Modulbeschrieb

Eingebettete Echt-Zeit Software

Allgemeine Informationen

Anzahl ECTS-Credits

3

Modulkürzel

TSM_EmbReal

Version

30. August 2009

Modulverantwortliche/r

Hans Buchmann, FHNW

Sprache

	Lausanne	Bern	Zürich
Unterricht	☒ E ☒ F	☐ D ☐ E ☐ F	☒ D ☒ E
Unterlagen	☒ E ☐ F	☐ D ☐ E ☐ F	☐ D ☒ E
Prüfung	☒ E ☒ F	☐ D ☐ E ☐ F	☒ D ☒ E

Modulkategorie

☐ Erweiterte theoretische Grundlagen
☒ Technisch-wissenschaftliche Vertiefung
☐ Kontextmodule

Lektionen

☒ 2 Vorlesungslektionen und 1 Übungslektion pro Woche
☐ 2 Vorlesungslektionen pro Woche

Kurzbeschreibung /Absicht und Inhalt des Moduls in einigen Sätzen erklären

Das Modul „Embedded Real-Time Software“ vermittelt die wichtigsten Grundlagen und Konzepte im Zusammenhang mit der Entwicklung von Applikationen für Embedded Systems. Behandelt werden Implementierungen mit oder ohne Echtzeit-Betriebssystem. Auch bei der Unterstützung durch IO-Services wird auf unterschiedlichen Plattform-Support eingegangen. Das Modul widmet sich den wesentlichen Schritten der Embedded Software-Entwicklung, von der Problemanalyse und vom Design über die Software-Modellierung und Implementierung bis hin zu Einsatz und Unterhalt.

Ziele, Inhalt und Methoden

Lernziele, zu erwerbende Kompetenzen

Nach Beendigung dieses Moduls sind die Studierenden in der Lage
verschiedene Architekturen von Embedded Systems zu beschreiben
verschiedene Arten synchroner und asynchroner Interaktionen zwischen Hardware und Software zu erkennen
ausgehend von den Software-Anforderungen mögliche Software-Architekturen und Interface-Spezifikationen herzuleiten
das Design der Software unter Berücksichtigung eingeschränkter Hardware-Ressourcen und Echtzeit-Anforderungen durchzuführen
zu verstehen, wie Verhaltensmodelle geprüft und implementiert werden
Grundlagen und Verhalten eines Echtzeit-Betriebssystems zu erklären
Eigenschaften, Vor- und Nachteile unterschiedlicher Real-Time Scheduling-Verfahren zu diskutieren
die Vor- und Nachteile aktueller Betriebssysteme zu diskutieren und je nach Anwendung ein geeignetes Betriebssystem auszuwählen
zu erklären, wie Programmierformalismen und Modelling Frameworks in unterschiedlichen Umgebungen eingesetzt werden (mit/ohne Betriebssystem)
Anwendungen zu programmieren, die auf die Hardware zugreifen (Treiber, Bootstrap Code u. a.)
mit Hilfe von Debuggern und Monitoringtools in Anwendungen Fehler zu finden
Anwendungen von einer Architektur auf eine andere zu portieren
robuste Programme zu schreiben

Modulinhalt mit Gewichtung der Lehrinhalte

Anwendungsentwicklung
- Problemanalyse:

- Bereichsausrichtung, Prozesse und Baugruppen, Funktionen und Antwortzeiten, Bereichs-bezogene Trennung von Belangen, generische Softwarearchitektur und Entwicklungsprozess.

- Modellierung reaktiver Komponenten:
  - Modell-basierte Abstraktion, Formalismus und Methoden, interagierende Zustandsautomaten, Architektur und Verhalten, synchroner und asynchroner Aufbau, Übereinstimmung im Verhalten, Modellüberprüfung und Codeerzeugung.
- Anschluss der Ein-/Ausgänge reaktiver Komponenten:
  - Wechselwirkung von externen Prozessen und reaktiven Komponenten, Spezifikation von Wechselwirkungsschichten, Wechselwirkungsfunktionen als Services zwischen den Schichten, Zyklus-basierte Ausführung von Wechselwirkungsfunktionen.
- Konzepte der Echtzeit-Ablaufplanung:
  - statische und dynamische Ablaufplanungsalgorhythmen, Prioritäten und Termine (deadlines), Zyklus-basierte Ausführungsarchitektur, Echtzeit-Taskserver, Hierarchien der Ablaufplanung, Überprüfung der Zulässigkeit (schedulability).

Ausführungsumgebungen von Echtzeit-Systemen

- Nebenläufigkeit
- Thread vs. EventDriven
- Echtzeitbetriebssystem (Real-Time Operating Systems, RTOS)
- RTOS-Architekturen und Subsysteme, Kernel-/User-Space und Syscalls, Speichermanagement, Task-Scheduling, Intertask-Kommunikation und Synchronisierung, Deadlock, RTOS Marktübersicht
- Multicore und Virtualisierung
- Prozessorarchitekturen und Multicore, SMP, Hyperthreading, Virtual Machine Monitor, Full- und Para-Virtualization, Adressraum und Treiberisolierung
- Umgang mit Fehlern

- Interaktion Hardware / Software
  - Systemarchitektur
  - *Architekturen, endlicher Zustandsautomat, Mono- und Multiprozessoren, Koprozessoren, symmetrisches Multi-Processing, non-uniform Memory Access*
  - Hardware Design mit Echtzeit-Anforderungen
  - *Bus-Kommunikation, Speicherarchitektur, Speicherhierarchie, Cache, Memory Management Unit, DMA, Interrupt Controller*
  - Software-Zugriff auf Hardware-Ressourcen
  - *Board Support Package, Bootloader, System Startup, Hardware Debugging, JTAG, Embedded File Systems (Flash, RAM u. a.)*
  - Treiber
- *Treiberstrukturen, Treiberklassifikation (Zeichen-, Block-, Netzwerktreiber u. a.), Prozessor-Mode Levels, Austausch zwischen Mode Levels, Treibermodelle, Major und Minor Numbers, Nebenläufigkeit im Kernel, Kommunikation zwischen User- und Kernel-Space, Interrupt Service Rouines (ISR) und verzögerte Verarbeitung*

Lehr- und Lernmethoden

- Vorlesungen
- Übungen
- Selbststudium (anhand von Unterlagen und Fallstudien)

Voraussetzungen, Vorkenntnisse, Eingangskompetenzen

Die Studierenden haben Grundkenntnisse in

- Programmierung
- Computerarchitekturen
- Betriebssystemen

Bibliografie

- Burns, A. Wellings. Real-Time Systems and Programming Languages. Addison-Wesley, third edition, 2001.
- F. Cottet, J. Delacroix, C. Kaiser, Z. Mammeri. /Scheduling in Real-Time Systems/. Wiley, 2002.
- G. C. Buttazzo. Hard Real-Time Computing Systems. Predictable Scheduling Algorithms and Applications. Second Edition, Kluwer Academic Publishers, 2005.
- L. Zaffalon. Programmation synchrone de systèmes réactifs avec Esterel et les SyncCharts. PPUR, 2005. (English Version planned for late 2008)

- Colin Walls. Embedded Software: The Works. Newnes, 2006
- Jack Ganssle. The Firmware Handbook. Newnes, 2004

Leistungsbewertung

Zulassungsbedingungen für die Modulschlussprüfung (Testatbedingungen)

keine

Schriftliche Modulschlussprüfung

Prüfungsdauer :	120 Minuten
Erlaubte Hilfsmittel:	Eigene Zusammenfassung