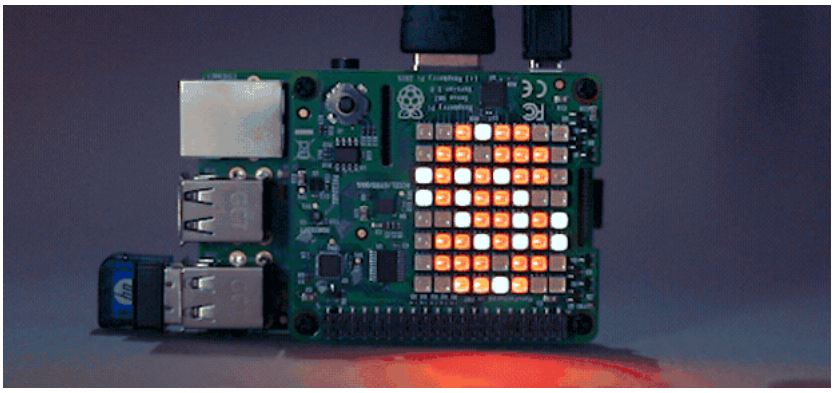
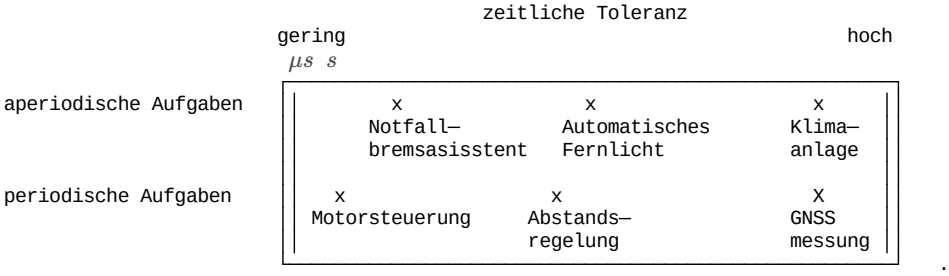
Scheduling

Parameter	Kursinformationen
Veranstaltung:	`Vorlesung Digitale Systeme`
Semester	`Sommersemester 2022`
Hochschule:	`Technische Universität Freiberg`
Inhalte:	`Konzepte eingebetteter Betriebssysteme`
Link auf den GitHub:	https://github.com/TUBAF-Ifi-LiaScript/VL_DigitaleSysteme/blob/main/lectures/07_RTOS_Konzepte.md
Autoren	Sebastian Zug, Karl Fessel & Andrè Dietrich



Ausgangspunkt und Begriffe

Nehmen wir an, dass wir ein autonom fahrendes Fahrzeug allein mit einem zentralen Rechner betreiben wollen. Welche Aufgaben müssen dabei abgedeckt werden und welche zeitlichen Eigenschaften sind dabei zu berücksichtigen?



Entsprechend gilt es eine beschränkte Zahl von Ressourcen auf eine Vielzahl von Aufgaben unterschiedlicher Priorität abzubilden.

> **Merke:** Echtzeitbetrieb nach DIN 44300 … Ein Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfall nach einer zeitlich zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen.

$$r + \Delta e \leq d$$

Die Endzeit d wird durch die Bereitzeit r und die Zeitdauer der Ausführung Δe unterschritten.

- *Harte Echtzeit* (Rechtzeitigkeit - timeliness) = die Abarbeitung einer Anwendung wird innerhalb eines bestimmten Zeithorizontes umgesetzt
- *Weiche Echtzeit* = es genügt, die Zeitbedingungen für den überwiegenden Teil der Fälle zu erfüllen, geringfügige Überschreitungen der Zeitbedingungen sind erlaubt

Intuitive Lösung - Nanokernel

Echtzeitimplementierung als „nanokernel“

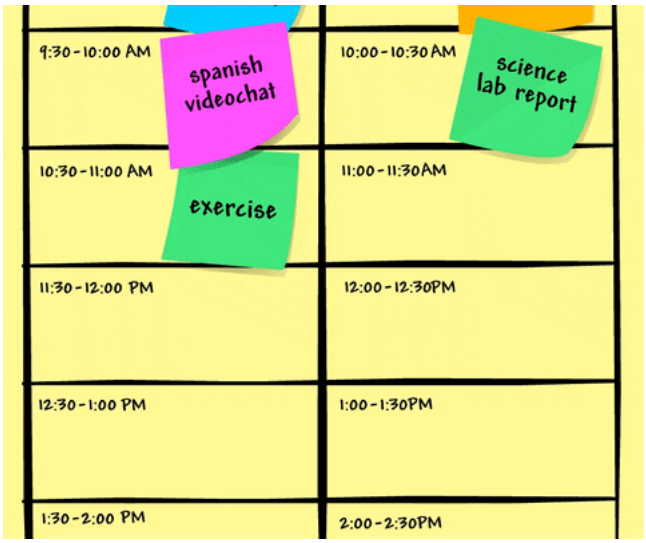
- Ausrichtung an einer minimalen festen Periode p
- Evaluation des Laufzeitverhaltens aller Tasks zwingend notwendig
- keine Schutzfunktionen des Speichers
- Polling als einzige Zugriffsfunktion auf die Hardware

```
switch off interrupts
setup timer
c = 0;
while (1) {
    suspend until timer expires
    c++;
    Task0();    //do tasks due every cycle
    if (((c+0) % 2) == 0)
        Task1(); //do tasks due every 2nd cycle
    if (((c+1) % 3) == 0)
        Task2(); //do tasks due every 3rd cycle, with phase 1
...
}
```

Vorteile	Nachteile
+ Einfache Umsetzbarkeit auf Mikrocontroller	- keine Prioritäten
+ Vereinfachter Hardwarezugriff	- keine Adaption zur Laufzeit
	- keine Interrupts / Unterbrechung
	- beschränkte Wiederverwendbarkeit des Codes
	- implizite Zeitannahmen

Wir benötigen ein systematisches Scheduling, dass eine variable Auswahl der Tasks zulässt!

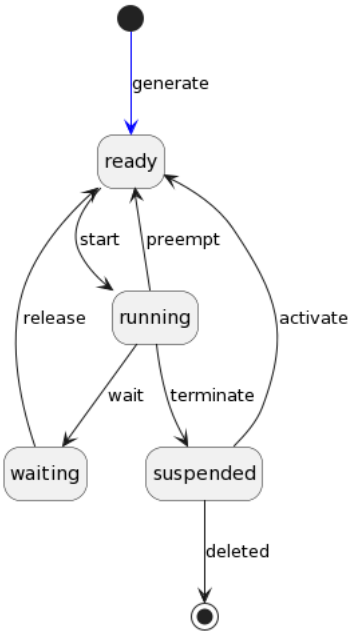
Scheduling Grundlagen



Taskmodel

Eine Task ist die Ausführung eines sequentiellen Programms auf einem Prozessor in seiner spezifischen Umgebung (Kontext). Eine Task...

- erfüllt eine von Programm spezifizierte Aufgabe
- ist Träger der Aktivität = Abstraktion der Rechenzeit die kleinste planbare Einheit



Zustand	Bedeutung
waiting	
ready	Der Scheduler wählt aus der Liste der Tasks denjenigen lauffähigen Prozess als nächstes zur Bearbeitung aus, der die höchste Priorität hat. Mehrere Tasks können sich im Zustand *ready* befinden.
running	In einem Ein-Prozessorsystem kann immer nur ein Task aktiv sein. Er bekommt die die CPU zugeteilt und wird ausgeführt, bis er entweder (1) sich selbst beendet (in den Zustand *suspended*) versetzt, (2) auf ein Betriebsmittel warten muss (z.B. auf das Ende eine I/O-Aufrufes), oder (3) nicht mehr die höchste Priorität hat, da beispielsweise die Wartebedingung eines höherprioren Prozesses erfüllt wurde.
suspended	Ein Task wird in den Zustand *suspended* versetzt, wenn nicht mehr alle Bedingungen zur direkten Ausführung erfüllt sind. Ein Task kann dabei auf unterschiedliche Bedingungen warten, beispielsweise auf das Ende von I/O-Aufrufen, auf den Ablauf einer definierten Zeitspanne oder auf das Freiwerden sonstiger Betriebsmittel.

Aufgabe: Grenzen Sie den Taskbegriff von Prozessen und Threads ab.

Charakterisierung von Tasks

1. **Zeitverhalten**

Periodische Tasks ... werden mit einer bestimmten Frequenz f regelmäßig aktiviert.

- Durchlaufen einer Regelschleife
- Pollendes Abfragen eines Sensors

Aperiodische Tasks ... lassen sich nicht auf ein zeitlich wiederkehrendes Muster abbilden.

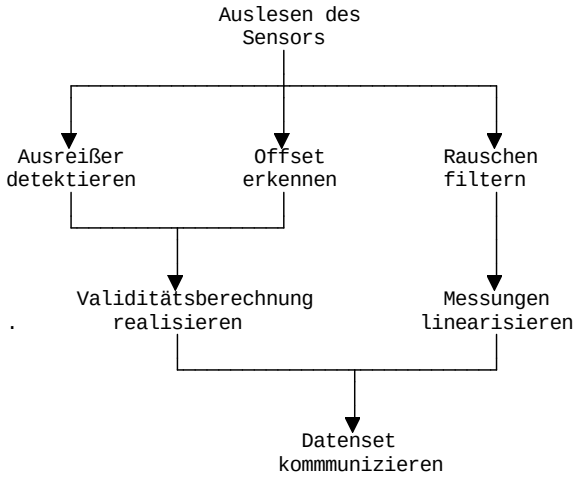
- Tastendruck auf einem Bedienfeld
- Freefall-Detektion

Sporadische Tasks ... treten nicht regulär auf. Man nimmt aber eine obere Schranke bzgl. der Häufigkeit ihres Aufrufs an.

- Fahrradtacho (obere Schranke = Geschwindigkeit)
- Anfragen auf einer Kommunikationsschnittstelle

2. **Abhängigkeiten**

unabhängige Tasks ... können in jeder beliebigen Reihenfolge ausgeführt werden *abhängige Tasks* ... werden durch Vorgänger-Relationen beschrieben und in einem Precedencegraphen dargestellt



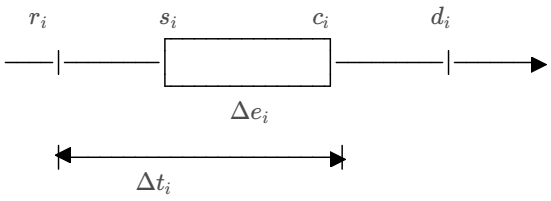
3. **Unterbrechbarkeit**

Unterbrechung einer Taskausführung beim Bereitwerden höherpriorer Tasks

Taskparameter

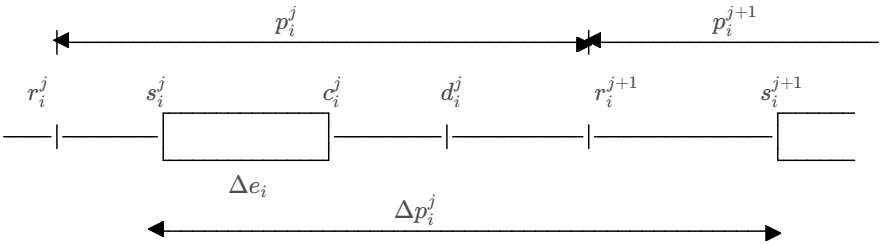
Singuläre Tasks

Parameter	Bedeutung
T	Tasktyp (Abfragen des Temperaturfühlers)
T_i	i-te Instanz des Tasktyp (Taskobjekt)
r_i	Bereitzeit (ready time)
Δe_i	Ausführungszeit (execution time)
s_i	Startzeit (starting time)
c_i	Abschlusszeit (completion time)
d_i	First (deadline)



Periodische Tasks

Parameter	Bedeutung
T_{ij}	j-te Ausführung der i-ten Instanz des Tasktyp (Taskobjekt)
r_i^j	Bereitzeit (ready time)
Δe_i	Ausführungszeit (execution time)
s_i^j	Startzeit (starting time)
c_i^j	Abschlusszeit (completion time)
d_i^j	First (deadline)
Δp_i^j	Zeitspanne zwischen den Startzeiten (Jitter)



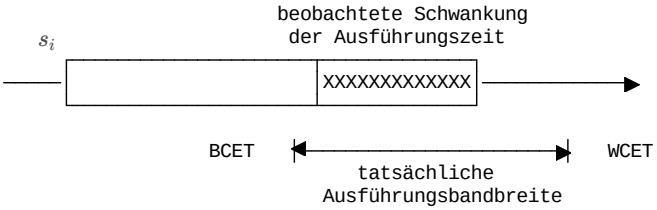
.

Herausforderung Ausführungsdauer Δe_i

Die Ausführungsdauer einer Programmsequenz wird durch

- die Programmlogik (Kontrollflussgraph),
- die Eingabedaten (Auswirkungen auf Schleifendurchlaufzahlen etc.),
- den Compiler (Optimierungsstufe) und
- die Architektur und Taktfrequenz des Ausführungsrechners (Cache- und Pipelining-Effekten)

bestimmt. Da einzelne Aspekte stochastisch verteilt sind, können keine präzisen Aussagen zur Verarbeitungsdauer getroffen werden.



.

Der Lösungsansatz besteht darin mit Hilfe eines entsprechenden Puffers eine untere und obere Schranke zu definieren, die das maximal mögliche Laufverhalten erfasst.

> *BCET* ... Best Case Execution Time
>
> *WCET* ... Worst Case Execution Time

Mögliche Messverfahren sind:

- die Erfassung von oszillierenden Pins
- das automatische Abzählen im Assembler Code
- die Schätzung auf höherabstrakter Codeebene (grafische Modellierungstools)

Ablauf der Vorhersage des Laufzeitverhaltens von Simulink Modellen (Quelle Masterarbeit Markus Wolf)

Ergebnis im Vergleich der Periodendauer von unterschiedlichen Modellen auf einem PPC750GL

Modellgröße	Schätzung	Laufzeitmessung	Differenz
160	10818	9816	10,2%
366	1762	1583	11,4%
756	13721	14946	-8,2%

Scheduling als NP Vollständiges Problem

Allgemeine Formulierung des Schedulingproblems:

Gegeben seien:

- eine Menge von Tasks $T = T_1, T_2, ..., T_n$
- eine Menge von Prozessoren $P = P_1, P_2, ..., P_m$
- eine Menge von Ressourcen $R = R_1, R_2, ..., R_s$

Scheduling bedeutet die Zuordnung von Prozessoren und Ressourcen zu Tasks, so dass alle für individuelle Tasks definierten Beschränkungen eingehalten werden.

> **Merke:** In seiner allgemeinen Form ist das Scheduling-Problem NP-vollständig. Dabei gelten folgende Annahmen:
>
> - gleiche Bereitzeiten aller Tasks,
> - keine Abhängigkeiten,
> - nur feste Prioritäten, ...

Dafür lassen sich unterschiedliche Optimierungskriterien definieren:

- Mittlere Antwortzeit $t_r = 1/n \sum_{(i=1,...,n)}(c_i - r_i)$
- Maximale Zeit bis zum Abschluss $t_c = max(c_i - r_i)$
- Maximale Auslastung des Systems $r = \sum_{(i=1,...,n)} \Delta e_i/t$
- ...

Diese Metriken sind auf Nicht-Echtzeitscheduling ausgerichtet und können nicht übernommen werden weil:

- keine Deadlines und
- keine unterschiedlichen Prioritäten der Tasks (Fairness) berücksichtigt werden
- Kurze Reaktionszeiten genügen nicht, Zeiten müssen garantiert sein
- keine weiteren Parameter wie Periodizität oder Abhängigkeiten abgebildet werden

> **Merke:** Echtzeit braucht die Evaluation der Deadlines. Entsprechend muss für die Latency gelten $L_{max} = 0$

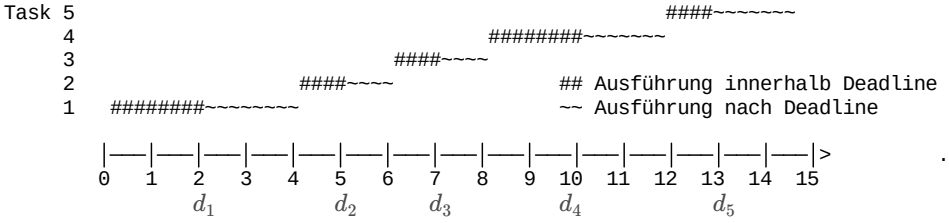
Weiche Echtzeitkriterien lassen eine Abweichung von dieser Vorgabe zu:

> Maximale Zahl von verspäteten Tasks

> Maximale Verspätung $L_{max} = max(c_i - d_i)$

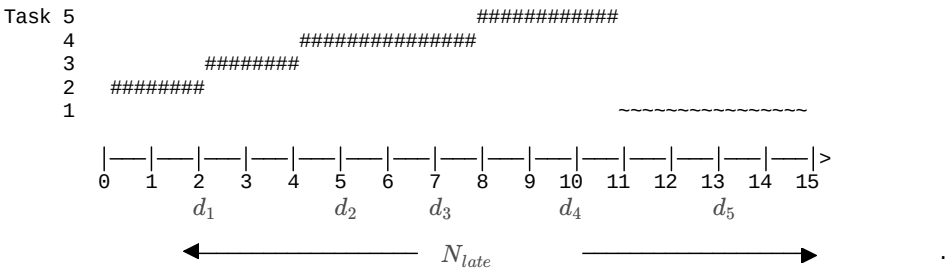
Bereitzeit für alle Tasks $T_1 - T_5$ = 0

Optimierung hinsichtlich der maximalen Verzögerung



$L_{max} = L_1 = L_4 = L_5 = 2, N_{late} = 5$

Optimierung hinsichtlich der Zahl der verzögerten Tasks



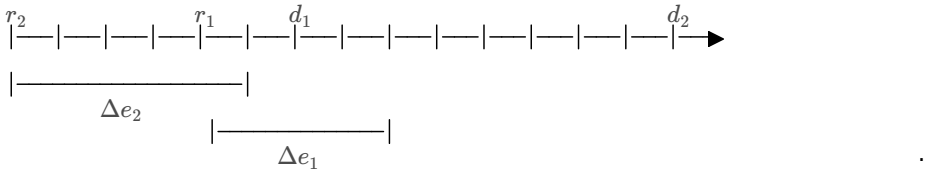
$L_{max} = L_1 = 13, N_{late} = 1$

Einplanbarkeitsanalyse

1. **Korrekte Konfiguration der Tasks**

Überlegung: Die Deadline muss hinreichend weit von der Startzeit entfernt sein

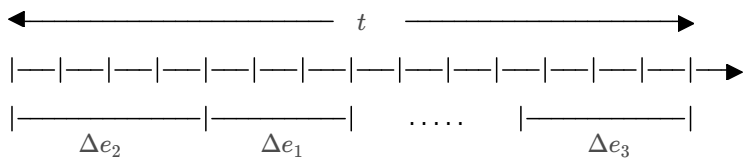
$\Delta e_i \leq (d_i - r_i) \leq p_i$



2. **Verfügbare (Prozessor-) Zeit**

Überlegung: Wir können nicht mehr Performance abfordern, als überhaupt vorhanden ist.

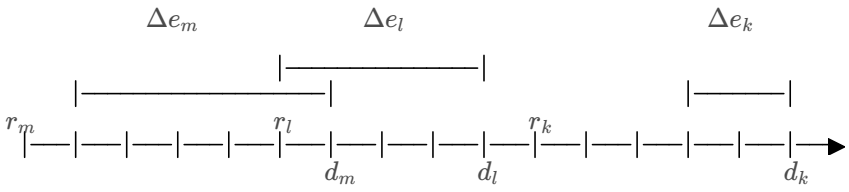
$\sum \Delta e_i \cdot f_i \leq P_{CPU}$



3. **Überlappende Tasks**

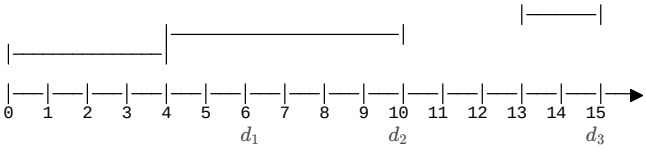
Überlegung: Die letztmögliche Ausführungszeit zweier Tasks überlappt nicht.

$d_i \leq d_j - \Delta e_j$

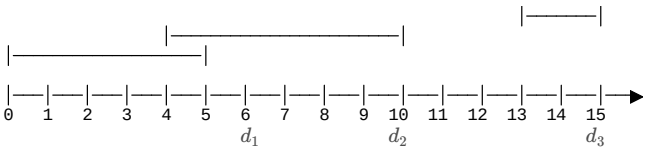


Achtung: Das Scheitern des Tests schließt die Existenz eines gültigen Schedules nicht aus! In beiden nachfolgenden Fällen scheitert das Kriterium!

$e_1 = 4, r_1 = 0, d_1 = 6$
$e_2 = 6, r_2 = 4, d_2 = 10$
$e_3 = 2, r_3 = 13, d_3 = 15$

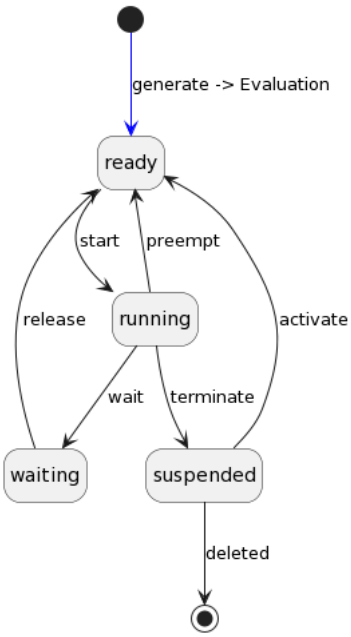


$e_1 = 5, r_1 = 0, d_1 = 6$
$e_2 = 6, r_2 = 4, d_2 = 10$
$e_3 = 2, r_3 = 13, d_3 = 15$



Kriterium	Bedeutung	Aussage
$e_i \leq (d_i - r_i) \leq p_i$	notwendige Ausführungsdauer	notwendig
$\sum \Delta e_i \cdot f_i \leq P_{CPU}$	verfügbare Rechenzeit	notwendig
$d_i \leq d_j - e_j$	Überlappung	hinreichend

Wann findet die Prüfung dieser Kriterien statt? Innerhalb unseres Taskmodell werden die Parametersets unmittelbar nach der Erzeugung untersucht.



Klassifikation Scheduling Verfahren

Statische Verfahren (offline-scheduling):

- Analyse der Durchführbarkeit zur Entwicklungszeit
- Fester Plan, wann welche Task beginnt (Task-Beschreibungs-Liste, TDL)
- Planung für periodische Tasks basierend auf Superperiode
- unflexibel gegenüber Änderungen
- maximale Auslastung stets erreichbar
- kaum Aufwand zur Laufzeit

Zeit	Aktion	WCET
10	starte T1	16
15	sende M5	
26	stoppe T1	
33	starte T2	37
...		

Nicht adaptive Verfahren

- Offline Analyse der Task-Laufzeit
- Online Berechnung von Scheduling-Plänen mit festen Prioritäten

Adaptive Verfahren

- Online Analyse der Task-Laufzeit; Schätzung der Ausführungszeit
- Online Erstellung des Plans unter variabler Priorisierung
- Fehlertoleranz notwendig, da keine Garantie gegeben werden kann

Zusammenfassung