COURSE DESCRIPTION CARD - SYLLABUS

Course name
Edge systems architecture [S2Inf1-PB>ASB]

Course

Field of study
Computing

Year/Semester
1/1

Area of study (specialization)
Edge Computing

Profile of study
general academic

Level of study
second-cycle

Course offered in
Polish

Form of study
full-time

Requirements
compulsory

Number of hours

Lecture
30

Laboratory classes
30

Other
0

Tutorials
0

Projects/seminars
0

Number of credit points

5,00

Coordinators

dr inż. Adam Turkot
adam.turkot@put.poznan.pl

Lecturers

Prerequisites

A student starting this course should have basic knowledge of network protocols HTML, TCP, UDP, SSH, ICMP, programming in C and C++, Bash command interpreter, Python and JavaScript. He/she should also understand the necessity to extend his/her competences / be ready to cooperate within a team. Moreover, in terms of social competences, a student must present such attitudes as honesty, responsibility, perseverance, cognitive curiosity, creativity, personal culture, respect for other people.

Course objective

To familiarise students with the methodology of designing edge system architectures. To provide students with extended knowledge in the scope of edge systems architectures and the hardware and software solutions used in them. Developing skills of programming techniques that ensure: efficient use of hardware resources of boundary systems. Optimal, for a given task, realisation of applications with the use of an appropriate hardware platform, with support for dedicated peripheral modules and taking into account the requirements related to energy saving and computational efficiency. Mastering the communication techniques between the controller and digital and analogue elements of edge systems. To introduce students to the possibilities and limitations of building edge systems based on controllers and single board computers. To train students" abilities to work in a team by realizing project elements and combining them into a whole

Course-related learning outcomes

Knowledge:
1.has advanced and deepened knowledge of broadly understood information systems, theoretical foundations of their construction, and methods, tools and environments
2. has well-ordered and theoretically grounded general knowledge connected with the key issues of computer science
3. knows advanced methods, techniques and tools applied in solving complex engineering tasks and carrying out research work in the selected field of computer science.

Skills:
1. is able to acquire information from literature, databases and other sources (in polish and english), integrate it, interpret and critically evaluate it, draw conclusions and formulate and exhaustively justify opinions.
2. is able to use analytical, simulation and experimental methods to formulate and solve engineering tasks and simple research problems
3. is able to integrate knowledge from different fields of computer science (and if necessary from other scientific disciplines) and apply a system approach, taking into account also non-technical aspects, when formulating and solving engineering tasks
4. is able to make a critical analysis of existing technical solutions and propose their improvements (enhancements)
5. is able to solve complex information technology tasks, including atypical tasks and tasks with a research component, using new conceptual methods.
6. is able to design a complex device, information system or process to a given specification taking into account non-technical aspects and implement the design - at least in part - using appropriate methods, techniques and tools including adapting existing tools or developing new ones.

Social competences:
1. understands that in computer science knowledge and skills become obsolete very quickly.
2. understands the importance of using the latest knowledge in computer science to solve research and practical problems.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Learning outcomes presented above are verified as follows:
Formal evaluation:
(a) for lectures: on the basis of answers to questions concerning the material discussed in previous lectures
(b) for laboratorys/exercises: based on an assessment of the current progress of the tasks,
Summary evaluation:
a) in the scope of lectures, the verification of the assumed educational results is carried out by: an oral examination combined with the defence of the project, in case of doubt a written part (an electronic test on the Moodle platform);
b) as far as laboratories are concerned, the verification of the assumed educational results is carried out by: oral examination combined with the project defence: evaluation of the student"s preparation for individual sessions of laboratory classes by checking the preparation of given projects/exercises and evaluation of skills related to the implementation of laboratory classes, continuous evaluation, during

each class (oral answers), rewarding the increase in the ability to use the learned principles and methods, evaluation of documentation created systematically along with the progress of project work; documentation prepared partly during the classes and partly after their completion; this evaluation also includes the ability to work in a team, evaluation and defence by the student of the report on project implementation,

Programme content

Fundamentals of edge systems architecture. Techniques for efficient use of hardware resources. Evaluation of hardware capabilities. Programming environment. Techniques of programming. Methods of code optimization. User interface. Hardware and software solutions for power management. Software security techniques (program integrity, resistance against unauthorized copying) Controller architectures. Local and shared resources, consequences of sharing resources. Buses in distributed systems. Hardware and software techniques for increasing reliability of communication links. Communication techniques and protocols used for communication with peripherals and between controllers and in the cloud. Increasing the reliability of unmanned systems, techniques to ensure energy management of autonomous systems.
Laboratory classes are conducted in the form of fifteen 2-hour exercises, held in the laboratory, preceded by a 2-hour instruction session at the beginning of the semester. The exercises are carried out by 2-person teams of students.
The labs include: Programming single board computer architectures. Programming of microcontroller system architectures. Communication protocols in particular SPI, I2C, UART. IoT communication protocols. Programming modular architectures on the example of Colibri iMX7.

Course topics

Fundamentals of edge systems architecture. Techniques for efficient use of hardware resources. Evaluation of hardware capabilities. Programming environment. Techniques of programming. Methods of code optimization. User interface. Hardware and software solutions for power management. Software security techniques (program integrity, resistance against unauthorized copying) Controller architectures. Local and shared resources, consequences of sharing resources. Buses in distributed systems. Hardware and software techniques for increasing reliability of communication links. Communication techniques and protocols used for communication with peripherals and between controllers and in the cloud. Increasing the reliability of unmanned systems, techniques to ensure energy management of autonomous systems.
Laboratory classes are conducted in the form of fifteen 2-hour exercises, held in the laboratory, preceded by a 2-hour instruction session at the beginning of the semester. The exercises are carried out by 2-person teams of students.
The labs include: Programming single board computer architectures. Programming of microcontroller system architectures. Communication protocols in particular SPI, I2C, UART. IoT communication protocols. Programming modular architectures on the example of Colibri iMX7.

Teaching methods

Teaching methods:
lecture: multimedia presentation, presentation illustrated with examples given on the board, presentation of selected student solutions.
laboratory exercises: practical exercises, performing experiments, discussion, teamwork.

Bibliography

Basic
Linux w systemach embedded Bis, Marcin. Wydawnictwo btc, 2011
Building embedded Linux systems Yaghmour, Karim. O"Reilly, cop. 2003.
Additional
Wbudowane systemy mikroprocesorowe Timofiejew, Aleksander., Akademia Podlaska (Siedlce). Wydawnictwo.Wydawnictwo Akademii Podlaskiej, 2010.

Breakdown of average student's workload

	Hours	ECTS
Total workload	125	5,00
Classes requiring direct contact with the teacher	60	2,50
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	65	2,50