Embedded Webserver

Projektarbeit

- Fachbereich Elektrotechnik -

Fachhochschule Braunschweig/Wolfenbüttel
University of Applied Sciences

Bearbeiter:

Frank Hoyer, Mat. Nr.: 20 06 45 92

Götz Issel, Mat. Nr.: 20 21 41 74

Dennis Kuhn, Mat. Nr.: 20 16 34 00

Betreuer:

Prof. Dr.-Ing. R. Bermbach

Wolfenbüttel, Wintersemester 2003/2004

eidesstattliche Erklärung

Wir versichern, dass diese Projektarbeit von uns selbstständig angefertigt und nur die angegebenen Hilfsmittel verwendet wurden.
Textstellen, die dem Sinn oder Wortlaut anderer Quellen entnommen sind, haben wir durch Angabe der Quellen kenntlich gemacht.

Wolfenbüttel, den 04.03.2004

Frank Hoyer

Götz Issel

Dennis Kuhn

Aufgabenstellung

Fachhochschule Braunschweig/Wolfenbüttel
Fachbereich Elektrotechnik

Prof. Dr.-Ing. Rainer Bermbach

Thema der Projektarbeit:

Embedded Webserver

geplanter Ausgabetermin: **September 2003**
Bearbeitungszeitraum: **Wintersemester 2003/2004**

Aufgabenbeschreibung:

Mit Hilfe einer Mikrocomputerbaugruppe mit integrierter Ethernet-Schnittstelle sowie eines TCP/IP-Stacks soll ein Embedded Webserver realisiert werden. Dazu ist als Beispielapplikation ein Messmodul (Temperatur- und Feuchtemeßgerät an den Webserver anzuschließen.

Ausgehend von einem Webserver mit integriertem TCP/IP-Stack soll das System lauffähig gemacht werden, bevor dann die Entwicklung eines eigenen Stacks (mit den notwendigen Funktionen) erfolgt. Die aufgenommenen Meßdaten des Meßmoduls (Beispielapplikation) sind über das Intranet (evtl.Internet) aufbereitet als Webseite zur Verfügung zu stellen.

Die zu erstellende Software soll flexibel gestaltet werden, so daß Anpassungen, Erweiterungen z.B. für weitere Sensoren und Meßdaten sowohl auf Seiten des Interfaces als auch auf Seiten der Darstellung leicht möglich sind.

Der Besuch der Vorlesung TCP/IP von Herrn Jongsma sollte für die Arbeit in diesem Projekt gehört werden bzw. worden sein.

Inhalt

1 Kurzfassung 6
2 Einleitung und Motivation 7
3 Grundlagen 8
3.1 Einführung in den IPC@CHIP SC12 ..... **8**
3.1.1 Die Hardwareeigenschaften des SC12 8
3.1.2 Das RTOS und dessen APIs 11
3.1.3 Die Prozessorplatine 13
3.2 Eigenschaften des Sensors SHT11 15
3.2.1 Temperatur- und Luftfeuchtigkeitsmessung, Taupunktberechnung 15
3.2.2 Funktionsweise des Sensors SHT11 17
3.3 Grundlagen der Rechnerkommunikation 18
3.3.1 Überblick: moderne Protokolle der Rechnerkommunikation 18
3.3.2 Vernetzung von Computern 19
3.3.3 Vor- und Nachteile von Netzwerken 20
4 Aufbau der Hardware und Erstellung der Software 21
4.1 Inbetriebnahme des SC12 21
4.1.1 Aufbau der Platine 21
4.1.2 Der erste Test 22
4.2 Der SC12 als Messstation 23
4.2.1 Beschaltung des Sensors 23
4.2.2 Anpassen der Prozessor-Platine an den Sensor 23
4.2.3 Implementieren der Software 24
4.2.4 Die ersten Messwerte 25
4.3 Entwicklung eines eigenen TCP/IP-Stacks 26
4.3.1 Ein neues Betriebssystem 26
4.3.2 Kommunikation zwischen zwei SC12 Mikrocontrollern 26
4.3.3 Software für einen ARP-Request 27
4.3.4 Software für eine Ping-Anfrage 28
4.3.5 Analyse der TCP/IP Funktionen der Fa. Beck 29
5 Ergebnisse 30
5.1 Einfache Realisierung eines Webservers 30
5.2 Ausgabe der Sensordaten auf einer Web-Seite 30
5.3 Gesamtergebnis 32
6 Bewertung und Ausblick 33
7 Anhang 34
7.1 Materialien (auf CD) 34
7.2 Quellenverzeichnis 35

Verzeichnis der Tabellen und Abbildungen

Tab.1: *Auflistung der SC12-Hardwareeigenschaften* 9
Tab.2: *Kurzbeschreibung der wichtigsten Signalpins* 10
Tab.3: *Die RTOS-Eigenschaften im Überblick* 11
Tab.4: *Die APIs des RTOS in Kurzform* 12
Tab.5: *Auflistung der Platinenkomponenten* 13
Tab.6: *Auflistung der Spezifikationen des SHT11 (Sensirion)* 15
Tab.7: *Ausschnitt aus den heute üblichen Kommunikationsprotokollen* 18

Abb. 1: *Der IPC@CHIP SC12* 8
Abb. 2: *Die Pinbelegung des SC12 [2]* 10
Abb. 3*: Haupt- und I/O-Speicher im Chip [2]* 10
Abb. 4*: Die gewählte Platine der Fa. Elektor im eingebauten Zustand* 14
Abb. 5*: Der SHT11 als Zeichnung mit Bemaßung [7]* 16
Abb. 6*: Der SHT11 Sensor mit Anschlussplatine und Adapterstecker* 17
Abb. 7: *Anfügen der Paketheader im Vier-Layer-Konzept [10, 12]* 20
Abb. 8: *Anfügen der Paketheader im Sieben-Layer-Konzept [10, 12]* 20
Abb. 9: *Hyperterminalbildschirm nach dem Start des SC12*.....22
Abb. 10: *Verschaltung von Controller und Sensor [7, 13]* 23
Abb. 11: *Ablaufdiagramm der Sensor-Abfrageroutine* 24
Abb.12*: Messdaten des Sensors im PC-Programm HyperTerminal* 25
Abb.13*: Der Sensor an einem weiteren SC12 (nicht von der Projektarbeit)* 25
Abb. 14*: Beispiel für ARP-Request Header* 27
Abb. 15*: ICMP Echo Request oder Reply [12]*.....28
Abb. 16*: Die Startseite des Webauftritts* 31
Abb. 17*: Webseite mit Temperaturdarstellung* 31

1 Kurzfassung

Die vorliegende Projektarbeit beschäftigt sich mit dem von der Fa. Beck produzierten Singlechip-Controller IPC@Chip SC12.

Dieser auf dem 80186 basierende Mikrorechner verfügt über eine komplette NE 2000 kompatible Ethernetschnittstelle und kann daher als Embedded Webserver verwendet werden.

Im ersten Schritt erfolgt der Aufbau und die Inbetriebnahme des SC12. Nach einigen Funktionstests wird der Prozesser mit einem Temperatur- und Luftfeuchtesensor, dem SHT11 der Fa. Sensirion verbunden, um den Rechner als Messzentrale zu verwenden. Der letzte Teil beschäftigt sich mit der Eigenentwicklung eines TCP/IP-Stacks, der den SC12 mit dem Intranet der FH verbinden soll, hierbei werden nur wenige Grundfunktionen realisiert.

2 Einleitung und Motivation

Die Firma Beck aus Wetzlar hat mit dem IPC@CHIP SC12 einen vollständigen Embedded Controller entwickelt, der auf der Basis eines 80186-Prozessorkerns aufgebaut ist. In nur einem einzigen Chip sind so Prozessor, Ein-/Ausgabeschnittstellen, Echtzeitbetriebssystem und ein kompletter Webserver realisiert.

Mit diesem Baustein, nachfolgend SC12 genannt, ist es möglich, diverse Produkte LAN- bzw. webfähig zu machen. Kapitel 3 dieser Projektarbeit befasst sich mit den Grundlagen des Mikrocontrollers und dessen Hard- und Software.

Zusammen mit dem Temperatur- und Luftfeuchtesensor SHT11 der Fa. Sensirion wird in dieser Arbeit der SC12 als Messzentrale verwendet, deren Daten man per Webseite abrufen kann. Der verwendete Sensor vereint auf kleinstem Raum sensible Messelektronik, deren Werte mit Hilfe des SC12 verarbeitet und veranschaulicht werden. Die dazu nötige Software wird an die speziellen Anforderungen des SC12 angepasst. Die Eigenschaften des SHT11 und dessen Anbindung an den SC12 beschreibt das Kapitel 3.2.

Nach der erfolgreichen Inbetriebnahme und der Funktion als Messzentrale (Kapitel 4.2) beschäftigt sich Kapitel 4.3 mit der Problematik der Vernetzung von Computern. Um in Erfahrung zu bringen, wie umfangreich die Rechnerkommunikation ist, wird mit der Programmierung eines eigenen TCP/IP-Protokolls begonnen. Die daraus resultierenden Erfahrungen und Ergebnisse fasst Kapitel 5 dieser Projektarbeit zusammen.

Der Anreiz zu dieser Arbeit besteht in der direkten praktischen Umsetzung des vermittelten Stoffes an der FH Braunschweig/Wolfenbüttel. Am Beispiel dieses kleinen Controllers kann die gelernte Theorie in die Praxis umgesetzt werden.

3 Grundlagen

Das Kapitel „Grundlagen“ beschäftigt sich mit den Eigenschaften des Mikrocontrollers SC12 und seiner Hard- und Software. Des weiteren wird der Temperatur- und Luftfeuchtesensor SHT11 vorgestellt und seine Funktionsweise erläutert.

Der letzte Abschnitt der Grundlagen befasst sich mit der Rechnerkommunikation und deren Besonderheiten. Das Kapitel schließt mit den Vor- und Nachteilen von vernetzten Computern.

3.1 Einführung in den IPC@CHIP SC12

Als Basis dieser Projektarbeit dient der Embedded Controller IPC@CHIP SC12 der Fa. Beck aus Wetzlar. Hier wurde in nur einem einzigen Baustein ein kompletter PC mit LAN bzw. Web-Anbindung realisiert. Zusammen mit der Schnittstellenplatine der Fa. Elektor ergibt der SC12 einen leistungsfähigen Controller für alle Anwendungen, die eine Ethernetschnittstelle benötigen und eine hohes Maß an Flexibilität der Hardware erfordern.

3.1.1 Die Hardwareeigenschaften des SC12

Aus der großen Zahl der verfügbaren Applikationen erschien den Autoren der SC12 als sinnvollste Variante. Der von der Fa. Beck entwickelte Einchipcomputer vereint auf einer 80186-Basis alle erforderlichen Elemente, die für einen Anschluss an ein Netzwerk nötig sind. Da die gesamten Funktionen des Chips in einem einzigen DIL-Gehäuse mit 32 Pins untergebracht sind, erfordert der Betrieb des Chips nur noch eine geringe Anzahl an externen Bauteilen. Selbst die benötigten Speicher RAM und ROM sind schon integriert. Da der Chip nicht als SMD-Variante verfügbar ist, erscheint seine Bauform als veraltet (siehe **Abb. 1**). Da auf eine räumliche Minimierung bei dieser Arbeit jedoch keinen Wert gelegt wird, spielt diese Gegebenheit keine Rolle.

Abb. 1: *Der IPC@CHIP SC12*

Der SC12 Chip hat neben der eigentlichen 80186 CPU eine Menge weiterer Ausstattungsmerkmale, die in der folgenden Tabelle verzeichnet sind

Tab.1: *Auflistung der SC12-Hardwareeigenschaften*

CPU:
- CPU 80C186
- Interner Taktgenerator mit 20 MHz Taktfrequenz
- zero wait states

Speicher:
- 512 kByte Flash Memory
- 512 kByte DRAM

Betriebssystem:
- RealTimeOperatingSystem (RTOS) mit Flash-Dateisystem
- Software download via serieller oder Ethernet-Schnittstelle

Weitere Eigenschaften:
- Intel® kompatibler multifunktionaler AD-Bus
- Ethernet 10BaseT
- TCP/IP, PPP, HTTP, FTP, Telnet, POP3, SMTP und DHCP
- sechs programmierbare Chip-Selects
- sechs interruptfähige Eingänge
- externer Interrupt Controller anschließbar
- drei Timer, Watchdog
- je zwei schnelle Timerein- und ausgänge
- zwei schnelle Serielle Schnittstellen mit TTL-RS232, incl. Baudratengenerator
- ein seriell-synchrones Interface (SSI)
- I^2C-Bus masterfähig
- 14 programmierbare E/A Pins

Spannungsversorgung:
- Betriebsspannung: 5 V ± 10%
- Stromausfallerkennung (NMI) mit Datenspeicherung

Bauform:
- DIL32-Gehäuse - 22 mm x 44 mm x 9,5 mm (Breite x Länge x Höhe)

Mit diesen Komponenten ist es den Entwicklern gelungen, einen komplett netzfähigen Embedded Controller zu entwickeln.

Besonders bemerkenswert ist dabei die Unterbringung der Chipkomponenten auf dieser relativ kleinen Fläche. Dies konnte nur ermöglicht werden, in dem einige Pins des SC12 mit bis zu vierfacher Belegung versehen sind. Per Software bzw. Unterprogrammen des Hardware-API im Betriebssystem sind diese Pins umsteuerbar. Es ist aber jeweils nur eine Funktion pro Zeiteinheit möglich.

Abb. 2 zeigt die Pinbelegung nach dem Reset. In **Tab. 2** sind kurz die Ein- und Ausgänge beschrieben. Eine ausführliche Dokumention der Pinbelegung und sonstigem Material ist im Anhang unter [1] zu finden. Ein Schema der Speichermap von Haupt- und I/O-Speicher zeigt **Abb.3**.

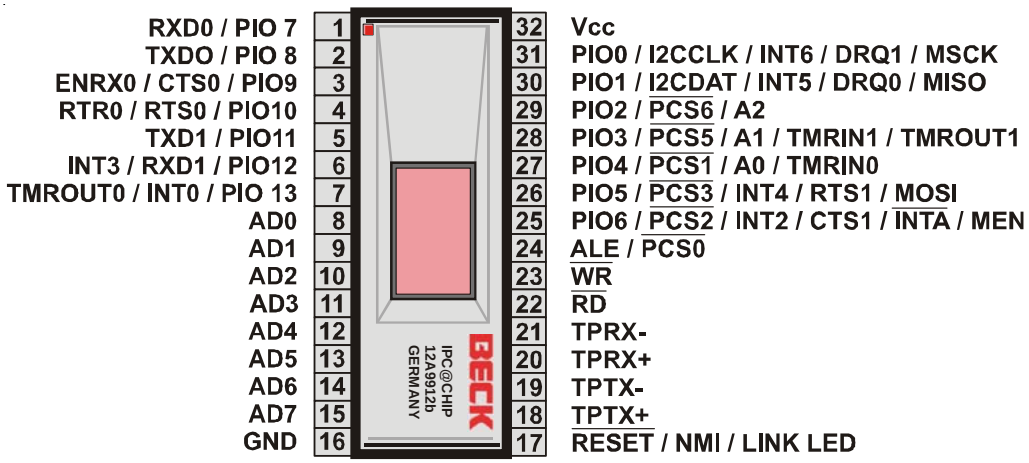


Abb. 2: *Die Pinbelegung des SC12 [2]*

Tab. 2: *Kurzbeschreibung der wichtigsten Signalpins*

PIO0..13:	Eingang/Ausgang - programmierbare I/O Torbits
/PCS0..6:	Ausgang - Chip-Selekt -Blocksignale
AD 7..0:	Eingang/Ausgang - Adress-Daten-Bus, tristate, synchron
ALE:	Ausgang - Address Latch Enable, synchron
/RD:	Ausgang - Read-Signal, synchron, nullaktiv
WR:	Ausgang - Write-Signal, synchron, nullaktiv
A2..0:	Ausgang - Adreß-Bus, tristate, synchron
I2CCLK:	Ausgang - Taktsignal des I²C-Bus
I2CDAT:	Eingang/Ausgang - Datensignal des I²C-Bus

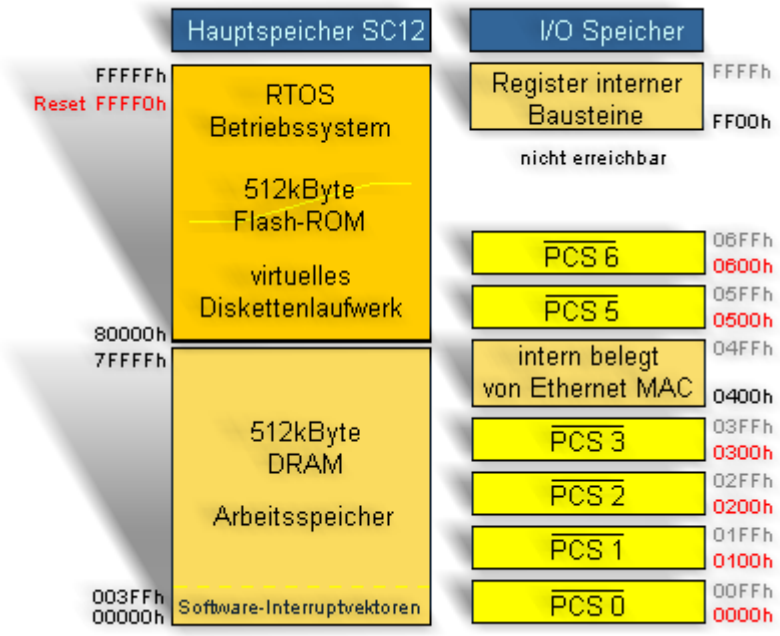


Abb. 3*: Haupt- und I/O-Speicher im Chip [2]*

3.1.2 Das RTOS und dessen APIs

Fast jeder Rechner benötigt heute ein Betriebssystem, sei es der PC zu Hause oder ein kleiner Mikrocontroller. Jeder von diesen Rechnern hat ohne diese Verwaltung keine Möglichkeit, mehrere Programme gleichzeitg laufen zu lassen. Auch der vorliegende SC12 besitzt ein solches Betriebssystem, das RTOS (Real Time Operating System). Dieses Kapitel beschäftigt sich näher mit der Spezifikation dieses Systems und seinen Fähigkeiten.

Mit dem RTOS Betriebssystem ist es dem Anwender möglich, alle gegebenen Funktionen des Chips anzusprechen. Hier die einzelnen Funktionen im Überblick:

Tab. 3: *Die RTOS-Eigenschaften im Überblick*

35 Tasks
Das Betriebssystem bietet die Möglichkeit, 12 verschiedene DOS-Programme quasi gleichzeitig auszuführen. Jedes dieser Programme läuft in einer eigenen Task des RTOS-Kerns. Die Programme können in den gängigsten Programmiersprachen geschrieben werden, es ist dabei lediglich darauf zu achten, dass der Maschinencode für den 80186 geschrieben wird (Compilereinstellung!).

15 Timer
Davon sind zwei frei programmierbar.

60 Semaphore
Diese „Zugriffstoken“ gewährleisten einen exklusiven Zugriff auf kritische Ressourcen.

10 Message Exchanges
Der Mailer besteht aus 4 Mailboxen (0...3, Maibox 0 hat die höchste Mailbox-Priorität). Die Länge der Nachrichten beträgt maximal 12 Byte (durch Zeigerübergabe auch größere Nachrichten möglich). Die Nachrichten werden in „Umschlägen“ verschickt, von denen 64 verfügbar sind. Jede Task kann allerdings eine maximale Mailboxtiefe von vier Nachrichten haben. Es können maximal 10 Nachrichten ausgetauscht werden.

2 Ereignisgruppen
Ereignisgruppen bieten eine Möglichkeit, Prozesse, die auf ein Ereignis warten, zu koordinieren.

APIs (Application Programmer Interface)
BIOS, DOS, RTOS, TCP/IP-Schnittstelle, DOS, CGI Web-Server, Hardware, I^2C-Bus, serielle Schnittstelle (Fossil)

RTOS-Dateisystem
8.3-Namenskonvention (vgl. MS-DOS), interner RAM-Speicher,interner FLASH-Speicher, externes Laufwerk (optional z.B. CF-Karte, Festplatte)

Protokolle
TCP, UDP, ARP, ICMP, Socket-Schnittstelle mit maximal 64 Sockets, drei Geräteschnittstellen: Ethernet (10BaseT), PPP-Server, PPP-Client

TCP/IP-Anwendungen:
HTTP, (T)FTP, Telnet-Server, DHCP-Client, SNMP (optional), UDP-Konfiguration für RTOS/Bootstrap-Updates

---

Dieses sind die gesamten RTOS-Ressourcen. Einige werden nur vom RTOS selbst benutzt, andere sind über die APIs dem Benutzer zugänglich.

Das RTOS ist in 6 verschiedenen Versionen verfügbar. Diese benötigen je nach Ausführung (z.B. mit oder ohne Web-Server, PPP, ...) mehr oder weniger Flash-ROM-Speicher.

Damit der Anwender mit der Controller-Hardware kommunizieren kann, sind diverse Befehle nötig, die das Zusammenspiel zwischen Hard- und Software ermöglichen. Diese Befehle sind aber sehr umfangreich und umständlich zu programmieren. Daher besitzt dieser Rechner die sog. APIs, die Application Programmer Interfaces, die diese Kommunikationsschwierigkeiten lösen. Zur Anwenderprogrammierung muss man also keine besonderen Kenntnisse der Hardware besitzen, lediglich die Informationen der APIs reichen für die Programmierung aus. Die folgende Kurzübersicht (**Tab.4**) benennt die Schnittstellen mit Namen und gibt erste Hinweise. Für eine genaue Dokumentation befindet sich im Anhang die API-Dokumentation der Fa. Beck [4]

Tab. 4: *Die APIs des RTOS in Kurzform*

Hardware-API : Ansteuerung der PINs, unterste Ebene des Betriebssystems

BIOS-API: Kontrolle aller Systemunterprogramme und der Standard-Ein-/Ausgabe

Fossil-API: Steuerung der beiden seriellen Schnittstellen (COM/EXT)

I²C-Bus-API: Kontrolliert die Benutzung dieses seriellen IC-Busses

externe Disk-API: Für den Anschluss eines externen Massenspeichers zuständig

DOS-API: dem BIOS folgende Ebene, stellt u.a. die Speicherverwaltung zur Verfügung

RTOS-API: befähigt das DOS, mehrere Programme gleichzeitig laufen zu lassen

TCP/IP-API: Hierauf basiert die Netzwerkfähigkeit des SC12, PPP-Verwaltung

Ethernet-API: Ermöglicht den Transport von Datenpaketen über das Ethernet

CGI Web-Server-API: Erstellung Treiberprogramme zur Herstellung von dynamischen http:// und ftp:// - Seiten, Netzwerkserver

---

3.1.3 Die Prozessorplatine

Bei der Platinenauswahl für die externen Baugruppen rund um den SC12 stehen mehrere Lösungen zur Verfügung. Drei von ihnen kamen in die engere Auswahl. Zum einen das Modul DK40 der Fa. Beck [3], das nur die Stromversorung und Schnittstellen, Ein- und Ausgänge auf Steck- bzw. Schraubbasis dem Anwender bereitstellt. Eine weitere Variante wird von der BBS aus Winsen [2] angeboten. Auch hier stehen Stromversorgung, Ein- und Ausgänge zur Verfügung, diese jedoch angeordnet auf einer Europakarte (Maße: 100 mm x 160 mm). Bei der Entwicklung dieser Platine wurde viel Wert auf den im SC12 integrierten I²C-Bus gelegt. Auch eine akkugepufferte Echtzeituhr ist vorhanden.

Die letzte und für dieses Projekt gewählte Variante wird von der Fa. Elektor vertrieben. Es bietet neben den oben genannten Eigenschaften der anderen Platinen auch eine Zusatzschnittstelle für eigene Hardwareanwendungen (über „gelatchte“ Ein- und Ausgänge). Diese Art der Anschlüsse ist für dieses Projekt von Vorteil, da hierdurch die Port-Pins des SC12 unempfindlich gegen unsachgemäße Handhabung geschützt sind. Zudem lässt sich über einen weitern separaten Ausgang direkt ein Dot-Matrix-Display anschliessen.
Die Vorteile dieses Boards benennt **Tab.5**:

Tab. 5: *Auflistung der Platinenkomponenten*

- vorgefertigte doppelseitige Platine (Kosten: 32 EUR)
- Ethernetschnittstelle mit Cat-5 Buchse und Übertrager FS22-101Y4 (bei Beck)
- zwei serielle Schnittstellen mit MAX233 (IC9, IC10) beschaltet
- Adresslatch (IC2, 74HCT573) zur Dekodierung des Adress/Datenbusses
- 16 gepufferte Eingänge (IC5, IC7, 74HCT541)
- 16 gepufferte Ausgänge (IC4, IC6, 74HCT574)
- GAL16V8 (IC3) als Adressdekoder
- ein LC Display (optional, hier: vierzeilig)
- I²C-Bus mit Interrupt auf 6-pol. Mini-DIN-Buchse gelegt
- geregelte Spannungsversorgung 5 V mit 7805 (IC8)

Die anderen, nicht gewählten Versionen stammen von der BSS Winsen [2] und Beck [3]. Diese wurden auf Grund der wenigen Anschlussmöglichkeiten nicht gewählt. Für eventuelle Laborversuche eignet sich das Board von Elektor am besten.

Zur Funktionsweise der Ein-/Ausgabeschnittstellen (siehe **Abb.4** und [18]):

Die Adressbits A0...A4 werden zur GAL geleitet. Zusammen mit den Chip-Select-Leitungen PCS1..5 und den Lese/Schreib-Signalen wählt die programmierte Logik den Empfänger oder den Sender der Daten aus. Diese Daten gelangen über die 16 gepufferten Ein- und Ausgänge auf den Datenbus. Ebenso wird das LC-Display angesteuert.
Alle digitalen Ein- und Ausgänge sind auf die beiden Pfostenverbinder K6 und K7 gelegt, wobei die Anschlüsse einer Portleitung jeweils gegenüber liegen.

Die gedemultiplexten Adressen A0...A7 und Daten D0...D7 liegen zusammen mit dem I²C-Bus (SDA, SLC), der Interruptquelle 0 (INT0) und dem ALE-Signal an einem weiteren Steckverbinder (K8). Dieser Anschluss wird ebenfalls von der GAL selektiert, getrennt nach Schreibvorgängen (EXT_WR) und Lesevorgängen (EXT_RD). Auch die Versorgungsspannung liegt auf diesem Verbinder. Die Anschlüsse des I²C-Busses sind zudem auf eine extra 6pol. Mini-DIN-Buchse herausgeführt.

Die „Standardanschlüsse“ der seriellen Schnittstellen - es gibt zwei - und die Ethernetschnittstelle sind auf der gegenüber liegenden Seite platziert. Die seriellen Schnittstellen sind über zwei MAX233 der Fa. MAXIM DATA[1] an der SC12 gebunden. Sie sorgen für eine Spannungsanpassung von TTL- auf 12 V-Pegel. Die Ethernetanbindung erfolgt mit einem Netzwerkfilter der Fa. Halo[1], dem FS22-101Y4. Dieser ist für die Übertragungsfrequenzen des 10BaseT-Netzwerks nach dem IEEE802.3 Standard geeignet.

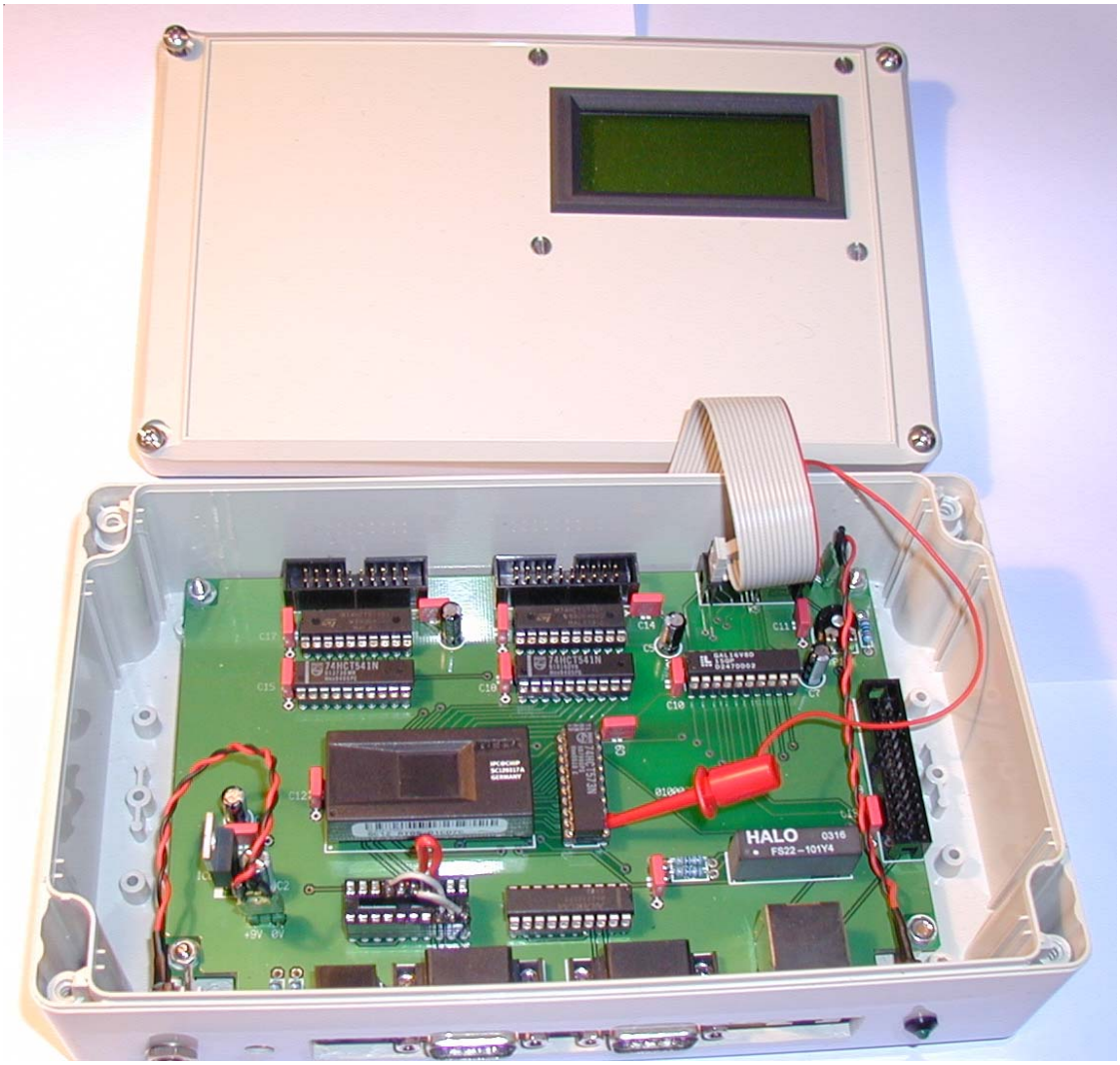


Abb. 4*: Die gewählte Platine der Fa. Elektor im eingebauten Zustand*

[1] Alle Datenblätter für die o.g. Bauteile, der komplette Bestückungsplan, sowie Materialliste, Layout und Beschreibung sind auf der CD im Anhang zu finden.

3.2 Eigenschaften des Sensors SHT11

Der SHT11 der Fa. Sensirion [7] besteht aus zwei Einzelsensoren für Temperatur und Luftfeuchte. Die beiden Sensoren sind zusammen mit einem Schnittstellentreiber auf einer Fläche von ca. 36 mm² untergebracht (**Abb. 5**). Dieser, als SMD ausgeführte Chip, lässt sich daher für Anwendungen mit sehr geringem Platzbedarf optimal nutzen. Für dieses Projekt wird der Sensor auf eine Anschlussplatine gelötet, um mit üblichen Leiterquerschnitten arbeiten zu können (**Abb. 6**). Die Sensoreigenschaften in Kurzform:

Tab. 6: *Auflistung der Spezifikationen des SHT11 (Sensirion)*

- 2 Sensoren für relative Luftfeuchte (r.F.) und Temperatur
- Präzise Taupunktberechnung möglich
- Messbereich: 0-100% r.F.
- Absolute Genauigkeit (r.F.): +/- 3.5% r.F.
- Genauigkeit Temp.sensor: +/- 0.5°C bei 25 °C
- Kalibriert und digital (2-Wire Schnittstelle)
- Schnelle Ansprechzeit (r.F.): < 4 sec.
- Niedriger Stromverbrauch (typ. 30 µW)
- preisgünstig (kostenlose Probe bei Sensirion erhältlich)

Die Fa. Sensirion hat den Autoren den Sensor SHT11 kostenfrei als Probemuster zur Verfügung gestellt.

3.2.1 Temperatur- und Luftfeuchtigkeitsmessung, Taupunktberechnung

Die Temperatur ist für uns Menschen keine sichbare Größe. Zu ihrer Bestimmung ist es nötig, einen Vergleichswert hinzu zu ziehen. Mit diesem Vergleich (z.B. Längenausdehnung) kann man über eine Skala die Temperatur direkt ablesen. Physikalisch ist sie über die mittlere Energie der Moleküle definiert [4]. Alle Materialien sind grundsätzlich zur Temperaturbestimmung geeignet. Dabei muss aber eine Reproduzierbarkeit der Messung gewährleistet sein.

Die in der Physik übliche Maßeinheit der Temperatur ist das Kelvin K. Die Kelvin-Temperatur zählt man ab dem nicht unterschreitbaren absoluten Nullpunkt, in dem die Moleküle völlig ruhen.

„...1K...ist ebenso groß wie 1 °C, das als $^{1}/_{100}$ des Abstandes zwischen dem Gefrier- und dem Siedepunkt des Wassers unter 1013 hPa Druck definiert ist. Bei diesem Druck liegt der Gefrierpunkt des Wassers bei 273,15 K, sein Siedepunkt bei 373,15 K“ [5, S. 2, S. 208ff].

Die Luftfeuchte gibt den gasförmigen Anteil des in der Luft enthaltenen Wassers an. Mit steigender Temperatur nimmt die Fähigkeit, Wasser aufzunehmen, zu. Wird die mit Wasserdampf gesättigte Luft wieder abgekühlt, scheidet sich Wasser in Tröpfchenform aus ihr ab. Die Wassertröpfchen kondensieren und schlagen sich z.B. an glatten Flächen nieder [6].

Luft hat für jede Temperatur eine bestimmte Sättigungsmenge. So kann z.B. ein Kubikmeter Luft der Temperatur von 10 °C eine Wassermenge von 9,41 g aufnehmen. Die gleiche Luftmenge nimmt aber bei 30 °C bis zu 30,38 g Wasser auf.

Die Wasseraufnahme der Luft ist von der Temperatur abhängig, nicht jedoch vom Druck. Das bedeutet, dass Druckluft nicht mehr Wasser pro Volumeneinheit aufnehmen kann als Luft unter Atmosphärendruck.

Unterscheiden muss man weiterhin zwischen absoluter und relativer Luftfeuchte. Die absolute Luftfeuchte wird in g/m³ angeben. Üblicherweise ist aber nur die relative Luftfeuche gebräuchlich, die in Prozent (%) (derzeitige absolute Luftfeuchte dividiert durch maximal mögliche Luftfeuchte bei derzeitiger Temperatur) angegeben wird. Gemessen wird die Luftfeuchte mit Hygrometern oder Psychrometern [6].

Der Taupunkt ist die Temperatur, bei der Kondensation eintritt. Bei diesem Zustand beträgt die relative Luftfeuchte 100%. Berechnet wird der Taupunkt aus relativer Luftfeuchte und der Temperatur. Dazu wird die folgende Formel benötigt, **Gl.1,** [7, 8]:

$$Taupunkt = \left[\left[\frac{0{,}66077 + 7{,}5 \cdot T}{237{,}3 + T} + \left(\lg(H) - 2\right)\right] - 0{,}66077\right] \cdot \frac{237{,}3}{0{,}66077 + 7{,}5 - \left[\frac{0{,}66077 + 7{,}5 \cdot T}{237{,}3 + T} + \left(\lg(H) - 2\right)\right]} \quad \textbf{(Gl.1)}$$

mit T = aktuelle Temperatur
und H = aktuelle rel. Luftfeuchte

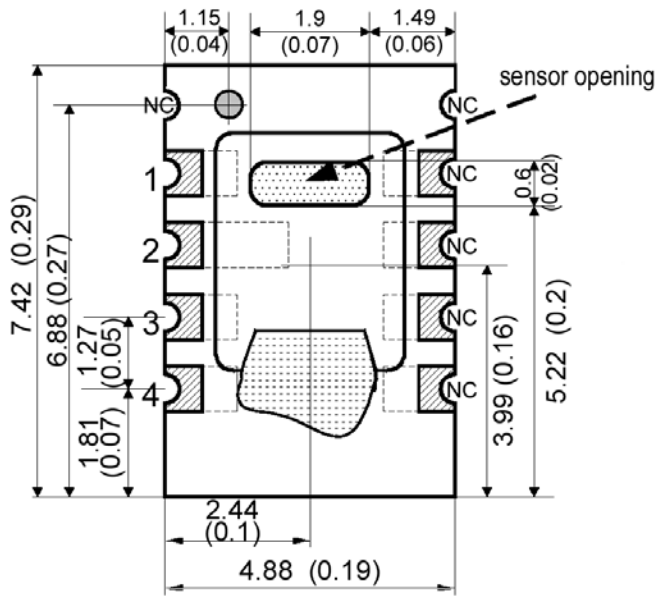


Abb. 5*: Der SHT 11 als Zeichnung mit Bemaßung [7]*

3.2.2 Funktionsweise des Sensors SHT11

Über die Funktion bis ins letzte Detail lässt sich leider nicht viel sagen, da der Hersteller nur zwei Datenblätter in englischer und deutscher Sprache zur Verfügung stellt [7, 8]. In diesen Datenblättern wird der SHT11 als „single chip“ beschrieben, der auf kleinstem Raum zwei Senoren für Temperatur und Luftfeuchte vereint. Des Weiteren verfügt der Chip über einen kalibrierten digitalen Ausgang, den man über eine Zweidrahtleitung ansprechen kann. Die Verarbeitung der Daten auf CMOS-Basis wird mittels einem patentierten Nachbearbeitungsverfahren, der CMOSens®-Technologie durchgeführt. Damit erreicht man eine hohe Zuverlässigkeit und eine sehr gute Langzeitstabilität der Sensoren.

Grundsätzlich erfolgt die Messung der Luftfeuchte über ein polymer-kapazitiv messendes Sensorelement. Hierbei wird der Sättigungsgrad der Luft mit Wasser gemessen und in die relative Luftfeuchtigkeit umgerechnet. Auf Grund ihrer Polymerbasis verfügt diese Art der Sensoren über eine lange Lebensdauer und ist zudem unempfindlich gegenüber aggressiven Gasen und Chemikalien. Der verwendete feuchtigkeitsabhängige Kondensator besteht aus einer Trägerplatte, auf der Elektroden aufgebracht sind und einer darüberliegenden Schicht aus Polymer. Diese Membran nimmt aus der zu messenden Luft Wassermoleküle auf oder gibt diese ab. Dadurch verändert sich die Kapazität des Sensors [9].

Die Temperatur wird ähnlich gemessen. Hierbei wird jedoch der Bandlücken-Effekt von Dioden ausgenutzt. Bei diesem Verfahren ändert sich in einem fast linearen Bereich die Durchlasspannung der Siliziumschicht mit der Temperatur. So ist es möglich, die Temperaturmessung auf kleinstem Raum durchzuführen.

Die gemessenen Spannungen beider Sensoren werden mit einem 14 bit A/D-Umsetzer digitalisiert. Diese Daten speichert der Sensor in zwei Statusregistern ab, die vom Prozessor mittels firmenspezifischem Datenbus gelesen werden. Weitere Informationen stehen unter [7, 8]

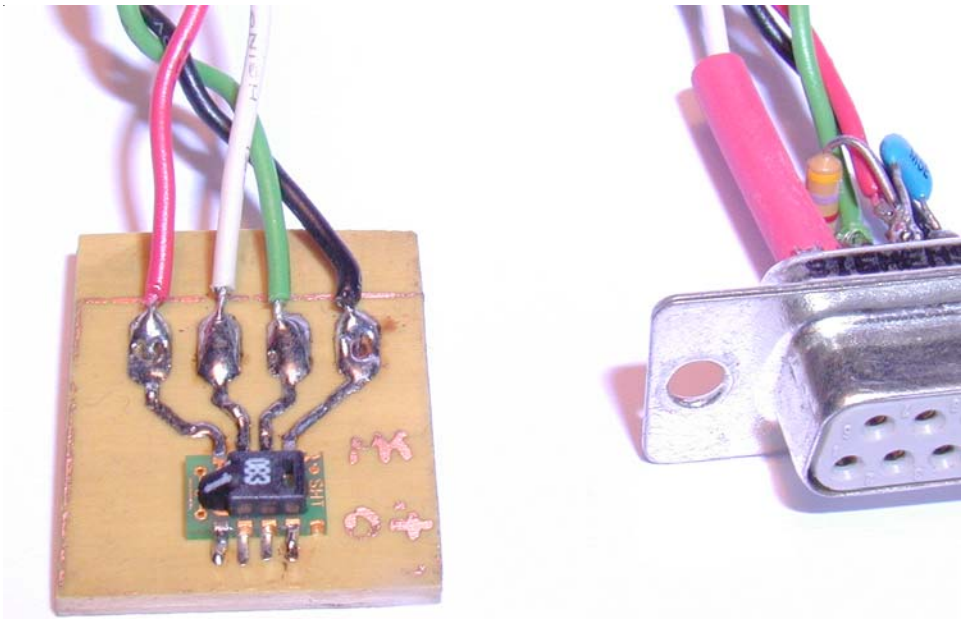


Abb. 6*: Der SHT11-Sensor mit Anschlussplatine und Adapterstecker*

3.3 Grundlagen der Rechnerkommunikation

Die Kommunikation zwischen zwei oder mehreren Computern ist aus dem alltäglichen Leben nicht mehr wegzudenken. Sei es im Haushalt, bei der Arbeit oder gar im Auto – jede moderne Elektronik hat ohne Anschluss an andere Geräte nur einen Bruchteil an Nutzwert. Ein Lesen und Schreiben der benötigten Daten erfolgt über eine oder mehrere Leitungen, meist elektrische Verbindungen. Aber auch optische Verbindungen und Übertragungen per Funk setzen sich immer mehr durch [10].

Mit der Hilfe von sog. Bussystemen ist es Rechnern möglich, Daten miteinander auszutauschen. Wichtig dabei sind in erster Linie die Geschwindigkeit, mit der die Daten transportiert werden und deren zuverlässige bzw. fehlerfreie Übertragung.

Nicht nur PCs im herkömmlichen Sinn besitzen die Möglichkeit einer Vernetzung. Jeder moderne Sensor, jede Festplatte, jedes dafür konzipierte IC arbeitet nicht nur eigenständig, sondern gibt auch anderen Steuerelementen die Fähigkeit, sie zu beeinflussen. Mit Hilfe der Rechnerkommunikation können viele ungenutzte Kapazitäten besser zugänglich gemacht und deren Wirkungsgrad erhöht werden.

3.3.1 Überblick: moderne Protokolle der Rechnerkommunikation

Wichtig bei der Datenübertragung ist das geeignete Protokoll. Ein ungeordneter Datenstrom, der aus elektrischer Sicht nur aus abwechselnden high- und low-Pegeln besteht, kann von keiner Maschine gelesen werden. Zur Ordnung und Strukturierung der Daten sind Strukturen, sog. Protokolle nötig, von denen zur Zeit eine Menge auf dem Markt gebräuchlich sind.

Tab. 7: *Ausschnitt aus den heute üblichen Kommunikationsprotokollen*

Abkürzung	Bezeichnung	Schichtname (Layer)
HTTP	Hyper Text Transfer Protocol	Application
FTP	File Transfer Protocol	Application
IMAP	Internet Mail Access Protocol	Application
POP3	Post Office Protocol	Application
TCP	Transmission Control Protocol	Transport
UDP	User Data Protocol	Transport
IP	Internet Protocol	Internet
ARP	Address Resolution Protocol	Internet
ICMP	Internet Control Message Protocol	Internet
PPP	Point-to-Point Protocol	Network
Ethernet	Ethernet Protocol	Network
MAC	Medium Access Control	*Physical*

Tab.7 fasst die wichtigsten Protokolle für die Kommunikation von Rechnern zusammen. Je nach Autor werden die Schichten in vier ([11] und **Abb. 7**) oder sieben Teile *(zus. Datalink, Session u. Presentation)* [12], geteilt. Für eine exakte Programmierung macht das „Sieben-Schichten-Modell“ mehr Sinn, da eine Implementierung der einzelnen Protokolle somit leichter durchzuführen, insgesamt aber wesentlich aufwendiger ist (**Abb. 8**).

3.3.2 Vernetzung von Computern

Die physikalische Vernetzung wird durch die MAC-Adresse (Medium Access Control) realisiert. Diese Adresse ist für jeden Netzwerkchip und somit auch für jeden PC einmalig. Die Adresse besteht aus einer sechsstelligen hexadezimal-codierten Zahlen-/Buchstabenkombination (48 bit), die zu Beginn einer Übertragung von bekannten Rechnern gesendet wird.

Die Realisierung des Physical Layer ist das Ethernet-Protokoll. Dies weitverbreitete und herstellerneutrale Medium macht es möglich, Datenpakete zwischen einzelnen Rechnern zu verschieben. Wie in **Abb. 8** zu sehen, liegt es im ersten Layer. In dem Standard IEEE 802.3 ist u.a. die Übertragungsgeschwindigkeit für das Ethernet-Protokoll definiert (10 Mbit/s bzw. 100 Mbit/s). Die Übertragung der Daten erfolgt über einen Kanal, auf den alle angeschlossenen Teilnehmer Zugriff haben. Verbunden werden die Rechner über differentielle Zweidrahtleitugen (10Base-T) oder BNC-Koaxleitung (10Base-2, veraltet) [11].

Das IP baut auf dem Ethernet-Protokoll auf. Es ver-/entpackt die übermittelten Daten ungeordnet, d.h. die Reihenfolge der Pakete ist diesem Protokoll egal. Auch eine Empfangs-/Sendebestätigung erfolgt nicht. Das IP-Protokoll garantiert auch keine verlässliche Datenübertragung von Host zu Host.

Noch eine Ebene höher liegt der Transport Layer mit dem dazugehörenden TCP oder UDP. Der Unterschied liegt in der Zuverlässigkeit der übermittelten Daten. UDP hat gegenüber TCP keine Rückmeldefunktion, ob die Daten auf der Gegenseite angekommen sind. TCP wird von Applikationen benutzt, welche einen verbindungsorientierten Übertragungsdienst benötigen, z.B. E-Mail [11].

Die letzte Ebene ist die Applikation, das Anwenderprogramm, welches die Daten schließlich verarbeitet.

Vereinfacht dargestellt läuft folgende Prozedur beim Senden (oder Empfangen) von Daten ab:

Ein Anwenderprogramm (z.B. Browser) möchte eine Seite verschicken. Dazu sendet das Programm die Daten aus dem Application Layer an den darunter liegende Transport Layer. Hier fügt das TCP (oder UDP) an die Daten einen Kopf (Header) an, in dem alle relevanten Daten (Quelle, Ziel, Länge der Daten, Prüfsumme) gespeichert sind. Das neue (und längere) Datenpaket wird an den IP-Layer weitergereicht und die gleiche Prozedur mit dem Header wiederholt sich (aber andere Headereigenschaften). Noch einmal bekommt dieses Paket einen Header, diesmal jedoch im Network-Layer, bevor das komplette Paket dann „auf die Leitung“ geschickt wird. Am Ziel erfolgt der gleiche Ebenendurchlauf, aber in umgekehrter Reihenfolge. Eine Ablauffolge ist in **Abb. 7** bzw **Abb. 8** zu sehen.

Sender							Daten		Empfänger
Application Layer				Application Head		Daten			Application Layer
Internet Layer			Session Head		Daten (SDU)				Internet Layer
Transport Layer		Transport Head		Daten (SDU)					Transport Layer
Network Layer	Network Head		Daten (SDU)						Network Layer
	Aktuelle Übertragung =>								

SDU = *Service Data Unit*: das, was in der betreffenden Schicht als Daten behandelt wird.
SDU + Header (+ Trail) = **PDU** = *Protocol Data Unit*

Abb. 7: *Anfügen der Paketheader im Vier-Layer-Konzept [10, 12]*

Sender							Daten		Empfänger
Application Layer						Application Head	Daten		Application Layer
Presentation Layer					Presentation Head	Daten (SDU)			Presentation Layer
Session Layer				Session Head	Daten (SDU)				Session Layer
Transport Layer			Transport Head	Daten (SDU)					Transport Layer
Network Layer		Network Head	Daten (SDU)						Network Layer
Data-Link Layer	Data-Link Head	Daten (SDU)					Data-Link Trail		Data-Link Layer
Physical Layer	Bits								Physical Layer
	Aktuelle Übertragung =>								

SDU = *Service Data Unit*: das, was in der betreffenden Schicht als Daten behandelt wird.
SDU + Header (+ Trail) = **PDU** = *Protocol Data Unit*

Abb. 8: *Anfügen der Paketheader im Sieben-Layer-Konzept [10, 12]*

3.3.3 Vor- und Nachteile von Netzwerken

Die Vor- und Nachteile sind mehr oder weniger Ansichtssache und hängen von dem gegebenen Umfeld ab. Sicher ist jedoch, dass ein nichtvernetzter Rechner (im sog. Stand-Alone-Betrieb) nur begrenzte Fähigkeiten besitzt. Eine Vernetzung bedeutet Kommunikation und Austausch von Daten auf sehr einfache Weise. Eine weitere Möglichkeit ist der Zusammenschluss von Rechenkapazitäten, um mit einer Vielzahl von Computern beliebig komplexe Berechnungen durchzuführen (z.B. Wettervorhersagen) [16].

Auch die Nachteile sollten nicht vernachlässigt werden. Seit der kommerziellen Nutzung des Internets tauchen immer wieder „schwarze Schafe“ auf, die das Netz zu Ungunsten anderer missbrauchen. Für mache ist das Ziel dabei, einen möglichst großen Schaden bei der Allgemeinheit anzurichten. Auch ganz neue rechtliche Probleme tauchen immer wieder in der virtuellen Welt auf.

4 Aufbau der Hardware und Erstellung der Software

Der Aufbau der Prozessorplatine und der Sensorperipherie gestaltet sich als problemlos. Nach einer kurzen Wartephase nach der Bestellung, erfolgten die Lieferungen der Firmen Elektor (Platine), Sensirion (SHT11), Beck (IPC@CHIP SC12) und Reichelt Elektronik (www.reichelt.de, restliche Bauteile) mehr oder weniger zügig.

Die Aufbau- und Testphase des SC12 ist innerhalb von wenigen Stunden erledigt. Für die Inbetriebnahme des SHT11 und die Anpassung der Software muss man etwas mehr Zeit einkalkulieren. Die meiste Zeit der Arbeit (viel auch reines (Miss-)Verständnis) muss in die Entwicklung des TCP/IP-Stacks investiert werden.

4.1 Inbetriebnahme des SC12

Für den reinen Betrieb des SC12 ist so gut wie keine Hardware erforderlich. Schon mit einer einfachen 5 V-Stromversorgung (stabilisiert) und dem seriellen bzw. Netzwerkanschluss gibt er sich betriebsbereit. Die genutzte Platine ist dennoch etwas komfortabler (siehe Kapitel 3.1.3). Die ersten Einstellungen werden mit Hilfe eines normalen PCs durchgeführt.

4.1.1 Aufbau der Platine

Die Platine der Fa. Elektor in doppelseitiger Ausführung erfordert keinen Einbau von Drahtbrücken und ist somit ohne weitere Schwierigkeiten mit den Standardbauteilen zu bestücken. Alle ICs sitzen in Fassungen, die bei einem Defekt des Bauteils den Austausch erleichtern. Der 5 V-Spannungsregler LM7805 [14] wird im Betrieb relativ heiß und benötigt daher zusätzlich noch einen Kühlkörper, der von Elektor nicht vorgesehen ist.

Die leider nicht standardisierte Form der Platine (üblich Europakartenformat, siehe [2]) macht einen Einbau in ein Gehäuse nicht einfach. Ein Gehäuse mit einer Aussparung für die Schnittstellen behebt dieses Problem. Es ist nicht nachvollziehbar, welcher Grund diese Formgebung der Platine war.

Auf weitere Gestaltungsmaßnahmen wird verzichtet.

4.1.2 Der erste Test

Nach dem problemlosen Aufbau erfolgt der erste Test. Bevor die ICs in die Fassungen gesetzt werden, wird eine Sichtkontrolle der Lötstellen durchgeführt, um Kurzschlüsse zu beseitigen. Danach erfolgt der Test der Versorgungsleitungen. Unter Spannung werden alle Fassungen an den entsprechenden Punkten auf Funktion mittels Multimeter gestestet. Nach dem positiven Ergebnis können die ICs eingesetzt werden. Dabei ist auf die Polung und Erdung zu achten, besonders der SC12 verzeiht keine grobe Behandlung.

Jetzt werden die Schnittstellenleitungen (seriell und Ethernet) angeschlossen und der Rechner unter Betriebsspannung gesetzt. Mit Hilfe des MS-Windows-Programms HyperTerminal (oder anderen Terminalprogrammen) wird die Ausgabe der seriellen Schnittstelle sichtbar. Einen ersten Eindruck zeigt **Abb. 9.**

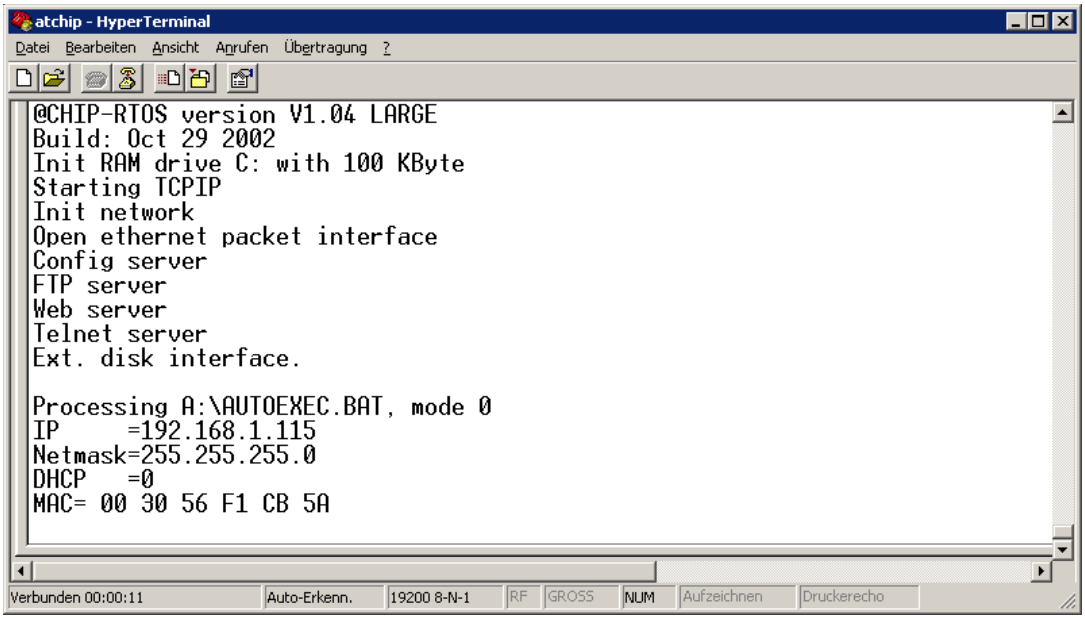


Abb. 9: *HyperTerminalbildschirm nach dem Start des SC12*

Auch der weitere Test der Ethernetschnittstelle mit dem Programm WSFTP 32 verläuft erfolgreich. Nun ist der SC12 bereit, weitere Aufgaben zu übernehmen.

4.2 Der SC12 als Messstation

Die Aufgabenstellung dieser Projektarbeit verlangt den Anschluss des SHT11-Sensors an den Mikrocontroller SC12. Damit der Sensor mit dem Prozessor kommunizieren kann, sind noch einige hardwaretechnische Maßnahmen notwendig. Auch Software muss erstellt werden, um den firmenspezifischen Datenbus des Sensors ansprechen zu können.

4.2.1 Beschaltung des Sensors

Die Beschaltung des Sensors (siehe **Abb. 10**) stellt keine großen Probleme dar, einzig der sehr miniaturisierte Aufbau fordert auch einen guten Elektroniker heraus. Die Schaltung wurde aus [7] und [13] entnommen.

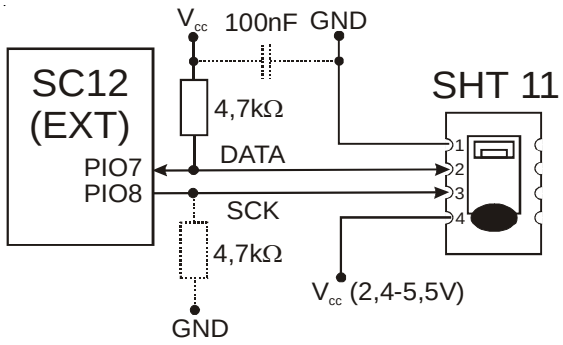


Abb 10: *Verschaltung von Controller und Sensor [7 , 13]*

4.2.2 Anpassen der Prozessor-Platine an den Sensor

Angesteuert wird der SHT11 über eine Zweidrahtschnittstelle (SCK, DATA), die (leider) nicht kompatibel zum I²C-Bus ist. Daher ist es für den Anschluss an den SC12 nötig, die Schnittstelle und ihr Protokoll selbst zu schreiben.

Ein Ansatz für diese Software ist auf der Internetseite der Fa. Sensirion verfügbar. Die Aufgabe der Autoren ist es, diese an den SC12 anzupassen, da der verfügbare C/C++ -Code für den 80C51-Prozessor verfasst ist.

Der Anschluss an die Hauptplatine des SC12 erfolgt über die zweite vorhandene serielle Schnittstelle (EXT). Dazu wird der zur Schnittstelle gehörende Baustein MAX233 herausgenommen. An den benötigten Stellen werden Brücken eingefügt, die eine Verbindung zwischen Sensor und Prozessor herstellen. Dies ist notwendig, da der SHT11 nur mit max. 5,5 V angesteuert werden kann, der Schnittstellenbaustein liefert aber 12 V. Alternativ kann für diese Art der Verdrahtung eine GAL gebrannt werden, die dann als „Kurzschlussbrücke“ (Eingang auf dazugehörigen Ausgang gelegt) benutzt wird. Das umgeschriebene Programmlisting für den SC12 steht im Anhang dieser Dokumentation.

Der Sensor kann nicht einfach an die I/O-Pins des Prozessors geführt werden. Damit definierte Zustände auf der Datenleitung liegt, wird diese mit einem Pull-Up/-Widerständen gegen V_{CC} gezogen. Als weiteres passives Bauteil kann ein 100 nF Kondensator zur Stabilisierung der Versorgungsspannung zwischen V_{CC} und Gnd geschaltet werden.

4.2.3 Implementieren der Software

Um aus dem SHT11 die gemessen Werte zu bekommen muss eine Software geschrieben werden, die der SC12 verarbeiten kann. Mit Hilfe dieses Programms ist es dann möglich, die Daten aus dem Sensor aufbereitet darzustellen.

Die Messung der Werte läuft wie folgt ab (vgl. **Abb 11**): Nach dem Start des Programms wird der Chip zurückgesetzt. Im nächsten Schritt sendet der SC12 einen Initialisierungsbefehl an den Sensor. Dieser quittiert bei positivem Empfang mit einem activ-low ACKNOWLEDGE-Bit. Danach schickt der Prozessor eine Befehlssequenz, die zum einen die Temperatur- und zum anderen die Luftfeuchtemessung auslöst. Jetzt muss der Prozessor in Wartestellung gehen, um dem Sensor genug Zeit (11-210 ms) für die Messung zu geben. Ist die Messung beendet, legt der Sensor die Leitung auf low-Pegel und der SC12 muss die Taktleitung (SCK) neu starten. Der Sensor überträgt nun zwei Byte mit Daten, die nach jedem Byte vom Prozessor bestätigt werden müssen. Ein drittes Byte beinhaltet die Checksumme. Nach dem Senden eines Bestätigungsbits fällt der Sensor in den „Sleep“-Modus. Um eine neue Messung zu starten, muss die gesamte Prozedur wiederholt werden. Nach der Messung, der erfolgreichen Übertragung der Werte und deren Korrektur werden daraus noch der Taupunkt berechnet und alle Werte auf dem Bildschirm ausgegeben.

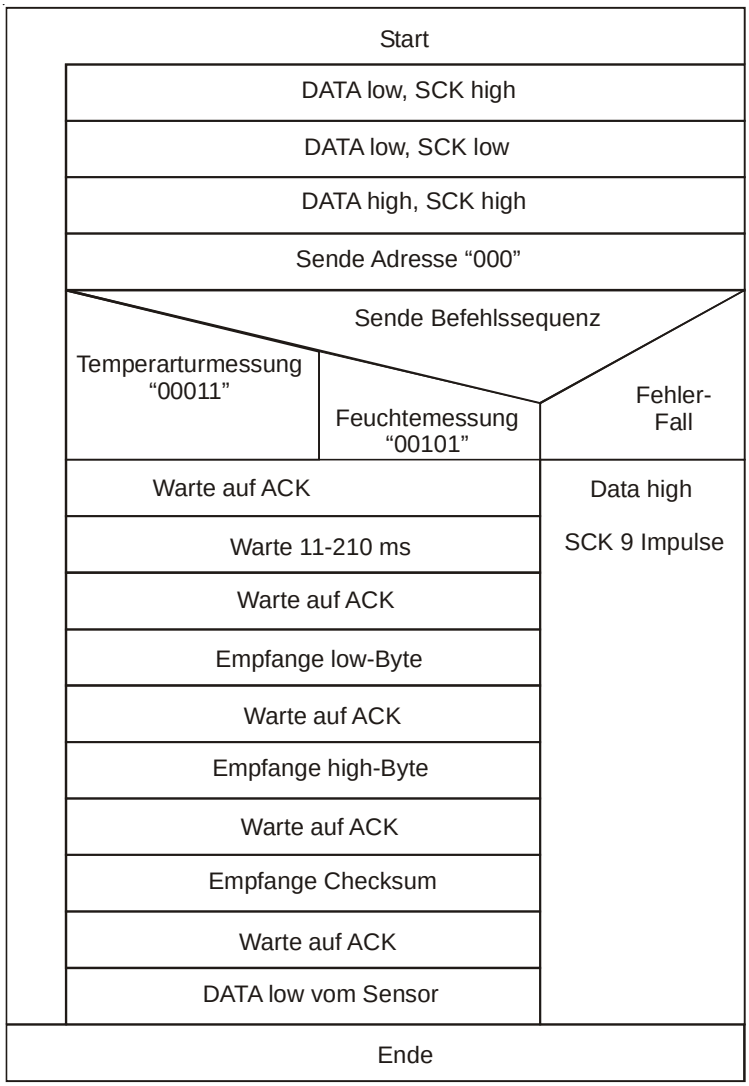


Abb 11: *Ablaufdiagramm der Sensor-Abfrageroutine*

Nach einem minimalen Abstand von 0,8 s (Vorgabe von Hersteller, damit der Chip nicht überhitzt) kann die Messung von neuem beginnen.

Die genauen Taktzeiten und Timingdiagramme können aus [7, 8] entnommen werden.

Für die Anpassung an den SC12 sind vor allem die Änderung der Takt- und Wartezeiten sowie die Änderung der Registerbefehle nötig. Die Daten der Messungen können in einem weiteren Schritt in eine Datenbank geschrieben werden. Dies wurde an dieser Stelle aber nicht realisiert, da der SC12 nur einen begrenzten Speicherplatz zur Verfügung stellt und somit nach kürzester Zeit „überliefe“.

Hauptaufgabe dieser Arbeit ist die Inbetriebnahme des SC12 und der Anschluss des Sensors. Ausser der Reihe wird ein zweiter SC12 verwendet, um die Daten auf dem dort angeschlossenen Display darzustellen, damit der SC12 auch ohne PC als Ausgabe betrieben werden kann.

4.2.4 Die ersten Messwerte

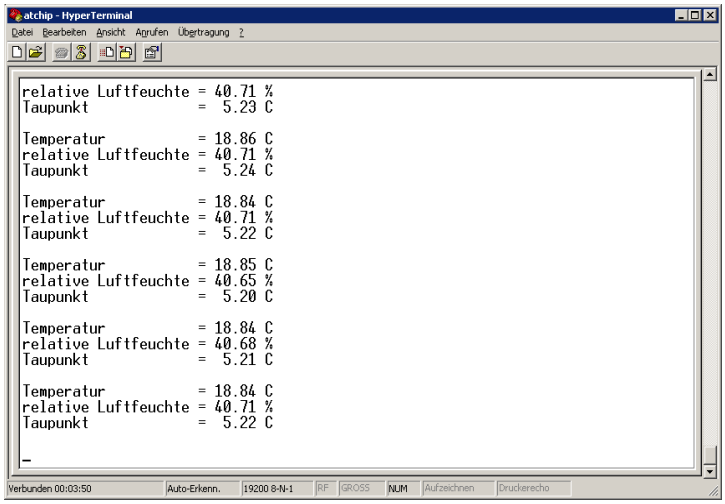


Abb. 12: *Messdaten des Sensors im PC-Programm HyperTerminal*

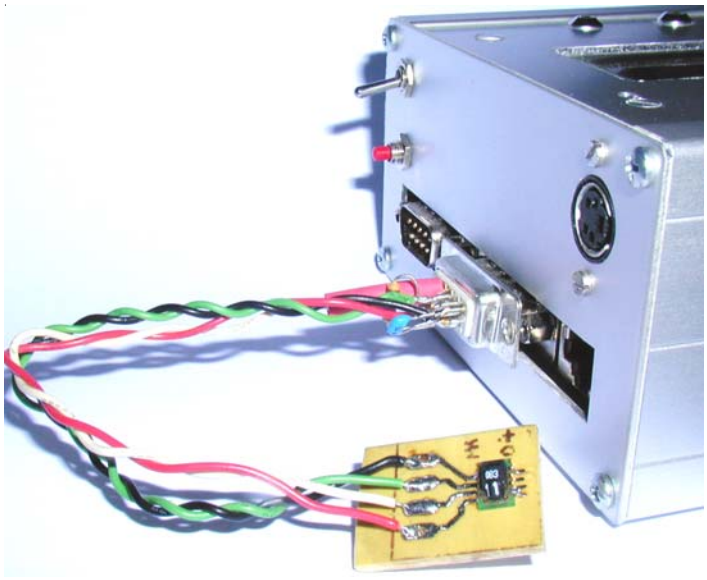


Abb. 13: *Der Sensor an einem weiteren SC12 (nicht von der Projektarbeit)*

Die ersten Messwerte des SHT11 gelangen nach einer umfangreichen Analyse des benötigten Protokolls zur Datenübermittlung und dem Anpassen der gegebenen Software (siehe Kapitel 4.2.2). Im Vergleich zum ersten Betrieb des SC12, dauert die Zeit bis zur Fertigstellung der Software doch erheblich länger. Probleme ergeben sich beim Verständnis der Datenübermittlung von Prozessor und Sensor. Mehrere Besuche des im Internet befindlichen SC12-Forums der Fa. Beck [15] bringen aber Klarheit und führen schließlich zum erfolgreichen Betrieb des Sensors und zur Darstellung der Messwerte.

Abb.12 zeigt die Messwerte des Sensors, ausgegeben über das HyperTerminal-Programm. Da die Autoren insgesamt über drei SC12 (unterschiedliche Ausführungen) verfügen, ist auf **Abb.13** ein anderer Rechner zu sehen.

4.3 Entwicklung eines eigenen TCP/IP-Stacks

Die folgenden Abschnitte beschreiben die unternommenen Arbeitsschritte, um einen eigenen TCP/IP-Stack zu entwickeln. Hierbei gibt es einige Realisierungsprobleme, die auch bis zum Abschluss der Arbeit nicht überwunden werden können. Daher folgen in dieser Arbeit nur die Beschreibung der Versuche, schrittweise Teilfunktionen zu realisieren. Grundfunktionen und Eigenschaften des TCP/IP-Stacks der Fa. Beck sind anhand des Quellcodes kurz erklärt.

4.3.1 Ein neues Betriebssystem

Die Firma Beck stellt neben dem vorinstallierten „Large“ RTOS noch weitere fünf Betriebssysteme auf ihrer Web-Seite zur Verfügung. Diese dienen dazu, den SC12 auch für andere Anwendungen zu nutzen, die keine Webanbindung, aber grösseren Speicherplatz im RAM benötigen.

Für diesen Teil der Projektarbeit ist das Aufspielen eines dieser Betriebssystemvarianten erforderlich. Die Autoren haben sich für das Modell „Tiny“ entschieden, da dieses eine eingeschränkte Netzwerkfähigkeit besitzt und eine Eigenentwicklung des TCP/IP-Stacks ermöglicht. Mehr über die vorhandenen Betriebssystemvarianten ist unter [17] zu finden.

Das neue Betriebssystem wird von der Internetseite der Fa. Beck heruntergeladen. Zum Aufspielen auf den SC12 muss dieser mit einem PC verbunden sein (seriell oder Ethernet). Die Software wird entpackt und per upload-Befehl an die gewählte Schnittstelle gesendet. Dabei darf die Stromzufuhr zum SC12 auf gar keinen Fall unterbrochen werden, da sonst der Flash-Speicher fehlerhaft beschrieben und dadurch der Chip irreparabel beschädigt wird.

Vollzieht man den Transfer per Ethernet, muss man sich darüber im Klaren sein, dass der SC12 nach dem Reboot nur noch über die serielle Schnittstelle angesprochen werden kann. Ein einfacher Datentransfer ist in diesem Modus daher nicht möglich. Mit Hilfe eines Terminalprogramms muss eine serielle Verbindung von PC zum SC12 hergestellt werden, über die man dann eine Datei auf dem Mikrocontroller generieren kann. Zum Senden einer Datei kann man die Sendefunktion des Terminal-Programmes nutzen, wobei das Protokoll „X-Modem“ genutzt wird. Auf der Kommandozeilenebene muss vorher beim SC12 der Befehl „xtrans com r filename.ext“ eingegeben werden. Im letzten Schritt der Dateiübertragung wird die zu sendende Datei in Blöcken auf den SC12 übertragen.

4.3.2 Kommunikation zwischen zwei SC12 Mikrocontrollern

Da in der Betriebssystemumgebung keine TCP/IP-Unterstützung integriert ist, es jedoch von der Herstellerfirma des SC12 ein Programmbeispiel gibt, welches eine einfache Kommunikation über die Ethernetschnittstelle demonstriert, bietet sich dieses als Grundlage für die weitere Entwicklung an. Die übermittelten Pakete enthalten nur einen Ethernet-Kopf, also eine Sender- und eine Empfänger-Ethernet-Adresse und eine Bezeichnung des Pakettyps. Im Datenteil des Gesamtpaketes befindet sich nur eine Zählvariable, die eine eindeutige Unterscheidung der Pakete ermöglicht. Die Paketgröße ist hier immer 1500 Zeichen lang.

Das Programm, welches beim Sender ausgeführt wird, erzeugt neue Pakete der oben beschriebenen Art und erwartet vom Empfänger jedes Mal eine Bestätigung. Im Terminal wird hier angezeigt, wie viele Pakete verschickt, bestätigt oder fehlerhaft waren. Auf der Empfängerseite wird für jedes korrekte Paket eine Bestätigung verschickt und auf dem Terminal angezeigt.

Mit einem gekreuzten Patchkabel (Crosslink) funktioniert die Verbindung und die Paketversendung und -bestätigung verläuft problemlos. In einem zweiten Versuch, bei dem die Rechner über einen Router verbunden sind, kommt die Verbindung nicht zustande, da die Router die Pakete nicht weiterleiten.

Auf Grund dieser Erkenntnisse basieren die nächsten Schritte zum Erstellen der eigenen TCP/IP Software.

4.3.3 Software für einen ARP-Request

Wird eine Kommunikation zwischen zwei Rechnern aufgebaut, so ist die Kenntnis der Empfänger-Internetadresse (IP-Adresse) bekannt, jedoch die dazugehörige Ethernet-Adresse (MAC) muss erst erfragt werden. Über eine so genannte ARP-Request werden alle im Netz verfügbaren Geräte gefragt, ob sie die gesuchte IP-Adresse belegen. Befindet sich der gesuchte Rechner im Netz, antwortet er mit einem so genannten ARP-Reply und gibt seine MAC-Adresse bekannt. Erst nach diesem Vorgang kann eine direkte TCP/IP-Kommunikation auf höheren Eben stattfinden.

Ein von den Autoren erstelltes Programm fragt testweise eine MAC-Adresse ab. Die Struktur des Paketes ist fest vorgegeben und hat folgende Form (**Abb.14**):

```
00000:  FF FF FF FF FF FF 00 01 02 B7 88 44 08 06 00 01    ÿÿÿÿÿÿ....^D....
00010:  08 00 06 04 00 01 00 01 02 B7 88 44 C0 A8 00 C8    ..........^DÀ¨.È
00020:  00 00 00 00 00 00 C0 A8 00 1E                      ......À¨..

ARP_RARP: Hardware Type = Ethernet (10Mb)
ARP_RARP: Protocol Type = 2048 (0x800)
ARP_RARP: Hardware Address Length = 6 (0x6)
ARP_RARP: Protocol Address Length = 4 (0x4)
ARP_RARP: Opcode = Request
ARP_RARP: Sender´s Hardware Address = 000102B78844
ARP_RARP: Sender´s Protocol Address = 192.168.0.200
ARP_RARP: Target´s Hardware Address = 000000000000
ARP_RARP: Target´s Protocol Address = 192.168.0.30
```

Abb. 14: *Beispiel für ARP-Request Header [11, S.9]*

Die Idee dieses ersten Tests ist, das oben angegebene Paket zu senden und den Empfang mittels eines Paket-Sniffers nachzuweisen. Die Rückmeldung bzw. der Empfang der angefragten Daten ist in diesem Teil noch nicht implementiert. Beim Betrieb der Testsoftware lässt sich Aktivität auf der Netzwerkleitung durch die Status-LED erkennen. Leider zeigt die Sniffer-Software kein ARP-Paket an. Mögliche Gründe hierfür sind:

- die Paketstruktur ist schon bei der Erstellung im Programm inkorrekt und der Empfänger verwirft es als defektes Paket
- das Paket wird durch die Hardware, also den Ethernetcontroller, verändert
- ggf. fehlende Initialisierungsnachrichten zwischen Sender und Empfänger
- Limitierungen der Sniffer-Software zum Erkennen einzelner ARP-Pakete

Um die Begrenzungen der Sniffer-Software auszuschließen wird im nächsten Versuch eine Ping-Anfrage programmiert.

4.3.4 Software für eine Ping-Abfrage

Um ein korrektes Ping-Paket zu erhalten, werden die Sniffer-Programme [19] genutzt und eine Kopie eines solchen Paketes in das Ping-Programm eingebaut. Das vorher aufgezeichnete Paket wird also genau übernommen und im anschließenden Betrieb ausgesendet. Die generelle Struktur eines solchen Paketes zeigt **Abb. 15**:

Type (8 or 0)	**Code (0)**	**Checksum**
Identifier		**Sequence Number**
Optional Data		
...		

Abb. 15*: ICMP Echo Request oder Reply [12]*

Für das Testprogramm sind entsprechend der Netzwerkumgebung die passenden Werte eingesetzt. Der Versuch verläuft ebenfalls erfolglos, denn die Sniffer-Software erkennt die Ping-Anfrage abermals nicht und es wird auch keine Antwort versendet. Die Gründe für diesen Fehlerversuch sind vermutlich:

- die Paketstruktur ist schon bei der Erstellung im Programm inkorrekt und wird vom Empfänger als defektes Paket verworfen
- das Paket wird durch die Hardware, also den Ethernetcontroller, verändert

Weitere Versuche, die Kommunikationssoftware zu erstellen haben die Autoren aus zeitlichen Gründen eingestellt.

4.3.5 Analyse der TCP/IP-Funktionen der Fa. Beck

Im folgenden Abschnitt wird grob beschrieben, welche Funktionen des Betriebssystems genutzt werden, wenn

- der SC12 sich im Netzwerk anmeldet
- bei Webserver-Betrieb eine Internetseite angefordert wird

Die im folgenden erwähnten Funktionen sind im Anhang unter [4], im Kapitel „TCP/IP API: Interface definition for the TCP/IP sockets“, zu finden.

Der SC12 kann über die Funktion „DHCP_USE“ so konfiguriert werden, dass die Netzwerkparameter automatisch von einem DHCP-Server angefragt werden. Ist zudem auch eine physikalische Verbindung zu einem anderen Netzwerkgerät vorhanden, so beginnt die Eigenkonfiguration. Über den UDP-Port 68 fordert der DHCP-Client, hier der SC12, Konfigurationsinformationen an. Hierzu wird mit der Funktion „API_BIND“ der Port einem Socket zugeordnet, über welchen die UDP-Nachricht gesendet werden soll. Generell müssen Sockets mit der Funktion „API_OPENSOCKET“ erstellt und durch „API_CLOSESOCKET“ wieder geschlossen werden.

Im Anschluss wird das Datenpaket für die DHCP-Anfrage mit dem Aufruf der Funktion „API_SENDTO“ abgesendet. Die Anfrage wird nicht an einen bestimmten Server gestellt, sondern wird als Broadcast verschickt. Die Antwort wird auf Port 67 erwartet. Deswegen wird mit den Funktionen „API_BIND“ und mit „API_RECVFROM“ ein Socket zugeordnet und auf ein ankommendes Paket mit den geforderten Daten gewartet. Der Status der Konfiguration wird über die Funktion „DHCP_STAT“ überprüft um festzustellen, ob die Daten noch erwartet werden oder schon korrekt empfangen und verarbeitet wurden.

Der Server, oft in einem Router integriert, antwortet, indem er dem Client eine IP-Adresse zuweist und ihm weitere wichtige Informationen über das Netzwerk, wie Netzmaske, Gateway und DNS-Server übermittelt.

Da die Anfrage sowie die Antwort als UDP-Nachricht versendet werden, also verbindungslos, lässt sich dieser Ablauf mit geringem Aufwand realisieren.

Ist der SC12 im normalen Webserver-Betrieb, so werden die Abläufe komplexer, da auch eine Verbindungsverwaltung realisiert werden muss. Die übliche Anfrage nach einer Internetseite geschieht über den Port 80. Auf dieser Tatsache basierend überwacht der SC12 diesen Port mit Hilfe der Funktion „API_LISTEN“. Trifft eine Verbindungsanfrage am Mini-Webserver ein, so wird von „API_ACCEPT“ eine neue Socket- und Port-Kombination definiert, über welche diese TCP-Verbindung abgewickelt wird. Jetzt können hierüber die gewünschten Datenpakete mit den Funktionen „API_RECV“ und „API_SEND“ ausgetauscht werden. Sobald über diese verbindungsorientierte Kommunikation die geforderten Daten übertragen wurden, wird die Verbindung mit „API_RESETCONNECTION“ getrennt.

Die oben beschriebenen Abläufe sind stark vereinfacht dargestellt. Betrachtet man zum Beispiel das Versenden einer TCP-Nachricht, so beinhaltet dies unter anderem noch Prüfsummenberechnung, Paketverwaltung, Neuübertragungen bei fehlenden Paketquittungen und Verbindungsüberwachung.

5 Ergebnisse

In diesem Kapitel sind die Ergebnisse der Bearbeitung dieser Projektarbeit zu finden. Der Betrieb des SC12 als Webserver, die Messung von Temperatur und Luftfeuchte sowie der Beginn der Stack-Programmierung sind die drei Hauptteile dieser Arbeit.

5.1 Einfache Realisierung eines Webservers

Der Aufbau und die Inbetriebnahme des als Webserver laufenden IPC@CHIP SC12 verläuft sehr einfach. Mit der kompakten Gestaltung der Hardware ist es in kürzester Zeit möglich, ein vorhandenes System mit Hilfe des SC12 netzwerktauglich zu machen und dadurch die Kontrolle per Intra-/Internet zu erlangen. Auch der scheinbar hohe Kostenaufwand für diesen hochintegrierten Chip ist durchaus angemessen.

Die Geschwindigkeit des Prozessors reicht für die Aufgaben im Embedded Bereich aus, will man jedoch höhere Geschwindigkeiten, müssen andere Lösungen in Erwägung gezogen werden. Für die Messung der SHT11 Sensordaten reicht der Chip völlig aus, ist vielleicht sogar etwas überdimensioniert. Als multifunktionaler Baustein mit Webanbindung ist er aber für viele Aufgaben sehr gut geeignet.

5.2 Ausgabe der Sensordaten auf einer Web-Seite

Die Projektaufgabe besteht darin, die gemessenen Werte auf einer Webseite darzustellen. Dazu wird die Ethernetfähigkeit des SC12 genutzt, der bei dieser Anwendung als Webserver fungiert. Hier ist auch die Stärke des SC12 zu erkennen. Mit einer solch kompakten Bauform (vernachlässigt man die Baugröße der „Experimentierplatine“) von Prozessor und Sensor(en) ist es möglich, Klimadaten an entfernter Stelle zu messen und diese über das Intranet/Internet auf einem entfernten Rechner darzustellen. Eine zusätzliche Speichermöglichkeit (Festplatte, RAM-Speicher) macht auch eine Auswertung der gemessenen Daten möglich, um daraus Verläufe, auftretende Änderungen, etc. zu berechnen und darzustellen.

Das eigentliche Programm zur Verarbeitung der Sensormesswerte wird beim Systemstart ausgeführt. Zum einen kommuniziert es mit dem Sensor und zum anderen stellt es die Webseite bereit. Die Besonderheit ist die dynamische Seitengenerierung bei jedem Messintervall. Übliche HTML-Seiten liegen statisch in einer festen Verzeichnisstruktur als Datei bereit. Der hier eingesetzte Webserver verfügt über eine CGI-Funktion, die eine dynamische Erstellung der HTML-Seite ermöglicht. Das Messprogramm „sht11ex.exe“ generiert im Speicher des SC12 eine virtuelle Internet-Seite, die im vorliegenden Fall den Namen „sht11“ trägt. Ruft der Browser diesen Namen (http://141.41.39.149/sht11) auf, so verweist er durch das CGI auf einen Speicherbereich, in dem immer eine Seite mit den aktuellen Messwerten vorliegt.

Der Funktionstest und die Inbetriebnahme konnten ohne größere Probleme durchgeührt werden. Um einen ansprechenden Ausgabestil auf der HTML-Seite, dem Terminal und dem LC-Display zu erstellen mussten umfangreiche Anpassungen vorgenommen werden. Das Ergebnis der HTML-Seiten ist in den **Abb. 16** und **Abb. 17** zu sehen.

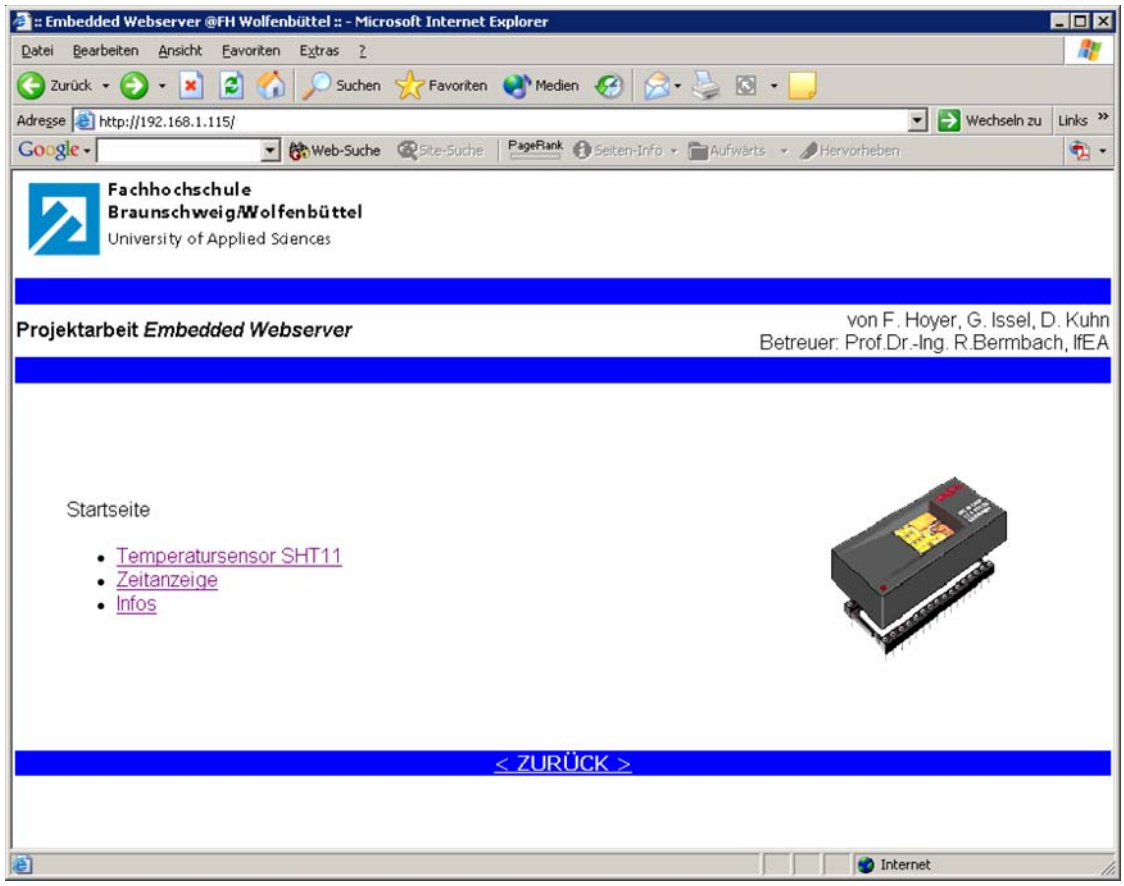


Abb. 16: *Die Startseite des Web-Auftritts*

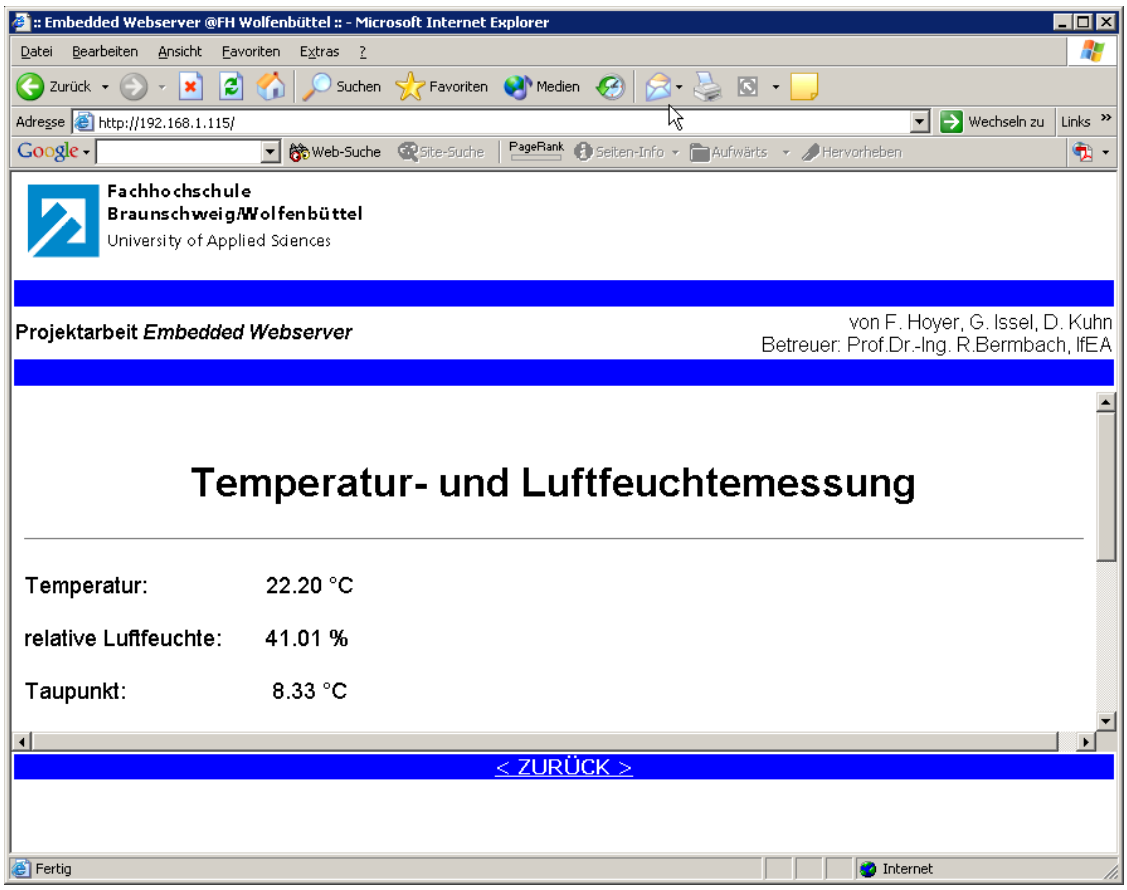


Abb. 17: *Webseite mit Temperaturdarstellung*

5.3 Gesamtergebnis

Im ersten Schritt wurde der IPC@CHIP SC12, Fa. Beck, auf einem Entwicklungsboard der Fa. Elektor in Betrieb genommen. Sowohl die Anbindung des Temperatur- und Luftfeuchtesensors SHT11 (Fa. Sensirion) als auch die Darstellung der gemessen Sensorwerte auf einer dynamischen Webseite verliefen erfolgreich. Zusätzlich konnten die Messwerte auf einem LC-Display oder auch zeitgleich auf einem Terminalfenster ausgegeben werden.

Die Auswahl des Prozessors fiel auf den SC12, da dieser ohne größeren Hard- und Softwareaufwand in Betrieb genommen werden konnte und auch bei vielen an deren Anwendungen erfolgreich im Einsatz ist.

Das Entwicklungboard ist besonders für den Laboreinsatz geeignet, da die Ein- und Ausgänge des Mikrocontrollers durch Latches geschützt sind. Auch der Anschluss des LC-Displays ist vom Hersteller vorbereitet.

Für den Anschluss des Temperatur- und Luftfeuchtesensors musste das Entwicklungsboard nur geringfügig angepasst werden. Um direkten Zugriff auf den seriellen Port des SC12 zu erhalten, wurde einer der gesockelten MAX323-Bausteine entnommen und die Fassung überbrückt. Der SHT11 liefert die Messdaten über eine serielle Zweidrahtverbindung, in einem firmenspezifischen Protokoll, welches dem des I²C-Bus ähnelt. Die dafür nötige Software wird von Fa. Beck kostenlos zur Verfügung gestellt, die Autoren haben diese für aussenstehende Anwender aufbereitet und dokumentiert. In diesem Code werden neben der Abfrage und Auswertung der Sensordaten auch Anfragen über den Webserver bedient. Bei jeder Anfrage generiert die Software über die CGI-Schnittstelle eine dynamische HTML-Seite, die die aktuellen Messwerte enthält. Nebenbei übergibt das Programm die Messdaten an das LC-Display und das Terminal-Fenster.

Die geforderten Anpassungsmöglichkeiten können ohne weitere Probleme in der Software realisiert werden, ein Anschluss von mehreren SHT11-Sensoren ist zum Zeitpunkt des Projektendes nicht möglich, da der Hersteller die Adressierung in der Hardware noch nicht implementiert hat. Eine Anpassung der Software gestaltet sich umfangreicher, will man andere Sensorentypen (z.B. LM 75, www.national.com) oder andere Kommunikationsprotokolle nutzen (z.B. I²C-Bus Anbindung).

Die Software zur Darstellung von Zeichen oder Zeichenketten auf dem LCD ermöglicht eine definierbare Ausgabeposition. Dadurch ist das Ansprechen des Displays erheblich vereinfacht und der Programmieraufwand für andere Applikationen ist sehr gering.

Der zweite Abschnitt beschreibt die Versuche einen TCP/IP Stack in Eigenentwicklung schrittweise zu erstellen. Die theoretischen Grundlagen konnten auf Grund ungeklärter Funktionen der Hardware nicht in die Praxis umgesetzt werden. Die Autoren haben die Abläufe der Kommunikation im Ethernet anhand vorhandener Quellcodes des TCP/IP Stack der Fa. Beck analysiert. Die Analyse sollte Hinweise auf Besonderheiten des integrierten Netzwerkcontrollers geben und mögliche Fehlerquellen aufdecken. Durch die Begrenzung der Projektdauer konnte kein funktionsfähiger TCP/IP Stack programmiert werden.

6 Bewertung und Ausblick

Das Projekt hat viele Einblicke in die praktische Nutzung von Mikrocontrollern gegeben. Die Umsetzung des gelernten theoretischen Hintergrundwissens aus Vorlesung und Selbststudium konnte für diese Arbeit genutzt werden. Im Verlauf des Projekts wurde deutlich, dass die vorhandenen Grundlagen die Komplexität dieser Arbeit nicht abdecken können. Das Studium der erweiterten Funktionen des Mikrocontrollers erforderten einen erheblichen Zeitaufwand.

Die erfolgreiche Inbetriebnahme der Mikrocontrollereinheit und die ersten Funktionstests konnten ohne großen Zeit- und Arbeitsaufwand bewältigt werden. Motiviert durch diesen Teilerfolg konnte zügig die hard- und softwaretechnische Realisierung der Temperatur- und Feuchtemessung durchgeführt werden. Als zusätzliches Funktionsmodul wurde innerhalb kurzer Zeit das LC-Display angesprochen, zum einen über den vorgesehenen Anschluss des Entwicklungsboards und zum anderen über den I²C-Bus mittels eines 8-Bit-Port-Bausteins.

Neben den fachlichen Arbeiten und Aufgaben stand die Teamarbeit und die Zeitplanung im Vordergrund. Die Autorengruppe besteht aus drei Personen, dadurch konnten von Beginn an klare Definitionen über die Projektmeilensteine und die Aufgabenverteilung erstellt werden. Damit war auch der Zeitplan realistisch erstellt worden. Mit Ausnahme der Entwicklung des TCP/IP-Stacks konnten alle Teilaufgaben zeitgerecht gelöst werden.

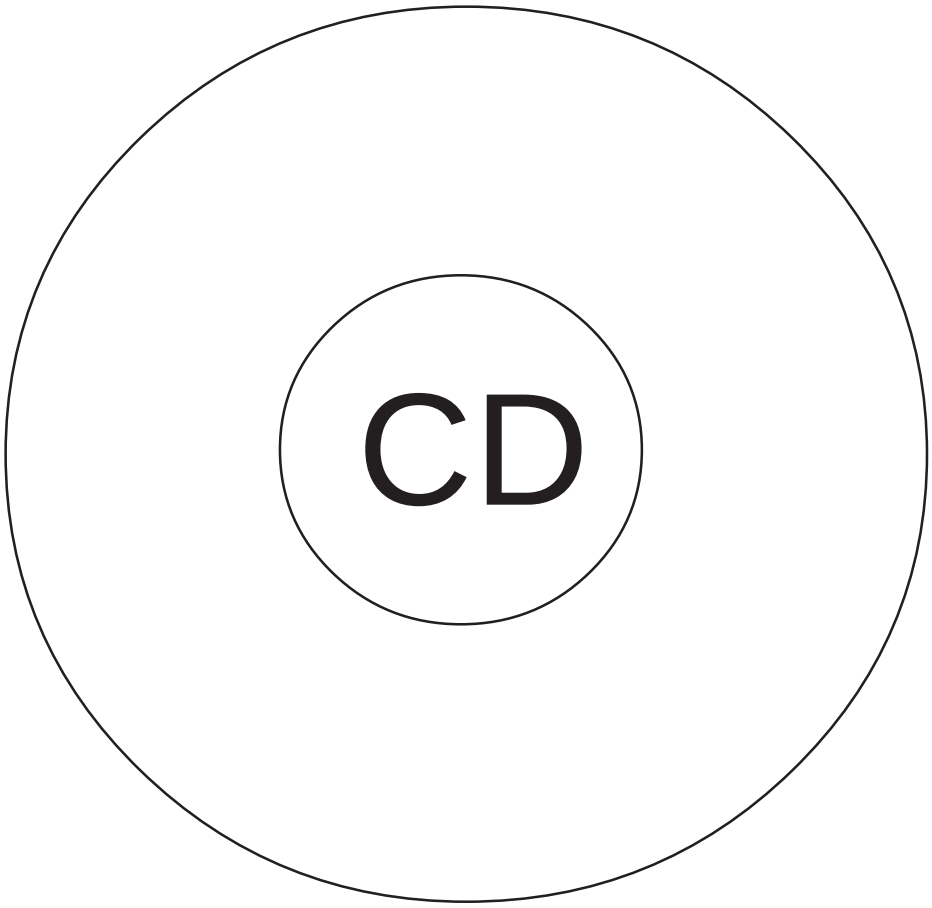
Da erste Funktionstests für den Stack fehlgeschlagen sind, musste viel Zeit für die Fehlersuche und Anpassung der Software investiert werden. Trotz der oft sehr hilfreichen Hinweise von Mitarbeitern der Fa. Beck zu Problemen bei der Stack-Programmierung, wurde die Motivation der Autoren gebremst, da mit den Hilfestellungen auch oft der Sinn einer Neuentwicklung hinterfragt wurde. Auf Grund des großen Zeitverlusts bei diversen Tests mit Programmen des Chip-Herstellers und der benötigten Anpassung verblieb ein zu geringer Zeitraum zur Lösung des Kommunikationsproblems.

Für weiterführende Projekte sind die Erfahrungen dieser Arbeit eine Grundlage, die die Probleme mit der Stack-Programmierung aufzeigen. Einige erkannte Fehler und Probleme können hiermit schon in die Grundüberlegungen mit einbezogen werden. Die Autoren bewerten diese Projektarbeit trotz der unvollständigen Aufgabenlösung als lehr- und erfolgeich.

7 Anhang

Der Anhang dieser Projektarbeit beinhaltet eine CD mit allen verwendeten Daten, Quellen, etc. und das Quellenverzeichnis

7.1 Materialien (auf CD)



Inhalt der CD:

- Datenblätter
- Dokumentation der Projektarbeit
- Dokumentation des SC12
- Elektor
- Listings
- Temperatursensor_SHT1x
- Webauftritt projekt.studentenbude.net

7.2 Quellenverzeichnis

[1] Beck: *Handbuch-SC12HW_v11*, .pdf.-Datei, siehe Anhang-CD

[2] Schwarzer, Malte u.a.: *Webauftritt der BBS Winsen*, Interner: http://www.goblack.de

[3] Beck: *DK40 Datasheet & Hardware Manual, Version 2*, DK40-Manual_engV2_020502.pdf, siehe Anhang-CD

[4] Beck: *IPC@CHIP Documentation Index,* APIDOC0104.pdf , siehe Anhang-CD

[5] Helmut Vogel: *Gerthsen Physik.*19. Aufl. Berlin u.a.: Springer, 1997.

[6] akademie.de: *Luftfeuchtigkeit*, Internet: http://www.net-lexikon.de/Luftfeuchtigkeit.html, siehe Anhang-CD

[7] Sensirion: *SHT1x/SHT7x*, Datasheet_SHT1x_SHT7x.pdf (engl.), siehe Anhang-CD oder Internet: http://www.sensirion.com

[8] Sensirion: *SHT1x/SHT7x,* datenblatt_sht11_d_lo_0202_a.pdf, siehe Anhang-CD

[9] Thomas Nezel: *Gassensoren*, Skript, Internet: http://www.chemsens.ethz.ch/~tomas/pdf_files/Skript.pdf, siehe Anhang-CD

[10] nach ISO 7498-1 (DIN ISO 7498) und ITU-T (CCIT) X.200

[11] Design & Elektronik extra: *Embedded Internet*, September 2000

[12] Comer, Douglas: *Internetworking With TCP/IP Volume 1-3,* zusammengefasst durch die Vorlesung *TCP/IP* an der FH WF/BS

[13] Beck, *IPC@CHIP SC12 Application Note SC12-SHT11,* SC12-SHT11.pdf

[14] STMicroelectronics: *L7800 Series*, e_l7800.pdf, siehe Anhang-CD

[15] Fa. Beck: *Newsgroup IPC@CHIP*, Internet: http://www.beck-ipc.com/ipc/support/forum/topic.asp?forum=1&sp=en

[16] Martin Fiutak: *Bundesbehörden nutzen gemeinsam Superrechner*, Internet: http://www.zdnet.de/news/business/0,39023142,39118415,00.htm

[17] Beck: *Scalable @CHIP-RTOS variants for the SC12 IPC@CHIP*, atchip_rtos_Variants_V110.pdf, Anhang-CD

[18] Elektor Verlag, *Mini-Web-Server*, Heft 5-9/2001, siehe Anhang-CD

[19] Snifferprogramme: *PacketMon v 1.0*, Internet: http://www.analogx.com, *EtherDetect v 1.1*, Internet: http://www.etherdetect.com