# Openhardware Workshop
# AVR Ethernet

Chemnitzer Linux-Tage 2010
Andreas Heik <andreas.heik@linux-tage.de>

im März 2010

Im Fokus steht die Programmierung von AVR Mikrocontrollern mit Open Source Werkzeugen.
Das Herz des für den Workshop erwerbbaren Bausatzes bildet ein Atmel ATmega32 in Verbindung mit einem Ethernet Controller. Ein softwarebasierter IP-Stack realisiert die Kommunikation der Demoplatine im Netz. Für eigene Erweiterungen sind 16 I/O-Pins herausgeführt, davon können 8 als analoge Eingänge genutzt werden. Ein OneWire-Temperatursensor ist auf der Platine integriert, der Bus für weitere Sensoren herausgeführt. Eine serielle Schnittstelle unterstützt die Softwareentwicklung und das Debugging. Die Stromversorgung erfolgt aus einer externen Quelle mit ca. 6V–, die Schaltung ist durchgängig für 3.3V konzipiert.

# Inhaltsverzeichnis

# 1 Allgemeine Hinweise

## 1.1 Funktion der Schaltung

Die Schaltung wurde mehrfach aufgebaut und getestet. Trotzdem können wir nicht sicherstellen, dass die Schaltung unter allen Bedingungen funktioniert.
Mit dem 3.3V–Design der Schaltung wird der Toleranzbereich des Mikrocontrollers minimal überschritten. Abhilfe können Maßnahmen wie Reduzierung des Takt oder Ersatz des Mikrocontrollers durch den pinkompatiblen Atmel ATmega644 schaffen.
Für Schäden, z.B. an Hardware können wir keine Haftung übernehmen!

## 1.2 Hinweise zum Löten

- Lötkolben werden bis zu 400 °C heiß, Verbrennungs- und Brandgefahr!
- Flußmitteldämpfe nicht einatmen!
- Lötzinn kann Blei enthalten, während der Arbeiten nicht Essen oder Trinken, nach den Arbeiten Hände waschen!
- Augen schützen, Spritzgefahr durch flüssiges Zinn (Vorsicht beim Auslöten)!

# 2 Bestückung der Platine

## 2.1 Bauteilliste

| | | | |
|---|---|---|---|
| C1,C2,C7,C8 | 22p | JP1 | Pinheader/Jumper |
| C3,C4 | 10n | L1 | 10$\mu$H |
| C5,C9,C10,C11, | 100n | LED1 | LED 3mm (grün) |
| C12,C13,C14,C15, | | Q1 | Quarz 25MHz |
| C18,C19,C20 | | Q2 | Quarz 11.0592MHz |
| C6,C17 | 10$\mu$ | R1,R2,R3,R4 | 49.9R |
| C16 | 220$\mu$ | R5,R6,R11,R13 | 270R |
| D1,D2 | 1N4004 | R7 | 2k2 |
| F1 | Sicherung, 0.5A | R8,R9,R10 | 10k |
| IC1 | ATMega32 | R12 | 470R |
| | + IC Sockel (40 pin) | S1 | Taster 6x6mm |
| IC2 | ENC28J60-DIL | SV1 | 20pol. Buchsenleiste |
| | + IC Sockel (28 pin) | SV2 | 5pol. Buchsenleiste |
| IC3 | MAX3232 | SV3 | Wannenstecker |
| | + IC Sockel (16 pin) | X1 | D-SUB-Buchse, 9pol. |
| IC4 | LM317 | X2 | Hohlstecker-Buchse |
| IC5 | DS1822 | X3 | Schraubklemme |
| J1 | HFJ11-2450E-L12 | | |

## 2.2 Bestückungsplan

Der Bestückungsplan befindet sich im Anhang als zusätzliches Dokument.

## 2.3 Reihenfolge der Bestückung

Beginnen Sie zunächst mit der Bestückung der kleinen Bauelemente wie Widerstände, Keramikkondensatoren, Dioden, Induktivität und der Sicherung. Zum Abwinkeln der liegenden Bauelemente liegt eine Biegelehre am Arbeitsplatz. Die Dioden sind an der Kathode mit einem Ring versehen, welcher auch im Bestückungsdruck zu erkennen ist.
Bestücken Sie als nächstes die beiden Quarze, die Elektrolytkondensatoren und die LED. Achten Sie bei den Elkos und der LED auf die richtige Polung.
Montieren Sie die 3 IC–Fassungen, beachten Sie dabei die Kerbe an der Fassung und den Aufdruck der Platine.
Im nächsten Schritt erfolgt die Bestückung von Pinheader, Buchsenleisten, Wannenstecker, Taster, Ethernet– und D-SUB-Buchse sowie die Anschlüsse für die Stromversorgung. Die Aussparung des Wannensteckers ist aufgedruckt und befindet sich auf der zum Mikrocontroller zugewandten Seite. Die Stromversorgung kann über die Hohlsteckerbuchse $X2$ oder die Schraubklemme $X3$ erfolgen. Dem Bausatz liegt ein passender Hohlstecker ($\varnothing_{innen} = 2.1mm$, $\varnothing_{aussen} = 5.5mm$) bei. Der Mantel wird mit dem Minus–Pol verbunden. Wird die Schraubklemme $X3$ nicht benötigt, bleibt der Bestückungsplatz frei. Ggf. kann hier eine Buchsenleiste aufgelötet werden, um die ungeregelte Versorgungsspannung beispielsweise zu einer Huckepack-Platine zu transportieren.
Löten Sie die aktiven Komponenten $IC4$ – Spannungsregler und bei Bedarf $IC5$ – Temperatursensor ein. Der Temperatursensor kann auch über ein Kabel an der I/O–Leiste $SV1$ angeschlossen werden.
Die ICs werden schrittweise während der Inbetriebnahme gesteckt.

> *Beachte:* Die ICs und die zugehörigen Fassungen sind mit einer Kerbe versehen. Die Gehäuseform des Temperatursensors und des Spannungsreglers sind auf der Platine aufgedruckt. Elkos sind polrichtig einzubauen, am Gehäuse ist meist der $\ominus$ Pol markiert. Die Kathode der Dioden ist mit einem Ring markiert, das LED–Gehäuse ist an der Kathode abgeflacht.

## 2.4 Inbetriebnahme

Legen Sie eine Betriebsspannung von ca. 6V an die Schaltung. Die LED sollte leuchten und am Ausgang des Spannungsregler $IC4$ sollten ca. 3.3V anliegen.
Setzen Sie jetzt $IC1$ – ATmega32 und $IC3$ – Max3232 ein. (vorher Betriebsspannung trennen) Stecken Sie den beiligenden Jumper auf Pinheader $JP1$ und legen Sie wieder die Betriebsspannung an. Jetzt sollte der Mikrocontroller über die ISP-Schnittstelle $SV3$ ansprechbar sein. (lesen der Signatur, FUSE–Bits)
Trennen Sie die Betriebsspannung und stecken Sie $IC2$ – ENC28J60. Jetzt ist es an der Zeit, sich mit der Firmware und dem Bootloader zu beschäftigen.

# 3 Schaltungsaufbau

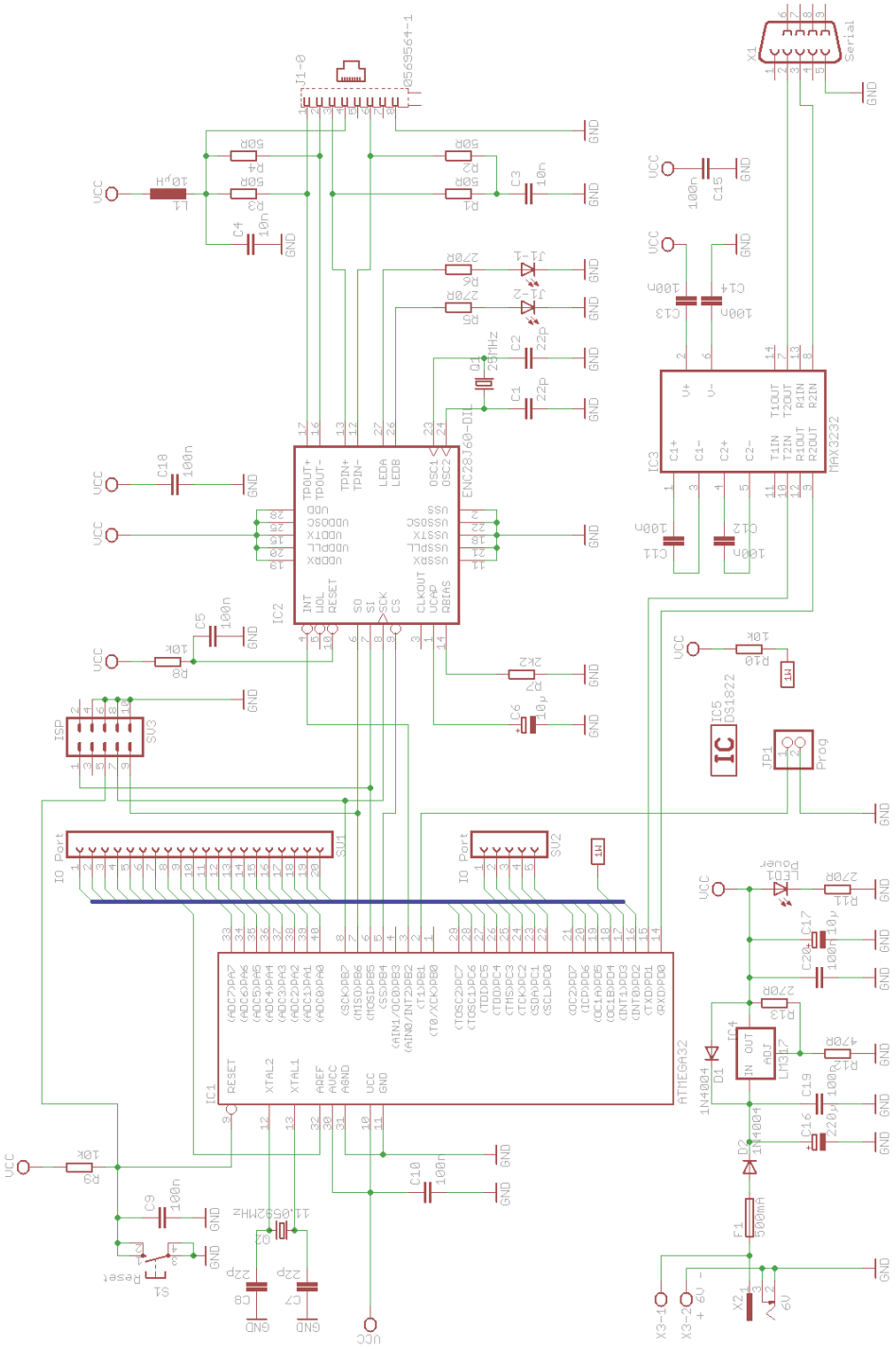


Abbildung 1: Schaltplan AVR Ethernet

## 3.1 Mikrocontroller

Zum Einsatz kommt ein AVR Mikrocontroller vom Typ ATmega32. Dieser verfügt über 32kB Flash Programmspeicher, 2kB SRAM und 1kB EEPROM für die nichtflüchtige Speicherung von Daten. Die Taktung erfolgt durch einen externen Quarz mit 11.0592MHz. Der Takt wurde so gewählt, daß die zulässige Toleranzgrenze zwischen Betriebsspannung und Takt nur minimal überschritten wird. Außerdem erlaubt die Frequenz ganzzahlige Teiler für Sekundentimer und häufig verwendete USART-Baudraten.

Für Projekte, welche die Ressourcen des ATmega32 überschreiten ist der Einsatz des pinkompatiblen ATmega644 mit 64kB Flash Programmspeicher, 4kB SRAM und 2kB EEPROM möglich.

## 3.2 ISP–Anschluss

ISP steht für In–System–Programming. D.h. der Mikrocontroller kann im unter Spannung stehendem System programmiert werden. Programmieren bedeutet hierbei das Anwendungsprogramm in den Flash–Speicher zu schreiben, FUSE–Bits zu setzen bzw. den Inhalt des EEPROM zu manipulieren.
Die Platine verfügt über einen 10–poligen Wannenstecker zur Programmierung. Die Pinbelegung folgt der Spezifikation von Atmel (auch Kanda-Standard) und kann mit üblichen Programmiergeräten benutzt werden (z.B. STK500).
Alternativ kann die Programmierung über den vorinstallierten Bootloader erfolgen. Hinweise dazu finden Sie im gleichnamigen Abschnitt.

## 3.3 serielle Schnittstelle

Der im Mikrocontroller integrierte USART-Baustein kann die Funktion einer seriellen Schittstelle übernehmen. Damit lassen sich mit wenigen Zeilen Programmcode Ausgaben, z.B. für Debugzwecke ausgeben.
Die Pegelwandlung zwischen Mikrocontroller (TTL-Pegel) und RS232-Schnittstelle z.B. eines PC übernimmt ein MAX3232.
Tabellen für die Einstellung der Baudrate in Abhängigkeit vom Prozessortakt finden Sie im Datenblatt des Mikrocontrollers.

> *B*eachte: Die Verbindung zur seriellen Schnittstelle eines PC erfolgt über ein serielles Verlängerungskabel, **kein** Null–Modem–Kabel mit gekreuzten Adernpaaren verwenden.

## 3.4 Ethernetcontroller

Die Ethernetschnittstelle wird über einen Stand–Alone Ethernetcontroller Microchip ENC28J60 realisiert.
Die Anbindung des Ethernetcontrollers an den Mikrocontroller erfolgt über das SPI-Interface. Eine zusätzliche Verbindung mit einem Interrupt–Pin des Mikrocontrollers ermöglicht die interruptgesteuerte Verarbeitung eingehender Pakete.

Die eingesetzte Ethernetbuchse verfügt über integrierte magnetische Übertrager, welche eine galvanische Trennung der Schaltung vom Ethernet realisieren. Die LEDs werden durch den Ethernetcontroller getrieben und indizieren standardmäßig die Stati *Link–Verbindung* und *Aktivität auf dem Netzwerk*.

### 3.5 Spannungsversorgung

Bedingt durch die Anforderung des ENC28J60 ist das AVR Ethernet Board auf eine Betriebsspannung von 3.3V ausgelegt.
Ein einstellbarer Spannungsregler LM317 sorgt für eine stabilisierte Betriebsspannung. Die Eingangsspannung von ca. $6V-$ wird über die Hohlstecker-Buchse $X2$ (Minus–Pol am Mantel) oder die Schraubklemmen $X3$ angelegt. Mit der Picofuse $F1$ wird die Schaltung mit 500mA abgesichert, $D2$ dient als Verpolungsschutz.
Bedingt durch die Technologie des Spannungsregler sollte die Eingangsspannung nicht zu hoch gewählt werden, da die Differenz zur Betriebsspannung multipliziert mit der Stromaufnahme in Verlustleistung umgewandelt werden. Ein Erwärmumg des Spannungsreglers ist somit normal. Der Spannungsregler verfügt über einen thermischen Überlastschutz.
Steigt durch Erweiterung der Schaltungen die Stromaufnahme, kann der Spannungsregler mit einem Aufsteckkühlkörper (für TO-220 Gehäuse) versehen werden.
An der Schraubklemme $X3$ kann die unstabilisierte Versorgungsspannung abgegriffen werden.

## 4 AVR Anwendungen (Firmware)

Das folgenden Kapitel gibt einen Überblick zu eingesetzten AVR Anwendungen auf dem AVR Ethernet Board.

### 4.1 Programmierwerkzeuge

Voraussetzung für das Erstellen von Anwendungen für AVR Mikrocontroller sind ein Cross–Compiler sowie Bibliotheken mit controllerspezifischen Funktionen.

- avr-binutils
- avr-gcc
- avr-libc

Unterschiedliche Version des `avr-gcc` erzeugen unterschiedlich große Firmware–Binaries. Dabei ist zu beachten, daß die Größe des erzeugten Hexfiles in den verfügbaren Flashspeicher des Mikrocontrollers passt. (`avr-size`)
Flashspeicher abzüglich Bootloader: $32768 - (2048 + 1) = 30719$

Für die Programmierung des Mikrocontrollers ist ein Programmiergerät und passende Programmiersoftware (`avrdude`) notwendig. Im Absatz *Bootloader* wird die Programmierung mit Hilfe eines Bootloaders über die integrierte serielle Schnittstelle diskutiert.

## 4.2 Mini Webserver

Die Firmware des Mini Webservers stammt aus dem Projekt *ETH_M32_EX* von Ulrich Radig. Anpassung beziehen sich hauptsächlich auf das Layout des AVR Ethernet Board. Die Unterstützung von OneWire–Devices (zum Beispiel Temperatursensoren Maxim DS18S20 und DS1822) wurde integriert.
Den Quellcode der Firmware finden Sie im Downloadbereich.

### 4.2.1 Programmablauf

Der Start des Mini Webservers gliedert sich in die Abschnitte *I*nitialisierung und *M*ain Loop. Innerhalb der Initialisierungsphase werden die I/O–Pins konfiguriert, die serielle Schnittstelle zur Ausgabe von Debugausschriften vorbereitet und Programmmoduln, wie *IP-Stack*, *HTTP–Daemon* gestartet.
Die Initialisierung des *IP-Stack* startet einen interruptgesteuerten Timer, welcher zur Sekundenzählung und als Watchdog benutzt wird und initialisiert das SPI–Interface zur Kommunikation mit der Ethernet Hardware.

### 4.2.2 I/O–Pins

Über die Buchsenleisten $SV1$ und $SV2$ sind die I/O–Pins $PA0..PA7$, $PC0..PC7$ und $PD2..PD7$ des Mikrocontrollers herausgeführt. In der Standardkonfiguration des Mini Webservers werden die Pins $ADC0..ADC3$ als analoge Eingänge, die Pins $PA4..PA7$ als digitale Eingänge mit aktivem Pullup–Widerstand, die Pins $PC0..PC7$ als Ausgänge und der Pin $PD7$ als OneWire–Bus konfiguriert.
Um definierte Pegel an den digitalen Eingänge festzulegen, sind interne Pullup-Widerstände konfiguriert. D.h. die Eingänge werden standardmäßig auf High–Pegel gezogen.
Die Digitalisierung (Messung) der analoge Eingänge erfolgt sequentiell in einer Endlosschleife mit einer Auflösung von 10 bit. Als Referenzspannung des Analag–Digial–Konverters ist die Betriebsspannung des Mikrocontrollers konfiguriert. Alternativ kann auch eine externe Referenzspannung an den Pin $AREF$ gelegt werden. Die externe Referenzspannung darf die Betriebsspannung des Mikrocontrollers nicht überschreiten.
Die Konfiguration der Quelle erfolgt im Register `ADMUX`.
Die Ausgänge nehmen die Pegel *low* oder *high* an und können eine LED direkt ansteuern.

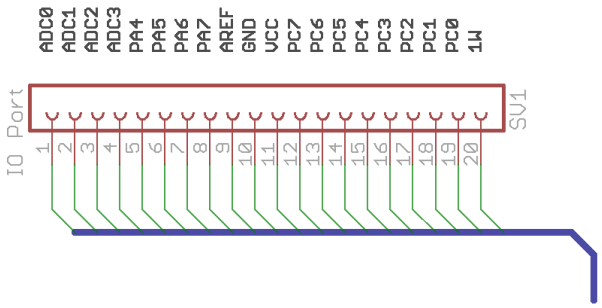


Abbildung 2: Pinbelegung I/O–Port

### 4.2.3 OneWire–Bus

Der OneWire–Bus wird am Pin $PD7$ herausgeführt. Am Bus können laut Spezifikation bis zu 100 Sensoren angeschlossen werden. Dabei erfolgt die Kommunikation zu den Sensoren über eine Datenleitung. Die Sensoren identifizieren sich durch eine eindeutige 64–bit Adresse. In der Firmware werden bis zu 4 Temperatursensoren unterstützt. Die Anzahl der Sensoren kann durch Vergrößerung der zugehörigen Datenstruktur erhöht werden. Für weitere Sensortypen müssen zusätzliche Funktionen zur Auswertung implementiert werden.
Auf der Platine befindet sich bereits ein Pullup–Widerstand für den OneWire–Bus. Dadurch können die Sensoren auch im Parasite–Power Modus (Stromversorgung über Datenleitung) betrieben werden. Dabei werden die Pins zur Stromversorgung ($V_{DD}$, $GND$) des Sensors mit $GND$, die Datenleitung $DQ$ mit dem OneWire–Bus verbunden.

### 4.2.4 serielle Schnittstelle

Der integrierte USART ist über eine Pegelanpassung (*MAX 3232*) an die D-SUB-Buchse der Platine herausgeführt. In der Mini Webserver–Firmware wird die serielle Schnittstelle für Debugausgaben und zur Konfiguration aktiviert. Für die Nutzung ist ein Terminalprogramm wie `minicom` oder `screen` erforderlich. Die serielle Schnittstelle wird mit den Einstellungen: 9600 $8N1$ initialisiert.

```
$ screen /dev/ttyUSB0
```

> *Beachte:* Der Name der seriellen Schnittstelle (*/dev/ttyUSB0*) kann je nach Gerät (USB, Steckkarte) variieren. Für das Gerät ist schreibender Zugriff erforderlich.
> Beenden von `screen` erfolgt mit der Tastenkombination [STRG]+[A] gefolgt von [SHIFT]+[K] und bestätigt mit [y].

### 4.2.5 Konfiguration

Die zentrale Konfiguration des Mini Webservers erfolgt im File `config.h`.
Die Konfiguration der I/O–Ports und Pullups erfolgt mit den Variablen $OUTA$, $OUTC$, $OUTD$ und den zugehörigen $PULLUPA$, ... als Bitmasken.
Der OneWire–Bus wird mit den Variablen $W1\_PIN$, $W1\_IN$, $W1\_OUT$ und $W1\_DDR$ konfiguriert.
Die Netzwerkkonfiguration ist statisch und erfolgt mit den Variablen $MYIP$, $NETMASK$, $ROUTER\_IP$. In der Firmeware ist eine Standardkonfiguration mit der IP-Adresse: 192.168.1.10 festgelegt. Die Netzwerkkonfiguration kann über die serielle Konsole geändert werden und wird im EEPROM des Mikrocontrollers persistent gespeichert. Verwenden Sie zum anzeigen und ändern die Kommandos `IP`, `NET` und `ROUTER`.

```
System Ready
Compiliert am Mar 10 2010 um 16:42:58
Compiliert mit GCC Version 4.4.2

NIC init:READY!
My IP: 192.168.1.10

** NTP Request gesendet! **
** NTP DATA GET! **

NTP TIME: 16:44:13
```

Die Authentifizierungsinformationen für den Webserver werden in der Variable $HTTP\_AUTH\_STRING$ festgelegt und sind auf Nutzer *root* mit Passwort *pass* voreingestellt.

- Startseite: [http://192.168.1.10/]
- Statusinformationen: [http://192.168.1.10/status.html]

## 4.3 Bootloader

Als Alternative zum Programmieren des Flash-Speichers mittels Programmiergerät kann ein Bootloader zum Einsatz kommen. Das Verfahren wird als Read-While-Write Self-Programming bezeichnet.
Der Bootloader muss über ein Programmiergerät bzw. den ISP-Anschluß in den Flashspeicher des Mikrocontrollers übertragen werden. Zusätzlich werden die FUSE–Bits des Mikrocontrollers so programmiert, dass nach Anlegen der Betriebsspannung der Bootloader–Code ausgeführt wird. Dieser erkennt über eine Bedingung (Jumper $JP1$ geschlossen) ob der Bootloader aktiviert werden soll oder (Jumper $JP1$ offen) der Programmcode ab Adresse $0x0000$ ausgeführt werden soll.

> *Beachte:* Übertragen des Programmcode über das ISP-Interface in den Flashspeicher des Mikrocontrollers überschreibt den Bootloader. Ist kein

Bootloader im Flashspeicher des Mikrocontrollers vorhanden, müssen die FUSE–Bits so gesetzt werden, daß der Programmcode ab Adresse $0x0000$ ausgeführt wird.

### 4.3.1 STK500-kompatibler Bootloader

Im Downloadbereich finden Sie den Quellcode eines STK500-kompatiblen Bootloaders. Der Code des Bootloaders belegt etwas mehr als $1kB$ des Flashspeichers. Deshalb wird bei der Programmierung der FUSE-Bits ein Speicherbereich von $2kB$ reserviert und damit die Startadresse des Bootloader auf $0x7800$ festgelegt.
Als Bedingung für das Ausführen des Bootloader–Code wird der Pin $PB1$ abgefragt.
Die Kommunikation des Bootloaders erfolgt über die serielle Schnittstelle des AVR Ethernet Boards. Die Initialisierung erfolgt mit einer Baudrate von 115200 bps. Diese Einstellung berücksichtigt die von *avrdude* unterstützten Baudraten und minimiert die Fehlerrate in Abhängigkeit von der Oszillatorfrequenz des Mikrocontrollers.
Der Bootloader unterstützt den byteweisen Zugriff auf die FUSE-Bits (lesend) und den EEPROM (lesend und schreibend).
Um den Bootloader vor Überschreiben (z.B. durch eine zu große Applikation) zu schützen, wird empfohlen den *BLB1 Mode 3* über die Lock-Bits $BLB11$ und $BLB12$ zu programmieren.

```
$ avrdude -p atmega32 -c stk500v2 -P /dev/ttyUSB0 -b 115200 -t
```

*Beachte:* Der Name der seriellen Schnittstelle (*/dev/ttyUSB0*) kann je nach Gerät (USB, Steckkarte) variieren. Für das Gerät ist schreibender Zugriff erforderlich.

# 5 Anhang

## 5.1 FAQ

#### Versteht die Firmware auf dem AVR Ethernet Board `DHCP`?

Nein, das DHCP-Protokoll ist nicht in der Firmware des AVR Ethernet Board implementiert. Eine Beispielimplementation eines DHCP–Clienten finden Sie im *uIP-Projekt*.

#### Antwortet der IP-Stack des AVR Ethernet Board auf `ping`?

Ja, das AVR Ethernet Board antwort auf `ping`.

#### Wie wird die Adresse des NTP–Servers konfiguriert?

Die aktuelle Uhrzeit kann von einem NTP–Server bezogen werden. Die Konfiguration erfolgt analog der IP–Konfiguration im File `config.h`. Änderungen sind in der seriellen Konsole mit dem Kommando `NTP` möglich und werden im EEPROM gespeichert.

Anfragen werden zyklisch an den NTP-Server gestellt (19 min Takt). Die Anzeige der Uhrzeit erfolgt auf der Statusseite: [http://192.168.1.10/status.html]

## 5.2 Links

- Downloadbereich Chemnitzer Linux–Tage 2010
  [http://chemnitzer.linux-tage.de/2010/vortraege/openhardware.html]
- GNU-Tools
  [http://www.gnu.org/software/]
- AVR-Libc
  [http://www.nongnu.org/avr-libc/]
- AVRDUDE
  [http://www.bsdhome.com/avrdude/]
- Ulrich Radig – ETH_M32_EX
  [http://www.ulrichradig.de/home/index.php/avr/eth_m32_ex]
- etherrape
  [http://www.lochraster.org/etherrape/]
- uIP TCP/IP stack
  [http://www.sics.se/ adam/uip/index.php/Main_Page]

## 5.3 Farbcode für Widerstände

| Farbe | erster Ring<br>erste Ziffer | zweiter Ring<br>zweite Ziffer | dritter Ring<br>Multiplikator | vierter Ring<br>Toleranz |
|---|---|---|---|---|
| schwarz | 0 | 0 | 1 | |
| braun | 1 | 1 | 10 | 1% |
| rot | 2 | 2 | 100 | 2% |
| orange | 3 | 3 | 1 000 | |
| gelb | 4 | 4 | 10 000 | |
| grün | 5 | 5 | 100 000 | |
| blau | 6 | 6 | 1 000 000 | |
| violett | 7 | 7 | 10 000 000 | |
| grau | 8 | 8 | 100 000 000 | |
| weiß | 9 | 9 | | |
| gold | | | 0,1 | 5 % |
| silber | | | 0,01 | 10 % |

Bei Metallschichtwiderständen bestimmen die ersten 3 Ringe den Wert, der 4. Ring den Multiplikator.
Die Farbcodierung für Induktivitäten erfolgt analog, wobei der Wert in $\mu$H angegeben wird.