse Aufgaben sicherlich effizienter in Maschinensprache lösbar wären. Die kostenlose Entwicklungsumgebung WinAVR (Soft-Link) nutzt den gcc-Compiler in einer Cross-Version für den ATMega 8535. Passend zum Compiler enthält das WinAVR-Kit eine vorgefertigte Header-Datei (io.h) mit den Definitionen der Register und Ports des AVR-Controllers. Atmel selbst bietet mit dem kostenfreien AVRStudio nur eine Entwicklungsumgebung für Assembler an (Soft-Link). Deren Debugger lässt sich allerdings auch mit von WinAVR übersetztem Code nutzen.

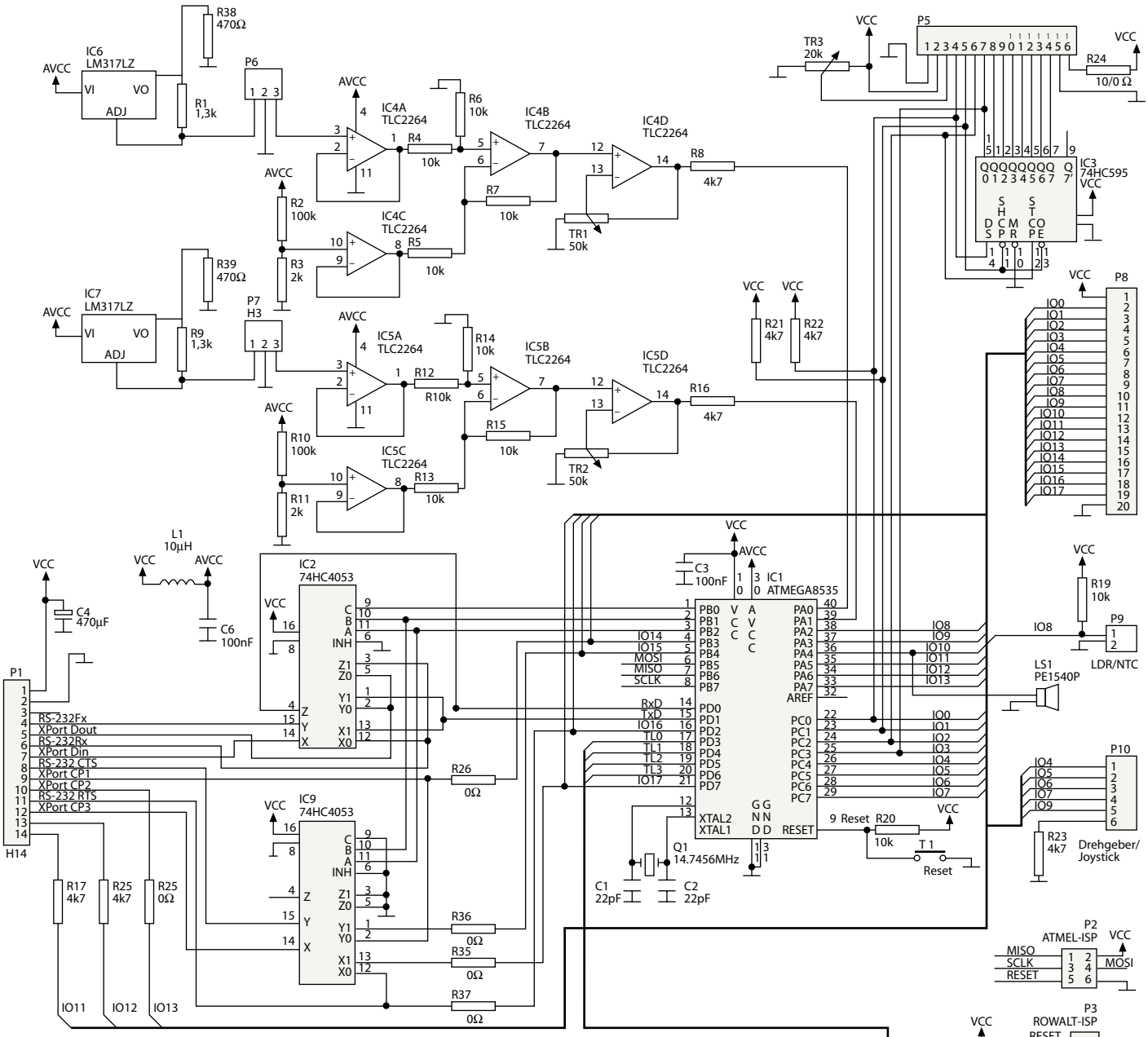


Auf dem A-Modul ist der XPort direkt mit dem RS-232-Pegelwandler verbunden. Damit auch der Mikrocontroller mit dem XPort und seriellen Geräten Daten austauschen kann, bricht das B-Modul diese Verbindung auf. Stattdessen kümmern sich fünf elektronische analoge Schalter um die korrekten Verbindungen zwischen den drei möglichen Kommunikationspartnern. Drei Portleitungen (PB0 bis PB2) steuern die fünf Schalter.



Mit einer Hand voll Bauteilen kann der Parallel-Port des PC das Flash-ROM des Atmel-Prozessors befüllen, ohne dass dieser aus dem B-Modul ausgebaut werden muss.

Ein 4x20-LCD mit integriertem Controller zeigt während der Entwicklung Debug-Informationen an und kann später zum Beispiel Messwerte darstellen.



Der 8-Bit-Mikrocontroller von Atmel besitzt einen integrierten 10-Bit-Analog-Digital-Umsetzer mit acht Kanälen. Zwei davon nutzen wir, um ein Ethernet-Thermometer zu realisieren.

Die vom Linker erzeugten Hex-Dateien überträgt das ebenfalls kostenlose Programm PonyProg (Soft-Link) in den Controller. Über die ISP-Schnittstelle kann der ATMega auch dann neue Software laden, wenn er bereits im Zielsystem eingebaut ist. Einen Adapter zur Programmierung über den Parallelport des PC haben wir im Rahmen des BlueMP3-Projekts vorgestellt [2]. Er kostet als Bausatz bei SEGOR-electronics zirka 6,60 Euro.

Bei der Programmierung des ATMega ist es unbedingt erforderlich, die Fuse-Bits des Prozessors korrekt einzustellen. Diese legen fest, woher der MCU seinen Takt bezieht und ob er weitere Schreib-/Lesezugriffe über den ISP-Port zulässt. Im Auslieferungszustand sind sie so gesetzt, dass der Prozessor, ohne einen Quarz zu benötigen, einen langsamen Systemtakt selbst generiert. Damit der Chip später mit dem richtigen Takt (14,7456 MHz) arbeitet, müssen die Fuses wie im Bild auf Seite 218 zu sehen gesetzt sein. Einen Prozessor mit falsch programmierten Fuse-Bits kann nur ein professionelles Programmiergerät wieder zurücksetzen.

Laufen lernen

Erste eigene Schritte in der Mikrocontroller-Programmierung sammelt man am besten mit der Ansteuerung der LEDs und der Abfrage der Taster. Hierzu sitzen auf der Platine vier Mikro-Taster (T2 bis T5), die zusammen mit jeweils einer LED an den Port-Pins PD3 bis PD6 hängen.

Die Steuerung der I/O-Pins und damit auch der LEDs übernehmen pro Port drei 8-Bit-Register. Im Data Direction Register (DDRx) entscheidet jedes Bit für

Pinbelegung B-Modul

P1 Verbindung A-Modul	
1	VCC (5V)
2	GND
3	V33 (3,3V)
4	RS-232 Tx
5	XPort Dout
6	RS-232 Rx
7	XPort Din
8	RS-232 CTS
9	XPort CP1
10	XPort CP2
11	RS-232 RTS
12	XPort CP3
13, 14	NC
P2 ISP-Anschluss (Atmel-Standard)	
1	MISO
2	VCC
3	SCLK
4	MOSI
5	RESET
6	GND
P3 ISP-Anschluss (Roland Walter)	
1	RESET
2	GND
3	VCC
4	SCLK
5	MISO
6	MOSI
P4 ISP-Anschluss (BlueMP3)	
1	MOSI
2	VCC
3	NC
4, 6, 8, 10	GND
5	RESET
7	SCLK
9	MISO
P5 Display	
1	GND
2	VCC
3	Kontrast
4	Select (PC0)
5	R/W (PC1)
6	Enable (PC2)
7	Data 0 (PC3)
8–14	Data 1–7
15	GND
16	Backlight
P6,P7 PT100-Fühler	
1	Versorgung (V+)
2	GND
3	Messeingang
P8 IO-Ports	
1	VCC
2–9	IO0-IO7 (PC0-PC7)
10–15	IO8-IO13 (PA2-PA7)
16, 17	IO14,IO15 (PB4,PB5)
18, 19	IO16,IO17 (PD2,PD7)
20	GND
P9 LDR/NTC	
1	V+
2	Data (IO8)
P10 Drehgeber	
1–4	IO4-IO7
5	IO9
6	V–

Der Mikrocontroller des B-Moduls nutzt die Ethernet-nach-Seriell-Brücke des ersten Teilprojekts zur Kommunikation mit einem PC. Für eigene Erweiterungen bietet er reichlich I/O-Ports und analoge Eingänge an.

einen Pin, ob er als Ein- (DDRx.y=0) oder Ausgang (DDRx.y=1) arbeitet. Für die vier LEDs setzt der C-Befehl:

```
DDRD |= 8+16+32+64;
```

die nötige Bit-Maske, ohne die Richtung der übrigen Port-Pins zu verändern. Eine Zeile mit #include „io.h“ am Anfang der C-Datei sorgt dafür, dass der Compiler die Namen der Register erkennt.

Das PORTx-Register legt dann fest, ob der Mikrocontroller den Ausgang aktiv auf Vcc (PORTx.y =1) oder GND (PORTx.y=0) zieht. Bei Input-Pins (DDRx.y=0) wählt das PORTx-Register aus, ob der interne Pull-Up-Widerstand aktiviert (PORTx.y=1) wird oder der Pin als Tri-State-Eingang fungiert. Die LEDs leuchten, sobald der Mikrocontroller den jeweiligen Pin gegen Masse zieht. Mit nur wenigen C-Befehlen entsteht dann der Grundstock eines Lauflicht-Programms:

```
PORTD |= 0x78;          // Alle aus
PORTD &= ~0x08;         // LED1 an
delay_100ms();
PORTD |= 0x08;          // LED1 aus
PORTD &= ~0x10;         // LED2 an
delay_100ms();
```

Aus dem dritten Port-Register (PINx) kann ein Mikrocontroller-Programm einlesen, welcher Wert (High/Low) an den als Eingang konfigurierten Pins anliegt. Der Zustand des Tasters 5 lässt sich prinzipiell schon mit drei Zeilen C-Code ermitteln:

```
DDRD &= ~0x40;          //Input
PORTD |= 0x40;          //Pullup an
taste5 = PIND && 0x40 > 6;
```

In einem realen Programm sollten noch ein paar Zeilen hinzukommen, um den Taster zu entprellen. Am einfachsten fragt man dazu den Pin zweimal innerhalb von kurzer Zeit ab und verwirft die Messung, wenn beide Werte nicht identisch sind. Konzepte für die weiter gehende Programmierung von Mikrocontrollern mit mehren Tasks beschreibt das c't-Projekt Elektronische Zwerge [3].

Zinn und Zange

Das Mikrocontroller-im-LAN-Projekt ist nur als Bausatz erhältlich; deshalb braucht man vor der Programmierung ein wenig Zeit, um die Schaltung mit einem feinen Lötkolben aufzubauen. Emedia vertreibt die doppelseitige Platine (8 Euro) und SEGOR-electronics einen kompletten Satz Bauteile (32,40 Euro ohne Display). Damit die Lötarbeiten auch für Anfänger möglich sind, haben wir bei der Auswahl der Bauteile komplett auf SMD-Bestückung verzichtet.

Allerdings geht es dadurch auf der Platine recht eng zu, weshalb man tunlichst die kleinen Bauteile wie Widerstände (R1 bis R39), Kondensatoren (C1 bis C6) und LEDs zuerst bestückt. Danach kommen die größeren Bauelemente an die Reihe (Spannungsregler, Quarz, Taster, Potenziometer und Piepser). Eine vollständige Stückliste findet sich auf unserer Projektseite im Web [3]. Zuletzt werden die Sockel für ICs und die Stiftleisten eingelötet. Achtung: Die weibliche Stiftleiste P1 kommt auf die Unterseite der Platine und dient der Verbindung zum A-Modul.

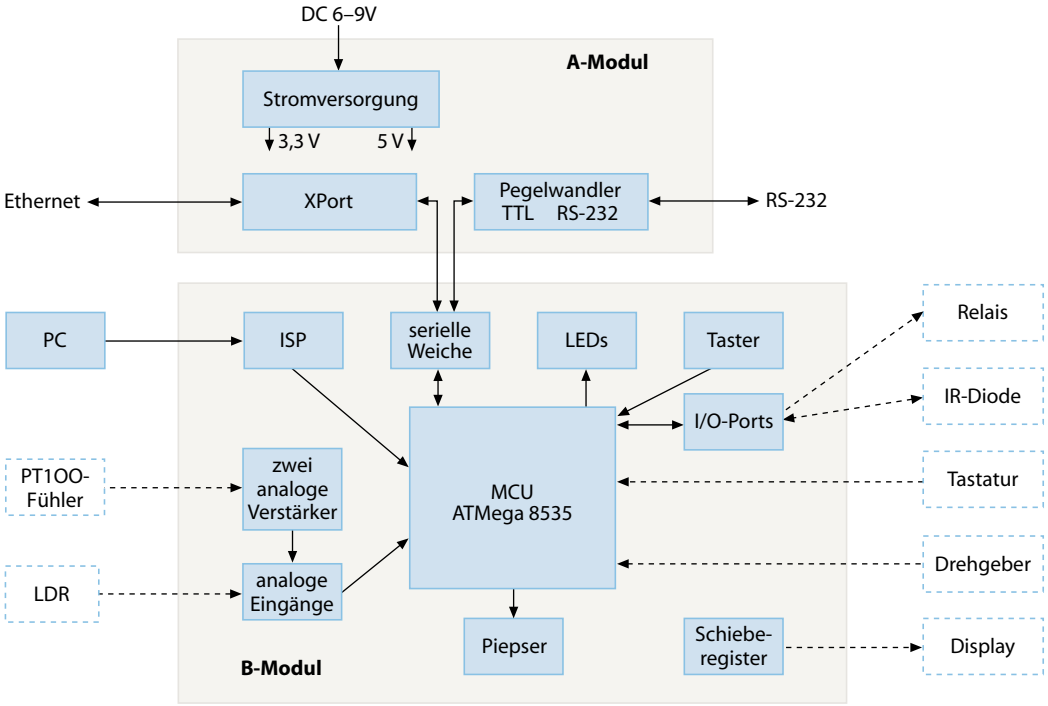
Dieses versorgt das B-Modul mit einer stabilisierten 5-Volt-Gleichspannung. Will man das B-Modul alleine betreiben, ist eine extern stabilisierte Versorgungsspannung (5 Volt DC) nötig. Der fertige Mikrocontroller-im-LAN sollte mit der Demo-Firmware von c't (beim Programmieren auf die Fuse-Bits achten!) einmal kurz piepsen und dann nacheinander alle LEDs einmal anschalten. Anschließend initialisiert er die serielle Schnittstelle und wartet dort auf Befehle.

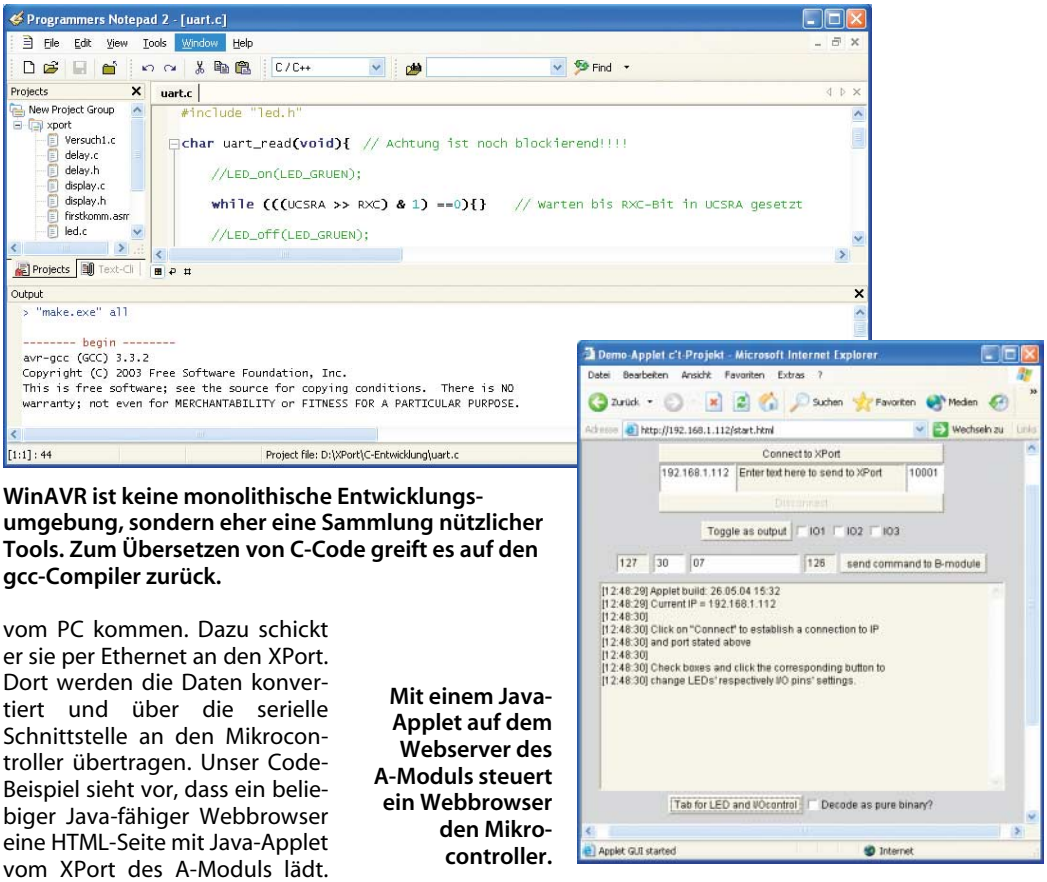
Konfiguration seriell

Steuer-leitung	µC <–> COM	XPort <–> COM	µC <–> XPort
PB0	1	egal	0
PB1	1	0	egal
PB2	egal	0	1

Kommandokette

Die Befehle für das B-Modul dürften in den meisten Fällen





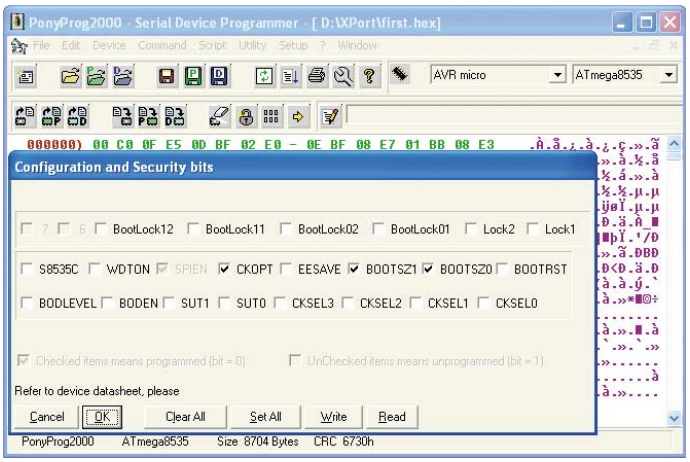
WinAVR ist keine monolithische Entwicklungsumgebung, sondern eher eine Sammlung nützlicher Tools. Zum Übersetzen von C-Code greift es auf den gcc-Compiler zurück.

Mit einem Java-Applet auf dem Webserver des A-Moduls steuert ein Webbrowser den Mikrocontroller.

vom PC kommen. Dazu schickt er sie per Ethernet an den XPort. Dort werden die Daten konvertiert und über die serielle Schnittstelle an den Mikrocontroller übertragen. Unser Code-Beispiel sieht vor, dass ein beliebiger Java-fähiger Webbrowser eine HTML-Seite mit Java-Applet vom XPort des A-Moduls lädt. Das Applet öffnet eine TCP-Verbindung zum XPort und dieser leitet die Daten an die serielle Schnittstelle des Atmel-Prozessors weiter. Zur Kommunikation zwischen Applet und MCU verwenden wir ein sehr einfaches Binärprotokoll ohne Prüfsummen oder andere Sicherungsmechanismen. Ein Byte mit festem Wert als Startcode markiert den Beginn eines Frames, der maximal 64 Byte lang sein darf und von einem 1-Byte-Stoppcode beendet wird.

Das Applet bietet eine Eingabemaske für die Frames und auf dem Mikrocontroller wartet eine Schleife auf einen gültigen Startcode. Danach liest er so lange einzelne Bytes in einen Puffer, bis der Stoppcode kommt. Ist dies nach 63 Byte nicht der Fall, verwirft er den Frame. Das zweite Byte des Frames enthält den Befehl für den Controller, in den folgenden Bytes stehen eventuell benötigte Daten. Die Firmware beherrscht bislang nur vier Kommandos zum Schalten der LEDs, zum Abfragen der Taster und Temperatursensoren und um das Lauflicht für eine bestimmte Zeit zu aktivieren.

Nach der erfolgreichen Bearbeitung gibt der XPort das gesendete Kommando wieder an das Applet zurück, wobei er für bestimmte Befehle die Rückgabewerte in den Daten-Bytes ablegt. Zur Kontrolle zeigt die Firmware den Befehl und die Antwort auf dem LC-Display an.



Beim Programmtransfer in den Controller ist es extrem wichtig, auf die korrekte Einstellung der Fuse-Bits zu achten, weil er sonst den ISP-Port zur Programmierung deaktiviert.

Ideen und Interessen

Das vorgestellte Protokoll weist noch einige Schwächen auf, und auch die Lese-Routinen auf dem Mikrocontroller und die Darstellung im Applet entbehren jeglicher Eleganz. Dafür zeigen sie, wie eine Website mit wenigen Befehlen mit dem Mikrocontroller-im-LAN kommuniziert. Über eine Socket-Verbindung kann auch jedes andere Programm Kommandos an den Controller senden. Raum für Diskussionen rund um eigene Erweiterungen bietet, wie schon für den ersten Teil, unser Leserforum [5] zu diesem Projekt.

Die Hardware bietet noch viele andere Einsatzmöglichkeiten, so könnte etwa ein Benutzer mit einem Drehgeber direkt am Gerät Schwell- und Steuerwerte verändern oder Messbereiche auswählen. Ein lichtempfindlicher Widerstand (LDR) erlaubt es, die Umgebung zu überwachen. Eine IR-Sende-/Empfangsdiode könnte Videorecorder fernsteuern oder die LIRC-Box [6] ersetzen und Befehle an einen entfernten Videoserver weiterleiten. Für Steueraufgaben stehen zahlreiche I/O-Ports bereit, die bis zu 40 mA Strom treiben können. Das reicht aus, um kleine Relais anzusteuern. Für Experimente bietet sich eine Lochrasterkarte mit den weiblichen Gegenstücken zu den Stiftleisten P5 und P8 als dritte Lage im c't-Sandwich an.

Zusatzgeräte wie Chipkartenleser oder Module für weitere analoge und digitale I/O-Ports lassen sich über einen I2C-ähnlichen Bus anbinden. Auch die RS-232-Schnittstelle eröffnet noch viele Möglichkeiten: So könnte beispielsweise der ATMega beim Überschreiten bestimmter Trigger Warnungen per SMS über ein angeschlossenes Modem versenden. Wer auf einfachstem Wege E-Mails versenden möchte, kann auch die Trigger des XPorts nutzen. Auslösen lassen sie sich über die drei I/O-Pins (CP1 bis CP3), die mit dem Atmel verbunden sind. (bbe)

Literatur

[1] Benjamin Benz, Brücken bauen, Umsetzer von Ethernet nach RS-232 im Eigenbau, c't 13/04, S. 200
[2] Dr. Till Harbaum, Blue Connection, Stereoton drahtlos per Bluetooth übertragen, Teil 2, c't 09/04 S. 208
[3] Harald Bögeholz, Elektronische Zwerge, Mikrocontroller-Praxis, Teil 4: Multitasking, c't 6/04, S. 248
[4] Webseite zum Projekt: www.heise.de/ct/ftp/projekte
[5] Leserforum zum Projekt: www.heise.de/ct/foren/go.shtml?list=1&forum_id=57426
[6] Ernst Ahlers, Jim Paris, LIRC-Box, Infrarot-Fernsteuern über das Netzwerk, c't 15/03, S. 184

Soft-Link 0414214

Bezugsquelle

Quelle	Bauteil
www.emedia.de	Platine
www.segor.de	Platine, Bauteilsatz