OST & SM project topics

September 2, 2025

Subjects

- Open source technologies and real-time data analytics
- Stream mining

General exam rules

- General exam rules are defined in the course intro presentations.
- Note: there will be 2+1 opportunities to attempt the theoretical part of the exam:
    - Regular exam opportunity in December and/or mid-January.
    - Re-take opportunity near the end of January. Available only to candidates who failed the regular exam attempt in December or January.

General project rules

OST+SM joint projects plan and milestones:

- Project topics are chosen and solved by 5-member teams.
    - Team formation deadline: **Sep 22, 2025**.
    - Project selection deadline: **Sep 26, 2025**. Max points deduction if this deadline is missed (-10%).
- System architecture and per team member workplan - Deadline: **Oct 17, 2025**. Sooner the better. Revise with project supervisor. Significant point deductions if this deadline is missed (-15%).
- Early project submission deadline will be **Nov 27, 2025 (12:00 CET)**. We will organize the theoretical part of both exams (SM & OST) in December for candidates submitting their projects by this deadline on the first submitted, first served basis. Submissions only via Canvas assignments.
- Early project defenses in the Dec 1-5 week, 2024.
- Late project submission deadline will be **Dec 4, 2025 (12:00 CET)**.

Minimum project implementations include at least the following elements for a passing grade (and meaningfully more for a higher grade):

- 2 different input datasets utilized - if feasible.
- 1 streaming system included in the system architecture.
- 2 stream processing stages developed by each team member doing the project in the SM course. Identify interesting algorithms in relevant scientific papers and implement and integrate such algorithms for a higher grade!
- 1 data storage tech where the input data stream or its aggregate is persisted.
- 1 batch processing stage trained and utilized on 'static' data in the data store.

- 1 visualization solution. Each team member creates at least one (meaningful) dashboard with at least 2 graphs included.
- Reference research papers (conference or journal) and implement the solutions or their derivatives for higher grades (4 and 5) in the SM course.

Project descriptions are provided by the course teams and include at least the following

- Title and short description.
- List of datasets to be used.
- Data analysis and flow stages diagram - might include the names of technologies used to implement each analytics.
    - Batch analytics.
    - Stream mining transformations.
- Deployment diagram showing the actual deployment of each utilised technology and data source on Docker containers or similar.
- List of open-source tools and technologies which must be included in the project.
- Useful references to papers and online resources if any.

Project submission

The project submission will consist of the following elements:

- Github repository for the code.
- Github (or other) repo for the data (unless otherwise instructed).
- A short-ish project report describing the system. The report should be integrated into the Github repository - it must contain the same elements in that case as well (see details below). Alternatively, and on request the report can be edited in Overleaf (LNCS template format).
- A **max** 10-slide PPT focusing on the system architecture and key contributions. This will be presented during the project presentations in the last two weeks of the semester. The structure closely follows the report structure. Additionally:
    - The PPT includes a well-defined description of duties performed by project team members, i.e. the team members tell us who did which part of the project. This will allow the course team to separately grade the team members, i.e. not all team members get the same grades in every case.

Project Documentation

LNCS template on Overleaf is available here:
https://www.overleaf.com/latex/templates/springer-lecture-notes-in-computer-science/kzwwpvhwnvfj

- It should contain the following elements/sections:
    - Structured abstract (up to 200 words). Background, goal, method, results, impact.
    - Introduction (up to 1 pp). Includes a references to relevant scientific papers, standards and technologies.
    - Data sets utilised (up to 1 pp).

    - System architecture (2 pp).
    - Testing/experiments (2 pp).
    - References. At least 10 entries (but max 15).
- The documentation will be written in English and checked for grammar errors by the team members (or third parties helping the team members). Very bad English will not be accepted, so-so bad English will affect the points received.
- The references will be listed in the APA or Harvard format. There will be at least 10 relevant references, but not more than 15 and which are not equal to the course materials and/or books. Find similar conference/journal papers and/or project reports and refer to them.
- The age of the references should not be older than 10 years. 2-3 exceptions are accepted if they are highly relevant and therefore necessary.
- The documentation will be shared with the course professors on Overleaf or Github and not in PDF or other non-shared document format.

Topic types and lists (2025)

Topic 0: Cybersecurity job market analyzer (CSOMA)

- **Description:** Develop a solution for collecting, storing and analysing cyber security job adverts on the European scale. Analyze and visualize the data.
- **Team size:** 5
- **Mentored by:** Imre Lendak
- **Dataset:** From online feed.
- **Open-source technologies to be used:** Kafka for real-time data ingest and stream mining, LinkedIn as a data source, sktime (or similar) for time-series analytics, Kubernetes for deployment.
- Batch processing task(s):
    - Batch 1: Train a (prediction) model for weekly/monthly per-country analytics.
    - Batch 2: Train a forecasting model for predicting per-country and European trends.
- **Example stream mining pipelines:**
    - Pipeline 1: Filter ads per country and maintain a Top 10 target list.
    - Pipeline 2: Filter ad countries and maintain a Top 10 country/region list.
    - Pipeline 3: Detect changes in job types per country/region and report.
    - Other ideas for higher grades: analyze Cyberseek and generate analyses based on their features.
- **Visualization:** Create a choropleth map of job ad frequency across different job types. Allow scroll-back through time. Create additional dashboards.

Topic 1: New-tech SCADA (NT-SCADA)

- **Description:** Develop a supervisory control and data acquisition platform based on open-source technologies. Implement ingesting, storing, analyzing and visualizing large volumes of analog and digital inputs collected from sensors. Add support for handling analog and digital outputs sent from the SCADA to actuators (e.g., breakers, valves and other equipment which turns electric signals into physical actions).
- **Team size:** 5
- **Mentored by:** Imre Lendak
- **Dataset:** SWaT and other datasets containing sensor and actuator data.
- **Open-source technologies to be used:** Kafka for real-time data ingest, InfluxDB (or similar) for storage, Flink for batch data analytics, Kafka for stream mining (or other on request), sktime (or similar) for batch time-series analytics, Kubernetes for deployment.
- Batch processing task(s):
    - Batch 1: Train at least one binary classification model.
    - Batch 2: Train at least one fine-grained classification model.
    - Batch 3: Analyse and visualise daily statistics.
- **Stream mining pipelines:**
    - Pipeline 1: Identify anomalous sensor data in real time utilising a binary classification model.
    - Pipeline 2: Perform fine-grained classification of sensor data.
    - Other ideas for higher grades: look for relevant network flow analysis solutions online and implement.
- **Visualization:**
    - Tabular visualization of sensor data.
    - Tabular visualization of actuator data.
    - Analog input and output plots.
    - Other dashboards and plots.

Topic 2: Open-source operational technology SIEM (OSOT-SIEM)

- **Description:** Develop a security information and event management system tailored for use in critical infrastructure settings. The OSOT-SIEM should collect at least intrusion detection system (IDS) logs, anti-malware logs, sensor/actuator values and network session information (from ip, to ip, from port, to port, kBs).
- **Team size:** 5
- **Mentored by:** Imre Lendak
- **Dataset:** Separate datasets containing IDS, anti-malware, sensor/actuator and network session data. Example datasets: SWAT 2015 and newer.
- **Open-source technologies to be used:** Kafka for real-time data ingest, InfluxDB (or similar) for storage, Kafka for stream mining (or other on request), sktime (or similar) for batch time-series analytics, Kubernetes for deployment.

- Batch processing task(s):
    - Batch 1: Train at least one time-series anomaly detection model.
    - Batch 2: Train at least one fine-grained classification model.
    - Batch 3: Analyse and visualise daily statistics.
- **Stream mining pipelines:**
    - Pipeline 1: Identify anomalous sensor data in real time utilising a binary classification model.
    - Pipeline 2: Perform fine-grained classification of IDS data.
    - Other ideas for higher grades: look for relevant network session analysis solutions online and implement, add at least one threat intelligence data source to the project.
- **Visualization:**
    - Tabular visualization of sensor/actuator data.
    - Event correlation maps.

Topic 3: Federated Deep Learning for IoT-Based Anomaly Detection in Real-Time Monitoring (FLEAD)

Description:

Develop a distributed framework for real-time anomaly detection in IoT-based monitoring systems using Federated Learning. The project will simulate multiple IoT devices (e.g., sensors in a smart city or industrial setting) that collaboratively train a deep learning model for detecting anomalies without sharing raw data. The framework focuses on privacy preservation and can handle heterogeneous IoT devices with varying computation power, data quality, and network constraints. A simulated IoT dataset will be used to replicate different devices' behaviors and data streams, making the project realistic while maintaining the benefits of distributed learning.

Team Size: 5

Mentored by: Jiyan Mahmud (jiyan@inf.elte.hu)

Open-Source Technologies to be Used:

- **Federated Learning Frameworks:** TensorFlow Federated
- **Kafka** for real-time data ingestion
- **Spark** for batch analytics and stream mining
- **Stream processing frameworks:** Kafka Streams or Apache Flink
- **Visualization Tools:** Grafana, Kibana, or Plotly Dash
- **Deployment:** Docker and Kubernetes for containerization and orchestration

Federated Stream Mining Pipelines:

- **Pipeline 1:** Receive and ingest streaming IoT sensor data in real time, enabling federated learning across edge devices.
- **Pipeline 2:** Apply preprocessing on each local IoT device (clean, normalize, and prepare the data). Handle missing values, scaling, and encoding categorical variables, while preserving the privacy of each device's data.
- **Pipeline 3:** Train the local deep learning anomaly detection models on each device using federated learning techniques and send model updates (not raw data) to a central server for aggregation.
- **Pipeline 4:** Aggregate the model updates on a central server, using Federated Averaging or another federated aggregation technique, to improve the global model.
- **Pipeline 5:** Use ensemble methods for anomaly detection by combining results from different federated models, apply voting-based techniques for consensus, and trigger alarms when anomalies are detected.

Topic 4:X-ICS-IncreADStream: Explainable Incremental Autoencoder-Based Real-Time Anomaly Detection for Industrial Control Systems

Description:

This project will build a system to find and explain unusual behavior (anomalies) in real-time data from Industrial Control Systems (ICS). These systems are used in places like water treatment plants, power stations, or factories. The system will use a deep learning model (an autoencoder) to learn how the system normally works and detect when something strange happens.

The system will also learn continuously over time (called incremental learning). This helps it adapt to changes in the system, such as slow wear and tear or different working conditions. When it finds an anomaly, the system will explain which sensors caused it, so engineers can fix the problem faster.

The system will work with any streaming data tool (like Kafka,or others). It will also include a live dashboard to show data and send alerts when needed.

Team Size: 5

Mentored by: Jiyan Mahmud (jiyan@inf.elte.hu)

Open-Source Technologies to be Used:

- **Data Simulation & Ingestion**: Use flexible streaming tools to simulate or ingest real-time ICS sensor data from datasets such as SWaT, WADI, or HAI.
- **Preprocessing & Feature Engineering**: Apply Python-based data processing libraries such as Pandas and NumPy for real-time data cleaning, normalization.
- **Anomaly Detection Model**: Use deep autoencoders for unsupervised anomaly detection. The model will learn normal system behavior and identify deviations through reconstruction error. Incorporate incremental learning techniques to update the model over time as system conditions evolve.
- **Stream Processing:** Integrate with any stream processing framework such as Kafka Streams, Apache Flink, or lightweight socket-based pipelines to manage data flow, windowing, and real-time model inference.
- **Explainability Module**: Use tools like SHAP, Integrated Gradients, or LIME to generate real-time explanations for each detected anomaly, identifying the most influential features or sensors contributing to the deviation.
- **Visualization & Alerts**: Implement real-time dashboards using Grafana, Streamlit, or other visualization tools to display sensor trends, anomaly scores, and explanations.
- **Deployment & Containerization**: Use Docker/Kubernetes to containerize the system components for easy deployment on local machines, cloud infrastructure, or edge devices. Support modular deployment to simulate distributed ICS environments.

Real-Time Processing Pipelines:

Pipeline 1: Simulating ICS Data

Simulate real-time sensor data from industrial control systems using a pre-recorded dataset (e.g., SWaT, WADI, or HAI). Stream this data continuously through a chosen streaming platform (e.g., Kafka, or custom solution), replicating the behavior of a live SCADA environment.

Pipeline 2: Data Preprocessing & Feature Engineering

Process the incoming data stream in real time by handling missing or noisy values,Normalize sensor readings,Extract time-series features.

Pipeline 3: Deep Autoencoder-Based Anomaly Detection

Train a deep autoencoder on historical “normal” ICS data to learn the typical behavior of the system. The model evaluates incoming data by comparing it to its learned patterns and calculating reconstruction error.

Pipeline 4: Incremental Learning & Real-Time Detection

Deploy the trained model into the streaming environment to detect anomalies on the fly. Use **incremental learning** to update the model continuously with newly observed “normal” behavior,

Pipeline 5: Explainability Engine

When an anomaly is detected, trigger an **explanation module** to identify which sensor features most contributed to the deviation.

Pipeline 6: Visualization & Alerting

Visualize real-time system metrics, anomaly scores, and explanations using tools like Grafana or custom dashboards. Set up automatic alerts via Slack, email, or SMS when critical anomalies are detected, ensuring quick response from operators.

Topic 5: Smart City Air Quality Monitoring with Real-Time Stream Analytics (SCAir-IoT)

Description:
This project will develop a system to collect, store, and analyze air quality sensor data from smart city IoT devices. The system will simulate multiple sensors measuring pollutants (PM2.5, CO_2, NO_2, O_3) and detect pollution spikes in real time. It will also forecast short-term air quality using batch analytics to support city planning and public health responses.

Team Size: 5

Mentored by: Loubna Seddiki (seddikiloubna@inf.elte.hu)

Dataset:

- UCI Air Quality Dataset
- Optionally enriched with open city datasets

Open-Source Technologies to be Used:

- **Data Simulation & Ingestion:** Use Kafka or MQTT (or both) to simulate and ingest sensor data streams from UCI Air Quality Dataset and optional city datasets.
- **Preprocessing & Feature Engineering:** Apply Python libraries such as Pandas and NumPy for real-time data cleaning, normalization, and aggregation by location.
- **Analytics Models:** Use scikit-learn and TensorFlow to train regression models for short-term forecasting and classification models to predict safe vs. unsafe air quality zones.
- **Stream Processing:** Integrate Spark Streaming or Apache Flink to manage data flow, windowing, and online detection of pollution spikes.
- **Visualization & Alerts:** Build dashboards using Grafana or Plotly Dash to display live sensor readings, forecasts, and heatmaps. Trigger alerts for unsafe air quality events via Slack, email, or SMS.
- **Storage:** Use InfluxDB to store historical sensor data and enable efficient queries for visualization and analysis.
- **Deployment & Containerization:** Use Docker/Kubernetes to containerize system components for reproducible deployment on local machines, cloud, or edge environments.

Batch Processing Tasks:

- Train a regression model to forecast pollution levels 1–3 hours ahead.
- Train a classification model to predict “safe” vs. “unsafe” air quality zones.

Real-Time Processing Pipelines:

- **Pipeline 1: Simulating Air Quality Data.** Stream air quality data continuously from UCI Air Quality Dataset (and optionally enriched datasets like weather/traffic) using Kafka or MQTT to emulate real-world citywide IoT sensors.
- **Pipeline 2: Data Preprocessing & Feature Engineering.** Clean and normalize sensor readings, aggregate values by location, and extract time-series features such as averages and rolling statistics in real time.
- **Pipeline 3: Pollution Spike Detection.** Apply anomaly detection and threshold-based methods to identify abnormal increases in pollutants (PM2.5, CO_2, NO_2, O_3) in the streaming data.
- **Pipeline 4: Forecasting & Classification.** Run regression models for short-term forecasts (1–3 hours ahead) and classification models to label current city zones as safe or unsafe.

Pipeline 5: Visualization & Alerting

Provide dashboards with live readings, forecast vs. actual comparison curves, and a city heatmap of pollution zones. Trigger alerts when pollutant levels exceed safe thresholds.

Topic 6: Real-Time Anomaly Detection in IoMT Device Communication (IoMT-AD)

Description:
This project will build a system to analyze and detect abnormal behavior on the Internet of Medical Things (IoMT) devices such as patient monitors, infusion pumps, and wearables. The focus is on identifying unusual communication patterns that may indicate device malfunctions or cyberattacks. By analyzing real-time traffic streams, the system will flag anomalies, trigger alerts, and provide visualization dashboards to help healthcare operators respond quickly.

Team Size: 5
Mentored by: Loubna Seddiki (seddikiloubna@inf.elte.hu)

Dataset:

- CICIoMT2023 Dataset (IoMT network traffic with both normal and attack scenarios)

Open-Source Technologies to be Used:

- **Data Simulation & Ingestion:** Use Kafka to simulate and ingest IoMT traffic from CICIoMT2023 in real time.
- **Preprocessing & Feature Engineering:** Extract communication features such as packet size, protocol type, connection duration, and flow statistics using Python (Pandas, Scapy).
- **Anomaly Detection Model:** Train LSTM autoencoders for unsupervised anomaly detection and use TensorFlow/Keras for model training and inference.
- **Stream Processing:** Use Apache Spark Streaming or Flink to process traffic flows, apply feature extraction, and run real-time inference.
- **Visualization & Alerts:** Implement dashboards with Grafana or Kibana to monitor device traffic, anomaly scores, and attack patterns. Integrate alerting mechanisms (email, Slack, SMS).
- **Storage:** Use InfluxDB to store IoMT traffic metrics, anomalies, and alerts for later analysis.
- **Deployment & Containerization:** Package system components with Docker and orchestrate them with Kubernetes for scalability in healthcare network environments.

Batch Processing Tasks:

- Train an LSTM autoencoder model on historical normal IoMT traffic to learn baseline device communication behavior.
- Train classification models (e.g., Random Forest, CNN) to distinguish between normal and malicious traffic for evaluation.
- Tune thresholds for anomaly detection based on reconstruction error distributions.

Real-Time Processing Pipelines:

- **Pipeline 1: Simulating IoMT Traffic.** Replay CICIoMT2023 traffic streams using Kafka to emulate real-world IoMT communication.
- **Pipeline 2: Data Preprocessing & Feature Extraction.** Extract relevant network features (packet size, duration, flow count, protocols) and normalize them in real time.
- **Pipeline 3: Anomaly Detection Model.** Apply the LSTM autoencoder to incoming traffic and compute reconstruction error to flag unusual device communication.
- **Pipeline 4: Alerting & Forensic Storage.** Generate alerts for abnormal behavior, store flagged sessions in InfluxDB, and mark them for forensic investigation.
- **Pipeline 5: Visualization & Monitoring.** Use Grafana/Kibana dashboards to show device health, anomaly frequency over time, and heatmaps of attack patterns per device type.