



AGH University of Krakow | Faculty of Mechanical Engineering and Robotics
| Department of Robotics and Mechatronics

MECHATRONIC ENGINEERING | II cycle | semester II | 2026/2027

Advanced AI Models in Engineering

Instruction:

Course guide – workflow, report structure, project themes, assessment

You will learn:

How to design and implement AI-based model for real-world engineering problems, selecting the right tools and justifying modeling decisions. Evaluate and improve those models through experimentation, comparative analysis, and self-assessment. Develop the ability to work with imperfect, domain-specific data and to select, adapt, and justify appropriate modeling strategies.

Additional materials:

- Course lectures (*obligatory*)
- [Additional supporting materials for further study]

Learning outcomes supported by this instruction:

IME2A_K01, IME2A_K02, IME2A_K03, IME2A_U01,
IME2A_U03, IME2A_U04, IME2A_U05, IME2A_U08

Course supervisor:

Ziemowit Dworakowski, zdw@agh.edu.pl

Instruction author:

Adam Machynia machynia@agh.edu.pl, Szymon Dziewic sdziewic@agh.edu.pl

1. How this course works

The course is structured around modular, medium-scale assignments that reflect real-world engineering challenges and require you to explore, adapt, and justify advanced AI models. It focuses on the meta-skills of using AI: formulating the problem, choosing a method, running experiments, and judging whether the results are actually useful. You design and test your own examples.

The course uses a **student-centered** approach. You choose **3 out of 6** projects themes (section 5). For each theme you propose a task that focuses on **the main aspect of the theme**, get it approved, implement and experiment with it, document it, and defend it. Each project runs over one project planning session, three 4-hour laboratory sessions, and a defense.

The workflow, step by step:

1. **Choose a project theme** – pick one of the 6 themes (sec. 5).
2. **Propose a task** – write a short task that targets the main idea of that theme.
3. **Get approved** – agree the scope with the teacher before you build anything.
4. **Implement it** – build the smallest thing that works, then improve it.
5. **Experiment** – compare variants and see what changes the outcome.
6. **Document** – document your work based on structure in section 3.
7. **Defend** – explain your choices, code and result to the teacher.

You are graded on the **quality of your reasoning and choices**, not on getting the best accuracy number. A result that you understand and can explain well is better than a great looking result you cannot justify.

2. Using GenAI in this course

Generative AI tools (chat assistants, code generators, coding agents, and similar) are **allowed** for learning, drafting code and getting unstuck, with one condition:

You can use GenAI – but you must own and understand every line.

What this means in practice:

- **Understand your code** – you must be able to explain what each part does and why it is there.
- **Explain your decisions** – for every method, feature, settings you choose, you need to be able to explain what you did and why.
- **Be able to change it** – you may be asked to modify your code or answer on the spot.
- **Acknowledge GenAI use** – write one or two lines on where you used a tool, which tool and how you used it (e.g. *“used an assistant to draft the data loading tool, then rewrote the normalization myself”*).

If you cannot explain something, it does not count – a correct result you cannot defend earns **zero tokens** for that part.

3. Project documentation

Every project must end with **tangible documentation**. A discussion with the teacher or live demonstration during the laboratory is **not** a substitute for it. The documentation is the record of what your team did and it's the basis for grading and for resolving any disagreement about the outcome. You also do a partial documentation during the project (section 3.3), so that your progress is recorded as it happens.

3.1 Accepted forms

You may choose the form that fits your project best. Examples of accepted forms:

- **Code repository** (e.g. GitHub, GitLab) with README that serves as the report,
- **Jupyter/Google Colab notebook** – executed, with all outputs and figures visible, and with the explanations written in text cells,
- **PDF report** – a classic technical report,
- **White paper style report** – a shorter, more brief report focused on the problem, the approach, the recommendation, and the solution with code implementation.
- **Presentation slides** (e.g. PowerPoint or PDF) – accepted only together with the code and enough written explanation to cover the seven sections below.

Whatever the form, the documentation must:

- be **submitted as a file or a link** through the channel indicated by the teacher (a link must stay accessible to the teacher until the end of the semester),
- include or link **the code** used to obtain the results in form that can be run to reproduce them (for MATLAB and Python alike),
- follow the **seven-section structure** in section 3.2
- contain the **GenAI note** (section 2)
- state **who did what** in the team.

3.2 Structure of the documentation

The seven main sections should remain fixed across all project themes. They follow the engineering reasoning process from defining the problem to evaluating and recommending a solution. The detailed requirements within each section may differ depending on the theme and specific task.

1. Task definition. What is the engineering problem, and what constitutes a successful solution? Define the objective, inputs and outputs, relevant constraints and assumptions, and the criteria used to evaluate the solution.

2. Data / system representation and preparation. Where does the data come from: measurements, simulations, synthetic generation, or a combination of these? Examine its relevant properties and potential quality issues, such as noise, missing values, outliers, imbalance, or limited coverage. Describe the required preprocessing and, where applicable, the division into training, validation, and test sets.

3. Input representation / features. What information is provided to the model or solution method, and why? Describe any feature extraction, engineering, or selection required. Consider whether the selected representation omits relevant information or introduces unnecessary redundancy.

4. Model(s) / solution method. Which model, algorithm, or solution method is used, and why is it suitable for the problem? Describe its structure, key parameters, and the training, fitting, identification, or optimization procedure. Where applicable, define a simple baseline or reference approach for comparison.

5. Experimental setup and comparative variants. What exactly is being investigated or compared? Define compared variants, the factors varied between experiments, and the conditions kept fixed to ensure a fair comparison. Specify the evaluation metrics before analyzing the results and use repeated runs where randomness may affect the outcome.

6. Results, validation and interpretation. Present and visualize the main results, with emphasis on validation using unseen data or operating conditions where applicable. Examine both successful and unsuccessful cases, identify relevant failure modes, and interpret the results in engineering terms rather than reporting numerical differences alone.

7. Conclusions, limitations and recommendations. Provide a clear answer to the original task or research question. State which solution is recommended and under what conditions, discuss its main limitations and trade-offs, and indicate possible directions for further improvement or investigation.

Note: Not every element will be equally relevant in every project. Focus on the elements that are important for the specific task and briefly justify any significant elements that are not considered.

3.3 Partial documentation during the project

You build the documentation step by step, not only at the end. Below are checkpoints for every project:

Checkpoint	When	What to submit
C1 – Project plan	At the end of the planning session, before implementation (after Task 6)	Section 1 and a draft of 1-5 tasks. Plan, data and methods, experiments, work split in the team.
C2 – Progress record	At the end of each laboratory.	The current state of the code and documentation (e.g. repository commit, or updated notebook or report) with a short note on what was done and by whom
C3 – Final documentation	Before the defense, by the deadline given by the teacher.	The complete documentation with all seven sections, the code, and the GenAI note.

4. Working on a project

Section 3 describes **what the documentation must contain**. This section **describes what you do** during a project and when. The 12 tasks below are one full pass through a single project. You repeat the cycle 3 times, once for each chosen project theme. You work together as a team (**2-4 students**), but every team member must be able to explain every part of the project.

Stage	Tasks	Result
Planning session	Tasks 1-6	Approved topic and work plan (checkpoint C1)
Laboratory session 1	Tasks 7-8 (start of Task 9)	Data or simulator ready, inputs prepared (C2)
Laboratory session 2	Tasks 9-10	Models and baseline working, experiments running (C2)
Laboratory session 3	Tasks 10-11	Experiments completed, results analyzed and discussed (C2)
Defense	Task 12	Final documentation (C3) and defense.

Planning

TASK 1.

Select a project topic. Choose one of the proposed topics or, preferably, propose your own topic within one of the thematic groups.

The topic must be accepted by the teacher before further work.

TASK 2.

Define the aim and scope of the project. State the main engineering objective, define the problem to be solved, and specify what will constitute a successful solution.

TASK 3.

Define the planned approach. Identify the expected data source, methods, models or algorithms, software tools, and evaluation criteria. At this stage, these choices may still be preliminary.

TASK 4.

Define the main experiments and comparisons. Decide which approaches, configurations, operating conditions, data variants, or other factors will be compared, and what questions these comparisons are intended to answer.

TASK 5.

Define the order of work. Specify the main stages of the project and their logical dependencies.

TASK 6.

Prepare the team work plan. Divide the planned tasks between team members and define the expected outputs of individual tasks.

The plan should be discussed with and accepted by the teacher before the main implementation begins.

Implementation and experiments

TASK 7.

Prepare the data or system representation. Acquire, generate, or use the provided data and perform the necessary exploration, cleaning, preprocessing, splitting, or simulation setup.

TASK 8.

Prepare the model inputs or features. Select and prepare the information used by the models or algorithms. Perform feature extraction, engineering, selection, or other transformations where required.

TASK 9.

Build and configure the models or solution methods. Implement, train, fit, identify, or configure the selected approaches. Include a suitable reference or baseline method where applicable.

TASK 10.

Conduct the planned experiments and comparisons. Follow the agreed experimental setup and record the relevant results in a systematic and reproducible way.

TASK 11.

Analyze and present the results. Compare the investigated variants, prepare appropriate tables and visualizations, identify important successes and failures, and interpret the findings from an engineering perspective. **Discuss the results with the teacher.**

Conclusion and defense

TASK 12.

Formulate conclusions, limitations, and recommendations. Answer the original project question, identify the preferred solution, discuss its limitations and trade-offs, and state possible directions for further work. **Present and discuss the final conclusions with the teacher.**

Note: attach your GenAI usage proof if you use any. It is a part of owning your work.

5. Project themes and example topics

You choose **3 out of the 6 themes** below. For each theme a few example topics are listed. They are only a starting point – you are **encouraged to propose your own topic** within a theme, as long as it focuses on the main aspect of that theme. Every topic, including the examples, **must be accepted by the teacher**.

5.1 Evolutionary Optimization in Engineering Design - application of evolutionary algorithms to complex, multi-objective engineering problems with simulation-based or experimental evaluation.

Example topics:

- Multi-objective PID controller tuning with an evolutionary / genetic algorithm (overshoot vs settling time vs control effort).
- Optimal sensor placement under a budget of N sensors (art gallery problem, memetic algorithm).
- Energy-efficient trajectory of a robotic manipulator subject to time and joint limits.

5.2 Physics-Informed and Context-Enriched Modeling - integration of domain knowledge and physical laws into machine learning models to improve generalization and interpretability.

Example topics:

- Physics-informed neural network (PINN) for a known ODE/PDE (damped oscillator, heat/diffusion, Burgers' equation).
- Physics-informed identification of an unknown parameter (damping coefficient, spring constant) from noisy trajectories.
- Prediction of mechanical response combining measured data with the known dynamic equation as a loss term.

5.3 Data-Driven Process Modeling and Control - modeling dynamic systems using learning-based approaches (e.g. LSTMs, neural-MPC) for prediction, simulation, or control.

Example topics:

- Multi-step prediction of a dynamic system (tank, thermal process, trajectory)
- Neural surrogate inside a model predictive control loop (neural-MPC)

5.4 Explainable and Trustworthy AI in Engineering - exploration of techniques for improving transparency, robustness, and interpretability of AI models in safety-critical systems.

Example topics:

- Interpretable vs black-box models – do the explanations agree with physics?
- Robustness of a classifier to measurement noise – finding the breaking point.
- Trust outside the training data – defining an operating envelope.

5.5 Surrogate Modeling and System Identification - development of fast, approximate models to replace complex simulations or unknown systems using data-driven learning.

Example topics:

- FEM surrogate model – predicting maximum stress or deflection of a mechanical component.
- Discovery of governing equations from measured trajectories with SINDy.

5.6 Learning from Sparse, Noisy, or Imperfect Data - investigation of strategies such as data augmentation, transfer learning, and robustness techniques to handle real-world data limitations.

Example topics:

- Robustness under increasing measurement noise, with and without denoising.
- Fault classification under limited training data (learning curves, augmentation, transfer).
- Strongly imbalanced classification (1:50, 1:100) – and why accuracy lies.
- Transfer learning across operating conditions with few target samples.

6. Assessment

Grading uses **tokens**, counted separately for two things: **knowledge** (from lectures) and **skills** (from labs and projects). You must pass both. The tables below show the thresholds used in this course.

Area (what it tests)	Max	Pass	Grade 5.0
Lectures – knowledge	20*	10	18
Labs & projects – skills	40	15	25

How the 20 knowledge tokens add up:

5 tokens - case study submission & description (15 mins during the lecture, once)

5 tokens - Lecture discussion (Individual, at consultations or after the lecture, up to 2 times)

5 tokens - Lecture participation (1 token per lecture, up to 5)

15 tokens – Final merit test (note: you don't need it if you are active! If you need it though, it increases the baseline – that is: the percentage grade is calculated with respect to)

How the 40 skill tokens add up:

- **Lab activity** – 0-3 per lab, earned individually, laboratory activities (27 max).
- **Project grade** – 3 group grade, for organization, discussion and conclusions (9 max).
- **Tasks at home / absence** – correction of absence, tasks performed at home.

In the labs you earn **0-3 tokens individually** each time, using this scale:

Tokens	What it means
3	Exceptional – real initiative and understanding (hard to get!)
2	Good, well done
1	Acceptable
0	Absent, not cooperative, or not prepared

How it adds up in practice:

2 projects, maxed out: 18 (labs) + 6 (discussion) + 4 (at home additional analysis) = **28** → **5.0**

2 projects, typical: 12 (labs) + 4 (project) = **16** → **barely pass (3.0)**

1 project, maxed out: 9 (labs) + 3 (discussion) = **12** → **not enough to pass**

“Barely participating” in everything: ≈ 15 → **exactly 3.0**

Most of your skill tokens are earned **individually** - not as a group. Of the 40 skill tokens, up to 27 come from your own performance in each lab (0-3 every time), assessed per person. Only the 9 project tokens are a shared group grade.

The tokens reward understanding, not output. The reasoning and choices you can defend are what matters.