# Edge-to-cloud Industry 5.0 Data Pipelines - M12

*D4.2*

| | |
|---|---|
| **Person responsible / Author:** | Nenad Stojanovic NISSATECH |
| **Deliverable N.:** | D4.2 |
| **Work Package N.:** | 4 |
| **Date:** | 31 December 2023 |
| **Project N.:** | 101092069 |
| **Classification:** | Public |
| **File name:** | D4.2 Edge-to-cloud Industry 5.0 Data Pipelines M12 v.0.7 |
| **Number of pages:** | 40 |



Funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Health and Digital Executive Agency (HaDEA). Neither the European Union nor HaDEA can be held responsible for them.

Status of deliverable

| Action | By | Date (dd.mm.yyyy) |
|---|---|---|
| **Submitted (author(s))** | Nenad Stojanovic (NISSATECH) | 29/03/2024 |
| **Responsible (WP Leader)** | Nenad Stojanovic (NISSATECH) | 28/03/2024 |
| **Approved by Peer reviewer** | Jorge Martinez-Gil (SCCH) | 28/03/2024 |

Revision History

| Date (dd.mm.yyyy) | Revision version | Author | Comments |
|---|---|---|---|
| 23/01/2024 | v. TOC | Nenad Stojanovic (NISSATECH) | TOC |
| 06/02/2024 | v. 0.1 | Nenad Stojanovic (NISSATECH) | Initial content |
| 20/02/2024 | v. 0.2 | Nenad Stojanovic (NISSATECH) | Platform details |
| 05/03/2024 | v. 0.3 | Stefan Marinkovic (NISSATECH) | Data Observation Validation |
| 18/03/2024 | v. 0.4 | Nenad Stojanovic (NISSATECH) | Refinement |
| 28/03/2024 | v. 0.5 | Jorge Martinez-Gil (SCCH) | Review |
| 29/03/2024 | v. 0.6 | Nenad Stojanovic (NISSATECH) | Refinement |
| 29/03/2024 | v. 0.7 | Gabriella Monteleone (POLIMI) | Final coordinator review before submission |

Author(s) contact information



| Name | Organisation | E-mail | Tel |
|---|---|---|---|
| Nenad Stojanovic | NISSATECH | Nenad.Stojanovic@nissatech.com | |
| Stefan Marinkovic | NISSATECH | Stefan.Marinkovic@nissatech.com | |
| Gabriella Monteleone | POLIMI | gabriella.monteleone@polimi.it | |

## Table of Contents

## Figures

## 1. Executive summary

The **goal of the AI REDGIO 5.0** project is to **move artificial intelligence (AI) and machine learning (ML) to the "edge"** which means closer to the source of manufacturing information. This "local cloud" depicts the possibility to have an efficient computing infrastructure locally, which can be of a considerable importance for manufacturing small and medium-sized enterprises (SMEs). AI REDGIO 5.0 will lower the complexity and cost barriers for manufacturing enterprises (notably SMEs) to develop, deploy and fully leverage the cloud/edge computing paradigm for the implementation of applications like predictive maintenance, quality management, Zero Defect Manufacturing, Lifecycle Assessment, Intelligent Asset Management, Human Robot Collaboration and Digital Twins, through providing:

- **free open source**, and **user-friendly tools** for developing end-to-end **pipeline for edge ML/AI applications**;
- **cloud/edge AI deployment patterns and blueprints** for different manufacturing applications.

During the project implementation **7 SMEs** are monitored and supported in their experiment's deployment and **14 Didactic Factories** around Europe are involved for the implementation of didactic experiments addressed to SMEs.

The present deliverable is the result of the Task 4.2 "Industry 5.0 Data Pipelines and Data Quality Assurance", whose main goal is creating **data pipelines and ensuring data quality** for the planned AI tasks.

This deliverable reports on the **software development of the Edge-to-cloud Industry 5.0 Data Platform**.

It addresses the **requirements** for the Edge-to-cloud Industry 5.0 Data Platform and explains the **architecture** and its details.

In addition, this deliverable provides details about the **Data Observation layer** which is responsible for ensuring the quality of the data collection process.

# 2. Introduction

## 2.1. Scope of the deliverable

The present deliverable is the result of the Task 4.2 “Industry 5.0 Data Pipelines and Data Quality Assurance”, whose main goal is creating **data pipelines and ensuring data quality** for the planned AI tasks.

This deliverable is the first release of the Platform. The second release is planned for M30.

## 2.2. Contributions to other WPs and deliverables

This deliverable will contribute to the next iteration of the Reference Architecture.

The **Edge-to-cloud Industry 5.0 Data Platform** will be validated in the context of WP6 during the implementation of the use cases.

## 2.3. Structure of the Document

The rest of the deliverable is structured as follows:

- Section 3: Backgrounds related to Data Pipelines
  - provides information about Data Pipelines this work is based on
- Section 4: Contextualization through Reference Architecture
  - explains the position of this work in the context of Reference Architecture
- Section 5: Requirements
  - provides the analysis of the requirements
- Section 6: Architecture and components
  - describes technical details of the the Edge-to-Clould Industry Data Platform
- Section 7: Data Observation details
  - provides information about Data Observation layer

# 3. Data Quality Pipelines Backgrounds

In this section we provide essential information for understanding the concept of Data Quality Pipelines. It is related to the work of Nissatech in the predecessor project AI REGIO.

## 3.1. Pipeline Architecture for AI applications

Data Quality Pipelines ensure data quality for AI applications. The main goal is to provide an efficient and easy to use infrastructure for enabling manufacturing SMEs to prepare their own data for the usage in AI applications. It includes the adapters for connecting relevant data sources and recipes (workflows) for defining data preparation pipelines (or using the available pre-configured ones). The main objective is to include the domain expertise in the data preparation process, but in a convenient way for non-technical experts. In addition, to ensure the reliability of collected data, methods for checking the completeness and validity of data will be applied on the edge (avoiding two most important problems for the AI: missing and corrupted data).

In the following figure, a high-level view on the data processing for AI applications is presented.

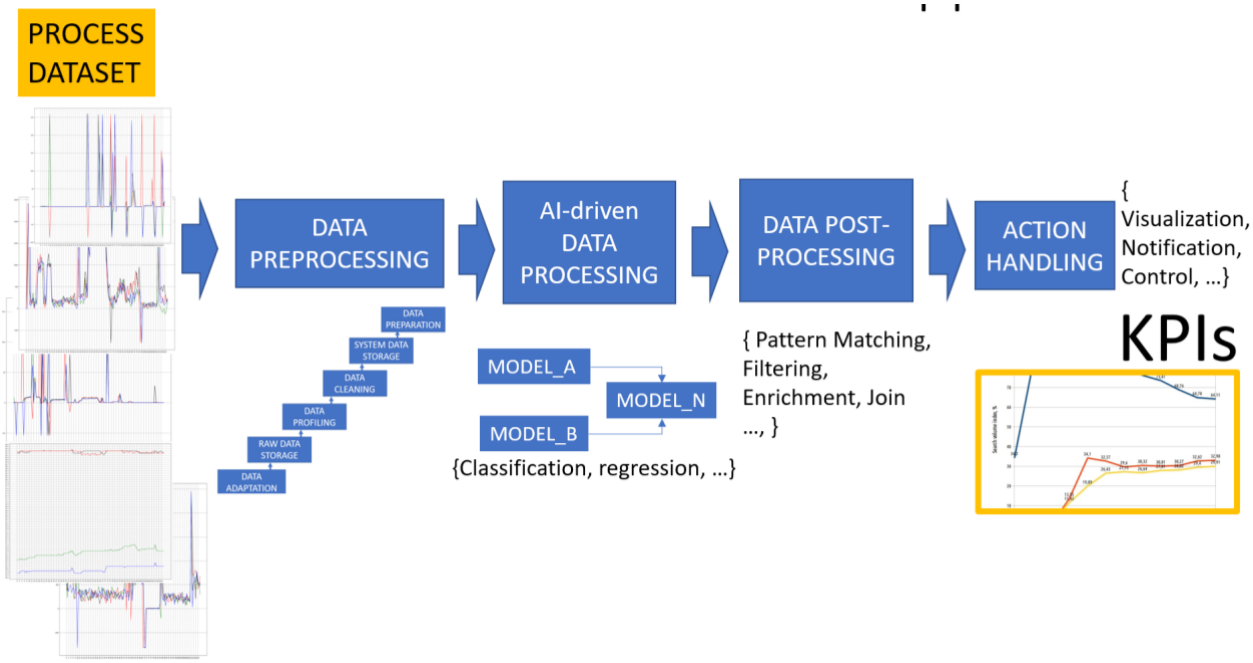



**Figure 1:** Data processing pipeline for AI applications

The steps of the pipeline are described briefly:

- Data Preprocessing is a processing pipeline which transforms raw data in well-formed data (valid structure) that can be processed by various data analysis methods
- AI-driven Data Processing is data analysis which can be done within or outside Data4AIPlatform
- Data Postprocessing enables preparation of the data for output (e.g., filtering)
- Action Handling is related to the delivery of the output to other (control, notification, visualization) systems

## 3.2. Data Preprocessing pipeline

The most important part of the architecture, presented in the previous subsection, for this deliverable is Data Preprocessing pipeline. As depicted in Figure 2, Data Preprocessing ensures the Data Quality from the syntax point of view (it is in the valid form and can be processed automatically).

In the following figure, the details of the Data Preprocessing pipeline are provided:

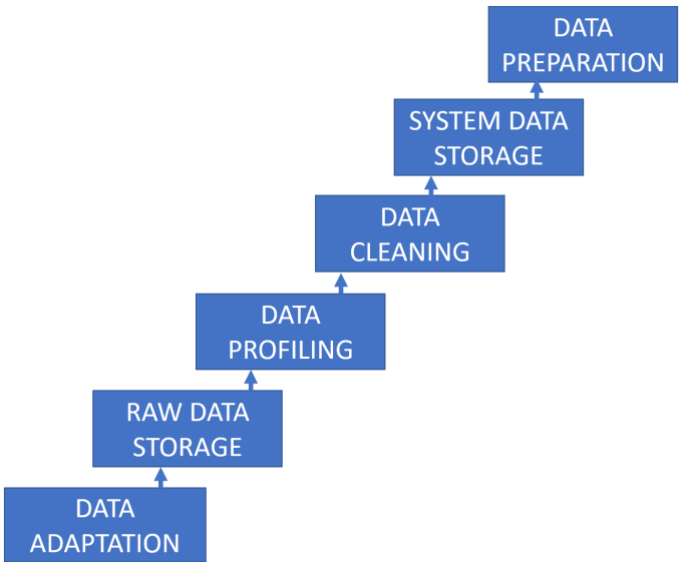



**Figure 2:** Data Preprocessing pipeline

The particular steps are described as following:

- Adapter – Reads the raw data from the relevant data sources (properly defined) and writes data into the raw data storage
- Raw data storage – Stores the raw data from the provided dataset into the previously defined format (d2twin, etc.)
- Profiling – Data Inspection (calculating profiling of the raw data stored in the raw data storage)
- Data cleaning – Data cleaning according to the info provided from data profiling (removing irrelevant data from the raw data)
- System data storage – Stores cleaned data after the profiling and cleaning are done
- Data preparation – Getting data from the system data storage and preparing the data for the analytics algorithms

## 4. Data pipelines in AI REDGIO 5.0 Reference Architecture

In this section we provide important information for understanding the positioning of the Data Quality Pipelines in the AI REDGIO 5.0 Reference Architecture (deliverable D4.1).

### 4.1. AI REDGIO 5.0 Reference Architecture overview

AI REDGIO 5.0 Reference architecture is presented in the following figure.

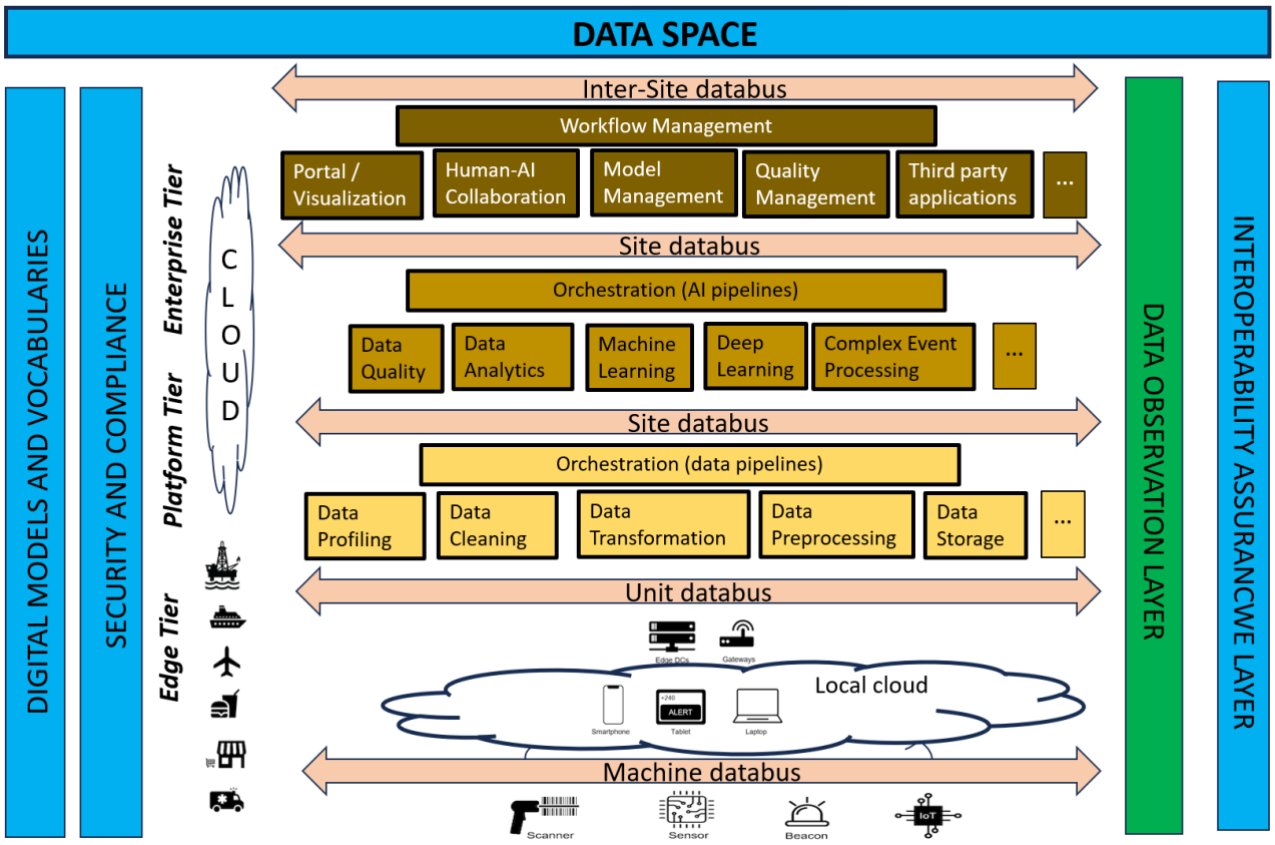



**Figure 3:** AI REDGIO 5.0 Reference Architecture

The most relevant part is the Edge, defined as following:

- **Edge**: ecosystem of heterogeneous devices (IoT (Internet of Things) Sensors, Machine Tools, IT (Information Technology) Systems, Robots, Cameras and so on) to gather machine data and make it available to the upper layers and IoTsystems.
- **Edge Tier**: computes some data management and analysis functions, in small datasets, using data, applications, and services contained in the edge.
    - **Data profiling**: the service used for making the information available within its environment and adapt its execution accordingly.
    - **Data cleaning**: the process able to guarantee the correctness of a huge amount of data, starting from data mining.

- **Data preprocessing**: in charge of data manipulation to prepare it in the format needed to be analyzed.
- **Data storage**: local storage, which enables advanced processing.
- **Orchestration**: implements the logic for defining complex data flow processes between devices, data services, applications, and people to produce desired outcomes. It is the link to the upper layer, allowing the data pipeline to communicate with the Platform services

Additionally, Data Observation Layer is very important for ensuring an efficient execution of the data pipelines, as described in the next section.

It is a set of services which process metadata related to the execution of any of the services in any of the layers.

## 4.2. Data Observation Layer

A data quality pipeline handles the collection, cleansing, transformation, and application of data to generate business insights.

Data-first companies have embraced data quality pipelines as an effective way to aggregate, operationalize, and democratize data at scale across the organization. As more organizations begin implementing data mesh architecture[1], and principles like data products, data teams also have to consider the data platform as the machine that develops, manages, surfaces, and governs data products.

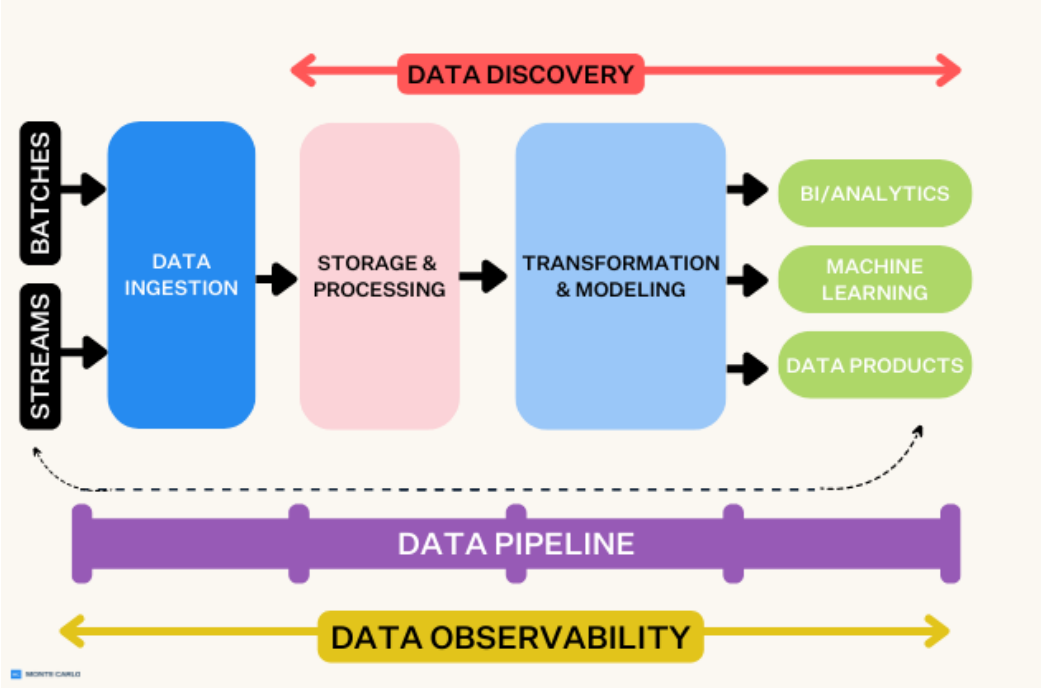



**Figure 4:** Data Observation for Data Pipelines

[1] https://www.montecarlodata.com/blog-what-is-a-data-mesh-and-how-not-to-mesh-it-up/

Data observability involves the complete monitoring, managing, and understanding of the modern data tech stack. These tools allow companies to better manage their data by helping them discover and solve real-time data issues and gain complete insight into the system's data health. Data observability tools help companies accelerate the adoption of data across departments. This helps in making strategic and data-driven decisions that benefit the entire organization.

The concept of data observability stems from best practices learned from DevOps software to manage impartial, inaccurate, or erroneous data. These best practices, which include optimizing logs, real-time insights, and so on, enable the creation of error-free and trusted data across the entire data stack, which includes data sources, data warehouses, ETL tools, ML/BI tools, etc.

Data observability tools are a part of DataOps platforms. DataOps platforms assemble several types of data management software into an individual, integrated environment. The platform unifies all the development and operations in data workflows. Data observability software focuses on monitoring the health of the data pipelines and the overall system.

Data observability tools differ from monitoring software since the latter focuses on pre-determined metrics to identify bugs, whereas data observability focuses on real-time detection and resolution. Data observability also differs from data quality software, wherein the former focuses on reducing the number of data incidents while accelerating resolution time. Data quality is the result of powerful data observability across the modern data stack.

Data observability layer must be able to monitor and alert for the following pillars of observability:

- **Freshness**: is the data recent? When was the last time it was generated? What upstream data is included/omitted?
- **Distribution**: is the data within accepted ranges? Is it properly formatted? Is it complete?
- **Volume**: has all the data arrived?
- **Schema**: what is the schema, and how has it changed? Who has made these changes and for what reasons?
- **Lineage**: for a given data asset, what are the upstream sources and downstream assets which are impacted by it? Who are the people generating this data, and who is relying on it for decision-making?

# 5. Data Quality pipeline Requirements

In this section requirements and specifications coming from the AI REDGIO 5.0 experiments are described.

## 5.1. Inputs from Experiments

**User requirement identifiers for Industry 5.0 Data4AI Platform & Data Spaces**

This subsection focuses on the user requirements of the 7 SME driven experiments and 14 Didactic Factories from the Industry 5.0 Data4AI Platform and Data Spaces (WP4). The users were prompted for input on what requirements were needed for their experiment to progress. The requirements were collected and are presented here in a generalised form to allow open solutions to be developed.

In the following table, we present the requirements from 7 SME driven experiments.

| Pilot | Organization name | Data4AI Platform & Data Quality |
|---|---|---|
| I | SCAMM | Missing values in input data cannot be filled by synthetic data, but must be addressed somehow |
| II | PERNOUD | Data preprocessing and cleaning activities are expected |
| III | GPALMEC | At this stage of the project, we do not plan to send data outside the machine. We might be interested in case of further development of the project. |
| IV | POLYCOM | To improve the robustness of the data processing pipelines, data validity needs to be verified before analytics to verify whether the ML algorithms are valid for the considered data set (detection of model extrapolation, detection of missing/defect data) |
| V | QUESCREM | Data preprocessing and cleaning activities are expected |
| VI | CAP | Industreweb.cloud platform will host Data Management Tools |
| VII | KATTYFASHION | QA (Quality Assurance) analysis results needs to be complex enough to be validated by a junior QA team member |

In the following table, we present the requirements from 14 Didactic Factories.

| DF | Organization name | Data4AI Platform & Data Quality |
|---|---|---|
| **I** | POLIMI - I4.0Lab | N/A |
| **II** | UNIBO - E2MECH | N/A |
| **III** | JSI - IJS Systems&Control Lab | Basic detection of data validity (basic detection of missing or faulty data) |
| **IV** | Brainpoint Industries-Flexible Manufacturing | N/A |
| **V** | UniTwente - AMC | N/A |
| **VI** | FBK - 4.0iLab | Data collected need to be processed (verify, correct, clean) at the Edge |
| **VII** | MAKE -PM50 | N/A - open to discussion |
| **VIII** | DMIW -Digital Manufacturing Innovation Hub | The AI interface will be built within the Industreweb platform using open source models |
| **IX** | MADE - BEhAI | Redundant data/potentially not useful data |
| **X** | TUIASI - I4.0 | In the end will be a percentage if there is a defect or not, according to a model (reference) |
| **XI** | CTU - RICAIP | Potential concept drift must be checked, recognized, and learned to prevent false positive detections. |
| **XII** | AAU - Smart Lab | The data will be collected, cleaned, and visualized for later processing. |
| **XIII** | PBN - amLab | The data will be collected, cleaned for later processing. |
| **XIV** | GRADIANT - Galicia Industrial Logistics Lab | Data preprocessing and cleaning activities are expected |

Both tables are derived from the tables presented in D4.1.

## 5.2. Requirements

In this section we present the requirements for the development of the Edge-to-cloud Industry 5.0 Data Platform.

The requirements are based on the list of user requirements presented in the previous section, as well as the requirements from our own experience in working with similar systems.

The following table contains a list of requirements for the Data Quality Pipelines and Data Observation.

| ID | Description | Priority | Stakeholders |
|---|---|---|---|
| **R1.1** | Enabling to collect the data | H | Process owner |
| **R1.2** | Enabling to upload a dataset | H | Process owner |
| **R1.3** | Enabling to define a set of criteria for data collection | H | Process owner |
| **R1.4** | Enabling to upload the information about past errors/anomalies | M | Process owner |
| **R1.5** | Calculation of the anomalous situations | H | Process owner |
| **R1.6** | Integration of results | H | Process owner |
| **R1.7** | Providing visual presentation of results | H | Process owner |
| | | | |
| **R2.1** | Enabling to select the data source | H | Data Quality engineer |
| **R2.2** | Enabling to define a set of criteria for data quality | M | Data Quality engineer |
| **R2.3** | Enabling to check / validate created pipeline | H | Data Quality engineer |
| **R2.4** | Enabling to deploy the pipelines in edge – cloud infrastructure | H | Data Quality engineer |
| **R2.5** | Provide automatic reporting about the data quality | H | Data Quality engineer |
| **R2.5** | Provide report about the data preprocessing results | H | Data Quality engineer |
| **R2.6** | Enable reconfiguration of the pipeline | M | Data Quality engineer |
| **R2.7** | Enable automatic validation of the pipeline (based on defined KPIs) | M | Data Quality engineer |
| | | | |

| ID | Description | Priority | Stakeholders |
|---|---|---|---|
| **R3.1** | Enabling the definition of the metadata for the data collection process | H | Data Quality manager |
| **R3.2** | Collecting the metadata from the data collection process | H | Data Quality manager |
| **R3.3** | Processing the metadata at the edge – cloud infrastructure | H | Data Quality manager |
| **R3.4** | Defining the alarm situations based on the metadata | H | Data Quality manger |
| **R3.5** | Creating the alarm reports | H | Data Quality manger |
| **R3.6** | Visualization of alarms | H | Data Quality manger |

# 6. Edge-to-cloud Industry 5.0 Data Platform

In this section we present the details related to the implementation of the Edge-to-cloud Industry 5.0 Data Platform, which is the main goal of task T4.3.

## 6.1. General Architecture

In our data collecting system, we have three major parts:

- Edge device part - the only part that is in the factory/edge itself and is responsible for collecting and preparing data that is stored on the server
- Server/Cloud part - responsible for data storage and authentication
- The data observability part - is also located on the server, but it is separated due to the very importance of this part. The role is monitoring of the data collecting system and alarming in precisely defined cases (cases that we assume are not normal behaviors in the system).

The general architecture of the entire system is presented in the following figure.

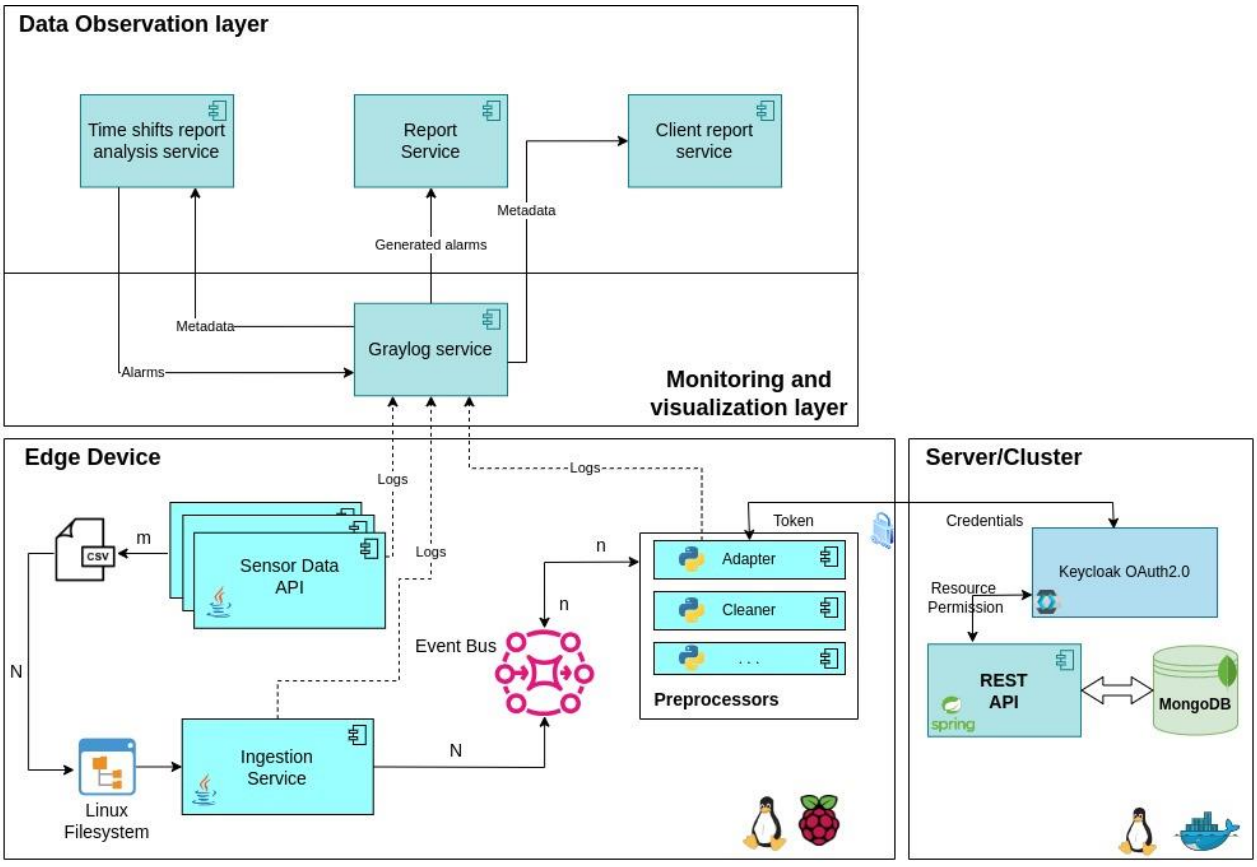



**Figure 5:** The architecture of the Edge-to-cloud Industry 5.0 Data Platform

## 6.2. Edge part

### 6.2.1. Sensor Data API

Sensor data API represents services that are a medium between the sensors themselves and the data collecting system. In the Sensor data API services, there are initial data from the sensors that are further forwarded to the data collecting system.

In our data collecting system we have three types of sensors:

- Environment sensors
- Vibration sensor
- Power (Energy) sensor

The following sequential diagrams show the types of sensor data API services and how they are included in the system. It is important to emphasize that the sensor data API services (Power, Environment and Vibration) work in parallel.

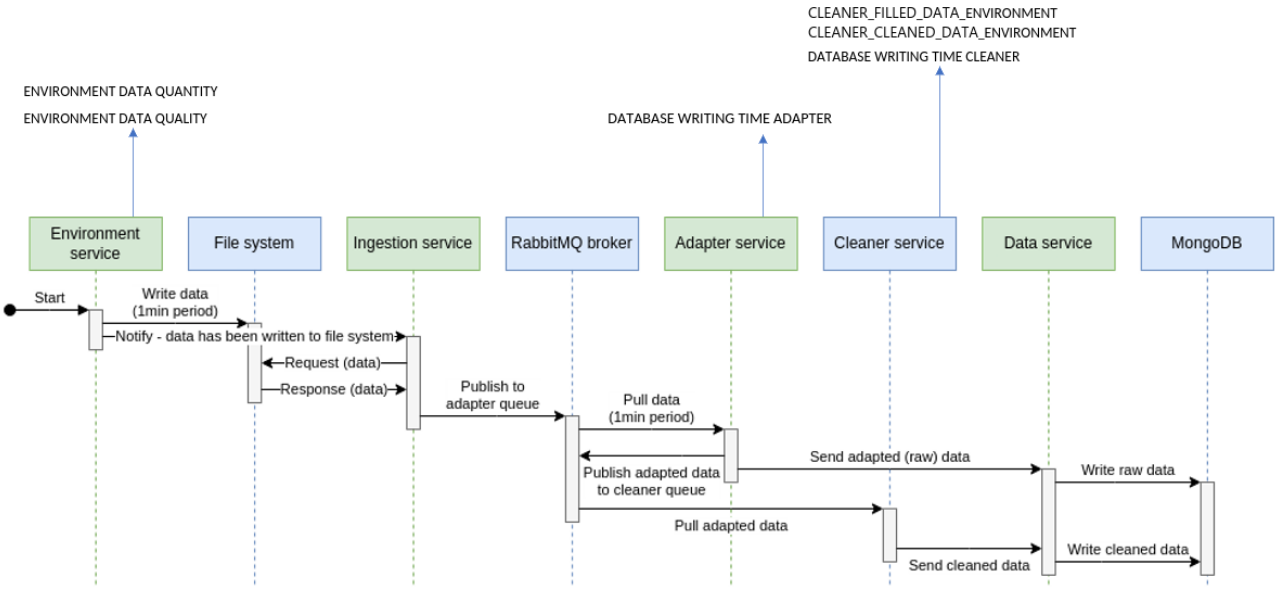



**Figure 6***:* Sequential diagram for Environment service (environment sensor data API)

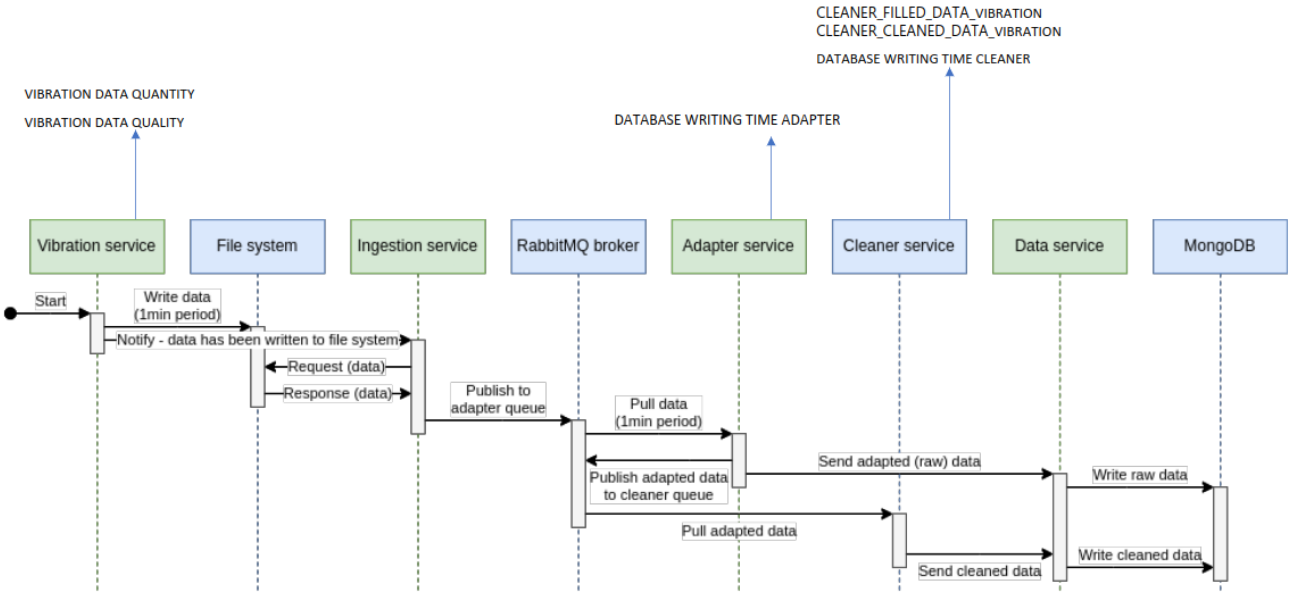



**Figure 7***:* Sequential diagram for Vibration sensor data API

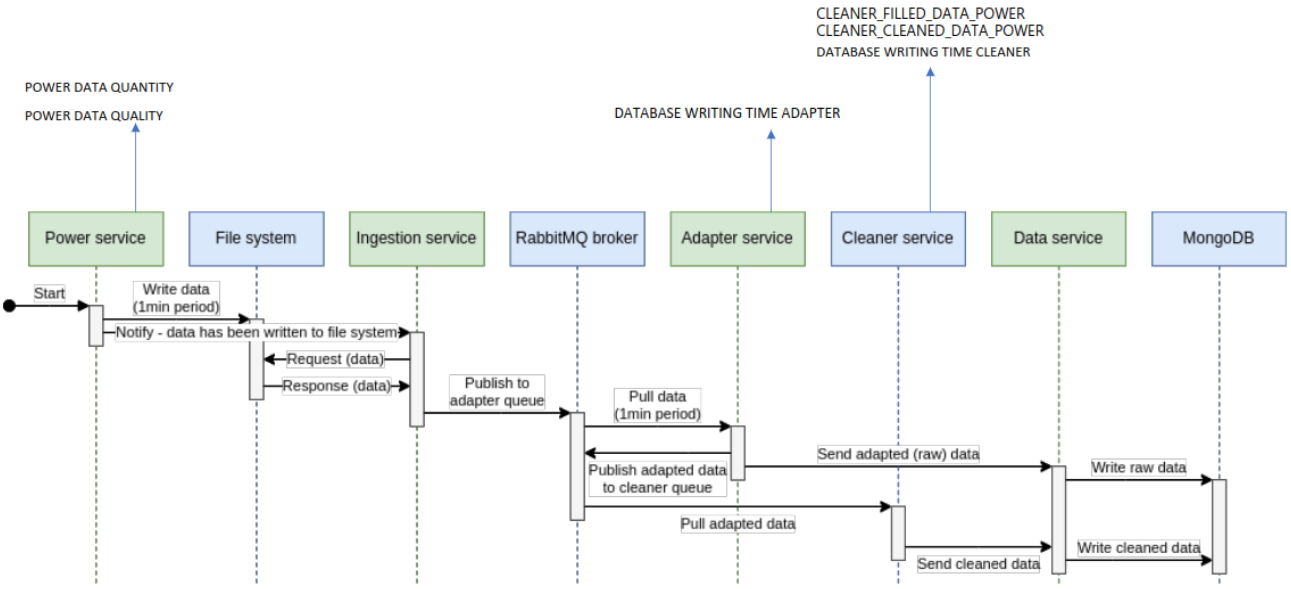



**Figure 8***:* Sequential diagram for Power sensor data API

### 6.2.2. Ingestion Service

Ingestion service is a component whose role is to obtain data from the EDGE file system and publish them to the Message Broker (RabbitMQ) broker queue.

This service is waiting for notification from various Sensor services. When it receives a notification, it means that the data has been written to the file system from which it can be read. When the Ingestion service reads the file, it publishes the contents of the file to the Adapter Queue that is listening to the Adapter service.

### 6.2.3. Adapter

The adapter is a component which is used to transform data from the data collection services to format used in D2Lab. The Adapter is the only component that "knows" how the raw data looks like, and how it should be mapped to our internally used format.

The Adapter listens for new raw data (adapter queue), parses it to the correct format and passes the result to Data service using its API for storage purposes. Also, the Adapter broadcasts needed information, to notify interested components (ex. Cleaner) that there is new raw data to be processed.

Adapter listens to *adapter* queue, takes the content from the queue, with several possible routing keys like “power”, “environment”, “vibrations”, “power_additional” etc. which concerns type of data that we are adapting and store. Further, the adapter adapts data to our internal format and stores it into DB using Data Service.

### 6.2.4. Cleaner

Adapter is equipped with the possibility to notify via *raw_product* routing key that new data has been received and processed. The adapter sends metadata to the broker, so that Cleaner can pick up that metadata and from that information pulls data from database collection “product_instances_raw”, and then continues the work of cleaning. Adapter is the last component which knows the original data format.

## 6.3. Server part

Monitoring data quality and data collection are essential for any industry. Poor data quality can lead to incorrect insights and bad decision-making, while inadequate data collection can result in gaps in the data that can lead to incorrect conclusions.

### 6.3.1. Monitoring and visualization layer

Graylog is a popular open-source log management platform that can be used to monitor data quality and data collection in industry. Here is a concept for using Graylog to monitor data quality and data collection in industry:

Collecting metadata (logs): The first step is to collect the metadata from various sources, such as machines and sensors. The data can be collected in real-time or at regular intervals, depending on the use case. We use logs as metadata. Our services send logs that contain all the necessary metadata that we use in our data observability system. (see the logs of the branch on the architecture of the data collecting system)

Storing metadata: Once the metadata is collected, it needs to be stored in a central location for analysis. Graylog provides a scalable and flexible platform for storing and indexing log data. Logs can be stored in Elasticsearch, which provides fast search and retrieval capabilities. Our logs are stored internally in the database (Graylog service saves them automatically). We use this data for various searches that we need for the functioning of the data observability system, as well as for calculating average window values, data visualization on Graylog, etc.

Analyzing data: The next step is to analyze the data to ensure its quality and completeness. Graylog provides powerful search and filtering capabilities to analyze the data. For example, metadata can be filtered based on a specific time range, source, or severity level. This allows users to quickly identify issues with the data. What they call "analyzing data" in Graylog in our perspective we have metadata searches that we use to calculate larger/smaller windows (window lengths - time, method - mean values, ...) based on the search, we alert - more description in the next item.

Alerting: Graylog can be configured to send alerts when specific events occur. For example, if a sensor stops sending data, an alert can be triggered, indicating a potential problem. This allows users to take proactive measures to address any issues before they impact operations. We also use an alerting system for

notifications on Teams when any of sensor services stop working, or to refresh time windows at certain periods, etc.

Visualization: Once the metadata is analyzed and any issues are identified, the next step is to visualize the data. Graylog provides a range of visualization tools, including dashboards, charts, and graphs, to help users understand the data. This allows users to quickly identify trends and patterns in the data, making it easier to make informed decisions. We use dashboards to visualize alarms and metadata.

In summary, using Graylog and our services to monitor data quality and data collection in industry involves collecting data from various sources, storing the data in a central location, analyzing the data for quality and completeness, alerting when issues occur, visualizing the data to identify trends and patterns, and generating reports to monitor data quality and data collection over time. By implementing this concept, organizations can ensure that they are collecting high-quality data that can be used to make informed decisions.

### 6.3.2. Data Observation layer

**Alarms**

Main goal of Data Observation Layer is to create alarms in the case of some unusual/problematic situations which happen in the data collection system. They are based on larger/smaller windows processing. We consider average signal values and calculate the difference between the larger and smaller windows in percentages. They are stored as those percentages as 'changes' in a special sequence (''Add result to standard deviation window'', cf. Figure Data Observation). Over that series of changes, we calculate the standard deviation, and if the current change is outside the frame of two standard deviations around the average value of that series, we generate an alarm.

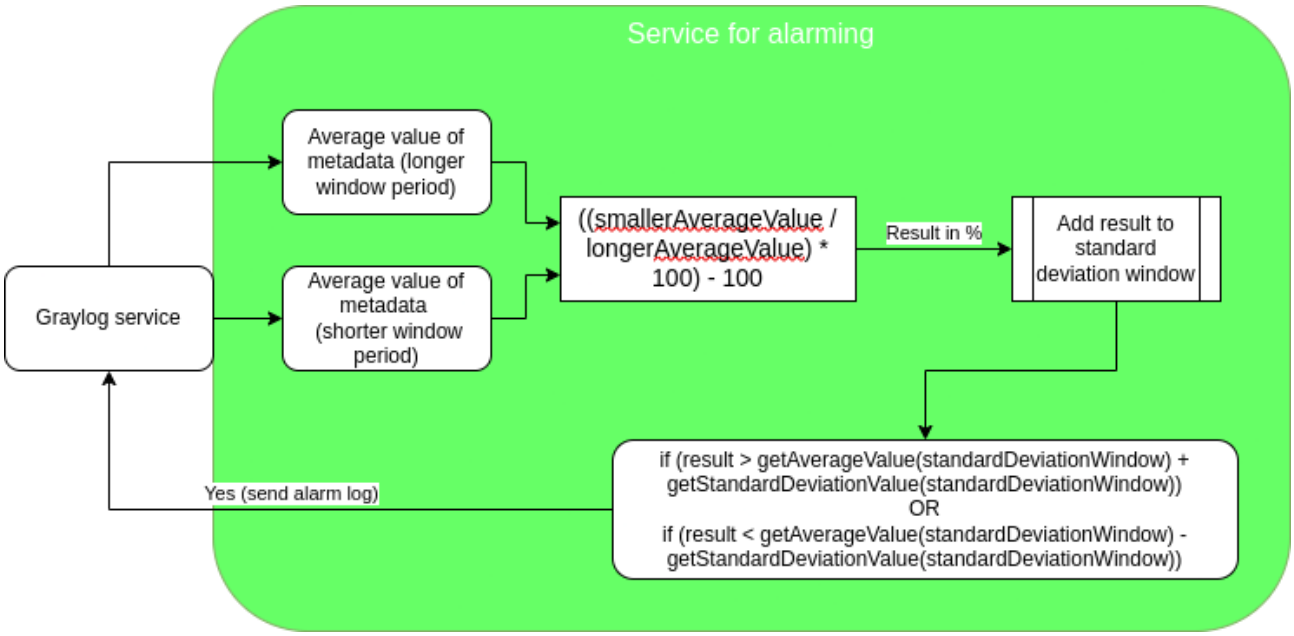



**Figure 9***:* Data Observation Alarms - calculation

- Average value of metadata (longer/shorter window period): This type of component represents the average value of a parameter in a certain period (example in our case - a 5-minute smaller window and a 14-day larger window)

- The formula used to calculate the difference between the two windows is as follows:
  **((smallerAverageValue/longerAverageValue) * 100) - 100**
  This formula determines the percentage of the difference between the two windows and stores it in this form (percentage) in the component described below.

- "Add result to standard deviation window" This is a component that represents a data structure that stores a certain number of percentage results. In our case, the size of this structure is 50. The last 50 results are saved and then overwritten (if all 50 places are filled, the 51st result/data will be written at the beginning of the sequence). Overwriting achieves constant monitoring of the signal state to act accordingly (alarms).

- The last part is generating the alarms. Therefore, when the result is calculated, it is compared with the previous results and if it is greater/less than N standard deviations around the average value it is an alarm (in our case, N=2 because for N=1, more alarms are generated that do not have to make sense in all cases, for two standard deviations there is a high probability that the alarms are really alarms).

**Implementation**

There are two implementations planned.

The current implementation (Graylog) consists of updating larger and smaller time windows for each monitored parameter and computing alarm in certain cycles. The windows are refreshed every two minutes, while the calculation of this type of alarm is done every 5 minutes (the size of the smaller window is 5 minutes, the size of the longer window is 14 days). These values are subject to change (we change them with a note that the same limits apply to all factories). The values are changed by creating an HTTP request (POSTMAN) and these values can be changed during the execution of the service - it is not necessary to shut down/restart the service. Changes are applied immediately.

There is another implementation using the Orion context broker. Analyzes are triggered every time the client refreshes an entity on the broker. Let us say client refreshes the smaller window entity every N minute - that means the analyses will be performed every N minutes. This is done because we want to adapt the analysis to individual clients. There is no need to run for all clients at the same time if some clients refresh more often and some less often.

**Reports**

There are three types of reports provided by the system

*Time shifts report analysis service*

This service is responsible for analysis and alarming of situations that are considered alarms. By default, analyzes are performed that include eight-hour shifts for which you receive email reports showing possible alarms that occurred during that period, these periods are subject to change.

*Report service*

This service is responsible for the reports themselves, accepts the data format from Graylog service (generated alarms), then transforms it into a human-readable format and sends an email. Mail report is one of the outputs of the Data Observation layer.

*Client report service*

Similar to the report service, the client report service also sends reports, but these reports are of a different nature (informative). These reports include basic information about the status of data collection, such as the average time of entry into the database or the average number of collected data points for data from sensors in a certain period (default 1 day).

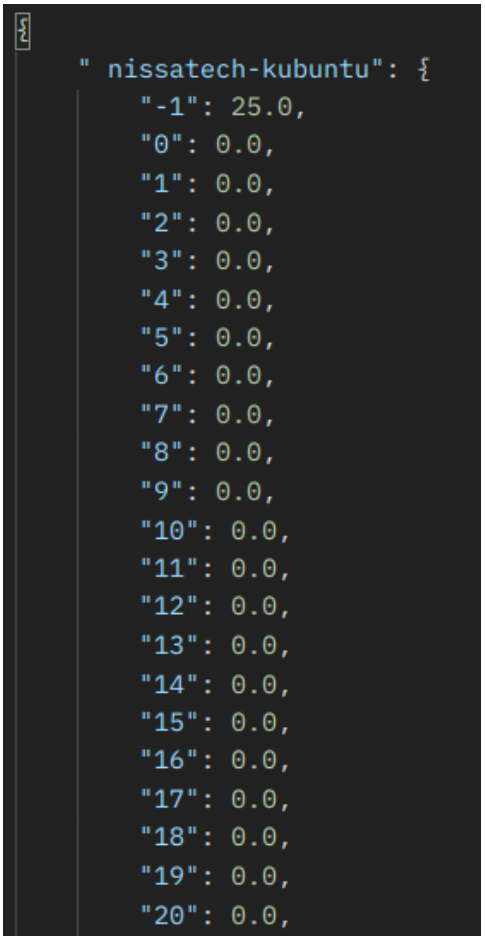
## 7. Data Observation: validation

In this section, we provide a detailed explanation and validation of the Data Observation part in the presented Edge-to-cloud Industry 5.0 Data Platform. We focus on this part since it is the most critical for a proper/accurate work of the Platform.

We illustrate the basic principles of Data Observation through a series of cases that show how the system reacts to certain situations that represent an unexpected state of the system.

### 7.1. Alarms - explanation

First, we started power service for collecting energy data (sending 60 - number of lost data points) and through time, standard deviation window (cf. Figure 9) is full (size of standard deviation window is 50, analysis is done by every 1 minute we track behavior in last 50 minutes of power service collecting system). The factory name is “nissatech-kubuntu” and the last 50 values in standard deviation window are zeroes (shorter and longer window values are same – state with no changes). The first element in window is the index where the current value will be written (element with index “-1”, not used in any calculations). This index goes from 0 to 49 and again (mod 50). This enables overwriting old values from window.



```
{
  " nissatech-kubuntu": {
    "-1": 25.0,
    "0": 0.0,
    "1": 0.0,
    "2": 0.0,
    "3": 0.0,
    "4": 0.0,
    "5": 0.0,
    "6": 0.0,
    "7": 0.0,
    "8": 0.0,
    "9": 0.0,
    "10": 0.0,
    "11": 0.0,
    "12": 0.0,
    "13": 0.0,
    "14": 0.0,
    "15": 0.0,
    "16": 0.0,
    "17": 0.0,
    "18": 0.0,
    "19": 0.0,
    "20": 0.0,
```


**Figure 10***:* Standard deviation window (stdDev)

When we have all zeroes in stdDev window, that means all is “normal” (no changes). Now we will change the number of data loss from 60 to 50 data points/min.

```
"18": 0.0,
"19": 0.0,
"20": 0.0,
"21": 0.0,
"22": 0.0,
"23": 0.0,
"24": 0.0,
"25": 0.0,
"26": 0.0,
"27": 0.0,
"28": 0.0,
"29": 0.0,
"30": 0.0,
"31": -3.916449086161876,
"32": -11.832061068702288,
"33": -15.632754342431767,
```


**Figure 11***:* Standard deviation window

We can see that in the last three indexes we notice negative values (in the power case, that means less loss). We also notice that we register alarm state in three iterations (31,32,33 indexes).

After "33" index, percentage value is starting to fall (if power service continues to send the same value (50), the system will detect this as normal state). Because of that, from index "33", values start to fall (going to be zero in some moment).


```
"30": 0.0,
"31": -3.916449086161876,
"32": -11.832061068702288,
"33": -15.632754342431767,
"34": -15.254237288135599,
"35": -14.893617021276597,
"36": -14.549653579676686,
"37": -14.221218961625283,
"38": -13.907284768211923,
"39": -13.60691144708423,
"40": -13.319238900634247,
"41": -13.043478260869563,
"42": -12.7789046653144,
"43": -12.524850894632195,
"44": -12.280701754385973,
"45": -12.045889101338432,
"46": -11.81988742964353
```


**Figure 12***:* Standard deviation window

But alarms are only values which exceeds standard deviation level (upper or lower), in this case, alarms are only the next values:

```
NISSATECH-KUBUNTU: Average power data loss: -3.916% than usual.
NISSATECH-KUBUNTU: Average power data loss: -11.832% than usual.
NISSATECH-KUBUNTU: Average power data loss: -15.633% than usual.
NISSATECH-KUBUNTU: Average power data loss: -15.254% than usual.
NISSATECH-KUBUNTU: Average power data loss: -14.894% than usual.
NISSATECH-KUBUNTU: Average power data loss: -14.550% than usual.
NISSATECH-KUBUNTU: Average power data loss: -14.221% than usual.
NISSATECH-KUBUNTU: Average power data loss: -13.907% than usual.
```

**Figure 13***:* Alarms

Or we can see when these alarms are triggered in graylog dashboard:

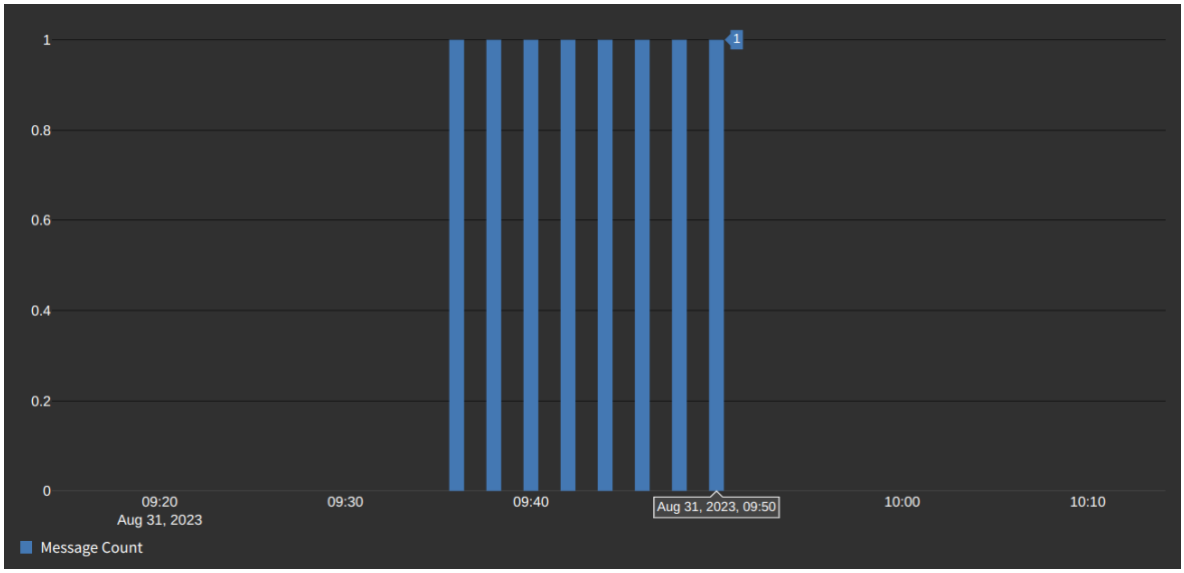



**Figure 14***:* Alarms on panel

Or through email report this change can be seen:

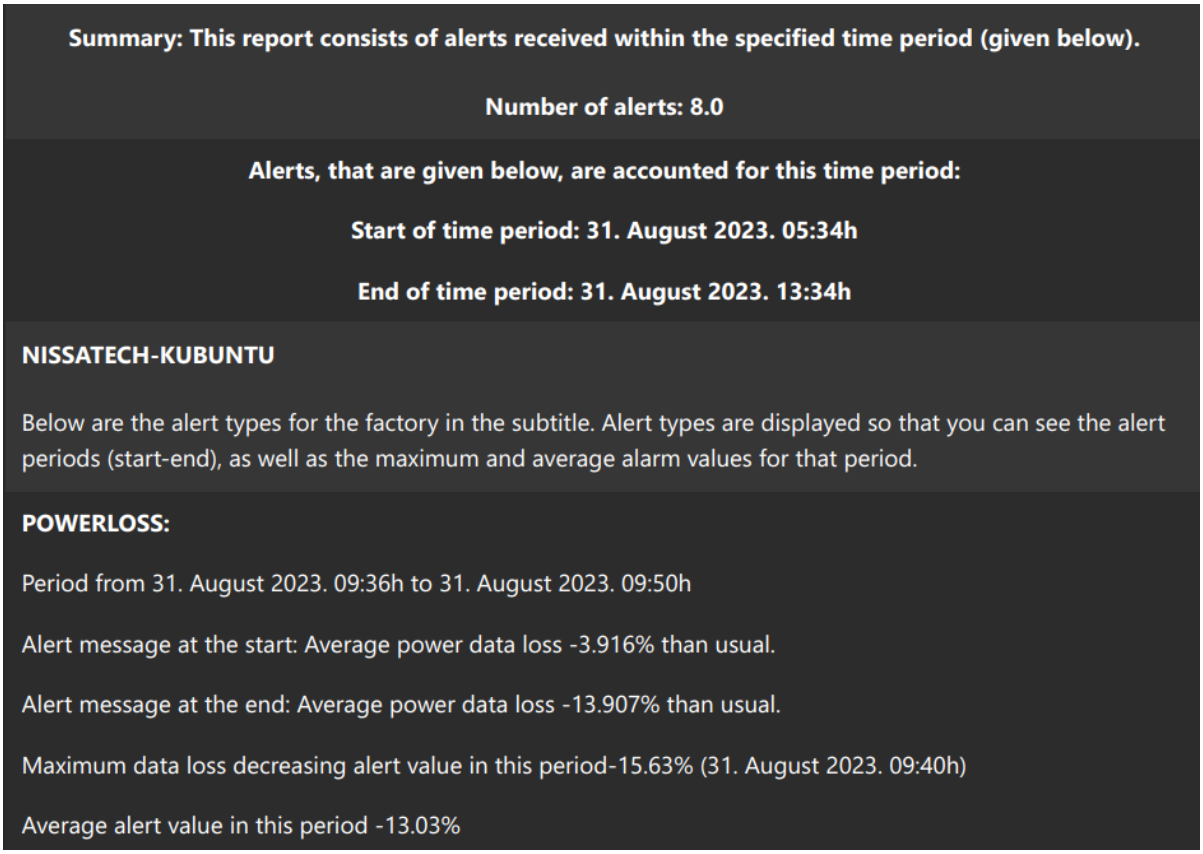



**Figure 15***:* Email report

Now we will change back from 50 to 60 data points loss. We can see that values starts growing in few iterations (+15% of data loss in “10” index)

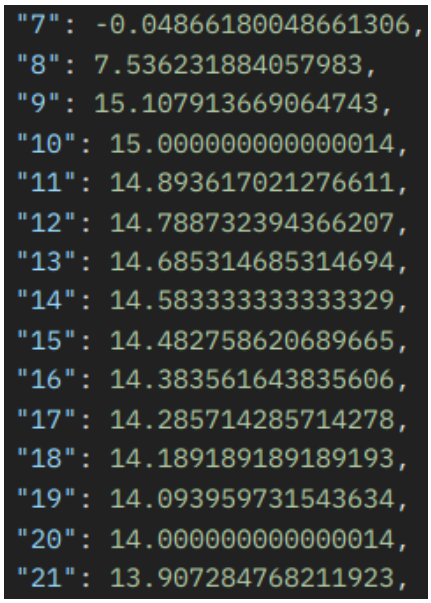



**Figure 16***:* Standard deviation window

Alarms are:

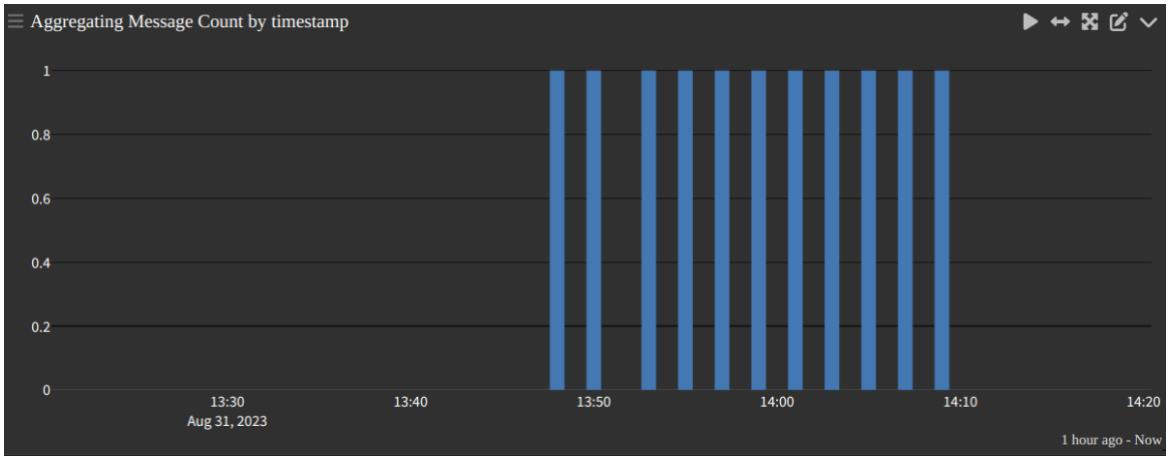



**Figure 17***:* Alarms on Graylog dashboard

Same alarms in code:

```
NISSATECH-KUBUNTU: Average power data loss: -0.049% than usual.
NISSATECH-KUBUNTU: Average power data loss: +7.536% than usual.
NISSATECH-KUBUNTU: Average power data loss: +15.108% than usual.
NISSATECH-KUBUNTU: Average power data loss: +15.000% than usual.
NISSATECH-KUBUNTU: Average power data loss: +14.894% than usual.
NISSATECH-KUBUNTU: Average power data loss: +14.789% than usual.
NISSATECH-KUBUNTU: Average power data loss: +14.685% than usual.
NISSATECH-KUBUNTU: Average power data loss: +14.583% than usual.
NISSATECH-KUBUNTU: Average power data loss: +14.483% than usual.
NISSATECH-KUBUNTU: Average power data loss: +14.384% than usual.
NISSATECH-KUBUNTU: Average power data loss: +14.286% than usual.
```

**Figure 18***:* Alarms

Email report (second period group - Period from 31. August 2023. 13:48h to 31. August 2023. 14:09h):

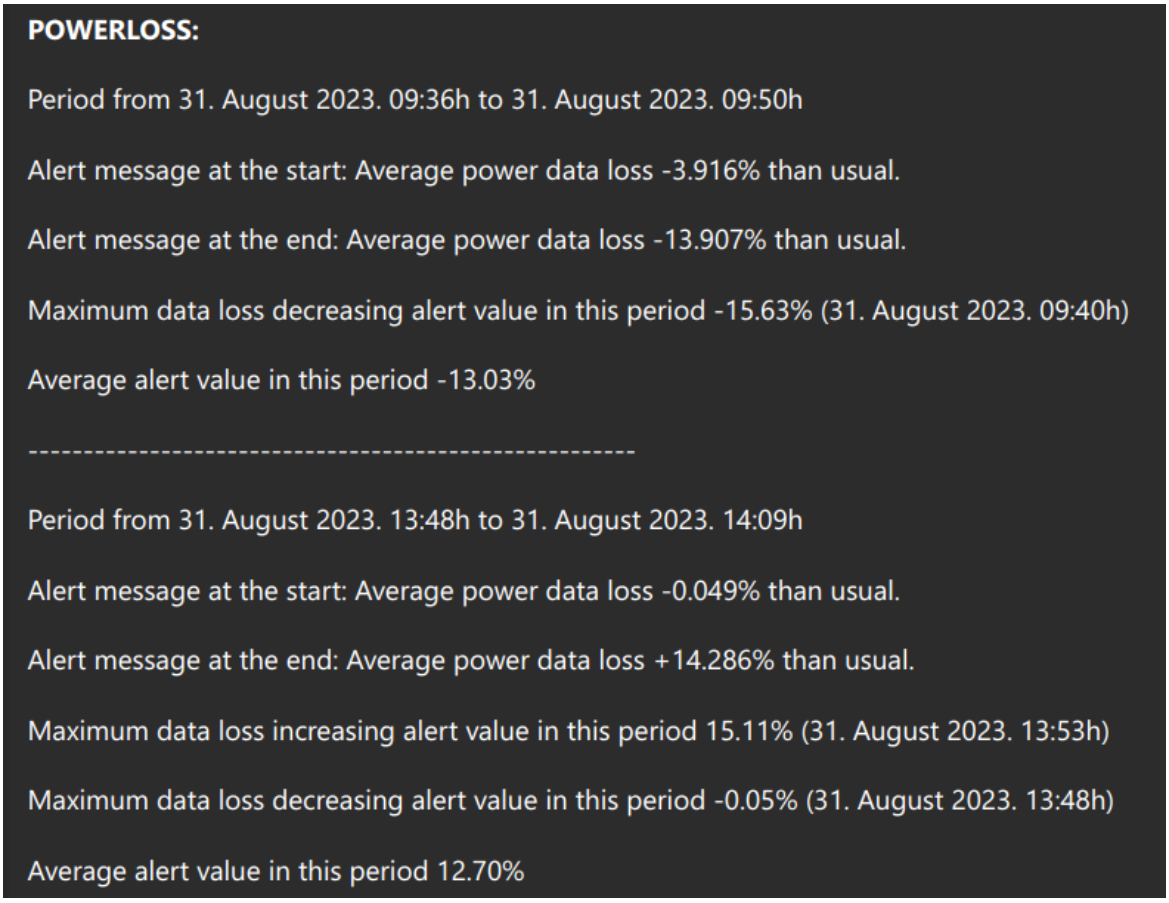



**Figure 19***:* Email report

The next use case would be stop sending data from power service (stop updating smaller window period). Now in few iterations, standard deviation window stops updating himself

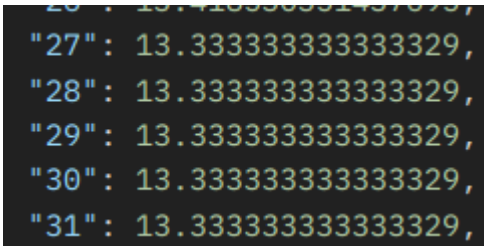



**Figure 20***:* Standard deviation value

Because shorter window value (average data loss in last 5 minute) is exactly 5 minutes, we will have five indexes with the same value in the standard deviation window (because of stopping sending data loss, last five minutes are the one (the same) average value). Then, after passing those five minutes, we start to get alarms that we do not get data for smaller time window.

```
NISSATECH-KUBUNTU: not getting data from smaller window period: Alert!
NISSATECH-KUBUNTU: not getting data from smaller window period: Alert!
NISSATECH-KUBUNTU: not getting data from smaller window period: Alert!
NISSATECH-KUBUNTU: not getting data from smaller window period: Alert!
NISSATECH-KUBUNTU: not getting data from smaller window period: Alert!
NISSATECH-KUBUNTU: not getting data from smaller window period: Alert!
NISSATECH-KUBUNTU: not getting data from smaller window period: Alert!
NISSATECH-KUBUNTU: not getting data from smaller window period: Alert!
```

**Figure 21***:* Alerts

Or in Graylog dashboard:

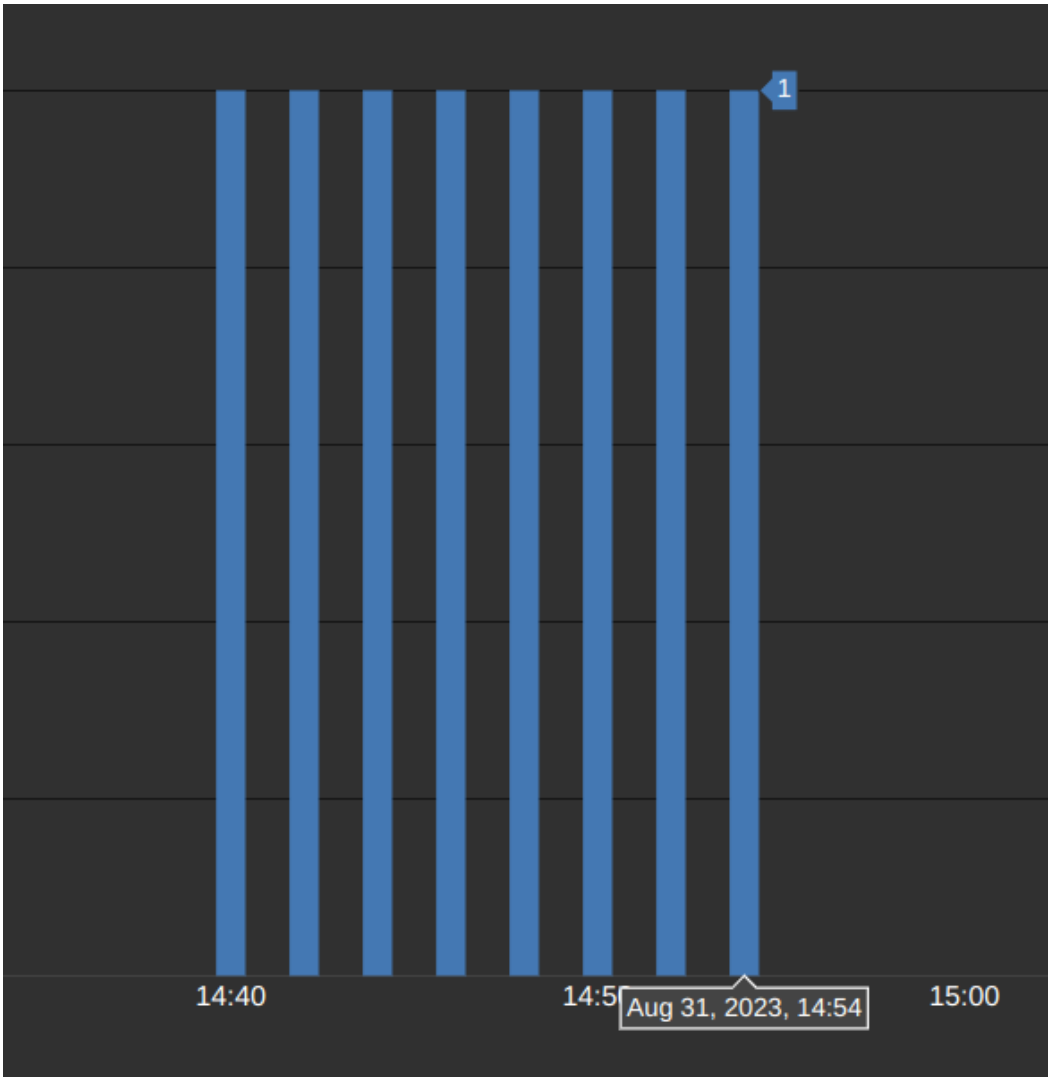



**Figure 22***:* Alarms in Graylog

Or in email report:

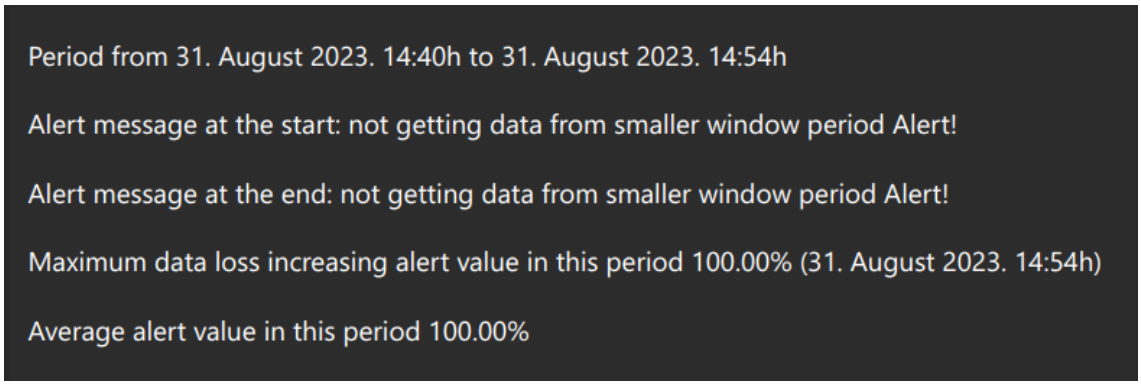



**Figure 23***:* Email report

## 7.2. Advanced alarming

In this section we present some additional alarm situations

1) The first case represents the case when data collection stops

The first case that we detect is the cessation of operation of the sensor service. Since the termination of these services does not necessarily mean the collapse of the system (shut down Raspberry Pi - unavailable) - it still stops data collection depending on the type of service that stops working. There are three types of data collection services (Energy, Environment and Vibration). The following figures will show examples of alarms in the Teams channel that indicate a change in the operation of a service and that it is necessary to check what happened:

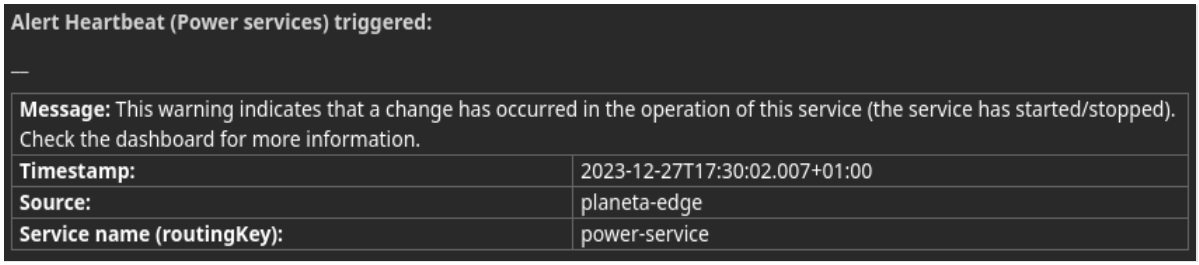



**Figure 24***:* Alarm for Energy data collecting service

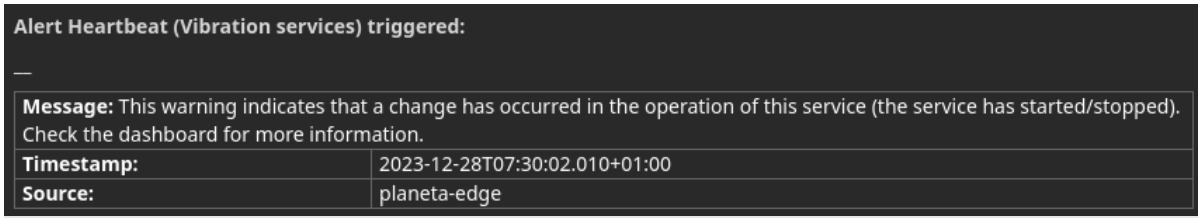
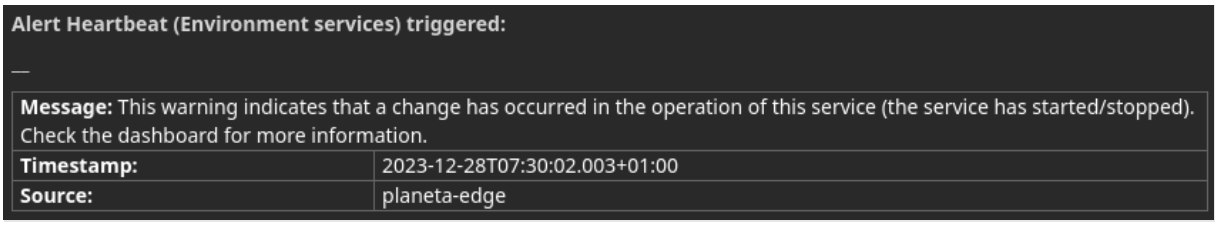



**Figure 25***:* Alarm for Vibration data collecting service



**Alert Heartbeat (Environment services) triggered:**

—

| | |
|---|---|
| **Message:** This warning indicates that a change has occurred in the operation of this service (the service has started/stopped). Check the dashboard for more information. | |
| **Timestamp:** | 2023-12-28T07:30:02.003+01:00 |
| **Source:** | planeta-edge |

**Figure 26***:* Alarm for Environment data collecting service

In such cases (a change in the operation of one of the services), the system detects the change in the worst-case one hour from the moment the change occurs.

After the alarm occurs, it is necessary to see in the dashboard what specifically happened for a certain service. By looking at the dashboard, we have complete information about what is happening with data collection for a specific service. An example of a dashboard for energy data will be shown below. Based on the time shown in the "Timestamp" field in the alarms from the images above, we look for the time in the dashboard:

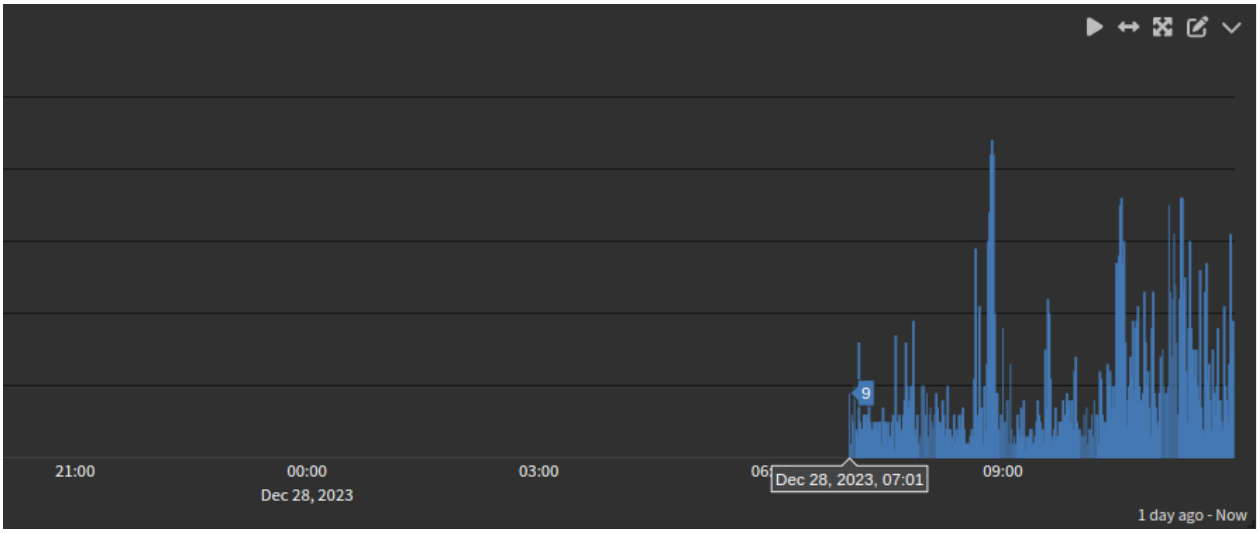



**Figure 27***:* Data Quantity Dashboard (Energy)

Since we said that alarms are generated at most one hour after the change occurs (here we have a case where the alarm occurred at 7:30 a.m.), it is therefore necessary to look at the period between 6:30 a.m. and 7:30 a.m. in the dashboard because that is when the change occurred. Here, the change is obvious, that is, the service obviously started working in that period.

Also, there is an alarm that indicates the cessation of operation of the entire device for data collection, which is an alarm for the cessation of operation of the edge device resource monitoring service (CPU

temperature, RAM, etc.). When such an alarm is generated, we consider the entire edge to have stopped working.

**Alert Heartbeat (Edge resource services) triggered:**

—

| | |
|---|---|
| **Message:** This warning indicates that a change has occurred in the operation of this service (the service has started/stopped). Check the dashboard for more information. | |
| **Timestamp:** | 2023-12-26T15:30:04.239+01:00 |
| **Source:** | pet-factory |

**Figure 28***:* Edge resource monitoring service alarm

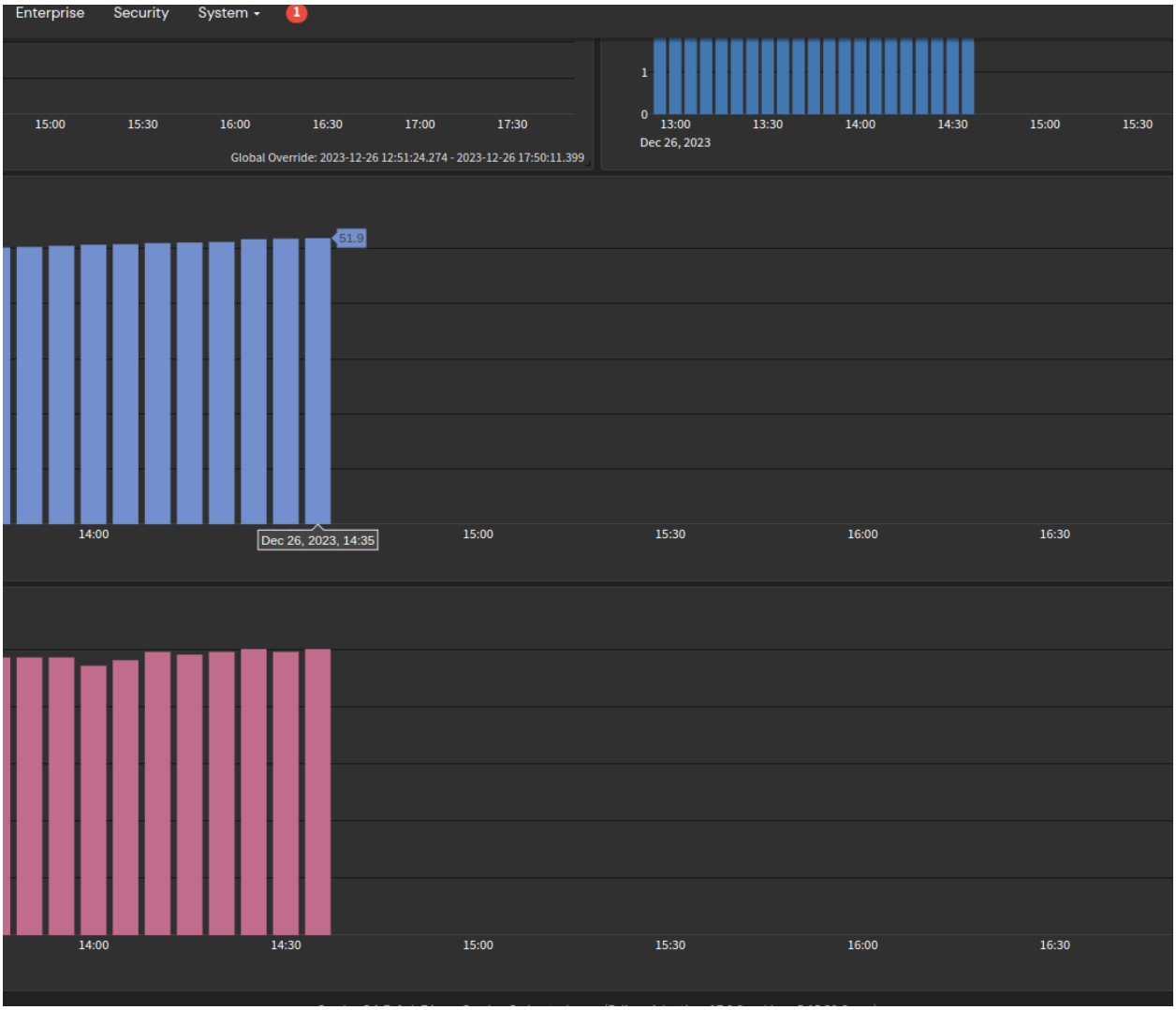



**Figure 29***:* Part of edge resource monitoring dashboard

## 7.3. An example of a real factory failure and its detection

In any case, the first indication that something has happened with the edge device comes from alarms related to edge resource monitoring (as explained in the previous part).

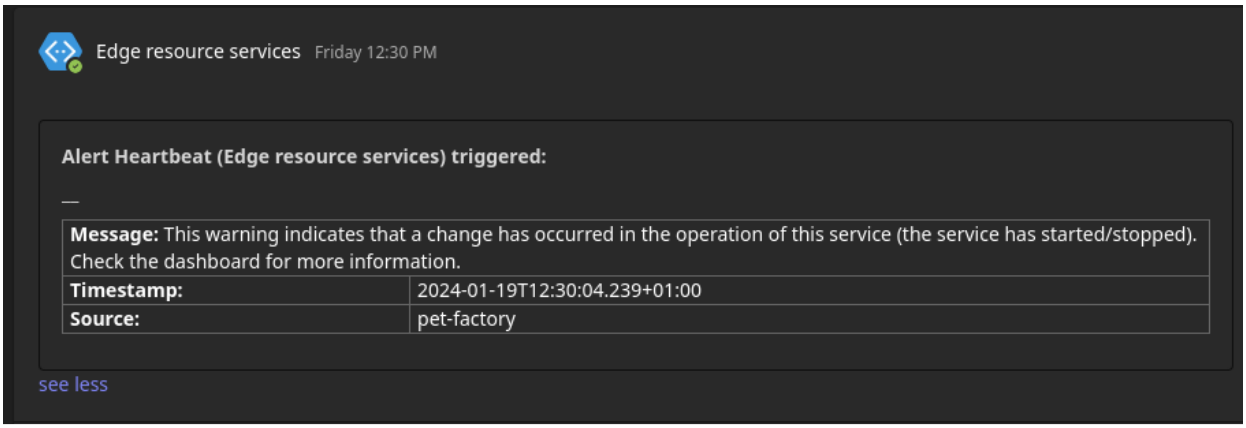



**Figure 30***:* Alarm

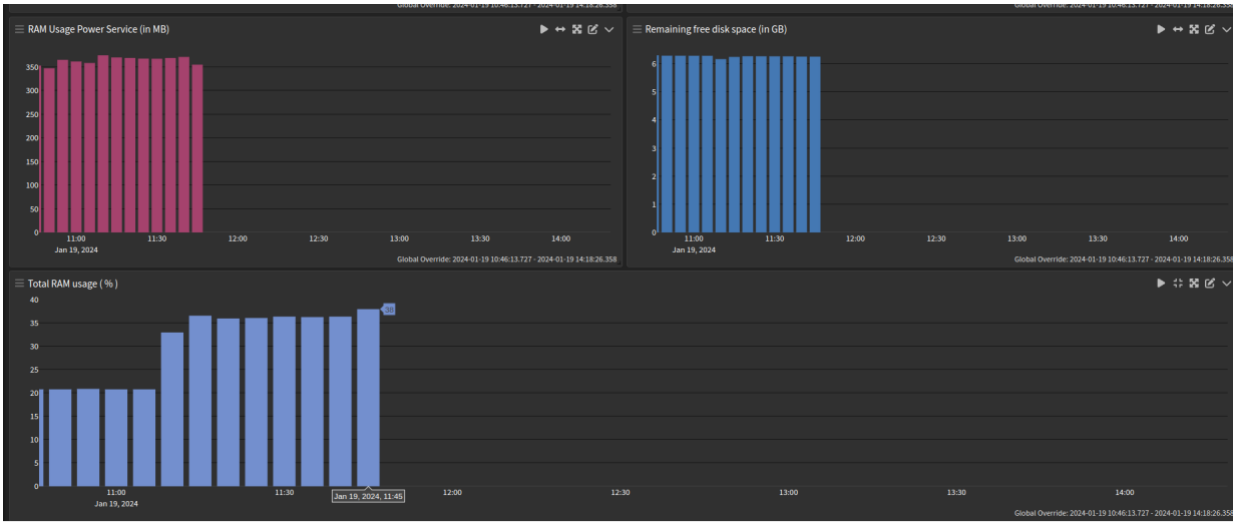



**Figure 31***:* Edge resource monitoring dashboard

Based on the alarm, we looked at the dashboard, where it was seen that the data had stopped arriving, which in this case means that the edge device has stopped working.

Also, the alarms through the time shift report soon detected the cessation of sending data from smaller time windows, which also gives a good sign that the alarm for edge resource monitoring is a good indicator of the cessation of edge device operation. In the picture below, only one of the parameters from the given factory is shown, but the termination can be observed in other parameters as well.

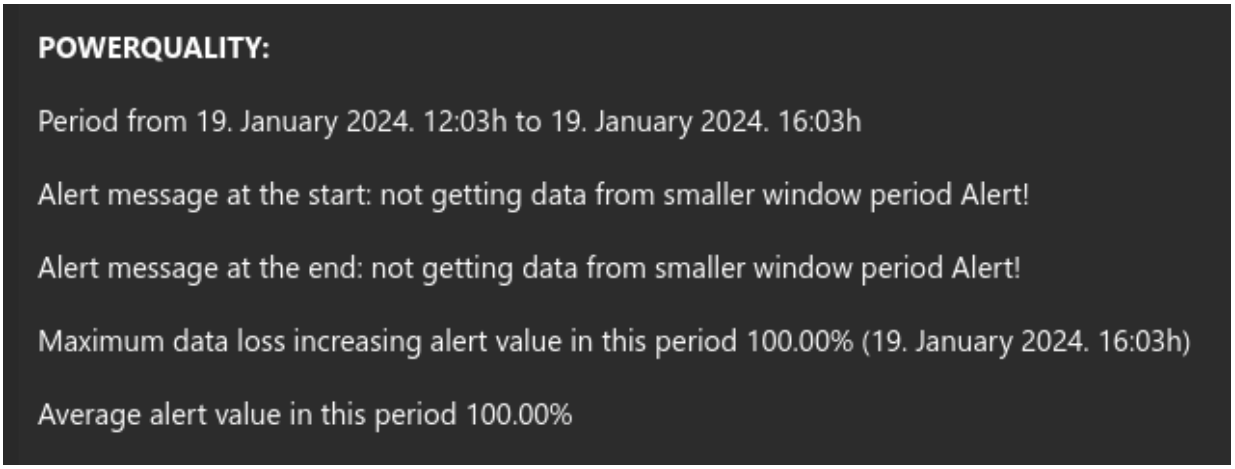



**Figure 32***:* Alarms in time Shift report

The following picture shows an example of the next report in order with the second parameter, where it can be seen that the data did not start arriving in that period either.

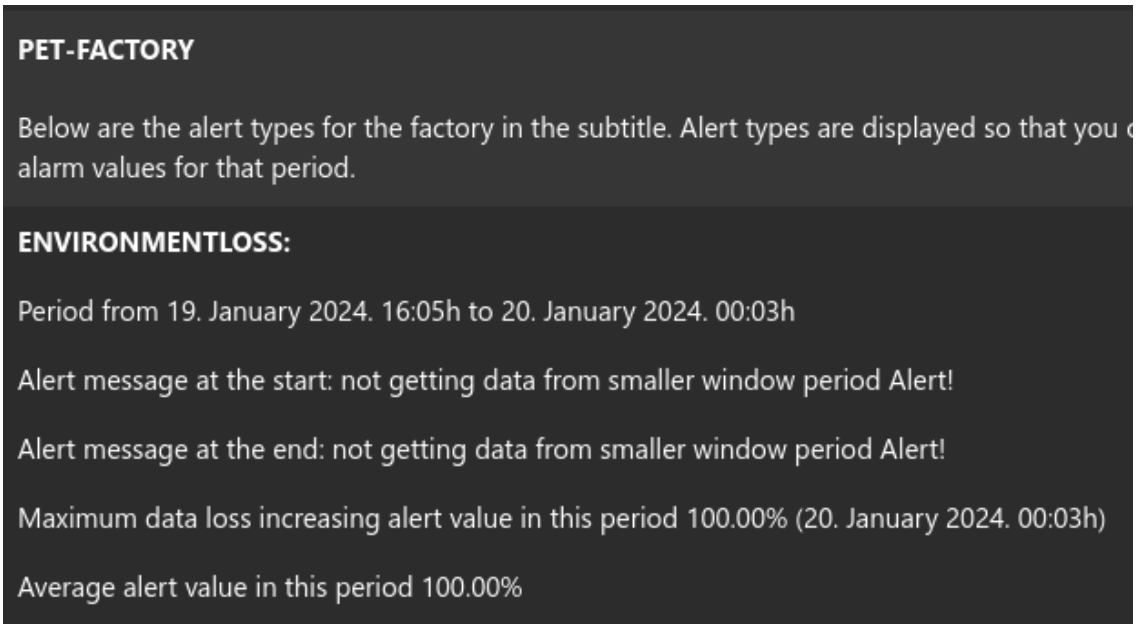



**Figure 33***:* Alarms in time shift report

Finally, in the next report we see that the period of non-receipt of data ended at 7:05 a.m. (the report is up to 8:05 a.m.), which means that the arrival of data began at that time.

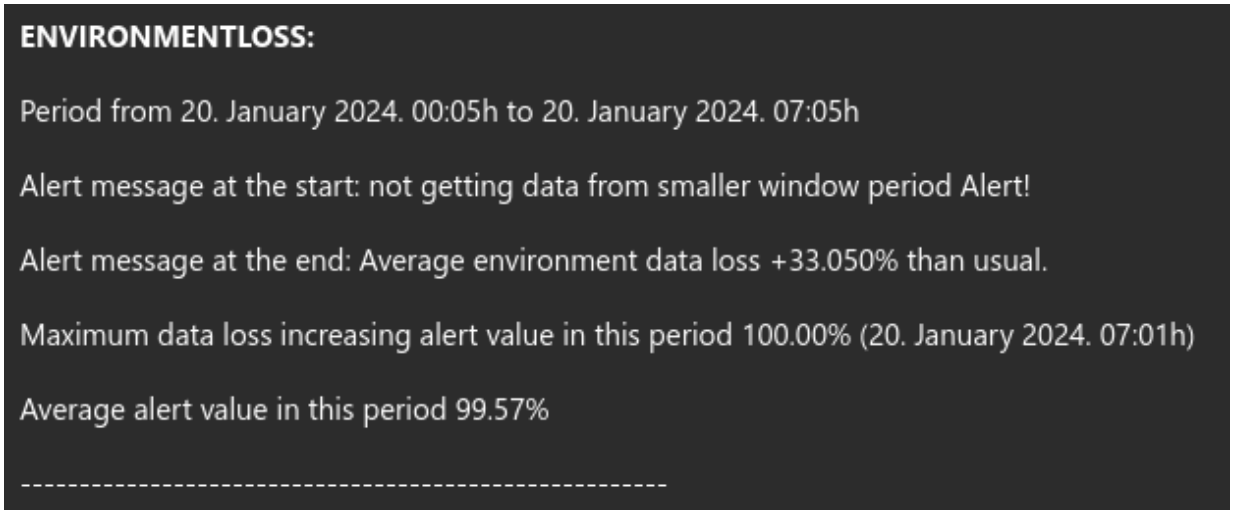



**Figure 34***:* Alarms in time shift report

Confirmation of this can also be seen as an alarm for a change in the operation of edge resource services

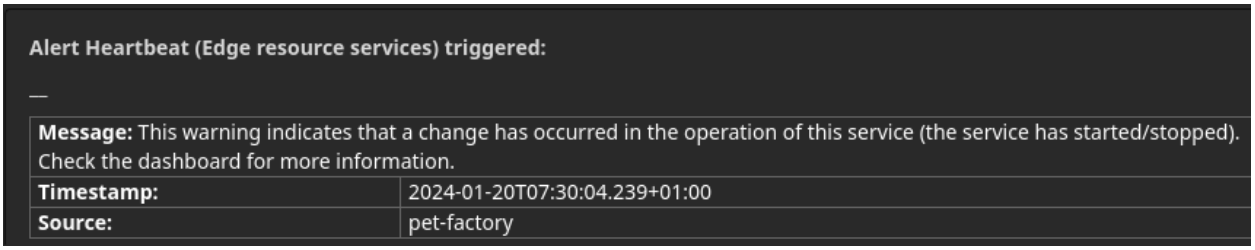



**Figure 35***:* Edge resource monitoring alarm

Dashboard view:

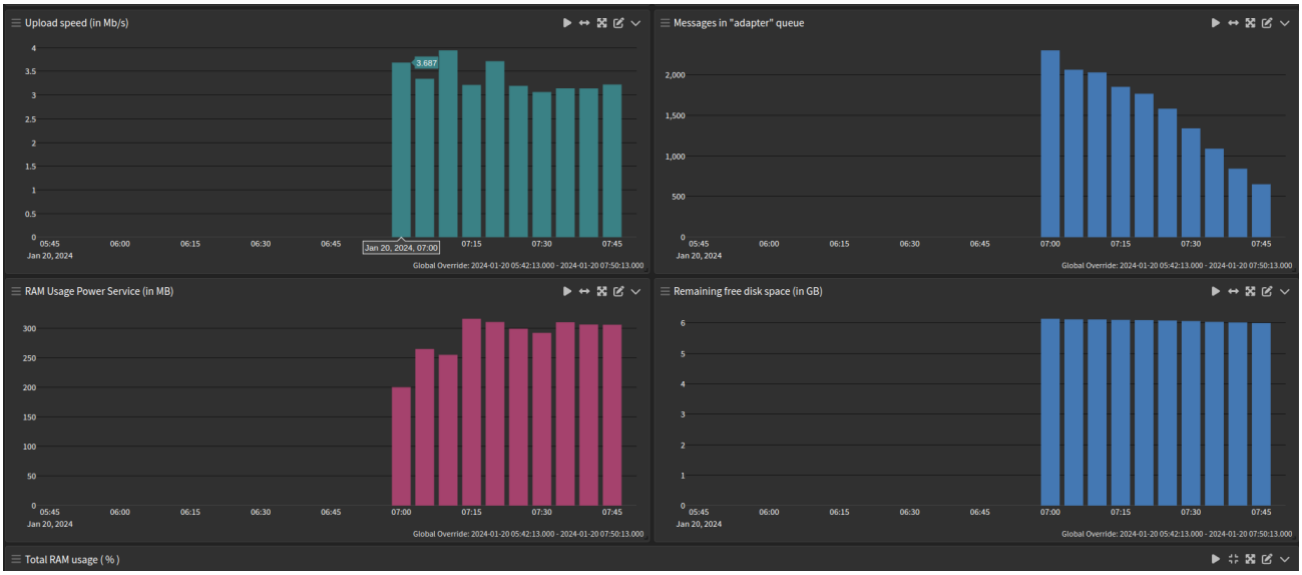



**Figure 36***:* Dashboard for the alarm situation presented in Figure 35

With this, we have shown that within 1 hour at the latest, we can detect the cessation of data arrival from the factory, then monitor how long the data does not arrive and finally see the start of the factory's operation.

## 7.4. An example for the data preprocessing alarms

This section shows more cases of alarms from factories and raw data states in the periods when alarms were generated. The first such example is the time of entry into the database for the factory we named Pet-factory in our system. This alarm belongs to the data preprocessing alarms

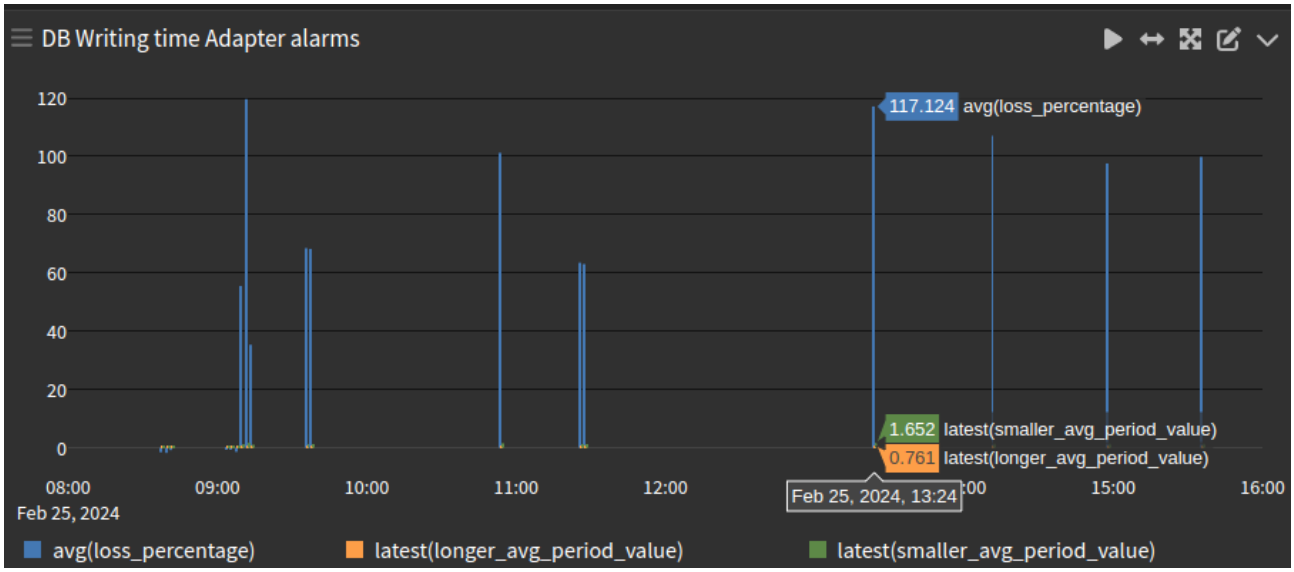



**Figure 37**: Alarms in dashboard

On the previous dashboard, we see an example of an alarm that occurred on February 25 at 1:24 p.m. And then the same alarm in the next picture, only in a different format - through an email report.

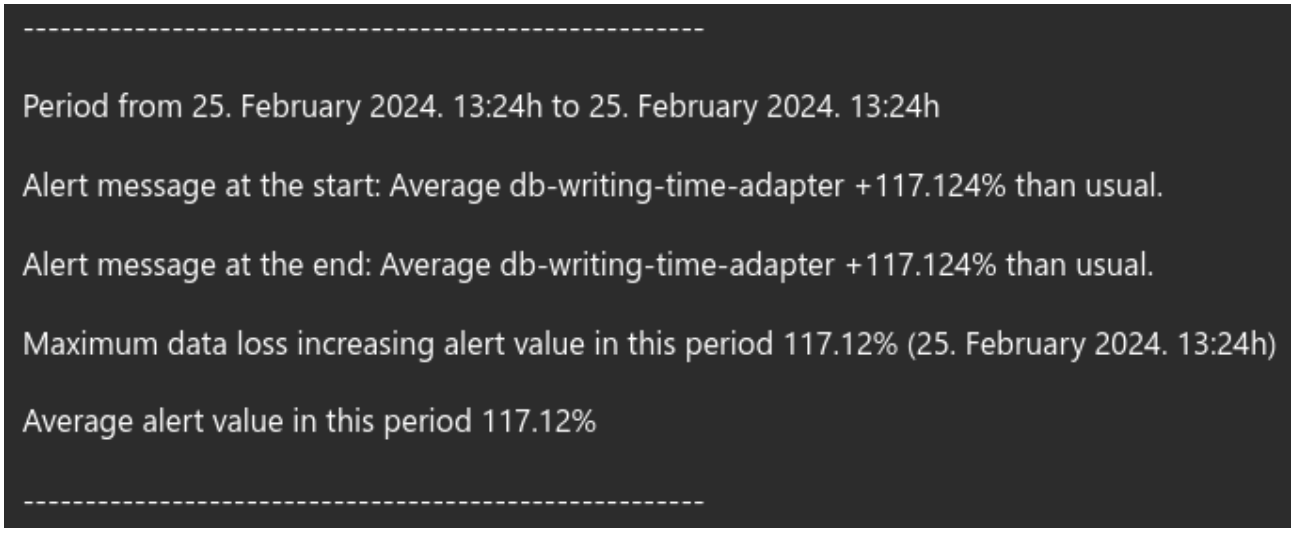



**Figure 38**: Alarm in email report

Here we conclude that in that period the mean value increased by 117,124%. This means that during that period the value of entry time in the database increased several times in a row. To clarify - if the database write time was 1 second every minute - this means that in the few minutes the write time slowed down and

caused a big change in the average value of the write time, which can be seen in the raw data image given below.

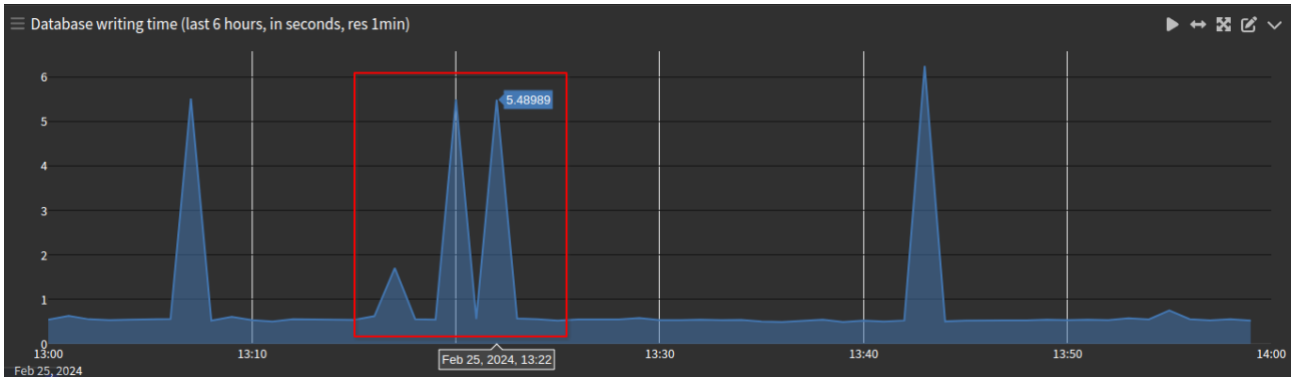



**Figure 39**: Raw data in the selected period

## 8. Conclusions

This deliverable is the result of the Task 4.2 “T4.2 Industry 5.0 Data Pipelines and Data Quality Assurance”, which main goal is creating data pipelines and ensuring data quality for the planned AI tasks.

The deliverable reports on the software development of the Edge-to-cloud Industry 5.0 Data Platform.

It addresses the requirements for the Edge-to-cloud Industry 5.0 Data Platform and explains the architecture and its details.

In addition, this deliverable provides details about the Data Observation layer which is responsible for ensuring the quality of the data collection process.

This deliverable contributes to the next iteration of Reference Architecture.

The Edge-to-cloud Industry 5.0 Data Platform will be validated in the context of WP6 during the implementation of the use cases.