COURSE DESCRIPTION CARD - SYLLABUS

Course name
Emulation techniques

Course

Field of study
Computing
Area of study (specialization)
Edge Computing
Level of study
Second-cycle studies
Form of study
full-time

Year/Semester
1/2
Profile of study
general academic
Course offered in
Polish
Requirements
elective

Number of hours

Lecture
15
Laboratory classes
15
Other (e.g. online)
Tutorials
Projects/seminars

Number of credit points

3

Lecturers

Responsible for the course/lecturer:
dr inż. Mariusz Naumowicz
email: mariusz.naumowicz@put.poznan.pl
tel. +48 61 665-2364
Faculty of Computing and Telecommunications
ul. Piotrowo 3 60-965 Poznań

Responsible for the course/lecturer:

Prerequisites

The student starting the course should have basic knowledge of operating systems and electronics. They should also understand the need to expand their competences and be ready to cooperate as part of the team.

Course objective

- Provide students with knowledge related to modern techniques of emulation of embedded systems.

- Familiarizing students with modern methods of designing, testing and prototyping embedded systems with the use of emulators.

- Developing students' skills in solving complex design problems in the field of building and testing embedded systems.

- Developing teamwork skills in students.

Course-related learning outcomes

Knowledge

1. Has ordered and theoretically founded general knowledge related to key issues in the field of computer science - [K2st_W2]

2. Has advanced detailed knowledge of selected issues in the field of computer science - [K2st_W3]

3. Has knowledge of development trends and the most important new achievements of computer science and other selected related scientific disciplines - [K2st_W4]

4. Has advanced and detailed knowledge of the life cycle of hardware or software information systems - [K2st_W5]

Skills

1. Can obtain information from literature, databases and other sources (in Polish and English), integrate them, interpret and critically evaluate them, draw conclusions and formulate and exhaustively justify opinions - [K2st_U1]

2. Can plan and carry out experiments, including measurements and computer simulations, interpret the obtained results and draw conclusions as well as formulate and verify hypotheses related to complex engineering problems and simple research problems - [K2st_U3]

3. Can use analytical, simulation and experimental methods to formulate and solve engineering tasks and simple research problems - [K2st_U4]

4. Can - when formulating and solving engineering tasks - integrate knowledge from various areas of computer science (and, if necessary, also knowledge from other scientific disciplines) and apply a system approach, also taking into account non-technical aspects - [K2st_U5]

5. can assess the usefulness and the possibility of using new achievements (methods and tools) and new IT products - [K2st_U6]

Social competences

1. understands that in computer science knowledge and skills very quickly become obsolete - [K2st_K1]

2. Understands the importance of using the latest knowledge in the field of computer science in solving research and practical problems - [K2st_K2]

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Formative assessment:

a) in the field of lectures: on the basis of answers to questions about the material discussed in previous lectures,

b) in the field of laboratories: on the basis of the assessment of the current progress in the implementation of tasks,

Summative assessment:

a) in the field of lectures, verification of the assumed learning outcomes is carried out by an oral exam combined with the project defense, in case of doubts the written part (an electronic test on the Moodle platform);

b) in the field of laboratories, verification of the assumed learning outcomes is carried out by means of a design test and an assessment of the tasks performed during each laboratory meeting;

Getting extra points for activity during classes, especially for:

- discussion of additional aspects of the issue,

- the effectiveness of applying the acquired knowledge while solving a given problem,

- the ability to cooperate as part of a team practically carrying out a detailed task in the laboratory.

Programme content

The lecture program includes the following topics:
Introduction to Linux, building the kernel, building modules (in and out of the tree), configuration, devicetree, bootargs, booting Linux. Introduction to tools supporting the design and testing of drivers. The structure of the driver, issues related to the operation of the driver in the operating system. Presentation of the basic issues of driver design on the example of a character device. Controller interaction with hardware (memory mapping, register access, continuous memory for DMA). Driver implementation that configures peripheral. Linux kernel interrupt handling, implementation of the interrupt handler in drivers (registration / dereregistration, definition in devicetree, multithreading). Linux subsystems. Construction of drivers used for communication in embedded systems.

Laboratory classes are conducted in the form of 2-hour lab exercises, preceded by a 2-hour instructional session at the beginning of the semester. Exercises are carried out by 2-person teams.
The program of laboratory classes includes the following topics:
Preparation of the development environment necessary to program drivers for a dedicated embedded

system. Configuring, modifying and building the Linux kernel. Programming the character controller. Copying data between user space and the kernel. IOCTLe. User space application for interaction with the driver. Interaction with hardware (memory mapping, access to registers, continuous memory for DMA). Implementation of the driver that configures peripheral. Interrupts (as Linux handles interrupts), implementation of the interrupt handler in drivers (registration / dereregistration, definition in devicetree, multithreading). Linux subsystems. Implementation of the I2C driver. Sysfs, an implementation of the industrial I / O subsystem in the driver.

Teaching methods

1. Lecture with multimedia presentation (diagrams, formulas, definitions, etc.) supplemented by the content of the board.
2. Laboratory exercises: multimedia presentation, presentation illustrated with examples given on the board and performance of tasks given by the teacher - practical exercises.

Bibliography

Basic

1. John L. Hennessy, John L. Hennessy, Computer Architecture: A Quantitative Approach, 4th Edition. Elsevier, 2007. ISBN: 0123704901.
2. Eldad Eilam, Reversing: Secrets of Reverse Engineering, Wiley, 2011. ISBN: 0764574817.

Additional

1. Noam Nisan, Shimon Schocken, The Elements of Computing Systems: Building a Modern Computer from First Principles, The MIT Press, 2005. ISBN: 0262640686.

Breakdown of average student's workload

	Hours	ECTS
Total workload	75	3,0
Classes requiring direct contact with the teacher	30	1,5
Student's own work (literature studies, preparation for laboratory classes, preparation for tests, techical reports preparation) [1]	45	1,5

1 delete or add other activities as appropriate