COURSE DESCRIPTION CARD - SYLLABUS

Course name
Linux drivers

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
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

-Providing students with knowledge related to modern embedded systems and the Linux operating system and its use in IoT systems.

- Familiarizing students with modern methods of designing, testing and prototyping drivers in Linux for embedded systems.

- Developing students' ability to solve complex design problems in the field of embedded systems and operating systems.

- Developing teamwork skills in students.

Course-related learning outcomes

Knowledge

1.Has advanced and in-depth knowledge of broadly understood IT systems as well as methods and tools used for their implementation, especially those related to building the hardware layer of reprogrammable systems - [K2st_W1]

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

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

4. He knows advanced methods, techniques and tools used in solving complex engineering tasks and conducting research in the selected area of computer science - [K2st_W6]

Skills

1. When formulating and solving engineering tasks, he/she is able to integrate knowledge from various areas of computer science (and, if necessary, also knowledge from other scientific disciplines) and to apply a systemic approach, also taking into account non-technical aspects - [K2st_U5]

2. Is able to assess the usefulness and the possibility of using new achievements (methods and tools) and new IT products – [K2st_U6]

3. He can, by using conceptually new methods - solve complex IT tasks, including unusual tasks and tasks with a research component – [K2st_U10]

4. Can, in accordance with the given specification, design a complex device, IT system or process and implement this project using appropriate methods, techniques and tools, including adapting existing or developing new tools for this purpose – [K2st_U11]

5. Is able to determine the directions of further learning and implement the process of self-education, including that of other people - [K2st_U16]

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 pass 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. Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman., Linux Device Drivers, 3rd Edition. O'Reilly Media, Inc. 2005. ISBN: 0596005903.
2. Alberto Liberal de los Ríos, Linux Driver Development for Embedded Processors - Second Edition: Learn to develop Linux embedded drivers with kernel 4.9 LTS, Independently Published, 2018. ISBN: 1729321828.

Additional

1. Daniel Bovet and Marco Cesati, Understanding the Linux Kernel, Third Edition, O'Reilly & Associates Inc., 2005. ISBN: 0596005652.

Breakdown of average student's workload

	Hours	ECTS
Total workload	75	3,0
Classes requiring direct contact with the teacher	30	1,0
Student's own work (literature studies, preparation for laboratory classes, preparation for tests, technical reports preparation) [1]	45	2,0

1 delete or add other activities as appropriate