Modulbezeichnung	**Praktische Informatik 1**
Modulverantwortliche(r)	Dr. Thomas Röfer
Modulart	Pflicht/Wahl ☒ Wahlpflicht ☐
Spezialisierungsbereich	
Dauer des Moduls	1 Semester
Kreditpunkte	8 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 112 h Übungsbetrieb/Prüfungsvorbereitung 128 h Summe 240 h
Turnus des Moduls	angeboten in jedem WiSe
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☒ Projekt ☐
Lernziele	• Grundlegende Informatikkonzepte wiedergeben und erklären können. • Konzepte einer imperativen Programmiersprache kennen, verstehen und anwenden können. • Anschauliche Sachverhalte im Modell der Objektorientierung ausdrücken können. • Einfache Algorithmen entwickeln und in Java umsetzen können. • Einfache in Java realisierte Algorithmen systematisch testen können. • Probleme in Teilprobleme zerlegen und diese Strukturierung mit Mitteln von Java umsetzen und aussagekräftig dokumentieren können. • Formale Syntaxbeschreibungen verstehen und für einfache Sprachen entwickeln können. • Operationelle Semantik einfacher While-Sprachen verstehen und zum Nachweis einfacher Programmeigenschaften anwenden können • Eine Entwicklungsumgebung nutzen können. • LaTex zur Erstellung einfacher Dokumente nutzen können. • Versionsverwaltungssysteme einsetzen können. • In Gruppen Probleme analysieren und gemeinsam Lösungsstrategien entwickeln und präsentieren können. Die Vorlesungen Praktische Informatik 1 und 2 vermitteln essenzielles Grundwissen und Basisfähigkeiten, deren Beherrschung für nahezu jede vertiefte Beschäftigung mit Informatik – sowohl in der industriellen Anwendung, als auch in der Forschung – Voraussetzung ist.

Lerninhalte	1. Basiswissen: von Neumannsche Rechnerorganisation – Grundlagen der Rechnerarchitektur – Programm und Prozess – Programmiersprachen – Compiler, Assembler, Loader, Linker, Interpreter, Laufzeitumgebungen, Betriebssysteme – Browser – Grafische Benutzungsschnittstellen – Shells 2. Datenstrukturen: Information und ihre Repräsentation – Datentypen und Typanalyse – Elementare und zusammengesetzte Datentypen – rekursive Datentypen – Kanonische Operationen auf den eingeführten Datenstrukturen 3. Algorithmen: Begriff des Algorithmus – Beschreibung von Algorithmen – Algorithmische Umsetzung kanonischer Operationen auf Datenstrukturen – Kontrollstrukturen – Rekursion – Grundlegende Strategien: Greedy-Strategie versus Divide-and-ConquerStrategie 4. Programmierparadigmen: (1) Imperative, funktionale und logische Programmierung, (2) Objektorientierte (imperative) Programmierung, (3) Sequenzielle Programme versus nebenläufige Programme 5. Grundkomponenten imperativer Programmiersprachen: Schnittstellen und Ein-/Ausgabe, Variablen und Zuweisungen, Kontrollstrukturen, Blöcke, Funktionen, Rekursion 6. Syntax und Semantik imperativer Programmiersprachen: Syntax und Methoden der Syntax-Spezifikation, reguläre Ausdrücke, (erweiterte) Backus-Naur-Form (E)BNF, Syntaxgraphen – operationelle Semantik für Zuweisungen und Kontrollstrukturen 7. Prinzipien der objektorientierten Programmierung: Geheimnisprinzip – Methoden – Operationen – Objekte – Klassen – Botschaften – Ereignisverarbeitung – Attribute – Vererbung – Polymorphismus – Overloading 8. Umsetzung der Punkte 2.-7. mit Java – Illustration anhand einfacher Algorithmen 9. Programmdokumentation und zugehörige Hilfswerkzeuge, z.B. JavaDoc – Doxygen 10. Testen von Programmen und zugehörige Hilfswerkzeuge, z.B. JUnit 11. Basisdienste im Internet: telnet, ftp und ihre sicheren Varianten ssh, scp, sftp 12. World-Wide-Web – Grundbegriffe von HTML Programmier-Praktikum: Programmentwicklung in Java – Realisierung einzelner, überschaubarer Programmieraufgaben
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Klausur
Literatur	• G. Saake und K.-U. Sattler: Algorithmen und Datenstrukturen. dpunkt.verlag, Heidelberg (2004) • R. Schiedermeier: Programmieren mit Java. Pearson, München (2005) Weitere Informationen (Beispielprogramme, Musterlösungen, im WWW verfügbare Literatur) sind auf der Web-Seite der Veranstaltung zu finden.

Modulbezeichnung	**Praktische Informatik 2**
Modulverantwortliche(r)	Prof. Dr. J. Peleska
Modulart	Pflicht/Wahl ☒ Wahlpflicht ☐
Spezialisierungsbereich	
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	angeboten in jedem SoSe
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Typische Datenstrukturen identifizieren und problemadäquat einsetzen können. • Datenstrukturen und Algorithmen in Java umsetzen können. • Wesentliche Algorithmen der Informatik erklären, anwenden und modifizieren können. • Algorithmische Alternativen bezüglich der Eignung für ein Problem beurteilen können. • Grundbegriffe der formalen Verifikation erläutern können. • Die Komplexität von einfachen Algorithmen analysieren können. • In Gruppen Probleme analysieren und gemeinsam Lösungsstrategien entwickeln und präsentieren können. Die Vorlesungen Praktische Informatik 1 und 2 vermitteln essenzielles Grundwissen und Basisfähigkeiten, deren Beherrschung für nahezu jede vertiefte Beschäftigung mit Informatik – sowohl in der industriellen Anwendung, als auch in der Forschung – Voraussetzung ist.

Lerninhalte	1. Komplexität von Algorithmen – O(n)-Notation und asymptotische Analyse 2. Suchen und Sortieren auf Arrays: Binäre Suche – Quicksort und weitere Sortieralgorithmen – Komplexitätsvergleiche 3. Mengen – Bags – Multimengen – Relationen – Funktionen: Datenstrukturen und Algorithmen zur Realisierung kanonischer Operationen (z.B. Mengenalgebra) 4. Listen – Stapel – Warteschlangen: Datenstrukturen zur Realisierung (Arrays versus Verkettung und dynamische Speicherallokation für Elemente), Algorithmen zur Realisierung kanonischer Operationen (Listentraversion, Anfügen, Einfügen, Löschen, Suchen, Stack-Operationen, FIFO-Warteschlangenoperationen) 5. Bäume: Binäre Bäume, AVL-Bäume, Rot-Schwarz-Bäume, B-Bäume – Suchen, Einfügen, Löschen, Traversion 6. Hashing: Hash-Array, Hashfunktion, Hash Buckets, offenes Hashing 7. Graphen: ungerichtete, gerichtete, gewichtete Graphen – Repräsentation durch Knoten- und Kantenlisten, durch Adjazenzmatrizen, Adjazenzlisten – Algorithmen auf Graphen: Breitensuche, Tiefensuche, Topologische Sortierung, kürzeste Wege auf gewichteten Graphen: Dijkstras Algorithmus, Maximaler Durchfluss, Realisierung markierter Transitionssysteme mit Graphen 8. Algorithmen zur Syntaxprüfung: Tokenizer und Parser – systematische ParserGenerierung aus EBNF-Grammatiken 9. Textsuche: Knuth-Morris-Pratt – Boyer-Moore – Pattern Matching für reguläre Ausdrücke 10. Spezifikation von Programmen: Vor- und Nachbedingungen – Invarianten 11. Verifikation: Partielle und totale Korrektheit sequenzieller Programme – Formale Verifikation, z.B. Hoare Logik (Pre-/Postconditions) – Eigenschaftsbeweis durch Strukturelle Induktion
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Klausur
Literatur	• G. Saake und K.-U. Sattler: Algorithmen und Datenstrukturen. dpunkt.verlag, Heidelberg (2004) • R. Schiedermeier: Programmieren mit Java. Pearson, München (2005) Weitere Informationen (Beispielprogramme, Musterlösungen, im WWW verfügbare Literatur) sind auf der Web-Seite der Veranstaltung zu finden.

Modulbezeichnung	**Technische Informatik 1**
Modulverantwortliche(r)	Prof. Dr. R. Drechsler
Modulart	Pflicht/Wahl ☒ Wahlpflicht ☐
Spezialisierungsbereich	
Dauer des Moduls	1 Semester
Kreditpunkte	8 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 84 h Übungsbetrieb/Prüfungsvorbereitung 156 h Summe 240 h
Turnus des Moduls	angeboten in jedem SoSe
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Grundlegende Konzepte moderner Rechner wiedergeben und erläutern können • Schaltkreismodellierung durch Boolesche Funktionen verstehen und erklären können • Hardware-Realisierungen von arithmetischen Funktionen darstellen können • Modellierung und Optimierungsansätze integrierter Schaltkreise umreißen können • Rechnersysteme anhand der eingeführten Konzepte selbständig beurteilen können • Unterschiedliche Hardware-Realisierungen unter den eingeführten Optimierungskriterien bewerten können • In Gruppen Probleme analysieren, gemeinsam Lösungsstrategien entwickeln und präsentieren können

Lerninhalte	I. Rechnerarchitektur 1. Rechnersichtweisen: Ebenen und Sprachen, Hierarchie, Compiler, Interpreter 2. Aufbau und Funktionsweise: Hardware, Software, Firmware, Aufbau eines von-Neumann-Rechners, Arbeitsspeicher, Speicherzelle, Arbeitsweise eines Prozessors, Speicher, I/OBusse 3. Befehlssatz: RISC, CISC, Designprinzipien 4. Pipelining 5. Speicher: Hierachie, Organisation, Caches, Hintergrundspeicher 6. Parallelität: Ausprägungen, Klassifikation von parallelen Rechnerarchitekturen, Exkurs über Verbindungsstrukturen II. Digitale Schaltungen: 1. Schaltkreise: Technologien, Definition, Kosten, Semantik von kombinatorischen Schaltkreisen, Simulation, Teilschaltkreise, Hierarchischer Entwurf, Beispiele 2. Kodierung: Zeichen, Zahlen, Zahlensysteme, Übertragung, Fehlerkorrektur, HammingCode, Huffman-Code, Festkommadarstellungen, Zahlendarstellung durch Betrag und Vorzeichen, Einer-/Zweierkomplement-Darstellung, Gleitkommadarstellung (IEEE-754 Format) 3. Boolescher Kalkül: Funktion, Algebra, Ausdrücke, alternative Funktionsdarstellung, z.B. durch Entscheidungsdiagramme 4. Zweistufige Schaltungen: Logiksynthese, Implikanten, Primimplikanten, Minimierung, Quine/McClusky, Überdeckungsproblem 5. Integrierte Schaltungen, arithmetische Schaltungen, ALU 6. Schaltungen mit speichernden Elementen
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben, Präsentation mindestens einer Lösung im Tutorium und Fachgespräch
Literatur	• B. Becker, R. Drechsler, P. Molitor, Technische Informatik – Eine Einführung, Pearson Studium, 2005 • A. S. Tanenbaum, J. Goodman, Computerarchitektur, 4. Aufl., Pearson Studium, 2001 • H. Wuttke, K. Henke, Schaltsysteme, Pearson Studium, 2002 W. Stallings, Computer Organization & Architecture, Prentice Hall, 2002 • C. Siemers, A. Sikora, Taschenbuch Digitaltechnik, Fachbuchverlag Leipzig, 2002 • T. Beierlein, O. Hagenbruch, Taschenbuch Mikroprozessortechnik, Fachbuchverlag Leipzig, 2001 • D. Patterson, J. Hennessy, Computer Organization & Design - The Hardware/Software Interface, Morgan Kaufmann Publishers, 1997

Modulbezeichnung	**Technische Informatik 2**
Modulverantwortliche(r)	Prof. Dr. U. Bormann
Modulart	Pflicht/Wahl ☒ Wahlpflicht ☐
Spezialisierungsbereich	
Dauer des Moduls	1 Semester
Kreditpunkte	8 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 84 h Übungsbetrieb/Prüfungsvorbereitung 156 h Summe 240 h
Turnus des Moduls	angeboten in jedem WiSe
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Praktische Informatik 2, Technische Informatik 1
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• In der Terminologie der Betriebssysteme und nebenläufigen Systeme kommunizieren können. • Abstraktionshierarchien (Speicherverwaltung, Dateisystem) in Bezug auf ihre Auswirkung auf die Systemleistung einschätzen können. • Lösungsvarianten für Systemsoftwarekomponenten und den Umgang mit Nebenläufigkeit bewerten können (s. unten). • Schutzmechanismen in Bezug auf Anwendungssicherheitsziele anwenden können. • Selbständiges Entwickeln von einfachen Systemkomponenten in C++ für Unix. • Die globalen Strategien auf einfache vorgegebene Einzelsituationen übertragen können. • In Gruppen Probleme analysieren, gemeinsam Lösungsstrategien entwickeln und präsentieren können.

Lerninhalte	I. Grundlagen der Betriebssysteme • Betriebssysteme: Aufgaben, Rechnerbetriebsformen und Elemente von Betriebssystemen, Anmerkungen zur Geschichte und Überblick über die Entwicklung der Betriebssysteme • Prozessverwaltung: Einfache Prozesse, Prozesseigenschaften, Unterbrechungen, Systemaufrufe, Ausnahmen, Echtzeitbetrieb • Speicherverwaltung: Ein-/Auslagerungsverfahren • Dateisystem: Namen, Baumstruktur; Zugriffsoperationen; Abbildung auf reale Geräte; Ein/Ausgabe; Sicherheit (Schutzmechanismen, Zugriffsrechte) • Befehlsinterpreter II. Nebenläufigkeit • Synchronisation: Semaphore, (bedingte) kritische Abschnitte, Ereignisse, Monitore, synchroner/asynchroner Nachrichtenaustausch, “Rendezvous”, Kanäle, verteilte Systeme mit Prozedurfernaufrufen • Verklemmungen, Lebendigkeit, Fairness; Korrektheit • Formale Beschreibung nebenläufiger Systeme, z.B. mit Petri-Netzen (Überblick) • Spezielle nebenläufige Systeme: Speisende Philosophen, Erzeuger/Verbraucher, Leser/Schreiber usw. • Grundlagen der Rechnernetze, Client/Server-Architekturen, lokale und globale Netze (Überblick, Ethernet,IP, TCP, HTTP), Sicherheit (Grundlagen der Kryptographie)
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch
Literatur	Andrew S. Tanenbaum: Modern Operating Systems, 4th Edition, Pearson Studium, 2016 (bzw. die deutsche Übersetzung: Moderne Betriebssysteme, 4. Auflage, Pearson Studium, 2016)

Modulbezeichnung	**Wissenschaftliches Arbeiten 1**
Modulverantwortliche(r)	R. E. Streibl
Modulart	Pflicht/Wahl ☒ Wahlpflicht ☐
Spezialisierungsbereich	
Dauer des Moduls	1 Semester
Kreditpunkte	1 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 20 h Übungsbetrieb 10 h Summe 30 h
Turnus des Moduls	angeboten in jedem WiSe als Blockkurs vor Semesterbeginn (alternativ semesterbegleitend)
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☒ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Wesentliche universitäre (Infra)Strukturen kennen. • Grundlegende wissenschaftliche Vorgehensweisen verstehen. • Mit wissenschaftlicher Literatur arbeiten können (Recherche, Umgang mit Quellen, Aufbau wissenschaftlicher Texte). • Arbeitsergebnisse in unterschiedlichen Kontexten präsentieren können. • Erste Erfahrungen mit Referaten im universitären Kontext machen und Ansätze für eine Feedback-Kultur entwickeln. • Fähigkeit zur (interkulturellen) Kooperation ist verbessert. • In Gruppen Probleme analysieren und gemeinsam Lösungsstrategien entwickeln und präsentieren können.

Lerninhalte	1. Problemformulierung und Recherchemethoden (Bibliothek, Internet) 2. Strukturierung und Formulierung im Rahmen wissenschaftlicher Argumentation 3. Aufbau wissenschaftlicher Arbeiten 4. Gestaltung von Präsentationen / Erprobung in Form einer Präsentationswerkstatt mit systematischem Feedback; 5. Ausgewählte Aspekte individuellen (Wahrnehmung, Gedächtnis, Zeitmanagement, . . .) und sozialen Lernens (Gruppenarbeit, Moderation) 6. Einführung in die Lernplattform StudIP, die Rechnerumgebung des Fachbereichs und Grundkenntnisse von La TeX als Hilfsmittel zur Erstellung von wissenschaftlichen Arbeiten Ablauf: Das Modul wird in der Regel als Blockkurs vor Beginn der Lehrveranstaltungen des ersten Semesters angeboten (nur in dringenden Ausnahmefällen sollte auf den semesterbegleitenden Ausweichkurs zurückgegriffen werden). Die Inhalte werden abwechselnd in Vorlesungsform, Seminarform und Gruppenarbeit vermittelt und erarbeitet. Die schriftlichen Übungsaufgaben werden in Arbeitsgruppen bearbeitet (für die erste Aufgabe zufällig zusammengesetzt). Alle TeilnehmerInnen halten im Laufe der Veranstaltung ein fünfminütiges Referat zu einem selbst gewählten Sachthema (aktiv: Erleben der Präsentationssituation, passiv: Entwicklung eines Qualitätsbewusstseins bzgl. Präsentationen und einer Feedbackkultur).
Prüfungsformen	Bearbeitung der Übungsaufgaben, Kurzreferat
Literatur	Einige Literaturempfehlungen (die Bücher sind weitgehend in der SuUB verfügbar sowie im Studienzentrum Informatik einsehbar): ● Sesink,W. (2010): Einführung in das wissenschaftliche Arbeiten. 8.Aufl. München: Oldenbourg. ● Franck, N.; Stary, J. (2009): Die Technik wissenschaftlichen Arbeitens: eine praktische Anleitung. 15.Auflage. Paderborn: Schöningh. – SuUB u.a. 14. Aufl. als eBook verfügbar. ● Eco, U. (2010):Wie man eine wissenschaftliche Abschlussarbeit schreibt. 13.Aufl. Heidelberg: UTB. ● Deininger, M.; Lichter, H.; Ludewig, J.; Schneider, H. (2005): Studien-Arbeiten. Ein Leitfaden zur Vorbereitung, Durchführung und Betreuung von Studien-, Diplom-, Abschluss- und Doktorarbeiten am Beispiel Informatik. 5. Aufl. Zürich: vdf. ● Balzert, H.; Schäfer, Ch.; Schröder, M.; Kern, U. (2008):Wissenschaftliches Arbeiten - Wissenschaft, Quellen, Artefakte, Organisation, Präsentation. Herdecke:W3L. ● Schubert-Henning, S. (2009):Toolbox. Lernkompetenz für erfolgreiches Studieren. Anleitung für ein erfolgreiches Studium: Von der Schule übers Studium zum Beruf. Bielefeld: UVW. ● Kruse, O. (2007): Keine Angst vor dem leeren Blatt: Ohne Schreibblockaden durchs Studium. 12.Aufl. Frankfurt: campus. ● Schlosser, J. (2008):Wissenschaftliche Arbeiten schreiben mit La TeX. Leitfaden für Einsteiger. 2.Aufl. Heidelberg: mitp.

Modulbezeichnung	**Software-Projekt-Vorlesung**
Modulverantwortliche(r)	Prof. Dr. R. Koschke
Modulart	Pflicht/Wahl ☒ Wahlpflicht ☐
Spezialisierungsbereich	
Dauer des Moduls	1 Semester
Kreditpunkte	5 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 70 h Übungsbetrieb/Prüfungsvorbereitung 80 h Summe 150 h
Turnus des Moduls	angeboten in jedem SoSe
Voraussetzung für die Teilnahme	Keine ☒ Folgende Formale Voraussetzungen: Praktische Informatik 1
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Das angestrebte Ergebnis des Moduls insgesamt ist es, dass die Studierenden die methodischen und praktischen Fähigkeiten erwerben, in einer Gruppe eine Software-Lösung für ein vorgegebenes nicht-triviales Problem zu finden und zu realisieren. • Die zu erwerbenden fachlichen Kompetenzen umfassen alle notwendigen Aktivitäten in der Softwareentwicklung von der Anforderungsanalyse und Aufwandsschätzung, über den Architekturentwurf bis zur Implementierung und den Test. Ebenso gehören dazu die begleitenden Managementaspekte der Gruppenarbeit, Entwicklungsprozess, Planung, qualitätssichernde Maßnahmen, die Dokumentation und das Konfigurationsmanagement. • Die zu erwerbenden sozialen Kompetenzen betreffen das Projektmanagement in einem Software-Projekt sowie die Gruppenarbeit über einen längeren Zeitraum und die hierfür notwendige Selbstkompetenz (Zeitmanagement, Übernahme von Verantwortung und mehr).

Lerninhalte	Die folgenden, für ein solches Projekt notwendigen Themen der Softwaretechnik werden in der Vorlesung vermittelt (die Notation UML wird in den entsprechenden Abschnitten als Mittel zum Zweck und im methodischen Zusammenhang eingeführt): Allgemeines • was ist Software? • Eigenschaften von Software • Software-Lebenszyklus • die besondere Bedeutung der Wartung und Evolution • Softwarekrise • was ist Softwaretechnik? Projektplanung • Grundbegriffe der Projektplanung • Vorgehen bei der Planung • Inhalt des Projektplans • Gantt-Diagramme und kritischer Pfad • Projektrisiken • Softwareentwicklungsprozesse Rechtlicher Rahmen der Softwareentwicklung • Betriebsverfassungsgesetz (BetrVG), PersVG • Arbeitsschutzgesetze, Verordnungen (BildscharbV) • Datenschutzgesetze (BDSG) • Normen und Richtlinien Anforderungsanalyse • Probleme bei der Anforderungsanalyse • Schritte der Anforderungsanalyse • Schritte der Ist-Analyse • Erhebungstechniken bei der Ist-Analyse (Fragebögen, Interview im Kontext) und Soll-Analyse (Varianten des Prototypings) • Aufbau und Inhalt der Anforderungsspezifikation • Produktqualitäten • Bedeutung und angestrebte Eigenschaften der Anforderungsspezifikation • Regeln für die Anforderungsspezifikation • Objektorientierte Anforderungsanalyse mit Anwendungsfällen, statischen und dynamischen Modellen mit Klassenbildung, die dem Liskovschen Substitutionsprinzip genügt (unter Verwendung der UML-Diagramme für Anwendungsfälle, Klassendiagramme, Interaktionsund Zustandsdiagramme) Prüfung der Anforderungsspezifikation • Software-Prüfungen im Allgemeinen • Review-Varianten • Abläufe von Reviews • Review-Regeln • Review-Checklisten • Fallen und Gegenmittel Software-Architektur • Was ist Software-Architektur? • Sichten (Views) und Blickwinkel (Viewpoints) der Software-Architektur • Einflussfaktoren für die Software-Architektur • Entwurf einer Software-Architektur • Architekturstile • Entwurfsmuster • Modularisierung, Separation of Concern, Abstraktion, Information Hiding • Architekturreview

Lerninhalte 2	Benutzungsschnittstellenentwurf • Software-Ergonomie: Aspekte und Qualitäten • Interaktionsformen und -mittel • Werkzeuge • Usability-Evaluationsverfahren Einsatz von Datenbanken • Aufgaben und Architektur von Datenbanksystemen; externe, konzeptionelle und interne Ebene • Objektorientierte und relationale Datenbankmodellierung • Abbildung von objektorientierten Schemata auf relationale Datenbankschemata • Relationale Datenbanksysteme • Structured Query Language (SQL): Schemadefinition, Datenmanipulation, Anfragen, Integritätsbedingungen • Normalformen: 1NF, 2NF, 3NF Implementierung • Feinentwurf (Klassen, Zustands- und Aktivitätsdiagramme der UML) • Programmiersprachen • Programmierrichtlinien • Code-Qualität und Metriken • Vermeidung von Code-Redundanz • Entwicklungsumgebungen Test • Möglichkeiten und Grenzen des Testens • Testarten (Komponenten-/Integrations-/Systemtests) • Test-Varianten: Black-Box, White-Box-Testen • Testabdeckungsmaße • Testvorbereitung, -durchführung und -protokollierung Dokumentation • interne Software-Dokumentation • Benutzungshandbücher und Online-Hilfen Änderungs- und Konfigurationsmanagement • Wartung, Evolution und Reengineering • Bedeutung der Software-Wartung • Gesetze von Lehman • Änderungsprozesse • Werkzeuge für das Konfigurationsmanagement
Prüfungsformen	s. Software-Projekt 1

Literatur	• R. Pressman: Software Engineering - A Practitioner's Approach. 6. Auflage, McGraw-Hill, 2004. • I. Sommerville: Software Engineering. 8. Auflage, Addison-Wesley, 2006. • W. Zuser, T. Grechenig, M. Köhle: Software Engineering mit UML und dem Unified Process. 2. Auflage, Pearson Studium, 2004. • B. Brügge, A. H. Dutoit: Objektorientierte Softwaretechnik mit UML, Entwurfsmustern und Java. Pearson Studium, 2004. • Jochen Ludewig, Horst Lichter: Software Engineering - Grundlagen, Menschen, Prozesse, Techniken. dpunkt.verlag, 2006. • Helmut Balzert: Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering. 3. Auflage, Spektrum Akademischer Verlag, 2009. • Helmut Balzert: Lehrbuch der Softwaretechnik: Softwaremanagement. 2. Auflage, Spektrum Akademischer Verlag, 2008. • H. Störrle: UML 2 für Studenten. Pearson Studium, 2005. • Chris Rupp, Stefan Queins, Barbara Zengler: UML 2 glasklar. 3. Auflage, Hanser Verlag, 2007. • Chris Rupp: Requirements-Engineering und -Management. 5. Auflage, Hanser Verlag, 2009. • Klaus Pohl, Chris Rupp: Basiswissen Requirements Engineering. dpunkt.Verlag, 2009. • Klaus Pohl: Requirements Engineering - Grundlagen, Prinzipien, Techniken. 2. Auflage, dpunkt.Verlag, 2008. • Ramez A. Elmasri, Shamkant B. Navathe: Grundlagen von Datenbanksystemen, 3. Auflage, Pearson Studium, 2009.

Modulbezeichnung	**Technische Informatik 1**
Modulverantwortliche(r)	Prof. Dr. R. Drechsler
Modulart	Pflicht/Wahl ☒ Wahlpflicht ☐
Spezialisierungsbereich	
Dauer des Moduls	1 Semester
Kreditpunkte	9 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 84 h Übungsbetrieb/Prüfungsvorbereitung 186 h Summe 270 h
Turnus des Moduls	angeboten in jedem SoSe
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Grundlegende Konzepte moderner Rechner wiedergeben und erläutern können • Schaltkreismodellierung durch Boolesche Funktionen verstehen und erklären können • Hardware-Realisierungen von arithmetischen Funktionen darstellen können • Modellierung und Optimierungsansätze integrierter Schaltkreise umreißen können • Rechnersysteme anhand der eingeführten Konzepte selbständig beurteilen können • Unterschiedliche Hardware-Realisierungen unter den eingeführten Optimierungskriterien bewerten können • In Gruppen Probleme analysieren, gemeinsam Lösungsstrategien entwickeln und präsentieren können

Lerninhalte	. I. Rechnerarchitektur 1. Rechnersichtweisen: Ebenen und Sprachen, Hierarchie, Compiler, Interpreter 2. Aufbau und Funktionsweise: Hardware, Software, Firmware, Aufbau eines von-Neumann-Rechners, Arbeitsspeicher, Speicherzelle, Arbeitsweise eines Prozessors, Speicher, I/OBusse 3. Befehlssatz: RISC, CISC, Designprinzipien 4. Pipelining 5. Speicher: Hierachie, Organisation, Caches, Hintergrundspeicher 6. Parallelität: Ausprägungen, Klassifikation von parallelen Rechnerarchitekturen, Exkurs über Verbindungsstrukturen II. Digitale Schaltungen: 1. Schaltkreise: Technologien, Definition, Kosten, Semantik von kombinatorischen Schaltkreisen, Simulation, Teilschaltkreise, Hierarchischer Entwurf, Beispiele 2. Kodierung: Zeichen, Zahlen, Zahlensysteme, Übertragung, Fehlerkorrektur, HammingCode, Huffman-Code, Festkommadarstellungen, Zahlendarstellung durch Betrag und Vorzeichen, Einer-/Zweierkomplement-Darstellung, Gleitkommadarstellung (IEEE-754 Format) 3. Boolescher Kalkül: Funktion, Algebra, Ausdrücke, alternative Funktionsdarstellung, z.B. durch Entscheidungsdiagramme 4. Zweistufige Schaltungen: Logiksynthese, Implikanten, Primimplikanten, Minimierung, Quine/McClusky, Überdeckungsproblem 5. Integrierte Schaltungen, arithmetische Schaltungen, ALU 6. Schaltungen mit speichernden Elementen Lehrveranstaltung(en): ● 03-IBGP-TI1 Technische Informatik 1: Rechnerarchitektur und digitale Schaltungen
Prüfungsformen	KP, SL:1, PL:1, Portfolio, Klausur, Fachgespräch
Literatur	● B. Becker, R. Drechsler, P. Molitor: Technische Informatik – Eine Einführung, Pearson Studium, 2005 ● B. Becker, P. Molitor: Technische Informatik - Eine einführende Darstellung, Oldenbourg Wissenschaftsverlag, 2008 ● D. Hoffmann: Grundlagen der Technischen Informatik, 5. Aufl., Hanser Verlag, 2016 ● A. S. Tanenbaum, T. Austin: Computerarchitektur, 6. Aufl., Pearson Studium, 2014 ● D. Patterson, J. Hennessy: Computer Organization & Design - The Hardware/Software Interface, Morgan Kaufmann Publishers, 5. Auflage, 2013 ● R. Drechsler, A. Fink, J.Stoppe: Computer – Wie funktionieren Smartphone, Tablet & Co.?, Springer, 2017

Modulbezeichnung	**Praktische Informatik 1**
Modulverantwortliche(r)	Prof. Dr. U. Bormann
Modulart	Pflicht/Wahl ☒ Wahlpflicht ☐
Spezialisierungsbereich	
Dauer des Moduls	1 Semester
Kreditpunkte	9 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 112 h Übungsbetrieb/Prüfungsvorbereitung 158 h Summe 270 h
Turnus des Moduls	angeboten in jedem WiSe
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☒ Projekt ☐
Lernziele	• Grundlegende Informatikkonzepte wiedergeben und erklären können. • Konzepte einer imperativen Programmiersprache kennen, verstehen und anwenden können. • Anschauliche Sachverhalte im Modell der Objektorientierung ausdrücken können. • Einfache Algorithmen entwickeln und in Java umsetzen können. • Einfache in Java realisierte Algorithmen systematisch testen können. • Probleme in Teilprobleme zerlegen und diese Strukturierung mit Mitteln von Java umsetzen und aussagekräftig dokumentieren können. • Formale Syntaxbeschreibungen verstehen können. • Eine einfache Entwicklungsumgebung nutzen können. • LaTeX zur Erstellung einfacher Dokumente nutzen können. • Versionsverwaltungssysteme einsetzen können. • In Gruppen Probleme analysieren und gemeinsam Lösungsstrategien entwickeln und präsentieren können. Die Vorlesungen Praktische Informatik 1 und 2 vermitteln essenzielles Grundwissen und Basisfähigkeiten, deren Beherrschung für nahezu jede vertiefte Beschäftigung mit Informatik – sowohl in der industriellen Anwendung, als auch in der Forschung – Voraussetzung ist.

Lerninhalte	1. Basiswissen: von Neumannsche Rechnerorganisation – Grundlagen der Rechnerarchitektur – Programm und Prozess – Programmiersprachen – Compiler, Assembler, Loader, Linker, Interpreter, Laufzeitumgebungen, Betriebssysteme – Grafische Benutzungsschnittstellen 2. Datenstrukturen: Information und ihre Repräsentation – Datentypen und Typanalyse – Elementare und zusammengesetzte Datentypen – rekursive Datentypen – Kanonische Operationen auf den eingeführten Datenstrukturen 3. Programmierparadigmen: (1) Imperative und funktionale Programmierung, (2) Objektorientierte (imperative) Programmierung, (3) Sequenzielle Programme versus nebenläufige Programme 4. Grundkomponenten imperativer Programmiersprachen: Schnittstellen und Ein-/Ausgabe, Variablen und Zuweisungen, Kontrollstrukturen, Blöcke, Funktionen, Rekursion 5. Syntax und Semantik imperativer Programmiersprachen: Syntax und Methoden der Syntax-Spezifikation, reguläre Ausdrücke, (erweiterte) Backus-Naur-Form (E)BNF 6. Prinzipien der objektorientierten Programmierung: Geheimnisprinzip – Methoden – Operationen – Objekte – Klassen – Botschaften – Ereignisverarbeitung – Attribute – Vererbung – Polymorphismus – Überladung – Generische Datentypen 7. Umsetzung der Punkte 2.-6. mit Java – Illustration anhand einfacher Algorithmen 8. Programmdokumentation und zugehörige Hilfswerkzeuge, z.B. JavaDoc 9. Testen von Programmen und zugehörige Hilfswerkzeuge, z.B. JUnit 10. Grundlagen der Netzwerkkommunikation: IP-Adressen, DNS, TCP, UDP 11. Grundkonzepte der Entwicklung graphischer Oberflächen Programmier-Praktikum: Programmentwicklung in Java – Realisierung einzelner, überschaubarer Programmieraufgaben Lehrveranstaltung(en): • 03-IBGP-PI1 Praktische Informatik 1: Imperative Programmierung und Objektorientierung
Prüfungsformen	KP; PL1: 70%, PL2: 30%; Portfolio, Klausur
Literatur	• David J. Barnes, Michael Kölling: Java lernen mit BlueJ - Objects first - Eine Einführung in Java. Aktuelle Auflage. Pearson Studium. Weitere Informationen (Beispielprogramme, Musterlösungen, im WWW verfügbare Literatur) sind auf der Web-Seite der Veranstaltung zu finden.

Modulbezeichnung	**Praktische Informatik 2**
Modulverantwortliche(r)	Prof. Dr. U. Bormann
Modulart	Pflicht/Wahl ☒ Wahlpflicht ☐
Spezialisierungsbereich	
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	angeboten in jedem SoSe
Voraussetzung für die Teilnahme	Keine ☒ Folgende Inhaltliche Voraussetzungen: Praktische Informatik 1
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Typische Datenstrukturen identifizieren und problemadäquat einsetzen können. • Datenstrukturen und Algorithmen in Java umsetzen können. • Wesentliche Algorithmen der Informatik erklären, anwenden und modifizieren können. • Algorithmische Alternativen bezüglich der Eignung für ein Problem beurteilen können. • Grundbegriffe der formalen Verifikation erläutern können. • Die Komplexität von einfachen Algorithmen analysieren können. • Eine komplexe Entwicklungsumgebung nutzen können. • Generische und funktionale Konzepte in eigenen Programmen einsetzen können. • In Gruppen Probleme analysieren und gemeinsam Lösungsstrategien entwickeln und präsentieren können. Die Vorlesungen Praktische Informatik 1 und 2 vermitteln essenzielles Grundwissen und Basisfähigkeiten, deren Beherrschung für nahezu jede vertiefte Beschäftigung mit Informatik – sowohl in der industriellen Anwendung, als auch in der Forschung – Voraussetzung ist.

Lerninhalte	1. Algorithmen: Begriff des Algorithmus – Beschreibung von Algorithmen – Algorithmische Umsetzung kanonischer Operationen auf Datenstrukturen – Grundlegende Strategien: Greedy, Divide-and-Conquer, Backtracking, dynamische Programmierung 2. Komplexität von Algorithmen – O(n)-Notation und asymptotische Analyse 3. Suchen und Sortieren auf Arrays: Binäre Suche – Quicksort und weitere Sortieralgorithmen – Komplexitätsvergleiche 4. Mengen – Multimengen – Relationen – Funktionen: Datenstrukturen und Algorithmen zur Realisierung kanonischer Operationen (z.B. Mengenalgebra) 5. Listen – Stapel – Warteschlangen: Datenstrukturen zur Realisierung (Arrays versus Verkettung und dynamische Speicherallokation für Elemente), Algorithmen zur Realisierung kanonischer Operationen (Listentraversion, Anfügen, Einfügen, Löschen, Suchen, Stack-Operationen, FIFO-Warteschlangenoperationen) 6. Bäume: Binäre Bäume, AVL-Bäume, Rot-Schwarz-Bäume, B-Bäume – Suchen, Einfügen, Löschen, Traversion 7. Hashing: Hash-Array, Hashfunktion, Hash Buckets, offenes Hashing 8. Graphen: ungerichtete, gerichtete, gewichtete Graphen – Repräsentation durch Knoten- und Kantenlisten, durch Adjazenzmatrizen, Adjazenzlisten – Algorithmen auf Graphen: Breitensuche, Tiefensuche, kürzeste Wege auf gewichteten Graphen: Dijkstras Algorithmus, minimal aufspannende Bäume: Algorithmen von Prim et al. und Kruskal 9. Spezifikation von Programmen: Vor- und Nachbedingungen – Invarianten 10. Verifikation: Partielle und totale Korrektheit sequenzieller Programme – Formale Verifikation, z.B. Hoare Logik (Pre-/Postconditions) – Eigenschaftsbeweis durch Strukturelle Induktion Lehrveranstaltung(en): • 03-IBGP-PI2 Praktische Informatik 2: Algorithmen und Datenstrukturen
Prüfungsformen	KP, PL1: 70%, PL2: 30%, Portfolio, Klausur
Literatur	• G. Saake und K.-U. Sattler: Algorithmen und Datenstrukturen. dpunkt.verlag, Heidelberg (2004) • R. Schiedermeier: Programmieren mit Java. Pearson, München (2005) Weitere Informationen (Beispielprogramme, Musterlösungen, im WWW verfügbare Literatur) sind auf der Web-Seite der Veranstaltung zu finden.

Modulbezeichnung	**Rechnerarchitektur und Eingebettete Systeme**
Modulverantwortliche(r)	Prof. Dr. R. Drechsler
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Systemsoftware / Eingebettete Systeme, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten in jedem WiSe
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Technische Informatik 1
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Den detaillierten Aufbau moderner Rechner analysieren und erklären können • Den modernen Systementwurf analysieren können • Die Funktionsweise von Compilern und Codegenerierung grundlegend verstehen • Syntheseansätze für Hardware kennen und darstellen können • Qualität von Systementwürfen beurteilen können • Aufgabenlösungen und Beispiele in den wöchentlichen Tutorien eigenständig bearbeiten und präsentieren können • Probleme beim Entwurf eines komplexen Systems selbständig erkennen können

Lerninhalte	Aufbau eines Rechners • Maschinensprachen • Datenpfad und Kontrollpfad • Pipelining Systementwurf - Modelle und Methoden • Zielarchitekturen für HW/SW-Systeme • Allokation, Bindung, Ablaufplanung • Partitionierung Software-Entwurf • Compiler • Codegenerierung • Registerallokation Hardware-Entwurf • Synthese • Verifikation • Verdrahtung • Test Schätzung der Entwurfsqualität
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	• T. Flik, Mikroprozessortechnik und Rechnerstrukturen, 7. Aufl., Springer, 2005 • B. Becker, R. Drechsler, P. Molitor, Technische Informatik – Eine Einführung, Pearson Studium, 2005 • R. E. Bryant, D. O'Hallaron, Computer Systems, Prentice Hall, 2003 • A. S. Tanenbaum, J. Goodman, Computerarchitektur, 4. Aufl., Pearson Studium, 2001 • H. Wuttke, K. Henke, Schaltsysteme, Pearson Studium, 2002 • W. Stallings, Computer Organization & Architecture, Prentice Hall, 2002 • C. Siemers, A. Sikora, Taschenbuch Digitaltechnik, Fachbuchverlag Leipzig, 2002 • T. Beierlein, O. Hagenbruch, Taschenbuch Mikroprozessortechnik, Fachbuchverlag Leipzig, 2001 • D. Patterson, J. Hennessy, Computer Organization & Design - The Hardware/Software Interface, Morgan Kaufmann Publishers, 1997 • Axel Sikora, Rolf Drechsler, Software-Engineering und Hardware-Design, Carl Hanser Verlag, 2002 • Jürgen Teich, Digitale Hardware/Software-Systeme, Springer, 1997

Modulbezeichnung	**Betriebssysteme**
Modulverantwortliche(r)	Prof. Dr. J. Peleska
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Systemsoftware / Eingebettete Systeme, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Technische Informatik 2
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	In dieser Vorlesung erwerben die Teilnehmer Kenntnisse der Grundkonzepte und Leistungsmerkmale moderner Betriebssysteme, sowie ihrer Anwendung in der Systemprogrammierung. Damit werden sie in die Lage versetzt, bei Entwurf und Entwicklung komplexer Anwendungen die richtigen Betriebssystemmechanismen und -dienste auszuwählen und korrekt in die Applikation zu integrieren. Die Ziele im Einzelnen: • Geeignete Betriebssystemdienste problemabhängig auswählen können. • Die Wirkung von Betriebssystemdiensten auf eine Gesamtanwendung einschätzen können. • Systemprogrammierung unter Unix effizient und korrekt entwickeln können. • Die Korrektheit komplexer Betriebssystemmechanismen verifizieren können • Zuverlässigkeitsmechanismen (Safety und Security) in Betriebssystemen bzgl. ihrer Wirksamkeit beurteilen können • Verteilte kommunizierende Anwendungen entwerfen und realisieren können

Lerninhalte	Vertiefung der Grundkonzepte heutiger Betriebssysteme: 1. Prozesse, Threads und Kommunikationsmechanismen 2. Speicherverwaltung 3. Dateisysteme 4. Ein-/Ausgabeverwaltung 5. Betriebsmittelvergabe 6. Synchronisation 7. Architekturen für Betriebssystemkerne 8. Zuverlässigkeitsmechanismen zur Gewährleistung von Safety, Security, Availability, Reliability 9. Verifikation von Betriebssystemmechanismen mit Hilfe formaler Spezifikationen und Modellprüfung. Die Übungen vertiefen den Vorlesungsstoff anhand von Aufgaben aus den Bereichen Systemprogrammierung – Entwicklung von Algorithmen für Betriebssystemmechanismen – Verifikation von Betriebssystemmechanismen. Beispiele werden vor allem aus dem Bereich der Unix-Betriebssysteme gewählt (Linux, Solaris). Programmierkenntnisse in C oder C++ sind Voraussetzung.
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	• A. Tanenbaum: Modern Operating Systems, Prentice Hall (dieses Buch wird regelmäßig neu aufgelegt; es wird die jeweils neueste Auflage empfohlen) • W. Stallings: Betriebssysteme, Pearson Studium (dieses Buch wird regelmäßig neu aufgelegt; es wird die jeweils neueste Auflage empfohlen) • W.R. Stevens: Unix Network Programming, Prentice Hall (dieses Buch wird regelmäßig neu aufgelegt; es wird die jeweils neueste Auflage empfohlen) • U. Vahalia: Unix Internals - The New Frontiers, Prentice Hall 1996. • J. Peleska: Formal Methods and the Development of Dependable Systems, Christian-Albrechts-Universität zu Kiel 1996.

Modulbezeichnung	**Rechnernetze**
Modulverantwortliche(r)	Prof. Dr. U. Bormann
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Systemsoftware / Eingebettete Systeme
Dauer des Moduls	1 Semester
Kreditpunkte	8 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 84 h Übungsbetrieb/Prüfungsvorbereitung 156 h Summe 240 h
Turnus des Moduls	i. d. R. angeboten in jedem SoSe
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Technische Informatik 2
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• In der Terminologie des Fachgebiets Rechnernetze kommunizieren können, Systemkomponenten anhand dieser Terminologie klassifizieren können. • Lösungsvarianten für kommunikationstechnische Probleme bewerten können; insbesondere für die Vielzahl der behandelten Techniken (s. unten): Voraussetzungen erkennen, Aufwände abschätzen und Einsatzgebiete (auch quantitativ) bewerten können. • Mechanismen der Marktdurchsetzung von technischen Spezifikationen verstehen und bewerten können. • Die globalen Strategien auf einfache vorgegebene Einzelsituationen übertragen können.
Lerninhalte	• ISO-Referenzmodell für offene Kommunikationssysteme (OSI-Modell) • Dienste und Protokolle (Übertragungstechnik/Modemstandards, HDLC, ISDN, LAN-Topologien, Ethernet, Internet-Protokolle, ASN.1/XDR, RPC, Betriebsprotokolle) • Anwendungsstandards (u.a. FTP, TELNET, Namensdienste, E-Mail, Web: SGML/HTML/XML, HTTP, Web Services/REST). • Sicherheit in Rechnernetzen • Standardisierungsprozesse
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	• Andrew S. Tanenbaum: Computer Networks, 5th Edition, Pearson, 2010 (bzw. die deutsche Übersetzung: Computernetzwerke, 5. Auflage, Pearson Studium, 2012) • http://rfc-editor.org/rfc.html (für die Internet-Standarddokumente)

Modulbezeichnung	**Softwaretechnik**
Modulverantwortliche(r)	Prof. Dr. R. Koschke
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Systemsoftware / Eingebettete Systeme
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Software-Projekt
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	Die Studierenden verfügen über die folgenden fachlichen Kompetenzen: • Methodenkompetenzen • Analyse-/Design- und Realisierungskompetenzen • Technologische Kompetenzen • fortgeschrittene Methoden der Softwaretechnik kennen, beurteilen und umsetzen können • Urteilsfähigkeit für technische Methoden • Zusammenführung einzelner Methoden zu einem Ganzen Die Studierenden verfügen über die folgenden sozialen Kompetenzen: • Projektmanagement-Kompetenz zu Software-Projekten

Lerninhalte	Software-Metriken • was ist eine Metrik? • Messtheorie • Skalen • Prozess-, Produkt- und Ressourcenmetriken Entwicklungsprozesse • alternative Software-Entwicklungsprozesse (z.B. Clean-Room und Agile Entwicklung) • Capability Maturity Model, Spice und Bootstrap • Prozessverbesserungen • Persönlicher Prozess Software-Architektur • Sichten und Blickwinkel, IEEE-Standard P1471 • Dokumentation von Software-Architektur und Architekturbeschreibungssprachen • Entwurfs- und Architekturmuster und Referenzarchitekturen • Qualitätseigenschaften • Entwurf von Architekturen • Analyse von Architekturen (insbesondere SAAM und ATAM) Software-Produktlinien • Definition und Beispiele • Vor- und Nachteile • Practice Areas • Einführung von Produktlinien • Ansätze zur technischen Realisierung • Beschreibungen und Notationen (z.B. Feature-Graphen) • Besonderheiten beim Requirementsengineering, Konfigurationsmanagement und Test • Konfiguration von Produktlinien Komponentenbasierte Entwicklung • Eigenschaften, Vor- und Nachteile • Komponentenmodell • Schnittstellen und Kontrakte • Managementfragen • Rahmenwerke • OMG CORBA und OMA • Microsoft DCOM, OLE und ActiveX • Sun Java und JavaBeans Modellgetriebene Entwicklung • Ideen, Eigenschaften, Vor- und Nachteile • Werkzeugunterstützung (z.B.Eclipse Open Architecture Ware) Kosten- und Aufwandsschätzung - insbesondere Function-Points und CoCoMo I und II Empirische Softwaretechnik • Bedeutung und Methoden der empirischen Softwaretechnik • Bestandteile kontrollierter Experimente und Fallstudien In der Vorlesung Softwaretechnik geht es um die Methodik der Software-Entwicklung nach Ingenieursprinzipien. Anhand der Projektsimulationssoftware SESAM kann die Durchführung eines Software-Projektes geübt werden. Das Kapitel ‘Empirische Softwaretechnik’ diskutiert grundlegende Methoden zum empirischwissenschaftlichen Erkenntnisgewinn bei der Softwareentwicklung.
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung

Literatur	• Paul Clements und Linda Northrop: Software Product Lines: Practices and Patterns, Addison Wesley Professional, 2002 • Clemens Szyperski, Dominik Gruntz, Stephan Murer: Component Software, Addison Wesley Professional, 2002 • Norman E. Fenton, Shari L. Pfleeger: Software Metrics A Rigorous & Practical Approach, Second Edition, PWS Publishing Company, 1997 • Roger Pressman: Software Engineering – A Practioner's Approach, fünfte Ausgabe, McGraw-Hill, 2003 • Ludewig, Jochen; Lichter, Horst: Software Engineering Grundlagen, Menschen, Prozesse, Techniken. dpunkt.verlag, 2006 • Ian Sommerville: Software Engineering, Siebte Ausgabe, Addison-Wesley, 2004. • Len Bass and Paul Clements and Rick Kazman: Software Architecture in Practice, zweite Auflage, Addison Wesley, 2003. • Frank Buschmann, Regine Meunier, Hans Rohnert and Peter Sommerlad, Michael Stal: Pattern-oriented Software Architecture: A System of Patterns, Volume 1, Wiley, 1996. • Christine Hofmeister, Robert Nord, Dilip Soni: Applied Software Architecture, Addison Wesley, Object Technology Series, 2000. • Software Cost Estimation with COCOMO II; Barry W. Boehm et al.; Prentice Hall, 2000. • Poensgen, Benjamin; Bock, Bertram: Die Function-Point-Analyse. Ein Praxishandbuch. Dpunkt Verlag, 2005. ISBN 978-3898643320 • Balzert, Helmut: Lehrbuch der Softwaretechnik Softwaremanagement. 2. Spektrum, Akademischer Verlag, 2008. ISBN 978-3-8274-1161-7 • Bunse, Christian ; Knethen, Antje von: Vorgehensmodelle kompakt. Spektrum-Akademischer Verlag, 2002. ISBN 978-3827412034 • Kruchten, Phillipe: The Rational Unified Process: An Introduction. Reading, Mass.: Addison-Wesley, 1998 • Beck, Kent: Extreme Programming Explained. Addison-Wesley, 2000 (The XP Series). ISBN 201-61641-6 • Kneuper 2006 Kneuper, Ralf: CMMI Verbesserung von Softwareprozessen mit Capability Maturity Model. 2. dpunkt.verlag, 2006. ISBN 3-89864-373-5 • Siviy, Jeannine M.; Penn, M. L.; Stoddard, Robert W.: CMMI and Six Sigma Partners in Process Improvement. Addison-Wesley, 2007 (SEI Series in Software Engineering). ISBN 978-0-321-51608-4 • Stahl, Thomas ; Volter, Markus ; Efftinge, Sven ; Haase, Arno: Modellgetriebene Softwareentwicklung Techniken, Engineering, Management. zweite Auflage. dpunkt.verlag, 2007 • Gamma, Erich ; Helm, Richard ; Johnson, Ralph ; Vlissides, John: Desig Patterns–Elements of Reusable Object-Oriented Software. Addison Wesley, 2003 • Pattern-oriented Software Architecture: A System of Patterns; Frank Buschmann, Regine Meunier, Hans Rohnert and Peter Sommerlad, Michael Stal; Volume 1, Wiley, 1996. • Endres, Albert ; Rombach, Dieter: A Handbook of Software and Systems Engineering. Addison Wesley, 2003 • Prechelt 2001 Prechelt, Lutz: Kontrollierte Experimente in der Softwaretechnik Potenzial und Methodik. Springer, 2001 • Yin, Robert K.: Case Study Research. Bd. 5. SAGE Publications, 2003. ISBN 0-7619-2553-8

Modulbezeichnung	**Informationssicherheit**
Modulverantwortliche(r)	Prof. Dr. C. Bormann
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Systemsoftware / Eingebettete Systeme
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Technische Informatik 2
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Grundkonzepte der Informationssicherheit kennen; • Die gängigsten Sicherheitsprobleme in heutigen IT-Infrastrukturen und deren Ursachen kennen; • Notwendigkeit für den Einsatz von Sicherheitstechnik erkennen; • Grenzen der im Einsatz befindlichen Technologien einschätzen können; • Verschiedene Bereiche von Sicherheitstechnik einordnen können; • Modelle und Methoden zur systematischen Konstruktion sicherer Systeme kennen.
Lerninhalte	• Grundbegriffe der IT-Sicherheit, Bedrohungen und Sicherheitsprobleme: Vertraulichkeit, Integrität, Verfügbarkeit etc.; Viren, Würmer, Trojanische Pferde etc. • Kryptografie (Symmetrisch, Asymmetrisch, Hash, PRF): DES, 3DES, AES; RSA, DSA; MD5, SHA1; TLS-PRF, PBKDF2 • Mechanismen zur Authentisierung und Integritätsprüfung digitaler Signaturen, Zertifikate, PKI • Zugriffskontrolle, Autorisierung, Rollen • Sicherheitsprotokolle, z.B. Schlüsselaustausch Diffie-Hellman, TLS (SSL), Kerberos • Probleme mit Protokollen, Angriffe (fehlende Bindung, Replay, . . .) • Netzsicherheit (Firewalls/IDS, VPN, Anwendungssicherheit) • Sicherheit in Layer 2 (GSM, WLAN, . . .) • Software-Zertifizierung: Common Criteria • Mobiler Code • Smart Cards, Trusted Computing Platform • Security Engineering • Organisationelle Sicherheit; Security: The Business Case
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung

Literatur	• (deutschsprachig:) Claudia Eckert: IT-Sicherheit: Konzepte - Verfahren - Protokolle; Oldenbourg 2009; 981 Seiten • (englischsprachig:) Ross Anderson: Security engineering: a guide to building dependable distributed systems; Wiley 2008; 1040 Seiten

Modulbezeichnung	**Informationssicherheit (Kopie vom Fri May 22 16:55:34 +0200 2020) (Kopie vom Fri May 22 17:00:49 +0200 2020)**
Modulverantwortliche(r)	Prof. Dr. C. Bormann
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Systemsoftware / Eingebettete Systeme
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Technische Informatik 2
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Grundkonzepte der Informationssicherheit kennen; • Die gängigsten Sicherheitsprobleme in heutigen IT-Infrastrukturen und deren Ursachen kennen; • Notwendigkeit für den Einsatz von Sicherheitstechnik erkennen; • Grenzen der im Einsatz befindlichen Technologien einschätzen können; • Verschiedene Bereiche von Sicherheitstechnik einordnen können; • Modelle und Methoden zur systematischen Konstruktion sicherer Systeme kennen.
Lerninhalte	• Grundbegriffe der IT-Sicherheit, Bedrohungen und Sicherheitsprobleme: Vertraulichkeit, Integrität, Verfügbarkeit etc.; Viren, Würmer, Trojanische Pferde etc. • Kryptografie (Symmetrisch, Asymmetrisch, Hash, PRF): DES, 3DES, AES; RSA, DSA; MD5, SHA1; TLS-PRF, PBKDF2 • Mechanismen zur Authentisierung und Integritätsprüfung digitaler Signaturen, Zertifikate, PKI • Zugriffskontrolle, Autorisierung, Rollen • Sicherheitsprotokolle, z.B. Schlüsselaustausch Diffie-Hellman, TLS (SSL), Kerberos • Probleme mit Protokollen, Angriffe (fehlende Bindung, Replay, . . .) • Netzsicherheit (Firewalls/IDS, VPN, Anwendungssicherheit) • Sicherheit in Layer 2 (GSM, WLAN, . . .) • Software-Zertifizierung: Common Criteria • Mobiler Code • Smart Cards, Trusted Computing Platform • Security Engineering • Organisationelle Sicherheit; Security: The Business Case
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung

Literatur	• (deutschsprachig:) Claudia Eckert: IT-Sicherheit: Konzepte - Verfahren - Protokolle; Oldenbourg 2009; 981 Seiten • (englischsprachig:) Ross Anderson: Security engineering: a guide to building dependable distributed systems; Wiley 2008; 1040 Seiten

Modulbezeichnung	**Rechnerarchitektur und Eingebettete Systeme**
Modulverantwortliche(r)	Prof. Dr. R. Drechsler
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Systemsoftware / Eingebettete Systeme, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten in jedem WiSe
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Technische Informatik 1
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Den detaillierten Aufbau moderner Rechner analysieren und erklären können • Den modernen Systementwurf analysieren können • Die Funktionsweise von Compilern und Codegenerierung grundlegend verstehen • Syntheseansätze für Hardware kennen und darstellen können • Qualität von Systementwürfen beurteilen können • Aufgabenlösungen und Beispiele in den wöchentlichen Tutorien eigenständig bearbeiten und präsentieren können • Probleme beim Entwurf eines komplexen Systems selbständig erkennen können

Lerninhalte	Aufbau eines Rechners • Maschinensprachen • Datenpfad und Kontrollpfad • Pipelining Systementwurf - Modelle und Methoden • Zielarchitekturen für HW/SW-Systeme • Allokation, Bindung, Ablaufplanung • Partitionierung Software-Entwurf • Compiler • Codegenerierung • Registerallokation Hardware-Entwurf • Synthese • Verifikation • Verdrahtung • Test Schätzung der Entwurfsqualität
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	• T. Flik, Mikroprozessortechnik und Rechnerstrukturen, 7. Aufl., Springer, 2005 • B. Becker, R. Drechsler, P. Molitor, Technische Informatik – Eine Einführung, Pearson Studium, 2005 • R. E. Bryant, D. O'Hallaron, Computer Systems, Prentice Hall, 2003 • A. S. Tanenbaum, J. Goodman, Computerarchitektur, 4. Aufl., Pearson Studium, 2001 • H. Wuttke, K. Henke, Schaltsysteme, Pearson Studium, 2002 • W. Stallings, Computer Organization & Architecture, Prentice Hall, 2002 • C. Siemers, A. Sikora, Taschenbuch Digitaltechnik, Fachbuchverlag Leipzig, 2002 • T. Beierlein, O. Hagenbruch, Taschenbuch Mikroprozessortechnik, Fachbuchverlag Leipzig, 2001 • D. Patterson, J. Hennessy, Computer Organization & Design - The Hardware/Software Interface, Morgan Kaufmann Publishers, 1997 • Axel Sikora, Rolf Drechsler, Software-Engineering und Hardware-Design, Carl Hanser Verlag, 2002 • Jürgen Teich, Digitale Hardware/Software-Systeme, Springer, 1997

Modulbezeichnung	**Betriebssysteme**
Modulverantwortliche(r)	Prof. Dr. J. Peleska
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Systemsoftware / Eingebettete Systeme, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Technische Informatik 2
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	In dieser Vorlesung erwerben die Teilnehmer Kenntnisse der Grundkonzepte und Leistungsmerkmale moderner Betriebssysteme, sowie ihrer Anwendung in der Systemprogrammierung. Damit werden sie in die Lage versetzt, bei Entwurf und Entwicklung komplexer Anwendungen die richtigen Betriebssystemmechanismen und -dienste auszuwählen und korrekt in die Applikation zu integrieren. Die Ziele im Einzelnen: • Geeignete Betriebssystemdienste problemabhängig auswählen können. • Die Wirkung von Betriebssystemdiensten auf eine Gesamtanwendung einschätzen können. • Systemprogrammierung unter Unix effizient und korrekt entwickeln können. • Die Korrektheit komplexer Betriebssystemmechanismen verifizieren können • Zuverlässigkeitsmechanismen (Safety und Security) in Betriebssystemen bzgl. ihrer Wirksamkeit beurteilen können • Verteilte kommunizierende Anwendungen entwerfen und realisieren können

Lerninhalte	Vertiefung der Grundkonzepte heutiger Betriebssysteme: 1. Prozesse, Threads und Kommunikationsmechanismen 2. Speicherverwaltung 3. Dateisysteme 4. Ein-/Ausgabeverwaltung 5. Betriebsmittelvergabe 6. Synchronisation 7. Architekturen für Betriebssystemkerne 8. Zuverlässigkeitsmechanismen zur Gewährleistung von Safety, Security, Availability, Reliability 9. Verifikation von Betriebssystemmechanismen mit Hilfe formaler Spezifikationen und Modellprüfung. Die Übungen vertiefen den Vorlesungsstoff anhand von Aufgaben aus den Bereichen Systemprogrammierung – Entwicklung von Algorithmen für Betriebssystemmechanismen – Verifikation von Betriebssystemmechanismen. Beispiele werden vor allem aus dem Bereich der Unix-Betriebssysteme gewählt (Linux, Solaris). Programmierkenntnisse in C oder C++ sind Voraussetzung.
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	• A. Tanenbaum: Modern Operating Systems, Prentice Hall (dieses Buch wird regelmäßig neu aufgelegt; es wird die jeweils neueste Auflage empfohlen) • W. Stallings: Betriebssysteme, Pearson Studium (dieses Buch wird regelmäßig neu aufgelegt; es wird die jeweils neueste Auflage empfohlen) • W.R. Stevens: Unix Network Programming, Prentice Hall (dieses Buch wird regelmäßig neu aufgelegt; es wird die jeweils neueste Auflage empfohlen) • U. Vahalia: Unix Internals - The New Frontiers, Prentice Hall 1996. • J. Peleska: Formal Methods and the Development of Dependable Systems, Christian-Albrechts-Universität zu Kiel 1996.

Modulbezeichnung	**Bildverarbeitung**
Modulverantwortliche(r)	PD Dr. B. Gottfried
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten in jedem WiSe
Voraussetzung für die Teilnahme	Keine ☐ Folgende: Inhaltliche Voraussetzungen: Praktische Informatik 2, Mathematische Grundlagen 2
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Die grundlegenden Verfahren, Methoden und Ansätze der digitalen Bildverarbeitung erklären und wiedergeben können. • In der Terminologie des Fachgebietes kommunizieren können. • Die einzelnen Methoden/Ansätze des Fachgebietes in den Gesamtkontext einordnen können und dadurch die einzelnen Methoden anhand der Terminologie klassifizieren können • Das Fachgebiet (oder Teile des Fachgebietes) im Kontext zu anderen Disziplinen einordnen können • Prinzipien - respektive grundlegende Verfahren - auf einzelne konkrete Aufgabensituationen übertragen können.
Lerninhalte	Es wird Schritt für Schritt der Stoff von den bildgebenden Verfahren über die Vorverarbeitung, Segmentierung und Merkmalsextraktion bis hin zur Klassifikation behandelt. So wird der Prozess vom „Pixel zum Objekt“ im Rahmen der Vorlesung beschritten. Die Inhalte sind dann im Einzelnen: • Grundlegende Begriffe der digitalen Bildverarbeitung • Bildgebende Verfahren • Vorverarbeitung: Kontrastverstärkende, entzerrende und auch rauschunterdrückende Verarbeitungsmethoden zur Bildverbesserung bzw. –restaurierung • Binärbildverarbeitung (spez. Morphologie) • Segmentierungsverfahren (Diskontinuitätskriterien, Homogenitätskriterien, hybride Ansätze) basierend auf Kanten-, Textur- und Farbmerkmalen • Bestimmung von statistischen, geometrischen und densitometrischen Merkmalen • Klassifikation von Merkmalen (Wahrscheinlichkeit, Diskriminanten- und Distanzfunktionen). Die Übungsaufgaben werden mit dem frei zugänglichen Tool “ImageJ” durchgeführt, dass in dem Buch von Burger und Burge (siehe Literatur) verwendet wird. Es vereint die Bildbearbeitung mit der Bildverarbeitung.
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Klausur oder mündliche Prüfung

Literatur	• Wolfgang Abmayr, Einführung in die digitale Bildverarbeitung, Teubner, 1994 • Wilhelm Burger (Autor) und Mark James Burge, Digitale Bildverarbeitung: Eine algorithmische Einführung mit Java, Springer, 2012 • David A. Forsyth and Jean Ponce, Computer Vision: A Modern Approach, Prentice Hall, 2002

Modulbezeichnung	**Grundlagen der Künstlichen Intelligenz**
Modulverantwortliche(r)	Prof. M. Beetz, PhD
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten in jedem SoSe
Voraussetzung für die Teilnahme	Keine ☒ Folgende: Inhaltliche Voraussetzungen: Programmier-Erfahrung, Logik, Wahrscheinlichkeiten
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Die grundlegenden Verfahren, Methoden und Ansätze der Künstlichen Intelligenz praktisch anwenden können • Fachliche Kompetenz insbesonders, aber nicht ausschließlich, in den Gebieten Suche, Logik, Planen, Maschinelles Lernen • Die Terminologie des Fachgebietes beherrschen • Die einzelnen Methoden/Ansätzen der KI in den Gesamtkontext einordnen können • Das Fachgebiete(oder Teile des Fachgebietes) im Kontext zu anderen Disziplinen einordnen können • Grundlegende Verfahren auf einzelne konkrete Aufgabensituationen übertragen und diese lösen können
Lerninhalte	Die Vorlesung soll einen Überblick über wichtige Arbeitsgebiete und Methoden der Künstlichen Intelligenz geben. Die Vorlesung führt Grundideen und Methoden der Künstlichen Intelligenz anhand des Lehrbuches von Russell und Norvig (s.u.) ein. Es werden folgende Themen behandelt: • Entwurfsprinzipien für und Spezifikation von “intelligenten” Agenten; • Problemlösen durch Suche: heuristische Suchverfahren, optimierende Suche; • Problemlösen mit wissensbasierten Methoden: Logik und Inferenz, Schlussfolgern über Raum und Zeit, Repräsentation von Ontologien, Repräsentation und Schlussfolgern über Alltagswissen; • Problemlösen mit unsicherem Wissen: Grundlagen der Wahrscheinlichkeits- und Entscheidungstheorie, Bayes Netze, Planen mit Markov-Entscheidungsprozessen; • Handlungsplanung: Generierung partiell geordneter Aktionspläne, Planung und Ausführung; • Maschinelles Lernen: Lernen von Entscheidungsbäumen, Lernen von Prädikaten mittels Beispiele, Reinforcement-Lernen.
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Klausur

Literatur	• Stuart Russell und Peter Norvig: Artificial Intelligence - A Modern Approach. Prentice Hall International, 2. Auflage (2003) • Uwe Schöning: Logik für Informatiker, Spektrum Akademischer Verlag, 5. Auflage (2000) • Artificial Intelligence: Foundations of Computational Agents von David L. Poole und Alan K. Mackworth von Cambridge University Press

Modulbezeichnung	**Robot Design Lab**
Modulverantwortliche(r)	Prof. Dr. F. Kirchner
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	8 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 184 h Summe 240 h
Turnus des Moduls	i.d.R. angeboten in jedem SoSe
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Verständnis der Robotik als integrierende Wissenschaft zwischen Elektrotechnik, Mechatronik und Informatik • Grundlegende Kenntnisse der Funktionsweise und sicherer technischer Umgang mit technologischen Komponenten für Robotik • Bewertung von Sensoren für Roboter in verschiedenen Anwendungsbereichen • Bewertung und Klassifikation von Motoren, Getrieben und Mechanismen für Roboter • Kenntnisse der wichtigsten Methoden und Verfahren zur Kontrolle und Steuerung von Robotern • Kenntnisse in Anwendung und Programmierung des STM32 Microcontrollers und des ROS Software-Frameworks. • In der Terminologie des Fachgebiets Robotik sicher kommunizieren können und Systemkomponenten anhand der Terminologie klassifizieren und bewerten können. • Durch den Übungsbetrieb in kleinen Gruppen wird die Kooperations- und Teamfähigkeit geübt.
Lerninhalte	• Sensor-Interfaces, Taster, Lichtsensoren, Widerstandspositionssensoren, Optosensoren, Enkoder • DC-Motoren, Getriebe, elektronische Kontrolle von Motoren, Servomotoren, • Einfaches Feedback Kontrolle, Proportional und Derivative Kontrolle, Reactive und Sequentielle • Kontrolle • Der STM32, FPGA's, ROS • Bildverarbeitung, Odometrie, Hindernisvermeidung, Steuerlogik
Prüfungsformen	Übungsaufgaben sowie Fachgespräch oder mündliche Prüfung
Literatur	Bräunl, Thomas. Embedded Robotics, Springer Berlin (2008)Martin, F. 'Robotic Explorations: A Hands on Introduction to Engineering', Prentice Hall, New Yersey (2001)

Modulbezeichnung	**Grundlagen des Maschinellen Lernens**
Modulverantwortliche(r)	Prof. Dr. T. Schultz
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik
Dauer des Moduls	1 Semester
Kreditpunkte	4 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 28 h Übungsbetrieb/Prüfungsvorbereitung 92 h Summe 120 h
Turnus des Moduls	i.d.R. angeboten in jedem SoSe
Voraussetzung für die Teilnahme	Keine ☐ Folgende Formale Voraussetzungen: Keine
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	Die Studierenden • können Probleme aus dem Bereich des maschinellen Lernens identifizieren. • können selbstständig Lösungsansätze für Probleme aus dem maschinellen Lernens vorschlagen. • kennen unterschiedliche Algorithmen für Klassifikations- und Regressionsprobleme und kennen deren Vorteile und Nachteile. • wissen wie Daten vorverarbeitet und visualisiert werden können. • wissen wie Maschinelles Lernen evaluiert werden kann.
Lerninhalte	Das Maschinelle Lernen (ML) ist eine Teilrichtung der künstlichen Intelligenz, die in den letzten Jahren rasant gewachsen ist und enorme Popularität erlangt hat. Die Vorlesung “Grundlagen des maschinellen Lernens” richtet sich an Bachelor-Studierende und soll ihnen das Rüstzeug geben, um Probleme aus dem Bereich ML selbständig lösen zu können. Der Fokus liegt dabei auf dem Kennenlernen der gängigen Methoden und deren Realisierung in Python. Daher werden zahlreiche praktische Anwendungsbeispiele herangezogen, statt alle Beweise zu führen oder stur eine Methode nach der anderen zu besprechen. Die Vorlesung findet einmal wöchentlich statt und hat keine Übung oder Übungsblätter. Die Themen werden auf Living Python Slides vermittelt! Besprochene Themen: • Machine Learning Basics • Classification • Clustering • Generative Modelle • Discriminative Modelle • Regression • Ensemble Methoden • Recommender Systems • (Tiefe) Neuronale Netzte (3 Blöcke)

Prüfungsformen	Klausur
Literatur	Alle notwendigen Unterlagen werden im Kurs zur Verfügung gestellt.

Modulbezeichnung	**Theorie reaktiver Systeme**
Modulverantwortliche(r)	Prof. Dr. J. Peleska
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Systemsoftware / Eingebettete Systeme
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Theoretische Informatik 1
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Semantische Alternativen für eingebettete Echtzeitsysteme bewerten können • Verständnis für die Grundkonzepte des Model Checkings entwickeln • Große (unendliche) Zustandsräume durch Abstraktion beherrschbar machen können • Semantische Modellierung zur Automatisierung bei Verifikation und Test einsetzen können
Lerninhalte	1. Modelle der operationellen Semantik: Zustands-Transitionssysteme, markierte Transitionssysteme („Labelled Transition Systems LTS“), Markierte Transitionssysteme mit Zeit („Timed LTS“), Transitionssysteme mit Codierung der Refusal-Information – Finite State Machines (FSM) – Interleaving-Semantics versus „true Parallelism“ : Harel’s StepSemantik für Statecharts – Kripke-Strukturen 2. Äquivalenz und Verfeinerung: Bisimilarität – Simulationsbeziehung - Verfeinerungen 3. Fundamentale Modelleigenschaften: Deadlockfreiheit – Livelockfreiheit - Safety- und Liveness-Eigenschaften – Fairness 4. Modell-orientierte Spezifikationsformalismen und ihre Semantik: Timed Automata – Hybrid Automata – Timed CSP 5. Implizite Spezifikationsformalismen und ihre Semantik: Trace Logik mit und ohne Zeit – Temporallogiken: Linear Time Logic (LTL), Computation Tree Logic (CTL), Timed Computation Tree Logic (TTCL) 6. Nachweis universeller Eigenschaften durch strukturelle Induktion über Syntax und operationelle Semantik. 7. Modellprüfung 8. Modellabstraktion
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung

Literatur	• Edmund M. Clarke, Orna Grumberg and Doron A. Peled: “Model Checking”, The MIT Press, 1999 • Christel Baier and Joost-Pieter Katoen: “Principles of Model Checking”, The MIT Press, 2008 • K. Apt, E.-R. Olderog: “Verification of Sequential and Concurrent Programs”, Springer, 1991

Modulbezeichnung	**Qualitätsorientierter System-Entwurf**
Modulverantwortliche(r)	Prof. Dr. R. Drechsler
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Systemsoftware / Eingebettete Systeme, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Jahre
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Technische Informatik 1
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Abläufe im Schaltkreisentwurf erklären können • Methoden zur Validierung von Entwürfen unterscheiden und bewerten können • Methoden und Algorithmen zur formalen Verifikation von Entwürfen verstehen und an Beispielen erläutern können • Probleme der Qualitätssicherung beim Systementwurf analysieren können • Aufgaben und Beispiele in den wöchentlichen Tutorien eigenständig präsentieren können
Lerninhalte	1. Entwurfsablauf 2. Hardware-Beschreibung durch VHDL 3. Verifikation 4. Formale Methoden 5. Graphenbasierte Funktionsdarstellung 6. Äquivalenzvergleich 7. Modellprüfung Aus der Übersicht lässt sich erkennen, dass ein überwiegender Teil der Vorlesung theoretisch/methodische Grundlagen behandelt. Insbesondere werden folgende theoretisch/methodischen Grundlagen im Zusammenhang dieser Inhalte behandelt: • Boolesche Funktionen und Boolesche Algebra • Datenstrukturen zur effizienten Repräsentation Boolescher Funktionen • effiziente Algorithmen zur Manipulation Boolescher Funktionen • Überführung von Systemen in automatentheoretische Modelle • Temporallogiken zur Beschreibung von Eigenschaften für die Modellprüfung • Erreichbarkeitsanalyse und Fixpunktiterationen in großen Zustandsräumen • Komplexitätstheoretische Betrachtung der Algorithmen

Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	• G. Hachtel, F. Somenzi, Logic Synthesis and Verification Algorithms, Kluwer Academic Publishers, 1996 • K.L. McMillan: Symbolic Model Checking, Kluwer Academic Publishers, 1993

Modulbezeichnung	**Spezifikation eingebetteter Systeme**
Modulverantwortliche(r)	Prof. Dr. J. Peleska
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Systemsoftware / Eingebettete Systeme, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Jahre
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Spezifikationsformalismen kennen und verstehen, die besonders für die Beschreibung von eingebetteten Steuerungssystemen mit Echtzeitbedingungen geeignet sind. • Semantische Grundlagen von Modellierungsformalismen für eingebettete Systeme verstehen. • Paradigmen (d.h. wiederkehrende Grundmuster) verstehen, nach denen typische Anforderungen an Echtzeitsysteme klassifiziert und beschrieben werden können. • Übersicht über die aktuellen Forschungsthemen auf diesem Gebiet haben. • Domänen-spezifische Beschreibungsformalismen entwerfen können und auf dieser Grundlage modell-basiert entwickeln können
Lerninhalte	Spezifikationsformalismen, Ausdrucksmächtigkeit, Semantik und Anwendung an Beispielen aus dem Gebiet Echtzeitsysteme: 1. Timed Automata, 2. Timed CSP, 3. Hybrid Statecharts für Systeme mit diskreten und analogen Steuerungsgrößen, 4. UML-Diagrammtypen mit Eignung für Echtzeitsysteme. 5. Domänen-spezifische Beschreibungsformalismen und ihre werkzeug-gestützte Anwendung 6. Modell-basierte Codegenerierung 7. Beschreibung von Modelleigenschaften mittels Temporallogik
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung

Literatur	• James Rumbaugh, Ivar Jacobson, Grady Booch: The Unified Modeling Language Reference Manual, Second Edition, Addison-Wesley Professional, 2004 • Steve Schneider: Concurrent and Real-Time Systems, John Wiley and Sons Ltd, 2000 • Juha-Pekka Tolvanen, Risto Pohjonen and Steven Kelly: Advanced Tooling for Domain-Specific Modeling: MetaEdit+ • Steven Kelly and Juha-Pekka Tolvanen: Domain-Specific Modeling - Enabling Full Code Generation. IEEE Computer Society Publications, John Wiley and Sons, (2008) • Rajeev Alur, David L. Dill: A Theory of Timed Automata, Theoretical Computer Science, Volume 126, No 2, 1994 • Zohar Manna, Amir Pnueli: The Temporal Logic of Reactive and Concurrent Systems, Specification, Springer, 1991

Modulbezeichnung	**Rechnernetze — Media Networking**
Modulverantwortliche(r)	Prof. Dr. U. Bormann
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Systemsoftware / Eingebettete Systeme
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten in jedem WiSe
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Rechnernetze
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• In der Terminologie des Fachgebiets Rechnernetze kommunizieren können, Systemkomponenten anhand dieser Terminologie klassifizieren können. • Lösungsvarianten für kommunikationstechnische Probleme bewerten können; insbesondere für die Vielzahl der behandelten Techniken (s. unten): Voraussetzungen erkennen, Aufwände abschätzen, Konfigurationen entwickeln und Einsatzgebiete (auch quantitativ) bewerten können. • Mechanismen der Marktdurchsetzung von technischen Spezifikationen verstehen und bewerten können. • Globale Strategien auf vorgegebene Einzelsituationen übertragen können.
Lerninhalte	• Mehrpunktkommunikation: Dienste, Routing, zuverlässiger Transport • Fernnetze: Übertragung und Vermittlung (SDH/ATM vs. MPLS/IP-Switching) • Funknetze: Übertragung und Vermittlung (Satellitenkommunikation, Mobilfunk, IoT, etc.) • Monomedia: Zeichen, Grafik, Bilder, Grafik, Video, Sprache • Protokollunterstützung für zeitabhängige Medienströme: RTP, QoS, Streaming • Anwendungen: Videokonferenzen, VoIP
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	• Andrew S. Tanenbaum: Computer Networks, 5th Edition, Pearson, 2010 (bzw. die deutsche Übersetzung: Computernetzwerke, 5. Auflage, Pearson Studium, 2012) • http://rfc-editor.org/rfc.html (für die Internet-Standarddokumente)

Modulbezeichnung	**Software-Reengineering**
Modulverantwortliche(r)	Prof. Dr. R. Koschke
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Systemsoftware / Eingebettete Systeme
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Software-Projekt
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	Die Studierenden verfügen über folgende Fachkompetenzen: • auf welchen Ebenen man Code analysieren kann, • wie man Schwachstellen des Codes auffindet, • wie man duplizierten Code automatisch aufspürt, • wie man Abhängigkeiten zwischen Anweisungen nachverfolgen kann • wie man Code-Muster findet, • wie man den Code automatisch transformieren kann, • wie man die Stellen im Code findet, die eine bestimmte Funktionalität implementieren, • wie man Vererbungshierarchien restrukturieren kann, • wie man Software visualisieren kann, • wie man Software-Architekturen rekonstruiert • wie man Reengineering-Projekte organisiert.

Lerninhalte	Software-Reengineering beschäftigt sich mit Wiedergewinnung verlorener Informationen über existierende Software-Systeme (Reverse Engineering), Restrukturierung der Beschreibung des Systems (Restructuring) und der nachfolgenden Implementierung der Änderungen (Alteration). Reengineering hat dabei nicht nur mit alter Software zu tun; gerade neuere objekt-orientierte Systeme erfordern oft schon bald eine Restrukturierung, weshalb sich ein guter Teil der Vorlesung speziell objekt-orientierter Software widmet (Restrukturierung von Klassenhierarchien, automatisches Refactoring). Auch im Kontext neuerer Ansätze des Software-Engineerings zur Entwicklung ähnlicher Produkte als Produktlinie findet Reengineering Einsatz. • allgemeiner Überblick über das Thema sowie Beziehung des Reengineerings zu verwandten Gebieten der Software-Wartung, Wrapping, etc. • Zwischendarstellungen für Programmanalysen (abstrakte Syntaxbäume, Program Dependency Graph, Static Single Assignment Form), Datenfluss-/Kontrollflussanalysen • Software-Metriken • Software-Architekturrekonstruktion: Reflexionsmethode, Software-Clustering, Symphony • Program Slicing • Klonerkennung • Mustersuche • automatische Code-Transformationen und Refactoring • Begriffsanalyse • Merkmalsuche • Analyse und Restrukturierung von Vererbungshierarchien • Software Visualisierung • Planung und Durchführung von Reengineering-Projekten, Prozessmodelle des Reengineerings Die Übungen dienen, neben der Wiederholung und praktischen Vertiefung des Vorlesungsinhalts, auch der Vorstellung existierender Reengineering-Werkzeuge. Die Vorlesung Software-Reengineering beschäftigt sich mit der Methodik des systematischen Informationengewinns über existierende Programme, die formale Repräsentation von Programmen sowie mit Methoden für semantikerhaltende Transformationen von Programmen. Die in der Vorlesung dargestellte formale Begriffsanalyse bildet eine mathematisch fundierte Methode zur Analyse verschiedener Relationen in Programmen, die auch in anderen Gebieten der Informatik eingesetzt werden kann.
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung

Literatur	Reengineering • Reengineering - Eine Einführung, Bernd Müller, B.G. Teubner Verlag Stuttgart, 1997 • Object Oriented Reengineering Patterns, Serge Demeyer, Stephane Ducasse, Oscar Nierstrasz, 2007. • Refactoring: Improving the Design of Existing Code, Martin Fowler, Addison-Wesley, 2000. • Modernizing Legacy Systems , Robert C. Seacord, Daniel Plakosh, and Grace A. Lewis. Addison-Wesley, 2003. • Anti Patterns: Entwurfsfehler erkennen und vermeiden, William J. Brown (Autor), Raphael C. Malveau, Mitp-Verlag; zweite überarbeitete Auflage, 2007. Wartung und Evolution • Legacy-Software, Dieter Masak, Springer Verlag, 2006. Prozesse und Management zur Wartung und Migration von Altsystemen. • Nutzung und Wartung von Software - Das Anwendungssystem-Management, Franz Lehner, Hanser Verlag, 1989. • Software-Produktmanagement: Wartung und Weiterentwicklung bestehender Anwendungssysteme Harry M. Sneed, Martin Hasitschka, Maria-Therese Teichmann, Dpunkt Verlag, 2004. • Software Evolution, Tom Mens, Serge Demeyer (Eds.), Springer Verlag, 2008. • Software-Wartung: Grundlagen, Management und Wartungstechniken, Christoph Bommer, Markus Spindler, Volkert Barr, DPunkt Verlag, 2008. • Practical Software Maintenance: Best Practices for Managing Your Software Investment, Thomas M. Pigoski, Wiley & Sons, 1996. Wartbarkeit • Code Quality Management: Technische Qualität industrieller Softwaresysteme transparent und vergleichbar gemacht, Frank Simon, Olaf Seng, Thomas Mohaupt, Dpunkt Verlag, 2006. • Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems von Michele Lanza und Radu Marinescu, Springer Verlag, 2006, ISBN-13 978-3540244295. Programmanalyse • Advanced Compiler Design and Implementation, Steven S. Muchnick, Morgan Kaufmann, 1997. • Principles of Program Analysis, Flemming Nielson, Hanne Riis Nielson, Chris Hankin, Springer Verlag, Auflage: 2., 2004. Software-Visualisierung • Software Visualization, Stephan Diehl, Springer Verlag, 2007. Debugging • Why Programs Fail: A Guide to Systematic Debugging, Andreas Zeller, Dpunkt Verlag, 2005.

Modulbezeichnung	**Testautomatisierung**
Modulverantwortliche(r)	Prof. Dr. J. Peleska
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Systemsoftware / Eingebettete Systeme, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Jahre
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Grundlagen von Test und Verifikation
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	Die Studierenden verfügen über ein vertieftes Verständnis für • Testfallentwurf • Bezug zwischen Anforderungen und Testfällen • Modell-basierte Testfallerzeugung • Algorithmen für die automatische Testfall-/Testdatenerzeugung • Äquivalenz zwischen erschöpfenden Tests und Korrektheitsbeweis
Lerninhalte	1. Vorgehensmodelle und Testprozess 2. Testarten auf unterschiedlichen Systemebenen 3. Modell-basiertes Testen - die W-Methode von Chow 4. Strukturelles Testen 5. Modell-basiertes Testen von Echtzeitsystemen 6. Spezialthemen aus den Gebieten • SMT-Solver für die Berechnung konkreter Testdaten • Äquivalenzklassentests für nebenläufige Echtzeitsysteme • Überdeckungskriterien und ihr Bezug zum Korrektheitsbeweis • Mutationstests
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung

Literatur	• R. Binder “Testing Object-Oriented Systems: Models, Patterns, and Tools”, Addison-Wesley, 2000 • A. Spillner, T. Linz “Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified-Tester”, dpunkt-Verlag, 2003. • J. Peleska und M. Siegel “Test Automation of Safety-Critical Reactive Systems”, South African Computer Journal, No. 19, pp. 53-77, 1997. • J. Peleska “Formal Methods and the Development of Dependable Systems”, Habilitationsschrift, Bericht Nr. 9612, Dezember 1996, Institut für Informatik und praktische Mathematik, Christian-Albrechts-Universität Kiel, 1997. • Tsun S. Chow “Testing Software Design Modeled by Finite-State Machines”, IEEE Transactions on Software Engineering, SE-4(3), pp. 178-186, März 1978.

Modulbezeichnung	**Grundlagen der Sicherheitsanalyse und des Designs**
Modulverantwortliche(r)	Prof. Dr. D. Hutter
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik, Systemsoftware / Eingebettete Systeme, Produktionstechnik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. jedes Jahr
Voraussetzung für die Teilnahme	Keine ☐ Folgende: Inhaltliche Voraussetzungen: Kenntnisse in formalen Methoden bzw. Informationssicherheit sind nützlich aber nicht zwingend erforderlich
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Verfahren der (formalen) Modellierung von (Informations)Sicherheitsanforderungen und Sicherheitsmechanismen kennen • Verschiedene Sicherheitsanalysetechniken einschätzen und bewerten können • Die Modellierungstiefe und deren Auswirkungen auf die Analyse einschätzen und bewerten können • Das Zusammenspiel von verschiedenen Sicherheitsanforderungen und -garantien verstehen
Lerninhalte	Grundlagen der Modellierung im Bereich der Informationssicherheit Design und Analyse von Sicherheitsprotokollen • Modellierung eines Angreifers • Prinzipien des Designs von Sicherheitsprotokollen • Analyse und Verifikation von Sicherheitsprotokollen Design und Analyse von Sicherheitspolitiken • Modellierung (formaler) Sicherheitspolitiken • Grundlagen der Informationsflusskontrolle, Vertraulichkeit und Integrität als Informationsflusseigenschaften • Zustandsbasierte Informationsflusskontrolle • sprachbasierte Informationsflusskontrolle und Programmanalyse • Realisierung von Informationsflusskontrolle durch Zugriffskontrolle Komposition verschiedener Sicherheitsmechanismen am Beispiel des Semantic Web
Prüfungsformen	Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung

Literatur	Skript bzw. Folien Dieter Gollmann: Computer Security, Wiley&Sons, 2006 Matt Bishop: Computer Security, Art und Science, Addison Wesley, 2003 Diverse Fachartikel

Modulbezeichnung	**Systeme hoher Sicherheit und Qualität**
Modulverantwortliche(r)	Prof. Dr. J. Peleska
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Systemsoftware / Eingebettete Systeme, Produktionstechnik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten in jedem WiSe
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Technische Informatik 2
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Entwicklung eines Grundverständnisses für Systemsicherheit (Safety&Security); • Verständnis der rechtlichen Grundlage, Normen und Standards bei der Entwicklung solcher Systeme; • Grundlegende Techniken zur Entwicklung sicherheitskritischer Systeme beherrschen und anwenden können. Dazu zu zählen formale Modellierungssprachen zur Spezifikation von Eigenschaften, und Verifikationsmethoden wie Test, statische Programmanalyse, Programmverifikation und Modelchecking.
Lerninhalte	• Der Begriff der Zuverlässigkeit (Dependability); • Aspekte des Qualitätsbegriffes; • Rechtliche Aspskte, Normen und Standards wie die funktionale Sicherheitsnorm IEC 61508 und die Common Criteria IEC 15408; • Softwareentwickoungsmodelle, Gefährdungsanalysen; • Klassifikation von Security-Attacken; • Formale Modellierung mit SysML und OCL; • Verifikationstechniken: Test, statische Programmanalyse, formale Verifikation, Modelchecking
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	• D. Smith & K.G.L. Simpson: Functional Safety. Elsevier, 2001 • Nancy G. Leveson: SAFEWARE: SYSTEM SAFETY AND COMPUTERS. Addison-Wesley ISBN: 0-201-11972-2. • N. Storey: Safety-Critical Computer Systems. Addison Wesley Longman 1996. • Dieter Gollmann: Computer Security, 2nd edition, Wiley and Sons, 2006 • Edmund M. Clarke, Orna Grumberg and Doron A. Peled: Model Checking, The MIT Press, 1999

Modulbezeichnung	**Agile Web-Entwicklung**
Modulverantwortliche(r)	Prof. Dr. C. Bormann
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Systemsoftware / Eingebettete Systeme
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 144 h vorbereitender Übungsbetrieb 36 h Summe 180 h
Turnus des Moduls	unregelmäßig
Voraussetzung für die Teilnahme	Keine ☐ Folgende: Formale Voraussetzungen: KeineInhaltliche Voraussetzungen: Fähigkeit zum Programmieren
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	Die Studierenden: • verstehen die Prinzipien Agiler Entwicklung und können diese in einem realistischen, kundenorientierten Projekt einsetzen • beherrschen die Grundlagen Web-basierter Anwendungssysteme und können moderne Architekturprinzipien anwenden • beherrschen moderne Werkzeuge, die bei der effizienten und agilen Entwicklung solcher Systeme heute eingesetzt werden • können Vor- und Nachteile verschiedener Frameworks, Methoden, Werkzeuge, und Komponenten in diesem Bereich einschätzen und in konkreten Projekten bewerten • können dynamische Programmiersprachen in realistischen Projekten einsetzen und verstehen ihren sinnvollen Einsatzbereich

Lerninhalte	Werkzeuge und Komponenten, sowie Entwicklungsmethoden: 1. Dynamische Programmiersprachen, Programmiersprache Ruby 2. Grundlagen und Standards Web-basierter Anwendungen: • Webstandards (HTML/HTML5, CSS, JavaScript) • Stukturen von Web-Anwendungen (HTTP; MVC und verwandte Modelle) • REST als Architekturprinzip • Ajax: Techniken, Einsatzbereich, Risiken 3. Framework Ruby on Rails, dabei u.a.: • DSL-Konzepte in dynamischen Programmiersprachen • Open-Source-Ökosystem 4. Versionskontrolle dritter Generation (Werkzeug: git) 5. Grundlagen der Agilen Entwicklung 6. Organisation Agiler Entwicklung; Iterationen; Einbindung von Stakeholdern 7. Werkzeuge zur Erhaltung der technischen Agilität, u.a.: • Don't repeat yourself (DRY) und Metaprogrammierung • Testgetriebene Entwicklung (TDD) 8. Grundlagen der Agilen Anwendungssicherheit
Prüfungsformen	Bearbeitung von Projektaufgaben, Präsentation und Fachgespräch
Literatur	• Agile Web Development with Rails, 4th Edition • The Rails 3 Way

Modulbezeichnung	**Informationssicherheit — Prozesse und Systeme**
Modulverantwortliche(r)	Prof. Dr. C. Bormann
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Systemsoftware / Eingebettete Systeme
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Informationssicherheit
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	Studierende: • haben vertiefte Kenntnisse in der Sicherung komplexer soziotechnischer Systeme • können komplexe kryptographische Sicherheitsprotokolle bewerten und in ihrem Einsatzbereich weiterentwickeln • verstehen Sicherheit als Prozess mit ihren technischen und nicht-technischen Komponenten • kennen wichtige Sicherheitsprozesse, so wie sie heute in ISMS eingesetzt werden, und können diese weiterentwickeln

Lerninhalte	Systeme: 1. Fortgeschrittene Anwendung von Kryptographie • ECC und seine Varianten • Lebenszyklus kryptographischer Verfahren; Stand aktueller Verfahren • Zero-Knowledge-Protokolle, Zero-Knowledge-Password-Proof • Zertifikate, Beweiswerterhaltung/LTANS • Composability von Sicherheitsprotokollen • Browserbasierte Sicherheitsprotokolle (SAML/Liberty, OpenID, OAuth) 2. Grundlagen manipulationssicherer Systeme (tamperproof systems) Prozesse: 1. Softwaresicherheit • Sicherheit im Software-Lifecycle • Statische Analyse, Symbolic Execution, Fuzzers usw. 2. Security Management • Awareness • Incident-Response • Logging/Auditing 3. Risk-Assessment • Risiko-Wahrnehmung • Qualitative und quantitative Modelle • Insider-Threat-Modelle 4. Security Usability • Usability als Sicherheitsfaktor • Benutzbare Autorisierung
Prüfungsformen	In der Regel Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	

Modulbezeichnung	**Entwurf eingebetteter Systeme mit Digitallogik**
Modulverantwortliche(r)	PD Dr. St. Bosse
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik, Systemsoftware / Eingebettete Systeme, Produktionstechnik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Verständnis der anwendungsspezifischen Digitallogik für den Hardware-Entwurf als Erweiterung und Ergänzung zum Software-Entwurfs • Grundlegende Kentnisse der Funktionsweise von Digitallogiksystemen • Entwurf und Abbildung von Schaltnetzen auf boolesche Algebra • Kenntnisse über Optimierung von Digitallogiksystemen • Einführung der Register-Transfer-Logik Architektur als wesentliche Architektur und Entwurfsmethode für die Datenverarbeitung • Abbildung von klassischen Programmen auf RTL mit Daten- und Kontrollpfadpartitionierung • Kenntnisse über programmierbare Digitallogikschaltungen (CPLD/FPGA/ASIC) • Fähigkeit zum Modellieren von Digitallogiksystemen und Abbildung von Algorithmen auf RT-Ebene sowie mit der Hardware-Beschreibungssprache VHDL • Aufzeigen der Möglichkeiten der Parallelisierung von Algorithmen durch Digitallogiksysteme • Der Übungsanteil soll die praktische Umsetzung des in der Vorlesung erworbenen Wissens vermitteln und deren Anwendung an Beispeieln üben (z.B. Algorithmen auf RTL abbilden mit Verwendung des ReTrO Simulators)

Lerninhalte	● Digitallogik, Boolesche Algebra, Boolesche Funktionen ● Konjunktive- und Disjunktive Normalformen, Ableitungen aus Schaltbedingungen ● Technologische Umsetzung mit Transistoren ● Darstellung von booleschen Funktionen und Schaltnetzen mittels grafischer Methoden und Optimierung (KV-Diagramme) ● Systematische Darstellung und Optimierung von booleschen Funktionen mittels Binary Descision Diagrams (BDD) ● Programmierbare Digitallogik für Rapid Prototyping: Systematik und Aufbau Abbildung von Und-Oder-Matrizen auf verschiedene Technologien: RAM/PAL/GAL/CPLD/FPGA/ASIC ● Verwendung von hoch-integrierten Field-Programmable-Gate-Arrays (FPGA) ● Standardzellen-ASIC: Architektur unf Entwurfsmethoden ● Hardware-Entwurfsmethodik und Syntheseverfahren im Überblick, Ebenen des Logikentwurfs ● Kombinatorische Logiksysteme ● Sequenzielle Logiksysteme ● Systementwurf mit Register-Transfer-Logik (RTL) Architekturen ● Abbildung von Algorithmen auf Daten- und Kontrollpfade und Umsetzung mittels RTL (+ Scheduling & Allokation des Datenpfades) ● Laufzeitprobleme in elektronischen Systemen oder warum die Formale Verifikation nur graue Theorie sein kann ● Zustandsautomaten (Moore- und Meleay) und ihre Anwendung ● Beschreibung und Modellierung von Digitallogiksystemen mittels einer Hardware-Beschreibungssprache (VHDL)
Prüfungsformen	Erfolgreiche Bearbeitung von Übungsaufgaben und mündliche Prüfung
Literatur	1. Stefan Bosse Anwendungsspezifische (programmierbare) Digitallogik und VHDL-Synthese Skript, 3. Auflage (2012) 2. Michael D. Ciletti Advanced Digital Design with the Verilog VHDL Prentice Hall, (2003) 3. J. Reichardt, B. Schwarz VHDL-Synthese Oldenbourg Verlag (2003)

Modulbezeichnung	**Parallele und verteilte eingebettete Systeme**
Modulverantwortliche(r)	PD Dr. St. Bosse
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik, Systemsoftware / Eingebettete Systeme, Produktionstechnik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	● Verständnis der Funktionsweise und Entwurf von paralleler Datenverarbeitung ● Verständnis von parallelen Programmen und Rechnerarchitekturen ● Klassische Parallelrechner-Architekturen sollen auf Hardware-Ebene abgebildet und skaliert werden können ● Einsatz klassischer Multi-Prozeß-Modelle mit Interprozeß-Kommunikation für die Abbildung und Synthese von Algorithmen auf Hardware ● Verständnins und Anwendung von Kommunikation und Synchronisation in parallelen und verteilten Systemen ● Abbildung von Kommunikation auf Schaltkreise ● Verständins von System-On-Chip (SoC) Lösungen ● High-level Syntheseverfahren auf Programmiersprachenebene als zukunftsfähiges Entwurfswerkzeug für komplexe SoC ● Praktische Anwendung der Vorlesungsinhalte in der Übung (Grundlagen des Entwurfs von nebenläufigen Prozessen und Datenverarbeitung sowie Kommunikation mit Simulator CPV und Multi-Agenten Simulator SeSaM

Lerninhalte	• Multiprozeß-Modelle (Multi-Threading) bei generischen Prozessoren und Skalierung auf anwendungsspezifische Logiksysteme • Multiprozeß-Architekturen (Parallel-Rechner) mit generischen Prozessoren und Skalierung auf RTL und anwendungsspezifische Logiksysteme • Interprozeß-Kommunikation {Mutex, Semaphore, Event, Queue, Barrier, Channel} in Software und Abbildung auf RTL und Hardware-Ebene • Parallele Algorithmen in Soft- und Hardware • Parallel-Architekturen in Hardware: RTL, SoC und FPGAs • Netzwerkstrukturen und Topologien, adapdiert für SoC-Entwürfe • Logik- und algorithmische Highlevel-Synthese-Verfahren • Pipeline-Architekturen in funktionalen und reaktiven Systemen
Prüfungsformen	Erfolgreiche Bearbeitung von Übungsaufgaben und mündliche Prüfung
Literatur	• Stefan Bosse: Hardware-Entwurf von parallelen Systemen, Logik- & High-Level-Synthese, Skript, 2. Auflage (2013) • David C. Ku & Giovanni De Micheli: High Level Synthesis Under Timing and Synchronization Constraints, Kluwer, (1992)

Modulbezeichnung	**Test von Schaltungen und Systemen**
Modulverantwortliche(r)	Prof. Dr. R. Drechsler
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Mechatronik, Systemsoftware / Eingebettete Systeme, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i.d.R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Das Problem des Testens verstehen und erklären können • Den Testverlauf für Schaltungen und Systeme kennen und anwenden können • Klassische und moderne Testverfahren kennen und anwenden können • Die Algorithmen auf (Schaltkreis-)Graphen anwenden können • Die Komplexität der Verfahren verstehen und erklären können
Lerninhalte	1. Physikalische Fehlerursachen 2. Abstraktion von der physikalischen Ebene, Fehlermodelle 3. Algorithmen zur Berechnung von Signalwahrscheinlichkeiten 4. Techniken zur Manipulation Boolescher Funktionen 5. Algorithmen zur Fehlersimulation 6. Algorithmen zur Testmustergenerierung 7. Nutzung strukturellen Wissens zur Effizienzsteigerung 8. Techniken zur Reduktion des Suchraumes, Fehleräquivalenz und -dominanz Aus den Inhalten ist deutlich zu erkennen, dass theoretisch/methodische Grundlagen einen wichtigen Teil dieser Vorlesung darstellen. Darüber hinaus werden für die vorgestellten Verfahren die Komplexitäten hinsichtlich Laufzeit und Speicher betrachtet.
Prüfungsformen	i. d. R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung

Literatur	• M.L. Bushnell, V.D. Agrawal: Essentials of Electronic Testing – for Digital, Memory & Mixed-Signal VLSI Circuits, New York: Springer, 2000. • N. Jha, S. Gupta: Testing of Digital Systems, Cambridge University Press, 2003. • A. Miczo: Digital Logic Testing and Simulation, 2. Auflage, Wiley, 2003. • H. Wojtkowiak: Test und Testbarkeit digitaler Schaltungen, Teubner, 1988. • H.-J. Wunderlich: Hochintegrierte Schaltungen: Prüfgerechter Entwurf und Test, Berlin: Springer, 1991.

Modulbezeichnung	**Integrated Intelligent Systems**
Modulverantwortliche(r)	Michael Beetz
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik, Systemsoftware / Eingebettete Systeme, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	einjährig
Voraussetzung für die Teilnahme	Keine ☐ Folgende: Formale Voraussetzungen: KeineInhaltliche Voraussetzungen: Kenntnisse der Grundlagen der Künstlichen Intelligenz (BB-710.01)
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	Die Vorlesung beschäftigt sich mit aktuellen Techniken zur Implementierung von technischen kognitiven Systemen, das heißt mit intelligenten Computersystemen, die über Sensoren und Aktuoren verfügen. Solche Systeme werden vor allem in Bereichen wie der Service-Robotik, in autonomen Raumsonden, in intelligenten Wohn- und Arbeitsbereichen und in Fahrerassistenzsystemen eingesetzt.
Lerninhalte	Es werden folgende Themen behandelt: Sensoren, Aktuatoren und physikalische Infrastrukturen von technischen kognitiven Systemen (u.a. Smart Sensors, Sensornetzwerke); Berechnungsmodelle zur Steuerung technischer kognitiver Systeme: dynamisches Systemmodell, rationales Agentenmodell, das Berechnungsmodell der technischen kognitiven Systeme; Grundlagen probabilistischer Zustandsschätzung: Bayes-Filter, Kalman-Filter, Partikel- Filter, Mechanism en zur Datenassoziation, Lernen von Sensor- und Aktionsmodellen, Hidden Markov Modelle, Expectation Maximization; Anwendungen probabilistischer Zustandsschätzung: Selbstlokalisierung, Umgebungskartierung, Objektverfolgung; Programmiermethoden für technische kognitive Systeme: nebenläufig reaktive Steuerungsmechanismen; Wissens- und planbasierte Steuerungstechniken
Prüfungsformen	Eine Auswahl aus mündlicher Prüfung, Klausur und Übungen mit Fachgespräch.
Literatur	

Modulbezeichnung	**Massively Parallel Algorithms**
Modulverantwortliche(r)	Prof. Dr. G. Zachmann
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Systemsoftware / Eingebettete Systeme, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☐ Folgende: Inhaltliche Voraussetzungen: Algorithmisches Denken. Gewisse Programmierfähigkeiten in C (empfohlen wird das "Propädeutikum C/C++")
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐

Lernziele	Die Ära der single-core Prozessoren ist zu Ende. Inzwischen gibt es neue, massiv-parallele Prozessoren (GPUs), die hunderte bis tausende von Threads parallel abarbeiten können. Diese entwickeln sich zur Zeit als Co-Prozessoren, die große Teile der Berechnung den (multi-core) CPUs abnehmen. Möglicherweise werden sich GPUs als neue Architektur für die Haupt-Prozessoren – gerade auch auf mobilen Geräten – etablieren, da diese mehr Compute-Power pro Energieeinheit bieten. Die große Zahl von parallelen Cores stellt das Design von Algorithmen und Software allerdings vor neue Herausforderungen, damit diese von der großen Parallelität profitieren können. Das Hauptziel dieser Vorlesung ist es, Studenten in die Lage zu versetzen, für solch massiv-parallele Hardware Algorithmen zu entwerfen. Simulation wird inzwischen gemeinhin als die dritte Säule der Wissenschaft angesehen (neben den Experimenten und der Theorie). In der Simulation wird ein ständig wachsender Bedarf an Rechenleistung benötigt; gerade diese wird aber durch die Verfügbarkeit von GPUs fast schon zu einer Commodity auf dem Desktop. Daher gibt es viele wissenschaftliche Bereiche, in denen Studenten das Wissen, das sie in dieser Vorlesung erwerben, gewinnbringend einsetzen können, wie z.B.: • Computer science (e.g., visual computing, database search) • Computational material science (e.g., molecular dynamics simulation) • Bio-informatics (e.g., alignment, sequencing, . . .) • Economics (e.g., simulation of financial models) • Mathematics (e.g., solving large PDEs) • Mechanical engineering (e.g., CFD and FEM) • Physics (e.g., ab initio simulations) • Logistics (e.g. simulation of traffic, assembly lines, or supply chains) Am Ende dieser Vorlesung werden Studenten • aktive Erfahrungen bei der Entwicklung von Software und Algorithmen für massiv-parallele Architekturen gesammelt haben; • eine Anzahl von massiv-parallelen Algorithmen-Patterns kennen; • in der Lage sein, eigene massiv-parallele Algorithmen zu entwickeln; • CUDA kennen. In der ersten Hälfte der Vorlesung werden Studenten sich anhand von kleinen und mittelgroßen Übungen und Frameworks mit der parallelen Programmier-Umgebung CUDA vertraut machen. In der zweiten Hälfte werden Studenten an einem eigenen Projekt arbeiten.
Lerninhalte	Diese Vorlesung führt Studenten in die grundlegenden und einige fortgeschrittene Methoden und Techniken der massiv-parallelen Algorithmen ein. Einige der vorgesehenen Themen sind: • die Programmierumgebung CUDA C; • die Speicher-Hierarchie und verschiedene Speicher-Charakteristiken; • die GPU Architektur • parallele Reduktion; • coalesced memory access; • massiv-parallele Matrix-Algorithmen; • Prefix-Sum und deren Anwendungen in der Bildverarbeitung; • Textur-Filterung; • Paralleles Sortieren (odd-even, bitonic, adaptive bitonic); • Bildverarbeitung; • Thrust;
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung

Literatur	• Jason Sanders, Edward Kandort: CUDA by Example. Addison-Wesley, Pearson Education. • Wen-Mei W. Hwu: GPU Computing Gems Jade Edition. Morgan Kaufmann. • David B. Kirk, Wen-Mei W. Hwu: Programming Massively Parallel Processors. Morgan Kaufmann. • NVidia: CUDA C Programming Guide.

Modulbezeichnung	**Material-integrierte Sensorische Systeme**
Modulverantwortliche(r)	PD Dr. Stefan Bosse
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik, Systemsoftware / Eingebettete Systeme, Produktionstechnik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb / Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	jedes WS
Voraussetzung für die Teilnahme	Keine ☐ Folgende Formale Voraussetzungen: Keine
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	Die Teilnahme an der Veranstltung soll Studenten interdisziplinär einen system-orientierten Zugang für die Modellierung, den Entwurf und die Anwendung von material-eingebetteten oder material-applizierten Sensorischen Systemen bieten, die aufgrund der technischen Realisierung und des Einsatzes spezielle Anforderungen an die Datenverarbeitung stellen und ein Verständnis des Gesamtsystems (inlkusive Aspekte der Materialwissenschaften und Technologien) voraussetzen. Diese neuen Sensorischen Materialen finden z. B. in der Robotik (Kognition) oder in der Produktionstechnik für die Materialüberwachung Anwendung. Folgende Kompetenzen sollen erworben werden: 1. Grundverständnis des technischen Aufbaus und der Funktionweise von Sensorischen Materialien: Elektronische Signalverarbeitung von Sensoren, Mechanisches Verhalten, Einfluss von Sensoren und Elektronik auf mechanische Eigenschaften des Trägermaerials 2. Datenverarbeitung in Sensornetzwerken unter harten Randbedingungen wie limitierten Energieangebot, Rechenleistung und Speicher, Fehleranfälligkeit 3. Parallele und verteilte Datenverarbeitung geeignet für low-ressource Sensornetzwerke: Architekturen, Kommunikation, Kooperation, Wettbewerb um Ressourcen 4. Grundlagen der Robustheit, Fehlernanalyse, Redundanz in solchen Sensornetzwerken

Lerninhalte	A. Einführung in die Thematik, was ist Sensorik, warum und wofür material-integrierte Sensorik, Sensorische Materialien, Anwendungen in der Strukturüberwachung und Robotik B. Systemebene und System Entwicklung, Anforderungen, Entwurfsmethodiken, Test & Simulation, Normen und Standards C. Grundlagen von Sensoren und Elektronik, Sensor- und Elektronikentwicklung für die Materialintegration D. Sensornetzwerke und Sensorknoten, Grundlagen, Metriken, Entwicklung E. Schnittstellen Technologien: Material - Sensor - Elektronik - Systeme (Vernetzung) F. Signal- und Datenverarbeitung in Sensornetzwerken (Knotenebene) G. Kommunikation in Sensornetzwerken (Netzwerkebene) H. Energie-Versorgung und Management in material-integrierten Sensornetzwerken und Energiegewinnung (Harvesting) I. Applikationen (Robotik, Strukturüberwachung usw.)
Prüfungsformen	Erfolgreiche Bearbeitung von Übungszetteln + Mündliche Prüfung
Literatur	

Modulbezeichnung	**Applied Computational Engines**
Modulverantwortliche(r)	Rüdiger Ehlers
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Systemsoftware / Eingebettete Systeme
Dauer des Moduls	1 Semester
Kreditpunkte	4 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 42 h Übungsbetrieb/Prüfungsvorbereitung 78 h Summe 120 h
Turnus des Moduls	Bei Interesse in jedem Sommersemester
Voraussetzung für die Teilnahme	Keine ☐ Folgende: Formale Voraussetzungen: KeineInhaltliche Voraussetzungen: Basic theoretical computer science and moderate proficiency of some programming language (for the practical exercises)
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	To be able to identify when difficult computational problems that can occur in the computer scientist's working life can be solved by standard computational engines. To know the strenghts and limits of a diverse set of computational engines, such as SAT solving, QBF solving, and linear programming. To be able to apply some commonly used computational engines to a wide variety of decision and optimization problems.
Lerninhalte	Topics include: • SAT Solving (Basic algorithms for SAT solving: unit propagation, backtracking, variable selection, and learning; Tseitin encoding and alternatives; SAT encodings in practice; Theory of tractability: "Backdoors") • Quantified Boolean Formula (QBF) solving • Integer Linear Programming (ILP) and Linear Programming (LP) as an "easy" subset (Definitions & encodings, Extension: Quadratic programming) • SMT solving (Basic idea and algorithms, SMT encodings of complex problems) • Supporting the encoding of difficult problems (Delta debugging & fuzz testing) • BDDs • Maximum flow algorithms & their applications • Automata for PSPACE-complete problems • Sub-engineering problems (clustering, . . .) • Robust problem solving: games of infinite duration • Applied branch-and-bound
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung

Literatur	• Armin Biere, Marijn Heule, Hans van Maaren, Toby Walsh (eds.): Handbook of Satisfiability, IOS Press, 2009 • Donald E. Knuth: The Art of Computer Programming (Volumes 1-4A), Addison Wesley, 2014 • Jon Kleinberg, Eva Tardos: Algorithm Design, 2006

Modulbezeichnung	**Real-time Operating Systems Development**
Modulverantwortliche(r)	Jan Peleska
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Systemsoftware / Eingebettete Systeme, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 0 h Übungsbetrieb/Prüfungsvorbereitung 180 h Summe 180 h
Turnus des Moduls	every year
Voraussetzung für die Teilnahme	Keine ☐ Folgende: Formale Voraussetzungen: KeineInhaltliche Voraussetzungen: Good programming skills in C are mandatory. A thorough understanding of basic operating systems concepts is very helpful for this lecture.
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	Students 1) know how to program a real-time application from scratch on “bare-metal”, that is, WITHOUT a supporting operating systems 2) know how to design an elegant real-time operating system kernel from scratch 3) understand the right balance between architectural beauty and optimised performance 4) know about basic benchmarks assessing the real-time capabilities of an RTOS 5) know how to do practical real-time application programming and RTOS development from scratch on a simple ARM-based computer architecture (BeagleBone Black)
Lerninhalte	. 1) Bare-metal programming on BeagleBode Black boards using the Code Composer Studio development environment (Eclipse-based) 2) The State Machine programming paradigm with cooperative multi-tasking, scheduling, watchdog monitor 3) Periodic time-controlled activities 4) Simple context switching: Programming user threads and associated schedulers 5) Inspiration from micro kernels: RTOS architecture with communication channels and ports 6) Filtered and prioritised real-time port handling 7) Real-time synchronisation mechanisms 8) Time-triggered versus event-based RTOS paradigms 9) RTOS Benchmarks
Prüfungsformen	Oral module examination or Exercises and oral technical discussion (Fachgespräch)

Literatur	• Wang, K.C. Embedded and Real-Time Operating Systems. DOI 10.1007/978-3-319-51517-5_2. Springer 2017. • Kopetz, H. Real-Time Systems: Design Principles for Distributed Embedded Applications. Second edition. Springer 2011. • Walls, C. Building a Real-Time Operating system. Rtos from the ground up. Elsevier Science & Technology 2007. • Cooling, J. Real-time Operating Systems Book 1. The Theory. Lindentree Associates, 2017. • Cooling, J. Real-time Operating Systems Book 2. The Practice. Lindentree Associates, 2017.

Modulbezeichnung	**Testautomatisierung**
Modulverantwortliche(r)	Prof. Dr. J. Peleska
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Systemsoftware / Eingebettete Systeme, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Jahre
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Grundlagen von Test und Verifikation
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	Die Studierenden verfügen über ein vertieftes Verständnis für • Testfallentwurf • Bezug zwischen Anforderungen und Testfällen • Modell-basierte Testfallerzeugung • Algorithmen für die automatische Testfall-/Testdatenerzeugung • Äquivalenz zwischen erschöpfenden Tests und Korrektheitsbeweis
Lerninhalte	1. Vorgehensmodelle und Testprozess 2. Testarten auf unterschiedlichen Systemebenen 3. Modell-basiertes Testen - die W-Methode von Chow 4. Strukturelles Testen 5. Modell-basiertes Testen von Echtzeitsystemen 6. Spezialthemen aus den Gebieten • SMT-Solver für die Berechnung konkreter Testdaten • Äquivalenzklassentests für nebenläufige Echtzeitsysteme • Überdeckungskriterien und ihr Bezug zum Korrektheitsbeweis • Mutationstests
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung

Literatur	• R. Binder “Testing Object-Oriented Systems: Models, Patterns, and Tools”, Addison-Wesley, 2000 • A. Spillner, T. Linz “Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified-Tester”, dpunkt-Verlag, 2003. • J. Peleska und M. Siegel “Test Automation of Safety-Critical Reactive Systems”, South African Computer Journal, No. 19, pp. 53-77, 1997. • J. Peleska “Formal Methods and the Development of Dependable Systems”, Habilitationsschrift, Bericht Nr. 9612, Dezember 1996, Institut für Informatik und praktische Mathematik, Christian-Albrechts-Universität Kiel, 1997. • Tsun S. Chow “Testing Software Design Modeled by Finite-State Machines”, IEEE Transactions on Software Engineering, SE-4(3), pp. 178-186, März 1978.

Modulbezeichnung	**Grundlagen der Sicherheitsanalyse und des Designs**
Modulverantwortliche(r)	Prof. Dr. D. Hutter
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik, Systemsoftware / Eingebettete Systeme, Produktionstechnik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. jedes Jahr
Voraussetzung für die Teilnahme	Keine ☐ Folgende: Inhaltliche Voraussetzungen: Kenntnisse in formalen Methoden bzw. Informationssicherheit sind nützlich aber nicht zwingend erforderlich
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Verfahren der (formalen) Modellierung von (Informations)Sicherheitsanforderungen und Sicherheitsmechanismen kennen • Verschiedene Sicherheitsanalysetechniken einschätzen und bewerten können • Die Modellierungstiefe und deren Auswirkungen auf die Analyse einschätzen und bewerten können • Das Zusammenspiel von verschiedenen Sicherheitsanforderungen und -garantien verstehen
Lerninhalte	Grundlagen der Modellierung im Bereich der Informationssicherheit Design und Analyse von Sicherheitsprotokollen • Modellierung eines Angreifers • Prinzipien des Designs von Sicherheitsprotokollen • Analyse und Verifikation von Sicherheitsprotokollen Design und Analyse von Sicherheitspolitiken • Modellierung (formaler) Sicherheitspolitiken • Grundlagen der Informationsflusskontrolle, Vertraulichkeit und Integrität als Informationsflusseigenschaften • Zustandsbasierte Informationsflusskontrolle • sprachbasierte Informationsflusskontrolle und Programmanalyse • Realisierung von Informationsflusskontrolle durch Zugriffskontrolle Komposition verschiedener Sicherheitsmechanismen am Beispiel des Semantic Web
Prüfungsformen	Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung

Literatur	Skript bzw. Folien Dieter Gollmann: Computer Security, Wiley&Sons, 2006 Matt Bishop: Computer Security, Art und Science, Addison Wesley, 2003 Diverse Fachartikel

Modulbezeichnung	**KI – Wissensakquisition und Wissensrepräsentation**
Modulverantwortliche(r)	Prof. M. Beetz, PhD
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten in jedem WiSe
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Die fortgeschrittenen Verfahren, Methoden und Ansätze der Künstlichen Intelligenz praktisch anwenden können • Fachliche Kompetenz insbesonders, aber nicht ausschließlich, in den Gebieten Ontologien, Verstehen und Parsen natürlicher Sparache und Multiagenten-Systeme • Die Terminologie des Fachgebietes beherrschen • Die einzelnen fortgeschrittenen Methoden/Ansätzen der KI in den Gesamtkontext einordnen können • Das Fachgebiet (oder Teile des Fachgebietes) im Kontext zu anderen Disziplinen einordnen önnen • Fortgeschrittene Verfahren auf einzelne konkrete Aufgabensituationen übertragen und diese lösen können

Lerninhalte	Wissensakquisition entspricht in weiten Grenzen der Systemanalyse, wie sie aus dem Software Engineering bekannt ist. So beschäftigt sich die Wissensakquisition damit, wie in Organisationen bestimmte Aufgaben so definiert werden können, daß sie z. B. einer maschinellen Bearbeitung zugänglich sind. Es ist schon lange bekannt, daß das früher benutzte einfache Bild der “Informationsextraktion” aus den Experten nicht trägt: es geht hier um einen modellbasierten Prozeß, der das zu nutzende Wissen zuerst verbal, dann semiformal und schließlich formal dargestellt, um die Kluft zwischen dem Expertenwissen und einer letzendlich in einer formalen Programmiersprache fixierten Anwendung schließen zu können. In diesem Kontext spielt eine implementierungs-unabhängige Wissensrepräsentation, die es erlaubt, statisches und dynamisches Wissen auf mehreren Ebenen zu formulieren und (mindestens) zu validieren, eine große Rolle. Modellierung komplexer bzw. realer Anwendungen erfordert zumeist eine Abbildung auf verteilte Systeme. Die Ausrichtung der Veranstaltung beinhaltet die Nutzung von aktuellen Werkzeugen, die für die einzelnen Lehrgebiete erhältlich und repräsentativ sind. Die Lehrinhalte sollen insbesondere Bezug zum Stand der Forschung aufweisen. Die Inhalte sind im Einzelnen: • Wissensakquisition (Maschinelle Lernverfahren, Data Mining) • Wissensrepräsentation (Formale Ontologien, spezielle Entscheidungsverfahren) • Verteiltes Wissen (Intelligente Agenten und Multiagentensysteme) Theoretisch/methodische Inhalte nehmen etwa die Hälfte des Semesters ein und behandeln insbesondere die folgenden Themen: • Wissensakquisition (Data Mining und C4.5) • Wissensrepräsentation (Formale Ontologien, Beschreibungslogiken) • Verteiltes Wissen (ACL und KQML als Agentenkommunikationssprachen, Konflikte bei Agentenkommunikation
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben sowie Fachgespräch oder mündliche Prüfung
Literatur	• Stuart Russell und Peter Norvig: Artificial Intelligence - A Modern Approach. Prentice Hall International, 2. Auflage (2003) • Günther Görz et al.: Handbuch der Künstlichen Intelligenz, Oldenbourg Verlag (2003) • Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From Data Mining to Knowledge Discovery: An Overview. In U. Fayyad & G. Piatetsky-Shapiro & P. Smyth (Eds.), Advances in Knowledge Discovery and Data Mining (pp. 1-34). Menlo Park, CA: AAAI Press/MIT Press (1996) • Kubat, M., Bratko, I., Michalski, R. S.: A Review of Machine Learning Methods. In R. S. Michalski & I. Bratko & M. Kubat (Eds.), Machine Learning and Data Mining: Methods and Application (2nd ed., pp. 3-69). Chichester: John Wiley & Sons Ltd. (1999) • Tom Mitchell: Machine Learning, McGraw Hill (1997) • Quinlan, J. R.,: C4.5 Programs for Machine Learning, Morgan Kaufmann (1993) • Uschold, M., Grüniger, M.,: Ontologies: Principles, Methods and Applications in Knowledge Engineering Review 11 (2), 93-155 (1996) • Baader, F., Calvanese, D., McGuinness, D.J., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook. Cambridge University Press, 2003, (2003) • Visser, U.,: Intelligent Information Integration for the Semantic Web. Vol. 3159, Lecture Notes in Computer Science, Springer-Verlag (2004) • Michael Wooldridge: An Introduction to MultiAgent Systems. Verlag John Wiley & Sons Ltd.. (2001) • Gerhard Weiss (ed): Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence. The MIT Press, Cambridge, Messachusetts (1999)

Modulbezeichnung	**Soft Computing**
Modulverantwortliche(r)	Prof. Dr. K. Schill
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	4 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 28 h Vortrag vorbereiten/Ausarbeitung schreiben 92 h Summe 120 h
Turnus des Moduls	i. d. R. angeboten in jedem WiSe
Voraussetzung für die Teilnahme	Keine ☐ Folgende
Lehr- und Lernformen	Seminar ☒ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Formale Methoden zum Umgang mit unsicherem Wissen kennen, definieren und verstehen könnnen • Zentrale Methoden des Schlussfolgerns in intelligenten Systemen kennen und verstehen können. • Grundlegende neuronale Netzarchitekturen und formale Methoden neuronaler Verarbeitung kennen und verstehen können • Den praktischen Einsatz wissensbasierter und neuronaler Methoden beispielhaft kennen und diskutieren können. • Hybride Systemarchitekturen, bei denen wissensbasierte und neuronale Ansätze integriert werden, beispielhaft kennen können. • Forschungsorientierte Literaturarbeit leisten können. • Forschungsarbeiten in englischer Sprache verstehen und im Plenum als Vortrag präsentieren können.
Lerninhalte	• Kalküle zum Umgang mit unsicherem Wissen • Reasoning-Strategien in wissensbasierten Systemen (z.B. informationsbasierte Strategien, hypothesengetriebene Strategien, Einbeziehung von Kosten und Nutzen) • Anwendungsbeispiele • Neuronale Netze – Prinzipien, Architekturen und Lernverfahren 1 – Theoretische Grundlagen: Perceptron, Multilayer Perceptron, Lineare Separierbarkeit, Feed-forward Netze, Backpropagation – Anwendungsbeispiele • Hybride Systeme 1 - Architekturen und Anwendungen
Prüfungsformen	i. d. R. mündlicher Vortrag, Handout

Literatur	• Shafer: A Mathematical Theorie of Evidence (1976) • Jensen: Bayesian networks and decision Graphs • Rojas: Theorie der neuronalen Netze (1996) • Russel, Norvig: Artificial Intelligence: A modern approach (1995) • ca. 10 Fachartikel zum Thema „Umgang mit unsicherem Wissen“

Modulbezeichnung	**Verhaltensbasierte Robotik**
Modulverantwortliche(r)	Prof. Dr. F. Kirchner
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	jährlich
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Es sollen die Grundlagen für moderne Roboterkontrollansätze vermittelt werden, die für vertiefende Diskussion und zur Erstellung von Steuerungsarchitekturen nutzbar sein sollen. • Dabei soll ein grundlegendes Verständnis von den Ursprüngen autonomer Roboter und aktueller Systeme zur Erklärung von Vor- und Nachteilen der vier Steuerungsarchitekturen (reaktiv, deliberativ, hybrid und verhaltensbasiert) abrufbar sein. • Verständnis von Herausforderungen bei der Entwicklung autonomer Roboter in Bezug auf Sensordatenverarbeitung und Generierung von Weltmodellen sowie geeigneter Verhalten • Der Umgang mit Werkzeugen und Techniken zur Realisierung von Roboterverhalten soll erlernt und geübt werden. Dabei insbesondere: • Kenntnisse zur Anwendung von Lokalisierungs- und Planungsalgorithmen • Erfahrung sammeln bei der Integration von Komponenten zur Sensordatenverarbeitung und Steuerung zu einem Gesamtsystem

Lerninhalte	• Einführung: Definition autonomer Roboter, Meilensteine, Spektrum der Roboterkontrollansätze, Definition von Verhalten, dezentrale Robotersteuerung und Bio-inspirierte Robotik • Sensoren und Aktuatoren (werden aus Sicht der Steuerungsarchitektur als Module zum Informationsgewinn und der Interaktionsmöglichkeit behandelt): Sensortypen, Vorverarbeitung, Umgang mit großen Datenmengen, Multimodale Sensorlösungen, Langzeitautonomie, Aktuatortypen, Regelung (PID, Kaskadenregler, dezentrale Regelung), Verschiedenen Regelungsziele z.B. Gravitationskompensation • Repräsentationen von Transformationen: für Robotik relevante Transformationen, Darstellungsmöglichkeiten von Rotationen z.B. durch Quaternionen, Vorteile durch das Wissen über algebraischer Eigenschaften der Transformationen in 2D und 3D • Lokalisierung: Mögliche Informationsquellen (z.B. Landmarken, Odometrie, Kameras, Laserscanner), Umgang mit Unsicherheit, probabilistische Lokalisierung mit dem Partikelfilter, Kartengenerierung mit SLAM • Planung: Verschiedene Repräsentationen, Restriktive Annahmen klassischer Planungssysteme, Plan-Space-Planung, Graphplanung, Temporale Planung, Pfad und Bewegungsplanung, Algorithmen (z.B. STRIPS und A*) • Steuerungsarchitekturen: Prinzipien und Beispiele von reaktiven, deliberativen, hybriden und verhaltensbasierten Ansätzen. Entwurf von Architekturen mit Verhaltensebenen, Motor Schema, emergentes Verhalten • State of the Art: Wie kommen die kennengelernten Konzepte und Methoden in aktuellen Systemen zum Einsatz? Moderne verhaltensbasierte Roboterarchitekturen am Beispiel von Lokomotion und Manipulation, Herausforderungen und Lösungsansätze bei der Steuerung von kinematisch komplexen Robotern in der realen Welt
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	Arkin, R.C., ‘Behaviour Based Robotics’, MIT Press (1998)

Modulbezeichnung	**Bestärkendes Lernen**
Modulverantwortliche(r)	Prof. Dr. F. Kirchner
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	jährlich
Voraussetzung für die Teilnahme	Keine ☐ Folgende: Inhaltliche Voraussetzungen: Robot Design Lab oder Verhaltensbasierte Robotik
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Grundlegende Kenntnisse des Bestärkenden Lernens (engl.: RL) • Kenntnisse der Anwendung und Anwendbarkeit von Lernverfahren für autonome Roboter • Kenntnis der Problemklasse „Markovsches Entscheidungsproblem“ (MDP) und des Konzepts der Wertfunktionen • Verständnis von Modell-bidenden (Dynamic Programming, Dyna-Architekturen) und Modell-freien (Monte-Carlo, Temporal Difference) Lernverfahren • Kenntnisse der wichtigsten Methoden und Verfahren zur Explorationskontrolle beim RL • Erlernen der Durchführung, Auswertung und Präsentation von empirischen Untersuchungen von Lernverfahren • Einarbeitung in die Literatur des aktuellen Stands der Technik

Lerninhalte	• Grundlagen des Bestärkenden Lernens (engl.: RL) • Problemklassen und Anwendungen für das Bestärkende Lernen • Grundlegende Probleme und Verfahren der Explorationskontrolle beim RL • Fortgeschrittene und aktuelle Themen des Bestärkenden Lernens (bspw. Direct Policy Search, Hierachisches RL, Deep RL, Multi-Agenten RL . . .) Insbesondere werden folgende theoretisch/methodische Grundlagen im Zusammenhang dieser Inhalte behandelt: • Theorie Markovscher Entscheidungsprozesse • Theorie des Dynamic Programming (Policy Iteration, Value Iteration) • Theorie der Monte Carlo Methoden • Theorie des Temporal Difference Lernens • Theorie von Modell-bildenden Verfahren • Einarbeitung und Verständnis von wissenschaftlichen Veröffentlichungen • Auswertung und Präsentation von Analysen / Algorithmen • Anfertigung von Diagrammen auf wissenschaftlichem Niveau • Anwendung von RL auf echten Systemen
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	Sutton, R., Barto, A. ‘Reinforcement Learning: An Introduction’, MIT-Press (1998)

Modulbezeichnung	**Biologische Grundlagen für autonome, mobile Roboter**
Modulverantwortliche(r)	Prof. Dr. F. Kirchner
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	jährlich
Voraussetzung für die Teilnahme	Keine ☐ Folgende: Formale Voraussetzungen: Studierende der HS Bremen, FB3 Uni HB und FB 4 Uni HBInhaltliche Voraussetzungen: Reinforcement Lernen (empfohlen)
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Verständnis der Robotik als integrierende Wissenschaft zwischen Elektrotechnik, Mechatronik und Informatik. • Grundlegende Kenntnisse des allg. Aufbau und der Funktion des zentralen Nervensystems • Kenntnisse der Entstehung, Weiterleitung und Beschreibung des Aktionspotentials bei Nervenzellen • Vertiefende Kenntnisse zu allgemeinen Grundlagen der motorischen Leistung bei Vertebraten • und Invertebraten • Bewertung der Informationsverarbeitung in biologischen Systemen • Bewertung und Klassifikation von biologischen Prinzipien im Bereich der Lokomotionskontrolle • Kenntnisse der Übertragbarkeit und Anwendung biologischer Prinzipien bei der Kontrolle mobiler autonomer Roboter • In der Terminologie des Fachgebiets Robotik sicher kommunizieren können und Systemkomponenten • Anhand der Terminologie klassifizieren und bewerten können. • Durch den Übungsbetrieb in kleinen Gruppen wird die Kooperations- und Teamfähigkeit geübt

Lerninhalte	• Allgemeiner Aufbau und Funktion des zentralen Nervensystems • Entstehung, Weiterleitung und Beschreibung des Aktionspotentials bei Nervenzellen • Allgemeine Grundlagen der motorischen Leistung bei Vertebraten und Invertebraten • Endogen aktive Zellen und zentrale Mustergeneratoren • Anwendung biologischer Prinzipien der Lokomotionskontrolle bei autonomen, mobilenRobotern Insbesondere werden folgende theoretisch/methodische Grundlagen im Zusammenhang dieser Inhalte behandelt: • Theorie der Synaptischen Signaltransduktion und Axonalen Signaltransmission in biologischen Systemen • Theorie der Erzeugung rhythmischer Lokomotion in biologischen Systemen • Theorie/Methodik der dezentralen Informationsverarbeitung in biologischen Systemen • Methodik der Übertragung biologischer Prinzipien der Lokomotionskontrolle auf Roboter
Prüfungsformen	i. d. R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	• Kandel, E., Schwartz, J, Jessel, T (eds)'Principles of Neural Science', Elsevier Science Publischers (1991)

Modulbezeichnung	**Anwendungen der Bildverarbeitung**
Modulverantwortliche(r)	Prof. Dr. U. Frese
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik, Produktionstechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• die wichtigsten Algorithmen der Bildverarbeitung verstehen • mit vorhandenen Bildverarbeitungsmodulen und anwendungsspezifischen Programmteilen BV-Anwendungen konzipieren, entwickeln und evaluieren können • geometrische Informationen in Bildern mit 3D-Koordinatensystemen und quadratischer Ausgleichsrechnung mit Programmen extrahieren können
Lerninhalte	• die wichtigsten Algorithmen der Bildverarbeitung – heuristische Segmentierung – Filter – Houghtransformation – Punktfeatures, SIFT, ORB – Matching, HoG, Bag of Words – Convolutional Neural Networks – FCNN – Faster R-CNN – Kameragleichung – RANSAC – least squares – bundle adjustment – stereo matching – 3d reconstruction • Methoden zur Konzeption, Entwicklung und Evaluierung von BV-Anwendungen durch Kombination existierender Libraries mit eigener Anwendungslogik – precision, recall, ROC-curve, test/training-Datensatz – Subalgorithmen mit und ohne mathematisch definierter Aufgabe – Debuggingstrategie bei Algorithmen mit Daten – Effekte und Einflüsse bei der Bildaufnahme

Prüfungsformen	Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	• Folien im Netz • Richard Szeliski, Computer Vision and Applications, Springer 2010 • Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016

Modulbezeichnung	**Theorie der Sensorfusion**
Modulverantwortliche(r)	Prof. Dr. U. Frese
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	I.d.R. angeboten alle 2 Jahre
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Fehlerbehaftete Größen in der Sprache der Stochastik (Kovarianzmatrix, Gaussverteilung) modellieren und damit Beweise führen können • Den (Extended/Unscented) Kalman Filter verstehen und anwenden können • Anschauliche Probleme der Sensorfusion mit Kalman Filter modellieren und lösen können • Anschauung und Theorie in Bezug bringen können, um Anwendungsprobleme und ihre Lösung mit Sensorfusionsalgorithmen beurteilen zu können
Lerninhalte	• Wahrscheinlichkeitsrechnung in R: Dichte, Erwartungswert, Varianz, Gaussverteilung • Fusion zweier Messwerte: Optimaler Schätzer • (Extended) Kalman Filter (1D) • Lineare Algebra: Vektoren und Matrizen • Wahrscheinlichkeitsrechng in R^n: Dichte, Erwartungswert, Kovarianzmatrix, mehrdimensionale Gaussverteilung • (Extended) Kalman Filter • Transformationen in 3D und homogene Koordinaten • Einführung [+]-Mannigfaltigkeiten • Unscented Kalman Filter auf [+]-Mannigfaltigkeiten
Prüfungsformen	Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	• Skript zur Vorlesung • S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, MIT Press 2006 • Y. Bar-Shalom, X.R. Li, T. Kirubarajan: Estimation with Applications to Tracking and Navigation, J. Wiley, 2001 • R. Hafner: Wahrscheinlichkeitsrechnung und Statistik, Springer, 1989

Modulbezeichnung	**Entwurf eingebetteter Systeme mit Digitallogik**
Modulverantwortliche(r)	PD Dr. St. Bosse
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik, Systemsoftware / Eingebettete Systeme, Produktionstechnik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Verständnis der anwendungsspezifischen Digitallogik für den Hardware-Entwurf als Erweiterung und Ergänzung zum Software-Entwurfs • Grundlegende Kentnisse der Funktionsweise von Digitallogiksystemen • Entwurf und Abbildung von Schaltnetzen auf boolesche Algebra • Kenntnisse über Optimierung von Digitallogiksystemen • Einführung der Register-Transfer-Logik Architektur als wesentliche Architektur und Entwurfsmethode für die Datenverarbeitung • Abbildung von klassischen Programmen auf RTL mit Daten- und Kontrollpfadpartitionierung • Kenntnisse über programmierbare Digitallogikschaltungen (CPLD/FPGA/ASIC) • Fähigkeit zum Modellieren von Digitallogiksystemen und Abbildung von Algorithmen auf RT-Ebene sowie mit der Hardware-Beschreibungssprache VHDL • Aufzeigen der Möglichkeiten der Parallelisierung von Algorithmen durch Digitallogiksysteme • Der Übungsanteil soll die praktische Umsetzung des in der Vorlesung erworbenen Wissens vermitteln und deren Anwendung an Beispeieln üben (z.B. Algorithmen auf RTL abbilden mit Verwendung des ReTrO Simulators)

Lerninhalte	● Digitallogik, Boolesche Algebra, Boolesche Funktionen ● Konjunktive- und Disjunktive Normalformen, Ableitungen aus Schaltbedingungen ● Technologische Umsetzung mit Transistoren ● Darstellung von booleschen Funktionen und Schaltnetzen mittels grafischer Methoden und Optimierung (KV-Diagramme) ● Systematische Darstellung und Optimierung von booleschen Funktionen mittels Binary Descision Diagrams (BDD) ● Programmierbare Digitallogik für Rapid Prototyping: Systematik und Aufbau Abbildung von Und-Oder-Matrizen auf verschiedene Technologien: RAM/PAL/GAL/CPLD/FPGA/ASIC ● Verwendung von hoch-integrierten Field-Programmable-Gate-Arrays (FPGA) ● Standardzellen-ASIC: Architektur unf Entwurfsmethoden ● Hardware-Entwurfsmethodik und Syntheseverfahren im Überblick, Ebenen des Logikentwurfs ● Kombinatorische Logiksysteme ● Sequenzielle Logiksysteme ● Systementwurf mit Register-Transfer-Logik (RTL) Architekturen ● Abbildung von Algorithmen auf Daten- und Kontrollpfade und Umsetzung mittels RTL (+ Scheduling & Allokation des Datenpfades) ● Laufzeitprobleme in elektronischen Systemen oder warum die Formale Verifikation nur graue Theorie sein kann ● Zustandsautomaten (Moore- und Meleay) und ihre Anwendung ● Beschreibung und Modellierung von Digitallogiksystemen mittels einer Hardware-Beschreibungssprache (VHDL)
Prüfungsformen	Erfolgreiche Bearbeitung von Übungsaufgaben und mündliche Prüfung
Literatur	1. Stefan Bosse Anwendungsspezifische (programmierbare) Digitallogik und VHDL-Synthese Skript, 3. Auflage (2012) 2. Michael D. Ciletti Advanced Digital Design with the Verilog VHDL Prentice Hall, (2003) 3. J. Reichardt, B. Schwarz VHDL-Synthese Oldenbourg Verlag (2003)

Modulbezeichnung	**Parallele und verteilte eingebettete Systeme**
Modulverantwortliche(r)	PD Dr. St. Bosse
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik, Systemsoftware / Eingebettete Systeme, Produktionstechnik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	● Verständnis der Funktionsweise und Entwurf von paralleler Datenverarbeitung ● Verständnis von parallelen Programmen und Rechnerarchitekturen ● Klassische Parallelrechner-Architekturen sollen auf Hardware-Ebene abgebildet und skaliert werden können ● Einsatz klassischer Multi-Prozeß-Modelle mit Interprozeß-Kommunikation für die Abbildung und Synthese von Algorithmen auf Hardware ● Verständnins und Anwendung von Kommunikation und Synchronisation in parallelen und verteilten Systemen ● Abbildung von Kommunikation auf Schaltkreise ● Verständins von System-On-Chip (SoC) Lösungen ● High-level Syntheseverfahren auf Programmiersprachenebene als zukunftsfähiges Entwurfswerkzeug für komplexe SoC ● Praktische Anwendung der Vorlesungsinhalte in der Übung (Grundlagen des Entwurfs von nebenläufigen Prozessen und Datenverarbeitung sowie Kommunikation mit Simulator CPV und Multi-Agenten Simulator SeSaM

Lerninhalte	• Multiprozeß-Modelle (Multi-Threading) bei generischen Prozessoren und Skalierung auf anwendungsspezifische Logiksysteme • Multiprozeß-Architekturen (Parallel-Rechner) mit generischen Prozessoren und Skalierung auf RTL und anwendungsspezifische Logiksysteme • Interprozeß-Kommunikation {Mutex, Semaphore, Event, Queue, Barrier, Channel} in Software und Abbildung auf RTL und Hardware-Ebene • Parallele Algorithmen in Soft- und Hardware • Parallel-Architekturen in Hardware: RTL, SoC und FPGAs • Netzwerkstrukturen und Topologien, adapdiert für SoC-Entwürfe • Logik- und algorithmische Highlevel-Synthese-Verfahren • Pipeline-Architekturen in funktionalen und reaktiven Systemen
Prüfungsformen	Erfolgreiche Bearbeitung von Übungsaufgaben und mündliche Prüfung
Literatur	• Stefan Bosse: Hardware-Entwurf von parallelen Systemen, Logik- & High-Level-Synthese, Skript, 2. Auflage (2013) • David C. Ku & Giovanni De Micheli: High Level Synthesis Under Timing and Synchronization Constraints, Kluwer, (1992)

Modulbezeichnung	**Machine learning for autonomous Robots**
Modulverantwortliche(r)	Prof. Dr. F. Kirchner
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Grundlegende Kenntnisse überwachter und unüberwachter maschineller Lernverfahren • Verständnis verschiedener Metriken und Auswertungsmethoden • Kenntnisse der Anwendung und Anwendbarkeit von maschinellen Lernverfahren für autonome Roboter • Erprobung von Algorithmen des maschinellen Lernens an Problemstellungen der Robotik • Stärkung der Kooperations- und Teamfähigkeit durch den Übungsbetrieb in kleinen Gruppen
Lerninhalte	• Grundlagen des unüberwachten Lernens (Merkmalsgenerierung, Merkmalsauswahl, Clustering) • Grundlagen des überwachten Lernens (Klassifikation und Regression) • Metriken und Evaluationsmethoden für das maschinelle Lernen • Erweiterte Kenntnisse zur Support Vektor Regression und Klassifikation • Grundlagen des Meta-Lernens • Grundlagen künstlicher neuronaler Netze • Einführung in Deep-Learning und fortgeschrittene Techniken neuronaler Netze • Anwednung von Verfahren maschinellen Lernens in der Robotik und angrenzender Felder
Prüfungsformen	i. d. R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	• Mitchell, T. , Machine Learning‘, Mcgraw-Hill (1997) • Mackay, D., Information Theory, Inference, and Learning Algorithms‘, Cambridge University Press (2003)

Modulbezeichnung	**Intelligente Umgebungen für die alternde Gesellschaft**
Modulverantwortliche(r)	Prof. Dr. K. Schill
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik
Dauer des Moduls	1 Semester
Kreditpunkte	4 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 28 h Vortrag vorbereiten/Ausarbeitung schreiben 92 h Summe 120 h
Turnus des Moduls	i.d.R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☒ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Die Entwicklung, Gestaltung und Einsatzmöglichkeiten informations- und kommunikationstechnischer Systeme zur Verbesserung der Selbständigkeit älterer Menschen kennen und verstehen. • Die Möglichkeiten und Grenzen assistiver Technologien und Umgebungen beurteilen und bewerten können • Methoden zur Aktivitätserkennung und zur Umgebungssteuerung kennen und verstehen. • Sich mit ethischen Fragen an Hand von Beispielen kritisch auseinander setzen können. • Die wesentlichen kognitiven und physiologischen Veränderungen im Alter kennen und verstehen.
Lerninhalte	Mittelpunkt dieses Seminars ist die differenzierte Auseinandersetzung mit technischen, sozialen und ethischen Aspekten des Einsatzes von Informationstechnologie in intelligenten, assistiven Umgebungen. Dazu findet eine Auseinandersetzung statt mit der Theorie, praktischen Beispielen und ethischen Aspekten zu: • Intelligente Umgebungen • Sensortechnologie • Sensorfusion • Aktivitätserkennung und Monitoring • Umgebungssteuerung • Kommunikations- und Interaktionshilfsmittel • Prothetik und Mobilitätshilfsmittel • Technikazeptanz • Kognitive und physiologische Veränderungen im Alter • Anpassbarkeit und Barrierefreiheit / “adaptability” und “accessability”
Prüfungsformen	mündlicher Vortrag und schriftliche Ausarbeitung
Literatur	Literatur wird in den einzelnen Seminaren bekanntgegeben.

Modulbezeichnung	**Integrated Intelligent Systems**
Modulverantwortliche(r)	Michael Beetz
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik, Systemsoftware / Eingebettete Systeme, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	einjährig
Voraussetzung für die Teilnahme	Keine ☐ Folgende: Formale Voraussetzungen: KeineInhaltliche Voraussetzungen: Kenntnisse der Grundlagen der Künstlichen Intelligenz (BB-710.01)
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	Die Vorlesung beschäftigt sich mit aktuellen Techniken zur Implementierung von technischen kognitiven Systemen, das heißt mit intelligenten Computersystemen, die über Sensoren und Aktuoren verfügen. Solche Systeme werden vor allem in Bereichen wie der Service-Robotik, in autonomen Raumsonden, in intelligenten Wohn- und Arbeitsbereichen und in Fahrerassistenzsystemen eingesetzt.
Lerninhalte	Es werden folgende Themen behandelt: Sensoren, Aktuatoren und physikalische Infrastrukturen von technischen kognitiven Systemen (u.a. Smart Sensors, Sensornetzwerke); Berechnungsmodelle zur Steuerung technischer kognitiver Systeme: dynamisches Systemmodell, rationales Agentenmodell, das Berechnungsmodell der technischen kognitiven Systeme; Grundlagen probabilistischer Zustandsschätzung: Bayes-Filter, Kalman-Filter, Partikel- Filter, Mechanism en zur Datenassoziation, Lernen von Sensor- und Aktionsmodellen, Hidden Markov Modelle, Expectation Maximization; Anwendungen probabilistischer Zustandsschätzung: Selbstlokalisierung, Umgebungskartierung, Objektverfolgung; Programmiermethoden für technische kognitive Systeme: nebenläufig reaktive Steuerungsmechanismen; Wissens- und planbasierte Steuerungstechniken
Prüfungsformen	Eine Auswahl aus mündlicher Prüfung, Klausur und Übungen mit Fachgespräch.
Literatur	

Modulbezeichnung	**Massively Parallel Algorithms**
Modulverantwortliche(r)	Prof. Dr. G. Zachmann
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Systemsoftware / Eingebettete Systeme, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☐ Folgende: Inhaltliche Voraussetzungen: Algorithmisches Denken. Gewisse Programmierfähigkeiten in C (empfohlen wird das "Propädeutikum C/C++")
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐

Lernziele	Die Ära der single-core Prozessoren ist zu Ende. Inzwischen gibt es neue, massiv-parallele Prozessoren (GPUs), die hunderte bis tausende von Threads parallel abarbeiten können. Diese entwickeln sich zur Zeit als Co-Prozessoren, die große Teile der Berechnung den (multi-core) CPUs abnehmen. Möglicherweise werden sich GPUs als neue Architektur für die Haupt-Prozessoren – gerade auch auf mobilen Geräten – etablieren, da diese mehr Compute-Power pro Energieeinheit bieten. Die große Zahl von parallelen Cores stellt das Design von Algorithmen und Software allerdings vor neue Herausforderungen, damit diese von der großen Parallelität profitieren können. Das Hauptziel dieser Vorlesung ist es, Studenten in die Lage zu versetzen, für solch massiv-parallele Hardware Algorithmen zu entwerfen. Simulation wird inzwischen gemeinhin als die dritte Säule der Wissenschaft angesehen (neben den Experimenten und der Theorie). In der Simulation wird ein ständig wachsender Bedarf an Rechenleistung benötigt; gerade diese wird aber durch die Verfügbarkeit von GPUs fast schon zu einer Commodity auf dem Desktop. Daher gibt es viele wissenschaftliche Bereiche, in denen Studenten das Wissen, das sie in dieser Vorlesung erwerben, gewinnbringend einsetzen können, wie z.B.: • Computer science (e.g., visual computing, database search) • Computational material science (e.g., molecular dynamics simulation) • Bio-informatics (e.g., alignment, sequencing, . . .) • Economics (e.g., simulation of financial models) • Mathematics (e.g., solving large PDEs) • Mechanical engineering (e.g., CFD and FEM) • Physics (e.g., ab initio simulations) • Logistics (e.g. simulation of traffic, assembly lines, or supply chains) Am Ende dieser Vorlesung werden Studenten • aktive Erfahrungen bei der Entwicklung von Software und Algorithmen für massiv-parallele Architekturen gesammelt haben; • eine Anzahl von massiv-parallelen Algorithmen-Patterns kennen; • in der Lage sein, eigene massiv-parallele Algorithmen zu entwickeln; • CUDA kennen. In der ersten Hälfte der Vorlesung werden Studenten sich anhand von kleinen und mittelgroßen Übungen und Frameworks mit der parallelen Programmier-Umgebung CUDA vertraut machen. In der zweiten Hälfte werden Studenten an einem eigenen Projekt arbeiten.
Lerninhalte	Diese Vorlesung führt Studenten in die grundlegenden und einige fortgeschrittene Methoden und Techniken der massiv-parallelen Algorithmen ein. Einige der vorgesehenen Themen sind: • die Programmierumgebung CUDA C; • die Speicher-Hierarchie und verschiedene Speicher-Charakteristiken; • die GPU Architektur • parallele Reduktion; • coalesced memory access; • massiv-parallele Matrix-Algorithmen; • Prefix-Sum und deren Anwendungen in der Bildverarbeitung; • Textur-Filterung; • Paralleles Sortieren (odd-even, bitonic, adaptive bitonic); • Bildverarbeitung; • Thrust;
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung

Literatur	• Jason Sanders, Edward Kandort: CUDA by Example. Addison-Wesley, Pearson Education. • Wen-Mei W. Hwu: GPU Computing Gems Jade Edition. Morgan Kaufmann. • David B. Kirk, Wen-Mei W. Hwu: Programming Massively Parallel Processors. Morgan Kaufmann. • NVidia: CUDA C Programming Guide.

Modulbezeichnung	**Material-integrierte Sensorische Systeme**
Modulverantwortliche(r)	PD Dr. Stefan Bosse
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik, Systemsoftware / Eingebettete Systeme, Produktionstechnik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb / Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	jedes WS
Voraussetzung für die Teilnahme	Keine ☐ Folgende Formale Voraussetzungen: Keine
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	Die Teilnahme an der Veranstltung soll Studenten interdisziplinär einen system-orientierten Zugang für die Modellierung, den Entwurf und die Anwendung von material-eingebetteten oder material-applizierten Sensorischen Systemen bieten, die aufgrund der technischen Realisierung und des Einsatzes spezielle Anforderungen an die Datenverarbeitung stellen und ein Verständnis des Gesamtsystems (inlkusive Aspekte der Materialwissenschaften und Technologien) voraussetzen. Diese neuen Sensorischen Materialen finden z. B. in der Robotik (Kognition) oder in der Produktionstechnik für die Materialüberwachung Anwendung. Folgende Kompetenzen sollen erworben werden: 1. Grundverständnis des technischen Aufbaus und der Funktionweise von Sensorischen Materialien: Elektronische Signalverarbeitung von Sensoren, Mechanisches Verhalten, Einfluss von Sensoren und Elektronik auf mechanische Eigenschaften des Trägermaerials 2. Datenverarbeitung in Sensornetzwerken unter harten Randbedingungen wie limitierten Energieangebot, Rechenleistung und Speicher, Fehleranfälligkeit 3. Parallele und verteilte Datenverarbeitung geeignet für low-ressource Sensornetzwerke: Architekturen, Kommunikation, Kooperation, Wettbewerb um Ressourcen 4. Grundlagen der Robustheit, Fehlernanalyse, Redundanz in solchen Sensornetzwerken

Lerninhalte	A. Einführung in die Thematik, was ist Sensorik, warum und wofür material-integrierte Sensorik, Sensorische Materialien, Anwendungen in der Strukturüberwachung und Robotik B. Systemebene und System Entwicklung, Anforderungen, Entwurfsmethodiken, Test & Simulation, Normen und Standards C. Grundlagen von Sensoren und Elektronik, Sensor- und Elektronikentwicklung für die Materialintegration D. Sensornetzwerke und Sensorknoten, Grundlagen, Metriken, Entwicklung E. Schnittstellen Technologien: Material - Sensor - Elektronik - Systeme (Vernetzung) F. Signal- und Datenverarbeitung in Sensornetzwerken (Knotenebene) G. Kommunikation in Sensornetzwerken (Netzwerkebene) H. Energie-Versorgung und Management in material-integrierten Sensornetzwerken und Energiegewinnung (Harvesting) I. Applikationen (Robotik, Strukturüberwachung usw.)
Prüfungsformen	Erfolgreiche Bearbeitung von Übungszetteln + Mündliche Prüfung
Literatur	

Modulbezeichnung	**Real-time Operating Systems Development**
Modulverantwortliche(r)	Jan Peleska
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Systemsoftware / Eingebettete Systeme, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 0 h Übungsbetrieb/Prüfungsvorbereitung 180 h Summe 180 h
Turnus des Moduls	every year
Voraussetzung für die Teilnahme	Keine ☐ Folgende: Formale Voraussetzungen: KeineInhaltliche Voraussetzungen: Good programming skills in C are mandatory. A thorough understanding of basic operating systems concepts is very helpful for this lecture.
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	Students 1) know how to program a real-time application from scratch on “bare-metal”, that is, WITHOUT a supporting operating systems 2) know how to design an elegant real-time operating system kernel from scratch 3) understand the right balance between architectural beauty and optimised performance 4) know about basic benchmarks assessing the real-time capabilities of an RTOS 5) know how to do practical real-time application programming and RTOS development from scratch on a simple ARM-based computer architecture (BeagleBone Black)
Lerninhalte	. 1) Bare-metal programming on BeagleBode Black boards using the Code Composer Studio development environment (Eclipse-based) 2) The State Machine programming paradigm with cooperative multi-tasking, scheduling, watchdog monitor 3) Periodic time-controlled activities 4) Simple context switching: Programming user threads and associated schedulers 5) Inspiration from micro kernels: RTOS architecture with communication channels and ports 6) Filtered and prioritised real-time port handling 7) Real-time synchronisation mechanisms 8) Time-triggered versus event-based RTOS paradigms 9) RTOS Benchmarks
Prüfungsformen	Oral module examination or Exercises and oral technical discussion (Fachgespräch)

Literatur	• Wang, K.C. Embedded and Real-Time Operating Systems. DOI 10.1007/978-3-319-51517-5_2. Springer 2017. • Kopetz, H. Real-Time Systems: Design Principles for Distributed Embedded Applications. Second edition. Springer 2011. • Walls, C. Building a Real-Time Operating system. Rtos from the ground up. Elsevier Science & Technology 2007. • Cooling, J. Real-time Operating Systems Book 1. The Theory. Lindentree Associates, 2017. • Cooling, J. Real-time Operating Systems Book 2. The Practice. Lindentree Associates, 2017.

Modulbezeichnung	**Grundlagen der Sicherheitsanalyse und des Designs**
Modulverantwortliche(r)	Prof. Dr. D. Hutter
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik, Systemsoftware / Eingebettete Systeme, Produktionstechnik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. jedes Jahr
Voraussetzung für die Teilnahme	Keine ☐ Folgende: Inhaltliche Voraussetzungen: Kenntnisse in formalen Methoden bzw. Informationssicherheit sind nützlich aber nicht zwingend erforderlich
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Verfahren der (formalen) Modellierung von (Informations)Sicherheitsanforderungen und Sicherheitsmechanismen kennen • Verschiedene Sicherheitsanalysetechniken einschätzen und bewerten können • Die Modellierungstiefe und deren Auswirkungen auf die Analyse einschätzen und bewerten können • Das Zusammenspiel von verschiedenen Sicherheitsanforderungen und -garantien verstehen
Lerninhalte	Grundlagen der Modellierung im Bereich der Informationssicherheit Design und Analyse von Sicherheitsprotokollen • Modellierung eines Angreifers • Prinzipien des Designs von Sicherheitsprotokollen • Analyse und Verifikation von Sicherheitsprotokollen Design und Analyse von Sicherheitspolitiken • Modellierung (formaler) Sicherheitspolitiken • Grundlagen der Informationsflusskontrolle, Vertraulichkeit und Integrität als Informationsflusseigenschaften • Zustandsbasierte Informationsflusskontrolle • sprachbasierte Informationsflusskontrolle und Programmanalyse • Realisierung von Informationsflusskontrolle durch Zugriffskontrolle Komposition verschiedener Sicherheitsmechanismen am Beispiel des Semantic Web
Prüfungsformen	Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung

Literatur	Skript bzw. Folien Dieter Gollmann: Computer Security, Wiley&Sons, 2006 Matt Bishop: Computer Security, Art und Science, Addison Wesley, 2003 Diverse Fachartikel

Modulbezeichnung	**Anwendungen der Bildverarbeitung**
Modulverantwortliche(r)	Prof. Dr. U. Frese
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik, Produktionstechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• die wichtigsten Algorithmen der Bildverarbeitung verstehen • mit vorhandenen Bildverarbeitungsmodulen und anwendungsspezifischen Programmteilen BV-Anwendungen konzipieren, entwickeln und evaluieren können • geometrische Informationen in Bildern mit 3D-Koordinatensystemen und quadratischer Ausgleichsrechnung mit Programmen extrahieren können
Lerninhalte	• die wichtigsten Algorithmen der Bildverarbeitung – heuristische Segmentierung – Filter – Houghtransformation – Punktfeatures, SIFT, ORB – Matching, HoG, Bag of Words – Convolutional Neural Networks – FCNN – Faster R-CNN – Kameragleichung – RANSAC – least squares – bundle adjustment – stereo matching – 3d reconstruction • Methoden zur Konzeption, Entwicklung und Evaluierung von BV-Anwendungen durch Kombination existierender Libraries mit eigener Anwendungslogik – precision, recall, ROC-curve, test/training-Datensatz – Subalgorithmen mit und ohne mathematisch definierter Aufgabe – Debuggingstrategie bei Algorithmen mit Daten – Effekte und Einflüsse bei der Bildaufnahme

Prüfungsformen	Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	• Folien im Netz • Richard Szeliski, Computer Vision and Applications, Springer 2010 • Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016

Modulbezeichnung	**Theorie der Sensorfusion**
Modulverantwortliche(r)	Prof. Dr. U. Frese
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	I.d.R. angeboten alle 2 Jahre
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Fehlerbehaftete Größen in der Sprache der Stochastik (Kovarianzmatrix, Gaussverteilung) modellieren und damit Beweise führen können • Den (Extended/Unscented) Kalman Filter verstehen und anwenden können • Anschauliche Probleme der Sensorfusion mit Kalman Filter modellieren und lösen können • Anschauung und Theorie in Bezug bringen können, um Anwendungsprobleme und ihre Lösung mit Sensorfusionsalgorithmen beurteilen zu können
Lerninhalte	• Wahrscheinlichkeitsrechnung in R: Dichte, Erwartungswert, Varianz, Gaussverteilung • Fusion zweier Messwerte: Optimaler Schätzer • (Extended) Kalman Filter (1D) • Lineare Algebra: Vektoren und Matrizen • Wahrscheinlichkeitsrechng in R^n: Dichte, Erwartungswert, Kovarianzmatrix, mehrdimensionale Gaussverteilung • (Extended) Kalman Filter • Transformationen in 3D und homogene Koordinaten • Einführung [+]-Mannigfaltigkeiten • Unscented Kalman Filter auf [+]-Mannigfaltigkeiten
Prüfungsformen	Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	• Skript zur Vorlesung • S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, MIT Press 2006 • Y. Bar-Shalom, X.R. Li, T. Kirubarajan: Estimation with Applications to Tracking and Navigation, J. Wiley, 2001 • R. Hafner: Wahrscheinlichkeitsrechnung und Statistik, Springer, 1989

Modulbezeichnung	**Entwurf eingebetteter Systeme mit Digitallogik**
Modulverantwortliche(r)	PD Dr. St. Bosse
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik, Systemsoftware / Eingebettete Systeme, Produktionstechnik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Verständnis der anwendungsspezifischen Digitallogik für den Hardware-Entwurf als Erweiterung und Ergänzung zum Software-Entwurfs • Grundlegende Kentnisse der Funktionsweise von Digitallogiksystemen • Entwurf und Abbildung von Schaltnetzen auf boolesche Algebra • Kenntnisse über Optimierung von Digitallogiksystemen • Einführung der Register-Transfer-Logik Architektur als wesentliche Architektur und Entwurfsmethode für die Datenverarbeitung • Abbildung von klassischen Programmen auf RTL mit Daten- und Kontrollpfadpartitionierung • Kenntnisse über programmierbare Digitallogikschaltungen (CPLD/FPGA/ASIC) • Fähigkeit zum Modellieren von Digitallogiksystemen und Abbildung von Algorithmen auf RT-Ebene sowie mit der Hardware-Beschreibungssprache VHDL • Aufzeigen der Möglichkeiten der Parallelisierung von Algorithmen durch Digitallogiksysteme • Der Übungsanteil soll die praktische Umsetzung des in der Vorlesung erworbenen Wissens vermitteln und deren Anwendung an Beispeieln üben (z.B. Algorithmen auf RTL abbilden mit Verwendung des ReTrO Simulators)

Lerninhalte	● Digitallogik, Boolesche Algebra, Boolesche Funktionen ● Konjunktive- und Disjunktive Normalformen, Ableitungen aus Schaltbedingungen ● Technologische Umsetzung mit Transistoren ● Darstellung von booleschen Funktionen und Schaltnetzen mittels grafischer Methoden und Optimierung (KV-Diagramme) ● Systematische Darstellung und Optimierung von booleschen Funktionen mittels Binary Descision Diagrams (BDD) ● Programmierbare Digitallogik für Rapid Prototyping: Systematik und Aufbau Abbildung von Und-Oder-Matrizen auf verschiedene Technologien: RAM/PAL/GAL/CPLD/FPGA/ASIC ● Verwendung von hoch-integrierten Field-Programmable-Gate-Arrays (FPGA) ● Standardzellen-ASIC: Architektur unf Entwurfsmethoden ● Hardware-Entwurfsmethodik und Syntheseverfahren im Überblick, Ebenen des Logikentwurfs ● Kombinatorische Logiksysteme ● Sequenzielle Logiksysteme ● Systementwurf mit Register-Transfer-Logik (RTL) Architekturen ● Abbildung von Algorithmen auf Daten- und Kontrollpfade und Umsetzung mittels RTL (+ Scheduling & Allokation des Datenpfades) ● Laufzeitprobleme in elektronischen Systemen oder warum die Formale Verifikation nur graue Theorie sein kann ● Zustandsautomaten (Moore- und Meleay) und ihre Anwendung ● Beschreibung und Modellierung von Digitallogiksystemen mittels einer Hardware-Beschreibungssprache (VHDL)
Prüfungsformen	Erfolgreiche Bearbeitung von Übungsaufgaben und mündliche Prüfung
Literatur	1. Stefan Bosse Anwendungsspezifische (programmierbare) Digitallogik und VHDL-Synthese Skript, 3. Auflage (2012) 2. Michael D. Ciletti Advanced Digital Design with the Verilog VHDL Prentice Hall, (2003) 3. J. Reichardt, B. Schwarz VHDL-Synthese Oldenbourg Verlag (2003)

Modulbezeichnung	**Parallele und verteilte eingebettete Systeme**
Modulverantwortliche(r)	PD Dr. St. Bosse
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik, Systemsoftware / Eingebettete Systeme, Produktionstechnik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	● Verständnis der Funktionsweise und Entwurf von paralleler Datenverarbeitung ● Verständnis von parallelen Programmen und Rechnerarchitekturen ● Klassische Parallelrechner-Architekturen sollen auf Hardware-Ebene abgebildet und skaliert werden können ● Einsatz klassischer Multi-Prozeß-Modelle mit Interprozeß-Kommunikation für die Abbildung und Synthese von Algorithmen auf Hardware ● Verständnins und Anwendung von Kommunikation und Synchronisation in parallelen und verteilten Systemen ● Abbildung von Kommunikation auf Schaltkreise ● Verständins von System-On-Chip (SoC) Lösungen ● High-level Syntheseverfahren auf Programmiersprachenebene als zukunftsfähiges Entwurfswerkzeug für komplexe SoC ● Praktische Anwendung der Vorlesungsinhalte in der Übung (Grundlagen des Entwurfs von nebenläufigen Prozessen und Datenverarbeitung sowie Kommunikation mit Simulator CPV und Multi-Agenten Simulator SeSaM

Lerninhalte	• Multiprozeß-Modelle (Multi-Threading) bei generischen Prozessoren und Skalierung auf anwendungsspezifische Logiksysteme • Multiprozeß-Architekturen (Parallel-Rechner) mit generischen Prozessoren und Skalierung auf RTL und anwendungsspezifische Logiksysteme • Interprozeß-Kommunikation {Mutex, Semaphore, Event, Queue, Barrier, Channel} in Software und Abbildung auf RTL und Hardware-Ebene • Parallele Algorithmen in Soft- und Hardware • Parallel-Architekturen in Hardware: RTL, SoC und FPGAs • Netzwerkstrukturen und Topologien, adapdiert für SoC-Entwürfe • Logik- und algorithmische Highlevel-Synthese-Verfahren • Pipeline-Architekturen in funktionalen und reaktiven Systemen
Prüfungsformen	Erfolgreiche Bearbeitung von Übungsaufgaben und mündliche Prüfung
Literatur	• Stefan Bosse: Hardware-Entwurf von parallelen Systemen, Logik- & High-Level-Synthese, Skript, 2. Auflage (2013) • David C. Ku & Giovanni De Micheli: High Level Synthesis Under Timing and Synchronization Constraints, Kluwer, (1992)

Modulbezeichnung	**Test von Schaltungen und Systemen**
Modulverantwortliche(r)	Prof. Dr. R. Drechsler
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Mechatronik, Systemsoftware / Eingebettete Systeme, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i.d.R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☒ Folgende
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• Das Problem des Testens verstehen und erklären können • Den Testverlauf für Schaltungen und Systeme kennen und anwenden können • Klassische und moderne Testverfahren kennen und anwenden können • Die Algorithmen auf (Schaltkreis-)Graphen anwenden können • Die Komplexität der Verfahren verstehen und erklären können
Lerninhalte	1. Physikalische Fehlerursachen 2. Abstraktion von der physikalischen Ebene, Fehlermodelle 3. Algorithmen zur Berechnung von Signalwahrscheinlichkeiten 4. Techniken zur Manipulation Boolescher Funktionen 5. Algorithmen zur Fehlersimulation 6. Algorithmen zur Testmustergenerierung 7. Nutzung strukturellen Wissens zur Effizienzsteigerung 8. Techniken zur Reduktion des Suchraumes, Fehleräquivalenz und -dominanz Aus den Inhalten ist deutlich zu erkennen, dass theoretisch/methodische Grundlagen einen wichtigen Teil dieser Vorlesung darstellen. Darüber hinaus werden für die vorgestellten Verfahren die Komplexitäten hinsichtlich Laufzeit und Speicher betrachtet.
Prüfungsformen	i. d. R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung

Literatur	• M.L. Bushnell, V.D. Agrawal: Essentials of Electronic Testing – for Digital, Memory & Mixed-Signal VLSI Circuits, New York: Springer, 2000. • N. Jha, S. Gupta: Testing of Digital Systems, Cambridge University Press, 2003. • A. Miczo: Digital Logic Testing and Simulation, 2. Auflage, Wiley, 2003. • H. Wojtkowiak: Test und Testbarkeit digitaler Schaltungen, Teubner, 1988. • H.-J. Wunderlich: Hochintegrierte Schaltungen: Prüfgerechter Entwurf und Test, Berlin: Springer, 1991.

Modulbezeichnung	**Integrated Intelligent Systems**
Modulverantwortliche(r)	Michael Beetz
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik, Systemsoftware / Eingebettete Systeme, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	einjährig
Voraussetzung für die Teilnahme	Keine ☐ Folgende: Formale Voraussetzungen: KeineInhaltliche Voraussetzungen: Kenntnisse der Grundlagen der Künstlichen Intelligenz (BB-710.01)
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	Die Vorlesung beschäftigt sich mit aktuellen Techniken zur Implementierung von technischen kognitiven Systemen, das heißt mit intelligenten Computersystemen, die über Sensoren und Aktuoren verfügen. Solche Systeme werden vor allem in Bereichen wie der Service-Robotik, in autonomen Raumsonden, in intelligenten Wohn- und Arbeitsbereichen und in Fahrerassistenzsystemen eingesetzt.
Lerninhalte	Es werden folgende Themen behandelt: Sensoren, Aktuatoren und physikalische Infrastrukturen von technischen kognitiven Systemen (u.a. Smart Sensors, Sensornetzwerke); Berechnungsmodelle zur Steuerung technischer kognitiver Systeme: dynamisches Systemmodell, rationales Agentenmodell, das Berechnungsmodell der technischen kognitiven Systeme; Grundlagen probabilistischer Zustandsschätzung: Bayes-Filter, Kalman-Filter, Partikel- Filter, Mechanism en zur Datenassoziation, Lernen von Sensor- und Aktionsmodellen, Hidden Markov Modelle, Expectation Maximization; Anwendungen probabilistischer Zustandsschätzung: Selbstlokalisierung, Umgebungskartierung, Objektverfolgung; Programmiermethoden für technische kognitive Systeme: nebenläufig reaktive Steuerungsmechanismen; Wissens- und planbasierte Steuerungstechniken
Prüfungsformen	Eine Auswahl aus mündlicher Prüfung, Klausur und Übungen mit Fachgespräch.
Literatur	

Modulbezeichnung	**Material-integrierte Sensorische Systeme**
Modulverantwortliche(r)	PD Dr. Stefan Bosse
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Automatisierung und Robotik, Mechatronik, Systemsoftware / Eingebettete Systeme, Produktionstechnik, Raumfahrt-Systemtechnik
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb / Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	jedes WS
Voraussetzung für die Teilnahme	Keine ☐ Folgende Formale Voraussetzungen: Keine
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	Die Teilnahme an der Veranstltung soll Studenten interdisziplinär einen system-orientierten Zugang für die Modellierung, den Entwurf und die Anwendung von material-eingebetteten oder material-applizierten Sensorischen Systemen bieten, die aufgrund der technischen Realisierung und des Einsatzes spezielle Anforderungen an die Datenverarbeitung stellen und ein Verständnis des Gesamtsystems (inlkusive Aspekte der Materialwissenschaften und Technologien) voraussetzen. Diese neuen Sensorischen Materialen finden z. B. in der Robotik (Kognition) oder in der Produktionstechnik für die Materialüberwachung Anwendung. Folgende Kompetenzen sollen erworben werden: 1. Grundverständnis des technischen Aufbaus und der Funktionweise von Sensorischen Materialien: Elektronische Signalverarbeitung von Sensoren, Mechanisches Verhalten, Einfluss von Sensoren und Elektronik auf mechanische Eigenschaften des Trägermaerials 2. Datenverarbeitung in Sensornetzwerken unter harten Randbedingungen wie limitierten Energieangebot, Rechenleistung und Speicher, Fehleranfälligkeit 3. Parallele und verteilte Datenverarbeitung geeignet für low-ressource Sensornetzwerke: Architekturen, Kommunikation, Kooperation, Wettbewerb um Ressourcen 4. Grundlagen der Robustheit, Fehlernanalyse, Redundanz in solchen Sensornetzwerken

Lerninhalte	A. Einführung in die Thematik, was ist Sensorik, warum und wofür material-integrierte Sensorik, Sensorische Materialien, Anwendungen in der Strukturüberwachung und Robotik B. Systemebene und System Entwicklung, Anforderungen, Entwurfsmethodiken, Test & Simulation, Normen und Standards C. Grundlagen von Sensoren und Elektronik, Sensor- und Elektronikentwicklung für die Materialintegration D. Sensornetzwerke und Sensorknoten, Grundlagen, Metriken, Entwicklung E. Schnittstellen Technologien: Material - Sensor - Elektronik - Systeme (Vernetzung) F. Signal- und Datenverarbeitung in Sensornetzwerken (Knotenebene) G. Kommunikation in Sensornetzwerken (Netzwerkebene) H. Energie-Versorgung und Management in material-integrierten Sensornetzwerken und Energiegewinnung (Harvesting) I. Applikationen (Robotik, Strukturüberwachung usw.)
Prüfungsformen	Erfolgreiche Bearbeitung von Übungszetteln + Mündliche Prüfung
Literatur	

Modulbezeichnung	**Introduction to System Identification**
Modulverantwortliche(r)	Matthew Hoelzel
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Mechatronik
Dauer des Moduls	1 Semester
Kreditpunkte	4 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 42 h Übungsbetrieb/Prüfungsvorbereitung 78 h Summe 120 h
Turnus des Moduls	in der Regel in jedem SoSe
Voraussetzung für die Teilnahme	Keine ☐ Folgende: Formale Voraussetzungen: KeineInhaltliche Voraussetzungen: A brief knowledge of linear systems and statistics.
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	• To understand the basic system identification process, which involves a combination of model selection, data analysis, and noise assumptions. • To acquire a knowledge of several system identification techniques, and to understand when each method is applicable. • To understand the effect of the model, noise, and system identification on the estimated model, that is, to understand the effects of the assumptions used to obtain a model estimate.
Lerninhalte	• Typical model structures used in system identification: state-space, polynomial matrix, impulse response, and frequency domain models. • Model properties: controllability, observability, reachability, and linearity. • Requirements for the identifiability of a model, specifically, persistency. • Regression and least-squares analysis for linear-in-the-parameters models. • Consistency of estimated models and other useful statistical properties. • Parameter estimation methods such as instrumental variable methods.
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung
Literatur	• C. T. Chen, “Linear System Theory and Design”, 3rd ed. New York: Oxford University Press, 1999. • M. Verhaegen and V. Verdult, “Filtering and System Identification: A Least Squares Approach”, 1st ed. New York: Cambridge University Press, 2007. • L. Ljung, “System Identification: Theory for the User”, 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 1999. • R. Pintelon and J. Schoukens, “System Identification: A Frequency Domain Approach”, 1st ed. New York: Wiley-IEEE Press, 2001.