



V1.0.20240922

Embedded Software Engineering 1

HS 2024 – Prof. Reto Bonderer
Autoren: Laurin Heitzer, Simone Stitz
<https://github.com/P4ntomime/EmbSW1>

Inhaltsverzeichnis

1	Embedded Systems – Allgemein	2			
1.1	Definition	2	1.5	Bare Metal Embedded System	2
1.2	Beispiele	2	1.6	Zuverlässigkeit	2
1.3	Deeply Embedded System	2	1.7	Verfügbarkeit	2
1.4	Betriebssysteme bei Embedded Systems	2	1.8	Abstraktionsschichten	2
			2	Real-Time System (Echtzeitsystem)	2

1 Embedded Systems – Allgemein

1.1 Definition

Ein Embedded System...

- ist ein System, das einen Computer beinhaltet, selbst aber kein Computer ist
- besteht üblicherweise aus Hardware (Mechanik, Elektronik) und Software
- ist sehr häufig ein Control System (Steuerung, Regelung)

Ein Embedded System beinhaltet typischerweise folgende Komponenten:

- Sensoren
- Mikrocomputer
- Hardware (Mechanik, Elektronik)
- Aktoren
- Software (Firmware)

1.1.1 Charakterisierung von Embedded Systems

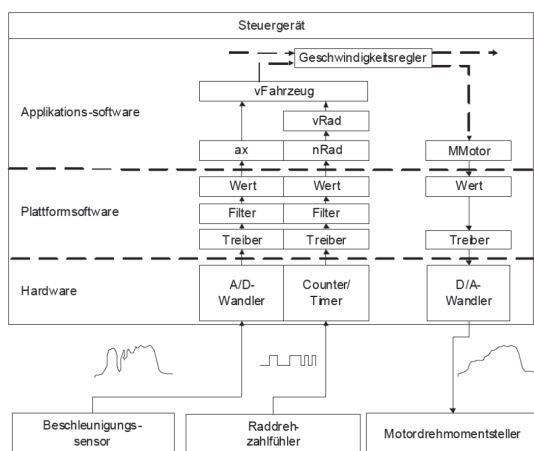
Embedded Systems können (**müssen aber nicht**) folgende Eigenschaften haben:

- **reactive systems:** Reaktive Systeme interagieren mit ihrer Umgebung
- **real-time systems:** Echtzeitsysteme haben neben funktionale Anforderungen auch definierbaren zeitlichen Anforderungen zu genügen
- **dependable systems:** Verlässliche Systeme sind Systeme, welche (sehr) hohe Zuverlässigkeitsänderungen erfüllen müssen
- **Weitere (häufige) Anforderungen:**
 - kleiner Energieverbrauch
 - kleine physikalische Abmessungen
 - Lärm, Vibration, etc.

1.1.2 Typischer Aufbau

Ein gutes Design beinhaltet unterschiedliche Abstraktionsschichten \Rightarrow Layer

\Rightarrow Siehe Abschnitt 1.8



1.2 Beispiele

Fahrrad-Computer

- GPS-Navigation
- Geschwindigkeits- und Trittfrequenzmessung
- Pulsmesser
- Drahtlosübertragung (ANT+)
- Interface zu elektronischer Gangschaltung
- Barometer, Thermometer
- Trainingsassistent
- Display

Weitere Beispiele

- Smartphone
- Mobile Base Station
- CNC-Bearbeitungszentrum
- Hörgerät

Auto

- Sicherheitsrelevante Aufgaben
 - ABS, ASR
 - Motorenregelung
 - Drive-by-wire
 - Autonom fahrende Autos
 - Unterhaltung / Komfort
 - Radio / CD / etc.
 - Navigation
 - Klima
 - Mehrere Netzwerke
 - CAN, LIN, Ethernet
 - Echtzeitteile und andere
 - Von einfachsten μ Cs bis DSPs und GPUs
- \Rightarrow Auto ist ein riesiges Embedded System

1.3 Deeply Embedded System

- 'Einfaches' Embedded System, mit **minimaler Benutzerschnittstelle**, üblicherweise mit **keinerlei GUI** und **ohne Betriebssystem**
- Beschränkt auf **eine** Aufgabe (z.B. Regelung eines physikalischen Prozesses)
- Muss oft zeitliche Bedingungen erfüllen \Rightarrow Echtzeitsystem

1.3.1 Beispiele – Deeply Embedded System

- Hörgerät
- ABS-Controller
- etc...
- Motorenregelung
- 'Sensor' im IoT

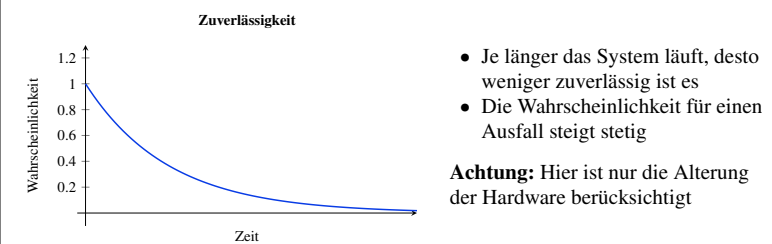
1.4 Betriebssysteme bei Embedded Systems

- Es kommen Betriebssysteme wie (Embedded) Linux oder Android zum Einsatz
 \Rightarrow **Achtung: Linux und Android sind nicht echtzeitfähig!**
- Wenn Echtzeit verlangt wird: real-time operating systems (RTOS)
 - Beispiele: Zephyr, Free RTOS (Amazon), TI-RTOS (Texas Instruments), etc.

1.5 Bare Metal Embedded System

- Es kommt **keinerlei Betriebssystem** zum Einsatz
- Bare Metal Embedded Systems sind recht **häufig**, insbesondere bei **Deeply Embedded Systems**
- Bare Metal Embedded Systems stellen besondere Ansprüche an Programmierung

1.6 Zuverlässigkeit



1.7 Verfügbarkeit

Die Verfügbarkeit A (Availability) ist der Anteil der Betriebsdauer innerhalb dessen das System seine Funktion erfüllt.

$$\text{Verfügbarkeit} = \frac{\text{Gesamtzeit} - \text{Ausfallzeit}}{\text{Gesamtzeit}}$$

1.8 Abstraktionsschichten

- Bei μ C-Programmierung (Firmware) müssen oft Bitmuster in Register geschrieben werden
- Solche Register-Zugriffe dürfen **nicht** 'willkürlich' überall im Code erfolgen
 \Rightarrow schlecht lesbar, schlecht portierbar, fehleranfällig
- **Damit Code lesbarer und besser auf andere Plattform portierbar wird, beinhaltet jeder professionelle Code einen Hardware Abstraction Layer (HAL)**
- HAL führt **nicht** zum Verlust bei Laufzeit, wenn korrekt implementiert

1.8.1 Hardware-abstraction-layer (HAL)

- Trennt HW-Implementierung von SW-Logik
- Gleiche SW kann auf verschiedene HW verwendet werden \Rightarrow Portabilität
- HW-Komponenten können einfach ausgetauscht werden \Rightarrow Flexibilität

2 Real-Time System (Echtzeitsystem)