Informatik Technischer Systeme

Pflichtfächer

Stand: 21.06.2018

1. Semester
   - Diskrete Mathematik
   - Grundlagen der Technischen Informatik
   - Programmiermethodik I
   - Programmiertechnik
   - Mess- und Sensortechnik
2. Semester
   - Automatentheorie und Formale Sprachen
   - Programmiermethodik II
   - Datenbanken
   - Grundlagen der systemnahen Programmierung
   - Analysis und lineare Algebra
3. Semester
   - Signalverarbeitung und Stochastik
   - Algorithmen und Datenstrukturen
   - Software Engineering I
   - Betriebssysteme
   - Intelligente Sensorsysteme
4. Semester
   - Mustererkennung und Machine Learning
   - Embedded System Engineering
   - Rechnernetze
5. Semester
   - Verteilte Systeme
   - Betriebswirtschaft

6. Semester
   - Cyber-physische Systeme

Modulbezeichnung	**Diskrete Mathematik**	Kürzel	DM / DMÜ
Lehrveranstaltung(en)	Vorlesung: Diskrete Mathematik Übung: Diskrete Mathematik	Semester	1
Arbeitsaufwand	36 Std. Vorlesung, 12 Std. Übung, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Stephan Pareigis	SWS	3+1
Dozenten	Prof. Dr. Stephan Pareigis , Prof. Dr. Reinhard Baran, Prof. Dr. Julia Padberg	Sprache	deutsch
Voraussetzungen	Inhalte des Mathematik-Vorkurs	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • können wichtige mathematische Strukturen sicher verwenden • beherrschen die logischen und algebraischen Grundlagen der theoretischen Informatik • können Definitionsprinzipien und Beweistechniken in unterschiedlichen Bereichen und an typischen Beispielen anwenden • können die Methoden der linearen Algebra anwenden		
Inhalte	• Mathematische Grundlagen: Mengen, Relationen, Abbildungen, Funktionen und deren Operatoren, Aussagenlogik, Boolsche Algebra • Mathematische Techniken: Grundlegende Beweisstrategien, Vollständige Induktion • Mathematisch Strukturen: Lösung von linearen Gleichungssystemen. Vektoren, Matrizen, Determinanten • Vertiefung in folgende Richtungen: o Konvergenz und Grenzwerte von Folgen und Reihen o Schaltalgebra o Graphentheorie • Kombinatorik, Diskrete Stochastik		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, Vorrechnung von Beispielaufgaben, Applets zur Veranschaulichung, freiwillige Übungsaufgaben, evtl. Tutorium Übung: selbstständiges Lösung von Übungsaufgaben		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführte Übung Übung: erfolgreich absolvierte Übung (PVL)		
	• Peter Hartmann: Mathematik für Informatiker. Vieweg, 4. Auflage Februar 2006, Kapitel 1-3, 12-14		

Modulbezeichnung	**Grundlagen der Technischen Informatik**	Kürzel	GT/GTP
Lehrveranstaltung(en)	Vorlesung: Grundlagen der Technischen Informatik Praktikum: Grundlagen der Technischen Informatik	Semester	1
Arbeitsaufwand	36 Std. Vorlesung, 12 Std. Praktikum, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Franz Korf	SWS	3+1
Dozenten	Prof. Dr. Reinhard Baran, Prof. Dr. Andreas Meisel, Prof. Dr. Franz Korf	Sprache	deutsch
Voraussetzungen	keine	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • können grundlegenden Begriffe der Informatik benennen • können grundlegende Rechnerarchitekturkonzepte wie zum Beispiel die von-Neumann-Architektur beschreiben • verstehen die Grundkonzepte der Instruction-Set-Architektur eines Prozessors • können einfache Assembler-Programme für einen ausgewählten Prozessors erstellen		
Inhalte	Elementare Begriffe und Definitionen der Informatik wie zum Beispiel: • Informatik, Information, Daten, Algorithmus, Programm, Theoretische Informatik, Praktische Informatik, Technische Informatik, Hochsprachen, • Assembler, Syntax und Semantik von Programmiersprachen • Darstellung von Daten im Computer • Rechnerarchitekturgrundlagen auf Instruction Set Architecture Ebene • Elemente eines Rechners, von Neumann, Harvard • Grundlegende Konzepte der Instruction Set Architecture einer ausgewählten Mikroprozessorfamilie • Abbildung von Daten- und Kontrollstrukturen prozeduraler Hochsprachen in maschinennahe Implementierungen		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, freiwillige Übungsaufgaben Praktikum: Programmieren in 2-er-Gruppen		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführtes Praktikum Praktikum: erfolgreiche Bearbeitung der Praktikumsaufgaben (PVL)		
Literatur	• A. S. Tanenbaum, James Goodman: Computerarchitektur • J. L. Hennessy, D. A. Patterson: Computer Architecture • W. Hohl: ARM Assembly Language: Fundamentals and Techniques • Bernd Becker, Rolf Drechsler, Paul Molitor (Autor): Technische Informatik - Eine Einführung. Pearson Studium (IT), Gebundene Ausgabe, Februar 2005		

Modulbezeichnung	**Programmiermethodik I**	Kürzel	PM1
Lehrveranstaltung(en)	Vorlesung: Programmiermethodik I	Semester	1
Arbeitsaufwand	48 Std. Vorlesung, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Michael Schäfers	SWS	4+0
Dozenten	Prof. Dr. Martin Hübner, Prof. Dr. Philipp Jenke, Prof. Dr. Michael Schäfers, Prof. Dr. Bernd Kahlbrandt, Prof. Dr. Axel Schmolitzky, N.N.	Sprache	deutsch
Voraussetzungen	keine	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • sind mit einem Programmierparadigma so gut vertraut, dass sie darin abstrakt beschriebene Programmierprobleme effizient mit korrektem ausführbarem Quelltext lösen können, • kennen die wesentlichen Eigenschaften der Basistypen (Zahlen, Zeichen, Wahrheitswerte, Zeichenketten) einer Programmiersprache und können diese zielgerichtet einsetzen, • können die Kernabstraktionen (Methoden/Prozeduren/Funktionen, Klassen) einer Programmiersprache geeignet parametrisieren, • kennen den Unterschied zwischen Wertsemantik und Referenzsemantik und können Referenzsemantik aktiv in eigenen Programmtexten anwenden, • kennen geeignete Darstellungen für Sammlungen von Werten/Objekten in einer Programmiersprache und können diese problemangemessen einsetzen, • und können eine abstrakte Problembeschreibung in einen programmierbaren Algorithmus übertragen.		
Inhalte	• Abstraktionsmechanismen - Funktionale Abstraktion - Datenabstraktion (ADT) - Kontrollabstraktion (z.B. Iteratoren, Streams) • Objektorientierung - Polymorphie: Overloading, Overriding, Dynamische Bindung • Ausgewählte Elemente objektorientierter Bibliotheken, z.B. - Collections - Streams, Channels • Typisierungskonzepte - Dynamische vs. statische Typisierung		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit		
Literatur	• Literaturhinweise werden je nach Programmiersprache und aktuellem Stand in der Vorlesung gegeben		

Modulbezeichnung	**Programmiertechnik**	Kürzel	PT/PTP
Lehrveranstaltung(en)	Vorlesung: Programmiertechnik Praktikum: Programmiertechnik	Semester	1
Arbeitsaufwand	24 Std. Vorlesung, 24 Std. Praktikum, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Michael Schäfers	SWS	2+2
Dozenten	Prof. Dr. Martin Hübner, Prof. Dr. Philipp Jenke, Prof. Dr. Michael Schäfers, Prof. Dr. Bernd Kahlbrandt, Prof. Dr. Axel Schmolitzky, N.N.	Sprache	deutsch
Voraussetzungen	keine	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden beherrschen handwerklichen Programmierfertigkeiten, elementaren Programmiertechniken sowie eine modernen Entwicklungsumgebung mit den zugehörigen Werkzeugen (z.B. Editor, Debugger, Testautomation, Version Control System,...) • können eine technische Realisierung von Systemen im Kleinen durchführen		
Inhalte	• Elementare Programmiertechniken und Syntax einer modernen Programmiersprache: - primitive Datentypen, - Unicode, - Arrays, - Referenztypen, - Sequenz, Selektion, Iteration		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, Beispielaufgaben, freiwillige Übungsaufgaben, Tutorium Praktikum: Bearbeitung der Aufgaben in 2-er Gruppen, Begutachtung der Lösungen, Gesprächsführung		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführtes Praktikum Praktikum: erfolgreiche Bearbeitung der Praktikumsaufgaben (PVL)		
Literatur	• Literaturhinweise werden je nach Programmiersprache und aktuellem Stand in der Vorlesung gegeben		

Modulbezeichnung	**Mess- und Sensortechnik**	Kürzel	MS/MSP
Lehrveranstaltung(en)	Vorlesung: Mess- und Sensortechnik Praktikum: Mess- und Sensortechnik	Semester	1
Arbeitsaufwand	36 Std. Vorlesung, 12 Std. Praktikum, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Andreas Meisel	SWS	3+1
Dozenten	Prof. Dr. Andreas Meisel, Prof. Dr. Wolfgang Fohl, Prof. Dr. Bernd Schwarz, Prof. Dr. Reinhard Baran, Prof. Dr. Tim Tiedemann	Sprache	deutsch
Voraussetzungen	Schulmathematik und -physik	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • können Messschaltungen mit resistiven Sensoren dimensionieren, aufbauen und auf elektrischer Ebene mit einem Computer verbinden. Dabei setzen sie grundlegende Methoden des Gleichstromkreises ein, wie die Knoten- und Maschenregel, das ohmsche Gesetz, Strom- und Spannungsteiler sowie das Ersatzquellenverfahren. Darüber hinaus können Sie Grundkonzepte der elektr. Messtechnik einsetzen, wie die Erweiterung des Messbereichs und die Ausschlagbrücke. Mit einem Multimeter können Sie die Schaltung kontrollieren (U-/I-/R) und die Ergebnisse interpretieren. • können Messschaltungen mit kapazitiven und induktiven Sensoren dimensionieren, aufbauen und auf elektrischer Ebene mit einem Computer verbinden. Dabei setzen Sie die komplexe Wechselstromrechnung ein. Mit Multimeter und Oszilloskop können Sie die Schaltung kontrollieren (U~/I~) und die Ergebnisse interpretieren. • können einfache Programme zur Messsignalaufnahme und -auswertung entwickeln, z.B. eine getaktete Messwertaufnahme (Timer, Polling), die Bestimmung von Maximal- oder Minimalwerten, die Überwachung von Schwellwerten oder eine grafische Ausgabe.		
Inhalte	• atomistische Grundlagen der Elektrotechnik, Strom, Spannung, Widerstand, ideale Quellen, Knoten- und Maschenregel, ohmsches Gesetz, Spannungs- und Stromteiler, Ersatzquellenverfahren • Potentiometer, resistive Sensoren (z.B. Länge, Druck, Licht, Temperatur, Kraft) und Ausschlagbrücke • Ladung, elektr. Feld, Influenz, Kapazität, Plattenkondensator, kapazitive Sensoren (z.B. Druck, Schall) • Durchflutung, magn. Fluss, magn. Widerstand, Induktivität, induktive Sensoren (z.B. Weg, Abstand) • Wechselstrom (Amplitude, Frequenz, Phase), komplexe Beschreibung von Wechselgrößen, Wechselstromverhalten von Kapazitäten und Induktivitäten • Messung von Wechselgrößen, Wechselstrom-basierte Sensoren (Metalldetektoren, Differentialsensoren wie Druckmessdose, Präzisionstaster oder Beschleunigungssensoren)		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, freiwillige Übungsaufgaben, evtl. Tutorium Praktikum: Elektrische Grundlagenversuche mit Programmieren in 2-er Gruppen		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführtes Praktikum Praktikum: erfolgreiche Bearbeitung der Praktikumsaufgaben (PVL)		
Literatur	• Literaturhinweise werden je nach Programmiersprache und aktuellem Stand in der Vorlesung gegeben		

Modulbezeichnung	**Automatentheorie und Formale Sprachen**	Kürzel	AF/AFÜ
Lehrveranstaltung(en)	Vorlesung: Automatentheorie und Formale Sprachen Übung: Automatentheorie und Formale Sprachen	Semester	2
Arbeitsaufwand	36 Std. Vorlesung, 12 Std. Übung, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Bettina Buth	SWS	3+1
Dozenten	Prof. Dr. Bettina Buth, Prof Dr. Köhler-Bussmeier, Prof. Dr. Franz Korf, Prof. Dr. Michael Neitzke, Prof. Dr. Julia Padberg	Sprache	deutsch
Voraussetzungen	Modul Diskrete Mathematik oder äquivalente Kenntnisse Anmerkung: benötigt werden Kenntnisse von formalen Beweisen, Prädikatenlogik, speziell Induktion	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • können formale Beweise erläutern und selber durchführen • können formale Modelle in Form von Automaten, Regulären Ausdrücken und Grammatiken erstellen • können Zusammenhänge zwischen Automatenmodellen, regulären Sprachen und Grammatiken herstellen und Modelle ineinander überführen • können formale Spezifikationen auf Problemstellungen der realen Welt anwenden		
Inhalte	• Grundlagen der Automatentheorie • Endliche Automaten • Reguläre Ausdrücke und Sprachen • Kontextfreie Grammatiken und Sprachen • Eigenschaften Kontextfreier Sprachen • Vertiefung in einer der folgenden Richtungen: Kellerautomaten, zelluläre Automaten, Zeitautomaten, Modellierung mit formalen Methoden		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, freiwillige Übungsaufgaben, Gruppenarbeit Übung: selbstständiges Lösen von Aufgaben		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführte Übung Übung: erfolgreiche Bearbeitung der Übungsaufgaben (PVL)		
Literatur	• J.E. Hopcroft, P. Motwani, J.D. Ullmann: Einführung in die Automatentheorie, Formalen Sprachen und Komplexitätstheorie. Pearson Studium • M.Kreuzer, S. Kühling: Logik für Informatiker. Pearson Studium		

Modulbezeichnung	**Programmiermethodik II**	Kürzel	PM2/PMP2
Lehrveranstaltung(en)	Vorlesung: Programmiermethodik II Praktikum: Programmiermethodik II	Semester	2
Arbeitsaufwand	36 Std. Vorlesung, 12 Std. Praktikum, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Michael Schäfers	SWS	3+1
Dozenten	Prof. Dr. Martin Hübner, Prof. Dr. Philipp Jenke, Prof. Dr. Michael Schäfers, Prof. Dr. Bernd Kahlbrandt, Prof. Dr. Axel Schmolitzky	Sprache	deutsch
Voraussetzungen	keine	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • beherrschen fortgeschrittene Fertigkeiten und Programmiertechniken, • können Systeme im Kleinen objektorientiert modellieren, auch mit Hilfe der UML und • beherrschen die technische Realisierung objektorientierter und interaktiver Systeme.		
Inhalte	• Ausgewählte Elemente objektorientierter Bibliotheken, z.B.: - GUI-Frameworks - Generics • Metasprachliche Konzepte - Annotationen - Reflection • Vertiefungen - Typ- vs. Implementierungshierarchie - elementare Entwurfsmuster - Modellierungen (anhand UML) - nebenläufige bzw. asynchrone Programmierung - systematische Fehlerbehandlung • Korrektheit - Design by Contract (Assertions, Invarianten, Teststrategien)		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, Beispielaufgaben, freiwillige Übungsaufgaben, evtl. Tutorium Praktikum: Bearbeitung der Aufgaben in 2-er Gruppen, Begutachtung der Lösungen, Gesprächsführung		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführtes Praktikum Praktikum: erfolgreiche Bearbeitung der Praktikumsaufgaben (PVL)		
Literatur	• Literaturhinweise werden je nach Programmiersprache und aktuellem Stand in der Vorlesung gegeben		

Modulbezeichnung	**Datenbanken**	Kürzel	DB/DBP
Lehrveranstaltung(en)	Vorlesung: Datenbanken Praktikum: Datenbanken	Semester	2
Arbeitsaufwand	36 Std. Vorlesung, 12 Std. Praktikum, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Thomas Clemen	SWS	3+1
Dozenten	Prof. Dr. Wolfgang Gerken, Prof. Dr. Stefan Sarstedt , Prof. Dr. Thomas Clemen, Prof. Dr. Olaf Zukunft	Sprache	deutsch
Voraussetzungen	Programmieren I (PR1, PRP1), Mathematische Grundlagen (MG)	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • kennen die Einsatzgebiete und Grenzen von relationalen Datenbanksystemen (RDBS), • beherrschen den Prozess des Datenbankentwurfs, • kennen die theoretischen Grundlagen relationaler Datenbanksysteme, • können einfache Datenbankanwendungen entwickeln, • beherrschen die Anfragesprache SQL im Rahmen des Standards und • haben einen Überblick über Alternativen zum relationalen Modell (sog. NoSQL-Datenbanken).		
Inhalte	• Grundkonzepte relationaler Datenbanksysteme • der logische Entwurf und die Überführung in das technische Design • Implementierung und Befüllung von Datenbanksystemen • Anfragen und Transaktionen • programmiersprachliche Schnittstellen • Alternativen zum relationalen Modell		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, freiwillige Übungsaufgaben Praktikum: Aufgabenbearbeitung in Kleingruppen		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführtes Praktikum Praktikum: erfolgreiche Bearbeitung der Praktikumsaufgaben (PVL)		
Literatur	• A. Kemper, A. Eickler: Datenbanksysteme - Eine Einführung. De Gruyter 2015. • R.A. Elmasri, S.B. Navathe: Grundlagen von Datenbanksystemen. Pearson 2009 • W. Gerken: Datenbanksysteme für Dummies. Wiley 2016.		

Modulbezeichnung	**Grundlagen der systemnahen Programmierung**	Kürzel	GS/GSP
Lehrveranstaltung(en)	Vorlesung: Grundlagen der systemnahen Programmierung Praktikum: Grundlagen der systemnahen Programmierung	Semester	2
Arbeitsaufwand	24 Std. Vorlesung, 24 Std. Praktikum, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Franz Korf	SWS	2+2
Dozenten	Prof. Dr. Reinhard Baran, Prof. Dr. Andreas Meisel, Prof. Dr. Franz Korf	Sprache	deutsch
Voraussetzungen	Grundlagen der Technischen Informatik	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • können prozedurale Programme in C erstellen, • verstehen und benutzen Zugriffe auf Hardwareschnittstellen, • können Anwendungen zum Ansteuern einfacher Sensoren und Aktoren erstellen und • verstehen die Schnittstelle zwischen einer Hochsprache und einem Assembler.		
Inhalte	• Methoden und Techniken zur prozeduralen - und maschinennahen Programmierung • für die technische Informatik relevante Konzepte einer hardwarenahen Programmiersprache, wie zum Beispiel C • Speicherverwaltung auf Hochsprachen- und Maschinenebene • C Projekte: Verwaltung, Modultechnik, Bibliotheken, Standardbibliotheken • Interfaces zur Verzahnung von Hochsprachen und Assembler • elementare Zeitmessungen • grundlegende Muster zur Programmierung eingebetteter Systeme (z.B. Direct Digital Control Programmiermuster)		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, freiwillige Übungsaufgaben Praktikum: Programmieren in 2-er-Gruppen		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführtes Praktikum Praktikum: erfolgreiche Bearbeitung der Praktikumsaufgaben (PVL)		
Literatur	• A. S. Tanenbaum, James Goodman: Computerarchitektur • Groll, U. Bröckl, M. Dausmann: C als erste Programmiersprache • B.W. Kernighan, D.M. Ritchie: Programmieren in C • W. Hohl: ARM Assembly Language: Fundamentals and Techniques • Michael Pont: Patterns for Time-Triggered Embedded Systems - Building Reliable Applications with the 8051 Family of Microcontrollers (with CD-ROM). Hardcover, Juli 2001		

Modulbezeichnung	**Analysis und lineare Algebra**	Kürzel	AA / AAÜ
Lehrveranstaltung(en)	Vorlesung: Analysis und lineare Algebra Übung: Analysis und lineare Algebra	Semester	2
Arbeitsaufwand	36 Std. Vorlesung, 12 Std. Übung, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Stephan Pareigis	SWS	3+1
Dozenten	Prof. Dr. Stephan Pareigis , Prof. Dr. Reinhard Baran, Lehrbeauftragte	Sprache	deutsch
Voraussetzungen	Diskrete Mathematik	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • beherrschen die analytischen Rechentechnik mit elementaren Funktionen einer Variablen • können mit Parametern, Transformationen und graphischen Darstellungen umgehen		
Inhalte	• Kombinatorik, Teilbarkeit, Restklassen • Gruppen, Vektorräume, Homomorphismen • Skalarprodukt, Orthogonalität, Norm und Metrik • Differenzialrechnung, Integralrechnung, Fourierreihen		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, Vorrechnen von Beispielaufgaben, Applets zur Veranschaulichung, freiwillige Übungsaufgaben, evtl. Tutorium Übung: selbstständiges Lösung von Übungsaufgaben		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführte Übung Übung: erfolgreiche Bearbeitung aller Aufgaben (PVL)		
Literatur	• Peter Hartmann: Mathematik für Informatiker. Vieweg 4. Auflage Februar 2006, Kapitel 4-8, 10, 15-17		

Modulbezeichnung	**Signalverarbeitung und Stochastik**	Kürzel	SV / SVÜ
Lehrveranstaltung(en)	Vorlesung: Signalverarbeitung und Stochastik Übung: Signalverarbeitung und Stochastik	Semester	3
Arbeitsaufwand	36 Std. Vorlesung, 12 Std. Übung, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Andreas Meisel, Prof. Dr. Stephan Pareigis	SWS	3+1
Dozenten	Prof. Dr. Stephan Pareigis, Prof. Dr. Wolfgang Fohl Prof. Dr. Andreas Meisel, Prof. Dr. Bernd Schwarz, Prof. Dr. Tim Tiedemann	Sprache	deutsch
Voraussetzungen	Analysis und lineare Algebra	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • können auf der Basis von Elementarsignalen und einfachen Grundoperationen (Gewichtung, Verschiebung, Dehnung) periodische und abgetastete Funktionen beschreiben. • können mit Hilfe der Fouriertransformation (von Elementarsignalen) und den Theoremen der FT die Frequenzeigenschaften von Impulsen, periodischen und abgetasteten Signalen beschreiben. • können den Einfluss von Frequenzbegrenzungen (Filter: TP, HP, BP) und Zeitbegrenzungen (Fensterfunktionen) auf die Zeit- und Frequenzeigenschaften der Signale beschreiben. • können Abtastsignale mit Hilfe der z-Transformation beschreiben und sowohl die Differenzengleichungen als auch die Übertragungsfunktionen digitaler Filter angeben. • können mit Hilfe von Werkzeugen digitale Filter (FIR, IIR) entwerfen und diese auf Zielsystemen (z.B. eingebetteten Systemen) unter Benutzung von Hardwaretimern und Interrupts realisieren. • können Verfahren zur Bestimmung von statistischen Signalkenngrößen (Erwartungswert, Varianz, Momente) und Ähnlichkeitsmaßen (Kreuzkorrelation, Autokorrelation) realisieren und die Ergebnisse interpretieren. • können ausgewählte Merkmale (zerocrossings, Momente, usw.) aus verschiedenen Anwendungsbereichen (Audioprocessing, medizinische Signalverarbeitung, usw.) aus Signalen extrahieren.		
Inhalte	• Elementarsignale, Grundoperationen (Verschieben, Dehnen, Abtastung, ..), Faltungsalgebra • Fouriertransformation, Faltungssatz, Diracstoß und Diracstoßfolgen • Abtastung, Periodisierung, ideale Filter, diskrete FT • Abtastsysteme, z-Transformation, Differenzengleichungen, Übertragungsfunktion, FIR- und IIR-Filter • Diskrete Fouriertransformation, Bandbegrenzung (Filterung), Zeitbegrenzung (Fensterung) • Entwurf und Realisierung digitaler Filter, schmalbandige Filter, Görtzel-Algorithmus • Realisierung von Abtastsystemen (Filter, Audioeffekte, …) mit Hardwaretimern und Interrupts unter Nutzung einer hardwarenahen Programmiersprache (z.B. C). • Grundbegriffe der Statistik: Wahrscheinlichkeitsbegriff, Zufallsvariablen, Verteilungs- und Verteilungsdichtefunktion, stat. Unabhängigkeit, bedingte Wahrscheinlichkeit, Satz von Bayes • Signale als Zufallsprozesse: Erwartungswert, Varianz, Schätzung stat. Signalkenngrößen • Ausgewählte Signalmerkmerkmale aus verschiedenen Anwendungsbereichen		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, Matlab Beispiele, freiwillige Übungsaufgaben Übung: Bearbeitung der Übungsaufgaben in 2-er-Gruppen mit Abnahmegespräch		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführte Übung Übung: erfolgreiche Bearbeitung der Übungsaufgaben (PVL)		
Literatur	• Hoffmann und Quint: Signalverarbeitung mit MATLAB und Simulink. Oldenbourg Verlag • D.Ch. von Grüningen: Digitale Signalverarbeitung. Fachbuchverlag Leipzig		

Modulbezeichnung	**Algorithmen und Datenstrukturen**	Kürzel	AD / ADP
Lehrveranstaltung(en)	Vorlesung: Algorithmen und Datenstrukturen Praktikum: Algorithmen und Datenstrukturen	Semester	3
Arbeitsaufwand	36 Std. Vorlesung, 12 Std. Praktikum, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Stephan Pareigis	SWS	3+1
Dozenten	Prof. Dr. Stephan Pareigis , Prof. Dr. Bernd Kahlbrandt	Sprache	deutsch
Voraussetzungen	Diskrete Mathematik, Programmieren I + II	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • haben Kenntnisse zur Bewertung und selbstständigen Entwicklung von Algorithmen und beherrschen die dazu erforderlichen Datenstrukturen.		
Inhalte	• Abstrakte Datentypen Abstrakte Datentypen, Signatur, Vor- und Nachbedingungen, Lineare Listen, Stack, Queue • Algorithmen und Komplexität Komplexität, Aufwandsfunktion, Asymptotischer Aufwand, Landau-Notation, Darstellung in logarithmischen Skalen, rekursive Verfahren • Sortierverfahren Nicht-rekursive Sortierverfahren und deren Komplexität, Rekursive Sortierverfahren (Quicksort und Mergesort) und deren Komplexität • Bäume und Graphen Implementationsmöglichkeiten, Binäre Suchbäume und Komplexität, Graphen und kürzeste Wege (Dijkstra) • Hashfunktionen offene Adressierung und separate chaining, Kollisionsvermeidungsstrategien		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, Vorrechnen von Beispielaufgaben, Applets zur Veranschaulichung, freiwillige Übungsaufgaben, evtl. Tutorium Praktikum: selbstständiges Lösung von Praktikumsaufgaben		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführtes Praktikum Praktikum: erfolgreiche Bearbeitung der Praktikumsaufgaben (PVL)		
Literatur	• Ottman, Widmayer: Algorithmen und Datenstrukturen. Spektrum Verlag, 4.Auflage, 2002		

Modulbezeichnung	**Software Engineering I**	Kürzel	SE1/SEP1
Lehrveranstaltung(en)	Vorlesung: Software Engineering I Praktikum: Praktikum Software Engineering I	Semester	3
Arbeitsaufwand	36 Std. Vorlesung, 12 Std. Praktikum, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Bettina Buth	SWS	3+1
Dozenten	Prof. Dr. Bettina Buth , Prof. Dr. Zhen Ru Dai, Prof. Dr. Thomas Lehmann, Prof. Dr. Stephan Pareigis, Prof. Dr. Philipp Jenke	Sprache	deutsch
Voraussetzungen	PT, PM1, PM2	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • können die Zusammenhänge zwischen Programmierung und Software Engineering erläutern • können Fachbegriffe des Bereichs Software Engineering erläutern • können vorgegebene Modelle (insbesondere Prozessmodelle und Modelle der UML Notationsfamilie) im Rahmen einer Softwareentwicklung interpretieren und analysieren, anwenden und in Code umsetzen. • können die Besonderheiten des Software Engineering bei technischen Anwendungen benennen und erläutern • können programmiernahe Techniken und Werkzeuge im Rahmen des Software Engineering einsetzen, speziell für die Anforderungsanalyse, die Modellierung und den Test. • können statische Analysen ebenso wie grundlegende dynamische Testtechniken anwenden • können sich eigenständig neue Techniken und Methoden und deren Anwendung an konkreten Beispielen aneignen • können fachliche Zusammenhänge schriftlich ausdrücken, speziell Dokumentation von Anforderungen, Architekturen, Tests erstellen und bewerten • können englische Dokumentation interpretieren		
Inhalte	• Grundkonzepte und Ziele des Software Engineering, insbesondere Lebenszyklusmodelle, Anforderungserhebung- und Analyse, Architekturen, Qualitätssicherung, • Modellierungstechniken, speziell auch im Hinblick auf technische Systeme, • Spezielle Entwicklungstechniken, wie z. B. Refactoring, Code Analyse, Continuous Integration, DSLs, Code Generierung, • Requirements Engineering, speziell Anforderungserhebung und -analyse • Prozessmodelle, speziell traditionelle Lebenszyklusmodelle im Vergleich zu agilen Entwicklungsansätzen • Im Praktikum: Arbeiten mit aktuellen Software-Entwicklungsumgebungen speziell zu Modellierung, Test, Refactoring, Requirements Engineering – auch: Beherrschen von Werkzeugketten (Konfiguration, Einsatz)		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, Übungsaufgaben, Gruppenarbeit Praktikum: Aufgabenbearbeitung in 2er-4er Teams, Diskussionsforen mit Betreuer		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführtes Praktikum Praktikum: erfolgreiche Bearbeitung der Praktikumsaufgaben (PVL)		
Literatur	• Sommerville: Software Engineering. Pearson Education • Spillner, Linz: Basiswissen Softwaretest. dpunkt Verlag • Hammerschall, Beneken: Software Requirements		

Modulbezeichnung	**Betriebssysteme**	Kürzel	BS/BSP
Lehrveranstaltung(en)	Vorlesung: Betriebssysteme Praktikum: Betriebssysteme	Semester	3
Arbeitsaufwand	36 Std. Vorlesung, 12 Std. Praktikum, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Franz Korf	SWS	3+1
Dozenten	Prof. Dr. Wolfgang Fohl, Prof. Dr. Tim Tiedemann, Prof. Dr. Martin Becke, Prof. Dr. Franz Korf	Sprache	deutsch
Voraussetzungen	PT, PM1, GT, GS	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • verstehen die Architektur, die Konzepte und die Funktionsweise moderner Betriebssysteme sowie des Zusammenspiels von Hard- und Software, • verstehen die Konzepte zur Implementierung systemnaher Software, • können das Verhalten von Computersystemen analysieren und beschreiben und • können Grundkonzepte der nebenläufigen Programmierung anwenden		
Inhalte	• Architekturen und Betriebsarten • Prozess- und Thread-Konzept, Scheduling • Synchronisation, Interprozesskommunikation, Deadlocks • Virtualisierungskonzepte • Hauptspeicherverwaltung, Virtueller Speicher • Verwaltung externer Geräte • Dateisysteme • Schutzmechanismen, Sicherheitsaspekte • Exemplarische Betrachtung aktueller Betriebssysteme		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, freiwillige Übungsaufgaben Praktikum: Programmieren in 2-er Gruppen		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführtes Praktikum Praktikum: erfolgreiche Bearbeitung der Praktikumsaufgaben (PVL)		
Literatur	• Andrew S. Tanenbaum: Modern Operating Systems. Pearson Studium Verlag Silberschatz, Galvin, Gagne: Operating System Concepts with Java. John Wiley & Sons • Eduard Glatz: Betriebssysteme - Grundlagen, Konzepte, Systemprogrammierung. dpunkt Verlag • Williams Stallings: Operating Systems		

Modulbezeichnung	**Intelligente Sensorsysteme**	Kürzel	ISS/ISSP
Lehrveranstaltung(en)	Vorlesung: Intelligente Sensorsysteme Praktikum: Intelligente Sensorsysteme	Semester	3
Arbeitsaufwand	36 Std. Vorlesung, 12 Std. Praktikum, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Tim Tiedemann	SWS	3+1
Dozenten	Prof. Dr. Andreas Meisel, Prof. Dr. Wolfgang Fohl, Prof. Dr. Bernd Schwarz, Prof. Dr. Reinhard Baran, Prof. Dr. Tim Tiedemann, Prof. Dr. Thomas Lehmann	Sprache	deutsch
Voraussetzungen	MS, PT, PM1, PM2, GS	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • können Sensorsysteme zur Erfassung physikalischer Größen entsprechend dem geforderten Einsatzbereich, den jeweiligen Datenblättern und Herstellerdokumentationen bewerten und auswählen, • sind in der Lage, die physikalischen Prinzipien, die Sensoren (etwa für Annäherung, Entfernung, Druck, Temperatur, Durchfluss, Beschleunigung, Drehung und Kraft) zugrunde liegen, zu beschreiben, • können die wesentlichen Grundlagen und Randbedingungen für den Einsatz komplexer Sensoren (wie beispielsweise Kamera, Mikrofon, Laserscanner) zusammenfassen, • können intelligente Sensoren mit Hilfe der Herstellerdokumentation über gängige Sensor-Bussysteme, wie z.B. I2C, CAN oder Ethernet, an Rechnersysteme anbinden, • können die Daten intelligenter Sensoren mit Hilfe der Herstellerdokumentation auf Rechnersystemen erfassen und Daten verschiedener Sensorsysteme kombinieren, • können beispielhaft mittels verschiedener Algorithmen einfache Rohsensordaten zu abstrakteren/komplexeren Informationen (sog. Features) aufbereiten/vorverarbeiten.		
Inhalte	• Physikalische Messprinzipien beispielsweise für Annäherung, Entfernung, Temperatur, Druck, Durchfluss, Beschleunigung, Drehung und Kraft. Wirkungsweise von z.B. Mikrofonen, Kameras, Laserscannern, Radarsensoren. • Konkrete Beispiele typischer Sensoren, insbesondere typischer intelligenter Sensoren. • Exemplarische Erörterung von Datenblättern. • Elektrischer Anschluss von Sensoren. Dazu – soweit für das Verständnis der Sensoren benötigt – Eigenschaften und Anwendungen von Dioden, Transistoren oder Operationsverstärkern. • Vorverarbeitung: Abtastung, Glättung, Mittelwertbildung, grafische Darstellung. • Generierung und Auswahl von Features aus Rohdaten, z.B. mittels maschineller Lernverfahren, Bildverarbeitungs- oder statistischer Methoden. Sensordatenfusion. • Schnittstellen/Bussysteme von intelligenten Sensoren: I2C, Ethernet, u.a. Rechnersysteme für den Anschluss von Sensoren, z.B. Industrie-PCs, on-board units (OBU), aber auch Arduino, Raspberry Pi oder Beaglebone Black. • Zeitverhalten von Rechnersystemen zur Auswertung von Sensordaten. • Programmtechnische Konfiguration und Programmierung von Sensoren. • Anwendungsbeispiele wie KFZ-Sensorik, industrielle Steuerungs- und Automatisierungstechnik, Smart Gadgets.		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, freiwillige Übungsaufgaben. Praktikum: Sensor-Anwendungen mit Programmieren in 2-er-Gruppen		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführtes Praktikum Praktikum: erfolgreiche Bearbeitung der Praktikumsaufgaben (PVL)		
Literatur	• Meijer et al. (Eds): Smart Sensor Systems. Wiley • Kimmo Karvinen et. al.: Sensoren. dpunkt-Verlag		

Modulbezeichnung	**Embedded System Engineering**	Kürzel	ESE/ESEP
Lehrveranstaltung(en)	Vorlesungen: Software Engineering II, Embedded Programming, System- und Echtzeitprogrammierung Praktikum: Praktikum Embedded System Engineering	Semester	4
Arbeitsaufwand	72 Std. Vorlesung, 24 Std. Praktikum, 204 Std. Eigenarbeit/Selbststudium	CP	12
Modulverantwortliche(r)	Prof. Dr. Franz Korf, Prof. Dr. Thomas Lehmann, Prof. Dr. Stephan Pareigis	SWS	6+2
Dozenten	Prof. Dr. Bettina Buth , Prof. Dr. Zhen Ru Dai, Prof. Dr. Wolfgang Fohl, Prof. Dr. Franz Korf, Prof. Dr. Thomas Lehmann, Prof. Dr. Stephan Pareigis, Prof. Dr. Bernd Schwarz, Prof. Dr. Philipp Jenke	Sprache	deutsch
Voraussetzungen	MS, PT, PM1, PM2, GS, BS, SE1	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • können konkrete Projekte unter Anwendung von Projektmanagementmethoden planen und durchführen, insbesondere unter Berücksichtigung unterschiedlicher Entwicklungsprozesse • können spezielle Aspekte eingebetteter Systeme, wie z. B. Safety, Robustness oder Fehlertoleranz, beim Entwurf und Entwicklung berücksichtigen • können technische Systeme auf der Systemebene modellieren und eingebettete Software als Teilsystem modellbasiert entwerfen, realisieren und testen • können verteilte nebenläufige Echtzeitsysteme auf Basis objektorientierter Architekturen unter Verwendung spezialisierter implementierungsnaher Pattern entwerfen und implementieren • können Echtzeitsysteme entsprechend ihres Einsatzes und Anforderungen klassifizieren • können ausgewählter Aspekte eingebetteter Echtzeitsysteme im Zusammenhang mit einem Echtzeitbetriebssystems (z. B. Zeitmanagement, Interrupt Management, Kommunikation, I/O) konzipieren und umsetzen		
Inhalte	• Vorlesung Embedded Programming (EP) - C++-Sprachprinzipien und Eignung für Echtzeitprogrammierung - Kapselung von Systemaufrufen, plattformunabhängige Programmierung - Reaktor Pattern, Speicherverwaltung, Fabrikpattern, Functor Pattern - Kommunikationsprinzipien und Message Passing - Generische Programmierung mit Templates - Code-Instrumentalisierung, Logging, Debuggen von Echtzeitsystemen • Vorlesung Software Engineering II (SE2) - Einführung in das Projektmanagement, Projektplanung und Teamorganisation - Grundlagen des Konfigurations- und Versionsmanagements - Einführung in das Qualitätsmanagement - Spezielle Aspekte des Software Engineering bei eingebetteten Systemen - Spezialisierungen von Techniken der Modellierens und des Testens auf eingebettete Echtzeitsysteme • Vorlesung System- und Echtzeitprogrammierung (SY) - Grundlagen, Klassifizierung, Einsatz, Anforderungen von eingebetteten Systemen. - Vertiefung ausgewählter Aspekte eingebetteter Echtzeitsysteme unter Einsatz eines Echtzeitbetriebssystems (z. B.: Zeitmanagement, Interrupt Management, Kommunikation, I/O). - Scheduling-Techniken für Echtzeitanwendungen - Umsetzung der Aspekte anhand eines ausgewählten Echtzeitbetriebssystems (z. B. QNX) • Praktikum - Projektorientierte Software-Entwicklung für ein ES auf Basis eines ausgewählten Echtzeitbetriebssystems (z. B. QNX). Das Praktikum erfordert im besonderen Maße auch Techniken und Fähigkeiten der vorausgesetzten Module. - Teamorientierte Softwareentwicklung		

Modulbezeichnung	**Embedded System Engineering**	Kürzel	ESE/ESEP
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht mit verschiedenen Medien Praktikum: Projektorientierte Entwicklung eines eingebetteten System in kleinen Teams (4-6 Personen)		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführtes Praktikum Praktikum: erfolgreiche Bearbeitung des Projektes (PVL)		
Literatur	• Sommerville: Software Engineering. 9. Auflage, 2012, Pearson Education • (GOF) Gamma, Erich et al.: Design Patterns. Addison Wesley, 1995 • Douglas Schmidt, Stephen Huston: C++ Network Programming. Volume 1, Addison Wesley, 2002 • (POSA 2) Schmidt, Stal, Rohnert, Buschmann: Pattern-Oriented Software Architecture. Wiley, 1999 • Bruce Powell Douglas: Real-Time Design Patterns. Addison-Wesley, 2003 • Andrei Alexandrescu: Modernes C++ Design. MITP, 2003 • Kalev, Danny: The ANSI/ISO C++ Profesional Programmer's Handbook.Que Corperation. 1999 • Meyers, Scott: Effective C++. 2nd Edition. Addison Wesley, 1998 • Meyers, Scott: More Effective C++. Addison Wesley, 1998 • A. Burns und A. Wellings: Real-time systems and programming languages : Ada 95, real-time Java and real-time POSIX. 3. ed, Pearson Addison-Wesley, 2003. • G. C. Buttazzo: Hard real-time computing systems: predictable scheduling algorithms and applications. Nr. 23 in Real-time systems series. 2nd, Springer, 2005. • H. Kopetz: Real-time systems - Design principles for distributed embedded applications. 8. ed, Kluwer Acad., 2004. • R. Krten: Getting started with QNX Neutrino : a guide for realtime programmers. QNX Software Systems. 2009. • W. Stallings: Operating systems : internals and design principles. 7. ed., Pearson, 2012.		

Modulbezeichnung	**Rechnernetze**	Kürzel	RN/RNP
Lehrveranstaltung(en)	Vorlesung: Rechnernetze Praktikum: Rechnernetze	Semester	4
Arbeitsaufwand	36 Std. Vorlesung, 12 Std. Praktikum, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof . Dr. Thomas Schmidt	SWS	3+1
Dozenten	Prof. Dr. Thomas Schmidt, Prof. Dr. Martin Becke	Sprache	deutsch
Voraussetzungen	Programmieren, Betriebssysteme	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • verstehen die Konzepte und die Funktionsweisen von Rechnernetzen • können auf der Socket-Schnittstelle basierende Client- / Server-Anwendungen erstellen • können Methoden und Werkzeuge für die Konfiguration und Administration von Rechnernetzen anwenden • können die Leistungsdaten von Rechnernetzen bewerten		
Inhalte	• Grundlagen der Datenübertragung • Protokolle der Sicherungsschicht • Protokolle und Dienste der Netzwerk- und Transportschicht, insbesondere die TCP/IP-Protokollsuite • Einführung in wichtige Anwendungsschichtprotokolle • Sicherheit in Netzwerken • Einführung in Netzwerkmanagement • Socket-Programmierung		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, freiwillige Übungsaufgaben Praktikum: Aufgabenbearbeitung in 2-er-Gruppen		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführtes Praktikum Praktikum: erfolgreiche Bearbeitung der Praktikumsaufgaben (PVL)		
Literatur	• Andrew S. Tanenbaum, David Wetherall: Computer Networks • Larry L. Peterson, Bruce S. Davie: Computer Networks – A Systems Approach • James F. Kurose, Keith W. Ross: Computer Networking: A Top-Down Approach Featuring the Internet • Ch. Meinel, H. Sack: Internetworking		

Modulbezeichnung	**Mustererkennung und Machine Learning**	Kürzel	MM/MMP
Lehrveranstaltung(en)	Vorlesung: Mustererkennung und Machine Learning Praktikum: Mustererkennung und Machine Learning	Semester	4
Arbeitsaufwand	36 Std. Vorlesung, 12 Std. Praktikum, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Andreas Meisel	SWS	3+1
Dozenten	Prof. Dr. Andreas Meisel, Prof. Dr. Wolfgang Fohl, Prof. Dr. Tim Tiedemann	Sprache	deutsch
Voraussetzungen	AA, SV	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • können mit ein- oder mehrdimensionale Zeitsignalen (z.B. von einem Sensorarray), Bildern oder auch Bildsequenzen datentechnisch umgehen (Datenstrukturen, effiziente Verarbeitung). • können zu einem gegebenen Mustererkennungsproblem (MEP) mit Hilfe von Faltungsoperatoren, Korrelation, spektralen Merkmalen, statistischen Merkmalen und ausgewählten höheren Merkmalen geeignete Merkmalsextraktionsverfahren auswählen und realisieren. • können zu einem MEP ein passendes Klassifikatorkonzept angeben. Dabei berücksichtigen sie, dass dieses analytisch (z.B. Fourierdeskriptoren), regelbasiert (scharf oder fuzzy) oder lernbasiert (statistisch, neuronal) sein kann und wägen die unterschiedlichen Vor- und Nachteile gegeneinander ab. • können passend zum Klassifikationskonzept einen geeigneten Klassifikator auswählen, konfigurieren und realisieren. Sie können lernende Systeme trainieren, testen und in die Zielanwendung integrieren.		
Inhalte	• Stufen der Mustererkennung (Vorverarbeitung, Merkmalsextraktion, Klassifikation) • analytische, regelbasierte und lernbasierte Verfahren der Mustererkennung • Grundzüge der Wahrscheinlichkeitstheorie, bedingte Wahrscheinlichkeit, Bayes-Regel • ausgewählte Verfahren zur Merkmalsextraktion, Hauptkomponentenanalyse • Klassifikation durch Funktionsapproximation (Polynomklassifikator, radiale Basisfunktionen) • Curse of dimensionality (Problem hochdimensionaler Merkmalsräume), Gradientenabstieg • Clusteringverfahren (z.B. k-Means, Self-organizing-map) • Grundlagen neuronaler Netze (NN): Neuron, Aktivierungsfunktion, Multilayer-Perzeptron Backpropagation u. Erweiterugen), Training, Overfitting • Support Vektor Machines und Kernelmethoden • Moving-window-NN und Recurrent NN für Zeitsignale, Dynamic-time-Warping • Konzepte des Deep-Learning • Ausgewählte Themen: Convolutional Neural Networks, LSTM-Netze		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, Vorrechnung von Beispielaufgaben, Applets und Matlab-Skripte zur Veranschaulichung, freiwillige Übungsaufgaben Praktikum: selbstständiges Lösung von Praktikumsaufgaben in 2-er Gruppen		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführtes Praktikum Praktikum: erfolgreiche Bearbeitung der Praktikumsaufgaben (PVL)		
Literatur	• Duda, Hart, Stork: Pattern Classification. Wiley interscience • Bishop: Pattern Recognition and Machine Learning. Springer • Haykin: Neural Networks and Learning Machines. Pearson international • Goodfellow , Bengio , Courville: Deep Learning - Adaptive Computation and Machine Learning. MIT Press		

Modulbezeichnung	**Verteilte Systeme**	Kürzel	VS/VSP
Lehrveranstaltung(en)	Vorlesung: Verteilte Systeme Praktikum: Verteilte Systeme	Semester	5
Arbeitsaufwand	36 Std. Vorlesung, 12 Std. Praktikum, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Thomas Schmidt	SWS	3+1
Dozenten	Prof. Dr. Thomas Schmidt, Prof. Dr. Martin Becke	Sprache	deutsch
Voraussetzungen	Programmieren, Betriebssysteme, Rechnernetze	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • kennen und verstehen die Verteilten Systemen zugrunde liegenden Probleme sowie die einschlägigen Leistungskategorien zu ihrer Lösung • beherrschen und verstehen einschlägige verteilte Programmiermodelle, können eine exemplarische Auswahl praktisch anwenden und können den zugehörigen Lösungsraum technisch beurteilen • beherrschen und verstehen einschlägige Algorithmen zur Realisierung verteilter Anwendungsszenarien und können diese auf reale Probleme übertragen • können eine System-Infrastruktur eines VS entwerfen und realisieren • können eine Middleware eines VS entwerfen und realisieren • können ein Konzept für replizierte Daten entwerfen und realisieren		
Inhalte	• Eine Einführung im Sinne einer Beschreibung der charakteristischen Eigenschaften verteilter Systeme • Interprozesskommunikation zwischen verteilten Prozessen und einschlägige Programmiermodelle • Namensdienste und exemplarische Anwendungen • Zeit , Koordination und Übereinstimmung • Wahlen, Wechselseitiger Ausschluss und Verteilte Transaktion • Verteilte Dateisysteme und Replikation • Ausgewählte Anwendungen verteilter replizierender Systeme • Sicherheit in verteilten Systemen		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, freiwillige Übungsaufgaben Praktikum: Aufgabenbearbeitung in 2-er-Gruppen		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführtes Praktikum Praktikum: erfolgreiche Bearbeitung der Praktikumsaufgaben (PVL)		
Literatur	• G. Coulouris, J. Dollimore, T. Kindberg: Verteilte Systeme. Pearson Studium • Andrew S. Tanenbaum, Maarten van Stehen: Verteilte Systeme - Grundlagen und Paradigmen. Pearson Studium		

Modulbezeichnung	**Betriebswirtschaft**	Kürzel	BW/BWÜ
Lehrveranstaltung(en)	Vorlesung: Betriebswirtschaft Übung: Betriebswirtschaft	Semester	5
Arbeitsaufwand	36 Std. Vorlesung, 12 Std. Übung, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Martin Hübner	SWS	3+1
Dozenten	Prof. Dr. Martin Hübner, Prof. Dr. Gerken	Sprache	deutsch
Voraussetzungen	keine	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • verstehen rechtliche, finanzielle und organisatorische Strukturen von Unternehmen, • verstehen die Bedeutung von wirtschaftlichen Vorgehensweisen und können entsprechende Controlling-Instrumente anwenden • können Kostenberechnungen selbstständig durchführen, • können Investitionsentscheidungen nach betriebswirtschaftlichen Kriterien treffen.		
Inhalte	• Das Unternehmen als System • Rechtsformen und Aufbauorganisation • Ablauforganisation und Methoden zu ihrer Beschreibung • Grundlagen der Finanzbuchhaltung (Buchführung und Jahresabschluss) • Kosten- und Leistungsrechnung • Finanzierung und Investitionsrechnung		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, freiwillige Übungsaufgaben Übung: selbstständiges Lösung von Übungsaufgaben		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich absolvierte Übung Übung: erfolgreiche Bearbeitung aller Aufgaben (PVL)		
Literatur	• G. Wöhe: Einführung in die allgemeine BWL. Verlag Franz Vahlen • A. J. Schwab: Managementwissen für Ingenieure. Springer-Verlag • Dietmar Vahs, Jan Schäfer-Kunz: Einführung in die BWL. Schäffer-Poeschel Verlag • Siegfried Schmolke, Manfred Deitermann: Industrielles Rechnungswesen IKR. Winklers Verlag		

Modulbezeichnung	**Cyber-physische Systeme**	Kürzel	CPS/CPSP
Lehrveranstaltung(en)	Vorlesung: Cyber-physische Systeme Praktikum: Cyber-physische Systeme	Semester	6
Arbeitsaufwand	36 Std. Vorlesung, 12 Std. Praktikum, 132 Std. Eigenarbeit/Selbststudium	CP	6
Modulverantwortliche(r)	Prof. Dr. Martin Becke, Prof. Dr. Franz Korf	SWS	3+1
Dozenten	Prof. Dr. Martin Becke, Prof. Dr. Franz Korf, NN	Sprache	deutsch
Voraussetzungen	Programmieren, Rechnerstrukturen, Maschinennahe Programmierung, Verteilte Systeme	Häufigkeit	semesterweise
Lernziele und Kompetenzen	Die Studierenden • verstehen die Anforderungen einer Verknüpfung von realen Objekten und Prozessen mit informationsverarbeitenden Objekten und Prozessen, • können reale und informationsverarbeitende Objekte in lokalen, wie auch globalen Kommunikationsnetze verbinden • verstehen die Kopplung von physikalischen und virtuellen Objekten		
Inhalte	Die Veranstaltungen fokussiert sich auf domainübergreifende und offene Systeme. • Einführung von CPS als neues Kommunikationsparadigma, das auf den verteilten Systemen aufbaut. • Adaption geänderter oder sich ändernder Umgebungsbedingungen • virtuelle Abbildung von physikalischen Interaktionen • unterschiedliche QoS Anforderungen innerhalb eines Systems. • Besprechung der Modellierung zur Umsetzung z.B. aus den Bereichen Car-to-X, Smart Grid, Produkt- oder Produktionssystemen • Modellierungs-, Spezifikations- und Verifikationstechniken für offene Systeme		
Lehr- und Lernformen	Vorlesung: Seminaristischer Unterricht, Tafelarbeit, Präsentation, freiwillige Übungsaufgaben Praktikum: Programmieren in 2-er Gruppen		
Studien- und Prüfungsleistungen	Vorlesung: nach Festlegung als benotete Klausur, benotete mündliche Prüfung, benotetes Referat, benotete Hausarbeit Prüfungsvorleistung (PVL): erfolgreich durchgeführtes Praktikum Praktikum: erfolgreiche Bearbeitung der Praktikumsaufgaben (PVL)		
Literatur	• Rajeev Alur: Principles of Cyber-Physical Systems. MIT Press • Christian Manzei , Linus Schleupner: Industrie 4.0 im internationalen Kontext - Kernkonzepte, Ergebnisse, Trends. VDE-Verlag • Dr. William Stallings: Foundations of Modern Networking: SDN, NFV, QoE, IoT and Cloud. Addison-Wesley		