Department of Computer Engineering

CENG 383

Real-Time Systems

Timing, Verification, and Predictable
AI-Enabled Systems

Course Syllabus

Academic Year:	2026–2027
Course Level:	Undergraduate
Department:	Computer Engineering
Instructor:	Tolga Ayav

İzmir Institute of Technology

Course Description

This course develops the principles and methods needed to design computer systems whose correctness depends on both what they compute and when they deliver results. Students study real-time task models, scheduling and schedulability analysis, resource sharing, real-time operating systems, and the timing behavior of real computer platforms.

The course then introduces timed automata and model checking with UPPAAL, connecting timing requirements to formal models, verification properties, counterexamples, and model-based testing. The final part examines Edge AI and LLM-enabled cyber-physical systems under timing and resource constraints, with an emphasis on runtime monitoring, validation, and fallback mechanisms. Experimental timing analysis and an integrated project connect analytical reasoning, formal verification, and implementation evidence.

Course Philosophy

Throughout the semester, students repeatedly address one central question:

> *How can we build systems that deliver the right result at the right time, and provide evidence for that claim?*

Predictability requires explicit assumptions about workloads, platforms, environments, and failure behavior. Students learn to distinguish a scheduling guarantee, a property verified on a model, and a result observed in an experiment, and to explain the scope and limitations of each.

Course Learning Outcomes

Upon successful completion of this course, students will be able to:

CLO1: Analyze real-time systems using task models, deadlines, response time, WCET, latency, jitter, and periodic/sporadic workloads.

CLO2: Apply scheduling algorithms and schedulability analysis with explicit assumptions and interpret the results.

CLO3: Explain and apply RTOS task, priority, preemption, interrupt, timer, synchronization, and resource-sharing concepts.

CLO4: Model real-time systems with timed automata and verify timing and safety properties through model checking, including UPPAAL.

CLO5: Design and evaluate Edge AI systems under timing and resource constraints, considering hardware/software trade-offs.

CLO6: Design and evaluate monitoring, validation, and fallback mechanisms for AI-enabled real-time systems against stated timing and safety requirements.

CLO7: Measure and evaluate timing variability, latency distributions, and deadline misses, and distinguish experimental evidence from worst-case guarantees.

Textbook

Author: Hermann Kopetz
Title: *Real-Time Systems: Design Principles for Distributed Embedded Applications*
Edition: Second edition
Publisher: Springer
Year: 2011
ISBN: 978-1-4419-8236-0
DOI: 10.1007/978-1-4419-8237-7

The textbook is complemented by lecture materials on scheduling, UPPAAL, experimental timing analysis, and AI-enabled systems.

Additional Resources

Lecture notes, worked examples, source code, modeling exercises, and project instructions will be shared through the course's designated online platform. Supplementary references include the official UPPAAL documentation, RTOS documentation, and selected readings on Edge AI and runtime assurance.

Learning Journey

The course is organized around four progressive stages:

1. **Specify and analyze timing:** translate requirements into task models and evaluate schedules and resource-sharing behavior.
2. **Implement and observe:** study operating-system mechanisms and measure timing on real computing platforms.
3. **Model and verify:** construct timed automata, check properties, and interpret diagnostic traces and model-derived tests.
4. **Integrate and justify:** combine AI workloads with timing budgets, runtime assurance, and evidence-based evaluation.

Lectures combine worked problems, diagrams, code walkthroughs, and modeling demonstrations.

Assessment and Grading

Student performance will be evaluated through one midterm examination, one final examination, and one project:

Assessment Component	Weight
Midterm Examination	30%
Final Examination	40%
Project	30%
Total	**100%**

$$G_{\text{course}} = 0.30\,G_{\text{midterm}} + 0.40\,G_{\text{final}} + 0.30\,G_{\text{project}}.$$

The midterm is in Week 9 (no lecture); project presentations are in Week 14. The final follows the university examination calendar. Assessment details and dates will be announced separately.

Weekly Schedule

Weeks 1–7: Foundations, implementation, and formal modeling

Week	Main Topic	Core Concepts	Weekly Learning Outcomes
1	Introduction to Real-Time Systems	Logical and temporal correctness; hard, firm, and soft deadlines; latency, response time, jitter, WCET; workload classes.	Classify real-time requirements; identify deadlines and timing quantities in a sensor-to-actuator system.
2	Real-Time Task Models and Scheduling	Execution time, period, and deadline; static schedules and cyclic executives; cooperative and preemptive execution; RM, DM, and EDF.	Construct task models and schedule traces; distinguish offline scheduling from runtime scheduling and fixed from dynamic priorities.
3	Schedulability and Response-Time Analysis	Utilization tests; Liu–Layland bound; response-time analysis; blocking; harmonic and non-harmonic task sets.	Apply schedulability tests under stated assumptions; calculate response times and interpret the scope of the conclusions.
4	Resource Sharing and Real-Time Concurrency	Critical sections; mutexes and semaphores; priority inversion; priority inheritance and ceiling protocols; deadlock.	Explain synchronization-induced blocking using execution traces; assess its effect on predictability and timing analysis.
5	Real-Time Operating Systems	Tasks and states; priorities; context switches; timers; interrupts and ISRs; IPC; FreeRTOS/Zephyr examples; RTAI; nano-kernel code.	Relate RTOS abstractions to implementation mechanisms; explain interrupt handling, scheduling, and synchronization in kernel examples.
6	Timing in Real Computer Systems	Measured execution time versus WCET; caches, pipelines, memory, interrupts, OS interference, and multicore contention.	Design a reproducible timing experiment; interpret variability and deadline misses; state the limitations of measurement evidence.
7	Timed Automata for Real-Time Systems	Formal definitions and semantics; clocks, locations, guards, invariants, channels; automata networks; reachability and state explosion.	Translate timing requirements into timed automata; distinguish a location from a state; explain synchronization and state-space growth.

Weekly Schedule (continued)

Weeks 8–14: Verification, examination, and AI-enabled systems

Week	Main Topic	Core Concepts	Weekly Learning Outcomes
8	Model Checking of Real-Time Systems	UPPAAL workflow and query language; reachability, safety, liveness, leads-to, and deadlock; counterexamples; model-based testing.	Formulate and interpret properties; analyze diagnostic traces; explain how model-derived test sequences support implementation testing.
9	Midterm Examination — No Lecture	Examination week; no new lecture content.	Demonstrate understanding of the material covered before the midterm, according to the announced examination scope.
10	Edge AI Systems and Inference under Real-Time Constraints	Sensor–inference–decision–actuator pipelines; CPU/GPU/NPU; memory, energy, and compute budgets; tail latency; model optimization.	Specify end-to-end timing and resource budgets; compare inference configurations using measured evidence and hardware/software trade-offs.
11	Scheduling and Resource Management for Edge AI	Mixed AI/RT workloads; shared accelerators; pipelines; batching; overload; deadline-aware execution; adaptive models.	Evaluate scheduling and resource-allocation choices; connect observed timing behavior to workload assumptions and deadline requirements.
12	Safe and Predictable AI-Enabled Real-Time Systems	Monitors, acceptance testers, shields, watchdogs, fallback, and degradation; selected SPARC runtime-assurance and evidence ideas.	Design a monitor/fallback contract; evaluate late or invalid outputs; distinguish verified properties, experimental evidence, and certification claims.
13	LLMs in Real-Time and Cyber-Physical Systems	Local/edge LLMs; variable latency; structured outputs; validation; bounded retries; safe fallback; end-to-end integration.	Evaluate a decision loop with explicit timing assumptions, retry budgets, output validation, and evidence for failure handling.
14	Project Presentations and Discussion	Student presentations and demonstrations; design choices, analysis, models, measurements, limitations, and course synthesis.	Present and defend an integrated real-time system and its supporting evidence; discuss limitations and alternative designs.

Practical Work and Project Component

Practical activities include scheduling problems, RTOS and nano-kernel code walkthroughs, timing experiments, UPPAAL verification, and AI inference analysis.

The integrated project connects requirements, architecture, timing analysis, formal modeling, implementation, and experimental evaluation. Students must state assumptions and justify conclusions with evidence. AI-enabled designs should address late or invalid outputs through validation, monitoring, and fallback.

The project brief will specify the required deliverables, collaboration rules, assessment rubric, and submission dates. Preparation and development are intended to progress through the following stages:

- **Week 8:** project introduction following model checking.
- **Weeks 10–11:** requirements, architecture, timing/resource budgets, and initial scheduling and verification models.
- **Weeks 12–13:** runtime assurance, integration, timing experiments, and failure-handling evaluation.
- **Week 14:** presentation, demonstration, and discussion of results and limitations.

These stages guide project development; exact deadlines will be announced in the project brief. The project constitutes 30% of the final course grade. Practice exercises support learning and do not form a separate grading category.

Course Narrative

Students follow a timing requirement through task modeling, scheduling, operating-system implementation, and hardware timing analysis. Formal models then make assumptions and verification claims explicit. This reasoning extends to AI-enabled systems, where variable execution times and potentially invalid decisions require runtime assurance. The project connects design, analysis, verification, and measurement into an argument about an implemented system's behavior.

Academic Integrity

Students are expected to comply with the academic integrity regulations of İzmir Institute of Technology. All submitted work must represent the student's own effort unless collaboration is explicitly permitted.

Unauthorized collaboration, plagiarism, fabrication of experimental or project results, and the use of unapproved materials or tools during examinations are not permitted.

Attendance and Examination Policy

Attendance requirements, eligibility conditions for examinations, make-up examination procedures, and other administrative matters are governed by the applicable undergraduate education regulations of İzmir Institute of Technology.

Students are responsible for following all course announcements and scheduled assessment dates.

Generative Artificial Intelligence (GenAI) Policy

Generative Artificial Intelligence (GenAI) tools (e.g., ChatGPT, Claude, Gemini, Copilot, or similar systems) may be used as learning aids throughout this course. Students are encouraged to use these tools to explore concepts, brainstorm ideas, improve writing, or receive explanations of technical topics.

However, GenAI tools are intended to support learning—not replace it. Students remain fully responsible for the accuracy, originality, and technical correctness of every statement, calculation, figure, design decision, piece of code, and conclusion submitted as their own work.

Students should therefore adhere to the following principles:

- Every submitted solution must reflect the student's own understanding.
- Students must be able to explain and justify every part of any submitted work, regardless of whether GenAI was used during its preparation.
- The use of GenAI does not transfer responsibility for errors, misconceptions, or academic misconduct.
- Course staff may ask students to explain, reproduce, or extend any submitted work during examinations, laboratory sessions, or oral discussions.

The goal of this policy is not to discourage the use of modern AI tools, but to promote responsible engineering practice and independent technical judgment.

> *An engineer is ultimately accountable for the systems they design. Likewise, students are accountable for every word, equation, diagram, design decision, and line of code they submit, regardless of the tools used during their preparation.*