



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

MIN-Fakultät
Fachbereich Informatik



64-210 Eingebettete Systeme

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

[https://tams.informatik.uni-hamburg.de/
lectures/2025ss/vorlesung/es](https://tams.informatik.uni-hamburg.de/lectures/2025ss/vorlesung/es)

Andreas Mäder



Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik
Technische Aspekte Multimodaler Systeme

Sommersemester 2025



1. Organisatorisches

Vorlesung

Übungen

Allgemein

2. Literatur





- ▶ Di. 12:15-13:45 D 125/129
Fr. 12:15-13:45 B 201
- ▶ Vorlesung folgt: „*Embedded System Design*“ von P. Marwedel
- ▶ Springer eBook
link.springer.com/book/10.1007/978-3-658-33437-6 -de-
link.springer.com/book/10.1007/978-3-030-60910-8 -en-
- ▶ Originalmaterial der TU Dortmund
daes.cs.tu-dortmund.de/embedded-system-text-book
- ▶ diverse gute Lehrbücher — Empfehlungen s.u.
- ▶ Informationen und Downloads in dem Moodle — **aktuell!**
- ▶ eingestreute Hinweise auf aktuelle Themen und Vertiefung



Organisation und Material

- ▶ Folien, Videomaterial und die Foren im MIN-Moodle:
<https://lernen.min.uni-hamburg.de>
- ▶ Vorlesung in Präsenz
ergänzend Videos (aus den Online-Jahren 2020 und 2021)
- ▶ bei Unklarheiten: Fragen in entsprechenden **Q&A** Foren stellen
⇒ die Antworten bilden dort eine Sammlung „zum Nachlesen“



- ▶ drei Gruppen: Di. nach und Fr. vor Vorlesung
- ▶ Gr.01 Di. 14:15-15:45 F 325
- Gr.02 —"— F 522 + F 525
- Gr.03 Fr. 10:15-11:45 F 325

- ▶ Beginn der Übungen (wg. Osterfeiertagen)
Gruppe 1 + 2 Di. 15.04. (KW 16)
Gruppe 3 Fr. 11.04. (KW 15)

- ▶ Durchführung
Vorbereitung: wichtig, sonst Zeitprobleme!
im Idealfall schon Code am eigene Rechner erstellen
ggf. simulativ (Proteus)
Übungstermin: praktische Arbeit mit 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, I²C. . .)
 - ▶ Kommunikation
 - ▶ . . .
- ▶ spätere Aufgaben bauen auf Vorherigen auf
 - ⇒ fertige Lösungen abspeichern und gut dokumentieren
 - ⇒ automatisierte Tests erstellen



Übungen: Hardwareplattformen

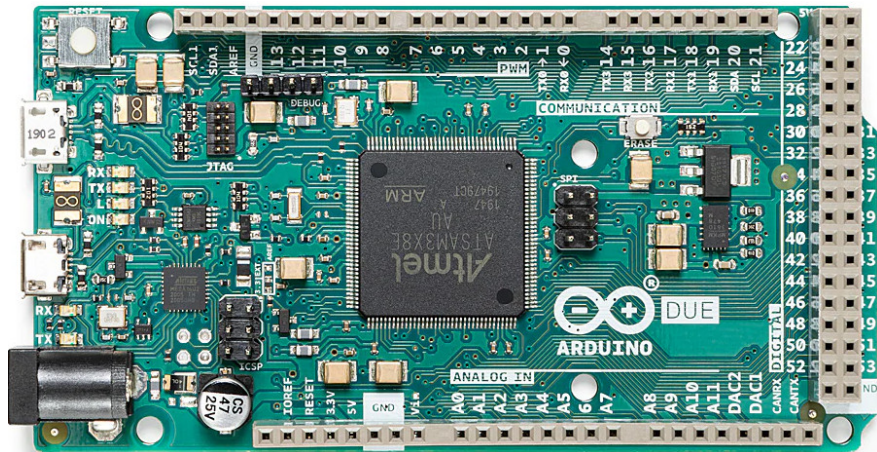
μ Controller System

Organisatorisches - Übungen

64-210 Eingebettete Systeme

► Arduino Due

www.arduino.cc





Übungen: Hardwareplattformen (cont.)

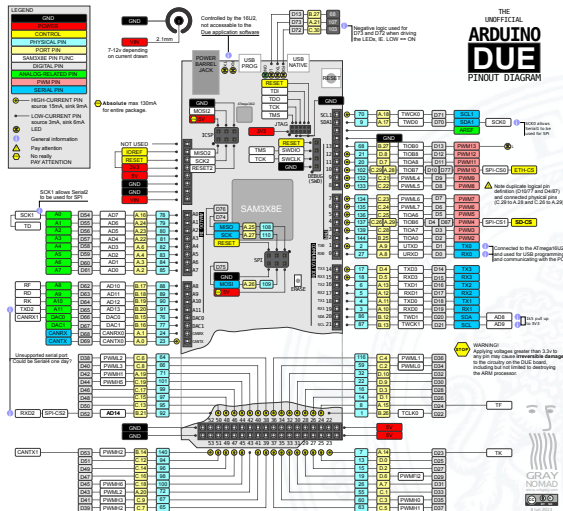
μ Controller System

Organisatorisches - Übungen

64-210 Eingebettete Systeme

- ▶ **Arduino Due** 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

► I/O-Schema





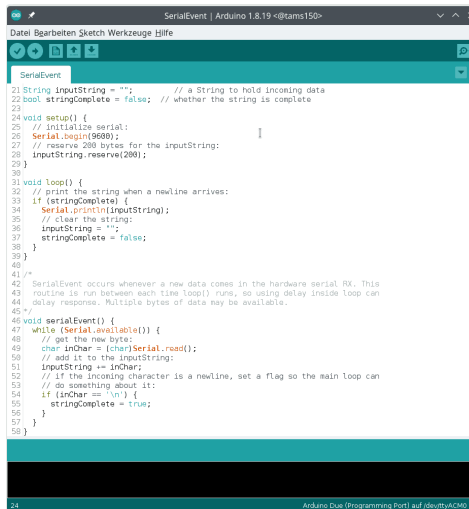
Übungen: Hardwareplattformen (cont.)

μController System

Organisatorisches - Übungen

64-210 Eingebettete Systeme

► Arduino-IDE / C-Code



```
SerialEvent | Arduino 1.8.19 <@tams150>
Datei Bearbeiten Sketch Werkzeuge Hilfe

SerialEvent

21 String inputString = "";           // a String to hold incoming data
22 bool stringComplete = false;      // whether the string is complete
23
24 void setup() {
25   // initialize serial:
26   Serial.begin(9600);
27   // reserve 200 bytes for the inputString:
28   inputString.reserve(200);
29 }
30
31 void loop() {
32   // print the string when a newline arrives:
33   if (stringComplete) {
34     Serial.println(inputString);
35     // clear the string:
36     inputString = "";
37     stringComplete = false;
38   }
39 }
40
41 /*
42  * SerialEvent occurs whenever a new data comes in the hardware serial RX. This
43  * routine is run between each time loop() runs, so using delay inside loop can
44  * delay 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();
50     // add it to the inputString:
51     inputString += inChar;
52     // if the incoming character is a newline, set a flag so the main loop can
53     // do something about it:
54     if (inChar == '\n') {
55       stringComplete = true;
56     }
57   }
58 }
```

24 Arduino Due (Programming Port) auf /dev/ttyACM0



Übungen: Hardwareplattformen

FPGA-Prototypensystem

Organisatorisches - Übungen

64-210 Eingebettete Systeme

► Altera DE0-Nano

www.terasic.com.tw

www.intel.de/content/www/de/de/products/programmable.html

► programmierbare Hardware: FPGA

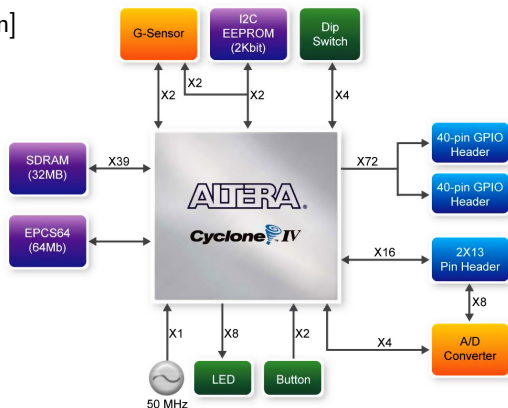
- Cyclone IV EP4CE22F17C6N [2009, 60 nm]
- 153 I/O Pins, gesamt 256 Pins
- 22 320 LEs $\approx$ 270 000 Gatter
- 594 Kbit (interner) Speicher
- 66 HW-Multiplizierer: 18×18 bit
- 4 PLLs

► On-Board Speicher

- 32MB SDRAM
- 2Kbit I²C EEPROM

► Konfiguration

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





Übungen: Hardwareplattformen (cont.)

FPGA-Prototypensystem

Organisatorisches - Übungen

64-210 Eingebettete Systeme

- ▶ 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

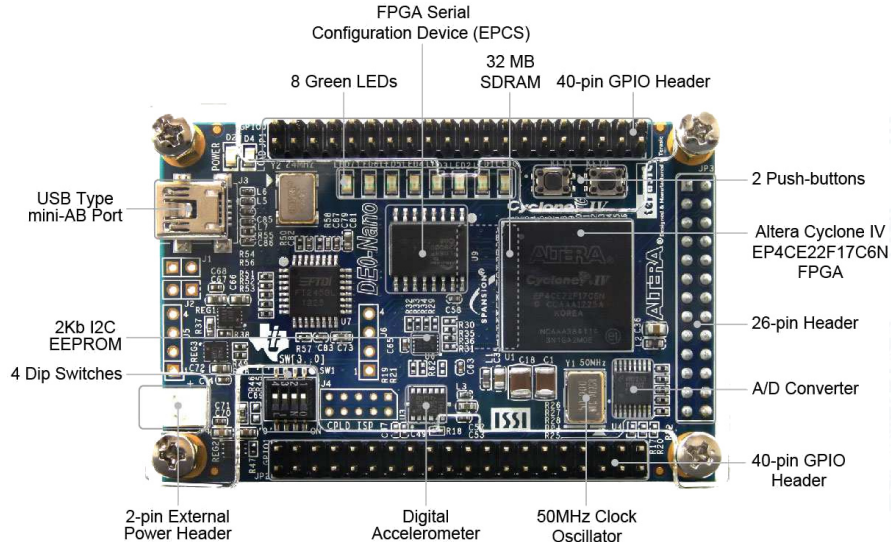


Übungen: Hardwareplattformen (cont.)

FPGA-Prototypensystem

Organisatorisches - Übungen

64-210 Eingebettete Systeme



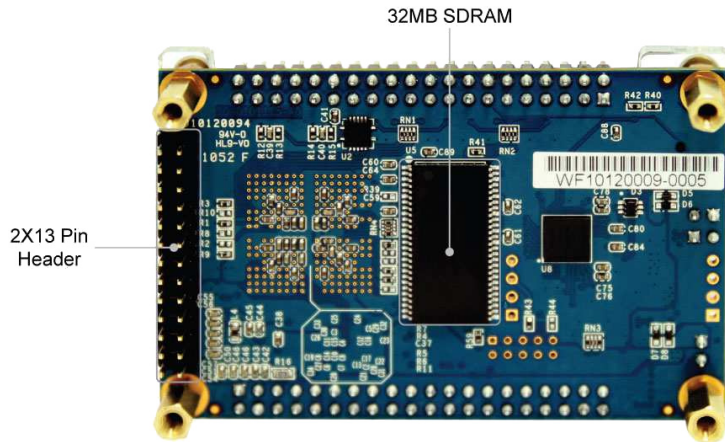


Übungen: Hardwareplattformen (cont.)

FPGA-Prototypensystem

Organisatorisches - Übungen

64-210 Eingebettete Systeme





Übungen: Hardwareplattformen (cont.)

FPGA-Prototypensystem

Organisatorisches - Übungen

64-210 Eingebettete Systeme

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



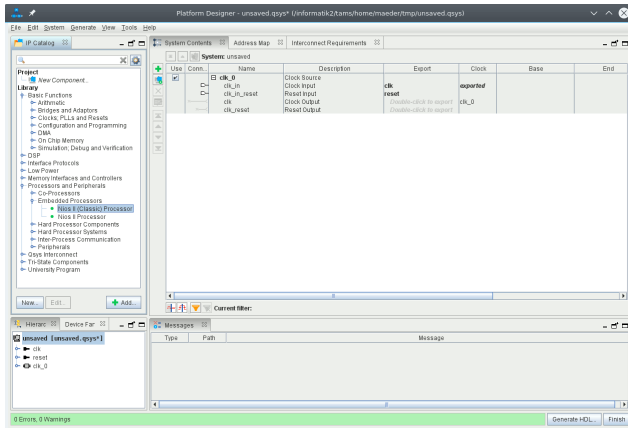
Übungen: Hardwareplattformen (cont.)

FPGA-Prototypensystem

Organisatorisches - Übungen

64-210 Eingebettete Systeme

► IP-Komponenten





Übungen: Hardwareplattformen (cont.)

FPGA-Prototypensystem

Organisatorisches - Übungen

64-210 Eingebettete Systeme

► NIOS Prozessor

The screenshot shows the Nios II (Classic) Processor configuration tool. On the left, the Block Diagram shows the nios2_qsys_0 block with its internal components: clock, reset, interrupt, and nios2_qsys_0. The nios2_qsys_0 block is connected to the data_master, instruction_master, tag_debug_module_reset, custom_instruction_master, and nios2_qsys_0. The right pane shows the 'Select a Nios II Core' dialog. It includes a table comparing Nios II/i, Nios II/s, and Nios II/t cores. Below the table, there are sections for Hardware Arithmetic Operation, Reset Vector, Exception Vector, and MMU and MPU settings.

	Nios II/i	Nios II/s	Nios II/t
Nios II 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. Strata I/O)	Two MBs (or eqv.)	Two MBs + cache	Three MBs + cache

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

Reset Vector
Reset vector memory: None
Reset vector offset: 0x00000000
Reset vector: 0x00000000

Exception Vector
Exception vector memory: None
Exception vector offset: 0x00000020
Exception vector: 0x00000000

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: 0x00000000
Fast TLB Miss Exception vector: 0x00000000

Errors and Warnings:
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.

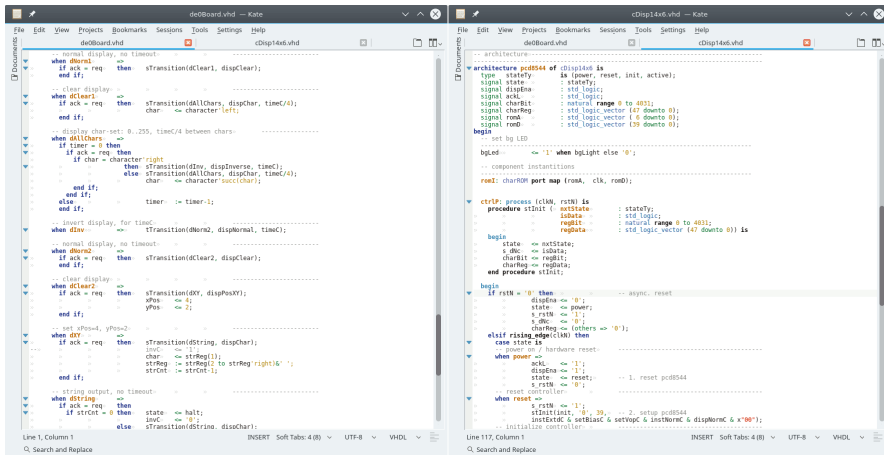
Übungen: Hardwareplattformen (cont.)

FPGA-Prototypensystem

Organisatorisches - Übungen

64-210 Eingebettete Systeme

► VHDL-Code



```
-- normal display, no timeout:
when dNorm1 =>
  if ack = req then sTransition(dClear1, dispClear);
end if;

-- clear display:
when dClear1 =>
  if ack = req 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 =>
  sTransition(dNorm2, dispNormal, timeC);

-- normal display, no timeout:
when dNorm2 =>
  if ack = req then sTransition(dClear2, dispClear);
end if;

-- clear display:
when dClear2 =>
  if ack = req then sTransition(dXY, dispPosXY);
  xPos <= 4;
  yPos <= 2;
end if;

-- set xPos=4, yPos=2:
when dXY =>
  if ack = req then sTransition(dString, dispChar);
  invC <= '1';
  char <= strReg(1);
  strReg := strReg(2 to strReg'right)&' ';
  strCnt := strCnt-1;
end if;

-- string output, no timeout:
when dString =>
  if ack = req then
    if strCnt = 0 then state <= halt;
    invC <= '0';
  else
    sTransition(dString, dispChar);
  end if;
end if;

-- halt:
when dHalt =>
  if ack = req then state <= halt;
end if;
```

```
-- architecture
architecture pc8544 of cDisp14x6 is
  type stateTy is (power, reset, init, active);
  signal state : stateTy;
  signal dispEna : std_logic;
  signal ackNL : 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 romB : std_logic_vector (39 downto 0);
begin
  -- set by LED
  bglEd <= '1' when bgLght else '0';

  -- component instantiations
  rom1: charROM port map (romA, clk, romB);

  ctrlP: process (clk, rstN) is
    procedure stInit (
      axSt : stateTy;
      isData : std_logic;
      regBit : natural range 0 to 4031;
      regData : std_logic_vector (47 downto 0) ) is
    begin
      state <= axSt;
      s.dnc <= isData;
      charBit <= regBit;
      charReg <= regData;
    end procedure stInit;

  begin
    if rstN = '0' then -- async. reset
      dispEna <= '0';
      state <= power;
      s.rstN <= '1';
      s.dnc <= '0';
      charReg <= (others => '0');
    elsif rising_edge(clk) then
      case state is
        -- power on / hardware reset
        when power =>
          ack <= '1';
          dispEna <= '1';
          state <= reset; -- 1. reset pc8544
          s.rstN <= '0';
        -- reset controller
        when reset =>
          s.rstN <= '1';
          s.initInit, '0', 39, -- 2. setup pc8544
            instExtC & setBiasC & setVopC & instNormC & dispNormC & x'00';
          -- initialize controller
      end case;
    end if;
  end process ctrlP;
end architecture pc8544;
```



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

Problem: unterschiedliches Vorwissen!

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



Dr. Andreas Mäder	E-Mail	<code>andreas.maeder@uni-hamburg.de</code>
	Tel.	+49 40 42883 2502
	Büro	Informatikum, Haus F 317

Übungen

Dr. Norman Hendrich	E-Mail	<code>norman.hendrich@uni-hamburg.de</code>
	Tel.	+49 40 42883 2399
	Büro	Informatikum, Haus F 314

Niklas Fiedler	E-Mail	<code>niklas.fiedler@uni-hamburg.de</code>
	Tel.	+49 40 42883 2440
	Büro	Informatikum, Haus F 331



[Mar21] P. Marwedel:

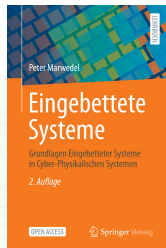
Eingebettete Systeme – Grundlagen eingebetteter Systeme in Cyber-Physikalischen Systemen.

2. Auflage, Springer-Verlag, 2021. ISBN 978-3-658-33437-6

Primärliteratur: Komplette Behandlung des Themas. Buch und Foliensatz sind Vorlage dieser Vorlesung.

Originalmaterial der Vorlesung von Peter Marwedel: daes.cs.tu-dortmund.de/embedded-system-text-book/downloads

eBook: link.springer.com/book/10.1007/978-3-658-33437-6



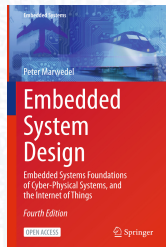
[Mar21e] P. Marwedel:

Embedded System Design – Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things.

4th edition, Springer-Verlag, 2021. ISBN 978-3-030-60910-8

Primärliteratur: die englische Originalversion.

eBook: link.springer.com/book/10.1007/978-3-030-60910-8





[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: link.springer.com/book/10.1007/978-3-540-46824-0

[HT10] C. Haubelt, J. Teich:

Digitale Hardware/Software-Systeme – Spezifikation und Verifikation.

Springer-Verlag, 2010.

ISBN 978-3-642-05356-6

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

eBook: link.springer.com/book/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.

[Brä22] T. Bräunl:

Embedded Robotics – From Mobile Robots to Autonomous Vehicles with Raspberry Pi and Arduino.

4th edition, Springer-Verlag, 2022.

ISBN 978-981-16-0804-9

Eigentlich ein „Robotik“-Buch, zeigt aber sehr schön den Zusammenhang mit „Eingebetteten Systemen“, passend zu den Übungen

eBook: link.springer.com/book/10.1007/978-981-16-0804-9

[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.

2nd edition, CRC Press, 2017.

ISBN 978-1-138-07406-4

Schwerpunkt auf PIC μ 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

[Zöb20] D. Zöbel:

Echtzeitsysteme – Grundlagen der Planung.

2. Auflage, Springer-Verlag, 2020.

ISBN 978-3-662-60421-2

eBook: link.springer.com/book/10.1007/978-3-662-60421-2



[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.

[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

7. Auflage: Ende Mai

[PH22] D.A. Patterson, J.L. Hennessy:

*Rechnerorganisation und Rechnerentwurf – Die Hardware/Software-Schnittstelle
– MIPS Edition.*

6. Auflage, De Gruyter Oldenbourg, 2022.

ISBN 978-3-11-075598-5

Schönes Lehrbuch von den Entwicklern der RISC/MIPS Prozessoren.

[PH20] D.A. Patterson, J.L. Hennessy:

*Computer Organization and Design – The Hardware Software Interface
– RISC-V Edition.*

2nd edition, Morgan Kaufmann Publishers Inc., 2020. ISBN 978-0-12-820331-6

... auch mit RISC-V Befehlssatz (oder als „ARM Edition“).



[MJW20] M. Margolis, B. Jepson, N. R. Weldin:

Arduino Cookbook – Recipes to Begin, Expand, and Enhance Your Projects.

3rd edition, O'Reilly, 2020.

ISBN 978-1-4919-0352-0

Die Arduino Referenz

[McR13] M. McRoberts:

Beginning Arduino.

Apress, 2013.

ISBN 978-1-4302-5016-6

Praxisnahe Beispiele und Projekte

eBook: link.springer.com/book/10.1007/978-1-4302-5017-3

