

Eingebettete Systeme

Prof. Dr. rer. nat. Peter Gerwinski

22. November 2018

Eingebettete Systeme

<https://gitlab.cvh-server.de/pgerwinski/es.git>

- 1 Einführung**
- 2 Einführung in Unix**
- 3 TCP/IP in der Praxis**
 - 3.0 Vorbereitungen
 - 3.1 IP-Adressen
 - 3.2 TCP- und UDP-Ports
 - 3.3 TCP-Protokolle
 - 3.4 Routing
 - 3.5 Netzwerkanalyse
 - 3.6 SSH
 - 3.7 X11
 - 3.8 GNU screen
 - 3.9 Programmierung
- 4 Bus-Systeme**
- 6 Echtzeit**

...



Änderungen
vorbehalten

3.0 Vorbereitungen

- Verkabelung: Twisted-Pair-Kabel, Switches
- Automatismen abschalten

3.1 IP-Adressen

- IP-Adresse zuweisen:

`ip addr` (Linux)

`ifconfig` (Unix allgemein)

`ipconfig` (MS Windows)

- Beispiel:

`ip addr add 192.168.42.197/24`

- Verbindung prüfen:

`ping <IP-Adresse>`

3.1 IP-Adressen

IPv4-Adressen:

- 32 Bit
- dezimal, 4 Gruppen zu je 8 Bit (0–255), durch Punkte getrennt

IPv6-Adressen:

- 128 Bit
- hexadezimal, 8 Gruppen zu je 4 Hex-Ziffern, durch Doppelpunkte getrennt
- Führende Nullen dürfen weggelassen werden.
- Zwei Doppelpunkte bedeuten: Mit Nullen auffüllen.
- Literatur und Beispiel: <https://de.wikipedia.org/wiki/IPv6>

3.2 TCP- und UDP-Ports

- `nc <IP> <Port>`
Verbindung zu Programm $\langle \text{Port} \rangle$ auf Rechner $\langle \text{IP} \rangle$ aufnehmen
- `nc -l <Port>` oder `nc -p <Port> -l`
auf eingehende Verbindungen warten („lauschen“)
- TCP-Ports: Verbindungskonzept, Netzwerk prüft
- UDP-Ports: einzelne Pakete, Anwendung muß selbst prüfen
- ICMP: keine Ports, nur Rechner:
Erreichbarkeit, Eigenschaften der Übertragung

Anwendung: HTTP, SMTP, ...
Transport: TCP-/UDP-Ports, ICMP
Internet: IP-Adresse
Netzwerkzugang: Hardware-/MAC-Adresse

3.3 TCP-Protokolle

- HTTP

```
$ nc -C <Rechner> 80  
GET / HTTP/1.1  
Host: www.hs-bochum.de  
(Leerzeile)
```

3.3 TCP-Protokolle

- HTTP

```
$ nc -C <Rechner> 80
```

```
GET / HTTP/1.1
```

```
Host: www.hs-bochum.de
```

```
(Leerzeile)
```

URL: Schema://Benutzer:Passwort@Rechner:port/Pfad?Query#Fragment

3.3 TCP-Protokolle

- **HTTP**

```
$ nc -C <Rechner> 80
GET / HTTP/1.1
Host: www.hs-bochum.de
(Leerzeile)
```

- **SMTP**

```
HELO cassini
MAIL FROM: <example@example.com>
RCPT TO: <beispiel@example.de>
(E-Mail-Header – Teil der Nutzdaten)
From: Eddie Example <example@example.com>
To: Bert Beispiel <beispiel@example.de>
Subject: Hello, world!
(Leerzeile)
Hi, there!
.
```

- Protokolle „mal eben“ selbst schreiben: [inetd](#)

3.4 Routing

- `ip route` (Linux)
- `route` (MS-Windows, Unix)
- `netstat -nr` (MacOS)

192.168.42.x 21 BW gateway - 192.168.7.27

80 Dimi

240 Skymaster

25 Heisenberg

„default“ = 0.0.0.0/0

192.168.42.66



192.168.42. ...

192.168.42.21 — ... — 195.37.15.82

↑
iptables -t nat -A POSTROUTING
-o <interface> -j MASQUERADE



Internet
Ⓢ Network Address Translation

7.27 ^{USB} 192.168.7.23
_{Net2} Weik El Berto

192.168.43.23

...43.7

...43.707

141.1.1.1

route add 192.168.42.0/24

gw ...

3.5 Netzwerkanalyse

- tcpdump
- wireshark
- ettercap

3.6 SSH

- `SSH <Rechner>`
- `-C`: Komprimierung
- `-L`: lokalen Port auf Remote-Port umleiten
- `-R`: Remote-Port auf lokalen Port umleiten

3.7 X11

- Grafik-Bildschirm und Eingabegeräte über's Netz
- `DISPLAY`-Variable: X-Server: Rechner und Bildschirm
- `ssh -X`: X11-Forwarding

3.8 GNU screen

- Text-Bildschirm und Eingabegeräte über's Netz
- `Ctrl+A c`: neues Fenster (create)
- `Ctrl+A Leertaste`: nächstes Fenster
- `Ctrl+A 3`: Fenster Nr. 3
- `Ctrl+A ESC`: hochscrollen, suchen, markieren (copy)
- `Ctrl+A Ctrl+]`: einfügen (paste)
- `Ctrl+A d`: von `screen` ablösen (detach)
- `screen -x`: an laufenden `screen` andocken
- ähnliche Funktionalität für Grafik: `x2go`

3.9 Programmierung

- `tcpip-server-0.c`: auf Port 1234 lauschen, „Hello, world!“ senden
- `tcpip-client-0.c`: Webseite <http://ngc224.gerwinski.de> abrufen

4 Bus-Systeme

4.1 Was sind Bus-Systeme?

Ein Bus ist ein System zur Datenübertragung zwischen mehreren Teilnehmern über einen gemeinsamen Übertragungsweg.

[https://de.wikipedia.org/wiki/Bus_\(Datenverarbeitung\)](https://de.wikipedia.org/wiki/Bus_(Datenverarbeitung))

Beispiele:

- Computer kommuniziert mit Peripherie
- Computer kommunizieren (direkt) miteinander
- Prozessor kommuniziert mit externem Speicher
- Teile eines Prozessors kommunizieren miteinander

4 Bus-Systeme

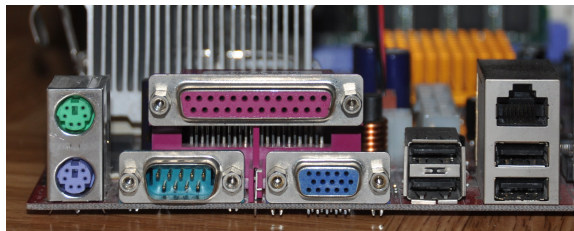
4.1 Was sind Bus-Systeme?

Standard-Computer:

- Einsteckkarten: PCI (und Vorgänger)
- Festplatten: SATA (und Vorgänger)
- USB, FireWire, ...
- Ethernet, CAN-Bus, ...
- WLAN, BlueTooth, IR, ...
- PS/2, RS-232, Centronics

Minimal-Hardware:

- RS-232
- I²C (TWI)
- SPI



4 Bus-Systeme

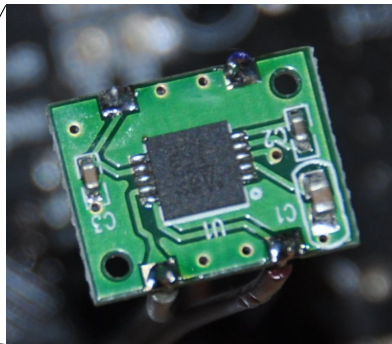
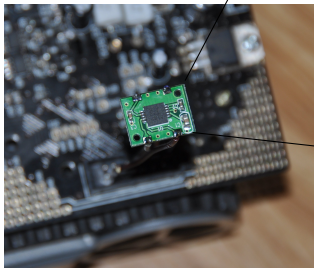
4.1 Was sind Bus-Systeme?

Standard-Computer:

- Einsteckkarten: PCI (und Vorgänger)
- Festplatten: SATA (und Vorgänger)
- USB, FireWire, ...
- Ethernet, CAN-Bus, ...
- WLAN, BlueTooth, IR, ...
- PS/2, RS-232, Centronics

Minimal-Hardware:

- RS-232
- I²C (TWI)
- SPI



4 Bus-Systeme

4.1 Was sind Bus-Systeme?

<i>seriell</i>	jedes Bit einzeln übertragen
<i>parallel</i>	mehrere Bits gleichzeitig
<i>synchron</i>	Abgleich über Steuerleitung: <i>Takt</i>
<i>asynchron</i>	Abgleich über Zeitvereinbarungen
<i>Punkt-zu-Punkt</i>	genau zwei Teilnehmer
<i>busfähig</i>	mehrere Teilnehmer, mit <i>Adressierung</i>

- I²C: seriell, synchron, mit Adressierung
- RS-232: seriell, asynchron, Punkt-zu-Punkt
- RS-485, USB, CAN: seriell, asynchron, mit Adressierung
- SPI: seriell, synchron, Punkt-zu-Punkt oder mit Adressierung

4 Bus-Systeme

4.2 RS-232

seriell

- *TX*: 1 Leitung für Daten
- *RX*: ggf. 1 Leitung für Daten in der anderen Richtung
- *GND*: gemeinsame *Masse*
- evtl. zusätzliche Steuerleitungen

asynchron

- *keine* Taktleitung für Abgleich, wann Daten anliegen
- Stattdessen: Abgleich über Zeitvereinbarungen

→ Jeder Teilnehmer braucht eine eigene Zeitbasis.

Punkt-zu-Punkt

- nur 2 Teilnehmer vorgesehen

4.2 RS-232

Synchronisation

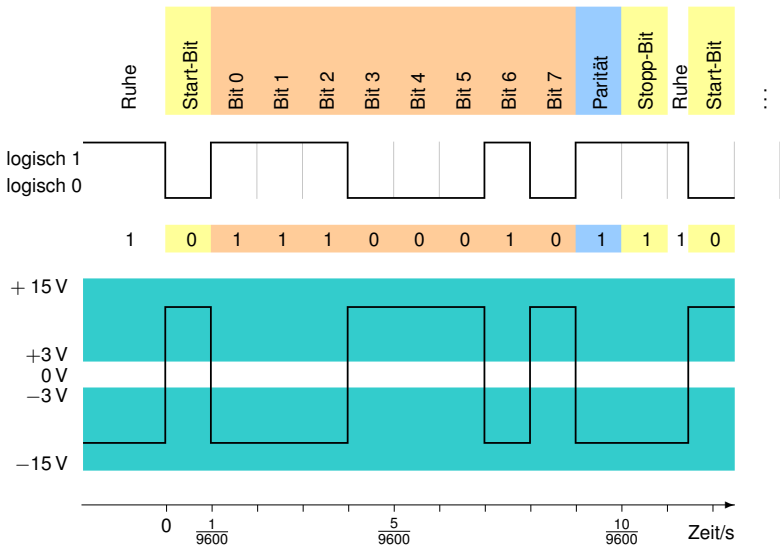
Daten

Check

9600 Baud, 8 Daten-Bits, ungerade Parität, 1 Stopp-Bit

Beispiel-Daten: ASCII „G“ = 71 = 0100 0111 binär

Übertragung der Daten von rechts (Bit 0) nach links (Bit 7)



4 Bus-Systeme

4.3 I²C (TWI)

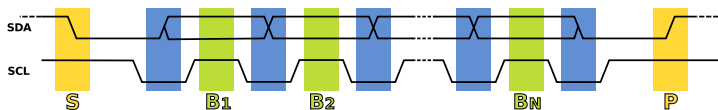
I²C = Inter-IC; TWI = Two-Wire-Interface

seriell

- *SDA*: 1 Leitung für Daten (in beiden Richtungen)
- *SCL*: Taktleitung (Clock)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung



busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- erstes gesendetes Byte: *Adresse* des Teilnehmers
- 2 Adressen pro Teilnehmer: Lesen/Schreiben

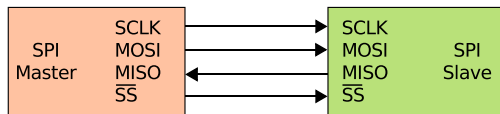
4 Bus-Systeme

4.4 SPI

Serial Peripheral Interface

seriell

- *MOSI*: Master Out, Slave In
- *MISO*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- \overline{SS} : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät



synchron

- Abgleich über Taktleitung

busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- *Slave* wird über *Slave Select* ausgewählt

4 Bus-Systeme

4.4 SPI

Serial Peripheral Interface

seriell

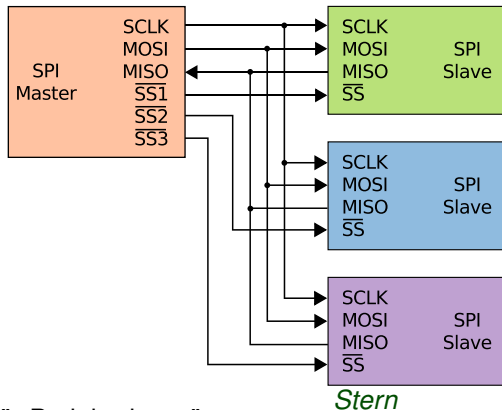
- *MOSI*: Master Out, Slave In
- *MISO*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- \overline{SS} : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung

busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- *Slave* wird über *Slave Select* ausgewählt



4 Bus-Systeme

4.4 SPI

Serial Peripheral Interface

seriell

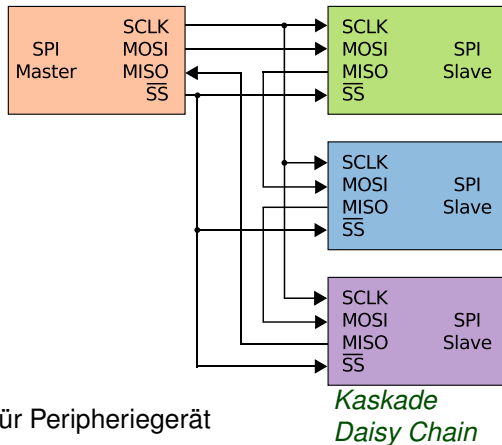
- *MOSI*: Master Out, Slave In
- *MISO*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- \overline{SS} : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung

busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- *Slave* wird über *Slave Select* ausgewählt



4 Bus-Systeme

4.4 SPI

Serial Peripheral Interface

seriell

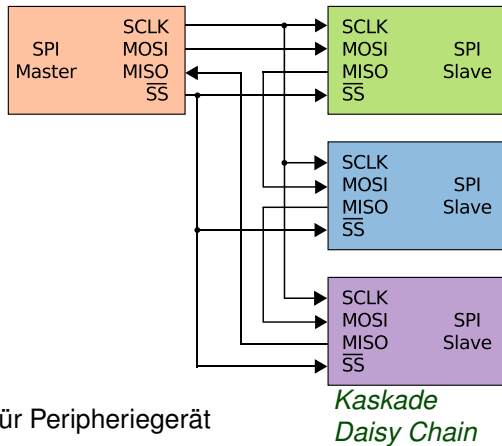
- *MOSI*: Master Out, Slave In
- *MISO*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- \overline{SS} : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung

busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- *Slave* wird über *Slave Select* ausgewählt



Slave gibt MOSI-Input um 1 Takt verzögert an MISO aus → Master setzt „im richtigen Moment“ \overline{SS}

Eingebettete Systeme

<https://gitlab.cvh-server.de/pgerwinski/es.git>

1 Einführung

2 Einführung in Unix

3 TCP/IP in der Praxis

4 Bus-Systeme

5.1 Was sind Bus-Systeme?

5.2 RS-232

5.3 I²C (TWI)

5.4 SPI

5.5 PWM

6 Echtzeit

5.1 Was ist Echtzeit?

5.2 Echtzeitprogrammierung

5.3 Multitasking

5.4 Ressourcen

5.5 Prioritäten

...



Änderungen
vorbehalten