



# 64-210 Eingebettete Systeme

<https://lernen.min.uni-hamburg.de>

[https://tams.informatik.uni-hamburg.de/  
lectures/2020ss/vorlesung/es](https://tams.informatik.uni-hamburg.de/lectures/2020ss/vorlesung/es)

Andreas Mäder



Universität Hamburg  
Fakultät für Mathematik, Informatik und Naturwissenschaften  
Fachbereich Informatik

Technische Aspekte Multimodaler Systeme

Sommersemester 2020



## 1. Organisatorisches

Vorlesung

Übungen

Allgemein

## 2. Literatur



... dieses Semester ist wegen *Corona* alles anders

- ▶ Videomaterial, Foren und die gesamte Organisation sind im MIN-Moodle hinterlegt:  
<https://lernen.min.uni-hamburg.de>
- ▶ eigenständiges Lernen zu Hause
  - ⇒ Screencast Videos im Moodle / auf Lecture2Go
- ▶ regelmäßige Online-Meetings
  - ▶ Vorlesung: Fr., ab 12:30
  - ▶ Übung: Di., ab 14:15 und Fr., ab 10:15
- ▶ bei Unklarheiten: Fragen in entsprechenden **Q&A** Foren stellen
  - ⇒ die Antworten bilden dort eine Sammlung „zum Nachlesen“

# Vorlesung (cont.)

[tams.informatik.uni-hamburg.de/lectures/2020ss/vorlesung/es](https://tams.informatik.uni-hamburg.de/lectures/2020ss/vorlesung/es)

- ▶ Vorlesung folgt: „*Embedded System Design*“ von P. Marwedel
- ▶ eBook: [dx.doi.org/10.1007/978-3-319-56045-8](https://dx.doi.org/10.1007/978-3-319-56045-8) UHH Campus
- ▶ Originalmaterial: [ls12-www.cs.tu-dortmund.de/daes/de/daes/mitarbeiter/prof-dr-peter-marwedel/embedded-system-text-book/slides/slides-2013.html](https://ls12-www.cs.tu-dortmund.de/daes/de/daes/mitarbeiter/prof-dr-peter-marwedel/embedded-system-text-book/slides/slides-2013.html)
- ▶ diverse gute Lehrbücher — Empfehlungen s.u.

- ▶ Einzelarbeit zu Hause, simulativ
- ▶ Betreuung in Online-Meetings: Di., ab 14:15 und Fr., ab 10:15  
Beginn: ab dem **28.04.**
- ▷ spätere Aufgaben: „vor Ort“, an der Hardware

## Grundidee

- ▶ praktische Ergänzung zur Vorlesung
- ▶ Programmieraufgaben mit Mikrocontrollern (und FPGAs)
  - ▶ Anschluss von Sensoren und Aktoren
  - ▶ Kontrollaufgaben (Steuerung durch endlichen Automaten)
  - ▶ Interruptbehandlung
  - ▶ einfache Schnittstellen (seriell, I<sup>2</sup>C...)
  - ▶ Kommunikation
  - ▶ ...
- ▶ spätere Aufgaben bauen auf Vorherigen auf
  - ⇒ fertige Lösungen abspeichern und gut dokumentieren
  - ⇒ automatisierte Tests erstellen

# Übungen: Hardwareplattformen

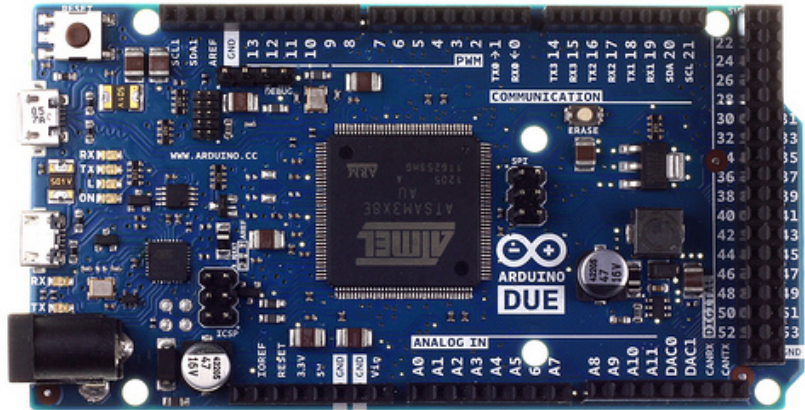
## µController System

Organisatorisches - Übungen

64-210 Eingebettete Systeme

### ► Arduino Due

[www.arduino.cc](http://www.arduino.cc)



# Übungen: Hardwareplattformen (cont.)

## $\mu$ Controller System

- ▶ Arduino Due [www.arduino.cc](http://www.arduino.cc)
  - ▶ Microcontroller AT91SAM3X8E ARM Cortex-M3, 32bit
  - ▶ Clock Speed 84 MHz
  - ▶ Operating Voltage 3,3V
  - ▶ Digital I/O Pin 54 (12  $\times$  PWM output)
  - ▶ Analog Input Pins 12
  - ▶ Analog Outputs Pins 2 (DAC)
  - ▶ SRAM 96 KB (two banks: 64KB and 32KB)
  - ▶ Flash Memory 512 KB
- ▶ typische Prototypenplatine mit frei belegbaren Ein-/Ausgängen
- ▶ Anschluss externer Komponenten über Pfostenleisten
- ▶ sehr große Anzahl fertiger „Shields“
  - ▶ Sensoren
  - ▶ Aktoren, Motortreiber
  - ▶ Displays
  - ▶ Schnittstellen

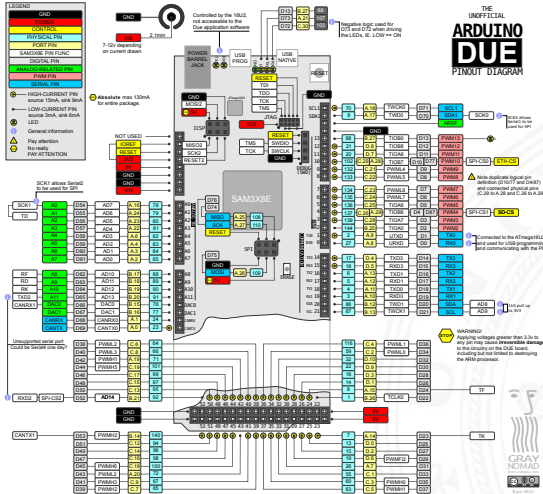
# Übungen: Hardwareplattformen (cont.)

## µController System

Organisatorisches - Übungen

64-210 Eingebettete Systeme

### ► I/O-Schema

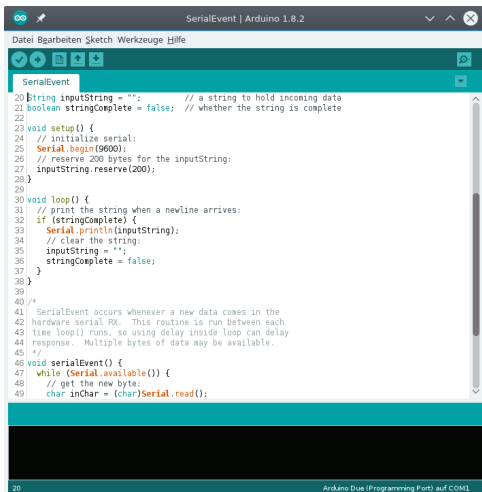




# Übungen: Hardwareplattformen (cont.)

## µController System

### ► Arduino-IDE / C-Code



```
SerialEvent | Arduino 1.8.2
Datei Bearbeiten Sketch Werkzeuge Hilfe
SerialEvent
20 String inputString = ""; // a string to hold incoming data
21 boolean stringComplete = false; // whether the string is complete
22
23 void setup() {
24   // initialize serial:
25   Serial.begin(9600);
26   // reserve 200 bytes for the inputString:
27   inputString.reserve(200);
28 }
29
30 void loop() {
31   // print the string when a newline arrives:
32   if (stringComplete) {
33     Serial.println(inputString);
34     // clear the string:
35     inputString = "";
36     stringComplete = false;
37   }
38 }
39
40 /*
41  * SerialEvent occurs whenever a new data comes in the
42  * hardware serial RX. This routine is run between each
43  * time loop() runs, so using delay inside loop can delay
44  * response. Multiple bytes of data may be available.
45  */
46 void serialEvent() {
47   while (Serial.available()) {
48     // get the new byte:
49     char inChar = (char)Serial.read();
```

# Übungen: Hardwareplattformen

## FPGA-Prototypensystem

### ▶ Altera DE0-Nano

[www.terasic.com.tw](http://www.terasic.com.tw)

[www.intel.de/content/www/de/de/products/programmable.html](http://www.intel.de/content/www/de/de/products/programmable.html)

### ▶ programmierbare Hardware: FPGA

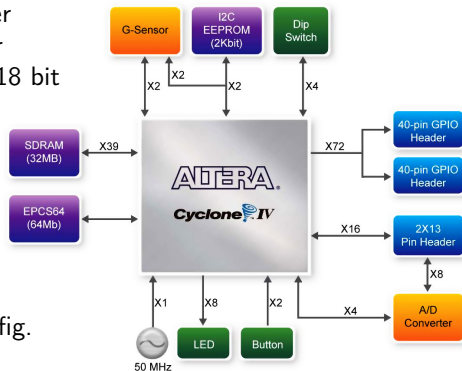
- ▶ Cyclone IV EP4CE22F17C6N [60 nm Prozess, 2009]
- ▶ 153 I/O Pins, gesamt 256 Pins
- ▶ 22 320 LEs  $\approx$  270 000 Gatter
- ▶ 594 Kbit (interner) Speicher
- ▶ 66 HW-Multiplizierer:  $18 \times 18$  bit
- ▶ 4 PLLs

### ▶ On-Board Speicher

- ▶ 32MB SDRAM
- ▶ 2Kbit I<sup>2</sup>C EEPROM

### ▶ Konfiguration

- ▶ über USB Schnittstelle
- ▶ EPCS64: Flash, serielle Konfig.





# Übungen: Hardwareplattformen (cont.)

## FPGA-Prototypensystem

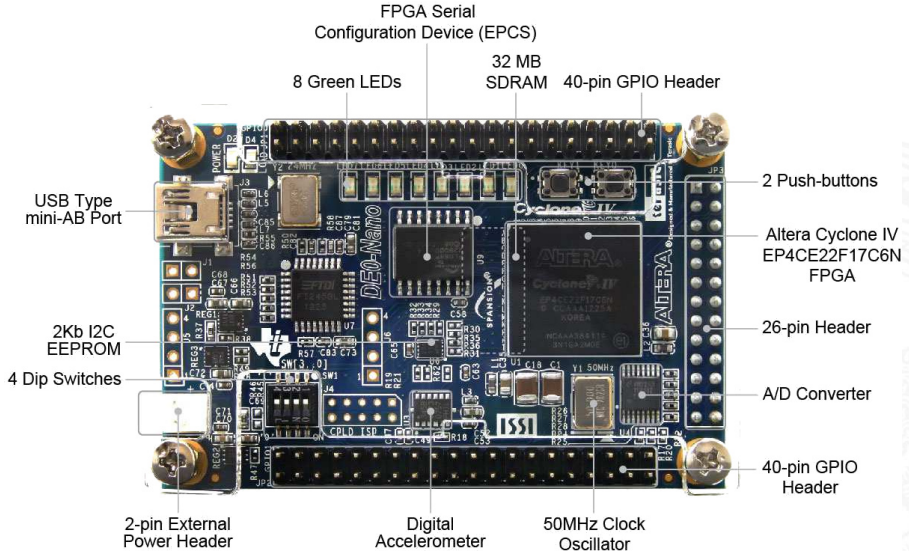
- ▶ Ein-/Ausgabe
  - ▶ 8 LEDs
  - ▶ 2 Taster
  - ▶ 4 DIP Schalter
- ▶ Beschleunigungssensor: ADXL 345, 3-Achsen, 13-bit Auflösung
- ▶ A/D Wandler: ADC128S022, 8-Kanal, 12-bit Auflösung
- ▶ Erweiterungsstecker
  - ▶ 2 × 40-Pin: 72 I/O Pins + Spannungsversorgung; 5V, 3,3V, Gnd
  - ▶ 26-Pin: 16 I/O Pins + 8 analoge Eingänge

# Übungen: Hardwareplattformen (cont.)

## FPGA-Prototypensystem

Organisatorisches - Übungen

64-210 Eingebettete Systeme



# Übungen: Hardwareplattformen (cont.)

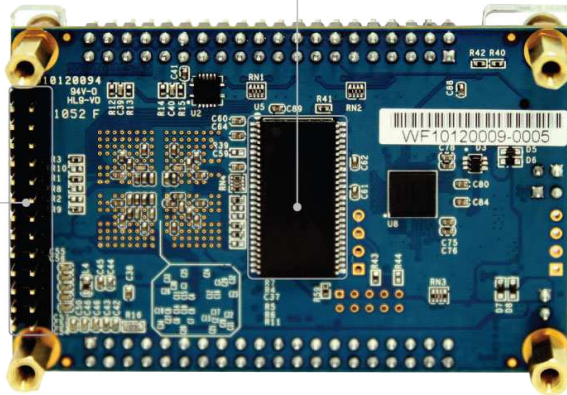
## FPGA-Prototypensystem

Organisatorisches - Übungen

64-210 Eingebettete Systeme

32MB SDRAM

2X13 Pin Header





# Übungen: Hardwareplattformen (cont.)

## FPGA-Prototypensystem

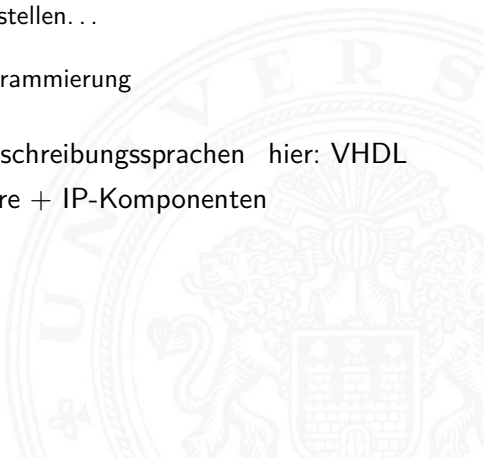
### Implementationsmöglichkeiten

#### 1. Softwareentwurf

- ▶ IP-Komponenten (*Intellectual Property*)  
Speicher, Busse, I/O-Schnittstellen. . .
- ▶ 32-bit Prozessor (NIOS II)
- ▶ Softwareentwicklung: C Programmierung  
Eclipse + Cross-Compiler

#### 2. Hardwareentwurf: Hardwarebeschreibungssprachen hier: VHDL

#### 3. gemischt: Hardware + Software + IP-Komponenten



# Übungen: Hardwareplattformen (cont.)

## FPGA-Prototypensystem

### ► IP-Komponenten

The screenshot shows the Platform Designer IP Catalog window. The 'Library' pane on the left is expanded to 'Clocks, PLLs and Resets'. The main pane displays a table of IP components:

Use	Conn.	Name	Description	Export	Clock	Base	End
		clk_0	Clock Source				
		clk_in	Clock Input				
		clk_in_reset	Reset Input	clk_reset	exported		
		clk	Clock Output			clk_0	
		clk_reset	Reset Output	enable-clk to export disable-clk to export			

The 'Messages' pane at the bottom is empty. The status bar at the bottom shows '0 Errors, 0 Warnings' and buttons for 'Generate HDL' and 'Finish'.

# Übungen: Hardwareplattformen (cont.)

## FPGA-Prototypensystem

Organisatorisches - Übungen

64-210 Eingebettete Systeme

### ► NIOS Prozessor

**Nios II (Classic) Processor**  
altera\_nios2\_qsys

**Block Diagram**  
Show signals

**Core Nios II** | Caches and Memory Interfaces | Advanced Features | MMU and MPU Settings | JTAG Debug Module

Select a Nios II Core

Nios II Core

Nios IIe  
 Nios II  
 Nios IIT

	Nios II/e	Nios II/s	Nios II/T
<b>Nios II</b> Selector Guide	RISC 32-bit	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide Barrel Shifter Data Cache Dynamic Branch Prediction
Memory Usage (e.g. Stratix IV)	Two MRMs (or equiv.)	Two MRMs + cache	Three MRMs + cache

**Hardware Arithmetic Operation**  
Hardware multiplication type: Embedded Multipliers  
 Hardware divide

**Reset Vector**  
Reset vector memory: None  
Reset vector offset: 0x0000000  
Reset vector: 0x0000000

**Exception Vector**  
Exception vector memory: None  
Exception vector offset: 0x0000000  
Exception vector: 0x0000000

**MMU and MPU**  
 Include MMU  
Only include the MMU using an operating system that explicitly supports an MMU.  
Fast TLB Miss Exception vector memory: None  
Fast TLB Miss Exception vector offset: 0x0000000  
Fast TLB Miss Exception vector: 0x0000000

❌ Error: nios2\_qsys\_0 Reset slave is not specified. Please select the reset slave  
❌ Error: nios2\_qsys\_0 Exception slave is not specified. Please select the exception slave  
⚠ Warning: nios2\_qsys\_0 Nios II Classic cores are no longer recommended for new projects

Cancel Finish



# Übungen: Hardwareplattformen (cont.)

## FPGA-Prototypensystem

Organisatorisches - Übungen

64-210 Eingebettete Systeme

### ▶ VHDL-Code

```
de0Board.vhd -- Kate
File Edit View Projects Bookmarks Sessions Tools Settings Help
de0Board.vhd cDisp14x6.vhd
-- normal display, no timeout:
when dNorm1 =>
  if ack = req then
    then sTransition(dClear1, dispClear);
  end if;
-- clear display:
when dClear1 =>
  if ack = req then
    then sTransition(dAllChars, dispChar, timeC/4);
    char << character'left';
  end if;
-- display char-set: 0..255, timeC/4 between chars:
when dAllChars =>
  if timer = 0 then
    if ack = req then
      if char = character'right'
        then sTransition(dInv, dispInverse, timeC);
        else sTransition(dAllChars, dispChar, timeC/4);
        char << character'succ(char);
      end if;
    else
      timer := timer-1;
    end if;
  else
    timer := timer-1;
  end if;
-- invert display, for timeC:
when dInv =>
  tTransition(dNorm2, dispNormal, timeC);
-- normal display, no timeout:
when dNorm2 =>
  if ack = req then
    then sTransition(dClear2, dispClear);
  end if;
-- clear display:
when dClear2 =>
  if ack = req then
    then sTransition(dXV, dispPosXY);
    xPos << 4;
    yPos << 2;
  end if;
-- set xPos=4, yPos=2:
when dXV =>
  if ack = req then
    then sTransition(dString, dispChar);
    invC << 1;
    char << strReg(1);
    strReg := strReg(2 to strReg'right)'1';
    strCnt << strCnt+1;
  end if;
-- string output, no timeout:
when dString =>
  if strCnt = 0 then
    state << halt;
    invC << 0;
  else
    sTransition(dStrIno, dispChar);
  end if;
Line 1, Column 1
INSERT Soft Tabs: 4 (8) UTF-8 VHDL
Q Search and Replace
```

```
cDisp14x6.vhd -- Kate
File Edit View Projects Bookmarks Sessions Tools Settings Help
de0Board.vhd cDisp14x6.vhd
-- architecture
architecture pc054 of cDisp14x6 is
  type stateTy is (power, reset, init, active);
  signal state : stateTy;
  signal dispEna : std_logic;
  signal ackL : std_logic;
  signal charBit : natural range 0 to 4031;
  signal charReg : std_logic_vector (47 downto 0);
  signal romA : std_logic_vector ( 6 downto 0);
  signal romD : std_logic_vector (39 downto 0);
begin
  -- set by LID
  bgLED << '1' when bgLight else '0';
  -- component instantiations
  rom: charROM port map (romA, clk, romD);
  ctrlP: process (clk, rstn) is
    procedure stInit is
      state << nextState;
      sReg << s0Data;
      charBit << regBit;
      charReg << regData;
    end procedure stInit;
  begin
    state << nextState;
    sReg << s0Data;
    charBit << regBit;
    charReg << regData;
    end procedure stInit;
  begin
    if rstn = '0' then -- async. reset
      dispEna << '0';
      state << power;
      sReg << 1;
      sReg << 0;
      charReg << (others << '0');
    elsif rising_edge(clk) then
      case state is
        -- power on / hardware reset
        when power =>
          ackL << '1';
          dispEna << '1';
          state << reset; -- 1. reset pc054
          sReg << 0;
          -- reset controller:
          when reset =>
            sReg << 1;
            stInitInit, 0, 39, -- 2. setup pc054
            intExtInC & setInIac & setVoc & intInIac & dispNormC & x'00';
            -- initialize controller.
      end case;
    end if;
  end process ctrlP;
end architecture pc054;
Line 117, Column 1
INSERT Soft Tabs: 4 (8) UTF-8 VHDL
Q Search and Replace
```



- ▶ über die Moodle Foren / E-Mail an mich
- ▶ Fehler und Ungenauigkeiten in den Folien und Materialien bitte melden
- ▶ Vorschläge und Hinweise auf Tools, schöne Lehrmaterialien etc. sind immer willkommen!

Problem: unterschiedliches Vorwissen!

- ▶ Voraussetzungen: insbesondere **Rechnerstrukturen** aber auch: Assemblerprogrammierung, Betriebssysteme, Netzwerke. . .
- ▶ generell: Interesse an Hardware



Dr. Andreas Mäder

E-Mail [maeder@informatik.uni-hamburg.de](mailto:maeder@informatik.uni-hamburg.de)

Tel. +49 40 42883 2502

Büro Informatikum, Haus F 317



[Mar18] P. Marwedel:

*Embedded System Design –  
Embedded Systems Foundations of Cyber-Physical Systems.*  
3rd edition. Springer-Verlag, 2018. ISBN 978-3-319-56043-4

**Primärliteratur:** Komplette Behandlung des Themas. Buch und Foliensatz sind Vorlage dieser Vorlesung. Die Folien basieren auf der zweiten Auflage des Buchs.

Originalmaterial der Vorlesung von Peter Marwedel: [ls12-www.cs.tu-dortmund.de/daes/daes/mitarbeiter/prof-dr-peter-marwedel/embedded-system-text-book/slides/slides-2013.html](http://ls12-www.cs.tu-dortmund.de/daes/daes/mitarbeiter/prof-dr-peter-marwedel/embedded-system-text-book/slides/slides-2013.html)

eBook in der Informatik-Bibliothek: [dx.doi.org/10.1007/978-3-319-56045-8](https://dx.doi.org/10.1007/978-3-319-56045-8)

[TH07] J. Teich, C. Haubelt:

*Digitale Hardware/Software-Systeme –  
Synthese und Optimierung.*

2. Auflage. Springer-Verlag, 2007. ISBN 978-3-540-46822-6

Schwerpunkte: Realisierung eingebetteter Systeme, also Synthese, Scheduling, Bindung von Ressourcen etc.

eBook in der Informatik-Bibliothek: [dx.doi.org/10.1007/978-3-540-46824-0](https://dx.doi.org/10.1007/978-3-540-46824-0)

[HT10] C. Haubelt, J. Teich:

*Digitale Hardware/Software-Systeme –  
Spezifikation und Verifikation.*

Springer-Verlag, 2010. ISBN 978-3-642-05355-9

Schwerpunkte: Methoden zur Spezifikation eingebetteter Systeme und der (formalen) Verifikation, Simulation etc.

eBook in der Informatik-Bibliothek: [dx.doi.org/10.1007/978-3-642-05356-6](https://dx.doi.org/10.1007/978-3-642-05356-6)

[VG02] F. Vahid, T. Givargis:

*Embedded System Design –  
A Unified Hardware/Software Introduction.*

John Wiley & Sons, 2002. ISBN 978-0-471-38678-0

Alle Themen im Überblick, etwas älter.

Mehrere Exemplare in der Informatik-Bibliothek vorhanden.

## [Brä08] T. Bräunl:

*Embedded Robotics – Mobile Robot Design and Applications with Embedded Systems.*

3rd edition. Springer-Verlag, 2008. ISBN 978–3–540–70534–5

Eigentlich ein „Robotik“-Buch, dient aber als Beispiel, dass die Robotik auf „Eingebetteten Systemen“ aufbaut (Teil 1)

eBook in der Informatik-Bibliothek: [dx.doi.org/10.1007/978-3-540-70534-5](https://dx.doi.org/10.1007/978-3-540-70534-5)

## [Lew13] D. Lewis:

*Fundamentals of embedded software – with the ARM Cortex-M3.*

2nd edition. Pearson Education, 2013.

ISBN 978–0–13–291654–7

Schwerpunkt: Software, viel praktische Programmierung in C und Assembler.

[SC17] J. Sanchez, M.P. Canton:

*Embedded Systems Circuits and Programming.*

CRC Press, 2017. ISBN 978-1-138-07406-4

Schwerpunkt auf PIC  $\mu$ Controller, enthält außerdem viele (elektro-) technische Grundlagen.

[Wol16] M. Wolf:

*Computers as Components –*

*Principles of Embedded Computing System Design.*

4th edition. Morgan Kaufmann Publishers Inc., 2016.

ISBN 978-0-12-805387-4

[Zöb08] D. Zöbel:

*Echtzeitsysteme – Grundlagen der Planung.*

Springer-Verlag, 2008. ISBN 978-3-540-76395-6

eBook in der Informatik-Bibliothek: [dx.doi.org/10.1007/978-3-540-76396-3](https://dx.doi.org/10.1007/978-3-540-76396-3)

## [BO15] R.E. Bryant, D.R. O'Hallaron:

*Computer systems – A programmers perspective.*

3rd global ed., Pearson Education Ltd., 2015. ISBN

978-1-292-10176-7

Rechnerarchitektur mit Schwerpunkt Software und Systeme, leider nicht ganz billig. Viele C-Programme und Systemprogrammierung. Beispiele anhand Intel x86 Architektur.

## [TA14] A.S. Tanenbaum, T. Austin:

*Rechnerarchitektur – Von der digitalen Logik zum Parallelrechner.*

6. Auflage, Pearson Deutschland GmbH, 2014.

ISBN 978-3-8689-4238-5

Guter Überblick zum Thema Rechnerarchitektur, klares didaktisches Konzept. Java VM, Intel x86, SPARC. Mit jeder Auflage komplett überarbeitet und aktualisiert.



[PH16a] D.A. Patterson, J.L. Hennessy:

*Computer Organization and Design – The Hardware Software Interface: ARM Edition.*

Morgan Kaufmann Publishers Inc., 2016.

ISBN 978-0-12-801733-3

Schönes Lehrbuch von den Entwicklern der RISC/MIPS Prozessoren.

[HP17] J.L. Hennessy, D.A. Patterson:

*Computer Architecture – A Quantitative Approach.*

6th edition, Morgan Kaufmann Publishers Inc., 2017.

ISBN 978-0-12-811905-1

Die Bibel zum Thema Rechnerarchitektur



[Mar11] M. Margolis:

*Arduino Cookbook –  
Recipes to Begin, Expand, and Enhance Your Projects.*  
2nd edition. O'Reilly, 2011. ISBN 978-1-4493-1387-6  
Die Arduino Referenz

[McR13] M. McRoberts:

*Beginning Arduino.*  
2nd edition. Apress, 2013. ISBN 978-1-4302-5016-6  
Praxisnahe Beispiele und Projekte  
eBook in der Informatik-Bibliothek: [dx.doi.org/10.1007/978-1-4302-5017-3](https://dx.doi.org/10.1007/978-1-4302-5017-3)

