# Modulhandbuch

# Master -Studiengang Angewandte Informatik / Infotronik

Fakultät Elektrotechnik, Medientechnik und Informatik

Prüfungsordnung 01.10.2015

Stand: Dienstag 19.02.2019 09:49

# 0 - 01 THEORETISCHE INFORMATIK

| | |
|---|---|
| Modul Nr. | 0 - 01 |
| Modulverantwortliche/r | Prof. Dr. Peter Faber |
| Kursnummer und Kursname | 0 - 01 - 1 Semantik, Berechenbarkeit und Komplexitätstheorie<br>0 - 01 - 2 Formale Sprachen und Compilerbau I |
| Lehrende | Prof. Dr. Peter Faber<br>Prof. Dr. Peter Jüttner |
| Semester | 1 |
| Dauer des Moduls | 1 Semester |
| Häufigkeit des Moduls | jährlich |
| Art der Lehrveranstaltungen | Pflichtfach |
| Niveau | |
| SWS | 6 |
| ECTS | 8 |
| Workload | Präsenzzeit: 90 Stunden<br>Selbststudium: 180 Stunden<br>Gesamt: 270 Stunden |
| Prüfungsarten | schr. P. 90 Min. |
| Dauer der Modulprüfung | 90 Min. |
| Unterrichts-/Lehrsprache | Englisch |

## Qualifikationsziele des Moduls

Die Theoretische Informatik liefert die fundamentalen Ideen und Berechnungsmodelle, die jedem Datenverarbeitungssystem zugrundeliegen.

Ihr Verständnis ist darum für ein Verständnis der Verifizierbarkeit und damit der Sicherheit, Korrektheit und Stabilität eines Systems essenziell.

In diesem Modul lernen die Studenten die theoretischen Grundlagen der Informatik wie Maschinenmodelle mit Anwendungen in Komplexitätstheorie und Berechenbarkeitstheorie kennen, formale Sprachen und ihre Hierarchien etc. Sie gewinnen dabei ein Verständnis für Komplexitätsabschätzungen und Verifikationsmöglichkeiten, sowie die Grundfunktionen von Datenverarbeitungssystemen.

## Zugangs- bzw. empfohlene Voraussetzungen

keine

## Inhalt

Das Modul besteht aus:

- Formale Sprachen und Compilerbau I: Diese Veranstaltung beleuchtet den theoretischen Hintergrund hinter formalen Sprachen und Compiler-Frontends.
- Semantik, Berechenbarkeit und Komplexitätstheorie: Hier werden theoretische (Maschinen-) Modelle zur formalen Definition von Semantik, zur Analyse von Berechenbarkeit und Komplexitätsabschätzung herangezogen.

Beide Fächer sind eng miteinander verwandt und ergänzen sich untereinander.

## 0 - 01 - 1 SEMANTIK, BERECHENBARKEIT UND KOMPLEXITÄTSTHEORIE

### Ziele

Ziel des Fachs ist es, formale Ansätze der Informatik in den Themenbereichen Sematik, Berechenbarkeit und Komplexitätstheorie zu vermitteln.
Fachliche Kompetenz:

- Anwendung formaler Berechnung der Semantik rekursiver Funktionen
- Anwendung unterschiedlicher Induktionsverfahren zum Beweis von Programmeigenschaften
- Anwendung der operativen und axiomatischen Semantik zum Beweis bestimmter Eigenschaften von Programmen.
- Anwendung unterschiedlicher Modelle der Berechenbarkeit
- Kenntnis der Berechnung der Komplexität verschiedener Problemklassen und Anwendung der daraus abzuleitenden Folgerungen für die Programmierung

Methodische Kompetenz

- Anwendung mathematischer Beweisverfahren

### Inhalt

- Semantik
  - Definition Semantik, Geschichte
  - Semantik Rekursiver Funktionen (Fixpunkttheorie)
  - Induktionsbeweise
  - Operative Semantik
  - Axiomatische Semantik

- Berechenbarkeit
  - Definition
  - Nicht berechenbare Funktionen
  - Turing Maschinen und ihre Programmierung
  - Turing-Berechenbarkeit
  - LOOP-, WHILE-, GOTO-Berechenbarkeit
- Komplexitätstheorie
  - Definition
  - O-Notation
  - Komplexitätsniveaus

## Zugangs- bzw. empfohlene Voraussetzungen

- Programmieren in einer höheren Programmiersprache (z.B. C, C++, Java, C#)
- Mathematik der natürlichen Zahlen (Induktion)
- Grundlagen der Aussagen- und Prädikatenlogik

## Prüfungsarten

Teil der Modulprüfung

## Methoden

Seminaristischer Unterricht, praktische Übungen

## Empfohlene Literaturliste


- John Longley, Lessons in „Formal Programming Language Semantics", University of Edinburgh, 2003
- F.L. Bauer, H. Wössner: Algorithmische Sprache und Programmentwicklung, Springer Verlag 1984
  (availble also in English)
- Rudolf Berghammer: Semantik von Programmiersprachen, Logos Verlag, 2001
- Juraj Hromkovic: Theoretische Informatik, Springer Verlag
- Uwe Schöning: Theoretische Informatik - kurz gefasst. Spektrum, 2008

- Hopcroft, Motwani, Ullman: Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 2001
- Hopcroft, Motwani, Ullman: Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie, Pearson, 2002.

## 0 - 01 - 2 FORMALE SPRACHEN UND COMPILERBAU I

### Ziele

Im Fach "Formale Sprachen und Compilerbau I" lernen die Studenten die formalen Sprachen (Chomsky-Hierarchie etc.) kennen. Sie beherrschen Begriffe und Konstruktionen aus der Automatentheorie und können entsprechende Beweise zu Sprachinklusionen, Berechenbarkeit etc. nachvollziehen und verstehen. Sie entwerfen selbst kleine Grammatiken und setzen syntaxgesteuerte Übersetzungsschemata um. Sie können die Mächtigkeit von Automaten einordnen und ggf. begründen. Der Einsatz dieser theoretischen Grundlagen wird anhand von Compilern für Programmiersprachen dargestellt.

Auf fachlicher Ebene kennen und verstehen die Studenten die informationstechnischen Grundlagen für Berechenbarkeit und Entscheidbarkeit und sind in der Lage, Beweise nachzuvollziehen, selbst zu führen. Auf fachlicher und persönlicher Ebene sind Sie in der Lage, aufgrund dieser analytischen Methodik Techniken zu vergleichen und zu verifizieren, Aussagen über die Lösbarkeit eines Problems zu treffen. Die hierzu nötige strukturierte und analytische Vorgehensweise haben die Studenten verinnerlicht.

### Inhalt

- Einführung / Übersetzer I -- Introduction and Translators I
- Übersetzer II / Formale Sprachen I -- Translators II / Formal languages I
- Formale Sprachen II / III -- Formal languages II / III
- Lexikalische Analyse I / II -- Lexical Analysis I / II
- Syntaktische Analyse I / II -- Syntax analysis I / II
- Syntaktische Analyse III / IV -- Syntax analysis III / IV
- Syntaxgesteuerte Übersetzung I / II -- Syntax directed translation I/II
- Zusammenfassung, ggf. weitere Thematiken (z.B. zum optimierenden Compilerbau) -- Wrap-up, possibly further topics (e.g., optimizing compilers)

### Zugangs- bzw. empfohlene Voraussetzungen

Programmierkenntnisse (idealerweise C) wünschenswert; UNIX-Kenntnisse vorteilhaft.

## Prüfungsarten

schr. P. 90 Min.

## Methoden

Seminaristischer Unterricht mit praktischem Anteil. Virtuelle Einheiten über Videoaufnahmen möglich (deutsch).

## Empfohlene Literaturliste

- Compilers – Principles, Techniques, and Tools; Aho, Lam, Sethi, Ullmann; 2nd edition; Addison-Wesley; 2007
- Engineering a compiler; Cooper, Torczon; 2nd Edition, Morgan Kaufmann 2012
- Introduction to Automata Theory, Languages, and Comoputation; Hopcroft, Motwani, Ullman; Addison-Wesley; 2001
- ggf. weitere Literatur nach Angabe des Dozenten

# O - 02 PRAKTISCHE INFORMATIK

| | |
|---|---|
| Modul Nr. | O - 02 |
| Modulverantwortliche/r | Prof. Dr. Peter Jüttner |
| Kursnummer und Kursname | 0 - 02 - 1 Advanced Software Engineering<br>0 - 02 - 2 Formale Sprachen und Compilerbau II<br>Programmierpraktikum |
| Lehrende | Prof. Dr. Peter Faber<br>Prof. Dr. Peter Jüttner |
| Semester | 1 |
| Dauer des Moduls | 1 Semester |
| Häufigkeit des Moduls | jährlich |
| Art der Lehrveranstaltungen | Pflichtfach |
| Niveau | |
| SWS | 6 |
| ECTS | 8 |
| Workload | Präsenzzeit: 90 Stunden<br>Selbststudium: 165 Stunden<br>Gesamt: 255 Stunden |
| Unterrichts-/Lehrsprache | Englisch |

## Qualifikationsziele des Moduls

Die Praktische Informatik führt die Studenten an die praktische Anwendung der theoretisch fundierten Techniken heran. Sie kennen Entwicklungsmethoden un- Werkzeuge genauso wie Entwicklungsprozesse eines Systems und können Ihre Kenntnisse mithilfe dieser Werkzeuge in der Praxis umsetzen.

## Zugangs- bzw. empfohlene Voraussetzungen

keine

## Inhalt

Das Modul besteht aus:

- Advanced Software Engineering: Hier erlernen die Studenten spezielle Techniken und Vorgehensweisem des Software-Engineerings.
- Programmierpraktikum: Hier setzen die Studenten Ihre Software-Engineering-Fähigeiten in einem echten kleinen Projekt -- i.d.R. in Teamarbeit -- in die Realität um.
- Formale Sprachen und Compilerbau II: Diese Veranstaltung beleuchtet die praxisrelevanten Aspekte wie das Backend eines Compilers mit Optimierungstechniken u.ä.

## 0 - 02 - 1 ADVANCED SOFTWARE ENGINEERING

### Ziele

Vertiefung ausgewählter, wichtiger Themen des Software Engieering.

Fachliche Kompetenz:

- Studenten kennen die agile Methode Scrum und können diese anwenden.
- Studenten kennen ausgewählte UML Diagramme in Theorie und Praxis
- Studenten können Reviewmethoden auf Entwicklungsergebnisse und Projekte anwenden.
- Studenten kennen die Besonderheiten des Tests objektorientierter Software.

### Inhalt

- Einleitung
- Agile Methoden allgemein
- Agile Methoden - Scrum
- UML allgemein
- Ausgewählte UML-Diagramme in Theorie und Praxis
- Software Reviewtechniken, Intensivreviews
- Besonderheiten des Tests objektorientierter Software

### Zugangs- bzw. empfohlene Voraussetzungen

- Kenntnisse der objektorientierten Programmierung.
- Grundkenntnisse im Bereich Software Engineering

### Prüfungsarten

schr. P. 90 Min.

### Methoden

Seminaristischer Unterricht mit praktischen Übungen, teilweise Gruppenarbeit, Workshops

### Empfohlene Literaturliste

- Peter Hruschka, Chris Rupp: Agile Softwareentwicklung mit der UML, Hanser Verlag, 2002
- Chris Rupp et. al: UML 2 Glasklar, Hanser Verlag, 2007
- Software Inspection, Tom Gilb and Dorothy Graham, Addison Wesley, 1993


## 0 - 02 - 2 FORMALE SPRACHEN UND COMPILERBAU II

### Ziele

Im Fach "Formale Sprachen und Compilerbau II" die Anwendung der formalen Sprachen (Chomsky-Hierarchie, Automatentheorie etc.) in der Technik kennen. Sie wenden Begriffe und Konstruktionen aus der Automatentheorie an, insbesondere im Compilerbau. Zudem lernen sie die Grundlagen des Compiler-Backends und der optimierenden Compilation. Sie verstehen die grundlegenden Techniken der Codegenerierung bis hin zur Code-Optimierung und können diese Techniken prototypisch einsetzen. Hierbei werden Compiler für Programmiersprachen dargestellt in der praktischen Anwendung betrachtet.

Fachlich kennen die Studenten prinzipielle Compilertechnologien und verstehen (In-)Effizienzen in Programmiersprachenkonstrukten. Zudem gewinnen sie fachlich Einblick in Optimierungsmethoden. Persönlich und fachlich gewinnen die Studenten Einblick in Arbeitsweisen des Programmierens, Strukturen der Programmierung und Programmiermethodiken.

### Inhalt

- Zwischencode-Generierung I -- IR generation I
- Zwischencode-Generierung II -- IR generation II
- Laufzeit-Umgebungen I -- RTEs I
- Laufzeit-Umgebungen II -- RTEs II
- Codegenerierung I -- Code generation I
- Codegenerierung II -- Code generation II
- Zusammenfassung, ggf. weitere Thematiken (z.B. zum optimierenden Compilerbau) -- Wrap-up, possibly further topics (e.g., optimizing compilers)

### Zugangs- bzw. empfohlene Voraussetzungen

"Formale Sprachen und Compilerbau I" (gleichzeitiger Besuch) wünschenswert; Programmierkenntnisse (idealerweise C) wünschenswert; UNIX-Kenntnisse vorteilhaft.

### Prüfungsarten

schr. P. 90 Min.

### Methoden

Seminaristischer Unterricht mit praktischem Anteil. Virtuelle Einheiten über Videoaufnahmen möglich (deutsch).

### Empfohlene Literaturliste

- Compilers – Principles, Techniques, and Tools; Aho, Lam, Sethi, Ullmann; 2nd edition; Addison-Wesley; 2007
- Engineering a compiler; Cooper, Torczon; 2nd Edition, Morgan Kaufmann 2012
- ggf. weitere Literatur nach Angabe des Dozenten

## PROGRAMMIERPRAKTIKUM

### Ziele

Die Studenten bearbeiten (i.d.R. in Teams) eine aktuelle Programmieraufgabe. Die Studierenden entwickeln
dabei eigene Lösungen für ein vorgegebenes Problem wie es im Einsatz in einem Softwarehaus geschehen
könnte. Auftraggeber und Kunde bzw. Schnittstelle zu diesen werden dabei vom Dozenten simuliert.

Persönlich und fachlich verstehen die Studenten einen Entwicklungsprozess. Fachlich haben sie entsprechende Werkzeuge genutzt und ihre Funktion verstanden. Auf persönlicher und sozialer Ebene haben sie die Herausforderungen eines Entwicklungsprozesses (idealerweise unter wechselnden Anforderungen und Problemstellungen, wie heutzutage häufig anzutreffen).

### Inhalt

- Projektvorstellung
- Bedarfsgerecht: Einführung in Techniken der Informatik und Programmierung
- Projektmeetings
- Ergebnispräsentation

### Zugangs- bzw. empfohlene Voraussetzungen

- Vorlesungen (Bachelor):
  - Grundlagen der Informatik
  - Einführung in die Programmierung

- Software-Engineering

- Gleichzeitiger Besuch des Moduls "Theoretische Informatik" sowie der anderen Veranstaltungen aus "Praktische Informatik"
- Kenntnisse in Programmierung und Softwareentwicklung

## Prüfungsarten

schr. P. 90 Min.

## Methoden

Die Studenten analysieren ein vorab vom Dozenten gegebenes Problem, entwickeln eigene Lösungsansätze
und implementieren diese.
Mit dem Dozenten werden je nach Aufgabenstellung Feedback-Schleifen vereinbart.
Unterstützung durch E-
Learning-System.

## Empfohlene Literaturliste

Literatur nach Angabe des Dozenten

# O - 03 AUSGEWÄHLTE THEMEN DER EMBEDDED SOFTWARE ENTWICKLUNG I

| | |
|---|---|
| Modul Nr. | O - 03 |
| Modulverantwortliche/r | Prof. Dr. Andreas Grzemba |
| Kursnummer und Kursname | O-03-1 Embedded Connectivity<br>O-03-2 Embedded Security |
| Lehrende | Prof. Dr. Andreas Grzemba<br>Prof. Dr. Martin Schramm |
| Semester | 1 |
| Dauer des Moduls | 1 Semester |
| Häufigkeit des Moduls | jährlich |
| Art der Lehrveranstaltungen | FWP, Pflichtfach |
| Niveau | |
| SWS | 4 |
| ECTS | 5 |
| Workload | Präsenzzeit: 60 Stunden<br>Selbststudium: 120 Stunden<br>Gesamt: 180 Stunden |
| Unterrichts-/Lehrsprache | Englisch |

## Zugangs- bzw. empfohlene Voraussetzungen

keine

## Inhalt

Studenten kennen wesentliche Kommunikationsprinzipen für Eingebettete Systems am Beispiel der KFZ-Kommunikation kennen. Es werden grundlegenden Eigenschaften Vermittlung von KFZ-Kommunikationssystemen vermittelt

Die Studierenden erlernen die wesentlichen Prinzipien der M2M-Kommunikation in automotiven Applikationen, um diese Bewertung und einsetzen zu können.

Die Studierenden erwerben die Fähigkeit, die Netzwerkkomponenten für komplexer verteilte Regel- und Steuerungssysteme bewerten und anwenden zu können

Die Studierenden erlernen die wesentlichen Prinzipien sicherer Programmierpraktiken und können diese umsetzen. Die Studierenden erwerben die Fähigkeit, gegebene Softwarekomponenten gezielt auf Schwachstellen hin zu analysieren, Risiken zu beurteilen, und eigenständig Lösungen für sichere Software-Komponenten zu erstellen.

## O-03-1 EMBEDDED CONNECTIVITY

## Ziele

Studenten lernen wesentliche Kommunikationssysteme für Eingebettete Software am Beispiel der KFZ-Kommunikation kennen.

## Inhalt

- Anwendungsbereiche von M2M-Kommunikation in Automotive
- Arten, Grundprinzipien und Auswahlkriterien von Kommunikationssystemen wie Zentrale Gateway und Switched Ethernet Architekturen
- Auslegung von Kommunikationssystemen
- Engineering und Betrieb von Kommunikationssystemen
- Bewertung des CAN-Protokolls
- Bestandteile und Standards von Automotive Ethernet
- Aufbau, Betrieb und Test eines automotiven Ethernet-Netzwerks

## Zugangs- bzw. empfohlene Voraussetzungen

keine

## Prüfungsarten

PstA

## Methoden

- Seminaristischer Unterricht mit praktischen Übungen, teilweise Gruppenarbeit

## Empfohlene Literaturliste


- Tanenbaum, A., Wetherall, D.: Computer Networks, 5. Auflage, 2011, Prentice Hall, ISBN 978-0-13-212695-3
- Glover, I.; Grant, P., Digital Communications, 2. Auflage, Prentice Hall, ISBN 0-13-089399-4
- Matheus, K., Königseder, T.: Automotive Ethernet, 2014, Cambridge University Press, ISBN 978-1107057289
- Marco Di Natale,Haibo Zeng,Paolo Giusto,Arkadeb Ghosal: Understanding and Using the Controller Area Network Communication Protocol, 2012, Springer, eISBN 978-1-4614-0314-2


## O-03-2 EMBEDDED SECURITY

## Ziele

Die Studierenden kennen die gängigen Kategoriesysteme für Software-Schwächen, Schwachstellen, und Risiken, und erwerben Wissen über Angriffsvektoren und Angriffsstrategien. Die Studierenden erlernen die wesentlichen Prinzipien sicherer Programmierpraktiken mittels sicherer Kodierungsstandards und können diese umsetzen. Die Studierenden erwerben die Fähigkeit, gegebene Softwarekomponenten gezielt auf Schwachstellen hin zu analysieren, Risiken zu beurteilen, und eigenständig Lösungen für sichere Software-Komponenten zu erstellen.

## Inhalt

- Einführung und Motivation
  - Gründe unsicherer Softwarekomponenten
  - Sicherer Software Development Life Cycle (SDLC)
  - Cybersecurity Framework
  - Terminologie
  - TOP 10 der sicheren Programmierpraktiken
- Software-Schwachstellen und -Verwundbarkeiten
  - Common Weakness Enumeration (CWE)
  - Common Vulnerabilities and Exposures (CVE) - Repository
  - TOP 25 der gefährlichsten Software Fehler
  - OWASP Top 10 der kritischen Web-Applikation Sicherheitsrisiken
- Eingabevalidierung / Gültigkeitsprüfung
  - Angriffsfläche
  - Eingabequellen
  - Whitelist- und Blacklist-Ansätze
  - Eingabevalidierung von Zahlen
  - Eingabevalidierung von Strings
    - Kodierungsverfahren
    - Regular Expressions
- Systemnahe Angriffsvektoren

- Pufferüberläufe
- Mehrfache Deallokation
- Hash-Kollisionsangriffe

- Interaktion mit weiteren Softwarekomponenten
  - Injection-Angriffe
    - SQL-Injection
    - CSV-Injection
    - Kommando-Injection
- Entwurf sicherer Softwarekomponenten
  - Entwurfsrichtlinien
  - Angriffs- / Bedrohungsmodellierung
- Sichere Kodierungsstandards
  - SEI CERT coding standards
  - OWASP Secure Coding Practices
- Statische und dynamische Code-Analyse-Tools

Fuzzing und Penetration-Testing

## Zugangs- bzw. empfohlene Voraussetzungen

keine

## Prüfungsarten

PstA

## Methoden

- Seminaristischer Unterricht mit praktischen Übungen, teilweise Gruppenarbeit

## Empfohlene Literaturliste


- Secure Programming for Linux and Unix HOWTO - Creating Secure Software, v3.72, 2015-09-19
- Robert C. Seacord, „Secure Coding in C and C++“, Addison-Wesley Professional, ISBN: 978-0321822130

- Robert C. Seacord, „CERT® C Coding Standard, Second Edition, The: 98 Rules for Developing Safe, Reliable, and Secure Systems“, Addison-Wesley Professional, ISBN: 978-0321984043
- Brian Chess, Jacob West, „Secure Programming with Static Analysis“, Addison-Wesley Professional, ISBN: 978-0321424778


# O - 04 AUSGEWÄHLTE THEMEN DER EMBEDDED SOFTWARE ENTWICKLUNG II

| | |
|---|---|
| Modul Nr. | O - 04 |
| Modulverantwortliche/r | Prof. Dr. Martin Schramm |
| Kursnummer und Kursname | Ausgewählte Themen der Embedded Software Entwicklung II |
| Semester | 1 |
| Dauer des Moduls | 1 Semester |
| Häufigkeit des Moduls | jährlich |
| Art der Lehrveranstaltungen | Pflichtfach |
| Niveau | |
| SWS | 4 |
| ECTS | 5 |
| Workload | Präsenzzeit: 60 Stunden<br>Selbststudium: 90 Stunden<br>Gesamt: 150 Stunden |
| Prüfungsarten | Klausur |
| Unterrichts-/Lehrsprache | Deutsch |

## Qualifikationsziele des Moduls

Die Studierenden sollen ausgewählte Themen des eingebetteten Software Projektmanagements vertieft kennen lernen und die entsprechenden Methoden sowohl in der wissenschaftlichen, als auch industriellen Praxis anwenden können. Die Studierenden erlernen die grundlegenden Konzepte der Versionsverwaltung und des Variantenmanagements und können diese bei der Umsetzung eines eigenständigen Projekts im Team anwenden. Weiterhin sind die Studierenden in der Lage, eine Aufwandsabschätzung für ein Projekt der Software-Entwicklung durchzuführen. Die Studierenden erwerben die Fähigkeit, Projektportfolio-Management Tools sinnvoll für die Softwareentwicklung einzusetzen, sowie die Kompetenz, kollaborativ die Projektdokumentation zu steuern und durchzuführen.

## Zugangs- bzw. empfohlene Voraussetzungen

- Grundkenntnisse im Bereich Programmierung
- Grundkenntnisse im Bereich Software Engineering

## Inhalt

- Versionsverwaltung und Variantenmanagement
  - Grundlegende Konzepte
  - Zentrale und Verteilte Systeme

- Versionsverwaltung im Team
- Repositories und Branches
- Umsetzung am Beispiel von Git

- Aufwandsabschätzung (Softwaretechnik)
  - Grundlagen der Aufwandsschätzung
  - Aufwandsschätzung am Beispiel von Scrum
    - Planspiele
    - Zeitplanung
  - Einfache Schätztechniken
    - Teile-und-herrsche-Verfahren
    - Analogieschlüsse
    - Expertenschätzungen
    - Zwei- und Drei-Punkt-Schätzungen
    - Fehlerrechnungen
  - Umsetzung am Beispiel von Planning Poker
- Projektportfolio-Management Tools
  - Multiprojektmanangement
  - Meilensteine und Tasks
  - Umsetzung am Beispiel von OrangeScrum
- Kollaborative Dokumentenbearbeitung
  - Dokumentation in Teams

Umsetzung am Beispiel von ShareLatex

## Lehr- und Lernmethoden

- Seminaristischer Unterricht mit praktischen Übungen, Gruppenarbeit

## Empfohlene Literaturliste

- René Preißel, Bjørn Stachmann, „Git: Dezentrale Versionsverwaltung im Team – Grundlagen und Workflows“, dpunkt Verlag, ISBN: 978-3864904523

- Stefan Luckhaus, „Aufwandsschätzungen in der agilen Softwareentwicklung: Einsatz von Methoden zur Messung des funktionalen Umfangs“, Tredition Verlag, ISBN: 978-3732365944
- Benjamin Michels, „Projektmanagement Handbuch 3 - Verschiedene Projekte gleichzeitig leiten & steuern“, CreateSpace Independent Publishing Platform, ISBN: 978-1545335482
- Herbert Voß, „Einführung in LaTeX: unter Berücksichtigung von pdfLaTeX, XLaTeX und LuaLaTeX“, Lehmanns Verlag, ISBN: 978-3865417985


## AUSGEWÄHLTE THEMEN DER EMBEDDED SOFTWARE ENTWICKLUNG II

### Prüfungsarten

schr. P. 90 Min.

# O - 05 SPEZIELLE MATHEMATISCHE METHODEN

| | |
|---|---|
| Modul Nr. | O - 05 |
| Modulverantwortliche/r | Prof. Dr. Johann Plankl |
| Kursnummer und Kursname | Spezielle Mathematische Methoden |
| Semester | 2 |
| Dauer des Moduls | 1 Semester |
| Häufigkeit des Moduls | jährlich |
| Art der Lehrveranstaltungen | Pflichtfach |
| Niveau | |
| SWS | 4 |
| ECTS | 5 |
| Workload | Präsenzzeit: 60 Stunden<br>Selbststudium: 90 Stunden<br>Gesamt: 150 Stunden |
| Unterrichts-/Lehrsprache | Englisch |

## Zugangs- bzw. empfohlene Voraussetzungen

keine

## Inhalt

Das Modul entspricht dem namensgleichen Modul aus dem Studiengang Master Elektro- und Informationstechnik.
Die Modulbeschreibung ist aus dem dortigen Handbuch zu entnehmen.

## SPEZIELLE MATHEMATISCHE METHODEN

### Prüfungsarten

schr. P. 90 Min.

# O - 06 BIS O - 10 WAHLMODULE

| | |
|---|---|
| Modul Nr. | O - 06 bis O - 10 |
| Modulverantwortliche/r | Prof. Dr. Peter Jüttner |
| Kursnummer und Kursname | Wahlmodule |
| Semester | 1 |
| Dauer des Moduls | 1 Semester |
| Häufigkeit des Moduls | jährlich |
| Art der Lehrveranstaltungen | FWP |
| Niveau | |
| SWS | 4 |
| ECTS | 5 |
| Workload | Präsenzzeit: 60 Stunden<br>Selbststudium: 120 Stunden<br>Gesamt: 180 Stunden |
| Unterrichts-/Lehrsprache | Deutsch |

## Qualifikationsziele des Moduls

Durch Wahlmodule sollen die Studierenden die Möglichkeit erhalten, ihre Kenntnisses gemäß eigenen Interessen in Richtung Elektronik oder Medieninformatik zu vertiefen. Darüber hinaus bieten die freie Wahl von bis zu zwei Fächern den Studierenden die Gelegenheit ihre Informatikkenntnisse zu ergänzen bzw. zu erweitern.

## Zugangs- bzw. empfohlene Voraussetzungen

keine

## Inhalt

Wahlmodule aus den Studiengängen Master Elektro- und Informationstechnik, Master Medientechnik, zusätzlich optionale Wahlfächer FWP.

Aus den aufgeführten Fächern der Studiengänge Master Elektro und Informationstechnik (ET) und Master Medientechnik (MT) müssen mindestens 3, maximal 5 Fächer ausgewählt werden. Werden weniger als 5 ausgewählt, so sind weitere einschlägige Wahlfächer mit mindestens 5 ECTS aus dem Angebot der Hochschule zu wählen, so dass insgesamt 5 Wahlfächer belegt werden.

Folgende Fächer aus dem Studiengang Master ET stehen zur Auswahl:
- Ausgewählte Kapitel der Mikro und Nanoelektronik
- Systeme der Hochfrequenz und Funktechnik
- Spezielle Bauelemente und Schaltungen
- Signale und Systeme der Nachrichtentechnik
- Ausgewählte Themen der berührungslosen Sensorik

- Automobile und industrielle elektrische Antriebssysteme
- Regenerative Energien

Folgende Fächer aus dem Studiengang Master ET stehen zur Auswahl:
- 3D Computeranimation
- Computervision
- Industrielle Bildverarbeitung
- Cyber Security
- Applikationsdesign
- Moderne Internettechnologien

Die Beschreibungen dieser Module sind den Modulhandbüchern der genannten Studiengänge zu entnehmen. Die Module aus dem Studiengang Master ET werden durchgängig in Englisch, Module aus dem Master MT in Deutsch angeboten. Die Module aus Master ET und Master MT finden im Wintersemester statt, FWP Module können auch im Sommersemester belegt werden.
Die angegebenen Werte für SWS und ECTS sind Mindestwerte bei der FWP Belegung.

## WAHLMODULE

### Prüfungsarten

schr. P. 90 Min.

# O - 11 FPGA PROGRAMMIERUNG

| | |
|---|---|
| Modul Nr. | O - 11 |
| Modulverantwortliche/r | Prof. Dr. Martin Schramm |
| Kursnummer und Kursname | FPGA Programmierung |
| Lehrende | Prof. Dr. Martin Schramm |
| Semester | 1 |
| Dauer des Moduls | 1 Semester |
| Häufigkeit des Moduls | jährlich |
| Art der Lehrveranstaltungen | Pflichtfach |
| SWS | 4 |
| ECTS | 5 |
| Workload | Präsenzzeit: 60 Stunden<br>Selbststudium: 90 Stunden<br>Gesamt: 150 Stunden |
| Unterrichts-/Lehrsprache | Englisch |

## Qualifikationsziele des Moduls

Die Studierenden lernen die wesentilchen Prizipien des FGGA Hardware Entwurfs mittels VHDL theretisch und in praktischen Beispielen kennen und können diese im beruflichen und wissenschaftlichen Umfeld anwenden.

## FPGA PROGRAMMIERUNG

### Ziele

Die Studierenden lernen die wesentilchen Prizipien des FGGA Hardware Entwurfs mittels VHDL theretisch und in praktischen Beispielen kennen und können diese im beruflichen und wissenschaftlichen Umfeld anwenden.

### Inhalt

- Einführung und Motivation
- Modellierung digitaler Systeme mit VHDL
- Grundlegende Konzepte von VHDL
- Verhaltens und Strukturbeschreibung
- Typkonzept
- sequentielle und nebenläufige Anweisungen
- Prozeduren und Funktionen

- Realisierung digitaler Schaltungen
- Methoden des Hardware Debugging
- Netzlistenanalyse
- Simulation digitaler Entwurfssysteme
- Logikanalyse mittels virtuellen Logikanalysator
- Systementwurf

## Zugangs- bzw. empfohlene Voraussetzungen

- Grundkenntnisse der Informatik, insbesondere der Programmierung
- Kenntnisse in Digitaltechnik
- Kenntnisse in Rechnernetzen
- Kenntnisse der Systemprogrammierung

## Prüfungsarten

schr. P. 90 Min.

## Methoden

Seminaristischer Unterricht mit praktischen Übungen, teilweise Gruppenarbeit

# O - 12 ALLGEMEINWISSENSCHAFTLICHES WAHLPFLICHTFACH (AWP)

| | |
|---|---|
| Modul Nr. | O - 12 |
| Modulverantwortliche/r | Prof. Dr. Peter Jüttner |
| Kursnummer und Kursname | Allgemeinwissenschaftliches Wahlpflichtfach I<br>Allgemeinwissenschaftliches Wahlpflichtfach II |
| Semester | 1 |
| Dauer des Moduls | 1 Semester |
| Häufigkeit des Moduls | jährlich |
| Art der Lehrveranstaltungen | Kern- / Wahlpflichtfach |
| Niveau | Postgraduate |
| SWS | 4 |
| ECTS | 4 |
| Workload | Präsenzzeit: 60 Stunden<br>Selbststudium: 60 Stunden<br>Gesamt: 120 Stunden |
| Unterrichts-/Lehrsprache | Deutsch |

## Qualifikationsziele des Moduls

Studierende vertiefen allgemeinwissenschaftliche Kenntnisse oder Fremdsprachenkenntnisse im Rahmen von 2 Wahlfächern aus dem Angebot des Career Service und des Sprachenzentrums

## Zugangs- bzw. empfohlene Voraussetzungen

gemäß Fächerwahl

## Inhalt

Allgemeinwissenschaftliches Wahlfach aus dem Angebot des Career Service oder Fremdspachenkurs aus dem Angebot des Sprachenzentrums

## Lehr- und Lernmethoden

gemäß Fächerwahl

## Besonderes

Prüfungsart gemäß Fächerwahl

### ALLGEMEINWISSENSCHAFTLICHES WAHLPFLICHTFACH I

### Prüfungsarten

schr. P. 90 Min.

## ALLGEMEINWISSENSCHAFTLICHES WAHLPFLICHTFACH II

### Prüfungsarten

schr. P. 90 Min.

# O - 13 MASTERARBEIT

| | |
|---|---|
| Modul Nr. | O - 13 |
| Modulverantwortliche/r | Prof. Dr. Peter Jüttner |
| Kursnummer und Kursname | Masterarbeit<br>Masterkolloquium |
| Semester | 3 |
| Dauer des Moduls | 1 Semester |
| Häufigkeit des Moduls | jährlich |
| Art der Lehrveranstaltungen | Pflichtfach |
| SWS | 0 |
| ECTS | 23 |
| Workload | Präsenzzeit: 0 Stunden<br>Selbststudium: 760 Stunden<br>Gesamt: 760 Stunden |
| Unterrichts-/Lehrsprache | Deutsch |

## MASTERARBEIT

### Prüfungsarten

Endnotenbildende PStA

## MASTERKOLLOQUIUM

### Prüfungsarten

mündl. Prüf.