COURSE DESCRIPTION CARD - SYLLABUS

Course name
Firmware systems

Course

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

Year/Semester
2/3
Profile of study
general academic
Course offered in
Polish
Requirements
elective

Number of hours

Lecture
30
Laboratory classes
30
Other (e.g. online)
Tutorials
Projects/seminars
20

Number of credit points

5

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 for embedded systems, electronics and programming. 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 firmware systems.

- Familiarizing students with modern methods of designing, testing and prototyping firmwares.

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

- 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 regarding building the hardware layer of reprogrammable systems - [K2st_W1]

2. Has advanced detailed knowledge of selected issues in the field of computer science, especially related to the construction of embedded systems - [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. Knows advanced methods, techniques and tools used in solving complex engineering tasks and conducting research in a selected area of computer science - [K2st_W6]

Skills

1. 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]

2. 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 systemic approach, also taking into account non-technical aspects - [K2st_U5]

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

4. Can make a critical analysis of the existing technical solutions and propose their improvements (improvements) - [K2st_U8]

5. Can - using, among others conceptually new methods - solve complex IT tasks, including non-standard tasks and tasks with a research component - [K2st_U10]

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

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 exam (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:

Boot Flow, FSP, SMM, Requirements, SMBIOS and ACPI, UEFI Secure Boot, UEFI Shell API, Scripts & CLI, Network Boot, Driver Execution Environment(DXE), UEFI Drivers , developing an open source software feature for firmware.

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 run selected emulators. Building your own embedded system based on the existing structures in the emulator. Preparation of the operating system for the emulated embedded system. Analysis of embedded system signals by means of an emulator.

Emulation of a complex IT system based on several IoT devices. Automatic testing with the use of an emulator.

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. Vincent Zimmer, Michael Rothman, Suresh Marisetty, Beyond BIOS, De Gruyter 2017. ISBN: 9781501505836.

Additional

1. Unified Extensible Firmware Interface - https://uefi.org/sites/default/files/resources/UEFI%20Spec%202_6.pdf

Breakdown of average student's workload

	Hours	ECTS
Total workload	125	5,0
Classes requiring direct contact with the teacher	80	3,5
Student's own work (literature studies, preparation for laboratory classes, preparation for tests, technical reports preparation) [1]	45	1,5

1 delete or add other activities as appropriate