# Round table discussion on
# Selected Problems in Programming in Mechatronics Education

Agenda for event organized by project PAEPEDE

Peter Bober

## Contents

# 1 Introduction

This material is for participants to prepare of the round table discussion for event organized by project PAEPEDE: Pathways to Excellence in Power Electronics and Electrical Drives Education.

## 1.1 The topic, objectives and outcomes

| | |
|---|---|
| **Topic** | Selected Problems in Programming in Mechatronics Education |
| **Objectives** | 1. Identify Key Challenges: To pinpoint the most significant obstacles faced by educators and students in teaching and learning programming within the context of mechatronics.<br><br>2. To exchange successful strategies and methods used by different institutions or educators to teach programming in mechatronics. |
| **Outcomes** | 1. A comprehensive list of challenges that need to be addressed, such as curriculum gaps, resource limitations, or student engagement issues.<br><br>2. A collection of effective teaching practices that can be adopted or adapted by participants to improve their own programs. |

## 1.2 Basic information

| | |
|---|---|
| Project Name | PAEPEDE: Pathways to Excellence in Power Electronics and Electrical Drives Education |
| Project ID | 22420041, |
| Grant agency | International Visegrad Fund, Visegrad Grants Call 2-2024 |
| Project partners | **Technical University of Košice, Slovakia**<br>Faculty of Electrical Engineering and Informatics, Dept. of Electrical Engineering and Mechatronics<br>**Wroclaw University of Science and Technology, Poland**<br>Faculty of Electrical Engineering, Department of Electrical Machines, Drives and Measurements<br>**Széchenyi István University, Hungary**<br>AUDI Hungária Faculty of Automotive Engineering, Department of Power Electronics and Electrical Drives |
| Date | 14.5.2025 |
| Time | 10:00 – 12:00 (can be changed) |
| Location | Technical University of Košice, Letná 1/9, Košice, room L9-B219 |

# 2 Round table discussion agenda

1. Welcome and Introduction

2. Participant Introductions

3. Presentation of Key Points

4. Open Discussion

5. Preparation of Ideas

6. Summary

# 3 Instructions for participants

We kindly ask participants of the round table discussion to follow instructions below. Sample text is in the next chapter.

1. Prepare short information on the use of programming in current educational subjects (PowerPoint, PDF, DOCX, web page,…). Content:
    1.1. Subjects, topics
    1.2. Programming languages, required skills
    1.3. Teaching methods
2. Prepare list of challenges and obstacles faced by educators and students in teaching and learning programming.
3. Prepare a list of the practices and methods you use to overcome challenges or obstacles:
    3.1. Current practices and methods in use.
    3.2. Practices and methods you would like to try.

# 4 Teaching and using programming in Košice

## 4.1 Introduction of participant

| | |
|---|---|
| Institution | **Technical University of Košice**, Slovakia<br>Faculty of Electrical Engineering and Informatics<br>Dept. of Electrical Engineering and Mechatronics |
| Study Programme | **Applied Electrical Engineering,** Bachelor programme<br>Graduates of the Applied Electrical Engineering study program have knowledge of electric drives, machines, devices, power semiconductor systems, pneumatic and hydraulic systems, including their modeling and control. They can apply the control algorithms for electrical systems, carry out their technical service, and test the products using diagnostic tools. They can apply their knowledge in the following fields – construction, installation, operation, maintenance, and management of electrical systems.<br><br>**Applied Electrical Engineering,** Master programme<br>Graduates of the Applied Electrical Engineering master's program possess a comprehensive understanding of core electrical engineering topics, including the latest scientific advancements and practical expertise in:<br>• Electrical Engineering Fundamentals: Mastery of power electronics, power semiconductor components, electronic circuits, electrical machines, instruments, and drives.<br>• Control and Programming: Proficiency in using signal processors and programmable controllers for the control and programming of electrical systems.<br>• Complex Electrical Systems: Ability to solve problems related to the operation and management of complex electrical systems, such as continuous production lines, converters for electromobility and renewable energy sources, controlled drives, and industrial automated systems. |
| Name | Peter Bober |
| Teaching Subjects | **Fundamentals of Algorithms and Programming**<br>Bc, first semester<br>**Programming**<br>Bc, second semester<br>**Pneumatics and Hydraulic Systems** (construction, electrical and PLC control) Bc second semester |
| Research interest | Motor Control, Optimization, Predictive models |
| Tools used | Matlab, Matlab/Simulink, Real Time Programming, Embeded Systems, |
| Languages | Assemblers, FORTRAN, Basic, Pascal, Modula, C, PHP, HTML, CSS, JavaScript, VBA, SQL, Matlab |

## 4.2 Subjects overview

Subjects overview in study programme **Applied Electrical Engineering**, Bachelor and Master.

| | | Language, Software, Platform | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Year Term | Subject | C | Unix | Arduino | Matlab/Simulink | Excel | Embedded ATmega | PLC prog. | Control Web | ABB RobotStudio | Embedded TI teal-time MCU | ROS Robot Operating System |
| | **Bachelor** | | | | | | | | | | | |
| 1 W | **Fundamentals of Algorithms and Programming** (ZAP) | **X** | X | | | | | | | | | |
| 1 S | **Programming** (Progr) | **X** | X | X | | | | | | | | |
| | **Computer Applications** (CA) | | | | X | X | | | | | | |
| 2 W | Electrical Machines Fundamentals (ZELS) | | | | X | | | | | | | |
| 2 S | Industrial Systems Control (RPS) | | | | X | | | | | | | |
| | **Fundamentals of Microcomputer Programming** (ZPM) | **X** | | | | | X | | | | | |
| | Pneumatic and Hydraulic Systems (PHS) | | | | | | | X | | | | |
| | **Modelling and Simulations in Electrical Engineering** (MSE) | | | | **X** | | | | | | | |
| 3 W | **Basics of PLC Programming** (ZPLC) | | | | | | | **X** | | | | |
| | Controlled Drives (RePo) | | | | X | | | | | | | |
| | User Interface Creating (TPR) | | | | | | | | X | | | |
| | Fundamentals of Robotics (ZR) | | | | | | | | | X | | |
| | Bachelor Project | X | X | X | X | X | X | X | X | X | X | |
| | **Master** | | | | | | | | | | | |
| 1 W | **Signal Microcontrollers** (SigMCU) | **X** | | | | | | | | | **X** | |
| | Artificial Intelligence in Control of Electromechanical Systems (UIREmS) | | | | X | | | | | | | |
| | Programming Advanced Robotic Applications (NpRA) | | X | | | | | | | | | X |
| 1 S | **Automated Systems with Programmable Logic Controllers** (RsPa) | | | | | | | **X** | | | | |
| | Servo Systems (ServoIng) | | | | X | | | | | | | |
| | **Application of Signal Microcontrollers** (ASigMCU) | **X** | | | | | | | | | **X** | |
| 2 W | Electromechanical Production Systems (EVS) | | | | | | | X | | | | |
| | Diploma Project | X | X | X | X | X | X | X | X | X | X | X |

**Bold** is used for programming intensive subjects. **W** – Winter Term, **S** – Summer Term

## 4.3 Teaching methods

Often used learning methods are:

- Thematic Learning - Curriculum is unified and centered around an identified theme or topic.
- Problem based learning - Solving simplified real-world problems.
- Project-Based Learning - Completing projects that span multiple disciplines.

In Košice we use **Thematic Learning** combined with **Problem based learning** for most standard subjects. Lectures uses Thematic Learning. Laboratory or Numerical exercises and Studios uses Problem based learning.

Some subject and bachelor and diploma thesis use **Project-Based Learning**.

## 4.4 Challenges and obstacles

1. Big differences in student programming knowledge levels.
2. Insufficient Skills:
   2.1. Mathematical Skills
   2.2. Physics Skills
   2.3. Integration Skills
      2.3.1. Problem-Solving: Ability to apply mathematical and physical principles to develop algorithms and solve complex problems.
      2.3.2. Analytical Thinking: Breaking down problems into smaller, manageable parts and using logical reasoning to find solutions.
      2.3.3. Computational Thinking: Understanding how to translate real-world problems into computational models and algorithms.
3. Motivation. High demands on first-year programming lessons are demotivating.
4. Language problems coming with internationalization of higher education and migration: teaching in English, integrating foreign students learning in Slovak language.

## 4.5 Current practices and methods in use

We use methods described in section 4.3 Teaching methods.

## 4.6 Practices and methods we would like to try

To address big difference in student knowledge we can try:

1. Pre-assessment: Conduct initial assessments to gauge students' programming skills.
2. Modular Curriculum: Design the curriculum in a modular fashion, allowing students to progress through different levels at their own pace.
3. Flexible Learning Paths: Provide flexible learning paths that allow students to choose topics or projects that interest them and align with their skill levels.

Measures that require time and cost, but cannot be avoided:

- Curriculum Enhancement
- Teacher Training and Support
- Student Engagement, organize some student events, competitions
- Community and Industry Partnerships

Incorporate primary and secondary school teachers to update them on the latest teaching methods and technologies, organize workshops and seminars for them.

# 5 Supplementary Materials and Inspiration

This chapter contains general information about:

- Round table discussion
- Innovative teaching methods for programming

- Programming languages and hardware used in mechatronics education
- Specialized personnel and a diverse skill set

The chapter was compiled with the help of artificial intelligence [1].

## 5.1 Round table discussion basics

A **round table discussion** is a structured conversation where participants gather to discuss a specific topic. Here are some key features:

- **Equal Participation**: Everyone has an equal opportunity to contribute, promoting diverse perspectives[1].
- **Moderator**: A moderator guides the discussion, ensuring it stays on track and everyone gets a chance to speak[2].
- **Format**: These discussions can be in-person or virtual, and they can vary in structure. Some have a set agenda, while others are more free-form[2].
- **Purpose**: They are used in various settings, from academic and community discussions to corporate and industry meetings[3].

### 5.1.1 Preparing a round table discussion

**1. Define Objectives**

- **Clarify Goals**: Determine what you want to achieve with this discussion. Are you looking to identify common challenges, share best practices, or develop solutions?

**2. Select Participants**

- **Diverse Expertise**: Invite participants with varied backgrounds in mechatronics, including educators, industry professionals, and students. This diversity will enrich the discussion with multiple perspectives[4].

**3. Prepare the Agenda**

- **Key Topics**: Outline the main issues to be discussed. For example:
  - Challenges in teaching programming concepts.
  - Integration of new technologies in the curriculum.
  - Balancing theoretical knowledge with practical skills[4].
- **Time Allocation**: Allocate specific time slots for each topic to ensure a balanced discussion.

**4. Develop Discussion Questions**

- **Open-Ended Questions**: Prepare questions that encourage detailed responses and stimulate conversation. For example:
  - What are the most significant challenges in teaching programming within mechatronics?
  - How can we better integrate practical programming skills into the curriculum?
  - What role do industry partnerships play in enhancing mechatronics education?

**5. Assign Roles**

- **Moderator**: Choose a moderator who is knowledgeable about the topic and skilled in guiding discussions. The moderator will ensure the conversation stays on track and everyone has a chance to speak[4].
- **Note-Taker**: Assign someone to take notes and summarize key points and action items.

**6. Set Up the Venue**

- **In-Person or Virtual**: Decide whether the discussion will be in-person or virtual. Ensure the venue or virtual platform is set up to facilitate smooth communication[4].

**7. Share Pre-Reading Materials**

- **Background Information**: Provide participants with relevant articles, studies, or reports on the topic. This will help them come prepared and contribute meaningfully to the discussion[4].

**8. Facilitate the Discussion**

- **Engage Participants**: Encourage active participation and ensure everyone's voice is heard. Use the prepared questions to guide the conversation but be flexible to explore new ideas that emerge[4].

**9. Summarize and Follow-Up**

- **Summary**: At the end of the discussion, summarize the key points and any agreed-upon actions.
- **Follow-Up**: Send out a summary of the discussion and any next steps to all participants.

By following these steps, you can create a structured and engaging round table discussion that effectively addresses the selected problems in programming in mechatronics education.

References

### 5.1.2 Possible objectives for round table discussion

Possible objectives for round table discussion on "Selected Problems in Programming in Mechatronics Education" are:

**1. Identify Key Challenges**

- **Objective**: To pinpoint the most significant obstacles faced by educators and students in teaching and learning programming within the context of mechatronics.
- **Outcome**: A comprehensive list of challenges that need to be addressed, such as curriculum gaps, resource limitations, or student engagement issues.

**2. Share Best Practices**

- **Objective**: To exchange successful strategies and methods used by different institutions or educators to teach programming in mechatronics.
- **Outcome**: A collection of effective teaching practices that can be adopted or adapted by participants to improve their own programs.

**3. Explore Technological Integration**

- **Objective**: To discuss how emerging technologies (e.g., AI, IoT, robotics) can be integrated into the mechatronics curriculum to enhance programming education.

- **Outcome**: Ideas and plans for incorporating new technologies into teaching methods and course content.

**4. Enhance Industry Collaboration**

- **Objective**: To explore ways to strengthen partnerships between educational institutions and industry to provide students with practical, real-world programming experience.
- **Outcome**: Strategies for developing internships, co-op programs, and industry-sponsored projects that benefit both students and companies.

**5. Develop Curriculum Improvements**

- **Objective**: To identify areas of the current curriculum that need improvement or updating to better prepare students for the demands of the mechatronics field.
- **Outcome**: Specific recommendations for curriculum changes, including new courses, updated content, and innovative teaching methods.

**6. Promote Student Engagement**

- **Objective**: To find ways to increase student interest and engagement in programming courses within the mechatronics program.
- **Outcome**: Techniques and activities that can make programming more appealing and accessible to students.

**7. Address Resource Needs**

- **Objective**: To discuss the resources (e.g., software, hardware, training) needed to effectively teach programming in mechatronics and how to acquire them.
- **Outcome**: A plan for securing necessary resources, including potential funding sources and partnerships.

**8. Foster Continuous Improvement**

- **Objective**: To establish a framework for ongoing evaluation and improvement of programming education in mechatronics.
- **Outcome**: A set of metrics and processes for regularly assessing and enhancing the quality of programming instruction.

**9. Encourage Interdisciplinary Collaboration**

- **Objective**: To promote collaboration between different departments (e.g., computer science, engineering) to provide a more holistic education in mechatronics.
- **Outcome**: Initiatives for interdisciplinary projects and courses that combine programming with other aspects of mechatronics.

**10. Create a Support Network**

- **Objective**: To build a community of educators, students, and industry professionals who can support each other in addressing the challenges of programming education in mechatronics.
- **Outcome**: A network for ongoing communication, resource sharing, and collaboration.

By setting clear objectives, you can ensure that your round table discussion is focused and productive, leading to actionable outcomes that benefit all participants.

References

1. https://www.socialtables.com/blog/event-planning/how-to-run-roundtable-discussion-2/
2. https://www.cloudpresenter.com/blog/roundtable-discussion
3. https://www.eventbrite.co.uk/blog/roundtable-discussion-ds00/
4. https://link.springer.com/article/10.1007/s10639-023-12024-9
5. https://ocw.mit.edu/courses/2-737-mechatronics-fall-2014/
6. https://peer.asee.org/practical-skills-for-students-in-mechatronics-and-robotics-education.pdf
7. https://unmudl.com/blog/decoding-the-future-with-mechatronics-engineering-the-ultimate-guide
8. https://insights.globalspec.com/article/1019/the-slow-emergence-of-mechatronics-as-an-engineering-discipline
9. https://www.cloudpresenter.com/blog/roundtable-discussion

## 5.2 Innovative teaching methods for programming

Innovative teaching methods can make programming more engaging and effective for students. Here are some approaches you might consider:

**1. Flipped Classroom**

- **Description**: Students learn new content at home through videos or reading materials and then apply that knowledge in the classroom through exercises and projects.
- **Benefits**: This method allows for more hands-on learning and personalized instruction during class time[1].

**2. Project-Based Learning (PBL)**

- **Description**: Students work on projects that require them to apply programming skills to solve real-world problems.
- **Benefits**: PBL fosters critical thinking, collaboration, and practical application of knowledge[1].

**3. Peer Instruction**

- **Description**: Students teach each other under the guidance of the instructor. This can involve pair programming or group discussions.
- **Benefits**: Enhances understanding through teaching and promotes collaborative learning[1].

**4. Gamification**

- **Description**: Incorporating game elements like points, badges, and leaderboards into the learning process.
- **Benefits**: Increases motivation and engagement by making learning fun and competitive[1].

**5. Problem-Based Learning**

- **Description**: Students learn by solving complex, open-ended problems that mimic real-world challenges.
- **Benefits**: Develops problem-solving skills and the ability to apply knowledge in practical situations[1].

**6. Design Thinking**

- **Description**: A creative problem-solving process that involves empathizing with users, defining problems, ideating solutions, prototyping, and testing.
- **Benefits**: Encourages innovation and user-centered thinking[1].

**7. Competency-Based Learning**

- **Description**: Students progress based on their ability to demonstrate mastery of a skill or concept, rather than time spent in class.
- **Benefits**: Allows for personalized learning paths and ensures students achieve a high level of proficiency[1].

**8. Visual-Based Learning**

- **Description**: Using visual aids like diagrams, flowcharts, and animations to explain programming concepts.
- **Benefits**: Helps students better understand abstract concepts and improves retention[1].

**9. Pair Programming**

- **Description**: Two students work together at one computer, with one writing code and the other reviewing each line as it is written.
- **Benefits**: Enhances collaboration, reduces errors, and improves code quality[1].

**10. Interactive Online Platforms**

- **Description**: Utilizing platforms like Codecademy, Khan Academy, or Coursera that offer interactive coding exercises and instant feedback.
- **Benefits**: Provides a flexible and engaging way for students to practice coding at their own pace[1].

By incorporating these innovative methods, you can create a dynamic and effective learning environment that helps students develop strong programming skills.

References

1 https://learningmatters.ai/blog/top-7-innovative-teaching-methods-any-school-can-incorporate
2 https://engage2learn.org/blog/innovative-teaching-methods
3 https://blog.eie.org/7-research-based-classroom-strategies-for-teaching-computer-science-cs
4 https://onlineprograms.education.uiowa.edu/blog/innovative-approaches-to-stem-teaching-preparing-the-next-generation
5 https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006023

## 5.3 Programming languages and hardware used in mechatronics education

In mechatronics education, a variety of programming languages and hardware are used to teach students how to integrate mechanical, electrical, and computer engineering principles. Here are some commonly used tools:

**Programming Languages**

1. **C/C++**

    - **Usage**: Widely used for embedded systems and microcontroller programming.
    - **Benefits**: Provides low-level access to hardware and efficient performance[1].
2. **Python**
    - **Usage**: Popular for its simplicity and versatility, often used in control systems, data analysis, and interfacing with hardware like Raspberry Pi.
    - **Benefits**: Easy to learn and has extensive libraries for various applications[2].
3. **MATLAB/Simulink**
    - **Usage**: Used for modeling, simulation, and control system design.
    - **Benefits**: Powerful tools for mathematical computations and system simulations[3].
4. **Java**
    - **Usage**: Sometimes used for developing user interfaces and applications that require portability.
    - **Benefits**: Platform-independent and has a large ecosystem[2].

**Hardware**

1. **Microcontrollers (e.g., Arduino, PIC)**
    - **Usage**: Fundamental for teaching embedded systems and control applications.
    - **Benefits**: Affordable, easy to use, and widely supported by educational resources[2].
2. **Single-Board Computers (e.g., Raspberry Pi)**
    - **Usage**: Used for more complex projects that require a full operating system.
    - **Benefits**: Versatile, supports multiple programming languages, and has extensive community support[2].
3. **Programmable Logic Controllers (PLCs)**
    - **Usage**: Common in industrial automation and control systems education.
    - **Benefits**: Robust and reliable for real-time control applications[4].
4. **Robotics Kits (e.g., LEGO Mindstorms, VEX Robotics)**
    - **Usage**: Used to teach robotics and automation concepts.
    - **Benefits**: Engaging and hands-on learning tools that simplify complex concepts[4].
5. **Sensors and Actuators**
    - **Usage**: Essential for teaching about data acquisition and control systems.
    - **Benefits**: Provide real-world data and interaction capabilities[4].

**Integration of Software and Hardware**

- **LabVIEW**

    - **Usage**: A graphical programming environment used for data acquisition, instrument control, and industrial automation.
    - **Benefits**: Intuitive interface and strong integration with hardware[4].

By using these programming languages and hardware, students in mechatronics programs can gain practical experience and develop the skills needed to design and implement complex systems.

References

1 https://unmudl.com/blog/decoding-the-future-with-mechatronics-engineering-the-ultimate-guide
2 https://coed.asee.org/2021/11/30/use-of-open-source-software-in-mechatronics-and-robotics-engineering-education-part-i-model-simulation-and-analysis/
3 https://en.wikipedia.org/wiki/Mechatronics
4 https://ocw.mit.edu/courses/2-737-mechatronics-fall-2014/

## 5.4 Specialized personnel and a diverse skill set

Teaching programming in mechatronics requires a combination of specialized personnel and a diverse skill set to effectively cover the interdisciplinary nature of the field. Here's a breakdown of the key personnel and skills needed:

**Key Personnel**

1. **Lead Instructor**
    - **Role**: Oversees the course, delivers lectures, and guides students through complex concepts.
    - **Skills**: Expertise in programming languages (e.g., C/C++, Python, MATLAB), experience with mechatronics systems, and strong teaching abilities[1].
2. **Lab Instructor/Technician**
    - **Role**: Assists with hands-on lab sessions, maintains equipment, and supports students during practical exercises.
    - **Skills**: Proficiency in handling microcontrollers (e.g., Arduino, Raspberry Pi), sensors, actuators, and familiarity with lab safety protocols[2].
3. **Teaching Assistants (TAs)**
    - **Role**: Provide additional support to students, help with grading, and facilitate group projects.
    - **Skills**: Strong understanding of course material, good communication skills, and the ability to assist with troubleshooting and debugging[3].
4. **Industry Experts/Guest Lecturers**
    - **Role**: Offer real-world insights and share industry trends and practices.
    - **Skills**: Extensive industry experience, knowledge of current technologies and applications in mechatronics[4].

**Essential Skills**

1. **Programming Skills**

    - **Languages**: Proficiency in C/C++, Python, and MATLAB for various applications in mechatronics[1].
    - **Embedded Systems**: Experience with programming microcontrollers and developing embedded systems[2].
2. **Mechanical and Electrical Engineering Knowledge**
    - **Mechanical Design**: Understanding of mechanical components and systems, including CAD software (e.g., SolidWorks)[3].
    - **Electrical Fundamentals**: Knowledge of circuit design, sensors, and actuators[2].
3. **Control Systems**
    - **Theory and Application**: Understanding of control theory and its application in mechatronics systems[4].
    - **Tools**: Familiarity with tools like Simulink for modeling and simulation[3].
4. **Problem-Solving and Analytical Skills**
    - **Troubleshooting**: Ability to diagnose and solve issues in both hardware and software[2].
    - **Analytical Thinking**: Strong analytical skills to understand and optimize mechatronic systems[3].
5. **Communication and Collaboration**
    - **Teaching**: Effective communication skills to explain complex concepts clearly[1].
    - **Teamwork**: Ability to work collaboratively with other instructors, TAs, and students[3].
6. **Project Management**
    - **Organization**: Skills in planning and managing projects, including timelines and resources[4].
    - **Leadership**: Ability to lead and motivate students through project-based learning[3].

By assembling a team with these roles and skills, you can create a comprehensive and effective learning environment for programming in mechatronics.

References

1 https://svcte.metroed.net/Course-Information/Career-Technical-Education-Courses/Mechatronics-Engineering/index.html

2 https://climbtheladder.com/mechatronics-engineer-skills/

3 https://www.zippia.com/mechatronics-engineer-jobs/skills/

4 https://education.nsw.gov.au/teaching-and-learning/curriculum/tas/tas-curriculum-resources-7-12/tas-11-12-curriculum-resources/programming-mechatronics

# 6 Resources

[1] Copilot. (2024, September 6). Conversation on programming in mechatronics education. Microsoft Copilot. Retrieved from [https://www.bing.com/chat?q=skills%20for%20teaching%20Programming%20in%20Mechatronics&qs=ds&form=ATCVAJ](https://www.bing.com/chat?q=skills%20for%20teaching%20Programming%20in%20Mechatronics&qs=ds&form=ATCVAJ)

| | | Language, Software, Platform | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Year Term | Subject | C | Unix | Arduino | Matlab/Simulink | Excel | Embedded ATmega | PLC prog. | Control Web | ABB RobotStudio | Embedded TI teal-time MCU | ROS Robot Operating System |
| | **Bachelor** | | | | | | | | | | | |
| 1 W | **Fundamentals of Algorithms and Programming** (ZAP) | **X** | X | | | | | | | | | |
| 1 S | **Programming** (Progr) | **X** | X | X | | | | | | | | |
| | **Computer Applications** (CA) | | | | X | X | | | | | | |
| 2 W | Electrical Machines Fundamentals (ZELS) | | | | X | | | | | | | |
| 2 S | Industrial Systems Control (RPS) | | | | X | | | | | | | |
| | **Fundamentals of Microcomputer Programming** (ZPM) | **X** | | | | | X | | | | | |
| | Pneumatic and Hydraulic Systems (PHS) | | | | | | | X | | | | |
| | **Modelling and Simulations in Electrical Engineering** (MSE) | | | | **X** | | | | | | | |
| 3 W | **Basics of PLC Programming** (ZPLC) | | | | | | | **X** | | | | |
| | Controlled Drives (RePo) | | | | X | | | | | | | |
| | User Interface Creating (TPR) | | | | | | | | X | | | |
| | Fundamentals of Robotics (ZR) | | | | | | | | | X | | |
| | Bachelor Project | X | X | X | X | X | X | X | X | X | X | |
| | **Master** | | | | | | | | | | | |
| 1 W | **Signal Microcontrollers** (SigMCU) | **X** | | | | | | | | | **X** | |
| | Artificial Intelligence in Control of Electromechanical Systems (UIREmS) | | | | X | | | | | | | |
| | Programming Advanced Robotic Applications (NpRA) | | X | | | | | | | | | X |
| 1 S | **Automated Systems with Programmable Logic Controllers** (RsPa) | | | | | | | **X** | | | | |
| | Servo Systems (ServoIng) | | | | X | | | | | | | |
| | **Application of Signal Microcontrollers** (ASigMCU) | **X** | | | | | | | | | **X** | |
| 2 W | Electromechanical Production Systems (EVS) | | | | | | | X | | | | |
| | Diploma Project | X | X | X | X | X | X | X | X | X | X | X |